Fachbereich Psychologie

Arbeitsgestaltung in der Softwareentwicklung: Ein empirischer Vergleich subjektiver Arbeitsmerkmale in proprietären und Open-Source-Softwareprojekten

Inaugural-Dissertation
zur Erlangung des Doktorgrades der

Philosophischen Fakultät
der
Westfälischen Wilhelms-Universität zu Münster (Westf.)

Vorgelegt von

Dipl. Psych. Dirk Jendroska
aus Wiesbaden

2010

Dekan:	Prof. Dr. Christian Pietsch
Referent:	Prof. Dr. Guido Hertel
Korreferent:	Prof. Dr. Bernad Batinic

Tag der mündlichen Prüfung: 11.08.2010

Für Isabel.

Ich möchte mich bei meinem Betreuer Prof. Dr. Guido Hertel für seine Unterstützung und vielen guten Ratschlägen bei der Erstellung dieser Arbeit bedanken. Mein besonderer Dank gilt Dr. Tanja Bipp für ihre Unterstützung bei aufgetretenen Herausforderungen. Ohne meine Frau, meiner Familie, der Familie meiner Frau und meinen Freunden wäre diese Arbeit nicht zu Stande gekommen.

"Sharks with lasers on their heads" Linus Torvalds, 2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In den vergangenen 20 Jahren hat sich neben kommerziell produzierter und vertriebener Software zunehmend eine alternative Form verbreitet: Open-Source-Software. Als Open-Source-Software werden solche Programme bezeichnet, die unter einer freien Lizenz veröffentlicht werden und damit unbeschränkt genutzt, kopiert und verändert werden können. Dabei unterscheidet Open-Source-Software nicht nur das Lizenzmodell, sondern auch die Art der Softwareentwicklung. Das Open-Source-Entwicklungsmodell zeichnet sich insbesondere durch Freiwilligkeit (Raymond, 1999b) und selbstständige Koordination (Adam, Feller & Fitzgerald, 2003; Crowston, Annabi, Howison & Masango, 2004; Garzarelli, 2002; Koch, 2004; Madey, Freeh & Tynan, 2002) über das Internet aus (Dafermos, 2001; Demil & Lecocq, 2003). Viele Entwicklungsprojekte werden außerhalb von Unternehmen und ohne finanzielle Unterstützung durchgeführt und weisen damit nicht die betrieblichen Strukturen auf, die sonst für kommerziell erstellte Software selbstverständlich sind (Bollinger, Nelson, Self & Turnbull, 1999; Koch, 2004). Diese zeichnet sich durch eine ausgeprägte Modularität, häufigere Veröffentlichungen neuer Versionen und engere Abstimmung mit den NutzerInnen aus (Raymond, 1999a). Viele Projekte basieren nicht auf einer hierarchischen Organisation, sondern sind wie ein "Marktplatz" organisiert (Raymond, 1999a). Die Ergebnisse von Open-Source-Projekten sind in Bezug auf Qualität, Sicherheit und Entwicklungszeit konkurrenzfähig, teilweise sogar überlegen (David, Waterman & Arora, 2003; Ghosh, Glott, Krieger & Robles, 2002; Mockus, Fielding & Herbsleb, 2000, 2002).

Nicht nur die Softwareindustrie ist auf diese neue Art der Softwareentwicklung aufmerksam geworden, sondern auch die Forschung. Dabei bietet das Open-Source-Entwicklungsmodell für verschiedene Fachbereiche interessante Fragestellungen. Neben Juristen (z. B. Fink, 2002), die sich mit den Fragen der Lizenz auseinandersetzen und Ökonomen (z. B. Lerner & Tirole, 2001), die z. B. mögliche Geschäftsmodelle betrachten, und Wissenschaftlern im Bereich Organisationstheorie (z. B. Ljungberg, 2000) befasst sich auch die sozialwissenschaftliche Forschung mit Open-Source. Dabei standen bisher überwiegend Fragestellungen zu Motiven und Motivation von Personen im Vordergrund, die ihre Arbeitskraft z. T. kostenlos für Open-Source-Projekte zur Verfügung stellen (Ghosh et al., 2002; Hars & Ou, 2002; Hertel, Niedner & Herrmann, 2003; Jorgensen, 2001; Lakhani & Wolf, 2003; Osterloh, Rota & Kuster, in Druck). Besonders häufig wurde intrinsische Motivation als Erklärung des Verhaltens von Open-Source-Entwicklern[1] herangezogen (z. B. Franck & Jungwirth, 2002a; Hertel et al., 2003; Luthiger, 2006; Osterloh, Rota & Kuster, in Druck). Ein wichtiger Einflussfaktor der intrinsischen

[1] Auch wenn aus Gründen der Lesbarkeit im Text die männliche Form gewählt wurde beziehen sich die Aussagen auf Angehörige beider Geschlechter.

Motivation, u. U. sogar der wichtigste, wurde bisher noch nicht systematisch untersucht: Charakteristika der Tätigkeit selbst (Hackman & Oldham, 1976).

Eine Vielzahl von Untersuchungen konnte den Einfluss der Tätigkeitsgestaltung auf die intrinsische Motivation zeigen (zusammenfassend Fried, 1991; Loher, Noe, Moeller & Fitzgerald, 1985). Jedoch sind dem Autor keine Veröffentlichungen bekannt, die die Tätigkeit in Open-Source-Projekten systematisch untersuchen. Dabei existieren viele Aspekte in Open-Source-Projekten, die sich von der "traditionellen" (proprietären) Softwareentwicklung unterscheiden (Bollinger et al., 1999; Koch, 2004). Für eine systematische Untersuchung der Tätigkeit in Open-Source-Projekten bieten sich die Werkzeuge und Methoden der Arbeitspsychologie an. Insbesondere im Bereich der Tätigkeitsanalysen existieren ausgereifte Instrumente und Theorien, die bereits erfolgreich zur Untersuchung von Tätigkeiten in der Softwareentwicklung eingesetzt wurden (Brodbeck & Frese, 1994; Couger & Zawacki, 1980; Goldstein, 1989; McKnight & Chervany, 1998).

Im Folgenden soll zunächst das Entwicklungsmodell von Open-Source-Software beschrieben werden (Kapitel 2) und dann mit den verbreiteten Modellen der kommerziellen bzw. proprietären Softwareentwicklung verglichen werden (Kapitel 3). Bei der Beschreibung des Open-Source-Entwicklungsmodells soll insbesondere herausgearbeitet werden, welche Tätigkeitsmerkmale besonders stark beeinflusst werden. Ausgehend von einem Vergleich proprietärer und Open-Source-Softwareentwicklung sollen in Kapitel 4 die zuvor identifizierten relevanten Merkmale der Tätigkeit in der Softwareentwicklung beschrieben und eingeordnet werden. In diesem Rahmen werden auch Hypothesen über die Tätigkeitsgestaltung in Open-Source-Projekten abgeleitet. Kapitel 5 beschreibt die eingesetzten Methoden für den empirischen Vergleich der beiden Softwareentwicklungsmodelle. In Kapitel 6 werden die Ergebnisse einer Vergleichsstudie präsentiert und anschließend wird die Bedeutung für die Arbeitsgestaltung in der Softwareentwicklung diskutiert (Kapitel 7).

2 Open-Source-Software

Im Folgenden soll Open-Source-Software definiert werden und ein kurzer Überblick über die Entstehung von Open-Source sowie die damit assoziierte Frage der Eigentumsrechte und der Nutzung von Open-Source in Unternehmen gegeben werden.

Wenn Programme auf einem Computer zur Ausführung kommen, müssen diese in einer Form vorliegen, die für den Computer interpretierbar ist. Diese Form wird als Binärcode bezeichnet. Der Binärcode kann zwar von Computern ausgeführt werden, ist für Menschen jedoch schwer oder gar nicht verständlich. Im Prozess der Entwicklung von Software wird der Binärcode aus dem sogenannten Quellcode erzeugt. Dieser wird von Menschen geschrieben und kann auch von ihnen interpretiert und verändert werden. Als Closed-Source-Software wird solche Software bezeichnet, die ohne Quellcode (engl. "source code") vertrieben wird. Ohne Zugriff auf den Quellcode, sind Anwender darauf beschränkt, Software in der vom Hersteller vorgesehenen Weise zu nutzen.

Im Gegensatz dazu zeichnet sich Open-Source-Software durch die freie Verfügbarkeit des Quellcodes aus, der für jeden Interessenten einsehbar ist. Die freie Verfügbarkeit des Quelltextes ist die zentrale, jedoch nicht die einzige Voraussetzung, die Open-Source-Software erfüllen muss. Gemäß der Definition der OSI (Open-Source Initiative, 2006) darf die Lizenz von Open-Source-Software keine Einschränkungen bzgl. der Weitergabe enthalten. Jeder soll die Gelegenheit haben, auch in der Kombination mit anderen Softwarebestandteilen, Open-Source-Software zu verschenken oder zu verkaufen. Es besteht allerdings eine Reihe weiterer Forderungen an die Lizenz von Open-Source-Software: so muss diese es zulassen, dass die Software verändert werden darf und die Ergebnisse der Modifikation unter denselben Lizenzbedingungen vertrieben werden dürfen wie die ursprüngliche Software. Die Lizenz darf keinerlei Einschränkungen bzgl. des Einsatzes durch spezielle Personen oder Gruppen beinhalten (z. B. darf nicht festgelegt werden, dass eine Anwendung im kommerziellen Umfeld nicht erlaubt ist). Auch darf das Einsatzgebiet nicht vorgegeben werden (z. B. ist eine Beschränkung auf den nichtmilitärischen Bereich unzulässig). Die Lizenz soll dabei auf jeden übergehen, der die Software erhält, ohne dass er die Lizenz käuflich erwerben muss. Open-Source-Software darf dabei weder an ein spezielles Softwarepaket gebunden sein, noch darf die Weitergabe in Kombination mit anderer Software (z. B. proprietärer Software) eingeschränkt werden.

Neben Open-Source-Software existiert noch ein weiteres Lizenzmodell, das sich als Gegenmodell zu Closed-Source-Software versteht: Free-Software. Weitgehend identisch mit der Definition von Open-Source-Software sind die Anforderungen der Free Software Foundation (FSF) an "Free-Software" (Stallman, 1999). Sowohl "freie Software" als auch Open-Source fordern die freie Verfügbarkeit von Quelltexten von Software. Die grund-

legenden Übereinstimmungen führen dazu, dass Lizenzen sowohl den Anforderungen der Free Software Foundation als auch der Open-Source Initiative genügen können.

Der wichtigste Unterschied liegt jedoch in der moralischen/politischen Grundüberzeugung beider Modelle. So betont die Free Software Foundation das "frei" ("free") in ihrer Definition. So gilt eine Software als frei, wenn die Lizenz die Möglichkeit einräumt, ein Programm zu jeden Zweck auszuführen, zu studieren und zu verändern, zu verbreiten sowie zu verbessern und zu verbreiten (*The Free Software Definition*, 2009). "Frei" ist dabei im Sinne von Freiheit und nicht im Sinne von kostenlos gemeint. Aus moralischer Überzeugung wird Software abgelehnt, deren Lizenzmodell Einschränkungen dieser Freiheiten beinhaltet. Im Gegensatz dazu liegt die Betonung bei Open-Source-Software auf dem damit verbundenen Entwicklungsmodell (Raymond, 2000). Open-Source-Software soll zu besseren und kostengünstigeren Programmen führen.

Im Rahmen einer Revision der für die Free Software Foundation maßgeblichen Lizenz GPL (GNU General Public License) zeigten sich die grundsätzlichen Unterschiede zwischen der Open-Source- und der Free-Software-Bewegung. In einer Debatte zwischen einer Führungspersönlichkeit der Free-Software-Bewegung (Richard Stallman) und Linus Torvalds, dem Initiator des Linux-Projekts und einem bedeutenden Vertreter der Open-Source-Bewegung, lehnte Linus Torvalds einen Entwurf der Version 3 der GPL-Lizenz ab (Lyons, 2006). Als Grund für die Ablehnung gab Torvalds zum einen seine Zufriedenheit mit der bestehenden Version 2 an. Diese reicht aus seiner Sicht dazu aus, den für ihn maßgeblichen Punkt zu erfüllen: "Ich lege meinen Quelltext offen und möchte, dass alle Nutzer dasselbe machen." ("I give out code, I want you to do the same." Lyons, 2006). Insbesondere kritisierte Torvalds eine Passage in der neuen Version 3 der GPL-Lizenz, die den Gebrauch der Software einschränkte. So sollte nach dem Entwurf der Lizenz es nicht zulässig sein, Software für Digital Rights Management (DRM) zu nutzen. Eine solche Einschränkung, die stark politisch motiviert ist, wird von Torvalds und anderen Vertretern im Open-Source-Umfeld abgelehnt. Zur historischen Entwicklung von Open-Source- und Free-Software vergleiche auch Kapitel 2.1.

Um die Gemeinsamkeiten der beiden Ansätze zu betonen, verwenden eine Reihe von Autoren den Begriff "Free and Open-Source Software" bzw. "Free/Libre/Open-Source Software" in ihren Arbeiten (z. B. Crowston & Howison, 2004; z. B. David et al., 2003; Ghosh, 2002; Krishnamurthy, 2005; Rossi, 2004). Hierzu werden auch Abkürzungen wie "F/OSS" oder "FLOSS" genutzt. Aus Gründen der besseren Leserbarkeit soll in dieser Arbeit nur der Begriff "Open-Source-Software" verwendet, wobei Software mit vergleichbaren Lizenz- und Entwicklungsmodellen eingeschlossen wird.

Das Gegenstück von Open-Source-Software soll in dieser Arbeit als "proprietäre Software" bezeichnet werden. Dabei soll proprietär nicht im juristischen Sinne als "urheberrechtlich geschützt" verstanden werden, da auch der Schutz von Open-Source-Software auf einer, die Urheberrechte schützenden Lizenz beruht. Die Open-Source-Lizenz unterscheidet sich von proprietären Softwarelizenzen durch die großen Freiheiten, die diese dem Nutzer/der Nutzerin einräumt. Eine andere gebräuchliche Bezeichnung für Nicht-Open-Source-Software ist "Closed-Source-Software".

Dass der Quellcode für jeden Interessierten einsehbar ist und Software ohne Einschränkungen weitergegeben werden kann, ist kein neues Phänomen, wie der kurze Abriss der historischen Entwicklung im folgenden Kapitel zeigt.

2.1 Historische Entwicklung von Open-Source-Software

Während der Anfänge einer intensiveren Nutzung von Computern (ca. 1950-1970) waren es hauptsächlich Wissenschaftler und Forschungsgruppen in Unternehmen, die Software erstellten. Meist dienten diese Programme zur Lösung spezifischer Probleme und wurden nicht als Massenware betrachtet (History of free software, 2009). Auch wenn der Anteil an kommerziell vertriebener Software langsam zunahm, gab es in den siebziger Jahren noch Unternehmen (z. B. IBM), die die Software für ihre Großcomputer kostenlos zur Verfügung stellten (O'Reilly, 1999). Dabei war es auch nicht ungewöhnlich, wenn NutzerInnen Software neu- oder weiterentwickelten, untereinander tauschten und diese auch wieder den Computerherstellern zur Verfügung stellten.

Ein wichtiges Ereignis für die Open-Source-Geschichte stellte die Entwicklung von Unix durch Mitarbeiter der AT&T Bell Labs im Jahr 1969 dar. Durch staatliche Gesetzgebung war AT&T damals an einer kommerziellen Nutzung des Produktes gehindert und stellte es aus diesem Grund Universitäten einschließlich des Quellcodes kostenlos zur Verfügung. Die sich aus der Weiterentwicklung des Quellcodes ergebenden Möglichkeiten wurden damals von vielen Studenten genutzt. Die bekannteste Weiterentwicklung von Unix und verwandten Produkten war die Berkeley Software Distribution (BSD) der University of California, Berkeley. Die vorgenommenen Änderungen an Unix wurden von anderen Entwicklern aufgegriffen, verbessert und wieder nach Berkeley zurückgeschickt. Auf diese Weise wurde Unix stetig verbessert und erweitert.

Richard Stallman machte, wie viele andere Nutzer in der Zeit, die Erfahrung, dass von ihm benötigte Funktionen nicht von kommerziell vertriebener Software angeboten wurden. Als Konsequenz aus diesem Mangel gründete er im Jahr 1985 die Free Software Foundation (Feller & Fitzgerald, 2002). Neben dem Anstoß für die Entwicklung von einigen heute weit verbreiteten Werkzeugen (GNU C Compiler, Emacs-Editor) formulierte Stallman Anforderungen an die "Freiheit" von Software. Diese An-

forderungen in Bezug auf Weitergabe- und Veränderungsmöglichkeiten wurden zwar bereits von einer Reihe von Entwicklern, die ihre Software kostenlos zur Verfügung stellten, implizit berücksichtigt, es gab jedoch noch keine explizite bzw. konsensuelle Formulierung der Ansprüche.

Der Begriff "Free Software" führte zu Missverständnissen, da "free" im Sinne von frei und damit kostenlos verstanden werden konnte. Nicht jede Software, die kostenlos vertrieben wird, erfüllt jedoch die Anforderungen von "Free Software", da z. B. die Möglichkeiten zur Weitergabe oder Veränderung eingeschränkt sein können. Auf der anderen Seite ist "Free Software" nicht automatisch kostenlos, sondern kann auch gegen Bezahlung vertrieben werden. Um die Verbreitung von freier Software auch im kommerziellen Umfeld zu fördern, wurde der Begriff "Open-Source" geprägt. Dies geschah im Zusammenhang mit der Veröffentlichung des Quelltextes des Browsers "Netscape" und sollte eine sprachliche Differenzierung zu der moralisch geprägten und wenig wirtschaftsfreundlichen Sichtweise der Free Software Foundation ermöglichen (*History of the OSI*, 2009). Eine genauere Betrachtung der Unterschiede und Gemeinsamkeiten von Open-Source- und Free-Software findet sich im Kapitel 2.

Das wahrscheinlich bekannteste Beispiel für Open-Source-Software ist das Betriebssystem Linux. Die Entwicklung wurde von Linus Torvalds im Jahr 1991 begonnen. Anlass für die Entwicklung war der Wunsch nach einem mit Unix vergleichbaren Betriebssystem für den PC. Die beginnende Verbreitung des Internets ermöglichte es, dass die Bitte um Unterstützung von Linus Torvalds die Mitarbeit von mittlerweile tausenden Entwicklern (McConnell, 1999; Raymond, 1999a) zur Folge haben konnte. Ein weiteres Beispiel für ein erfolgreiches Open-Source-Projekt ist der Webserver Apache. An ihm wird seit 1995 entwickelt und er ist weiterhin der populärste Webserver (Apache HTTP Server, 2006). Hervorgegangen ist das Projekt aus einer Gruppe von Entwicklern, die sich zusammenfanden, um die Korrektur bzw. Erweiterung des bestehenden NCSA-Servers zu koordinieren. Weitaus älter ist der Ursprung des Textsatzsystems TeX, dessen Entwicklung von Donald Knuth seit 1977 betrieben wird (TeX, 2006). Anlass für die Entwicklung war, wie in vielen Open-Source-Projekten, der persönliche Bedarf von Donald Knuth. Er benötigte ein Textsatzsystem, das ein ästhetisch zufriedenstellendes Ergebnis liefert. Dabei machte Donald Knuth den Quellcode von TeX verfügbar, was zu reger Beteiligung bei der Suche nach Fehlern und zur Erstellung einer Reihe von Werkzeugen, die die Nutzung von Tex unterstützen, führte.

Wie oben beschrieben, wurde der Begriff Open-Source ganz bewusst gewählt, um sich von den politischen und sozialen Zielen der Free-Software-Bewegung abzugrenzen. Damit wollte man Open-Source-Software auch für Unternehmen attraktiver machen und somit auch die Verbreitung und Vermarktung fördern. Dies war durchaus erfolgreich, wie im folgenden Kapitel gezeigt wird.

2.2 Open-Source-Software in Unternehmen

Open-Source-Software hat sich von einer Bewegung engagierter Freizeitentwickler zu einem Geschäftsfeld von bedeutenden Unternehmen der Informationstechnologie entwickelt. So haben Unternehmen wie Apple, Corel, HP, IBM, Intel, Novell oder Oracle mittlerweile Initiativen gestartet, die das Wachstum von Open-Source-Software unterstützen (Bollier, 1999; Dafermos, 2001; Feller & Fitzgerald, 2000). Beispielsweise gab IBM schon im Jahr 2000 bekannt, dass man im Laufe des nächsten Jahres 1 Milliarde US-Dollar im Umfeld von Linux zu investieren beabsichtigt (Görling, 2003).

Open-Source-Software hat auf verschiedene Weisen Einzug in Unternehmen gefunden. So wird Open-Source-Software von Unternehmen als Alternative zu proprietären Softwareprodukten genutzt. Dies ermöglicht neben der Einsparung von Lizenzkosten ein höheres Maß an Transparenz und durch die Nutzung von offenen Standards auch eine verminderte Abhängigkeit von einzelnen Softwareanbietern (Deng, Seifert & Vogel, 2003).

IT-Unternehmen treten nicht nur als Nutzer auf, sondern eine Reihe von Unternehmen hat sich auch als Dienstleister etabliert, die z. B. die Distribution von Open-Source-Software (z. B. Suse oder RedHat) oder Schulung und Unterstützung bei technischen Problemen (z. B. IBM) anbieten (Hecker, 1999). Einige Unternehmen sind zudem aktiv an der Entwicklung und Weiterentwicklung von Open-Source-Software beteiligt. Dabei gibt es Unternehmen, die proprietäre Produkte unter einer Open-Source-Lizenz freigeben und sich in der Weiterentwicklung engagieren, obwohl sie dadurch das geistige Eigentum nicht mehr für sich beanspruchen können. Ein Beispiel hierfür ist StarOffice der Firma Sun Microsystems, die weiterhin maßgeblich an der Entwicklung des Bürosoftwarepakets beteiligt ist, von dem auch ein Open-Source-Produkt (OpenOffice) frei verfügbar ist. Auch der Einstieg in laufende Open-Source-Projekte ist für Unternehmen nicht ungewöhnlich. So haben die Unternehmen Red Hat, Intel, IBM, Novell, Oracle zusammen mehr als ein Drittel der Codezeilen des Linux-Kernels beigetragen (Asay, 2009). Diese versprechen sich von ihrem Engagement eine Verbesserung der Qualität, eine schnellere Weiterentwicklung oder den Einbau von benötigten Funktionen. Dazu werden entweder Mitarbeiter freigestellt oder Entwicklungsaufträge vergeben und entlohnt (Köppen & Nüttgens, 2000).

Open-Source-Software zeichnet sich nicht nur durch ihr Lizenzmodell aus, sondern auch die Art und Weise, wie sie entwickelt wird, unterscheidet sich in vielen Aspekten von der kommerziellen Softwareentwicklung. Im folgenden Kapitel sollen daher gebräuchliche Modelle der proprietären und der Open-Source-Softwareentwicklung verglichen werden.

3 Modelle der Softwareentwicklung

Noch bevor auch nur die ersten mechanischen Rechenmaschinen zur Verfügung standen, wurde von Ada Lovelance in den Jahren 1842/43 ein erstes Programm entwickelt (Computerprogramm, 2009). Mit der zunehmenden Verbreitung von elektronischen Rechenmaschinen hat sich in den zurückliegenden sechzig Jahren auch die Softwareentwicklung beständig weiterentwickelt. Spätestens im Jahr 1968 begann mit der Konferenz "Software Engineering" (Naur & Randell, 1968) die systematische Untersuchung der Frage, welches die beste Methode zur Herstellung von Software darstellt. Aus der Forschung zur Softwareentwicklung sind eine Reihe unterschiedlicher Ansätze hervorgegangen, die heute Anwendung in der Softwareentwicklung finden.

Softwareentwicklungsmodelle dienen primär dazu, die Reihenfolge der einzelnen Phasen der Softwareentwicklung festzulegen und die Kriterien für den Übergang von einer zur nächsten Phase festzulegen (Boehm, 1988). Dabei haben sich die Methoden der Softwareentwicklung beständig weiterentwickelt. Von starren Modellen kann man eine Entwicklung zu Modellen beobachten, die zunehmend mehr Interaktion zwischen einzelnen Phasen des Projektes erfordern. Diese Ansätze werden im folgenden Kapitel 3.1 beschrieben.

Open-Source-Softwareentwicklung weist einige Parallelen dazu auf, wie im proprietären Umfeld Software entwickelt wird. Bei genauerer Betrachtung ergeben sich jedoch eine Reihe von Besonderheiten, die sich insbesondere von starren Modellen der Softwareentwicklung unterscheiden. Diese Eigenschaften sollen im Kapitel 3.2 beschrieben werden und es soll geprüft werden, ob es ein Modell der Open-Source-Softwareentwicklung gibt.

3.1 Softwareentwicklungsmodelle

In der frühen Phase der Softwareentwicklung war ein unstrukturierter Zugang gebräuchlich, den Boehm (1981) als "Kodieren-Reparieren-Modell" (code-and-fix model) bezeichnet. Der Prozess beginnt mit der Programmierung erster Codefragmente und es folgt anschließend ein Bereinigen der im Testlauf festgestellten Fehler, um dann mit der Erstellung weiterer Fragmente fortzufahren. Die vielen notwendigen Änderungen führen zu einem gering strukturierten Code und verursachen hohe Kosten. Zudem ist die Software nur unzureichend an die Bedürfnisse der NutzerInnen angepasst. Die Unzulänglichkeiten dieses Ansatzes machen deutlich, dass neben der Implementierung und Tests noch Phasen der Bedarfsanalyse und des Designs für eine erfolgreiche Softwareentwicklung nötig sind. Alle elaborierten Methoden der Softwareentwicklung beinhalten die gleichen Aktivitäten, binden diese jedoch in eine Struktur ein und ergänzen sie. Je nach

Softwareentwicklungsmodell weisen sie unterschiedliche Schwerpunkte auf. Die gebräuchlichsten Methoden sollen in den folgenden Kapiteln kurz vorgestellt werden.

3.1.1 Phasenmodelle

Phasenmodelle der Softwareentwicklung legen einzelne Abschnitte der Entwicklung (Phasen) fest, die in einer vorgegebenen Reihenfolge sequenziell durchlaufen werden. Das bekannteste Phasenmodell ist das Wasserfallmodell, welches in seiner klassischen Form eine streng stufenartige Abfolge von diskreten Phasen vorsieht (Royce, 1970). Jede weitere Phase wird erst nach der vollständigen Fertigstellung der vorangehenden Phase begonnen. Als einzelne Phasen werden unterschieden: Festlegung von System- und Softwarebedarf, Analyse, Entwurf, Implementierung, Test und Inbetriebnahme (vgl. auch Abbildung 1).

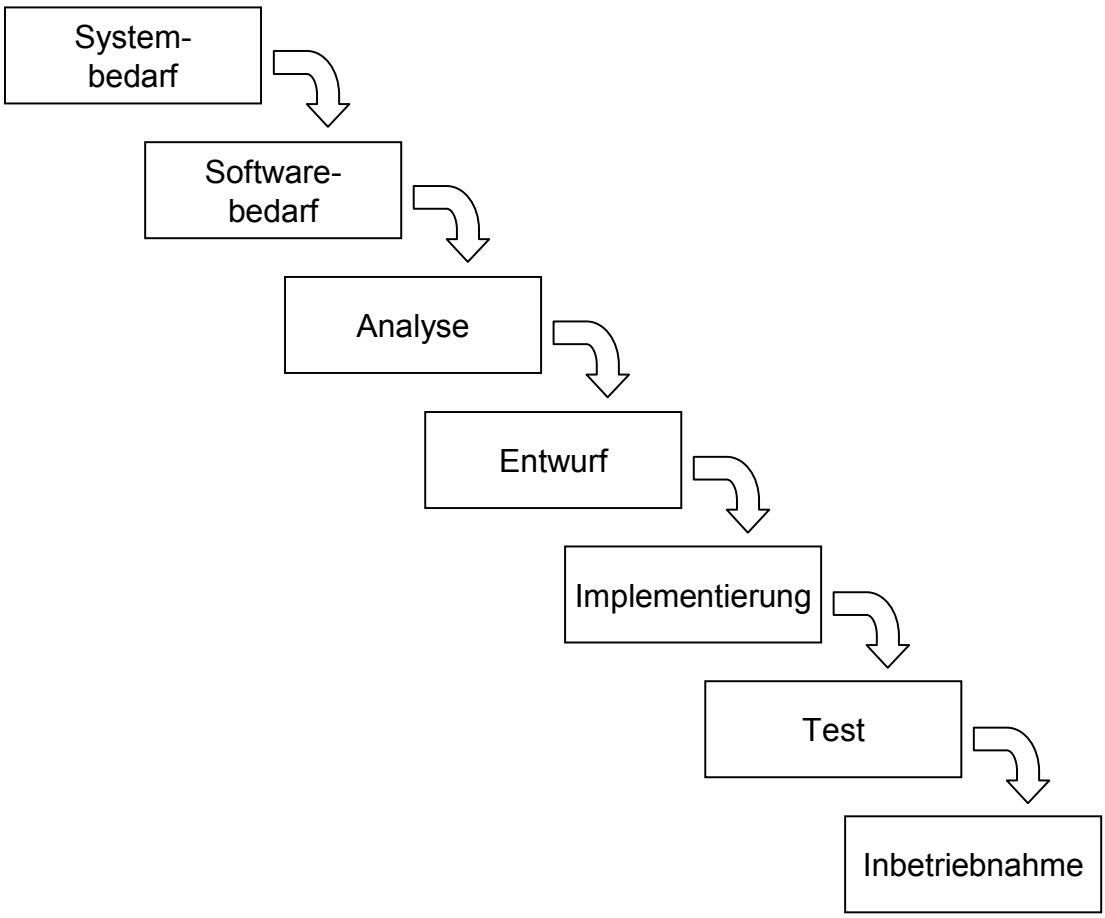

Abbildung 1: Das Wasserfallmodell (Royce, 1970)

Insgesamt ist das Wasserfallmodell leicht verständlich und vermittelt den Eindruck, durch klar definierte Meilensteine einfach zu kontrollieren zu sein (Hibbs, Jewett & Sullivan, 2009). Eine der Konsequenzen aus einer strikten Einhaltung der Entwicklungs-

schritte ist, dass vor jeder Implementierung eine detaillierte Designphase durchgeführt werden muss ("Big Design Up-Front"). Die in die Designphase investierte Zeit ermöglicht es, eventuelle Probleme und Fehler früh zu erkennen und entsprechend im Konzept zu berücksichtigen (Boehm, 1988). Fehler, die frühzeitig im Entwicklungsprozess erkannt werden, können mit geringerem Aufwand beseitigt werden als solche, die zu späteren Zeitpunkten gefunden werden.

> *"Many times, thinking things out in advance saved us serious development headaches later on. [...] Making this change in the spec took an hour or two. If we had made this change in code, it would have added weeks to the schedule." (Spolsky, 2005)*

Ein weiterer Vorteil des Wasserfallmodells liegt in der ausführlichen Dokumentation von Spezifikationen (Davis, Bersoff & Cromer, 1988). Bei Projekten, die vor ihrer Fertigstellung ab- oder unterbrochen werden, gibt es mit der Dokumentation ein Teilergebnis, das für eine spätere Wiederaufnahme genutzt werden kann. Dokumentiertes Wissen macht ein Projekt unabhängiger von der Fluktuation der beteiligten Mitarbeiter.

Kritiker wenden ein, dass das Wasserfallmodell in der Anwendung zu vielfältigen Schwierigkeiten führt. Ein Hauptproblem sind unzureichende Fähigkeiten von Softwareentwicklern (Boehm, 1988). So ist es nur bei trivialen Problemen oder wenn die Entwickler viel Erfahrung aus vorhergehenden Projekten haben möglich, eine Phase vollständig und korrekt abzuschließen, ohne auf Erfahrungen aus nachfolgenden Phasen zurückzugreifen. In den meisten Fällen müsste die Erfahrung aus späteren Phasen herangezogen werden, um einen zufriedenstellenden Abschluss für die aktuelle Phase zu finden. Ein weiteres Problem ist, dass Auftraggeber häufig ihre Anforderungen nur ungenau formulieren und ihre Anforderungen erst dann konkretisieren können, wenn der Entwicklungsprozess bereits so weit fortgeschritten ist, dass Ergebnisse, wie z. B. ein funktionsfähiger Prototyp sichtbar werden (Boehm, 1988). Die so entstandenen Änderungswünsche macht eine Revision der in früheren Entwicklungsphasen ausgeführten Arbeiten notwendig. Bei einem Entwicklungsmodell wie dem Wasserfallmodell, das einen hohen Aufwand in frühen Phasen betreibt und nur langsam zu sichtbaren Ergebnissen führt, besteht ein erhöhtes Risiko, dass größere Teile der bisher ausgeführten Arbeiten verworfen werden müssen. Ebenso hat das Wasserfallmodell Defizite in der Abschätzung der benötigten Zeit und der entstehenden Kosten, da Erfahrungen aus den nachfolgenden Prozessschritten erst zu einem späten Zeitpunkt in die Bewertung einfließen können.

Trotz seiner Einschränkungen ist das Wasserfallmodell weiterhin beliebt und findet häufig in der Praxis Verwendung. Zum Beispiel wird es von amerikanischen Regierungsorganisationen (US Air Force, Department of Defense, NASA) als vorherrschendes

Modell verwendet (Hibbs et al., 2009). In Deutschland kommt der "Entwicklungsstandard für Systeme des Bundes", das V-Modell, zum Einsatz. Dessen Elemente sind mit dem des Wasserfallmodells vergleichbar, jedoch sind Feedbackprozesse zwischen den einzelnen Phasen explizit vorgeschrieben und auch die strenge hierarchische Abfolge der einzelnen Phasen ist aufgeweicht (V-Modell, 2009).

In der ursprünglichen Publikation des Wasserfallmodells nutzte Royce (1970) es als ein Beispiel für ein häufig angewandtes, aber dennoch problematisches Modell. Schon in dieser Publikation wurde versucht, durch die Einführung von diversen Feedbackschleifen zwischen den einzelnen Phasen den Unzulänglichkeiten des Modells zu begegnen. Diese Erweiterungen haben wie auch andere Weiterentwicklungen nur begrenzten Eingang in die Praxis gefunden (Boehm, 1988). Eine systematische Nutzung von Rückmeldungen wurde erst mit den iterativen und inkrementellen Modellen eingeführt, die im folgenden Kapitel vorgestellt werden.

3.1.2 Traditionelle iterative und inkrementelle Modelle

Iterative bzw. inkrementelle Entwicklung von Software zeichnet sich dadurch aus, dass die einzelnen Phasen der Softwareentwicklung (Anforderungsanalyse, Entwurf, Implementierung etc.) nicht nur einmal (siehe Wasserfallmodell; Kapitel 3.1.1), sondern häufiger durchlaufen werden. Dabei erlaubt jeder neue Durchlauf eine Verbesserung der einzelnen Phasen, da auf den Erfahrungen des vorhergehenden Durchlaufs aufgebaut werden kann. Bei iterativer Entwicklung ist der aktuelle Projektstand weniger deutlich zu erkennen, dafür entfällt das schwierige vollständige Design der gesamten Software zu Beginn des Projektes.

Das Spiralmodell (Boehm, 1988) ist nicht das erste iterative Modell. Es ist jedoch das erste Modell, welches die Auswirkungen von iterativem Vorgehen erklärte und es gilt noch heute als eines der populärsten. Besondere Beachtung findet in diesem Modell die Kontrolle von Risiken. Aus diesem Grund beginnt die Entwicklung mit kleinen Teilen des Gesamtsystems, die jeweils nach der Fertigstellung geprüft und bewertet werden. Die Prüfung kann bei jedem Zyklus zu einer Einstellung des gesamten Vorhabens führen oder zu neuen Fragen und Hypothesen, die im folgenden Durchlauf begutachtet werden. Jeder Zyklus besteht dabei in dem Modell von Boehm (1988) aus vier Phasen. Die erste Phase beginnt mit der Festlegung von Zielen, Alternativen und Rahmenbedingungen. In der zweiten Phase sollen vorhandene Alternativen bewertet und vorhandene Risiken erkannt und reduziert werden. Die dritte Phase besteht aus der Realisierung und Überprüfung des Zwischenprodukts. Wenn zu diesem Zeitpunkt noch kein Endprodukt erreichet wurde, schließt sich die vierte Phase an, in welcher die Fortsetzung des Projektes geplant wird (vgl. Abbildung 2). Mit den gewonnenen Erkenntnissen wird im nächsten

Zyklus an der Verbesserung des bisherigen Ergebnisses gearbeitet. Mit jedem Durchlauf nähert sich dabei das Produkt dem Endergebnis an.

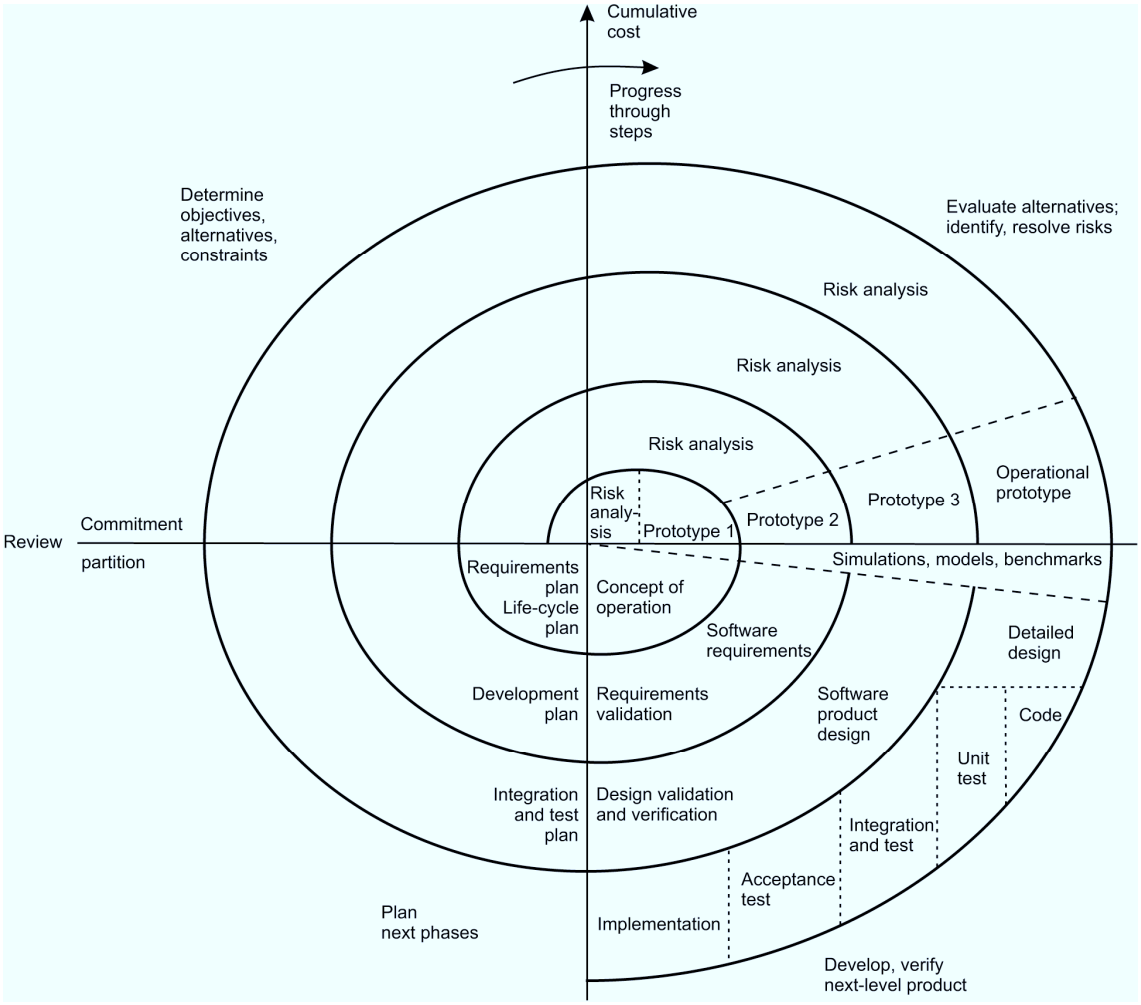

Abbildung 2: Spiralmodell (nach Boehm, 1988)

Im Vergleich zum Wasserfallmodell werden im Spiralmodell dieselben Phasen nicht einmal, sondern wiederholt durchlaufen. Dieses Vorgehen verlangt von den Entwicklern nicht, dass sie das Problem und alle Risiken schon zu Beginn erkennen, sondern sie können ihre Erkenntnisse im Projektfortschritt stetig erweitern. Insbesondere für große und komplexe Projekte stellt eine kontinuierliche Annäherung an ein Ziel einen realistischeren Ansatz dar. Die von den Entwicklern im Laufe des Projekts gesammelten Erkenntnisse erlauben eine zunehmend zuverlässigere Schätzung von Kosten und Zeitbedarf. Die ständige Bewertung der Zwischenergebnisse reduziert das Risiko des Scheiterns des Projektes (Boehm, 1988). Für die Bewertung der Risiken ist allerdings eine ausreichende Erfahrung notwendig. Trotz dieser Vorteile ist das Spiralmodell nicht so weit verbreitet wie einfachere Phasenmodelle. Dies ist auf Vorbehalte von Kunden und

Entwicklern zurückzuführen, denen der bereits am Anfang hohe Detaillierungsgrad in Phasenmodellen den Eindruck von guter Planbarkeit vermittelt.

Eine besondere Form der iterativen und inkrementellen Modelle stellt die Untergruppe der Agilen Softwareentwicklung dar. Diese Ansätze werden im folgenden Kapitel näher dargestellt.

3.1.3 Agile Softwareentwicklung

Agile Softwareentwicklung ist ein konzeptueller Rahmen für eine Reihe von Softwareentwicklungsmethoden, die sich dadurch auszeichnen, dass sie besonders gut auf Veränderungen der Anforderungen reagieren können (Highsmith, 2002). Dieses wird durch eine möglichst unbürokratische Entwicklung in kleinen abgeschlossenen Schritten erreicht. Die Zykluszeiten sind im Bereich von Wochen und damit im Gegensatz zu anderen iterativen und inkrementellen Modellen (vgl. Kapitel 3), die üblicherweise Zykluszeiten von mehreren Monaten haben, sehr kurz. Das Ziel jedes Durchlaufs ist ein neues funktionierendes Programm, nach dem der Fortschritt des Projektes bemessen wird. Damit gehört Agile Softwareentwicklung auch zu den iterativen bzw. inkrementellen Methoden (vgl. Kapitel 3), jedoch erweitert es diese um eine Reihe von Merkmalen, die im Folgenden besprochen werden.

Alle agilen Methoden haben gemeinsam, dass sie persönliche Interaktion höher schätzen als Prozesse und Werkzeuge. Außerdem erhält funktionierende Software einen höheren Stellenwert als eine umfassende Dokumentation. Einer guten Zusammenarbeit mit dem Kunden wird der Vorzug vor vertraglichen Vereinbarungen eingeräumt und Reaktionen auf Veränderungen sind wichtiger als eine genaue Einhaltung des ursprünglichen Plans (Beck et al., 2001). Für einen Prozess, der nach agilen Prinzipien ausgerichtet ist, bedeutet dies, dass die Entwurfsphase auf ein Minimum reduziert wird. Auf diese Weise soll möglichst schnell ein funktionsfähiger Prototyp erstellt werden, der dem Auftraggeber vorgelegt und mit diesem gemeinsam abgestimmt wird. Die hieraus gewonnenen Erkenntnisse werden direkt wieder in einen neuen Prototyp umgesetzt und abgestimmt.

Agile Methoden sind noch relativ neu und haben bisher nur bei einer begrenzten Zahl von Softwareprojekten Einsatz gefunden. Agile Methoden lassen sich erfolgreich einsetzen, wenn Teams über eine geringe Entwickleranzahl (15-20 Personen) und Mitarbeiter mit ausreichender Erfahrung in der Softwareentwicklung verfügen (Cockburn & Highsmith, 2001). Auch bei Entwicklungsaufgaben mit häufig wechselnden Anforderungen sind agile Methoden erfolgreich. Nur wenig Erfahrung liegt bisher für große Teams, die Entwicklung von sicherheitskritischen Anwendungen und für Unternehmenskulturen, die Planung schätzen, vor (Lindvall et al., 2002). Auch vertragsrechtliche Probleme können dem Einsatz von agilen Methoden widersprechen.

Das bekannteste Beispiel für eine agile Methode ist das "Extreme Programming" (XP), dessen Name sich aus dem extremen Vorgehen dieser Methode ableitet (Helmke, Höppner & Isernhagen, 2007; Ruf & Fittkau, 2007). Typisch für das "Extreme Programming" sind sehr kurze Zykluszeiten (typisch eine Woche). In jedem Zyklus werden nur die unbedingt benötigten Teile implementiert, die Formulierung eines strikten Anforderungskatalogs entfällt und der Code wird als die am besten geeignete Form der Dokumentation angesehen (Beck, 2004).

XP integriert dabei Aspekte des Entwicklungsprozesses, die bisher von anderen Methoden nicht oder nur in geringem Maß berücksichtigt wurden: Aspekte der sozialen Interaktion von Entwicklern und Kunden. XP wird neben anderen Eigenschaften als Vereinbarung von Humanität und Produktivität sowie eine Methode zur Herbeiführung von "social change" (sozialem Wandel) beschrieben (Beck, 2004).

Dies spiegelt sich in den fünf zentralen Werten des XP wider. Diese beinhalten keine konkreten Handlungsanweisungen für Projektteams, sondern beschreiben nur ein erstrebenswertes Ziel. Diese Werte sind nach Beck (2004, S. 29ff):

1. *Kommunikation*: Das Ziel der Kommunikation im XP ist es, allen Entwicklern die gleiche Auffassung des Gesamtsystems zu vermitteln; eine Auffassung, die auch von den Nutzer geteilt werden soll. Hierzu werden die Zusammenarbeit von Entwicklern und Nutzer sowie eine offene und verbale Kommunikation der Entwickler untereinander gefordert.
2. *Einfachheit*: Beim XP soll das Design der Software nur den aktuellen Anforderungen genügen. Es steht somit im Gegensatz zu traditionellen Gestaltungsansätzen, die durch Flexibilität auch zukünftigen Anforderungen gerecht werden sollen. Im Erweiterungsfall werden dann aufwendige Neustrukturierungen (refactoring) nötig. Die Vergrößerung der Flexibilität erhöht die Komplexität einer Software und bedeutet zudem einen unnötigen Mehraufwand, wenn antizipierte Funktionen nie genutzt werden.
3. *Feedback*: Alle Beteiligten an einen XP-Entwicklungsprojekt sollen möglichst schnell Rückmeldung über ihre Tätigkeit erhalten. Drei Quellen von Feedback werden im XP intensiv genutzt: Zum Ersten Feedback vom System (Lauffähigkeit von Erweiterungen wird direkt getestet), zum Zweiten Feedback vom Kunden (stellt Anforderungen an die Software, die diese im Funktionstest erfüllen muss) und zum Dritten das Feedback vom Team (gibt Feedback zu neuen Anforderungen des Kunden).
4. *Mut*: Im XP wird der Mut gefördert zu erkennen, wann es sinnvoll ist, bisher geleistete Arbeit zu verwerfen. Dies ist z. B. der Fall, wenn durch eine Neustrukturierung eine größere Einfachheit erreicht werden kann. Als Beispiele für

Mut beschreibt Beck (2004) eine Projektsituation, in der trotz bereits laufender Tests noch grundsätzliche Veränderungen am Design vorgenommen wurden. Ein anderes Beispiel betrifft eine Alltagssituation, bei der sich ein mutiger Entwickler dazu entschließt, seinen am Tag zuvor erstellten Code vollständig zu verwerfen, um stattdessen die bereits gewonnenen Erfahrungen und Erkenntnisse dazu zu nutzen, um dieselbe Funktion einfach und schlank zu erstellen.

5. *Respekt*: In der ursprünglichen Fassung der Werte war Respekt noch nicht enthalten. Es wird als tiefer liegender Wert betrachtet, der die Basis für die anderen vier Werte bildet (Beck, 2004, S. 35). Respekt soll im XP gegenüber anderen Teammitgliedern und auch gegenüber der Arbeit selbst aufgebracht werden. Unter Respekt gegenüber der Arbeit wird im XP das Streben nach hoher Qualität und dem bestmöglichen Entwurf verstanden.

Der am häufigsten genannte Kritikpunkt an XP ist der Umgang mit Änderungen in den Anforderungen (Adolph & Kruchten, 2008). So können in einem XP-Projekt Kunden auf informellem Weg Änderungen herbeiführen, die zu Veränderung des Umfangs eines Projektes oder zu einer Erhöhung von Kosten führen können. In anderen Entwicklungsprojekten sind solche informellen Eingaben nicht möglich, sondern müssen in Gremien abgestimmt und entschieden werden. Weitere Kritikpunkte sind der Verzicht auf Dokumentation und die geringe Anstrengung, die für den Entwurf zu Anfang des Projektes betrieben wird. Kritiker argumentieren, dass dies insgesamt zu mehr Aufwand führt als die nachträgliche Integration von Änderungen in einen umfassenden, zu Beginn angefertigten Entwurf (Adolph & Kruchten, 2008). Ungeklärt ist auch die Frage, ob XP als Methode für jeden Entwickler gleich gut geeignet ist, da hierfür spezifische Fähigkeiten, z. B. Kommunikationsfähigkeiten, erforderlich sind (Copeland, 2001).

Agile Methoden sind eine Antwort auf vorherrschende Modelle der Softwareentwicklung, die nur in geringem Maß Flexibilität erlauben und auf die Bedürfnisse der Entwickler nur unzureichend eingehen. Auch wenn das Open-Source-Entwicklungsmodell eine Reihe von Gemeinsamkeiten mit der agilen Softwareentwicklung aufweist, so existieren doch auch viele Unterschiede, die im folgenden Abschnitt beschrieben werden sollen.

3.2 Softwareentwicklung in Open-Source-Projekten

Softwareentwicklung in Open-Source-Projekten weist eine Reihe von Gemeinsamkeiten mit den verbreiteten Methoden und Praktiken auf, wie sie in proprietären Softwareentwicklungsprojekten angewendet werden. Viele Entwickler aus Open-Source-Projekten sind auch in der proprietären Softwareentwicklung tätig, und häufig verfügen beide Gruppen über dieselbe Ausbildung, sodass es kein Zufall ist, dass proprietäre und Open-

Source-Softwareentwicklung Gemeinsamkeiten aufweisen. So identifiziert Koch (2004) eine Reihe von Gemeinsamkeiten zwischen agiler Softwareentwicklung und Open-Source-Softwareentwicklung: beide Methoden bestehen nicht aus einer genauen Definition eines Prozesses, sondern aus einer Reihe von Prinzipien. Ebenso stehen bei beiden Ansätzen Führungspersönlichkeiten im Mittelpunkt der Entwicklerteams, die von einer größeren Anzahl von Personen unterstützt werden. Sowohl in agilen als auch Open-Source-Softwareentwicklungsprojekten organisieren sich die Teams eigenständig. Beide Ansätze zeichnen sich durch engere Kooperation zwischen Abnehmer/Nutzer und Entwickler aus. In Open-Source-Projekten ist diese Kooperation sogar besonders eng, wenn es sich bei den Entwicklern gleichzeitig um NutzerInnen der Software handelt. Eine weitere Gemeinsamkeit besteht im ausgeprägten Fokus auf die Erstellung von Code und geringerer Investition in Entwurf und Dokumentation.

Neben dem Vergleich mit der Agilen Softwareentwicklung zeigen sich auch Überschneidungen mit dem Spiralmodell. Nach Bollinger et al. (1999) kann Open-Source-Softwareentwicklung als ein sehr intensives Spiralmodell im Sinne von Boehm (1988) betrachtet werden.

Neben den Gemeinsamkeiten gibt es jedoch eine Reihe von originären Eigenschaften, sodass es gerechtfertigt erscheint, von einem eigenständigen Modell auszugehen. Aufgrund der großen Differenzen zwischen verschiedenen Open-Source-Projekten (z. B. Nakakoji & Yamamoto, 2001) ist die Frage, was das Open-Source-Entwicklungsmodell ist, nicht eindeutig zu beantworten. Es existieren jedoch eine Reihe gemeinsamer Eigenschaften, die in fast allen Open-Source-Projekten anzutreffen sind: Open-Source-Projekte bestehen in der Regel aus einer *virtuellen Gemeinschaft* von Entwicklern, die sich primär per computervermittelter Kommunikation austauschen. Der Einsatz des Internets erlaubt hierbei eine globale Verteilung der Entwickler und auch eine große Anzahl von Personen, die in die Entwicklung einbezogen werden können. Die Einzelbeiträge der Entwickler werden von anderen ProjektteilnehmerInnen im sogenannten *Peer-Review*-Verfahren auf Fehler überprüft. Die Projekte weisen einen hohen Grad der *Selbstorganisation* auf. Durch die Freiwilligkeit und die geringen Kosten für einen Beitritt bzw. Austritt aus einem Projekt gibt es eine starke *Selbstselektion*. Zudem weist die *Arbeitsorganisation* einige Besonderheiten auf, wie die sehr ausgeprägte Modularisierung, das hoch parallele Arbeiten und die häufige Veröffentlichung von neuen Versionen. In diesem Kapitel sollen die wesentlichen Charakteristika der Softwareentwicklung in Open-Source-Projekten aufgezeigt werden. Es sollen Unterschiede zur proprietären Softwareentwicklung herausgearbeitet und es soll gezeigt werden, auf welche Merkmale der Tätigkeit sich die Eigenschaften der Open-Source-Softwareentwicklung auswirken. Eine detaillierte Analyse der Tätigkeitsmerkmale folgt im Kapitel 4.

Am Anfang eines neuen Open-Source-Softwareprojekts steht zumeist das Bedürfnis eines Einzelnen nach einer Software mit einer speziellen Funktion (Moody, 2001; Vixie, 1999). Dabei kann bisher noch keine Software verfügbar sein, die diese Anforderungen erfüllt, oder es gibt sie nur in einer kommerziellen Fassung. Als hilfreich für den Start eines Projektes hat sich zudem ein funktionsfähiger Prototyp erwiesen, der mitsamt dem Quellecode veröffentlicht wird. Bei ausreichend attraktiven Ideen finden sich weitere Interessenten, die das Projekt durch eigene Beiträge erweitern. Der weitere Projektablauf besteht darin, dass neue Funktionen implementiert oder Fehler durch Erweiterungen bzw. Modifikationen bereinigt werden. Dies geschieht gewöhnlich in Form von kleinen Quellcodeteilen, die von den Entwicklern in das bestehende System übermittelt werden, den sogenannten "Patches".

Es gibt verschiedene, Arten wie Personen einen Beitrag zu Open-Source-Projekten leisten können. Die wichtigsten Rollen in einem Open-Source-Projekt übernehmen die aktiven und passiven NutzerInnen, die Mitentwickler, die Kernentwickler und der Initiator (Crowston, Scozzi & Buonocore, 2002), wobei jeder Beteiligte durchaus mehrere Rollen ausüben kann. Personen, die nur die Open-Source-Software nutzen, ohne zur Entwicklung beizutragen, werden als *passive NutzerInnen* bezeichnet. Der *aktive Nutzer/die aktive Nutzerin* beteiligt sich am Open-Source-Projekt durch die Meldung von aufgetretenen Fehlern oder den Wunsch nach neuen Funktionen. Aktive NutzerInnen erstellen im Gegensatz zu den *Softwareentwicklern* jedoch keinen Quellcode. Die Entwickler lassen sich in zwei Gruppen unterteilen: die Mitentwickler und die Kernentwickler. Die *Mitentwickler* tragen nur sporadisch zu einem Projekt bei. Zumeist geschieht dies in der Form von Fehlerbeseitigung oder Verbesserung an bereits bestehenden Programmteilen. Üblicherweise trifft man neben dem Initiator einen *Kern von Entwicklern*, die durch besonders hohes Engagement auffallen. Oft wird in diesem Kreis von Entwicklern nicht nur die meiste Arbeit geleistet (vgl. Kapitel 3.2.1), sondern auch maßgeblich Einfluss auf die grundlegende Entscheidung des Projektes genommen (Crowston et al., 2002). In vielen großen Projekten[2] (z. B. auch dem Linux-Kernel-Projekt) ist es einen kleinen Kreis vorbehalten, darüber zu entscheiden, welche neuen Beiträge (Patches) in die offizielle Version aufgenommen werden und welche nicht. Im Linux-Projekt werden die zentralen Entscheidungen von einem wechselnden Release Manager getroffen. Dem Release Manager arbeiten die "Maintainer" zu. Jeder Maintainer übernimmt die Verantwortung für einen Teilbereich der Entwicklung. Sie entscheiden selbstständig, welche Teile aufgenommen bzw. verschoben oder verworfen werden sollen (Erenkrantz, 2003).

[2] In anderen Projekten ist dies einem wesentlich größeren Kreis von Entwicklern möglich. So hatten im Debian-Linux-Projekt im Jahr 2001 z. B. 700 Entwickler vollständige Zugriffsrechte (Hill, 2001).

3.2.1 Virtuelle Gemeinschaft

Im Gegensatz zu den traditionellen Modellen der Softwareentwicklung gehören die Beteiligten eines Open-Source-Projektes üblicherweise nicht einem Unternehmen oder einer Organisation an. Open-Source-Projekte stehen grundsätzlich jedem Interessierten offen (Shah, 2003). Open-Source-Projekte lassen sich als "virtuelle Gemeinschaft" charakterisieren. Diese zeichnen sind nach Döring (2000a) aus durch:

- Überwiegend computervermittelte Kommunikation
- Regelmäßige Kommunikation
- Stabilität über einen längeren Zeitraum
- Soziale Strukturierungen (Macht, Status, Normen, Regeln etc.)

Im Gegensatz zu anderen virtuellen Gemeinschaften, die sich zu Themen wie Hobbys, Spielen oder sozialer Unterstützung zusammengefunden haben, steht bei der Open-Source-Softwareentwicklung nicht der gegenseitige Austausch, sondern die Erstellung eines Softwareprodukts im Mittelpunkt des Interesses (O'Mahony & Ferraro, 2004). Gruppen, bei denen der Schwerpunkt auf der Erfüllung gemeinsamer Aufgaben liegt, werden von Cramton (2001) als virtuelle Teams angesehen. Er definiert virtuelle Teams als "Gruppen, die mit einem gemeinsamen Ziel, die zu unterschiedlichen Zeiten und Orten interdependente Tätigkeiten ausführen und dabei hauptsächlich über elektronische Medien kommunizieren" (übersetzt nach Cramton, 2001, S. 346). Abweichend von Cramton (2001) gibt es noch eine Reihe weiterer Definition von virtuellen Teams. Als Mindestanforderungen gelten 1. zwei oder mehr Personen, 2. interaktive Zusammenarbeit zum Erreichen von gemeinsamen Zielen, 3. mindestens ein Mitglied des Teams arbeitet an einem anderen Ort und 4. Kommunikation basiert hauptsächlich auf elektronischen Medien (Hertel, Geister & Konradt, 2005a). Im Gegensatz zu Cramton (2001) betont diese Definition die Zusammenarbeit der Teammitglieder und die unterschiedlichen Orte der Teammitglieder. Hertel et al. (2005a) machen dagegen keine Aussage zur Interdependenz der Aufgaben. Da bei Open-Source-Projekten in den meisten Fällen mehrere Personen an der Erstellung von Software arbeiten bzw. viele Projekte mit weltweiten Entwicklergruppen arbeiten und die Kommunikation meist ausschließlich über elektronische Medien stattfindet, können Open-Source-Projekte auch nach dieser Definition als virtuelle Teams angesehen werden. Da die beiden letzten Punkte der Definition (anderer Ort der Teammitglieder und Einsatz von elektronischen Medien) nicht als Eigenschaften, sondern als Dimensionen gelten, können viele Open-Source-Projekte als sehr ausgeprägte virtuelle Teams angesehen werden.

Einen wichtiger Einflussfaktor ist sowohl in virtuellen Gemeinschaften als auch in virtuellen Teams die Anzahl der Mitglieder. Gesicherte Zahlen für die Größe von Open-Source-Projekten gibt es nicht. Für das Linux-Kernel enthält die "Credits"-Datei eine Übersicht aller Entwickler, die maßgeblich zum Linux-Kernel beigetragen haben (Tuomi,

2004). Für die Version "2.6.32-rc5-ccs-1.7.1-pre" des Linux aus dem Oktober 2009 finden sich in der "Credits"-Datei insgesamt 505[3] eingetragene Personen. Diese Angaben erlauben jedoch keine Aussage über das gesamte Linux-Projekt, da nur die wichtigsten Entwickler in dieser Datei aufgeführt werden und das Linux-Kernel nur einen kleinen Teil des gesamten Linux-Projekts darstellt. Das Größenverhältnis wird deutlich, wenn man betrachtet, dass die Version 4 der Debian Linux Distribution insgesamt ca. 283 Millionen Zeilen enthält (Robles, 2007), wovon 3 % (ca. 8 Millionen Zeilen in der enthaltenen Version 2.6.18: Kroah-Hartman, 2007) auf den Linux-Kernel entfallen. Schätzungen für das Projekt Linux, das sicher zu den größten Projekten gehört, belaufen sich auf 40.000 (Raymond, 1999a) oder "Hunderttausende" (Torvalds & Diamond, 2001) Entwickler. Neben einigen Großen gibt es aber auch eine Vielzahl von sehr kleinen Open-Source-Projekten; diese bestehen z. T. nur aus einem oder wenigen Entwicklern. So betrug der Mittelwert der Anzahl an Entwicklern in den 100 untersuchten Projekten von Krishnamurthy (2002) nur 6,6. Der Median lag bei 4, der Modus nur bei 1.

Die Anzahl der tatsächlich an sozialen Interaktionen beteiligten Entwickler sollte noch geringer sein, denn ein großer Anteil des Entwicklungsaufwands wird von einer kleinen Gruppe an Entwicklern geleistet (z. B. Crowston & Scozzi, 2002b; Koch & Schneider, 2002; Mockus et al., 2002; Ye & Kishida, 2003). So haben z. B. zum Open-Source-Projekt Apache ca. 3.000 Personen einen Beitrag in Form von Fehlerberichten geleistet, immerhin noch 400 haben Programmteile erstellt und übermittelt, jedoch wurden 88 % der Software (gemessen an der Anzahl der Quellcodezeilen) von nur 15 Entwicklern erstellt (Mockus et al., 2002). Etwas breiter verteilt sich die Arbeit im GNOME-Projekt, bei dem 15 der 301 Programmierer für immerhin 48 % des Quellcodes verantwortlich sind. Insgesamt waren es 1.900 Personen, die ihre Aktivität durch mindestens einen Beitrag im E-Mail-Verteiler demonstrierten (Koch & Schneider, 2002). Einen Überblick über eine Vielzahl von Projekten ermöglichte eine Studie zum Entwicklungsforum Sourceforge, das Infrastruktur für über 1.000.000 Entwickler und mehr als 100.000 Projekte bereitstellt (Stand März 2006). Bei einer Stichprobe von 12.000 Entwicklern und mehr als 7.000 Projekten stellte sich heraus, dass die aktivsten 10 % der Entwickler für mehr als 70 % des gesamten Quellcodes verantwortlich sind (Crowston & Scozzi, 2002b). Ebenfalls nachteilig auf soziale Interaktionen sollte sich die höhere Fluktuation in elektronischen Gruppen auswirken (Cummings, Butler & Kraut, 2002b), die stabile Beziehungen erschwert. Die Intensität von sozialen Interaktionen hat einen Einfluss auf die sozialen Aspekte der Tätigkeit, wie die *Rückmeldung durch Personen* oder die *soziale Unterstützung*.

[3] Die "Credits"-Datei ist Bestandteil jeder Linux Distribution. Die Zahl 505 ist durch Zählen der dort befindlichen Einträge ermittelt worden.

Eine Reihe von Forschern (z. B. Elliott & Scacchi, 2004; Kogut & Metiu, 2001; Lanzara & Morner, 2003; Lee & Cole, 2003; Manville, Markus & Agres, 1999; Moon & Sproull, 2002; Neus, 2001; Osterloh, Rota & Kuster, 2004; Scacchi, 2002; Tuomi, 2000) betont den Aspekt der gemeinsamen Tätigkeit und des Wissenserwerbs und bezeichnet Open-Source-Projekte als virtuelle "Communities of Practice[4]". Communities of Practice sind informelle Gruppen, die durch gemeinsames Wissen und Interesse verbunden sind. Die Mitglieder organisieren sich selbst, arbeiten häufig an vergleichbaren Aufgaben, unterstützen sich gegenseitig und tauschen Erfahrungen aus (Brown & Duguid, 1991; Wenger, 1998). Communities of Practice werden häufig in Organisationen als Methode zur Verbesserung des Wissenserwerbs eingesetzt (Boland & Tenkasi, 1995; Brown & Duguid, 1998; Davenport & Prusak, 1998). Communities of Practice[5] existieren nicht nur innerhalb von Organisationen und sind auch nicht zwangsläufig auf die Lokalisierung an einem Ort angewiesen (Faraj & Wasko, 2001). Ein großes Netzwerk, bestehend aus flüchtigen Bekanntschaften, kann bei der Gewinnung von Wissen hilfreich sein. Zumindest legt dies die Theorie der Weak Ties (weak ties, Granovetter, 1973; Granovetter, 1982) nahe. Nach dieser Theorie erhält man mehr nützliches Wissen, wenn man auf ein großes Netzwerk mit Weak Ties (z. B. Bekanntschaften) zurückgreift, als wenn man sich auf ein Netzwerk mit starken Beziehungen (Freunde, Familie, enge Arbeitskollegen) verlässt. Möglich wird dies, weil Weak Ties zumeist über wesentlich mehr zusätzliches Wissen verfügen, als in einem Netzwerk von starken Beziehungen vorhanden ist (Lin, 2001). Über die Weak Ties können zusätzliche Netzwerke von Wissensträgern erreicht werden, die sonst nicht zur Verfügung stehen würden (Burt, 1983; Granovetter, 1982). Im Gegensatz zu den Communities in Open-Source-Projekten wird proprietäre Softwareentwicklung üblicherweise von einem Team betrieben. Teams verfügen in der Regel über deutlich weniger Mitglieder als Communities (Hertel et al., 2005a; Wenger, 1998) und weisen auch stärkere Bindungen auf. Dies führt dazu, dass weniger Wissen in solchen Projekten für die TeilnehmerInnen verfügbar ist. Der größere Umfang an erreichbarem Wissen führt zu einer Erhöhung des *Qualifikationspotenzials* in Open-Source-Projekten (vgl. Kapitel 4.3.6).

---

[4] Virtual Communities of Practice (Neus, 2001) erscheinen in der Literatur unter unterschiedlichen Bezeichnungen: Electronic Communities of Practice (Wasko & Faraj, 2000), Distributed Community of Practice (Kogut & Metiu, 2001), Network of Practice (Faraj & Wasko, 2001).

[5] Faraj & Wasko (2001) benutzen nicht die Bezeichnung Virtual Community of Practice, sondern Network of Practice. Dabei unterscheiden sich Networks of Practice of Communities of Practice durch das Fehlen der Begrenzung auf Organisationsmitglieder oder einen gemeinsamen Ort. Zudem ist die Mitgliedschaft für jeden offen und Nachrichten sind für die gesamte Gemeinschaft sichtbar.

3.2.2 Computervermittelte Kommunikation

Die Nutzung von computervermittelter Kommunikation ist eines der entscheidenden Merkmale von virtuellen Gemeinschaften. Der universale Zugang zu computervermittelter Kommunikation durch die Verbreitung des Internets war eine der Voraussetzungen für die Entstehung von Open-Source-Communities. Bei der Entwicklung von Linux benutzte Linus Torvalds schon in der Anfangsphase das Internet, um die ersten Entwürfe des Betriebssystems zu veröffentlichen und Mitentwickler zu suchen (Dafermos, 2001; Demil & Lecocq, 2003). Erst das Internet ermöglichte es, die Interessenten an diesem Projekt zusammenzuführen und den hohen Kommunikationsbedarf in einen Softwareentwicklungsprojekt zu befriedigen.

Durch die hohen Interdependenzen zwischen einzelnen Bestandteilen von Software erfordert die Entwicklung ein hohes Maß an Koordination und Kommunikation der TeilnehmerInnen (Herbsleb & Mockus, 2003). Untersuchungen zur Softwareentwicklung in Unternehmen identifizieren Koordination und Kommunikation als eine der zeitaufwendigsten Tätigkeiten im Alltag der Entwickler. So wird im ersten Monat eines Projektes ungefähr 50 % der Arbeitszeit in Form von Gruppenarbeit verbracht. Bei fortgeschrittenen Projekten beträgt dieser Anteil immerhin noch 10 % der Arbeitszeit (Herbsleb et al., 1995). Ein großer Teil der Kommunikation wird dabei im persönlichen Gespräch (Face-to-Face) abgewickelt (Hauptman, 1986; Krasner, Curtis & Iscoe, 1987; Page & Caskey, 1988). Die Entwickler in der Untersuchung von Perry, Staudenmayer und Votta (1994) verbrachten beispielsweise 75 Minuten pro Tag mit ungeplanter arbeitsbezogener Interaktion. Im Gegensatz dazu ist in Open-Source-Softwareentwicklung die Face-to-Face-Kommunikation die Ausnahme und als häufigstes Medium wird das Internet genutzt. Verschiedene Untersuchungen deuten darauf hin, dass Distanz und der damit einhergehende geringere direkte persönliche Austausch einen erheblichen Einfluss auf die Qualität der Kommunikation hat. Für Ingenieure und Wissenschaftler konnte gezeigt werden, dass schon geringe Distanzen (30 Meter) ausreichen, um die Häufigkeit der Interaktion drastisch zu reduzieren (Allen, 1977; Kraut, Egido & Galegher, 1988). Es gibt einige empirische Hinweise, die darauf schließen lassen, dass Entfernung es schwieriger macht, Softwareeentwicklung zu koordinieren (Carmel, 1999; Curtis, Krasner & Iscoe, 1988; Herbsleb & Grinter, 1999) und die Entwicklung verlangsamt (Herbsleb, Mockus, Finholt & Grinter, 2001).

Zu den ersten untersuchten Erklärungsansätzen für die Auswirkungen von Kommunikation mithilfe von Computern gehören die Theorie der Kanalreduktion und die Theorie des Herausfilterns sozialer Hinweisreize. Die Ansätze zur Kanalreduktion basieren auf der Annahme, dass computervermittelte Kommunikation im Vergleich zu einer Face-to-Face-Situation ("natürliche" Gesprächssituation, bei der alle Kommunikationspartner an einem Ort zusammenkommen) aufgrund von fehlenden

Sinneskanälen defizitär und unpersönlich ist (siehe zusammenfassend Döring, 2000b). Die Theorie des Herausfilterns sozialer Hinweisreize (Culnan & Markus, 1987; Kiesler, Siegel & McGuire, 1984) geht ebenfalls von einem Informationsverlust durch weniger Sinneskanäle aus, jedoch postuliert sie als Konsequenz einen enthemmenden Effekt. Dieser kann einerseits zu unpersönlichen und feindseligen Diskussionen führen (sogenannte "Flame Wars"), begünstigt aber andererseits auch Offenheit, Ehrlichkeit und Egalität. Beiden Theorien ist gemeinsam, dass sie das Verhalten der NutzerInnen aus einer technikdeterministischen Sicht erklären, die Personen und Situationen geringe Einflussmöglichkeiten zugesteht. Sowohl Personen als auch Situationen weisen in Open-Source-Projekten eine Reihe von Besonderheit auf, die im Folgenden betrachtet werden sollen.

Im Mittelpunkt der Media-Richness-Theorie (Daft & Lengel, 1986) stehen die Anforderungen der spezifischen Kommunikationssituation. Nach diesem Konzept gibt es eine Hierarchie von Medien, und zwar in Abhängigkeit ihrer Reichhaltigkeit. Die Reichhaltigkeit eines Mediums wird durch das Ausmaß der Reichhaltigkeit der übertragenen Informationen (Information Richness) festgelegt. Reichhaltigere Medien können demnach in einem kürzeren Zeitintervall das Verständnis verändern (Daft & Lengel, 1984; Daft & Lengel, 1986). Die Reichhaltigkeit eines Mediums basiert dabei auf vier Kriterien: der Möglichkeit für Rückmeldungen, der Anzahl der nutzbaren Hinweisreize bzw. Kanäle, der Möglichkeit der Personalisierung und der Gelegenheit zur sprachlichen Variation (Daft & Lengel, 1986). Jede Kommunikationsaufgabe stellt unterschiedliche Anforderungen an die Reichhaltigkeit des Mediums, und gemäß der Theorie der Media Richness steigt die Effizienz der Kommunikation, wenn ein geeignetes Medium ausgewählt wird. Wenn Aufgabenanforderungen und Medium nicht zusammenpassen, führt dies – je nach Auswahl von zu reichhaltigen oder nicht ausreichend reichhaltigen Medien – entweder zu erhöhten Kosten oder zum Scheitern der Aufgabe.

Untersuchungen zeigen, dass es Präferenzen bei der Wahl von bestimmten Aufgaben gibt. So werden z. B. für das "Kennenlernen" oder das "Lösen von Konflikten" nichtelektronische Kommunikationsformen wie die Face-to-Face-Kommunikation oder das Telefon bevorzugt (Rice, 1993). Ein Unterschied bzgl. der Zufriedenheit mit dem Medium konnte genauso wenig nachgewiesen werden wie ein konsistenter Unterschied in der Qualität von Entscheidungen (Kinney & Dennis, 1994; Suh, 1999; Valacich, Mennecke, Wachter & Wheeler, 1994). Beim Einsatz von textbasierten Medien war jedoch mehr Zeit für die Lösung notwendig als bei reichhaltigeren Medien (Kinney & Dennis, 1994; Kinney & Watson, 1992; Suh, 1999; Valacich et al., 1994).

Nicht nur die Kommunikationsaufgaben in Open-Source-Projekten weisen Besonderheiten auf, sondern auch die KommunikationsteilnehmerInnen. Im Vergleich zu anderen NutzerInnen von elektronischen Kommunikationsmitteln verfügen Softwareentwickler

über eine hohe Technikkompetenz und -neigung. Hinzu kommt noch ein Prozess der Selbstselektion: Entwickler, die weniger bereitwillig elektronische Kommunikationsmedien benutzen, werden sich weniger häufig in Open-Source-Projekten wiederfinden. Theorien zur computervermittelten Kommunikation, die neben den Eigenschaften von Medien und Aufgaben auch Einflussfaktoren berücksichtigen, die in der sozialen Umwelt bzw. der Person liegen, sind das Social-Influence-Modell der Technologienutzung (Fulk, Schmilz & Steinfield, 1990), die Theorie der sozialen Informationsverarbeitung (Walther, 1992; Walther, 1996) und das Modell zum Einfluss von Persönlichkeitsfaktoren (Hertel, Schroer, Batinic, Konradt & Naumann, 2005b).

Das Social-Influence-Modell betont im Gegensatz zum Konzept der Media Richness (s. o.) die nicht-rationalen Komponenten bei der Auswahl des Mediums. Dies ist neben den sozialen Normen der potenziellen Kommunikationspartner die Bedienungskompetenz des Nutzers. Eine größere Kompetenz (schnelleres Tippen, Verwendung von Akronymen und Abkürzungen) führt dazu, dass ein Medium lebendiger bzw. reichhaltiger eingeschätzt wird und im stärkeren Maße genutzt wird (für empirische Ergebnisse am Beispiel von E-Mail siehe Schmitz & Fulk, 1991).

Die Theorie der sozialen Informationsverarbeitung (Walther, 1992; Walther, 1996) geht noch einen Schritt weiter: nach dieser verursacht die eingesetzte Technik keine Veränderungen der Kommunikation, sondern führt dazu, dass Menschen durch verändertes Nutzungsverhalten die Einschränkungen kompensieren. Diese Kompensation kann sich in der Entwicklung von neuen sozialen Fertigkeiten bei der Erstellung und Interpretation von Texten zeigen (z. B. durch Textzeichen die Emotionen ausdrücken, Emoticons). Störungen in der Kommunikation sind dann zu erwarten, wenn die NutzerInnen ungeübt sind, netzspezifische Kommunikationscodes nicht beherrschen, unter Zeitdruck stehen, einander kaum kennen und auch kein Interesse am Kennenlernen haben.

Das Modell von Hertel et al. (2005b) berücksichtigt noch Eigenschaften von Persönlichkeiten. Es wurde gezeigt, dass sich Extraversion und Neurotizismus über die sozialen Fähigkeiten und die soziale Ängstlichkeit auf die Wahl des Mediums auswirken. So ziehen introvertierte Personen in Konfliktsituationen den Einsatz von E-Mails einem persönlichen Gespräch vor. Bei Routineanlässen zeigt sich kein Unterschied zwischen introvertierten und extravertierten Personen. Für Neurotizismus war die Präferenz des Mediums unabhängig von der Gesprächssituation: emotional stabile Personen bevorzugten stärker das persönliche Gespräch, während emotional instabile Personen E-Mails präferierten.

Der Vergleich von Face-to-Face-Situationen und elektronischer Kommunikation fällt in der Bewertung von Döring (2000b) kritisch aus:

> *"[D]ie These von der vollständigen Übersetzbarkeit in Textzeichen [erscheint] jedoch recht gewagt." (Döring, 2000b, S. 363)*

In der Softwareentwicklung, mit hoch interdependenten und mehrdeutigen Aufgaben, sollte sich der Einsatz von reichhaltigeren Medien förderlich auswirken (Daft & Lengel, 1986). Bei der Verwendung von elektronischer Kommunikation sollte sich in Open-Source-Projekten als zusätzlicher Nachteil erweisen, dass die beteiligten Entwickler sich zumeist nicht persönlich kennen, was nach der sozialen Informationsverarbeitung zu Kommunikationsstörungen führen kann (s. o.). Auch die negativen Effekte von zeitlichen Restriktionen (Walther, Anderson & Park, 1994) sind in Open-Source-Projekten nicht unwesentlich. Ein großer Anteil der Entwickler in solchen Projekten ist beruflich aktiv oder studiert und verfügt daher nur über limitierte zeitliche Ressourcen. Selbst in großen und global verteilten Projekten können entsprechend seltene Face-to-Face-Kontakte einen wesentlichen Einfluss auf wichtige Entscheidungen nehmen (O'Mahony & Ferraro, 2004). Aus einer Vielzahl von eigenen Untersuchungen schließen Cummings et al. (2002b), dass auch über elektronische Medien soziale Beziehungen geknüpft und aufrechterhalten werden können, aber dass diese bei einem direkten Vergleich mit Face-to-Face-Kontakten und Telefonaten Defizite aufweisen. So erwies sich E-Mail als weniger nützlich für die Entwicklung und Aufrechterhaltung von persönlichen Beziehungen, E-Mail-Verteiler eignen sich weniger gut, um ein Zugehörigkeitsgefühl zu entwickeln und soziale Unterstützung zu gewähren. Insgesamt schätzen die Autoren Internet-Beziehungen als weniger eng ein, als andere Beziehungen.

Die stärkere Abhängigkeit von elektronischen Medien sollte sich also tendenziell eher nachteilig auf die Qualität der Kommunikation in der Open-Source-Softwareentwicklung auswirken.

Neben dem Austausch von Informationen kann Kommunikation auch *soziale Unterstützung* bieten. Diese stellt bei der Arbeit eine wichtige Ressource dar (Karasek et al., 1998). Dass soziale Unterstützung generell auch über elektronisch vermittelte Kommunikation möglich ist, zeigt die Existenz einer großen Vielzahl von Angeboten im Internet, bei der sich virtuelle Gemeinschaften mit beispielsweise medizinischen Problemen und schwierigen Lebenslagen beschäftigen. Allein bei Yahoo!Groups (www.yahoo.com) werden mehr als 25.000 elektronische Themengruppen gelistet, die sich nur mit den Themen Gesundheit und Wellness beschäftigen (Eysenbach, Powell, Englesakis, Rizo & Stern, 2004). Die große Anzahl von NutzerInnen verbunden mit den Suchmöglichkeiten des Internets, erlaubt es auch bei ausgefallenen Fragen und Problemen, Menschen in der gleichen Situation zu finden (Wellman & Gulia, 1999).

Dass in elektronischen Selbsthilfegruppen nicht nur Informationen ausgetauscht werden, sondern tatsächlich Unterstützung geleistet wird, kann man z. B. an der Studie von

Dunham et al. (1998) erkennen, bei der junge Mütter sich über ein elektronisches Schwarzes Brett (Electronic Bulletin Board) austauschten. Dabei wurde der Hauptteil der erstellten Beiträge (56 %) von den Forschern als positive emotionale Unterstützung klassifiziert. Neben der Möglichkeit, seinen Standpunkt auszudrücken, wird in virtuellen Gemeinschaften die Möglichkeit, soziale Unterstützung zu erhalten, als wichtigster Teilnahmegrund angegeben (Herring, 1996). In Gruppen, die elektronisch vermittelt kommunizieren, können neue Personen kennengelernt, Freundschaften geschlossen und eben auch emotionale Unterstützung gegeben werden (Galegher, Sproull & Kiesler, 1998). Selbst in fachlich orientierten computervermittelten Diskussionen mit TeilnehmerInnen, die sich nie persönlich treffen, ist ein nennenswerter Anteil der Kommunikation durch sozioemotionale Inhalte geprägt (Rice & Love, 1987; Wiertz, de Ruyter & Streukens, 2003). In Studien von Rice und Love (1987) bestand 30 % der Kommunikation aus sozioemotionalen Beiträgen.

Vergleichbar anderen fachlich orientierten Gruppen findet die Kommunikation in Open-Source-Projekten nicht mit dem Ziel statt, soziale Unterstützung zu leisten, sie ist jedoch auch ein Bestandteil der Kommunikation. Im Vergleich mit anderen Softwareentwicklungsmodellen stellt sich die Frage nach der Qualität dieser Unterstützung. In der proprietären Softwareentwicklung befinden sich die Teammitglieder zumeist an einem gemeinsamen Standort und haben regelmäßig die Möglichkeit, Face-to-Face miteinander zu kommunizieren. Insbesondere die agilen Methoden der Softwareentwicklung betonen den persönlichen Austausch von Teammitgliedern (und Kunden) als wichtigen Bestandteil des Entwicklungskonzepts (Beck et al., 2001). Computervermittelte Kommunikation wird dann verstärkt genutzt, wenn Unterstützung über andere Quellen nur im geringeren Ausmaß zur Verfügung steht (Davison, Pennebaker & Dickerson, 2000; Galegher et al., 1998; McKenna & Bargh, 1998; Mickelson, 1997). Dies könnte ein Indiz dafür sein, dass computervermittelte Kommunikation für bestimmte Formen des Austauschs nicht das Mittel der Wahl ist, solange Alternativen zur Verfügung stehen. Aus ihren Untersuchungen schließen auch Cummings et al. (2002b), dass Diskussionsforen (listservs) weniger gut gemeinsame Identität fördern und soziale Unterstützung leisten als reale Gruppen.

Diese Eigenschaften von computervermittelten Gruppen werden durch das Vorherrschen von Weak Ties erklärt (Cummings, Kiesler & Sproull, 2002a; Cummings et al., 2002b; Parks & Roberts, 1997). Gruppen mit Weak Ties neigen dazu, weniger soziale Unterstützung zu bieten als solche mit starken Beziehungen (Wellman, 1992). Ursächlich für Weak Ties in elektronischen Gruppen sind primär die niedrigen Kosten für den Beitritt und Austritt (Pickering & King, 1995) und die hohe Fluktuation (Butler, 2001).

Insgesamt erscheint die soziale Unterstützung (vgl. Kapitel 4.4.1) bei der Tätigkeit in Open-Source-Projekten geringer als bei Softwareentwicklung, die auf persönlicher

Kommunikation beruht. Trotz möglicher Herausforderungen von Kommunikation auf computervermitteltem Wege können auch große Gruppen in Open-Source-Gemeinschaften zusammenkommen und sich erfolgreich koordinieren (Koch, 2004). Dies zeigt sich auch darin, dass einige der Eigenschaften von elektronischen Medien die Kommunikation begünstigen.

Ein großer Vorteil von elektronischen Medien, wie dem Internet, ist die Möglichkeit, mit geringem Aufwand eine Gruppe von Personen zu informieren (Rafaeli & La Rose, 1993). Zudem können bei vielen im Internet zur Kommunikation eingesetzten Techniken die Inhalte der Kommunikation dauerhaft archiviert und öffentlich zugänglich gemacht werden (z. B. bei Newsgroups, elektronischen Foren). Dies ermöglicht es jedem Beteiligten, auf einfache Weise alle relevanten Informationen über das Projekt abzurufen. Auch wenn bei der Erstellung von proprietärer Software in vielen Unternehmen ebenfalls elektronische Medien eingesetzt werden, so geschieht dies häufig in einem geringeren Umfang. Die Mehrzahl von Entscheidungen wird in persönlichen Gesprächen getroffen, die weder sinnvoll archiviert werden können noch sollen. Die Anzahl der TeilnehmerInnen an Entscheidungen ist begrenzt. Eine zu umfangreiche Beteiligung an Entscheidungen oder die Freigabe von Informationen widerspricht unternehmenspolitischen Interessen. Dies hat einen Einfluss auf die *Partizipationsmöglichkeiten* (vgl. Kapitel 4.3.1). Open-Source-Projekte sollten über umfangreichere Partizipationsmöglichkeiten verfügen als proprietäre Softwareentwicklung.

Ein weiterer Aspekt, der durch die umfassende Verfügbarkeit von Informationen und die Möglichkeit, diese zu kommentieren, beeinflusst wird, ist die *Rückmeldung von anderen Personen* (vgl. Kapitel 4.4.1). Der Beitrag von Rückmeldungen gehört zu den Verhaltensnormen innerhalb der Open-Source-Gemeinschaft. Dabei sind sowohl bestärkende als auch kritische Rückmeldungen gefordert. Die Verhaltensregeln fordern, auf "demotivierende" Rückmeldungen (Beschimpfungen, Beschuldigungen oder Aufforderungen zu Verbesserungen, Shah, 2003) zu verzichten. Das Feedback in Open-Source-Projekten hat für die TeilnehmerInnen einen hohen Stellenwert und ist eine der wichtigsten Motivationsquellen für die Teilnahme an Projekten. Für 27 % der individuellen TeilnehmerInnen an Open-Source-Projekten war die Rückmeldung einer ihrer vier wichtigsten Gründe für ihr Engagement (Ghosh et al., 2002). Von Firmen, die sich an Open-Source-Projekten beteiligten, gaben 41 % an, dass die Nützlichkeit der Rückmeldungen für sie der wichtigste Grund für eine Veröffentlichung von Code ist (Bonaccorsi & Rossi, 2004).

Es gibt zwei große Gruppen von Personen, die Rückmeldung geben: die aktiven NutzerInnen der Software (vgl. Kapitel 3.2) und die Entwickler. Die Art der Rückmeldung unterscheidet sich zwischen diesen beiden Gruppen. Die aktiven NutzerInnen geben Rückmeldung über Funktionsfehler (bug reports) und vermisste Funktionen. Die

Rückmeldung der Entwickler betrifft nicht nur sichtbare Fehler in der Funktion, sondern auch Fehler und Mängel, die sie beim Studium des Quellcodes erkennen. Diese Form wird in der Open-Source-Softwareentwicklung systematisch genutzt und als Peer-Review bezeichnet (vgl. Kapitel 3.2.3). Die Rückmeldungen beschränken sich oft auf direkte Kommentare zu erstellten Quellecodeteilen. Auf Fragen zum Design ist es schwer, Rückmeldungen von der Entwicklergemeinschaft zu erhalten (Jorgensen, 2001). Ein weiterer problematischer Punkt der Rückmeldung in Open-Source-Projekten ist, dass bei einzelnen leitenden Entwicklern sehr häufig eine Rückmeldung angefragt oder abgegeben wird. Dies führt zu einem hohen Zeitaufwand, der für die Beantwortung aufgewendet werden muss (Peyrache, Cremer & Tirole, 2000).

Auch in der proprietären Softwareentwicklung gibt es Rückmeldungen von NutzerInnen zu Fehlern. Die Möglichkeit zu Rückmeldungen wird in der proprietären Softwareentwicklung u. a. durch die begrenzte Verfügbarkeit des Quellcodes eingeschränkt. Es bleibt einem kleinen Kreis von Entwicklern vorbehalten, Einblick in den Quellcode zu nehmen und Rückmeldungen abzugeben (Franck & Jungwirth, 2002a).

3.2.3 Peer-Review

Open-Source-Projekte nutzen einen Mechanismus zur Qualitätssicherung, der eine große Verbreitung in allen akademischen/wissenschaftlichen Fachbereichen hat, den Peer-Review. Insbesondere vor der Veröffentlichung von Zeitschriften werden Artikel von einem oder mehreren Experten des entsprechenden Fachbereichs begutachtet.

In Open-Source-Projekten wird Peer-Review intensiv genutzt, um Fehler und Konstruktionsmängel aufzudecken. Beim Peer-Review wird durch eine oder mehrere Personen der Quellcode eines Entwicklers gesichtet, um Mängel zu identifizieren (Ambati & Kishore, 2004) und damit die Leistungsfähigkeit der Software zu verbessern (Krogh, Späth & Lakhani, 2003b). Das Verfahren ist vergleichbar mit dem Peer-Review, wie es im wissenschaftlichen Bereich zur Bewertung und Überarbeitung von Beiträgen üblich ist. Da durch das Peer-Review-Verfahren sehr effektiv Fehler erkannt und behoben werden können, wird hierdurch ein signifikanter Beitrag zur Qualität der Software geleistet (Fagan, 1976). Der intensive Einsatz von Peer-Reviews in der Open-Source-Softwareentwicklung gilt als einer der Vorteile gegenüber der proprietären Softwareentwicklung (Stark, 2002).

In der proprietären Softwareentwicklung werden unterschiedliche Kontrollverfahren zur Qualitätssicherung eingesetzt (Lehman, 1979; Selig, 1986). Ein Verfahren, das in Intensität und Umfang mit dem Peer-Review in Open-Source-Projekten vergleichbar ist, findet jedoch selten praktische Anwendung (Barnett & Schwaber, 2004; Cubranic, 1999b). So ist z. B. in einem einflussreichen Prozessmodell zur Beurteilung der Qualität des Software-

prozesses, dem "Capability Maturity Model" (CMM), die Inspektion des Arbeitsproduktes explizit vorgesehen. In der überarbeiteten Fassung, dem "Capability Maturity Model Integration", wird dieser Prozess sogar als "Peer-Review" bezeichnet. Auch andere Modelle der Softwareentwicklung sehen eine wechselseitige Kontrolle des Quellcodes vor. Auch in den agilen Softwareentwicklungsmodellen gibt es Peer-Review-Verfahren. Ein besonderes Beispiel hierfür ist die im Extreme Programming übliche Vorgehensweise des paarweisen Programmierens (Pair Programming). Dabei teilen sich zwei Programmierer einen Rechner (Barnett & Schwaber, 2004). Nur einer der beiden bedient den Rechner, der Zweite beobachtet die Arbeit. In regelmäßigen Abständen werden die Rollen getauscht.

Im Unterschied zum Vorgehen in der proprietären Softwareentwicklung zeichnet sich das Peer-Review in Open-Source-Projekten durch eine größere Unabhängigkeit und Diversität der Reviewer aus. Die Unabhängigkeit stellt sich dadurch ein, dass das Peer-Review durch eine große Gruppe durchgeführt wird, die an der Erstellung nicht direkt beteiligt ist (Robbins, 2004). Dies macht die Überprüfung unvoreingenommener und damit genauer (Feller & Fitzgerald, 2000). In Open-Source-Projekten weist die Gruppe, die den Quellecode prüft, zudem eine größere Diversität auf, was eine Grundvoraussetzung für den Erfolg von Peer-Review ist (Freedman & Weinberg, 1990). Nach Einschätzung von Beobachtern sind die TeilnehmerInnen eines Open-Source-Reviews auch besonders motiviert und verfolgen keine eigenen Interessen (Johnson, 2001b); allerdings liegen hierzu keine empirischen Untersuchung vor.

Dass ein Peer-Review erfolgreicher ist als die Suche nach Fehlern in den eigenen Arbeitsprodukten wird im Rahmen der Methode des "Egoless-Programming" angedeutet (Weinberg, 2004). Das "Egoless-Programming" basiert auf der Theorie der kognitiven Dissonanz (Festinger, 1957) und den Erkenntnissen zum normgemäßen Verhalten von Weinberg (2004). So kann aus der Theorie der kognitiven Dissonanz geschlossen werden, dass Fehler in der eigenen Arbeit eine Dissonanz zur wahrgenommenen eigenen Kompetenz darstellen. Dieses kann vermieden werden, wenn man solche Fehler nicht entdeckt. Dies führt dazu, dass eigene Fehler weniger erfolgreich gefunden werden als fremde Fehler, da sie keine Gefahr für die Wahrnehmung der eigenen Kompetenz darstellen (Weinberg, 2004). Aus diesem Grund ist das gegenseitige Review eine der zentralen Komponenten des Egoless Programming (Brodbeck, 1994a). Wichtig ist dabei, dass durch weitreichende kommunikative Beziehungen zwischen den Review-TeilnehmerInnen ein soziales Umfeld geschaffen wird, das das Geben und Empfangen von Kritik erleichtert (Weinberg, 2004).

Peer-Review bewirkt eine Verbesserung der Softwarequalität auf zwei Arten: zum einen durch den Gruppendruck, welcher motiviert, Fehler zu vermeiden, und zum anderen durch die Fehler, die tatsächlich gefunden werden (Mills, 1971). Dass Peer-Review tat-

sächlich erfolgreich ist, wird von einigen Autoren vermutetet (z. B. Mockus et al., 2000; Wiegers, 2002) und scheint sich durch automatische Fehleranalysen zu bestätigen (Chelf, 2006, vgl. Tabelle 1). Fraglich bleibt allerdings, ob Peer-Review auch geeignet ist, um Mängel auf hoher Ebene der Softwarearchitektur und Fehler in extrem selten genutzten Codeabschnitten zu erkennen (Johnson, 2001b).

Tabelle 1: Fehlerhäufigkeit in proprietären und Open-Source-Projekten

Softwaretyp	Fehler/1000 Zeilen
Industrie-Durchschnitt	15-50*
Microsoft-Anwendungen (Testphase)	10-20*
Microsoft-Anwendungen (ausgelieferte Version)	0,5*
Software mit Methode "Clean room Development"	0,1*
Open-Source-Projekte (allgemein)	0,05-1,24**
Open-Source-Projekte (Lamp-Softwarebündel)	0,19-0,47**

* (McConnell, 2004); ** Durch automatische Analyse ermittelt (Chelf, 2006)

Das Peer-Review bietet für alle ProjektteilnehmerInnen eine hervorragende Möglichkeit, ihre Programmierfähigkeiten zu verbessern (Moody, 2001; Raymond, 1999a; Wayner, 2000). Dabei werden alle erstellten Programmteile vor und nach der Veröffentlichung einer intensiven Kontrolle unterzogen und per persönlicher E-Mail, Forenbeiträgen etc. an den Autor zurückgemeldet. Die Rückmeldungen umfassen u. a. fehlerhafte Logik, schlechten Programmierstil oder die Verletzung von Konventionen (Lakhani & Wolf, 2003). Wie schon in Kapitel 3.2.1 besprochen, ist die Zahl der Rückmeldungen in Open-Source-Projekten sehr hoch. Da sowohl die Quelle als auch der Empfänger der Rückmeldungen Restriktionen nicht fürchten müssen (Cubranic, 1999b) und auch organisations- bzw. unternehmenspolitische Überlegungen nur eine untergeordnete Rolle spielen, bietet das Peer-Review-Verfahren ausgezeichnete Möglichkeiten auch für qualitativ hochwertige Rückmeldungen. Quantität und Qualität der Rückmeldungen zusammengenommen sollten für die Tätigkeit in Open-Source-Projekten ein hohes *Qualifikationspotenzial* (vgl. Kapitel 4.4.4) bedeuten.

3.2.4 Selbstorganisation

Open-Source-Projekte gelten bei einer Reihe von Autoren als Beispiele für eine selbstorganisierende Struktur (Adam et al., 2003; Crowston et al., 2004; Garzarelli, 2002; Koch, 2004; Madey et al., 2002). Allerdings verfügen sie über eine Reihe von Eigenschaften, die eher typisch für formale Organisationen sind, was sie zu einer hybriden Organisationsform macht (Lanzara & Morner, 2003).

Zu den mit formalen Organisationen vergleichbaren Charakteristika zählen Lanzara und Morner (2003) das Vorhandensein eines Kreises von Entwicklern mit geringer Fluktuation (Kernentwickler), definierte Führungsmechanismen für einige Entwicklungs- und Kommunikationsaufgaben, Stabilisierung und Selbstvervielfältigung durch automatische Dokumentationsprozesse und eindeutige Repräsentanten. Diese Punkte weisen große Differenzen in der Ausprägung in unterschiedlichen Open-Source-Projekten auf. So gibt es eine Reihe von Projekten, die durch einen "wohlwollenden Diktator" (z. B. Linux-Projekt, Shaikh & Cornford, 2003). Dieser verfügt üblicherweise über eine umfassende Kontrolle zu allen entscheidenden Fragen des Projektes. Die Führung muss nicht an eine Person gebunden bleiben, sondern die Rolle kann auch von verschiedenen Mitgliedern im Wechsel übernommen werden (rotating dictatorship; z. B. im Perl Projekt). Andere Projekte (z. B. im Apache-HTTP-Projekt) werden von einem Gremium gelenkt, das von allen Entwicklern frei gewählt wird (Markus, Manville & Agres, 2000).

Trotz der Gemeinsamkeiten mit formalen Organisationen gibt es eine Vielzahl von Eigenschaften, die Open-Source-Projekte von traditionellen Organisationsformen unterscheiden und den Vergleich mit selbst organisierenden Systemen nahelegen. Selbstorganisation ist dabei ein in Biologie, Physik und Chemie bekanntes Konzept, das alle Prozesse erfasst, "die aus einem System heraus von selbst entstehen und in diesem 'Selbst' Ordnung entstehen lassen, verbessern oder erhalten" (Probst, 1992). Die Charakteristika von Selbstorganisationen in sozialen Systemen sind Komplexität, Selbstreferenz, Redundanz und Autonomie (Probst, 1987a). Auf diese Punkte soll in den folgenden Absätzen eingegangen werden.

Komplexität weisen selbstorganisierende Systeme auf, "weil die resultierende Ordnung eine Konsequenz vieler interagierender Teile eines Netzwerkes ist" (Probst, 1987b). Dieses Netzwerk bildet in Open-Source-Projekten die virtuelle Gemeinschaft, deren Verhalten durch die Erhaltung und Pflege von Beziehungen und Interaktionen geprägt ist (vgl. Kapitel 3.2.1). Als Folge dieser Komplexität ist das Verhalten weder vollständig analytisch bestimmbar noch eindeutig vorhersagbar (Probst, 1992).

Im Gegensatz zu traditionellen Organisationen, bei denen die Ordnung durch Eingaben von außen entstehen soll (z. B. durch Führungshandlungen), entsteht in selbstreferenziellen Organisationen die Ordnung aus den eigenen Elementen: "[J]edes Verhalten wirkt auf sich selbst zurück und wird Ausgangspunkt für weiteres Verhalten." (Probst, 1987b) Als Charakteristika bzw. Konsequenz der Selbstreferenz führt Probst (1992) z. B. sinnvolle Aufgabenstellungen, teamorientierte Führung und Kooperation sowie Lernen an. Diese Eigenschaften von selbstorganisierenden Systemen bieten ein hohes Maß an *Qualifikationspotenzial* (vgl. Kapitel 4.3.6). So betrachtet Brodbeck (1994b) in der Softwareentwicklung einen hohen Grad an Selbstorganisation als Voraussetzung

für die Schaffung von Prozessen, die eine kollektive Nutzung von Informationen und gegenseitiges Lernen ermöglichen.

In redundanten Systemen ist jeder Beteiligte des Systems gleichzeitig auch ein Gestalter oder zumindest ein potenzieller Gestalter. Handeln kann jeder, der über Informationen verfügt und das muss nicht der Vorgesetzte sein wie bei einer Organisation nach dem Hierarchieprinzip (Probst, 1987b). Dies hat eine Dezentralisierung zur Folge, bei der Qualifikationen mehrfach innerhalb einer Organisation aufgebaut werden, und zwar mit entsprechender Qualifikationsvielfalt für die Beteiligten (Probst, 1992). Wie die Selbstreferenz bietet auch die Redundanz ein erhöhtes *Qualifikationspotenzial* (vgl. Kapitel 4.3.6). Da es keine Trennung mehr von Entscheidung und Ausführung gibt, steigt die *Aufgabengeschlossenheit* (vgl. Kapitel 4.3.3). In erster Linie eröffnet die Redundanz für die Beteiligten *Wahlmöglichkeiten* (vgl. Kapitel 4.3.1) bei der Ausübung der Tätigkeit. Als Gestalter haben sie die Möglichkeit, ihre eigenen Arbeitsabläufe zu bestimmen und verfügen damit über eine große *Autonomie* (vgl. Kapitel 4.3.1). Bei einer Selbstorganisation erschöpft sich die Möglichkeit des Gestaltens nicht bei der eigenen Tätigkeit, sondern sie gibt darüber hinaus die Option, an Entscheidungen mitzuwirken, die die gesamte Organisation betreffen (vgl. *Partizipation*, Kapitel 4.3.1).

Autonomie, im Sinne der Selbstorganisation, liegt dann vor, wenn "die Beziehung und Interaktion, die das System als Einheit definieren, nur das System selbst involvieren und keine anderen Systeme" (Probst, 1987a). Probst (1987a) geht dabei nicht von der vollkommenen Unabhängigkeit von allen Umwelteinflüsse aus, sondern nur von einer Autonomie in Bezug auf bestimmte Kriterien. Diese Kriterien sind die Selbstgestaltung und -lenkung (self governance) sowie die Selbstregulierung. Als Folge der Autonomie ergeben sich für die Beteiligten größere Handlungsspielräume, es existieren nur minimale Spezifikationen und das gesamte System ist nur lose gekoppelt. Auch wenn der Begriff der Autonomie von Probst (1987a) anders definiert wird als in der Organisationslehre üblich, so bietet eine Selbstorganisation (z. B. ein Open-Source-Projekt) Tätigkeiten mit sehr umfangreichen *Wahlmöglichkeiten* (vgl. Kapitel 4.3.1).

Erste verfügbare empirische Ergebnisse lassen darauf schließen, dass auch in Open-Source-Projekten in Teams gearbeitet wird (Hertel et al., 2003). So arbeiten im Linux-Kernel-Projekt etwa 60 % der Entwickler in einem Team und damit in einer Arbeitsform, die auch in der proprietären Softwareentwicklung weit verbreitet ist. Die Teams in Open-Source-Projekten sind wie die gesamte Organisation selbst organisiert und verfügen damit über wesentlich mehr Wahlmöglichkeiten, als dies bei traditionellen Teams der Fall ist. Es gibt auch im wirtschaftlichen Umfeld Versuche, den Autonomiegrad von Teams zu erhöhen. Die sogenannten "(teil)autonomen Arbeitsgruppen" zeichnen sich dadurch aus, dass sie eine funktionale Einheit innerhalb der regulären Organisationsstruktur bilden, deren Mitglieder konstant zusammenarbeiten und denen die Erstellung

eines kompletten (Teil-)Produkts mehr oder weniger eigenverantwortlich übertragen wurde. In autonomen Arbeitsgruppen werden neben der Ausführung der Tätigkeit Aufgaben der Organisation, Planung und Kontrolle übernommen (Bungard & Antoni, 1995). Typische Handlungsspielräume von teilautonomen Arbeitsgruppen sind dabei die Verantwortung für Termin und Qualität, Gestaltung von Arbeitsabläufen, Aufgabenverteilungen, Leistungsbewertung, Mitwirkung bei der Auswahl der Teammitglieder und Planung des Personleinsatzes, Planung und Durchführung von Qualifizierungen und Personalentwicklung (Kühl & Kullmann, 1999). Gemeinsam ist autonomen Arbeitsgruppen und Open-Source-Projekten, dass viele Aufgaben nicht von einer übergeordneten Instanz der Organisation wahrgenommen werden. Im Unterschied zu autonomen Arbeitsgruppen liegt in Open-Source-Projekten häufig mehr Verantwortung bei den einzelnen Entwicklern und nicht bei der Gruppe. So kann beispielsweise ein Entwickler im Open-Source-Bereich seine Arbeitsabläufe, seinen "Personaleinsatz" und Fragen der Qualifizierung ohne Abstimmung mit einer Gruppe frei gestalten.

Autonome Arbeitsgruppen führen zu humanerer Arbeit im Sinne von Reduktion einseitiger Belastungen, verbesserten sozialen Kontakten und einer höheren Qualifikation der Mitarbeiter (Ulich, 1998). Es wurden auch positive ökonomische Auswirkungen beobachtet: Produktqualität, Produktivität und Flexibilität steigen, Fehlzeiten und Fluktuation sinken (zusammenfassend: Ulich, 1998). Trotz dieser positiven Auswirkungen sind weder gruppenspezifische (z. B. autonome Arbeitsgruppen) noch individuelle Formen der Selbstorganisation verbreitet. Nach einer Befragung in 10 EU-Staaten können 8 % der Arbeitsplätze als gruppenspezifisch und 12 % als individuell selbst organisiert gelten (Sisson, 2000).

Im Vergleich zum industriellen Bereich ist der Anteil an selbst organisierten Arbeitsplätzen in der IT-Branche größer (Helfen & Krüger, 2002). Eine der ersten dokumentierten Einführungen von autonomen Arbeitsgruppen (self managed teams) im Bereich der Softwareentwicklung war die Einführung bei McDonnell Douglas im Jahr 1988 (Klepper, Litecky & Jones, 1989). Ziel der Einführung war es, einen kontinuierlichen Verbesserungsprozess anzustoßen, um Qualität, Quantität der Softwareentwicklung und die Mitarbeiterzufriedenheit zu erhöhen. Autonome Arbeitsgruppen mit sehr weitreichenden Befugnissen wurden bei Texaco Inc. eingeführt (Hirschheim & Miller, 1993). Die zentrale IT-Abteilung mit mehr als 800 Mitarbeitern bestand nach der Implementierung aus 131 Arbeitsgruppen. Jeder Mitarbeiter gehört mindestens zwei dieser Arbeitsgruppen an. Alle Arbeitsgruppen können sich ein eigenes Leitbild geben, innerhalb gegebener Grenzen Budgetentscheidungen selbstständig treffen und die Zuteilung von Ressourcen eigenverantwortlich vornehmen (z. B. Gehaltserhöhungen, Einkauf von Ausrüstung, Zuteilung von Büroräumen). Die Beurteilung der Projektmitglieder wird auf der Basis von selbst gewählten Bewertungsgrundsätzen eigenständig

vorgenommen. Das Team kann seinen Leiter frei wählen, der das Team dann auch im nächsthöheren Führungsteam vertritt (Hirschheim & Miller, 1993). In einer quantitativen Studie zu autonomen Arbeitsgruppen (self-directed work teams) in der Softwareentwicklung zeigte sich ein leicht positiver Zusammenhang zwischen dem Grad der Autonomie einer Arbeitsgruppe und den Ergebnissen der Gruppe (Janz, 1998).

Auch das Konzept des Egoless Programming (vgl. auch 3.2.3) enthält Aspekte der Selbstorganisation. Bei diesem Konzept wird ebenfalls auf eine Steuerung von außen verzichtet. Ein Entwicklerteam besteht aus 10 oder weniger Personen mit unterschiedlichen Fähigkeiten und Expertisen. Die Leitung wird durch die Personen übernommen, deren Fähigkeiten dem aktuellen Bedarf am besten entsprechen; bei neuen Anforderungen wird die Leitung von einer besser qualifizierten Person übernommen. Innerhalb eines Teams gibt es keine Hierarchie, entsprechend existiert auch kein Top-down- oder Bottom-up-Kommunikationsfluss, sondern jedes Teammitglied steht in Kontakt mit allen anderen Teammitgliedern.

Die Struktur des Egoless Programming sollte in quantitativer und qualitativer Hinsicht anderen Strukturen der Softwareerstellung überlegen sein. Eine Einschränkung in der Anwendbarkeit auf unterschiedliche Aufgabentypen zeigt Mantei (1981) auf. Seine Kritik beruht allerdings auf Erkenntnissen zur Leistungsfähigkeit von dezentral gesteuerten Gruppen und erscheint daher nicht ausreichend belegt. So sollen Aufgaben mit hohem Kommunikationsbedarf für dezentrale Gruppen nicht geeignet sein, weil in solchen Gruppen ein größerer Bedarf an Kommunikation entsteht, der zu Verzögerungen im Projekt führen kann. In hierarchischen Organisationen können trotz geringerem Kommunikationsaufkommen Engpässe dadurch entstehen, dass höhere Ebenen in der Hierarchie einen "Flaschenhals" darstellen und mehr Informationen aufnehmen und verteilen müssen, als es ihre Kapazität erlaubt. Solche Engpässe können ebenfalls zu Verzögerungen führen. Eine differenziertere Betrachtung ist an dieser Stelle notwendig. Obwohl Weinberg sein Konzept des Egoless Programming bereits vor über 30 Jahren vorgestellt hat, führen Vorbehalte in Unternehmen nur in Ausnahmefällen zu einer Anwendung.

Die Möglichkeit zur Gestaltung innerhalb einer Organisation bzw. in einem Team stellt ein wichtiges Merkmal einer Tätigkeit dar. Für die Selbstorganisation stehen dabei theoretische Konstrukte von Probst (1987a) zur Verfügung, während die autonome Organisation in Teams mit teilautonomen Arbeitsgruppen in vielen Unternehmen zum Einsatz kommt. In beiden Bereichen bieten Open-Source-Projekte eine besonders ausgeprägte Gestaltungsmöglichkeit. Sowohl in Bezug auf die Gesamtorganisation als auch in Bezug auf die Autonomie in Teams sind die Befugnisse der einzelnen Entwickler ausgesprochen weitgehend, sodass Open-Source-Projekte als selbst organisiert und autonom betrachtet werden können. Dies sollte, wie oben bereits aufgeführt, Auswirkungen auf

das *Qualifikationspotenzial* (vgl. Kapitel 4.3.6), die *Aufgabengeschlossenheit* (vgl. Kapitel 4.3.3) und *Wahlmöglichkeiten* (vgl. Kapitel 4.3.1) haben.

3.2.5 Selbstselektion

Wie zuvor gezeigt, führt die Selbstorganisation in Open-Source-Projekten für die TeilnehmerInnen zu deutlich mehr Freiheitsgraden, als dies üblicherweise in traditionellen Organisationen und damit in proprietären Softwareentwicklungsprojekten der Fall ist. Diese Freiheiten ermöglichen es den Mitgliedern, Arbeitsabfolge und -methoden frei zu bestimmen. Neben der Entscheidung, *wie* die Tätigkeit ausgeführt wird, gibt es in Open-Source-Projekten für einen großen Teil der Entwickler die freie Entscheidung, *was* sie machen (Elliott & Scacchi, 2002; Elliott & Scacchi, 2004; Holck & Jorgensen, 2004). Dieser Vorgang wird in der Open-Source-Literatur als Selbstselektion (self-selection, Garzarelli, 2002; Moody, 2001; Raymond, 1999a; Robbins, 2004) bezeichnet. Selbstselektion ist eine wichtige Voraussetzung für die Leistungsfähigkeit von Open-Source-Projekten (Raymond, 1999a). Wie in anderen Communities of Practice (vgl. auch Kapitel 3.2.1) kann man auch in Open-Source-Projekten davon ausgehen, dass die Transaktionskosten für das Zuordnen von Personen zu Aufgaben gegenüber hierarchischen- oder marktorientierten Ansätzen deutlich geringer ausfallen (Benkler, 2002).

Der Prozess der Selbstselektion wurde in Open-Source-Projekten noch nicht untersucht. Es wird vermutet, dass Entwickler Tätigkeiten aus Arbeitsgebieten wählen, mit denen sie bereits vertraut sind und bei denen ihnen die Entwicklungsmethoden bekannt sind (Robbins, 2004). Rossi (2004) geht davon aus, dass Entwickler die Aufgaben auswählen, die ihren Fähigkeiten am besten entsprechen. Aus einer empirischen Studie zur Apache User Group geht jedoch hervor, dass aktive Mitglieder der User Group mit ausreichend Fähigkeiten zur direkten Modifikation der Software diese auch einsetzten (Franke & Hippel, 2003). Entwickler mit weniger ausgeprägten Programmierfähigkeiten beteiligten sich nicht an dieser Aufgabe. Dabei gab es keinen Unterschied zwischen den Anforderungen und Bedürfnissen an die Software zwischen diesen beiden Gruppen.

Die Auswahl der Tätigkeiten ist für alle Entwickler freigestellt, die sich aus eigener Motivation an der Open-Source-Entwicklung beteiligen. Obwohl einige Autoren Open-Source als eine Bewegung von unbezahlten Freiwilligen gesehen haben (Raymond, 1999b), ist mittlerweile der Anteil bezahlter Entwickler nicht unerheblich. In Abhängigkeit von der Stichprobe und Definition von Bezahlung schwankt der Anteil an Personen, die für ihre Entwicklungstätigkeit bezahlt werden, zwischen 43 und 54 % (vgl. Tabelle 2). Für das Linux-Kernel-Projekt gehen Schätzung davon aus, dass mehr als 70 % des Quellcodes von Entwicklern beigesteuert werden, die dafür bezahlt werden (Kroah-Hartman, Corbet & McPherson, 2009).

Tabelle 2: Empirische Befunde zur Entlohnung in Open-Source-Projekten

Projekt/e	Geldleistung	Anmerkungen
Diverse (David et al., 2003)	57 % erhalten *keine* Geldleistung	Beinhaltet nicht nur Geldleistungen für Entwicklungstätigkeit, sondern auch für Tätigkeiten, die nur in Zusammenhang mit Open-Source-Software stehen (z. B. Wartung eines Open-Source-Programms)
Diverse (Ghosh et al., 2002)	46 % erhalten *keine* Geldleistung	Beinhaltet nicht nur Geldleistungen für Entwicklungstätigkeit, sondern auch für Tätigkeiten, die nur in Zusammenhang mit Open-Source-Software stehen (z. B. Wartung eines Open-Source-Programms)
Diverse (Hars & Ou, 2002)	45 % erhalten Geldleistung	Davon 16 % durch direkte Bezahlung und 29 % durch ein Entgelt
Linux-Kernel (Hertel et al., 2003)	43 % erhalten Geldleistung	Davon 20 % regelmäßig und 23 % ab und zu
FreeBSD (Jorgensen, 2001)	43 % erhalten Geldleistung	
Diverse (Lakhani & Wolf, 2003)	40 % erhalten Geldleistung	13 % erhielten Direktzahlungen und 38 % ein Entgelt

Die Formen der Geldleistungen sind sehr unterschiedlich und resultieren in einem unterschiedlichen Ausmaß an Freiheit bei der Auswahl der Tätigkeiten. Zum einen gibt es Entwickler, die bei einem Unternehmen angestellt sind und teilweise oder ausschließlich mit der Entwicklung von Open-Source-Software beschäftigt sind (z. B. Linus Torvalds, der zwischen 1997 und 2003 für das Unternehmen Transmeta arbeitete und sich weiter mit Linux beschäftigte: Linus Torvalds, 2009). In anderen Fällen erhalten selbstständige Entwickler den Auftrag, gegen Bezahlung bestimmte Funktionen in eine Software zu implementieren bzw. Fehler zu korrigieren (z. B. sind in der Liste der Firmen, die zum Linux Kernel beigetragen haben, freischaffende Entwickler (Consultants) mit einen Beitrag von 2,5% gelistet: Kroah-Hartman, Corbet & McPherson, 2009). Es gibt auch Entwickler, die von einer Stiftung oder einem Verein bezahlt werden, die/der das Open-Source-Projekt fördert (z. B. ist Linus Torvalds seit 2003 beim Open-Source Development Labs (OSDL), einer Non-Profit-Organisation zur Förderung von Linux im Unternehmensbereich angestellt: Linus Torvalds, 2009).

In Organisationen wird üblicherweise die Zuordnung von Personen zu Aufgaben auf zwei Ebenen vorgenommen. Die erste Ebene ist die Einstellung der Person, bei der über die Aufnahme in die Organisation entschieden wird. Hierbei werden Wissen, Fertigkeiten und Fähigkeiten der Bewerber mit den Anforderungen der Stelle verglichen und von den Mitgliedern der Organisation wird eine Entscheidung über die Aufnahme getroffen. Selbstselektion findet bei der Personalauswahl nur einseitig statt, wenn sich potenzielle Bewerber auf Basis der vorhandenen Informationen nicht bewerben bzw.

wenn sie aufgrund von neuen Informationen, die sie z. B. im Bewerbungsverfahren gewonnen haben, die Bewerbung zurückziehen. Die zweite Ebene der Selbstselektion stellt die konkrete Zuordnung von Mitarbeitern zu Aufgaben dar. Dies wird üblicherweise durch den Vorgesetzten oder aber, bei relativ autonomen Gruppen, durch die Gruppe selbst vorgenommen.

Eine Eignungsfeststellung wird nie bei allen Mitarbeitern zu einer optimalen Übereinstimmung von Fähigkeiten und Eigenschaften von Person und Position führen. Aus diesem Grund muss eine auf Effizienz ausgerichtete Organisation auf die individuellen Unterschiede der Organisationsmitglieder reagieren. Obwohl schon seit den 1930er Jahren die Forderung besteht, dass den Mitarbeitern "die Wahl der Arbeitsmittel und Arbeitsmethoden in möglichst hohem Grade überlassen bleiben soll" (Lipman, 1932, zitiert nach Ulich, 1998) und seit geraumer Zeit empirische Befunde zum Erfolg differenzieller Arbeitsgestaltung vorliegen, ist die Umsetzung im betrieblichen Alltag kaum anzutreffen. Nach dem Prinzip der differenziellen Arbeitsgestaltung werden jedem Beschäftigten unterschiedliche Arbeitsstrukturen angeboten, aus denen er wählen kann. Damit die Strukturen auch nach Lernfortschritt und Persönlichkeitsentwicklung den Anforderungen des Beschäftigten entsprechen, wird die differenzielle Arbeitsgestaltung durch das Prinzip der dynamischen Arbeitsgestaltung ergänzt. Diese beinhaltet die Möglichkeit zur Erweiterung bestehender und zur Schaffung von neuen Arbeitsstrukturen entsprechend der Entwicklung des Beschäftigten (Ulich, 1978, 1983, 1990). In Untersuchungen, z. B. bei der Montage von Motoren (Triebe, 1980, 1981) oder bei der Produktion von elektronischen Elementen (Vogt, Hofmann & Zülch, 2000; Zülch & Starringer, 1984) wurde gezeigt, dass bei ausreichender Freiheit in der Arbeitsgestaltung interindividuell unterschiedliche Vorgehensweisen möglich sind und diese unterschiedlichen Vorgehensweisen durchaus eine vergleichbare Effektivität und Effizienz aufweisen können. In anderen Untersuchungen im Bereich der Bildschirmtätigkeit konnte gezeigt werden, dass selbst gestaltete Tätigkeiten effizienter sind als vorgegebene, vermeintlich "optimale" Arbeitsabläufe (Bildschirmtätigkeit: Ackermann, 1986; Ulich, 1987). Dass differenzielle Arbeitsgestaltung nicht auf den Produktionsbereich beschränkt ist, schließt Zink (1978) aus seinen Untersuchungen: "[D]ie Übertragbarkeit auf den Verwaltungsbereich in Organisation ist dagegen weitgehend uneingeschränkt möglich."

Insbesondere durch die dynamische Natur der Branche und die häufigen Veränderungen in der Organisationsstruktur sind im Bereich der Informationstechnologien Zuordnungen von Mitarbeitern zu Aufgaben besonders schwierig. Das Ergebnis dieses Zuweisungsprozesses entspricht häufig nicht den Bedürfnissen der Mitarbeiter. So zeigt eine Untersuchung von Nelson (1996), dass bei Mitarbeitern aus der IT-Branche die Erwartungen an die Tätigkeit signifikant von den tatsächlichen Eigenschaften abweichen. Die gewünschten Tätigkeiten sollten z. B. größere Herausforderungen, einen stärkeren

Zusammenhalt der Mitarbeiter, ein besseres Verhältnis zu Vorgesetzten und einen geringen Arbeitsumfang bieten.

In der proprietären Softwareentwicklung stellt die Personalauswahl einen wichtigen Schritt zur Zuweisung von Personen zu Aufgaben dar. Insbesondere in projektorientierten Organisationen können auch nach der Einstellung noch Personen unterschiedliche Aufgaben annehmen; hier gibt es schon eine, möglicherweise bedeutende Einschränkung auf die Tätigkeiten, die innerhalb des Unternehmens benötigt werden. Zum Zeitpunkt der Einstellung ist ein Teil der Informationen nur näherungsweise und in beschränktem Umfang verfügbar (Triebe & Ulich, 1977). Damit kann das Ergebnis der Personalauswahl auch nicht zu einen perfekten Fit zwischen Person und Job führen.

Eine Selektion von Aufgaben auf der Basis der Selbsteinschätzung der arbeitenden Person bietet die Möglichkeit, zu besseren Ergebnissen zu führen:

> *"[T]he process of self evaluation fits the belief that individuals are in the best position to assess since they have access to a large data base on their own successes and failures in their abilities." (Harrington, 1995)*

Auch bei einer Selbstselektion in Open-Source-Projekten kann bezweifelt werden, dass ein Individuum vor der Entscheidung, einem Projekt beizutreten oder eine Aufgabe zu übernehmen, alle Information zur Verfügung hat und verarbeiten kann, die notwendig sind, um die Schwierigkeit einer Aufgabe beurteilen zu können. Dass es in Open-Source-Projekten trotzdem möglich ist, Aufgaben mit der optimalen Schwierigkeit zu bearbeiten, liegt daran, dass bei einer irrtümlichen Wahl die Kosten für einen Wechsel von Projekt oder Aufgabe sehr gering sind. Die geringen Wechselkosten bieten zudem ein effektives Mittel, um einer weiteren Herausforderung der Selektion zu begegnen: der Veränderung. Da sich sowohl Anforderungen von Aufgaben und Tätigkeiten als auch persönliche Präferenzen verändern, bieten Open-Source-Projekte mit ihren einfachen Möglichkeiten zum Wechsel eine gute Möglichkeit, um eine hohe Person-Job-Kongruenz zu erreichen.

Bei den gegebenen Möglichkeiten zum einfachen Wechsel von Tätigkeiten in und zwischen Open-Source-Projekten sollten es Open-Source-Entwickler möglich sein, ein Projekt bzw. eine Tätigkeit zu finden, die ein optimales Maß an Beanspruchung bietet. Erste empirische Ergebnisse deuten darauf hin, dass die Herausforderungen in Open-Source-Projekten tatsächlich als optimal wahrgenommen werden (Luthiger, 2005). Es erscheint plausibel zu postulieren, dass eine solche Tätigkeit auch ein optimales Maß an *Anforderungs-* und *Aufgabenvielfalt* (vgl. Kapitel 4.3.4) bereitstellt. Des Weiteren sollte diese Tätigkeit auch ein optimales *Qualifikationspotenzial* (vgl. Kapitel 4.3.6) bieten.

3.2.6 Arbeitsorganisation

Die Arbeit in Open-Source-Projekten zeichnet sich dadurch aus, dass eine Vielzahl von Entwicklern an der Software parallel arbeitet. Damit diese gleichzeitige Arbeit möglich ist, müssen möglichst abgegrenzte Teilaufgaben (Module) geschaffen werden. Da keine formellen Mechanismen existieren, die steuern, wer zu welcher Zeit an welchem Modul arbeitet, ist es notwendig, die parallel bearbeiteten Bestandteile möglichst häufig zusammenzuführen. Nur so ist gewährleistet, dass alle Teile reibungslos zusammenarbeiten. Auf die Aspekte Parallelisierung der Aufgaben, häufige Veröffentlichung und Modularisierung soll nun näher eingegangen werden.

Die Parallelisierung von Aufgaben findet in der Open-Source-Softwareentwicklung auf zwei verschiedene Arten statt. Die erste Art bezieht sich darauf, dass in unterschiedlichen Phasen der Entwicklung, wie z. B. Design, Implementierung und Beseitigung von Fehlern, nicht nacheinander, sondern parallel gearbeitet wird (Feller & Fitzgerald, 2000). Dies steht im Gegensatz zu iterativen und traditionellen Phasenmodellen der Softwareentwicklung (vgl. Kapitel 3.1.1 und 3), entspricht aber dem Vorgehen, wie es für die agile Softwareentwicklung empfohlen wird (vgl. Kapitel 3.1.3). Die zweite Art der Parallelisierung ist das zeitgleiche Arbeiten von mehreren Personen an der Entwicklung einer Funktion. Solche Doppelarbeit wird, soweit wie möglich, in der proprietären Softwareentwicklung vermieden, ist aber in Open-Source-Projekten Bestandteil des normalen Vorgehens (Iannacci, 2003). Die Folge ist, dass mehr als eine Lösung vorgelegt wird, von der aber nur eine in das Endprodukt integriert werden kann. Die hieraus entstehende Wettbewerbssituation, bei der sich die bessere Lösung durchsetzen sollte, wird als eine der Ursachen für die hohe Qualität von Open-Source-Software angesehen (Raymond, 1999a; Rossi, 2004). Allerdings kann die parallele Entwicklung auch in Open-Source-Projekten nur bei kritischen Softwareabschnitten erwünscht sein. Viele Aufgaben erfahren durch die Erstellung multipler Lösungen nur eine unwesentliche Verbesserung der Qualität, aber verlangsamen den Projektfortschritt, was zu Frustration und verminderter Motivation bei den Entwicklern von abgelehnten Lösungen führen kann.

Um dieses Problem so gering wie möglich zu halten, werden möglichst häufig alle neuen Teile integriert und der Allgemeinheit zur Verfügung gestellt. Die Aktualisierungen sind häufig erst wenige Stunden oder sogar nur Minuten alt (Cubranic, 1999a; Ettrich, 2004). Wenn die Veränderungen einen gewissen, teilweise vorab festgelegten Umfang erreichen, kann die Projektleitung entscheiden, ein neues Release auszurufen. Mit einem Release wird Projektzwischenstand festgelegt. Dieser größtenteils willkürliche Akt führt dazu, dass das Projekt verstärkte Aufmerksamkeit erfährt und nun auch weniger aktive TeilnehmerInnen und NutzerInnen sich dafür entscheiden, die neue Version zu testen bzw. einzusetzen. Die häufige Veröffentlichung von neuen Versionen ist eine der Voraussetzungen für den Erfolg von Open-Source-Software (Raymond, 1999a) und kann

die Qualität des Endprodukts steigern (MacCormack, 2001; Rothfuss, 2002). Neben der Koordination der Tätigkeiten und der Vermeidung einer Integrationsphase erst zum Abschluss des Projekts (González-Barahona & Robles, 2003) hat die Veröffentlichung auch eine motivierende Funktion (Cubranic, 1999a; Jorgensen, 2001; Muffatto & Faldani, 2003). Mit der Veröffentlichung werden die Beiträge zum Gesamtprojekt einem größeren Kreis von Entwicklern und NutzerInnen zugänglich gemacht, was in der Regel zu schnellen und häufigen Rückmeldungen führt (Bonaccorsi & Rossi, 2003). Dabei sind zwei Quellen der Rückmeldung zu unterscheiden: die Rückmeldung durch die Tätigkeit selbst und die Rückmeldung durch andere Personen. Eine Software, die lauffähig ist und die beabsichtigten Funktionen erfüllt, gibt Entwicklern eine klare Information darüber, ob sie ihre Aufgabe erfüllt haben. Diese Form der Rückmeldung nennt sich *Rückmeldung durch die Tätigkeit* (vgl. Kapitel 4.3.2). Auch die *Rückmeldung durch andere Personen* (vgl. Kapitel 4.4.1) wird durch häufige Veröffentlichungen gefördert. Der Entwickler erhält Informationen über die Erfahrungen der NutzerInnen. Die NutzerInnen können Zustimmung, Lob und Dank ausdrücken, jedoch häufiger erstellen sie Fehlerberichte (bug reports). Für die Mitteilung von Fehlern existieren in vielen Projekten formalisierte Prozesse und unterstützende Werkzeuge. Dies soll sicherstellen, dass Fehler erfasst, dokumentiert und bearbeitet werden und stellt für die Entwickler eine umfangreiche Quelle von Rückmeldungen dar. Auch Unternehmen, die proprietäre Software entwickeln, nutzen die Möglichkeit, die einzelnen Beiträge der Entwickler häufig zusammenzustellen und zu einem lauffähigen Programm zu integrieren. Agile Methoden der Softwareentwicklung fordern dies sogar explizit (Koch, 2004). Zum Beispiel wird bei Microsoft mittlerweile einmal täglich ein neues lauffähiges Programm (build) erstellt (Rothfuss, 2002). Diese Praxis ist jedoch in Unternehmen weniger verbreitet als in Open-Source-Projekten. In einer Untersuchung von 150 proprietären Softwareprojekten wurde nur in einem Drittel aller Projekte von der Möglichkeit Gebrauch gemacht, täglich ein lauffähiges Programm zu erstellen (Cusumano, MacCormack, Kemerer & Crandall, 2003). Zudem werden diese neuen Versionen der Software häufig nur einem kleinen Kreis von Mitentwicklern und keinen oder nur sehr wenigen NutzerInnen zur Verfügung gestellt. Entsprechend geringer fällt damit der Umfang der Rückmeldung aus. Auch in Open-Source-Projekten geben Mitentwickler auf Basis der häufigen Veröffentlichungen Rückmeldungen, was als Peer-Review bezeichnet wird (s. u.). Die häufigen Veröffentlichungen leisten auch einen Beitrag zur höheren *Aufgabengeschlossenheit* (vgl. Kapitel 4.3.3) im Bereich der Open-Source-Softwareentwicklung, weil hierdurch das Endprodukt präsenter wird (Cubranic, 1999a) und es damit einfacher wird, den eigenen Beitrag im Endprodukt zu identifizieren.

Die wohl wichtigste Voraussetzung für eine effiziente parallele Entwicklung ist eine Unterteilung der Gesamtaufgabe in Teilaufgaben. Die Modularisierung stellt eine

extreme Form der Arbeitsteilung dar, bei der der Aufwand für die Koordination auf ein Minimum reduziert wird (Moon & Sproull, 2002). Die Modularisierung ermöglicht das parallele Arbeiten von mehreren Entwicklern an einer Aufgabe (Moon & Sproull, 2002; Muffatto & Faldani, 2003; Osterloh et al., in Druck). Die wichtigste Voraussetzung ist eine entsprechend sorgfältige Modularisierung der Gesamtaufgabe (Torvalds, 1999). Als Folge der Modularisierung können die Entwickler weitgehend unabhängig voneinander an einzelnen Teilen arbeiten (Moody, 2001), der Aufwand für die Koordination verringert sich (Moon & Sproull, 2002) und Konflikte zwischen einzelnen Programmteilen werden geringer (Parnas, 1972). In der proprietären Softwareentwicklung wird dem Problem der Doppelentwicklung durch eine umfangreiche Planung und Koordination begegnet. Beides findet in Open-Source-Projekten in deutlich geringerem Umfang statt (Ettrich, 2004). Die Annahme, dass Open-Source-Software sich durch eine größere Modularität von proprietärer Software unterscheidet (Barnett & Schwaber, 2004; Feller & Fitzgerald, 2002; Raymond, 1999a), wurde mehrfach empirisch überprüft, es konnte jedoch kein eindeutiges Ergebnis ermittelt werden. Paulson, Succi und Eberlein (2004) konnten die Hypothese der größeren Modularität von Open-Source-Software nicht bestätigen. Die Hypothese wurde an einer sehr kleinen Stichprobe (3 Open-Source-Projekte, 3 proprietäre Projekte) auf der Basis eines statistischen Kriteriums geprüft. MacCormack, Rusnak und Baldwin (2005) untersuchten, wie sich die Modularität des proprietär entwickelten Produktes "Netscape" nach der Freigabe und Weiterentwicklung als Open-Source-Software veränderte und stellten dabei eine zunehmende Modularisierung fest. Die geringen Möglichkeiten zum persönlichen Treffen und Austausch machten eine stärkere Modularisierung der Softwarestruktur notwendig (MacCormack et al., 2005). Bei einem Vergleich der Wartungsfähigkeit stellte sich Open-Source-Software als wartungsfreundlicher als proprietäre Software heraus. Der verwendete Index für die Wartungsfähigkeit enthielt als wichtigstes Kriterium die Modularisierung der Software (Samoladas & Stamelos, 2005).

Für den einzelnen Entwickler bedeutet eine größere Modularität eine geringe *Interdependenz* (vgl. Kapitel 4.4.3) und eine höhere *Autonomie* (vgl. Kapitel 4.3.1) bei der Ausführung seiner Tätigkeit (Kiesler & Cummings, 2002). Durch die klare Abgrenzung der Aufgabe wird es zudem für den Entwickler einfacher zu erkennen, welchen Teil er zum Gesamtprojekt beigetragen hat. Diese erhöhte Widererkennbarkeit der Teilaufgabe im Gesamtergebnis sollte zu einer höheren *Aufgabengeschlossenheit* (vgl. Kapitel 4.3.3) führen.

3.2.7 Zusammenfassung Open-Source-Softwareentwicklung

Wie im vorangegangenen Abschnitt gezeigt, steht eine Reihe der Eigenschaften des Open-Source-Entwicklungsmodells in direktem Zusammenhang mit den Merkmalen der Tätigkeit. In Tabelle 3 sind die gefundenen Merkmale zusammengefasst.

Tabelle 3: Identifizierte Tätigkeitsmerkmale in Open-Source-Projekten

Eigenschaft von Open-Source-Projekten	Beeinflusste Tätigkeitsmerkmale
Virtuelle Gemeinschaft	Rückmeldung durch andere Personen
	Partizipation
Peer-Review	Qualifikationspotenzial
Elektronisch vermittelte Kommunikation	Partizipation
	Soziale Unterstützung
Selbstorganisation	Qualifikationspotenzial
	Autonomie
	Aufgabengeschlossenheit
	Partizipation
Selbstselektion	Aufgabenvielfalt
	Anforderungsvielfalt
	Qualifikationspotenzial
Häufige Veröffentlichung	Rückmeldung durch andere Personen
	Rückmeldung durch die Tätigkeit
	Aufgabengeschlossenheit
Modularisierung	Interdependenz
	Autonomie
	Aufgabengeschlossenheit

Neben den bereits beschriebenen Merkmalen gibt es noch eines, das nicht direkt zu einer der hier diskutieren Eigenschaften von Open-Source-Projekten zugeordnet werden kann: die *Bedeutsamkeit der Aufgabe*. Die Bedeutsamkeit der Aufgabe ist von dem Einfluss der Tätigkeit auf andere Personen innerhalb und außerhalb der Organisation abhängig. Damit wird dieses Tätigkeitsmerkmal stärker durch die Eigenschaften des Produktes als durch die Art und Weise der Herstellung beeinflusst.

Produkte aus Open-Source-Projekten weisen eine Reihe von Eigenschaften aus, die sie von anderen Softwareprodukten und vielen sonstigen Produkten unterscheiden. Bemerkenswert ist, dass bei Open-Source-Projekten die "Tragödie der Allmende" nicht auftritt (Osterloh et al., 2004). Unter der "Tragödie der Allmende" versteht man, dass es bei der Nutzung bzw. der Erstellung eines öffentlichen Gutes zwangsläufig zu einer Übernutzung bzw. Unterversorgung kommen muss (Hardin, 1968). Betrachtet man die Bedingungen von Open-Source-Software-Lizenzen (vgl. auch Kapitel 2), wird sehr

schnell deutlich, dass es sich hier um ein öffentliches Gut handelt. So ist es jedem freigestellt, Open-Source-Software entsprechend des eigenen Bedarfs zu nutzen, ohne dass hierfür eine Gegenleistung erbracht werden muss. Software verbraucht sich nicht durch Nutzung, womit eine Übernutzung ausgeschlossen werden kann. Jedoch muss Software erstellt und der Allgemeinheit verfügbar gemacht werden und damit stellt sich die Frage, warum keine Unterversorgung auftritt. Die Situation stellt sich als "Soziales Dilemma" dar: Personen, die keinen Beitrag zur Entwicklung einer Software geleistet haben, können diese trotzdem nutzen und somit sollte auf den ersten Blick niemand ein Interesse daran haben, an der Erstellung mitzuwirken. Wenn jedoch niemand an der Erstellung mitwirkt, gibt es keine Software und niemand hat einen Nutzen (Dawes, 1980). Es gibt eine Reihe von möglichen Ursachen, warum ein soziales Dilemma in der Open-Source-Entwicklung vermieden werden kann bzw. warum es dort erst gar nicht auftritt. Bei den Ursachen lassen sich solche, die in extrinsischen Motiven begründet sind, von solchen unterscheiden, die auf intrinsischen Motiven beruhen.

Zu den extrinsischen Motiven gehört die Erstellung von Software, um den eigenen Bedarf zu befriedigen (Lerner & Tirole, 2002b), und die Steigerung der eigenen Reputation durch sichtbare Beiträge an einen Entwicklungsprojekt (Hars & Ou, 2002; Kollock & Smith, 1999; Markus et al., 2000). Zudem darf nicht vergessen werden, dass zumindest in den großen und bekannten Open-Source-Projekten die Mehrheit einen Lohn für ihre Mitarbeit an Open-Source-Projekten erhält (vgl. auch Kapitel 3.2.5). Zu den intrinsischen Motiven gehört der Spaß an der Tätigkeit selbst (Luthiger, 2004) bzw. die Überzeugung, zu einer "guten Sache" beizutragen (vgl. auch Kapitel 4.5.1).

Im folgenden Kapitel werden die identifizierten Tätigkeitsmerkmale detailliert beschrieben und aus vorhandenen Quellen Unterschiede zwischen Open-Source- und proprietärer Softwareentwicklung abgeleitet.

4 Tätigkeitsmerkmale in Open-Source- und proprietärer Softwareentwicklung

Das Merkmal Arbeitstätigkeit stellt für alle Arten von Tätigkeiten und Berufen eine der wichtigsten Möglichkeiten zur Beeinflussung der Arbeitsergebnisse und des Befindens der Mitarbeiter dar. Die Softwareentwicklung ist keine Ausnahme, sondern ein Bereich, in dem Arbeitsgestaltung besondere Aufmerksamkeit verdient: Zum Ersten kann die Gestaltung der Arbeit einen wesentlichen Einfluss auf die Zufriedenheit der Mitarbeiter haben. Zufriedene Mitarbeiter weisen weniger Fehltage und eine geringere Fluktuation auf. In Zeiten, in denen der Bedarf an Informatikern in vielen Bereichen größer ist als das Angebot, ist dies ein Vorteil bei der Anwerbung und Bindung von qualifizierten Mitarbeitern. Zum Zweiten hat Arbeitsgestaltung in gewissem Umfang einen positiven Einfluss auf die Leistung der Mitarbeiter (vgl. Kapitel 4.5.3). Die Softwareentwicklung ist trotz Bemühungen der Automatisierung und Standardisierung weiterhin ein personalintensiver Arbeitsbereich, dessen Leistungsfähigkeit eng mit der Leistung seiner Mitarbeiter verbunden ist. Zum Dritten hängt die Qualität der Arbeitsergebnisse mit der Gestaltung der Arbeit zusammen. Eine hohe Qualität sicherzustellen, ist bei immer komplexeren Softwareprodukten zunehmend schwierig. Mangelhafte Qualität bedroht in Zeiten von Viren und Würmern, eine unzureichende Datensicherheit kann zu Imageschäden führen. Im Bereich der Medizin- und Weltraumtechnik kann durch Fehler sogar die Gesundheit von Menschen gefährdet werden. Die Bedeutung der Arbeitsgestaltung für die Mitarbeiter in der IT-Branche zeigt sich z. B. darin, dass Herausforderung, Verantwortung und Arbeitsatmosphäre mehr als das Gehalt geschätzt werden (Meares & Sargent, 1999). Zudem gibt es Hinweise darauf, dass die Arbeitsgestaltung bei Tätigkeiten im Bereich der Softwareentwicklung einen größeren Einfluss auf die Arbeitszufriedenheit hat als in anderen Berufsfeldern (Carayon, Hoonakker, Marchand & Schwarz, 2003).

In Kapitel 4.1 sollen zuerst Unterschiede zwischen verschiedenen Arten von Tätigkeiten in der Softwareentwicklung veranschaulicht werden. Das folgende Kapitel 4.2 beschreibt publizierte Tätigkeitsanalysen im Bereich der Softwareentwicklung. Im Kapitel 4.3 werden die relevanten Merkmale der Komplexität der Arbeit beschrieben und in Kapitel 4.4 die sozialen Aspekte der Arbeit. Die organisationalen Kriterien sind das Thema des Kapitels 4.5.

4.1 Tätigkeit und Tätigkeitsfelder

Softwareentwickler (teilweise auch Softwarearchitekten genannt) beschäftigen sich mit der systematischen Herstellung von Computerprogrammen. Zu ihren Aufgaben zählt

nicht nur das Programmieren, sondern auch das Erarbeiten von Anforderungen, das Erstellen von Softwarearchitektur, die Planung der Umsetzung und der Vorbereitung des Testens der Software (Software-Entwickler, 2006). Häufig verfügen Softwareentwickler über ein abgeschlossenes Hochschul- oder Fachhochschulstudium.

Der Beruf des Softwareentwicklers besteht überwiegend aus intellektuell anspruchsvollen Tätigkeiten und kann als Prototyp für intellektuelle Arbeit angesehen werden (Brodbeck & Frese, 1994). Von traditionellen Arbeitsformen unterscheidet sich die Tätigkeit des Softwareentwicklers durch die geringe räumliche, zeitliche und inhaltliche Strukturierung (Wieland, Klemens, Scherrer, Timm & Krajewski, 2004), hohe Komplexität, wenig Routinetätigkeiten und hohe Dynamik (Brodbeck, 2001).

Die Kernaufgabe des Softwareentwicklers besteht in der Herstellung und Erweiterung von Software. Hierfür muss die jeweilige Aufgabe spezifiziert werden, die entwickelte Lösung codiert, das Ergebnis getestet und Fehler müssen bereinigt werden. Nach der Analyse von Brodbeck (1994b) verbringen Softwareentwickler 43 % ihrer Arbeitszeit mit diesen Tätigkeiten. Neben diesen "typischen Softwareentwicklungstätigkeiten" gibt es noch die "unterstützende Softwareentwicklungstätigkeiten". Hierzu gehören die Dokumentation, die Systempflege und die technische Unterstützung (insgesamt 12 % der Arbeitszeit). Tätigkeiten, die nicht der Kernaufgabe angehören, machen mit gut einem Drittel einen beachtlichen Teil der Arbeitszeit aus. Dazu gehören Sitzungen, Beratungen und Gespräche (21 %), Organisation (12 %), selbstständiger Wissenserwerb (6 %) und Sonstiges (6 %).

Neben der Kommunikation stellt die hohe Innovationsrate bei den Werkzeugen und Methoden der Softwareentwicklung die Anforderung der ständigen fachlichen Weiterbildung und Wissensakquise dar (Brodbeck, 1994b). Als Belastungen werden von den Entwicklern vornehmlich Zeitdruck, nicht geplanter Zusatzaufwand, Unterbrechungen und unzureichende Möglichkeiten zur Weiterbildung empfunden. Demgegenüber bietet die Tätigkeit im Projekt Ressourcen wie z. B. die freie Einteilung der Arbeitszeit, das kooperative Verhältnis der Kollegen und der hohe Anregungsgrad der Arbeit (Latniak & Gerlmaier, 2006).

Die typische Arbeitsform in der Softwareentwicklung ist die Projektarbeit. Diese erfordert ein hohes Maß an Kommunikation, da eine hohe Abhängigkeit der Teilaufgaben von unterschiedlichen Gruppen und Personen besteht und zumeist eine interdisziplinäre Zusammenarbeit notwendig ist (Brodbeck, 2001). Die Kommunikation zwischen Softwareentwicklern wird jedoch zunehmend durch das Ansteigen von räumlicher Distanz der Projektmitglieder erschwert. Die Entfernung wächst durch Trends wie Outsourcing und durch Unternehmen mit global verteilten Mitarbeitern (Herbsleb & Grinter, 1999). Insbesondere die nicht direkt arbeitsbezogene, informelle Kommunikation der Ent-

wickler wird bei zunehmender Distanz schwieriger (Herbsleb & Grinter, 1999). Die Folgen sind längere Entwicklungszeiten und ein höherer Personalaufwand. Einer der Gründe für diesen Produktivitätsverlust liegt in der geringeren Bereitschaft von Entwicklern, sich in Fällen von großen Arbeitsumfängen gegenseitig auszuhelfen (Herbsleb et al., 2001).

Das Aufgabenfeld eines Softwareentwicklers ist ausgesprochen vielfältig und komplex. Wie in anderen Berufen führt dies auch bei der Softwareentwicklung zu Spezialisierungen in bestimmten Teilbereichen. Diese Spezialisierung ist für die Tätigkeitsgestaltung relevant, da unterschiedliche Tätigkeitsfelder unterschiedliche Ansprüche stellen. So sind z. B. die Anforderungsanalyse (eine typische Aufgabe für Analysten) und das Beseitigen von Fehlern (eine typische Aufgabe für Programmierer) mit besonders hohen Belastungen verbunden (Fujigaki, 1990). In der Forschung zu den Tätigkeitsfeldern der IT-Branche wird häufig zwischen Programmierung, Analyse, Wartungstätigkeiten, Bedienpersonal, User Support und Führungstätigkeiten unterschieden (vgl. Tabelle 4).

Tabelle 4: Übersicht untersuchte Tätigkeitsfelder in Studien zur Arbeitsgestaltung in der Softwareentwicklung

	(Brodbeck & Frese, 1994)	(Couger & Zawacki, 1980)	(Goldstein, 1989)	(McKnight & Chervany, 1998)
Softwareentwickler	Softwareentwickler			
Programmierer		Programmer	Development Programmer	
Programmierer/Analysten		Programmer/Analyst		
Analysten		Analyst	Development Analyst	
Wartungstätigkeiten			Maintainer	
Benutzervertreter	Benutzervertreter			
Bedienpersonal für DV-Anlagen		Data Processing Operator		Critical System Operator
User Support			Supporter	
Führungskraft	Teilprojekt-/ Projektleiter	Manager		

Zu der Entwicklung von Software im engeren Sinne gehören der Analyst und der Programmierer. Der Schwerpunkt der Tätigkeit eines Analysten liegt auf der Spezifikation eines Computersystems und der des Programmierers auf der Umsetzung der Spezifikation. Ob die Tätigkeit des Analysten positivere Merkmale aufweist als die des

Programmierers, ist nicht eindeutig geklärt. Während in Studien von Couger und Zawacki (1980) Analysten durchgehend positivere Arbeitsbedingungen als Programmierer aufweisen, konnte Goldstein (1989) nur für die Rückmeldung durch die Tätigkeit einen signifikanten Unterschied zugunsten der Analysten finden. Eine mögliche Erklärung bieten die Unterschiede in den erfassten Tätigkeitsfeldern. So wird in der Studie von Couger und Zawacki (1980) nicht zwischen Wartungstätigkeiten und Entwicklungstätigkeiten unterschieden. Eine Unterscheidung zwischen Wartungstätigkeiten und Entwicklungstätigkeiten nimmt Goldstein (1989) vor und stellt fest, dass Programmierer, die viele Wartungstätigkeiten aufweisen, ihre Tätigkeit negativer bewerten als Analysten, die wenig Wartungstätigkeiten auszuführen haben. So könnten die schlechteren Ergebnisse der Programmierer durch einen im Vergleich zu den Analysten höheren Anteil an Personen mit Wartungsaufgaben hervorgerufen worden sein (Goldstein, 1989). Insbesondere bei den Merkmalen Anforderungsvielfalt, Autonomie und Bedeutsamkeit der Aufgabe wiesen Wartungsaufgaben niedrigere Werte als Entwicklungstätigkeiten auf.

Ob sich die Tätigkeit von Bedienpersonal (Operator) von der Tätigkeit der Entwickler in ihrem motivierenden Potenzial unterscheidet, ist nicht eindeutig zu klären. In der Studie von Couger und Zawacki (1980) wies das Bedienpersonal ein niedrigeres Gesamtmotivationspotenzial auf als alle 500 Tätigkeiten, die zu diesem Zeitpunkt in der Datenbank zum JDS von Hackman und Oldham erfasst waren. In einer Untersuchung von Bedienpersonal von kritischen Systemen ("critical system operator"), die 18 Jahre später veröffentlicht wurde, konnten diese niedrigen Werte dagegen nicht bestätigt werden (McKnight & Chervany, 1998). Für diese Untergruppe des Bedienpersonals wurden sogar höhere Werte als für Entwickler gefunden. Neben den unterschiedlichen Tätigkeiten, die beiden Studien zugrunde liegen, kann auch davon ausgegangen werden, dass sich die Tätigkeit des "Operators" in diesem Zeitraum stark verändert hat. So war eine wichtige Aufgabe eines Operators in den 1980er Jahren das "Füttern" des Großrechners mit Lochkarten, auf denen Programme und Daten in codierter Form vorlagen. Ein solche Aufgabe hat für Bedienpersonal zum Ende der 1990er Jahre keine Rolle mehr gespielt, da zu diesem Zeitpunkt Lochkarten bereits fast vollständig durch magnetische und optische Speichermedien ersetzt waren.

Wenig überraschend unterscheidet sich die Tätigkeit der Führungskräfte von anderen Softwareentwicklern dadurch, dass sie einen größeren Teil ihrer Zeit mit Organisation und Kommunikation verbringen (Brodbeck, 1994b). In allen vom JDS erfassten Merkmalsbereichen bewerteten Führungskräfte in der IT-Branche ihre Tätigkeit signifikant positiver als ihre Mitarbeiter. Auch im Vergleich zu Managern außerhalb der IT-Branche zeigten die Tätigkeiten der IT-Manager ein höheres Motivationspotenzial bei

allen Merkmalen, wobei alle Unterschiede, mit Ausnahme von Rückmeldung durch die Arbeit, signifikant sind (Couger & Zawacki, 1980).

4.2 Tätigkeitsanalysen in der Softwareentwicklung

Die systematische Analyse der Arbeitsgestaltung in der Softwareentwicklung hat eine beachtliche Anzahl von Untersuchungen hervorgebracht (vgl. Tabelle 5). Die angewendeten Verfahren wurden der psychologischen Forschung entnommen und gehen zumeist auf das Job Characteristics Model (JCM) von Hackman und Oldham (1976) zurück. Das JCM ist eines der einflussreichsten Modelle der Arbeitsgestaltung (Parker & Wall, 1998; Wall & Martin, 1994) und dominiert die Forschung in diesem Bereich (Morgeson & Campion, 2003). Das Modell postuliert fünf Tätigkeitsmerkmale (Anforderungsvielfalt, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe, Autonomie und Rückmeldung), die über medierende Konstrukte intrinsische Arbeitsmotivation, allgemeine Arbeitszufriedenheit, Zufriedenheit mit dem Wachstum und Effektivität beeinflussen (vgl. Abbildung 3).

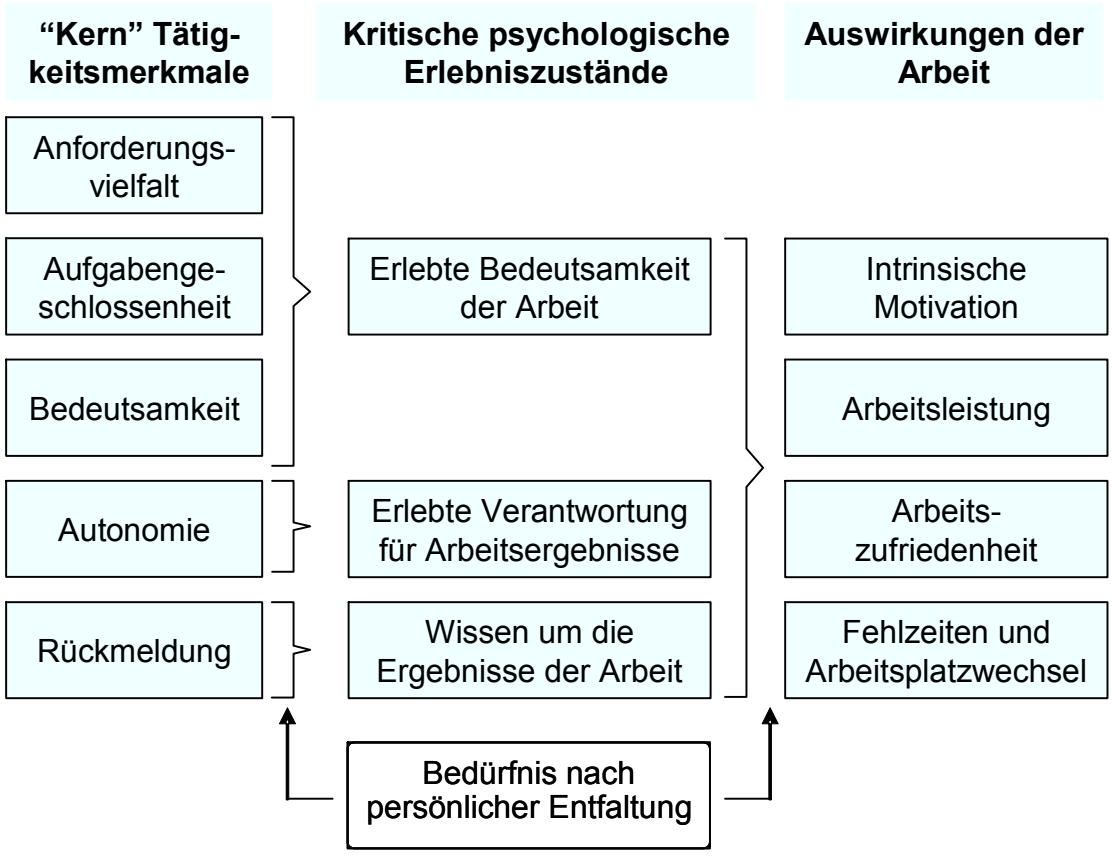

Abbildung 3: Job Characteristic Model (JCM) (übersetzt nach: Hackman & Oldham, 1975)

Die Auswahl der Tätigkeitsmerkmale des JCM wird unterstützt durch die Ergebnisse der handlungstheoretischen Ansätze. Unabhängig von der Forschung zur intrinsischen Motivation, aus der das JCM entsprungen ist, wurden durch handlungstheoretische Forschungsarbeiten im deutschsprachigen Raum ganz ähnliche Tätigkeitsmerkmale als bedeutsam hervorgehoben. So weist das Merkmal Aufgabengeschlossenheit z. B. erhebliche Überlappung mit dem Konstrukt "vollständige Tätigkeit" (Hacker, 1986) auf. Anforderungsvielfalt und Autonomie stimmen gut mit "Tätigkeitsspielraum" und "Entscheidungs- und Kontrollspielraum" überein (Ulich, Groskurth & Bruggemann, 1973). Aufgrund der Bedeutung des JCM wird in den folgenden Kapiteln häufig auf dieses Bezug genommen.

Zur Erhebung der Variablen des JCM dient der "Job Diagnostic Survey" (JDS). Bei einer Vielzahl von Studien in der proprietären Softwareentwicklung wurde der JDS oder eine geringfügig ergänzte Form verwendet (z. B. Carayon et al., 2003; Couger, Borovits & Zviran, 1989; Couger & Zawacki, 1980; Goldstein, 1989; Lending & Chervany, 1997; McKnight & Chervany, 1998). Im deutschsprachigen Raum kamen auch andere Instrumente zum Einsatz, so z. B. die Synthetische Belastungs- und Arbeitsanalyse (SynBA, Wieland-Eckelmann et al., 1996; SynBA, Wieland, Metz & Richter, 2002) in der Untersuchung von Wieland et al. (2004) oder die Salutogenetische subjektive Arbeitsanalyse (SALSA, Rimann & Udris, 1993, 1997) in den Untersuchungen von Klemens (2003) und Wieland et al. (2004). Eine detaillierte Analyse der Tätigkeiten von Brodbeck und Frese (1994) basiert auf der Analyse der Aufgabenstruktur in der Softwareentwicklung durch die Auswertung von 200 strukturierten Interviews und standardisierten Fragebögen. Tabelle 5 enthält eine Übersicht von Tätigkeitsanalyse im Bereich der Softwareentwicklung.

Tabelle 5: Beispiele für Studien mit Tätigkeitsanalysen in der Softwareentwicklung

Autoren	Verwendete/s Instrument/e	Erhobene Tätigkeitsmerkmale
(Brodbeck & Frese, 1994)	Eigenent-wicklung	Kommunikationsdichte und -intensität, Qualifikationsanforderungen
(Carayon et al., 2003)	JDS	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit
(Couger & Zawacki, 1980)	JDS	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit
(Couger et al., 1989)	JDS	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit
(Goldstein & Rockart, 1984)	JDS und andere	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit, Eigenschaften von Vorgesetzten und Kollegen
(Goldstein, 1989)	JDS und Eigenent-wicklung	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit, User Feedback, User Interaction
(Klemens, 2003)	SALSA	Ganzheitlichkeit, Qualifikationsanforderungen & Verantwortung, Überforderung, Unterforderung, Vorgesetztenverhalten, äußere Tätigkeits-bedingungen, Aufgabenvielfalt, Qualifikations-potenzial, Tätigkeitsspielraum, Partizipations-möglichkeiten, Gestaltungsmöglichkeiten, Spiel-raum für persönliche Dinge, Sozialklima, soziale Unterstützung
(Lending & Chervany, 1997)	JDS und andere	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit, Wachstum, Sicherheit, Umgang mit Anderen, Freundschaft, Feedback von anderen Personen
(McKnight & Chervany, 1998)	JDS	Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe, Aufgaben-geschlossenheit
(Wieland et al., 2004)	SynBA, SALSA	Ganzheitlichkeit, Qualifikationsanforderungen & Verantwortung, Überforderung, Unterforderung, Vorgesetztenverhalten, äußere Tätigkeits-bedingungen, Aufgabenvielfalt, Qualifikations-potenzial, Tätigkeitsspielraum, Partizipations-möglichkeiten, Gestaltungsmöglichkeiten, Spiel-raum für persönliche Dinge, Sozialklima, soziale Unterstützung

Die Ergebnisse dieser Studien zeigen, dass im Vergleich zu anderen Berufen die Tätigkeit des Softwareentwicklers im Allgemeinen motivations- und gesundheitsförderlich ist (vgl. Brodbeck & Frese, 1994; Carayon et al., 2003; Couger & Zawacki, 1980; Klemens, 2003; Lending & Chervany, 1997; McKnight & Chervany, 1998) und hohe Kommunikations- und Lernanforderungen bietet (Brodbeck, 1994b).

Die Arbeitsgestaltung in der proprietären Softwareentwicklung ist seit den Anfängen in den 1980er Jahren einem stetigen Wandel unterworfen. Es wird zunehmend versucht, die Tätigkeiten durch neue Werkzeuge zu unterstützen (CASE: computer-aided software engineering). Diese versprechen eine deutliche Erhöhung der Produktivität. So gibt es Schätzungen, die von einer Steigerung um 30 bis 100 % ausgehen (Inmon, 1988). CASE-Werkzeuge bewirken eine Veränderung der Entwicklungstätigkeit. Sie verlagern den Schwerpunkt von der Programmierungs- in die Spezifikationsphase. Ziel von CASE-Werkzeugen ist es, auf Basis der Spezifikation 80-90 % des Systems automatisch zu generieren (Pelley, Kappelman & Vanacek, 1992). Nach der Studie von Lending und Chervany (1998a) verbringen Entwickler, die CASE-Werkzeuge einsetzen, ca. 46 % ihrer Arbeitszeit mit Systemanalyse und -design, bei den sonstigen Entwicklern sind das nur ca. 26 %. Dafür entfallen auf Programmierung und Testen beim Einsatz von CASE-Werkzeugen nur noch ca. 21 % der Zeit, während es ohne CASE-Werkzeuge 39 % sind. Trotz einer stärkeren Standardisierung von Aufgaben nimmt das Gesamtmotivationspotenzial der Entwicklertätigkeit nicht ab. Es sind lediglich Verschiebungen zwischen einzelnen Gestaltungsmerkmalen zu beobachten (Lending & Chervany, 1997).

Aus dem Bereich der Open-Source-Softwareentwicklung sind dem Autor keine Untersuchungen bekannt, die Tätigkeitsmerkmale systematisch analysieren. Auch existieren nach Wissen des Autors keine Theorien zur Tätigkeitsgestaltung in Open-Source-Projekten. Diese Arbeit soll helfen, diese offenen Fragen zu beantworten. Als Basis sollen Studien zur proprietären Softwareentwicklung und sonstige Kenntnisse zur Tätigkeitsanalyse herangezogen werden. In methodischer Hinsicht kann man bei vielen der Tätigkeitsanalysen in der Softwareentwicklung die Beschränkung auf die fünf Tätigkeitsmerkmale Rückmeldung, Autonomie, Anforderungsvielfalt, Bedeutsamkeit der Aufgabe und Aufgabengeschlossenheit von Hackman und Oldham (1976) bemängeln. Dies führt zu der Vernachlässigung anderer wichtiger Merkmale wie z. B. Vielfalt der Tätigkeit und Partizipation (Oldham, 1996), die auch zu einer höheren Schwierigkeit und damit zu einer höheren mentalen Belastung der Tätigkeit beitragen. Diese Art von Tätigkeitsmerkmalen wird in der Gliederung von Morgeson und Campion (2003) zu den Merkmalen der Komplexität der Arbeit (job complexity) gezählt. Die zweite Gruppe in der Gliederung sind die Merkmale des sozialen Umfeldes einer Tätigkeit. Zu dieser Gruppe gehören die Merkmale Rückmeldung durch Personen und die soziale Unterstützung. Die dritte und letzte Gruppe sind die Eigenschaften der physischen Umgebung wie z. B. körperliche Anstrengung oder Ergonomie, die insbesondere bei physisch anspruchsvoller Arbeit relevant sind. Bei vornehmlich intellektuellen Tätigkeiten wie der Softwareentwicklung spielen diese eine untergeordnete Rolle und werden nicht weiter betrachtet.

4.3 Komplexität der Tätigkeit

Wie bei anderen geistigen Tätigkeiten ist auch bei der Softwareentwicklung die intellektuelle Herausforderung, im Gegensatz zur physischen Beanspruchung und dem sozialen Umfeld, der dominierende Merkmalsbereich (Glass, Vessey & Conger, 1992). Aus bisherigen Untersuchungen geht hervor, dass Tätigkeiten in der Softwareentwicklung eine hohe Komplexität aufweisen (z. B. Frese & Hesse, 1993). Die Komplexität wird von den Entwicklern geschätzt, denn sie erhöht den Stolz auf die eigene Arbeit (Frese & Gail, 1993 zitiert nach Frese & Hesse, 1993, S. 67). Zudem trägt sie zur Vermeidung des Burn-out-Syndroms bei (Sonnentag, Brodbeck, Heinbokel & Stolte, 1994).

In der Forschung zur Arbeitsgestaltung werden überwiegend Tätigkeitsmerkmale erhoben, die mit der Komplexität der Arbeit in Verbindung gebracht werden (Morgeson & Humphrey, 2003). Im einflussreichsten Modell der Arbeitsgestaltung, dem JCM (Hackman & Oldham, 1976), spiegeln alle fünf erfassten Tätigkeitsmerkmale primär die Komplexität der Arbeit wider (Aldag, Barr & Brief, 1981; Drasgow & Miller, 1982; Stone & Gueutal, 1985). In einer Metaanalyse fanden Loher et al. (1985). Hinweise, die diese Auffassung bestätigten. Auch bei empirischen Untersuchungen zu den Dimensionen des JDS wurde die Einfaktorlösung mit Komplexität der Tätigkeit ("job complexity") gefunden (Dunham, 1976). Bei anderen Instrumenten mit vergleichbaren Dimensionen, wie dem "Job Characteristics Inventory" (JCI), wurde die Kombination aller Skalen ebenfalls als Maß für die Komplexität verwendet (Ganzach, 1998). Diese Ergebnisse deuten darauf hin, dass Komplexität der bestimmende Faktor ist, und sie werfen die Frage auf, ob Mitarbeiter überhaupt detaillierte Merkmale (wie Autonomie oder Vielfalt) als Eigenschaften ihrer Tätigkeiten wahrnehmen und unterscheiden können.

Dass Mitarbeiter die untersuchten Merkmale jedoch differenziert wahrnehmen/unterscheiden, konnte in Untersuchungen gezeigt werden, die bei Faktorenanalysen des JDS Lösungen mit zwei, drei, vier bzw. fünf Faktoren fanden (Dunham, Aldag & Brief, 1977; Green, Armenakis, Marbert & Bedeian, 1979). Eine Ursache für das Problem, die theoretisch postulierten Faktoren empirisch nachzuweisen, könnte darin liegen, dass die Differenzierung der verschiedenen Tätigkeitsmerkmale eine kognitiv anspruchsvolle Tätigkeit darstellt, die nicht von allen Personen geleistet werden kann. So konnte in Stichproben von Führungskräften, jungen Personen und gut Ausgebildeten die Struktur der fünf Faktoren repliziert werden, während dies bei älteren und schlechter ausgebildeten Personen nicht gelang (Fried & Ferris, 1986). Eine weitere Ursache könnte die unzureichende psychometrische Qualität der Tätigkeitsanalysen sein. Für den JDS gibt es Hinweise, dass die teilweise negative Formulierung der Antwortalternativen eine korrekte Bearbeitung der Skalen erschwert. Bei einer überarbeiteten Version des JDS ohne negativ formulierte Antwortskalen wurden die fünf postulierten Faktoren gefunden

(Cordery & Sevastos, 1993; Idaszak & Drasgow, 1987). Auch die Tatsache, dass andere Arbeitsanalyseverfahren wie z. B. der JCI eine Vielzahl an Dimensionen aufweisen, die sich über die Zeit und bei verschiedenen Stichproben als stabil erwiesen (Griffin, 1981; Griffin, Moorhead, Johnson & Chonko, 1980), deutet darauf hin, dass eine Unterscheidung verschiedener Merkmale der Komplexität möglich ist.

In den theoretischen Überlegungen zum Unterschied von Open-Source- und proprietärer Softwareentwicklung (vgl. Kapitel 3.2) sind eine Reihe von Tätigkeitsmerkmalen identifiziert worden, die dem Bereich der Komplexität der Arbeit zuzurechnen sind. Dazu gehören Autonomie, Partizipation, Rückmeldung durch die Tätigkeit, Aufgabengeschlossenheit, Anforderungsvielfalt, Aufgabenvielfalt, Bedeutsamkeit der Aufgabe und das Qualifikationspotenzial. Diese Merkmale werden im folgenden Kapitel detailliert besprochen.

4.3.1 Wahlmöglichkeiten

Die Wahlmöglichkeiten einer Person bei der Ausübung ihrer Tätigkeit werden unter verschiedenen Namen in der Arbeitsanalyse geführt (z. B. Autonomie, Handlungs-, Entscheidungs- und Gestaltungsspielraum, Freiheitsgrade Udris, 2001). Diese Konzepte unterscheiden sich hauptsächlich im Umfang der enthaltenen Aspekte (Hacker, 1986). In der vorliegenden Arbeit sollen zwei Aspekte der Wahlmöglichkeiten unterschieden werden: zum einen die Autonomie, die sich auf den Entscheidungs- und Gestaltungsspielraum des Mitarbeiters im Hinblick auf seine eigene Tätigkeit bezieht, zum anderen die Partizipation, die sich auf Aspekte der Organisation beziehen soll, die nicht im direkten Zusammenhang mit der eigenen Tätigkeit stehen.

Autonomie

Die Definition von Autonomie ist bei unterschiedlichen Forschern weitgehend identisch. Turner und Lawrence (1965) verstehen unter Autonomie den Ermessensspielraum, der von einer Person bei der Ausübung ihrer Tätigkeit erwartet wird. Er setzt sich zusammen aus dem Spielraum des Mitarbeiters über die einzusetzenden Methoden, die Abfolge der Tätigkeiten, der Arbeitsgeschwindigkeit, der Auswahl von Dienstleistungen und der Möglichkeit, Material zurückzuweisen. Ein wenig allgemeiner wird Autonomie von Hackman und Oldham (1976) beschrieben. Hier ist die Autonomie das Ausmaß an Freiraum und Unabhängigkeit, mit der ein Mitarbeiter die eigenen Aufgaben selbstständig planen, die notwendigen Arbeitsschritte festlegen und ausführen kann.

Auswirkungen von Autonomie gehen insbesondere aus Studien der Forschungstradition der soziotechnischen Systeme hervor. Dort wurden bei autonomen Gruppen positive Effekte auf die Mitarbeiterzufriedenheit (Trist, Susman & Brown, 1977), eine Erhöhung der Innovationsleistung und dadurch eine Reduktion der Kosten (Walton, 1977) be-

obachtet. Außerdem zeigte ein größerer Handlungsspielraum eine Leistungssteigerung (Pasmore, 1978), eine Verringerung von Fehlzeiten und Fluktuation (Walton, 1977), einen Rückgang der Unfallzahlen (Goodman, 1979; Walton, 1977) und verstärktes organisationales Commitment (Emery, 1959). Nicht alle diese Effekte konnten bestätigt werden, was z. T. den unzureichenden methodischen Standards einiger dieser Studien zuzurechnen ist (Parker & Wall, 1998).

Als methodisch gut abgesichert können die Ergebnisse der Metaanalyse von Fried und Ferris (1987) gelten. Autonomie zeigt eine hohe positive Korrelation mit der Zufriedenheit mit dem Wachstum ("growth satisfaction") und eine hohe negative Korrelation mit den Fehlzeiten auf. Von den untersuchten Tätigkeitsmerkmalen wies Autonomie sogar die stärksten Zusammenhänge mit diesen beiden organisationalen Kriterien auf (Algera, 1998). Auch in der Metaanalyse von Spector (1986) konnte der Zusammenhang von Zufriedenheit und Wachstum bestätigt werden. Zusätzlich zeigten sich noch mittlere Korrelationen mit Zufriedenheitsvariablen (allgemeine Zufriedenheit, Arbeitszufriedenheit und Zufriedenheit mit Vorgesetzten) und verschiedenen Verhaltensvariablen, u. a. auch der Leistung. Im Modell von Hackman und Oldham (1976) erhält die Autonomie neben der Rückmeldung den höchsten Stellenwert der Tätigkeitsmerkmale: sie wird bei der Berechnung des Motivationspotenzials einer Tätigkeit mit am höchsten gewichtet. Dies steht im Einklang mit neuen Forschungsrichtungen, die die Bedeutung der Selbstbestimmung betonen ("theory of planned behavior" und "cognitive evaluation theory", vgl. Behson, Eddy & Lorenzet, 2000).

Auf welche Art Autonomie und Tätigkeitsmerkmale im Allgemeinen die organisationalen Kriterien beeinflussen, ist noch weitgehend ungeklärt. Neben dem JCM, das empirisch noch nicht vollständig bestätigt werden konnte (s. o.), existieren nur wenige alternative Erklärungsansätze. In einer Untersuchung zur Autonomie von Maschinenbedienern postulierten Wall, Jackson und Davids (1992) zwei mögliche Ursachen: Zum Ersten mussten Maschinenbediener bei Störungen nicht mehr auf jemanden warten, der das Problem löst, sondern konnten es schnell beheben. Zum Zweiten konnten sie ein besseres Verständnis für die Ursache der Probleme entwickeln und damit Probleme antizipieren und verhindern. Für den Zusammenhang zwischen Autonomie und Stress stellt Frese (1989) drei mögliche Wirkungsmodelle vor: 1) Menschen haben ein Bedürfnis nach Autonomie und die Befriedigung dieses Bedürfnisses führt zu Wohlbefinden, 2) durch gesteigerte Autonomie ist es besser möglich, stressfördernde Arbeitsbedingungen zu vermeiden, 3) obwohl der Stress unverändert bleibt, können Menschen durch mehr Autonomie besser damit umgehen.

Tätigkeiten in der Softwareentwicklung weisen einen überdurchschnittlichen Handlungsspielraum auf (Klemens, 2003). Ein Vergleich mit anderen Berufsgruppen zeigt, dass Programmierer nur von Architekten, Naturwissenschaftlern, Elektroingenieuren und

Landwirten bei der Entscheidungsfreiheit ("decision latitude") übertroffen werden (Karasek, 1991). Einen besonderen Einfluss auf die Autonomie haben die eingesetzten Entwicklungsmethoden und -werkzeuge. Der Einsatz von strukturierten Entwicklungsmethoden hat u. a. eine Reduktion der Autonomie zur Folge (zusammenfassend: Baroudi & Michael, 1986). Bei einem Vergleich von unterschiedlichen Programmiertechniken wiesen Tätigkeiten, bei denen objektorientiertes Programmieren oder Prototyping zum Einsatz kamen, die geringste Autonomie auf. Einen positiven Einfluss hatte dagegen der Einsatz von Methoden, die die NutzerInnen integrieren (Lending & Chervany, 1997). Die gleiche Auswirkung hat der Einsatz von CASE-Werkzeugen (Lending, 1997); sie führen zu einer stärkeren Formalisierung der Tätigkeit und damit zu einer geringeren Autonomie (Lending & Chervany, 1998b). Diese negativen Folgen könnten die Ursache dafür sein, dass nur wenige Unternehmen CASE-Werkzeuge trotz der versprochenen Produktivitätsvorteile überhaupt einsetzen. Zudem nutzt in Unternehmen, die sich für einen Einsatz entschieden haben, nur ein geringer Anteil der Entwickler diese Werkzeuge tatsächlich (Lending, 1997; Lending & Chervany, 1997; Stolte & Heinbokel, 1993). Stolte und Heinbokel (1993) zeigen, dass der Einsatz von Werkzeugen die Autonomie nicht zwangsläufig reduziert. Bei Tätigkeiten mit hohem Handlungsspielraum war der Einsatz von fortgeschrittenen Werkzeugen besonders gut dazu geeignet, die Arbeit effizient zu unterstützen.

Nicht alle in der Softwareentwicklung Beschäftigten verfügen über dasselbe Ausmaß an Autonomie. Aus den bisherigen Untersuchungen geht hervor, dass Systementwickler, Systemanalysten und Führungskräfte über das größte Ausmaß an Autonomie verfügen. Programmierer verfügen über einen mittleren Freiraum in ihrer Tätigkeitsgestaltung und Bedientätigkeit sowie Wartungsarbeiten weisen den geringsten Entscheidungsspielraum auf (Couger & Zawacki, 1980; Hacker & Schoenfelder, 1986; Lending & Chervany, 1997).

Tätigkeiten in der Softwareentwicklung verfügen nicht nur über ein besonders hohes Ausmaß an Autonomie, sondern es gibt auch Hinweise darauf, dass Softwareentwickler in einem besonderen Maße von einer hohen Autonomie profitieren können: Bei komplexen Tätigkeiten, die eine hohe Beanspruchung darstellen, führt eine hohe Autonomie zu höherer Kreativität der Mitarbeiter (Shalley, Gilson & Blum, 2000). So reichten Mitarbeiter mit hoher Autonomie z. B. signifikant mehr Verbesserungsvorschläge ein als Mitarbeiter mit niedriger Autonomie (Hatcher, Ross & Collins, 1989). Ein weiterer Grund für die Notwendigkeit von Autonomie in der Softwareentwicklung ist der schnelle technologische Fortschritt, der große Unsicherheiten zur Folge hat. Auf diese können weder starre Vorschriften noch eine ausschließlich zentrale Entscheidungsfindung angemessen reagieren, sondern alle Mitarbeiter benötigen einen angemessen großen Entscheidungsspielraum (Böhler & Schatz, 1999; Parker & Wall, 1998; Wright & Cordery, 1999). Bei großer Variabilität von externen Quellen (z. B. Kundenanforderungen oder

Marktschwankungen), die sich direkt auf die Anforderungen der Mitarbeiter auswirken, sind autonome Arbeitsgruppen am besten geeignet (Cummings & Blumberg, 1987; Goodman, Devadas & Griffith-Hughson, 1988).

Open-Source-Projekte bestehen aus autonomen Arbeitsgruppen, ansonsten ist jedoch relativ wenig gesichertes Wissen über die Autonomie in Open-Source-Projekten bekannt. Nach Einschätzung von einigen Autoren unterscheiden sich Open-Source-Projekte im Grad der Autonomie (Markus et al., 2000). So schätzen Manville et al. (1999) z. B. den Autonomiegrad in Perl-Projekten als größer ein, als in Apache-, GNU-/Linux- oder Mozilla-Projekten. Die Computerwoche ("Software-Engineering – Auch freie Projekte folgen festen Regeln. Open-Source: Chaos oder strukturierte Entwicklung?," 2001) schreibt z. B. über das Apache-Projekt:

> *"Das Organisationsdiagramm von Apache vermittelt den Eindruck einer stark hierarchischen Struktur. Tatsächlich arbeiten die Projektteams aber weitgehend autonom."*

Insgesamt scheinen jedoch Open-Source-Projekte im Vergleich zu proprietärer Softwareentwicklung über einen hohen Grad an Autonomie und Partizipation zu verfügen (Noll & Scacchi, 1999; Osterloh et al., 2004). Ein hoher Grad an Autonomie ist ein wichtiges Merkmal von allen Formen der Freiwilligenarbeit. Da die Koordinatoren in der Freiwilligenarbeit nur ein geringes Ausmaß an Kontrolle über ihre Mitarbeiter haben, verfügen diese über die Freiheit, ihre Tätigkeit zu gestalten (Allen, 1987). Der Organisation kommt dabei die Rolle der "dankbaren Empfängerin" für die geleistete Arbeit der Freiwilligen zu (Allen, 1987). Weiterhin begünstigt wird das freiwillige Engagement in Organisationen durch eine geringe Standardisierung der Arbeitsabläufe (Hertel, Bretz & Moser, 2000). In Open-Source-Projekten wirkt sich auf die Autonomie auch noch positiv die räumliche Distanz der Entwickler (Moon & Sproull, 2002) und die erhebliche Dezentralisierung des Erstellungsprozesses (Peyrache et al., 2000) aus. So arbeiten in der Open-Source-Softwareentwicklung einzelne TeilnehmerInnen oder ganze Teams autonom an bestimmten Teilen des Projektes (Noll & Scacchi, 1999). Das selbstständige Arbeiten wird durch die starke Modularisierung und die damit einhergehende Unabhängigkeit von Teilprojekten unterstützt (Johnson, 2001a). Dabei haben die TeilnehmerInnen die Möglichkeit, ihre Aufgaben weitgehend selbstständig auszuwählen (Dietze, 2003a; Rossi, 2004), was die wahrgenommene Autonomie erhöht (Osterloh et al., in Druck). Neben der Auswahl der Tätigkeit sind Open-Source-Entwickler sowohl frei in der Auswahl von Methoden als auch in der Auswahl von Werkzeugen.

Bei den Methoden existieren für Open-Source-Entwickler gewöhnlich nur Richtlinien und Empfehlungen, deren Einhaltung aber nicht kontrolliert wird. Die einzige Kontroll-

möglichkeit, die ein Projekt ausüben kann, ist das Ablehnen des Arbeitsergebnisses (Holck & Jorgensen, 2004).

Bei der Auswahl der Werkzeuge haben die TeilnehmerInnen ebenfalls große Freiheiten. Oft sind Werkzeuge zur Verwaltung des Quellcodes ("Source Code Repositories"), für die automatische Kontrolle von Beiträgen ("Verification Machines") oder für die Nachverfolgung von Fehlern ("Bug Tracking Systems") vorgegeben (Holck & Jorgensen, 2004). Welche Werkzeuge bei der Erstellung des Quellcodes (z. B. die Entwicklungsumgebung) genutzt werden, bleibt den TeilnehmerInnen weitgehend überlassen. Ebenfalls für eine größere Autonomie in Open-Source-Projekten spricht der eingeschränkte Einsatz von strukturierten Entwicklungsmethoden: Die Nutzung von solchen führt zu einem geringen Grad an Autonomie (Goldstein, 1982; Kraft, 1977).

Die TeilnehmerInnen an Open-Source-Projekten haben nicht nur Autonomie in der Frage, welche Aufgaben sie wie erfüllen, sondern auch in der Frage, wann sie diese erfüllen. In Open-Source-Projekten existieren Zeitplanungen für das Gesamtprojekt, daraus resultieren aber keine festen Abgabetermine für die TeilnehmerInnen. Raymond (1999a) sieht darin eine wichtige Eigenschaft von Open-Source-Projekten. Die Abwesenheit von Zeitdruck kann zu einer höheren intrinsischen Motivation (Deci, Koestner & Ryan, 1999; Eisenberger, Pierce & Cameron, 1999) und zu einer Förderung der Kreativität (Stukas, Snyder & Clary, 1999) führen. Im Vergleich zu proprietärer Softwareentwicklung sollten Open-Source-Projekte auch einen geringeren Grad an Beeinflussung durch Personen außerhalb des Projektes aufweisen. So gehen von den Änderungsanträgen in kommerziellen Projekten 60 % auf externe Quellen zurück, 14 % gehen auf das Management oder andere Personen im Team zurück, während die Softwareentwickler nur für 25 % der Änderungen verantwortlich sind (Frese & Hesse, 1993).

Insgesamt geben diese Hinweise zu der Vermutung Anlass, dass Tätigkeiten in Open-Source-Projekten über ein größeres Ausmaß an Autonomie verfügen als Tätigkeiten in der proprietären Softwareentwicklung.

Partizipation

Der Bereich der Arbeitsgestaltung stellt eine Synthese sowohl aus Elementen der Arbeits- als auch Organisationspsychologie dar (Morgeson & Campion, 2003). Wie auch die Autonomie betrifft das Tätigkeitsmerkmal Partizipation die Wahlmöglichkeiten eines Mitarbeiters. Im Unterschied zur Autonomie liegt der Schwerpunkt der Partizipation nicht auf dem Gestaltungsspielraum beim Ausüben der persönlichen Tätigkeit, sondern auf der Möglichkeit, an Entscheidungen mitzuwirken, die noch weitere Personen (z. B. Vorgesetzte) betreffen (Haim, 2002; Spector, 1986). Irreführenderweise werden diese beiden Konstrukte häufig synonym verwendet (Haim, 2002) bzw. werden Aspekte der Partizipation mit denen der Autonomie vermischt (Spector, 1986). Zu den Partizipations-

möglichkeiten gehört z. B. die Möglichkeit der Einflussnahme auf Urlaubspläne, Arbeitszeit oder Anschaffungen (Semmer, Zapf & Dunckel, 1999).

In vielen Studien konnten positive Zusammenhänge zwischen Partizipation und Zufriedenheit und Leistung (Ives & Olson, 1984; Katz, Kochan & Weber, 1985; Locke, 1966; Locke & Schweiger, 1979; Lowin, 1968; Neider, 1980; Searfoss & Monczka, 1973; Tosi, 1970; Vroom, 1959), Produktqualität (Katz et al., 1985; Locke & Schweiger, 1979), Entscheidungsgüte (Locke & Schweiger, 1979), Erreichung von Zielen (Searfoss & Monczka, 1973), Identifikation (White & Ruh, 1973) und Motivation (Edwards, Scully & Brtek, 1999; White & Ruh, 1973) nachgewiesen werden. In einer Meta-Analyse konnte (Spector, 1986) diese Zusammenhänge bestätigen. Das Fehlen von Partizipationsmöglichkeiten stellt einen bedeutsamen Risikofaktor für Burn-out dar (Wieland et al., 2004). Auch in neueren Studien zur Einführung von IT-Systemen konnten diese Auswirkungen beobachtet werden (Baronas & Louis, 1988; Igbaria, Parasuraman & Badawy, 1994; Jarvenpaa & Ives, 1991). Dabei zeigten sich schon positive Effekte bei ausschließlich passiver Einbeziehung, z. B. durch umfangreiche Informationen über Vor- und Nachteile eines neuen Systems (Griffith & Northcraft, 1993, 1996).

Nach Antoni (1999) basieren diese positiven Auswirkungen von Partizipation auf kognitiven und motivationalen Faktoren. Zu den kognitiven Faktoren zählt Antoni die Verbesserung des Informationsflusses zur unteren hierarchischen Ebenen, eine bessere Integration und Ausnutzung von Wissen (Rosenberg & Rosenstein, 1980) sowie ein besseres Problem- und Arbeitsverständnis der Mitarbeiter. Zu den motivationalen Faktoren gehört die größere Akzeptanz von Veränderungen sowie eine stärkere Einbindung der Mitarbeiter, was eine stärkere Identifikation mit der Organisation zur Folge hat.

In den Arbeitsanalysen zur Softwareentwicklung wurde Partizipation in der Regel nicht erhoben. Es gibt Hinweise darauf, dass die Mitwirkungsmöglichkeiten in der Softwareentwicklung im Vergleich zu anderen Tätigkeitsfeldern groß sind. So ist z. B. das Team eine gängige Organisationsform in der Entwicklung von Softwareprodukten. Diese Arbeitsgruppen in der Softwareentwicklung können Teile ihrer Tätigkeit eigenständig gestalten und z. B. über Arbeitsbeginn und -ende entscheiden. Weitgehende Entscheidungsspielräume stehen ihnen oft nicht zur Verfügung. Fragen der Priorisierung, Zeitplanung und Organisation der Arbeitsaufgaben wird häufig von Projektverantwortlichen und zum Teil von Kunden festgelegt, was von den Projektmitarbeitern als belastend erlebt wird. Bei der Gestaltung von Rahmenbedingungen und bei Entscheidungen über den Einsatz von Ressourcen bleibt die Mehrheit der Projektmitarbeiter ausgeschlossen (Latniak & Gerlmaier, 2006). Eine unzureichende Beteiligung, schon in der Form eines unzureichenden Informationsflusses, ist einer der größten Stressoren in

der Arbeitssituation von Softwareentwicklern (Ivancevich, Napier & Wetherbe, 1983, 1985; Sonnentag, 1994).

Die Partizipationsmöglichkeiten in Open-Source-Projekten sind sehr umfangreich (Osterloh et al., 2004).

> *Obwohl die Entscheidungsverfahren verschiedener Open-Source-Projekte sehr unterschiedlich sind [...], ist ihnen doch gemein, dass sie ein hohes Ausmaß an Partizipation und Autonomie gewährleisten." (Osterloh et al., 2004)*

Durch umfassende Partizipation können sich TeilnehmerInnen an Open-Source-Projekten nicht nur über die wichtigen Entscheidungen des eigenen Projektes informieren, sondern sich auch umfangreich an Entscheidungen beteiligen. In vielen Projekten existieren demokratische Prozesse für die Verabschiedung von grundlegenden Entscheidungen. Diese Form der Entscheidungsfindung soll am besten geeignet sein, um Interessenkonflikte in der Softwareentwicklung zu überwinden (Bjerknes, Ehn & Kyng, 1987) und sie führt in Zusammenhang mit hoher Kommunikationsdichte zu einer hohen Prozess- und Produktqualität (Brodbeck, 1994a).

Für eine Reihe der organisatorischen Eigenschaften von Open-Source-Projekten konnte in Studien belegt werden, dass diese einen positiven Einfluss auf die wahrgenommene Partizipation haben. Dazu gehört die dezentrale Entscheidungsbefugnis der TeilnehmerInnen (Abrahamsson, 1977; Etzioni, 1971; Lohmann & Prümper, 2002; Tannenbaum, Kavcic, Rosner, Vianello & Wieser, 1974), der nicht-materielle Charakter der meisten Anreize (McMahon & Camilieri, 1975; Rothschild-Whitt, 1979; Tannenbaum et al., 1974; Warner & Heffernan, 1967), die Freiwilligkeit der Betätigung (Espinosa & Zimbalist, 1978; Rosenstein, 1977; Warner, 1964) und der freie Austausch von Informationen über formelle und informelle Kanäle (Hage & Aiken, 1967; Jain, 1978; Pennings, 1973). Viele Open-Source-Projekte werden durch eine Führungsperson an der Spitze der Hierarchie maßgeblich beeinflusst. Dieses sollte eigentlich negative Auswirkungen auf die Partizipationsmöglichkeiten haben. Bei Projekten, die durch Wahl oder Rotation diesen Posten besetzen, sollten die negativen Folgen gänzlich ausbleiben (Donaldson & Warner, 1979; Tannenbaum et al., 1974). Insgesamt sollten Open-Source-Projekte ein deutlich größeres Ausmaß an Partizipation bieten, als das in proprietären Projekten üblich ist.

4.3.2 Rückmeldung durch die Tätigkeit

Rückmeldungen sind die Grundlage für Mitarbeiter, um ihre Leistung einschätzen und verbessern zu können. Dabei kann man zwei wichtige Quellen von Rückmeldungen unterscheiden: zum einen die Informationen, die Mitarbeiter von ihrer Tätigkeit erhalten, und zum anderen Informationen von anderen Personen, wie Kollegen und Vor-

gesetzte (Hackman & Lawler, 1971). Inwieweit eine Tätigkeit Rückschlüsse darüber ermöglicht, wie gut oder wie schlecht die Leistung des Mitarbeiters ist, soll in diesem Kapitel besprochen werden. Im Kapitel 4.4.1 werden die Rückmeldungen durch andere Personen dargestellt.

In der Konzeption des JCM von Hackman und Oldham (1974) wird ausschließlich die Rückmeldung durch die Tätigkeit betrachtet, die Rückmeldung durch andere Personen wird nicht thematisiert. Rückmeldung durch die Tätigkeit hat in diesem Konzept eine hohe Bedeutung, was daran zu erkennen ist, dass sie neben der Autonomie als Faktor und nicht als Summand in die Berechnung des Gesamtmotivationspotenzials eingeht. Von den 5 Tätigkeitsmerkmalen des JDS weist Rückmeldung sehr hohe oder sogar die höchsten Korrelationen zur Arbeitszufriedenheit (z. B. Becherer, Morgan & Richard, 1982; Lee & Graham, 1986; O'Reilly & Caldwell, 1979) und zum Commitment auf (Lee & Graham, 1986). Rückmeldung weist nicht nur Zusammenhänge zu Einstellungs-, sondern auch zu Verhaltensvariablen auf. Von den Tätigkeitsmerkmalen, die Fried und Ferris (1987) in ihrer Meta-Analyse vergleichen, hat Rückmeldung den stärksten Zusammenhang mit den Fehlzeiten (Korrelation von -0,19) und der Leistung (Korrelation von 0,22). Der Zusammenhang zwischen Feedback und Leistung konnte auch die Zielsetzungsforschung bestätigen (Algera, 1990). Rückmeldung durch die Tätigkeit gilt als das einflussreichste Tätigkeitsmerkmal, und Verbesserungen sollten den größten Nutzen für eine Organisation erzielen (Fried, 1991).

Eine Erklärung für die Wirkungsweise von Rückmeldung bietet die Theorie der Selbstwirksamkeit ("self-efficacy", Bandura, 1977) bzw. das verwandte Konzept der Kompetenz (Thomas & Velthouse, 1990). Diese Theorien gehen davon aus, dass sowohl die Entwicklung als auch das Wissen über die eigene Kompetenz auf dem erhaltenen Feedback beruht. Feedback vermittelt direkte und klare Informationen über die eigene Effizienz (Hackman & Oldham, 1980). Daraus resultiert ein erhöhter Einsatz und größere Ausdauer bei der zukünftigen Erfüllung dieser Tätigkeit (Bandura, 1977). Entsprechend den Prognosen der Theorie der kognitiven Evaluierung (Deci, 1971) vermittelt positive Rückmeldung Kompetenz, was zu einer Erhöhung der intrinsischen Motivation führt (Ergebnisse einer umfangreichen Metaanalyse, Deci et al., 1999).

Eine wichtige Quelle der Rückmeldung für die Tätigkeit des Programmierers ist das Übersetzungsprogramm ("Compiler"), das den Quellcode in eine maschinenlesbare Form umwandelt. Dabei wird das Ergebnis der Programmiertätigkeit vor allem auf syntaktische, aber auch einige semantische Fehler hin untersucht. Der Entwickler erhält direkte Rückmeldung über seine Arbeit aus Fehlerberichten, die im Prozess der Übersetzung erstellt werden. Eine weitere Quelle von Rückmeldung durch die Arbeit sind die Ergebnisse des manuellen oder automatisierten Testprozesses. Hierbei wird die erstellte Software mehr oder weniger systematisch in alle denkbaren Zustände versetzt und die

Reaktionen werden mit den Erwartungen verglichen. Insbesondere in ausgereiften Projekten und bei der Wartung gehört das Testen zu den wichtigsten Tätigkeiten. Daher ist auch bei diesen Tätigkeiten das Ergebnis von Testläufen eine bedeutsame Quelle für Rückmeldungen. Dass computergenerierte Rückmeldung geeignet ist, positive Auswirkungen wie Leistungssteigerung hervorzurufen, wurde in einer Meta-Analyse gezeigt (Kluger & DeNisi, 1996).

Den Einfluss von Entwicklungswerkzeugen auf die Rückmeldung hat Burroughs (1991) in einem Quasi-Experiment untersucht. Die Verwendung eines interaktiven Werkzeugs zur Programmkontrolle konnte die wahrgenommene Rückmeldung signifikant steigern. Auch die Untersuchung von Lending und Chervany (1998b) spricht dafür, dass Entwicklerwerkzeuge einen großen Einfluss auf die Qualität der Rückmeldung haben: Tätigkeiten, bei denen mehr Entwicklungswerkzeuge eingesetzt werden (Programmierung, Wartung und Testen), zeigten höhere Werte für die Rückmeldung durch die Tätigkeit als solche, bei denen weniger Entwicklungswerkzeuge genutzt werden (Führungskräfte, Systementwickler und Systemanalysten).

Wie im Kapitel 3.2 gezeigt, ist sowohl für individuelle TeilnehmerInnen als auch für Unternehmen die Rückmeldung einer der wichtigsten Gründe für die Teilnahme an Open-Source-Projekten (Bonaccorsi & Rossi, 2004; Ghosh et al., 2002). So gehört für immerhin 27 % der TeilnehmerInnen an Open-Source-Projekten die Rückmeldung zu den vier wichtigsten Gründen für eine Teilnahme (Ghosh et al., 2002). 41 % der Unternehmen beteiligen sich an Open-Source-Projekten aufgrund der Nützlichkeit der Rückmeldung (Bonaccorsi & Rossi, 2004).

Der wichtigste Gegenstand der Rückmeldung sollte dabei das Arbeitsergebnis eines Entwicklers, der von ihm erzeugte Code sein. Der Quellcode in Open-Source-Projekten wird durch häufige Veröffentlichung regelmäßig in das Gesamtprojekt integriert. Damit steht in Open-Source-Projekten regelmäßiger ein Endprodukt zur Verfügung, welches eine Rückmeldung an den Entwickler geben kann (z. B. ob die von ihm eingebrachten Entwicklungen ihre Aufgabe erfüllen). Zudem haben Open-Source-Entwickler eine größere Freiheit bei der Auswahl der Werkzeuge (vgl. Kapitel 3.2.6 und Kapitel 4.3.1) und damit die Möglichkeit, solche Werkzeuge auszuwählen, die ein optimales Maß an Rückmeldung ermöglichen.

Insgesamt sollten Open-Source-Entwickler mehr Rückmeldung aus ihrer Tätigkeit erhalten, als das bei proprietären Softwareentwicklern der Fall ist.

4.3.3 Aufgabengeschlossenheit

Tätigkeiten, die eine abgeschlossene Aufgabe beinhalten, wie das Bereitstellen einer Dienstleistung oder die Montage eines vollständigen Produktes, sind interessanter als

solche, die nur aus Bruchteilen der Gesamtaufgabe bestehen (Hackman & Oldham, 1980). Zu abgeschlossenen Aufgaben gehört, dass diese von Anfang bis Ende bearbeitet werden und die eigene Arbeit im Endprodukt identifizierbar ist (Hackman & Lawler, 1971). Die Aufgabengeschlossenheit gehört zu den fünf Merkmalen, die von Hackman und Oldham (1974) in das JCM aufgenommen wurden. Dabei sollte Aufgabengeschlossenheit neben anderen Charakteristika über den Mediator "erlebte Bedeutsamkeit" positive Auswirkungen auf die intrinsische Arbeitsmotivation und die allgemeine Arbeitszufriedenheit haben. Die Mediation über die "erlebte Bedeutsamkeit" konnte in einer Reihe von Studien nicht nachgewiesen werden. So wurden Effekte entgegen der vorhergesagten Richtung gefunden (Miner, 1980), zeigten sich in anderen Studien keinerlei Effekte zu kritischen psychologischen Erlebniszuständen (Fox & Feldman, 1988; Hackman & Oldham, 1976) oder es fanden sich stärkere Zusammenhänge zu den Erlebniszuständen "erlebte Verantwortung" und "Wissen um die Ergebnisse" (Arnold & House, 1980; Fried & Ferris, 1987; Johns, Xie & Fang, 1992). Auch wenn der Einfluss der Mediatoren noch unklar ist, weist die Aufgabengeschlossenheit, zusammen mit der Rückmeldung, den stärksten Zusammenhang mit der Arbeitsleistung auf (Fried & Ferris, 1987). Renn und Vandenberg (1995) vermuten, dass die wahrgenommene Wirksamkeit und Kompetenz als Mediator wirken.

In der Softwareentwicklung weisen Tätigkeiten Unterschiede in der Aufgabengeschlossenheit auf. So haben Systemdesigner im Vergleich zu Systemanalysten, Führungskräften und Wartungsfachkräften eine geringere Aufgabengeschlossenheit. Programmierer und mit dem Testen von Software beschäftigte Mitarbeiter zeigen den höchsten Grad an Aufgabengeschlossenheit (Lending & Chervany, 1997). Aufgabengeschlossenheit ist das einzige Merkmal, das bei Softwareentwicklungstätigkeiten im Bereich der Wissensverarbeitung höhere Werte aufweist als bei anderen Entwicklungstätigkeiten (Couger, 1986b). Die Ursache hierfür könnte darin liegen, dass in dem damals neuen Bereich die Tätigkeiten aufgrund mangelnder Erfahrung mit Aufgabenteilung zum größten Teil von einer Person vollständig erledigt wurden.

Untersuchungen zur Aufgabengeschlossenheit in Open-Source-Projekten liegen nicht vor. Einen wichtigen Einfluss auf die Aufgabengeschlossenheit hat die angewendete Methode der Softwareentwicklung: strukturierte Entwicklungsmethoden senken die Aufgabengeschlossenheit (Goldstein, 1982; Kraft, 1977). Auch wenn Open-Source-Softwareentwicklung kein vollständig chaotischer Prozess ist, so sind die verwendeten Methoden nur bedingt mit der proprietären Softwareentwicklung vergleichbar (Norin & Stöckel, 1998) und nicht klar definiert (McConnell, 1999). Der geringere Grad an Strukturiertheit des Open-Source-Softwareentwicklungsprozesses lässt vermuten, dass die Aufgabengeschlossenheit höher liegt als in der proprietären Softwareentwicklung.

Das Konzept der "objektorientierten Programmierung" (Programmierparadigma, bei dem Daten und Funktionen in einem Objekt zusammengefasst werden) wird mit einer geringeren Aufgabengeschlossenheit in Verbindung gebracht (Lending & Chervany, 1997). Auch wenn dieses Konzept durch seine Betonung von Modularität und Wiederverwendbarkeit gut mit den Eigenschaften des Open-Source-Entwicklungsprozesses zusammenpasst (Bonaccorsi & Rossi, 2003) und eine Reihe von Projekten dieses Konzept auch anwenden (Robbins, 2004), kann aufgrund der fehlenden Daten nicht entschieden werden, ob dies häufiger als in proprietären Softwareprojekten geschieht.

In seinen Studien im Bereich der Softwareentwicklung konnte Couger (1986b) zeigen, dass ein direkter Zusammenhang besteht zwischen der Aufgabengeschlossenheit und der Zeit, die ein Mitarbeiter in einem Anwendungsbereich verbringt. Auch wenn mehr als zwei Drittel der Open-Source-Entwickler in mehreren Projekten tätig sind (Hars & Ou, 2002), kann daraus nicht abgeleitet werden, dass diese eine geringere Aufgabengeschlossenheit aufweisen. Denn diese leitet sich in der proprietären Softwareentwicklung mutmaßlich aus der größeren Verantwortung ab, die Mitarbeitern eingeräumt wird, wenn sie sich ausschließlich auf ein einzelnes Projekt konzentrieren (Baroudi & Michael, 1986). Ein solcher Zusammenhang zwischen Projektanzahl und Verantwortung ist in Open-Source-Projekten, bei denen Aufgaben und Verantwortung von den TeilnehmerInnen selbstständig gewählt werden, unwahrscheinlich.

Ebenfalls für eine größere Aufgabengeschlossenheit spricht die stärkere Modularisierung von Open-Source-Projekten. Diese auf den ersten Blick widersprüchlich erscheinende Aussage erklärt sich damit, dass ein größere Modularisierung in einem großen und komplexen Projekt eine Voraussetzung sein kann, um auf einer Subprojektebene selbstständig arbeiten zu können. Für diese Teilmodule können dann Entwickler den gesamten Prozess der Softwareentwicklung von Planung/Architektur bis zum Testen eigenständig durchführen (Hertel, 2007).

Auch wenn noch empirische Ergebnisse ausstehen und die theoretischen Vorhersagen nicht einheitlich sind, kann davon ausgegangen werden, dass die Aufgabengeschlossenheit in Open-Source-Projekten größer ist als in proprietären Projekten.

4.3.4 Vielfalt

Die Ausführung von verschiedenen Tätigkeiten unter Einsatz von unterschiedlichen Fähigkeiten soll Arbeit interessanter machen. Die Vielfalt ist auch ein zentraler Bestandteil des Konzepts der Aufgabenerweiterung (Herzberg, 1968), was zu einer Verbesserung der sog. "Kontentfaktoren" und damit zur Erhöhung der Arbeitsmotivation führen soll. Bei der Vielfalt kann man zwischen der Aufgabenvielfalt (task variety) und der Anforderungsvielfalt (skill variety) unterscheiden. Morgeson und Humphrey (2003) stellt

einen Zusammenhang zwischen der Aufgabenvielfalt und dem Konzept der horizontalen Aufgabenerweiterung (job enlargement Herzberg, 1968) her. Dabei wird durch das Hinzufügen von mehreren gleichartigen oder ähnlich einfachen Aufgaben der Arbeitszyklus vergrößert. Einen größeren Einfluss auf die Arbeitsmotivation sollte die Anforderungsvielfalt haben, welche durch eine vertikale Aufgabenerweiterung (job enrichment, Herzberg, 1968) beeinflusst werden kann. Hierbei werden regulatorische und ausführende Tätigkeiten zusammengeführt, welche die Anwendung von unterschiedlichen Fähigkeiten erforderlich machen.

Aufgabenvielfalt

Unterschiede in der Aufgabenvielfalt (task variety) beziehen sich auf die Anzahl von unterschiedlichen gleichartigen Aufgaben innerhalb einer Tätigkeit, wie sie z. B. durch Rotation erreicht werden können. Die Aufgabenvielfalt ist eng mit dem Konzept der horizontalen Aufgabenerweiterung (job enlargement) verbunden (Herzberg, 1968). Inwieweit eine große Anzahl von Aufgaben geeignet ist, die Motivation der Mitarbeiter zu beeinflussen, ist umstritten. Herzberg äußert sich dazu skeptisch:

> *"Null plus Null ergebe ebenso wie Null mal Null wiederum lediglich Null."*
> *(zitiert nach Ulich, 1998, S. 185)*

Im Gegensatz dazu ist Hulin (1971) der Auffassung, dass nicht-routinierte und nicht-wiederholende Tätigkeiten sich positiv auf die Motivation auswirken. Eine Erklärung hierzu könnte die Activation-Theorie von Scott (1966) liefern. Diese besagt, dass Anzahl und Unterschiedlichkeit von Reizen dazu dienen können, einen Mitarbeiter zu motivieren und ein hohes Leistungsniveau aufrechtzuerhalten.

Empirische Ergebnisse zur Aufgabenvielfalt liegen aus Untersuchungen mit dem Job Characteristics Inventory (JCI Sims, Szilagyi & Keller, 1976) vor. Dieses enthält die Skala "Variety", die sich aus Items zum Umfang der unterschiedlichen ausgeführten Tätigkeiten zusammensetzt und damit als Maß für die Aufgabenvielfalt gelten kann. Eine Meta-Analyse von Fried (1991) zeigt dabei Zusammenhänge mit Mitarbeiterzufriedenheit und Leistung. In einer Pfadanalyse bei einer Stichprobe von Vertriebspersonal konnte kein Zusammenhang zwischen der Aufgabenvielfalt und der Zufriedenheit nachgewiesen werden (Bhuian & Menguc, 2002).

Untersuchungen von Zusammenhängen mit dem Tätigkeitsmerkmal "Aufgabenvielfalt" im Bereich der Softwareentwicklung liegen bisher nicht vor. Ergebnisse von Klemens (2003) deuten darauf hin, dass im Bereich der Informationstechnologie eine überdurchschnittliche Vielfalt an Aufgaben existiert. Insbesondere ist eine große Vielfalt an Aufgaben zu erwarten, wenn Entwickler gleichzeitig an mehreren Projekten arbeiten (Baroudi & Michael, 1986).

In der Open-Source-Softwareentwicklung könnte die Aufgabenvielfalt weitaus größer sein als in der proprietären Softwareentwicklung. Hierfür spricht, dass Open-Source-Entwickler sich ihre Projekte und ihre Tätigkeiten frei aussuchen können (vgl. Kapitel 3.2.5). Ein Indiz für die größere Vielfalt ist, dass mehr als zwei Drittel der Entwickler in mehr als einem Projekt beschäftigt sind (Hars & Ou, 2002). Zudem haben Analysen von Open-Source-Projekten zu dem Ergebnis geführt, dass die meisten TeilnehmerInnen mehrere Aufgaben mit unterschiedlicher Intensität bearbeiten (Dietze, 2003b). Dies sollte dazu führen, dass der Grad der Aufgabenvielfalt in der Open-Source-Softwareentwicklung höher ist als in der proprietären Softwareentwicklung.

Anforderungsvielfalt

Die Anzahl der verschiedenen Aufgaben hat einen Einfluss auf die Zufriedenheit und Motivation der Mitarbeiter. Dieser Einfluss sollte größer sein, wenn sich die Anforderungen der Aufgaben voneinander unterscheiden. Dies ist z. B. der Fall, wenn durch vertikale Aufgabenerweiterung (job enrichment) neben der Ausführung noch planende oder kontrollierende Tätigkeiten hinzukommen (Van der Zwaan & Molleman, 1998). Dies macht die Anwendung verschiedener Fähigkeiten erforderlich, was häufig eine Herausforderung darstellt (Morgeson & Humphrey, 2003). Wie groß der Einfluss der Anforderungsvielfalt ist, zeigt sich darin, dass diese von den fünf Merkmalen des JDS den stärksten Zusammenhang mit der intrinsischen Motivation aufweist (Korrelation von 0,52). Zudem hat die Anforderungsvielfalt noch eine hohe Korrelation mit der Mitarbeiterzufriedenheit (0,45) und eine vergleichsweise niedrige mit der Leistung (0,09, Fried & Ferris, 1987).

Die Gestaltung von Tätigkeiten mit einer großen Anforderungsvielfalt kann Mitarbeiter motivieren und an das Unternehmen binden (Thatcher, Liu & Stepina, 2002a). In den unterschiedlichen Bereichen der Softwareentwicklung gibt es große Unterschiede in der Vielfalt der Anforderungen, die eine Tätigkeit aufweist. So haben überwiegend konzeptionelle Tätigkeiten (Systemanalysten, Systemdesigner, Führungskräfte) eine höhere Anforderungsvielfalt als Tätigkeiten, die einen geringeren konzeptionellen Anteil (Programmierer, Wartungsarbeiten) enthalten (Couger & Zawacki, 1980; Goldstein, 1989; Lending & Chervany, 1997). Trotz der Abwesenheit von konzeptionellen Aufgaben wird sowohl bei der Tätigkeit des Testens von Software (Lending & Chervany, 1997) als auch bei Bedienpersonal von kritischen Systemen (McKnight & Chervany, 1998) eine besonders hohe Anforderungsvielfalt festgestellt. Eine Erklärung könnte die relativ kurze Zyklusdauer der Tätigkeiten darstellen (McKnight & Chervany, 1998). Außerdem verfügen neue Aufgabenfelder wie z. B. in der Wissensverarbeitung ("knowledge engineering") aufgrund der Neuheit der notwendigen Fähigkeiten und Techniken über eine hohe Anforderungsvielfalt (Couger, 1986b). Eine große Vielfalt kann bei weniger

abwechslungsreichen Tätigkeiten auch erreicht werden, wenn ein Mitarbeiter in unterschiedlichen Bereichen beschäftigt wird (Couger, 1986b).

In Studien (Goldstein, 1982; Kraft, 1977; Lending & Chervany, 1997) konnte gezeigt werden, dass auch die Anforderungsvielfalt durch die angewendete Methode beeinflusst wird. Wie auch bei der Aufgabengeschlossenheit und der Autonomie senken strukturierte Methoden die Anforderungsvielfalt (Goldstein, 1982; Kraft, 1977) genauso wie objektorientierte Softwareentwicklung (Lending & Chervany, 1997).

Bei der Beurteilung der Anforderungsvielfalt in der Open-Source-Softwareentwicklung kann man nicht davon ausgehen, dass im Open-Source-Bereich mehr konzeptionelle Tätigkeiten anfallen, da ein Entwurf für ein Projekt sinnvollerweise nicht beliebig oft gemacht werden kann, sodass nur noch die Aufgabe der Umsetzung verbleibt. Wie bereits zuvor diskutiert (Kapitel 3.2.6), kann man davon ausgehen, dass in der Open-Source-Softwareentwicklung weniger strukturierte Methoden eingesetzt werden, was allein schon eine höhere Anforderungsvielfalt vermuten lassen könnte. Ebenfalls für eine höhere Anforderungsvielfalt spricht, dass in selbst organisierten Teams, wie sie im Open-Source-Bereich praktiziert werden (vgl. Kapitel 3.2.4), für die erfolgreiche Erfüllung der Aufgabe eine größere Anzahl von Fähigkeiten von den TeilnehmerInnen notwendig ist und auch eingesetzt werden kann (Cordery, Mueller & Smith, 1991). Zu den Fähigkeiten, die eingesetzt werden können, gehören auch solche, die sich aus der vertikalen Aufgabenerweiterung (job enrichment) ergeben. Bei autonomen Gruppen gehören dazu das

> *"eigenständige Setzen von Zielen bzw. Teilzielen, die Übernahme von Planungs- bzw. Dispositionsfunktionen und das gemeinsame Treffen von Entscheidungen" (Ulich, 1998).*

Diese Aufgaben erfordern jeweils eine Vielzahl von Fähigkeiten und sind voraussichtlich häufiger in diesem Umfang in der Open-Source-Softwareentwicklung einzusetzen, was Anlass zu der Vermutung gibt, dass in diesem Bereich eine deutlich höhere Anforderungsvielfalt anzutreffen ist.

4.3.5 Bedeutsamkeit der Aufgabe

Die Ausführung von Tätigkeiten beeinflusst durch produzierte Waren und Dienstleistungen das Leben von anderen Personen innerhalb und außerhalb der Organisation. Je stärker dieser Einfluss ist, desto größer ist die Bedeutsamkeit einer Tätigkeit. Im Konzept von Hackman und Oldham (1976) ist Bedeutsamkeit eines der fünf Tätigkeitsmerkmale des JCM. In deren Konzeption bewirkt das Verständnis der Auswirkung der Tätigkeit auf andere eine Steigerung der wahrgenommenen Wichtigkeit seiner Arbeit und hat damit Einfluss auf die Arbeitszufriedenheit und intrinsische Motivation. Der

Zusammenhang zwischen Bedeutsamkeit einer Tätigkeit und Arbeitszufriedenheit stellt eine der fundamentalen Komponenten des JCMs dar (Todd, 2004). In der Meta-Analyse von Fried und Ferris (1987) zeigte sich eine mittlere Korrelation (0,35) mit Arbeitszufriedenheit.

Insbesondere bei Tätigkeiten, die der Bewahrung und Förderung von menschlichem Leben (z. B. im Gesundheitswesen oder bei Sicherheitsunternehmen) dienen, kann eine hohe Bedeutsamkeit der Aufgabe erwartet werden und wurde auch gefunden (Morgeson & Humphrey, 2003). Obwohl Tätigkeiten in der Softwareentwicklung in den meisten Fällen nicht in diese Kategorie fallen, wurden bei deren Analyse teilweise überdurchschnittliche Werte für die Bedeutsamkeit der Aufgabe gefunden (Couger & Zawacki, 1980). Besonders hohe Werte weisen die Tätigkeiten von Führungskräften in der Softwareentwicklung auf, während sich zwischen den anderen Aufgabenbereichen (Systemdesigner, Systemanalysten, Wartungsfachkräfte, Programmierer und Mitarbeiter, die für das Testen von Software zuständig sind) keine signifikanten Unterschiede ergeben (Lending & Chervany, 1997). Besonders niedrige Werte wiesen 1986 Tätigkeiten im Bereich der Wissensverarbeitung auf ("knowledge engineering": ein Spezialbereich der Computerwissenschaften mit Fokus auf Wissensrepräsentation, -verarbeitung und künstliche Intelligenz). Damals wurde vermutet, dass dies an der relativen Neuheit des Tätigkeitsfeldes liegen könnte: fehlende Erfahrung führte dazu, dass die tatsächliche Bedeutung noch nicht erkannt wurde (Couger, 1986b). Die Bedeutsamkeit einer Tätigkeit kann auch negative Auswirkungen haben (Carayon et al., 2003). Bei Tätigkeiten in der IT-Branche konnte nur im Falle von weiblichen Angestellten ein Zusammenhang zur Anspannung nachgewiesen werden (Carayon et al., 2003).

Auch wenn Tätigkeiten in der proprietären Softwareentwicklung als bedeutsam erlebt werden, so werden vermutlich nicht viele Entwickler ihre eigene Tätigkeit als außergewöhnlich nützlich für die Gesellschaft wahrnehmen. Im Gegensatz dazu bezieht eine ganze Reihe von Open-Source-Entwicklern ihre Motivation aus dem gesellschaftlichen Nutzen ihrer Tätigkeit (Lakhani, Wolf, Bates & DiBona, 2002). Sie stellen ihre Arbeitskraft für diesen Nutzen aus ideologischen Gründen zur Verfügung. Sie sehen darin eine Möglichkeit, Stellung zu Ideen wie Freiheit, Teilen, unbeschränkter Austausch von Informationen, Antikapitalismus und Anti-Monopolismus zu beziehen (Franck & Jungwirth, 2002b; Manville et al., 1999; Markus et al., 2000; Perkins, 1999; Raymond, 1999a). Moralische und politische Aspekte sind besonders wichtig für Entwickler, die sich der Free-Software-Gemeinschaft zugehörig fühlen (vgl. Kapitel 2). Der Beitrag zu einem Free-Software- oder Open-Source-Projekt kann auch als aktives politisches Statement gegen monopolistische Softwarekonzerne und kapitalistische Einschränkungen von Informationsweitergabe verstanden werden.

Dass Open-Source-Software auch kostenlos erhältlich ist, hilft z. B. auch Menschen aus Entwicklungsländern, am weltweiten digitalen Informationsaustausch teilzunehmen (Feller & Fitzgerald, 2002). So sind Open-Source-Entwickler beispielsweise auch an der Entwicklung der Softwareausstattung für das OLPC (One Laptop per Child)-Projekt beteiligt. Bei diesem Projekt sollen Kinder in Entwicklungsländern mit einem Laptop ausgestattet werden. Diese Geräte, die nicht mehr als 100 US-Dollar kosten sollen, arbeiten ausschließlich mit speziell entwickelter oder angepasster Open-Source-Software. Insbesondere bei Entwicklern, die aus ideologischen Gründen Einschränkungen durch Softwarelizenzen ablehnen, kann das Gefühl der Bedeutsamkeit noch durch den politischen Wert der Entwicklung von Open-Source-Software gesteigert werden.

Bei Open-Source-Produkten, die Marktführer sind (wie der Apache-Server) oder die millionenfache Verbreitung gefunden haben (Linux, Mozilla Firefox oder OpenOffice.org), spiegelt sich die Bedeutsamkeit der Aufgabe im großen Interesse der NutzerInnen sowie auch der Öffentlichkeit und Presse wider.

Ideologische Motive sind zwar für einen großen Teil der Open-Source-Entwickler nicht das Hauptmotiv zur Teilnahme (Lakhani & Wolf, 2003), dennoch unterstützt eine Mehrheit die ideologischen Aspekte von Open-Source. So sind über 75 % der Entwickler davon überzeugt, dass NutzerInnen das Recht haben sollten, Einblick in den Quellcode zu nehmen und mehr als 65 % gehen davon aus, dass alle Softwareentwicklung unter einem Open-Source-Lizenzmodell stattfinden werden sollte (David et al., 2003). Auch wenn in der vorliegenden Untersuchung nicht zwischen Entwicklern von Open-Source- und Free-Software-Projekten unterschieden wird, so sollte der Anteil an überzeugten Anhängern von freier Software in Free-Software-Projekten größer sein.

Da Open-Source-Softwareentwickler im Bewusstsein arbeiten, dass sie qualitativ hochwertige Software kostenlos NutzerInnen außerhalb ihrer Organisation zur Verfügung stellen, kann davon ausgegangen werden, dass sie ihre Tätigkeit als bedeutsamer wahrnehmen, als dies im Bereich der proprietären Softwareentwicklung der Fall ist. Die einzigen Erkenntnisse, die gegen diese Annahme sprechen könnten, entstammen einer Studie zu Tätigkeitsmerkmalen im Bereich der freiwilligen-gemeinnützigen Arbeit (Güntert & Wehner, 2005). Vergleichbar zu Open-Source-Projekten werden auch bei diesen Tätigkeiten die TeilnehmerInnen nicht entlohnt und sind ebenfalls ideologisch motiviert. Im Vergleich zur Erwerbsarbeit hat man in diesem Bereich jedoch einen niedrigeren Grad an Bedeutsamkeit der Aufgabe festgestellt. Es erscheint fraglich, ob ein direkter Vergleich zwischen den Tätigkeiten der Softwareentwicklung und dem Engagement im Sport, der Jugendarbeit, in kulturellen Vereinen und im sozialkaritativen Bereich möglich ist.

4.3.6 Qualifikationspotenzial

Im Rahmen von Arbeitsanalysen ist Qualifikation eine häufig untersuchte Variable. Allerdings wurde in vielen Fällen Qualifikation nicht als Tätigkeitsmerkmal (im Sinne von Qualifikationspotenzial einer Tätigkeit), sondern als organisationales Kriterium oder Moderator berücksichtigt. So ist im Konzept des JCM die Zufriedenheit mit den Qualifikations- bzw. Wachstumsmöglichkeiten (growth satisfaction, Hackman & Oldham, 1980) ein organisationales Kriterium. Zusätzlich taucht das Bedürfnis nach Wachstum (growth need strength, Hackman & Oldham, 1976, 1980) als Moderator auf. Das Wachstumsbedürfnis moderiert zwischen den Tätigkeitsmerkmalen und der Motivation sowie der Zufriedenheit: für Personen mit besonders hohem Wachstumsbedürfnis sollte der Zusammenhang besonders ausgeprägt sein. Ein hohes Wachstumsbedürfnis entspricht einem hohen Bedürfnis an Selbstverwirklichung, persönlicher Weiterentwicklung und Lernen (Hackman & Oldham, 1976). Das Wachstumsbedürfnis ist der am häufigsten untersuchte Moderator (Morgeson & Campion, 2003), seine moderierende Eigenschaft konnte jedoch noch nicht abschließend empirisch bestätigt werden. In zwei Meta-Analysen konnte der moderierende Effekt der Growth Need Strength zwar bestätigt werden (Fried & Ferris, 1987; Loher et al., 1985), aber aufgrund der Anzahl der eingeschlossenen Studien und wegen methodischer Probleme verbleiben erhebliche Zweifel am tatsächlichen Ausmaß des Effekts (für eine Kritik siehe: Morgeson & Campion, 2003). Umfangreiche aktuelle Studien fanden keinerlei moderierenden Effekt (Rentsch & Steel, 1998; Tiegs, Tetrick & Fried, 1992).

In anderen Modellen (Hacker & Iwanowa, 1984) ist das Lernpotenzial ein Merkmal von vollständigen Tätigkeiten, und im Belastungs-Ressourcen-Gesundheits-Modell, das dem Arbeitsanalyseverfahren SALSA zugrunde liegt, stellt das Qualifikationspotenzial eine "Ressource" (Schutzfaktor) zur Aufrechterhaltung und Wiederherstellung von Gesundheit dar (Udris & Rinmann, 1999). Ein niedriges Qualifikationspotenzial konnte als Risikofaktor für negatives Befinden (z. B. Anspannung, Nervosität, Unlust) identifiziert werden (Wieland et al., 2004). Ein hohes Qualifikationspotenzial gilt dagegen als eine wichtige Auslösebedingung für ein Flow-Erlebnis (Rau & Riedel, 2004). In einer Untersuchung von Agho, Price und Mueller (1992) wird die "Routiniertheit der Tätigkeit" mit der Skala von Price und Mueller (1986a) erfasst, deren vier Items die Anwendung und Erweiterung der eigenen Fähigkeiten durch die Tätigkeit abfragen und damit eng mit dem Qualifikationspotenzial verwandt ist. Zwischen Routiniertheit und Mitarbeiterzufriedenheit zeigte sich ein starker negativer Zusammenhang. In einer Untersuchung zu den Auswirkungen von Training zeigte sich, dass Training keinen Einfluss auf die Arbeitszufriedenheit, dafür aber einen signifikanten Einfluss auf die intrinsische Motivation hat. Der Prädiktor Training klärt dabei die Varianz auf, die von den Tätigkeitsmerkmalen des JCM nicht erklärt werden kann (Kahn & Robertson, 1992). Während

der Einfluss des Wachstumsbedürfnisses als Moderator unklar ist, scheinen existierende Untersuchungen darauf hinzuweisen, dass Qualifikation als eigenständiges Tätigkeitsmerkmal zusätzliche Varianz aufklären kann. Eine Aufnahme dieser zusätzlichen Tätigkeitsmerkmals erscheint somit sinnvoll.

Das Qualifikationspotenzial ist nicht als Tätigkeitsmerkmal in der Kategorisierung von Morgeson und Campion (2003) enthalten. Eine Zuordnung zum Merkmalsbereich der Komplexität erscheint trotzdem angemessen. So wurden direkt Zusammenhänge mit der Komplexität der Tätigkeit in einige Studien nachgewiesen (Berlew & Hall, 1966; Campbell, 1988; McCauley, Ruderman, Ohlott & Morrow, 1994). Zusätzlich wurden noch in anderen Untersuchungen Zusammenhänge zwischen dem Qualifikationspotenzial und einzelnen Aspekten der Komplexität wie Aufgabenvielfalt (Frese, 1994), Tätigkeitsspielraum (Frese, 1994; Karasek, 1998; Lankenau, 1984) und Rückmeldung (Bergmann, 2000a; Frese & Altmann, 1989; Kluger & DeNisi, 1996) nachgewiesen.

Die Tätigkeit des Softwareentwicklers weist eine hohe Dynamik und Veränderungsgeschwindigkeit auf und erfordert flexible individuelle und organisatorische Anpassungsprozesse (Wieland et al., 2004). Eine fortwährende Qualifikation ist dabei für die kontinuierliche Verbesserung der Produktivität und Qualität der Softwareentwicklung notwendig. Dabei müssen Entwickler bei neuen Aufgaben das Wissen des Anwendungsbereichs erlernen bzw. aktualisieren (Curtis et al., 1988; Selig, 1986; Walz, Elam & Curtis, 1993; Weltz & Ortmann, 1992).

Des Weiteren besteht die Notwendigkeit, mit den Veränderungen von Methoden und Werkzeugen im Bereich der Softwareentwicklung Schritt zu halten (Brodbeck, 1994b). Für diese Aufgabe wird nach Analysen von Brodbeck (1994b) im Durchschnitt 6 % der Arbeitszeit aufgewendet. In frühen und insbesondere in mittleren Projektphasen kann der Anteil der Zeit für Qualifikation deutlich höher sein. Am weitesten verbreitet ist dabei die Wissensakquise durch autodidaktisches Lernen, das meist in die Arbeit der Softwareentwicklung integriert stattfindet. Im Vergleich zu anderen Bereichen weisen Tätigkeiten in der Informationstechnologie ein überdurchschnittliches Qualifikationspotenzial auf (Klemens, 2003). Aufgrund der Bedeutung von Qualifikation bei Mitarbeitern aus dem Bereich Informationssysteme stellt auch das Qualifikationspotenzial (learning opportunity) ein Schlüsselmerkmal in diesem Bereich dar (Bostron, 1978 zitiert nach Sein & Bostrom, 1991).

Die Notwendigkeit zur Weiterbildung in der Softwareentwicklung scheint auch einer der wichtigsten Gründe für die Teilnahme an Open-Source-Projekten zu sein. Durch die Übertragbarkeit der Erkenntnisse aus Open-Source-Projekten können diese auch bei der Entwicklung kommerzieller Produkte hilfreich sein (Lerner & Tirole, 2002a). Bei den mittlerweile häufiger untersuchten Motiven zur Teilnahme an Open-Source-Projekten

erwies sich die Möglichkeit zum Lernen konsistent als das wichtigste oder eines der wichtigsten Motive. In der Befragung von Lakhani und Wolf (2003) war mit 42 % die Verbesserung der Programmierfähigkeiten das am zweithäufigsten genannte Motiv. Bei der Frage nach dem persönlichen Nutzen aus der Teilnahme gaben die meisten TeilnehmerInnen (93 %) an, dass sich ihr Wissensstand verbessert hat (Lakhani et al., 2002). In der Befragung von Ghosh et al. (2002) war mit 79 % das Lernen und Entwickeln von Fähigkeiten der häufigste Grund für den Beitritt in ein Open-Source-Projekt und die zweithäufigste Ursache war der Austausch von Wissen und Fähigkeiten (50 %). Zu vergleichbaren Resultaten kamen Hars und Ou (2002), bei deren Befragung 71 % der Entwickler angaben, an Open-Source-Projekten teilzunehmen, um ihre Programmierfähigkeiten zu verbessern. Auch in der Untersuchung von Long (2003) gaben 79 % der TeilnehmerInnen an, mit der Teilnahme ihre Fähigkeiten verbessern und 50 % ihr Wissen teilen zu wollen.

Die Möglichkeiten, etwas zu lernen, werden durch die Einsehbarkeit des gesamten Quellcodes, durch Kontaktmöglichkeiten mit Experten, durch Austausch mit den Peers zu technischen Fragestellungen und durch direkte Rückmeldung auf die eigene Arbeit in Open-Source-Projekten gefördert (Krogh, Haefliger & Spaeth, 2003a). Im Gegensatz zur proprietären Softwareentwicklung können sich die TeilnehmerInnen durch die selbstständige Auswahl von Projekten solche auswählen, die ihnen die besten Lernmöglichkeiten eröffnen (Hars & Ou, 2002). Zudem gibt es Hinweise darauf, dass die Geber von Hilfestellung hierdurch ebenfalls lernen (Lakhani & Hippel, 2000). Ebenfalls förderlich sollte sich die Organisationsstruktur von Open-Source-Projekten auswirken. Mantei (1981) vermutet, dass in der Softwareentwicklung dezentralisierte demokratische Gruppen (democratic decentralized groups) schneller lernen als kontrollierte zentralisierte oder dezentralisierte (controlled centralized groups, controlled decentralized groups), wie sie üblicherweise in der Softwareentwicklung anzutreffen sind. Dem Qualifikationspotenzial in der proprietären Softwareentwicklung könnten auch unterschiedliche Interessen von Mitarbeitern und Unternehmensleitung im Wege stehen: Während Mitarbeiter aus dem Bereich der Informationstechnologie sich Trainings aus dem Bereich Führungskräfteentwicklung wünschten, bevorzugten deren Vorgesetzte Trainings, die im direkten Zusammenhang zur gegenwärtigen Tätigkeit des Mitarbeiters stehen (Burn, Ma & Ng Tye, 1995). Auch eine Erweiterung der Aufgaben in horizontaler und vertikaler Richtung führt zu Tätigkeiten mit einem höheren Qualifikationspotenzial (Schumann, 1998). Dabei wachsen erweiterte Tätigkeiten üblicherweise mit den Möglichkeiten und Fortschritten des Mitarbeiters mit und eröffnen beständig neue Herausforderungen (Parker, 1998).

Viele Open-Source-TeilnehmerInnen sind auch in der proprietären Softwareentwicklung tätig und engagieren sich dennoch in Open-Source-Projekten. um zusätzliche Qualifika-

tionen zu erwerben. Eigenschaften wie Peer-Reviews, eigenständige Wahl von Tätigkeiten und die Organisationsstruktur in Open-Source-Projekten geben zu der Hypothese Anlass, dass Open-Source-Projekte eine qualifikationsfördernde Wirkung haben, die über Tätigkeiten in Unternehmen hinausgeht. Die Tätigkeiten im Open-Source-Bereich sollten damit über ein deutlich höheres Qualifikationspotenzial verfügen als Tätigkeiten im proprietären Umfeld.

4.3.7 Zusammenfassung: Komplexität der Arbeit

Wie im vorangegangen Abschnitt gezeigt wurde, ist für die Tätigkeit von Open-Source-Softwareentwicklern ein größeres Maß an Autonomie und Partizipation als bei Tätigkeiten proprietärer Softwareentwickler zu erwarten. Weiterhin sollte die Rückmeldung durch die Tätigkeit, die Aufgabengeschlossenheit, die Anforderungsvielfalt und die Aufgabenvielfalt in der Open-Source-Softwareentwicklung größer sein als in der proprietären Softwareentwicklung. Da diese Merkmale zur Komplexität der Tätigkeit beitragen, wird folgende Hypothese aufgestellt:

> *Hypothese 1: Die Komplexität ihrer Tätigkeitsmerkmale wird von Open-Source-Softwareentwicklern höher eingestuft als von Beschäftigten im Bereich der proprietären Softwareentwicklung.*

Neben den Merkmalen der Komplexität der Arbeit sind auch soziale Aspekte für die Open-Source-Softwareentwicklung von Bedeutung. Diese werden im folgenden Kapitel diskutiert.

4.4 Soziale Aspekte der Tätigkeit

Die interpersonalen und sozialen Aspekte der Arbeit erhielten in der bisherigen Forschung weit weniger Aufmerksamkeit als motivationale Merkmale (Morgeson & Campion, 2003; Parker & Wall, 2002). Die geringe Aufmerksamkeit scheint nicht gerechtfertigt, da sowohl von Forschern (Trist & Bamforth, 1951) als auch von Stelleninhabern (Stone & Gueutal, 1985) diese für wichtige Aspekte der Arbeit befunden werden. In der Softwareentwicklung stellt die Kommunikation einen wichtigen Aspekt der Tätigkeit dar (Brodbeck, 1994a; Brodbeck, 2001). Dies macht eine Aufnahme der interpersonalen Aspekte in eine vollständige Arbeitsanalyse in diesem Tätigkeitsfeld unabdingbar, obwohl sie in vielen Untersuchungen ausgelassen wurden (z. B. Carayon et al., 2003; z. B. Goldstein & Rockart, 1984; McKnight & Chervany, 1998). Die geringe Bedeutung der sozialen Aspekte lässt sich vermutlich mit der überproportionalen Bedeutung des JCM von Hackman und Oldham (1974) erklären, welches keine sozialen Aspekte einschließt. In der Untersuchung, die dem JCM zugrunde liegt (Yale Job Inventory: YJI), wurden noch interpersonale Tätigkeitsmerkmale erfasst (dealing with others, friendship

opportunities); diese zeigten keine Beziehungen zu Motivationalen- und Leistungsvariablen (Hackman & Lawler, 1971) und wurden nicht in das JCM aufgenommen[6].

Im Gegensatz zu den in der psychologischen Forschung häufig betrachteten Tätigkeiten von Industriearbeitern sind die Tätigkeiten von Softwareentwicklern komplexer, weniger routiniert und dynamischer. Dies führt dazu, dass für diese Tätigkeiten ein höheres Maß an interpersonaler Interaktion erforderlich ist (Brodbeck, 2001). Es kann davon ausgegangen werden, dass Softwareentwickler 30 bis 50 % ihrer Arbeitszeit mit Kommunikation verbringen (Hauptman, 1986; Krasner et al., 1987; Page & Caskey, 1988; Sommerville, 1989). Dabei stellen "Kernaufgaben" der Softwareentwicklung (z. B. Spezifizieren, Codieren, Testen, Fehlerentfernung) gleichmäßig hohe Kommunikationsanforderungen (Brodbeck, 1994b). Die Bedeutung von sozialen und interpersonalen Aspekten scheint gerechtfertigt.

Umstritten bleibt die Frage, wie hoch das Bedürfnis von Softwareentwicklern nach sozialer Interaktion ist. In seiner in der Informatik stark beachteten Publikation zur "Psychologie des Programmierers" vermutet Weinberg (2004), dass die meisten Programmierer auf die Frage nach der bevorzugten Arbeitsweise angeben würden, dass sie am liebsten allein arbeiten, und zwar an einem Ort, an dem sie nicht durch andere Personen gestört werden. Während die Ergebnisse von Couger und Zawacki (1978) bestätigen, dass Softwareentwickler nur über ein vernachlässigbares Bedürfnis nach Zusammenarbeit mit anderen Personen verfügen, konnte dies in einer weiteren Untersuchung mit derselben Methode nicht bestätigt werden (Ferratt & Short, 1986). Mitarbeiter aus dem Bereich der Informationstechnologie zeigen keine von anderen Bereichen abweichenden sozialen Bedürfnisse. Auch die Rangfolge der Bedürfnisse von Fitz-enz (1978) zeigt, dass der Bedarf nach interpersonellem Austausch bei Softwareentwicklern eine vergleichbare Bedeutung hat wie bei anderen Berufsgruppen.

4.4.1 Soziale Unterstützung

Soziale Unterstützung ist eine Hilfestellung durch Personen, die im beruflichen Kontext durch Mitarbeiter oder Vorgesetzte geleistet wird. Dabei kann die soziale Unterstützung in Form von direkter Hilfeleistung, durch Hinweise, durch emotionale oder durch bewertungsbezogene Unterstützung erfolgen (vgl. Zapf, 1998). Als Tätigkeitsmerkmal wurde der sozialen Unterstützung in der Forschung bisher nur wenig Aufmerksamkeit geschenkt (Parker & Wall, 1998), obwohl z. B. positive Auswirkungen von sozialer Unterstützung auf die Gesundheit sowie die Verarbeitung von Stress vielfach bestätigt wurden (z. B. Udris, 1987). Bei Tätigkeiten mit einem geringen Ausmaß an sozialer

---

[6] Das Merkmal „dealing with others“ wurde nicht in das JCM aufgenommen, jedoch als „supplementary measure“ in das zugehörige Instrument (JDS; Hackman & Oldham, 1980).

Unterstützung und begrenzten Kontrollmöglichkeiten für Mitarbeiter sind negative Folgen für die Gesundheit zu erwarten (zusammenfassend Karasek et al., 1998). Neben der Relation zu Gesundheit wurden auch Zusammenhänge von sozialer Unterstützung mit der Arbeitszufriedenheit (z. B. Morgeson & Humphrey, 2003; Nelson & Quick, 1991; Parkes, 1982; z. B. Seers & Graen, 1984) und der Effektivität von Teams (z. B. Campion, Papper & Medsker, 1996; Gladstein, 1984; Seers & Graen, 1984) beobachtet. Für die erhöhte Leistung existieren unterschiedliche Erklärungsansätze. Zum einen vermutet Reichers (1987), dass die Leistungssteigerung durch neue Fähigkeiten und adäquates Verhalten erreicht wird, welche durch soziale Interaktion erlernt werden. Zum anderen hat soziale Interaktion eine aktivierende Wirkung, was auch bei banalen Aufgaben zu einer längeren Aufrechterhaltung der Bemühungen beitragen kann (Campion, Medsker & Higgs, 1993).

Die soziale Unterstützung im Bereich der Informationstechnologie wird sowohl von Mitarbeitern als auch von Vorgesetzten höher eingeschätzt als in anderen Berufen (Klemens, 2003). Auch ein Zusammenhang zwischen der Abwesenheit von sozialer Unterstützung und negativen Auswirkungen auf das Wohlbefinden der Mitarbeiter konnte gefunden werden (Latniak & Gerlmaier, 2006; Wieland et al., 2004). Unterschiedliche Tätigkeitsarten in der Softwareentwicklung gestatten ein unterschiedliches Ausmaß an sozialer Interaktion ("dealing with others"). So haben Systemanalysten und Führungskräfte viele Möglichkeiten zur sozialen Interaktion. Während Systemdesigner und Wartungsfachkräfte eine mittlere soziale Interaktion aufweisen, wird diese von Programmierern und von mit dem Testen von Software beschäftigten Mitarbeitern als niedrig eingeschätzt (Lending & Chervany, 1997). Zusammen mit geringen Partizipationsmöglichkeiten gehört eine geringe soziale Unterstützung zu den wichtigsten Quellen von Stress in der Softwareentwicklung (Sonnentag, 1994).

Trotz der beeindruckenden Größe von manchen Open-Source-Projekten (vgl. Kapitel 3.2.1) und der prinzipiellen Möglichkeit, soziale Unterstützung auch über elektronische Medien zu leisten (vgl. Kapitel 3.2.2), ist es fraglich, ob soziale Unterstützung im gleichen Ausmaß geleistet werden kann wie in proprietären Entwicklungsprojekten. Wie in Kapitel 3.2.1 beschrieben, wird die Größe von Open-Source-Projekten dadurch relativiert, dass neben großen Projekten eine Vielzahl von kleinen Projekten existiert. Aus diesem Grund kann angenommen werden, dass die meisten Open-Source-Entwickler in ihren Projekten nur über ein Netzwerk begrenzten Ausmaßes verfügen. Hinzu kommt, dass in diesem Netzwerk viele Entwickler nicht sonderlich aktiv sind und damit auch nur gelegentlich zu einem Austausch zur Verfügung stehen. Weiterhin instabil wird dieses Netzwerk durch eine hohe Fluktuation der TeilnehmerInnen. Insgesamt ist die Größe des sozialen Netzwerks und die Anzahl der Kontakte nicht mit dem Ausmaß an sozialer Unterstützung gleichzusetzen (Schaefer, Coyne & Lazarus,

1981), sondern ein soziales Netzwerk stellt nur die Voraussetzung für soziale Unterstützung dar (Berkman, 1984). Die Anzahl der Kontakte im Netzwerk sagt noch nichts über die Qualität der sozialen Unterstützung aus, zudem können Kontakte auch als Belastung empfunden werden.

Es kann insgesamt davon ausgegangen werden, dass proprietäre Entwickler mehr soziale Unterstützung aus ihren Netzwerke beziehen (vgl. dazu auch Burt, 1987; Fischer, Sollie, Sorell & Green, 1989; Seeman & Berkman, 1988). Den wichtigeren Einfluss auf das Wohlbefinden übt nicht die Quantität, sondern die Qualität der sozialen Unterstützung aus (Antonucci & Akiyama, 1987; Israel & Antonucci, 1987). Auch in diesem Punkt zeigen Open-Source-Netzwerke durch ihre geringe Stärke der sozialen Beziehungen Defizite (vgl. Kapitel 3.2.2); sie bieten damit weniger soziale Unterstützung (Wellman, 1992). Diese Punkte zusammengenommen, sollten zu einem geringeren Ausmaß an sozialer Unterstützung in OSS-Projekten führen.

4.4.2 Rückmeldung durch Personen

Neben der Rückmeldung durch die Tätigkeit selbst (vgl. 4.3.2) erhalten Mitarbeiter Rückmeldung über die Ausführung ihrer Tätigkeit von Vorgesetzten und Kollegen. Während Hackman und Lawler (1971) noch in ihrer Konzeption zu Rückmeldung (feedback) ausführen, dass neben der Rückmeldung durch die Ausführung der Tätigkeit auch noch die Rückmeldung durch andere Personen besteht, fand dieser Aspekt scheinbar keine explizite Berücksichtigung in der Operationalisierung dieses Tätigkeitsmerkmals und wurde bei der nachfolgenden Konzeption des JDS nicht mehr berücksichtigt (Hackman & Oldham, 1976). Obwohl die Rückmeldung von Kollegen und Vorgesetzten vermutlich einen wichtigen Einfluss ausübt (Hackman & Oldham, 1980), ist sie in der Forschung zur Arbeitsgestaltung nur bedingt berücksichtigt worden. Deutlich mehr Aufmerksamkeit erfährt die Rückmeldung im Rahmen der pädagogischen und organisatorischen Forschung (Kluger & DeNisi, 1996). Dabei wurde neben den Quellen und Formen von Rückmeldung insbesondere die Wirkung auf Lernen, Motivation und Leistung untersucht (zusammenfassend Fairhurst, 2004). Rückmeldung soll positive Auswirkungen sowohl auf Motivation als auch auf die Zufriedenheit haben (Ammons, 1956). Die vermutlich umfassendste Untersuchung betrachtet in einer Meta-Analyse die Auswirkungen von Rückmeldung auf die Leistungen (Kluger & DeNisi, 1996). Dabei wurde eine mittlere Effektstärke (d=0,41) von sozialer Rückmeldung auf Leistung nachgewiesen. Dieser Effekt schließt keine Untersuchungen ein, die auf der Rückmeldung durch die Tätigkeit basieren. Vergleichbare Effektstärken (d=0,35) fanden sich auch bei einer Meta-Analyse, die Effekte von unterschiedlichen organisationalen Interventionen miteinander vergleicht (Guzzo, Jette & Katzell, 1985). Was in Untersuchungen vor Kluger und DeNisi (1996) häufig nicht ausreichend beachtet wurde, sind die Moderatoren für den Zu-

sammenhang mit Leistung. Ein deutlicher Hinweis auf solche Moderatoren ist, dass in mehr als einem Drittel (38 %) der Studien als Folge von Rückmeldung eine Reduktion der Leistung auftritt (Kluger & DeNisi, 1996).

Rückmeldung ist auch ein wichtiges Thema in der Softwareentwicklung. So sieht z. B. Eunice (1998) in "aggressivem" Feedback eine der Grundlagen von erfolgreicher Softwareentwicklung. In bisherigen Untersuchungen sind Softwareentwickler meist unzufrieden mit der Rückmeldung, die sie erhalten (Couger, 1986a; Smits, Tanner & McLean, 1993). Die Förderung von Rückmeldung ist in der Softwareentwicklung eine heikle Intervention, da es sich bei Softwareentwicklung um eine komplexe Tätigkeit handelt (vgl. Kapitel 4.2). Die Schwierigkeit ergibt sich aus dem moderierenden Einfluss der Komplexität auf den Zusammenhang von Rückmeldung und Leistung (Kluger & DeNisi, 1996). Bei komplexen Tätigkeiten könnte Rückmeldung beim Empfänger den Fokus auf eine Verbesserung der Fähigkeiten lenken, ohne dass dieser über ausreichend Informationen verfügt und damit eine Reduktion seiner Leistung herbeiführt (Kluger & DeNisi, 1996). Ein moderierender Einfluss wurde auch bei der Verwendung von Rückmeldungen zur Festlegung von Beförderungen oder bei der Entlohnung beobachtet (DeNisi & Kluger, 2000). Für den überwiegen Teil der TeilnehmerInnen an Open-Source-Projekten existieren solche Formen der Belohnung nicht und entsprechend positiv sollte sich damit die Rückmeldung für diese Gruppe auf die Leistung auswirken. Ein wichtiger Moderator ist das Setzen von Zielen (goal setting), welches nach Locke und Latham (1990) eine entscheidende Voraussetzung für die Wirksamkeit von Rückmeldung ist. Die sehr offene Handhabung von Abgabeterminen und die begrenzten Einflussmöglichkeiten von Führungspersonen lassen vermuten, dass Zielsetzung in Open-Source-Projekten weniger ausgeprägt ist als in proprietären Projekten.

Rückmeldung durch Personen genießt in der Open-Source-Softwareentwicklung einen hohen Stellenwert und ist eine der zentralen Eigenschaften des Open-Source-Prozesses (Robbins, 2004). Da der Quellcode öffentlich zugänglich ist, bietet dieses vielen Personen die Möglichkeit der Rückmeldung (Barnes, 2003). In der proprietären Softwareentwicklung ist dagegen der Einblick in den Quellcode nur einem kleinen Kreis von Personen vorbehalten (Franck & Jungwirth, 2002a), wodurch deutlich weniger Personen die Möglichkeit zur Rückmeldung haben. Open-Source-Entwickler erhalten ihre Rückmeldung zumeist aus Meldungen über Fehler und aus der anschließenden Diskussion in Foren oder Mailinglisten (Krogh et al., 2003a). Dabei profitieren Open-Source-Softwareentwickler davon, dass neue Programmteile schnell in das bestehende Projekt integriert und neue Versionen häufig veröffentlicht werden (Raymond, 1999a; Robles, 2004; Rothfuss, 2002). In manchen Projekten verläuft der Veröffentlichungsprozess so schnell, dass die Entwickler täglich Rückmeldung erhalten (Spaeth, 2003). Häufige Veröffentlichung wird zunehmend auch in der proprietären Softwareentwicklung praktiziert (z. B.

bei Microsoft bei der Entwicklung von Windows 7); insbesondere neue Entwicklungsmodelle wie Agile Softwareentwicklung zeichnen sich hierdurch aus (Sillitti & Succi, 2005), jedoch scheint der Veröffentlichungszyklus im Allgemeinen noch deutlich länger zu sein.

Die Quantität von Rückmeldungen in Open-Source-Projekten wird ein Problem, wenn (vor allem leitende) Entwickler zur Bewältigung der Rückmeldungen zunehmend mehr Zeit aufwenden müssen (Peyrache et al., 2000). Der Nutzen der Rückmeldung in Open-Source-Projekten wird weiter durch die nicht immer hohe Qualität der Rückmeldung eingeschränkt. Wie auch andere Gemeinschaften, die auf computervermittelte Kommunikation angewiesen sind, werden auch in der Open-Source-Softwareentwicklung feindselige Diskussionen (vgl. Kapitel 3.2.2) beobachtet. Entsprechende Verhaltensregeln fordern die TeilnehmerInnen zu "positiven" als auch "negativen" Rückmeldungen auf. Dagegen sollen "demotivierende" Rückmeldungen (Beschimpfungen, Beschuldigungen oder Aufforderungen zu Verbesserungen) vermieden werden (Shah, 2003). Nicht in allen Bereichen gibt es zudem gleich viel und gleich gute Rückmeldungen. Während Kommentare zu konkreten Quelletextabschnitten relativ einfach zu bekommen sind, tut sich die Entwicklergemeinschaft mit Rückmeldungen zu abstrakteren Designfragen eher schwer (Jorgensen, 2001).

Wie oben beschrieben, kann Rückmeldung in Open-Source-Projekten bei einzelnen Entwicklern sehr umfangreich und damit belastend sein. Zudem kann es zu typischen Auswüchsen computervermittelter Kommunikation kommen und Rückmeldungen können unsachlich oder sogar verletzend sein. Dies lässt darauf schließen, dass das Ausmaß an Rückmeldungen durch Personen in Open-Source-Projekten geringer ist als in Projekten der proprietären Softwareentwicklung.

4.4.3 Interdependenz der Aufgabe und der Ziele

Sobald mehr als eine Person in die Erfüllung einer Aufgabe involviert ist, entstehen zwischen den Beteiligten Abhängigkeiten. Von Interdependenzen einer Tätigkeit spricht man, wenn die Ergebnisse einer Tätigkeit von einer anderen Tätigkeit abhängig sind (Kiggundu, 1983). Schon seit den Untersuchungen des Tavistock-Instituts ist bekannt, dass Abhängigkeiten Einfluss auf die Gesundheit, die Moral, die Fehltage und die Produktivität haben (Trist & Bamforth, 1951; Trist, Higgins, Murray & Pollock, 1963). Diese frühen Studien machten die Interdependenzen des Einzelarbeitsplatzes zum Gegenstand ihrer Untersuchungen. Durch die zunehmende Popularität von Gruppenarbeit werden noch andere Formen von Interdependenz relevant. Dabei ist Interdependenz häufig die Ursache für Gruppenarbeit und eines ihrer definierenden Merkmale (zusammenfassend Campion et al., 1993). Neben der Interdependenz der Aufgabe können in Gruppen auch Interdependenzen zwischen den Zielen und der Be-

lohnung vorkommen (Wageman, 2001). Da es Hinweise gibt, dass auch in Open-Source-Projekten Gruppenarbeit vorkommt (Hertel et al., 2003), kann davon ausgegangen werden, dass zumindest für einen Teil der Softwareentwickler auch eine Betrachtung der gruppenspezifischen Interdependenzen relevant ist. Von den drei Aspekten der Interdependenz (Aufgaben-, Ziel- und Belohnungsinterdependenz) soll die Interdependenz der Belohnung aufgrund der geringen Bedeutung extrinsischer Anreize im Open-Source-Kontext nicht weiter betrachtet werden.

Interdependenz hat sowohl bei Einzelarbeitsplätzen (Kiggundu, 1983) als auch in einer Gruppensituation (Campion et al., 1993) einen positiven Einfluss auf die Motivation. Auch einen positiven Einfluss auf Leistung konnte für den Prädiktor Interdependenz gezeigt werden (Campion et al., 1993), für die einzelnen Bestandteile (Aufgabeninterdependenz und Interdependenz der Ziele) konnte dies jedoch nicht in allen Fällen bestätigt werden (Campion et al., 1993; Campion et al., 1996). Es gibt Hinweise darauf, dass die Interdependenz der Aufgaben auch negative Auswirkungen haben kann: Sowohl bei dem Kriterium Motivation (Wong & Campion, 1991) als auch bei der Zufriedenheit (Corbett, Martin, Wall & Clegg, 1989) führte sehr hohe Interdependenz zu einer Abnahme. Hohe Interdependenzen könnten zu sehr eingeschränkten Tätigkeiten mit nur wenig stimulierender Wirkung führen. Aus der Sicht der Gesamtorganisation betrachtet, ist die größte Herausforderung bei Tätigkeiten mit hoher Interdependenz der hohe Koordinationsaufwand. Aufgaben mit hoher Interdependenz verfügen nur über eine geringe Autonomie (Thompson, 1967).

Theoretische Betrachtungen (Brooks, 1982; Pressman, 2000) und empirische Ergebnisse (Glass, 1997; Solheim & Rowland, 1993) weisen darauf hin, dass in der Softwareentwicklung Abhängigkeiten zwischen Tätigkeiten schwieriger zu beherrschen sind als bei anderen vergleichbaren Aufgaben. Auch von den Softwareentwicklern wird ein hohes Ausmaß an Aufgabeninterdependenz wahrgenommen (Brodbeck, 1994b). Die Koordination der Teilaufgaben hat einen hohen Kommunikationsaufwand zur Folge, der hauptsächlich in Form von persönlicher (Face-to-Face-)Kommunikation abgewickelt wird (Curtis et al., 1988; Hauptman, 1986; Kraut & Streeter, 1995; Scacchi, 1984; Shneiderman, 1980).

Da in der Open-Source-Softwareentwicklung vergleichbare Produkte mit vergleichbarer Komplexität entwickelt werden, könnte man annehmen, dass die entstehenden Interdependenzen der Aufgaben ebenfalls vergleichbar sind. Für geringere Interdependenz spricht dagegen die höhere Modularität in Open-Source-Projekten (vgl. Kapitel 3.2.6); diese sollte insbesondere zu einer Verringerung der Aufgaben-Interdependenz führen (Baldwin & Clark, 2003; Iannacci, 2005; Johnson, 2001b; Kaisla, 2001; Narduzzo & Rossi, 2003). Die Entwickler sind dadurch weniger auf die Arbeitsergebnisse von anderen TeilnehmerInnen angewiesen (sequenzielle Interdependenz) und auch die wechselseitige

Abhängigkeit ist reduziert. Der Einzelbeitrag steht vermehrt in Beziehung zum Endergebnis, das sich aus den Teilergebnissen (additive Interdependenz, pooled/additive interdependence) zusammensetzt (Bonaccorsi & Rossi, 2003; Crowston & Scozzi, 2002a). Bei kleinen Gruppen bedeutet das ausschließliche Vorliegen von additiver Interdependenz, dass im eigentlichen Sinne keine Interdependenz vorliegt (Wageman, 2001).

Entwickler in Open-Source-Projekten verfolgen gemeinsame Ziele, wie die Gewährleistung von Diversität der Software oder die Erstellung von Software zur Erfüllung eigener Bedürfnisse (Hertel et al., 2003). An der Vielzahl von Motiven der einzelnen Entwickler (vgl. Ghosh et al., 2002; Hars & Ou, 2002; Hertel et al., 2003; Lakhani & Wolf, 2003) wird deutlich, dass nicht für alle Beteiligten identische Ziele existieren müssen. So können auch einzelne Ziele in Konflikt miteinander stehen: Das Interesse, möglichst viel in einem Projekt zu lernen, ist zeitaufwendig und steht damit im Konflikt zu dem Wunsch, möglichst schnell eine funktionsfähige Software zu erhalten. Im Gegensatz dazu ist die Organisation in der proprietären Softwareentwicklung stärker bemüht, alle TeilnehmerInnen auf gemeinsame Ziele auszurichten und kann dieses durch den Einsatz von Autorität besser erreichen (O'Mahony & Ferraro, 2004). Das geringere Ausmaß an Zielkohäsion im Open-Source-Bereich verringert die Möglichkeit des Aufstellens von gemeinsamen Gruppenmaßstäben. Das Fehlen gemeinsamer Gruppenmaßstäbe hat zur Folge, dass die Leistung der Gruppe nicht an einem solchen Maßstab gemessen werden kann, was zu einer geringeren Interdependenz der Ziele führt (Wageman, 2001).

4.4.4 Zusammenfassung soziale Aspekte der Tätigkeit

Wie im vorangegangenen Abschnitt gezeigt, ist bei der Tätigkeit von Open-Source-Softwareentwicklern von einem geringen Ausmaß an Rückmeldung durch andere Personen, an sozialer Unterstützung sowie Aufgaben- und Zielinterdependenz auszugehen als bei Tätigkeiten im Bereich der proprietären Softwareentwicklung. Dieses führt zu der zweiten Hypothese:

> *Hypothese 2: Die sozialen Aspekte ihrer Tätigkeitsmerkmale wird von Open-Source-Softwareentwicklern niedriger eingestuft als von Beschäftigten im Bereich der proprietären Softwareentwicklung.*

Neben der Komplexität der Tätigkeit und den sozialen Aspekten sind noch zwei Aspekte identifiziert worden, die sich nicht eindeutig einer dieser beiden Kategorien zuordnen lassen. Diese beiden Aspekte werden im anschließenden Kapitel diskutiert.

4.5 Organisationale Kriterien

Eine Reihe von Feldexperimenten konnte aufzeigen, dass einzelne Tätigkeitsmerkmale in einer kausalen Beziehung zu verschiedenen organisationalen Kriterien wie Arbeitszufriedenheit, Job-Involvement, intrinsische Motivation oder Arbeitsplatzwechsel stehen (Griffeth, 1985; Griffin, 1983; Locke, Sirota & Wolfson, 1976; Orpen, 1979). Wie die Forschung zu Tätigkeitsmerkmalen ist auch die Forschung zu den Auswirkungen von Arbeit von motivationalen Theorien, insbesondere dem JCM, geprägt (Parker & Wall, 1998). Das JCM enthält sowohl Auswirkungen auf die Einstellung als auch auf das Verhalten. Zu den Einstellungsvariablen gehören die allgemeine Arbeitszufriedenheit, die intrinsische Motivation und die Zufriedenheit mit den Entfaltungsmöglichkeiten. Die Verhaltensvariablen umfassen die Arbeitsleistung, die Fehlzeiten und die Fluktuation (Hackman & Oldham, 1976). In nachfolgenden Untersuchungen wurden z. T. alternative oder zusätzliche Auswirkungen aufgenommen, so z. B. Mitarbeiter-Commitment und Zufriedenheit mit der Bezahlung oder den Vorgesetzten (Spector, 1985).

Im Unterschied zu vorherigen und vielen nachfolgenden Untersuchungen zu Tätigkeitsmerkmalen beinhaltet das JCM ein Wirkungsgefüge, das den Zusammenhang zwischen Tätigkeitsmerkmalen und Auswirkungen erklären soll. Zur Erklärung werden drei kritische psychologische Erlebniszustände (erlebte Bedeutsamkeit der Arbeit, erlebte Verantwortung für die Arbeitsergebnisse, Wissen um die Ergebnisse der Arbeit) herangezogen. Diese Erlebniszustände stehen als intermedierende Konstrukte zwischen den Tätigkeitsmerkmalen und den Auswirkungen der Arbeit. Das Bedürfnis nach persönlicher Entfaltung hat die Funktion eines Moderators (Hackman & Oldham, 1976).

Empirische Untersuchungen des Modells haben nicht alle prognostizierten Relationen der kritischen psychologischen Erlebniszustände bestätigen können. So zeigte Ganzheitlichkeit einen negativen Zusammenhange mit erlebter Bedeutsamkeit (Miner, 1980), keine Beziehung zu einem der Erlebniszustände (Fox & Feldman, 1988; Hackman & Oldham, 1976) oder einen ausgeprägteren Zusammenhang zu einem der anderen Erlebniszustände (Arnold & House, 1980; Fried & Ferris, 1987; Hackman & Oldham, 1976). Autonomie (Arnold & House, 1980; Fried & Ferris, 1987) und Rückmeldung (Fried & Ferris, 1987) zeigten gleich stark ausgeprägte Relationen zu allen drei Erlebniszuständen. Udris und Rinmann (1999) gehen davon aus, dass die kritischen psychologischen Erlebniszustände nicht von subjektiv wahrgenommenen Merkmalen der Arbeit zu unterscheiden sind. Von vielen Forschern wurden die Variablen entweder ignoriert oder, in Abwandlung des JCM, als abhängige Variablen genutzt (zusammenfassend Wall & Martin, 1994). Auch in der vorliegenden Arbeit sollen daher die kritischen Erlebniszustände nicht aufgenommen werden, und es wird ein direkter Zusammenhang zwischen Gestaltung der Tätigkeit und den organisationalen Kriterien postuliert. Trotz

dieser Unzulänglichkeiten des JCM konnten für die Tätigkeitsmerkmale positive Beziehungen zu verschiedenen Auswirkungen der Arbeit nachgewiesen werden (Fried & Ferris, 1987; Loher et al., 1985). Für die Tätigkeitsmerkmale des JCM, aber auch für die meisten anderen Tätigkeitsmerkmale, wird ein linearer Zusammenhang mit den organisationalen Kriterien implizit oder explizit postuliert. Dazu im Widerspruch stehen die Ergebnisse von DeJonge und Schaufeli (1998), die auf einen nicht-linearen Zusammenhang zwischen Autonomie und sozialer Unterstützung mit den organisationalen Kriterien Arbeitszufriedenheit, Angst und emotionale Erschöpfung hinweisen. Auch ist es nicht gesichert, dass eine Verbesserung eines Tätigkeitsmerkmals zwangsläufig zu einer Erhöhung der organisationalen Kriterien führen muss, sondern auch zu einer Verringerung führen kann. Die Ergebnisse von Xie und Johns (1995) und Warr (1987) deuten solche nicht stetigen Beziehungen zu organisationalen Kriterien an. In beiden Untersuchungen zeigte sich eine umgekehrte U-Funktion für das Merkmal Handlungsspielraum (job scope, Xie & Johns, 1995) und für das Merkmal Autonomie (Warr, 1987).

In Untersuchungen im Bereich der Softwareentwicklung dominieren auch bei den organisationalen Kriterien diejenigen, die im JDS berücksichtigt werden. Daneben sind noch eine Reihe zusätzlicher oder alternativer Kriterien aufgenommen worden (vgl. Tabelle 6).

Tabelle 6: Erhobene Auswirkungen der Arbeit aus Tätigkeitsanalysen in der Softwareentwicklung

Autoren	Verwendete/s Instrument/e	Erhobene Auswirkungen der Arbeit
(Brodbeck & Frese, 1994)	Eigenentwicklung	Stress, Gereiztheit/Belastbarkeit, Burn-out, Stolz auf die Leistung des Teams, Flow-Erleben
(Carayon et al., 2003)	JDS	Arbeitszufriedenheit, Arbeitsbelastung, Organisationale Identifikation, Organisationales Involvement
(Couger & Zawacki, 1980)	JDS	Allgemeine Arbeitszufriedenheit, soziale Arbeitszufriedenheit, Zufriedenheit mit dem Vorgesetzten, intrinsische Motivation
(Couger et al., 1989)	JDS	Zufriedenheit, Zielvereinbarung
(Goldstein & Rockart, 1984)	JDS und andere	Allgemeine Arbeitszufriedenheit, Zufriedenheit mit den Entfaltungsmöglichkeiten, Zufriedenheit mit den Kollegen und Vorgesetzten
(Goldstein, 1989)	JDS und Eigenentwicklung	Arbeitszufriedenheit
(Klemens, 2003)	SALSA	Beanspruchungspotenzial
(Lending & Chervany, 1997)	JDS und andere	Intrinsische Motivation, Zufriedenheit mit den Entfaltungsmöglichkeiten, allgemeine Arbeitszufriedenheit
(McKnight & Chervany, 1998)	JDS	Intrinsische Motivation, Arbeitszufriedenheit, Arbeitsleistung
(Wieland et al., 2004)	SynBA, SALSA	Kurzfristige Beanspruchungszustände, langfristige Beanspruchungsfolgen

Die Anzahl der vergleichenden Untersuchung von Tätigkeiten in der Softwareentwicklung mit anderen Tätigkeiten ist gering. Zumindest Wieland et al. (2004) konnte zeigen, dass das durchschnittliche Beanspruchungspotenzial in der IT-Branche keine auffälligen Abweichungen zu traditionellen Arbeitsplätzen zeigt.

Für den besonderen Kontext der Open-Source-Softwareentwicklung erscheinen neben der intrinsischen Motivation, Zufriedenheit und Leistung auch das Organizational Citizenship Behavior (OCB), das Flow-Erleben und die organisationale Identifikation von besonderem Interesse. In den folgenden Abschnitten werden diese Konzepte beschrieben und die Beziehungen zu proprietärer und Open-Source-Softwareentwicklung dargelegt.

4.5.1 Intrinsische Motivation

Intrinsische Motivation ist sowohl in der Forschung zu Tätigkeitsanalysen als auch in der Forschung zu Open-Source-Softwareentwicklung von besonderer Bedeutung. Intrinsische Motivation ist definiert:

> *"[...] as the doing of an activity for its inherent satisfactions rather than for some separable consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external prods, pressures, or rewards." (Ryan & Deci, 2000, S. 56)*

Bei intrinsischer Motivation liegen die motivierenden Aspekte in der Tätigkeit selbst. Von extrinsischer Motivation spricht man, wenn das Verhalten auf die Ergebnisse bzw. Folgen der Handlung ausgerichtet ist.

Hackman und Oldham (1976) postulieren in ihren JCM, dass insbesondere die von ihnen ausgewählten "Kern-Tätigkeitsmerkmale" einen wichtigen Einfluss auf intrinsische Motivation haben. Von den im JCM enthaltenen Tätigkeitsmerkmalen (Autonomie, Rückmeldung durch die Tätigkeit, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe und Vielfalt) weist in den vorliegenden Meta-Analysen die Vielfalt den stärksten Zusammenhang mit der intrinsischen Motivation auf (Fried, 1991; Loher et al., 1985). Auch Tätigkeitsmerkmale, die nicht im JCM enthalten sind, weisen Zusammenhänge mit der intrinsischen Motivation auf. Die Interdependenz von Tätigkeiten führt allgemein zu einer höheren Motivation (Kiggundu, 1983; Wong & Campion, 1991), während sehr hohe Interdependenzen die intrinsische Motivation verringern können (Corbett et al., 1989). Dies könnte daran liegen, dass ein extremes Ausmaß an Interdependenz zu Tätigkeiten mit sehr begrenzter Vielfalt führt (Morgeson & Campion, 2003).

Das Ausmaß der intrinsischen Motivation, das durch Tätigkeiten der Softwareentwicklung generiert wird, kann aufgrund von fehlenden Angaben in publizierten Studien nur geschätzt werden. Für die umfassendere Gruppe der Tätigkeiten in der Datenverarbeitung (data processing) liegen Daten aus der Studie von Couger und Zawacki (1980) vor. Diese weisen auf der Skala der intrinsischen Motivation einen Mittelwert von 5,7 aus. Mit demselben Instrument wurde auch für andere Tätigkeiten die intrinsische Motivation erhoben. Die dabei vorgefundenen Mittelwerte sind allgemein niedriger, nur die Tätigkeiten von Experten und Technikern (professionals, technicans) weisen einen höheren Mittelwert (5,8) auf, der Mittelwert von Verkaufs- und Servicepersonal (sales, service) ist mit 5,7 ebenso hoch wie der von Datenverarbeitern (Hackman & Oldham, 1980). In der Stichprobe von Couger und Zawacki (1980) sind auch weniger qualifizierte und entsprechend weniger anspruchsvolle Tätigkeiten enthalten (z. B. Bedienpersonal), die eine geringere intrinsische Motivation aufweisen sollten als solche in der Softwareentwicklung (Programmierer, Analysten). Dass solche Unterschiede vorliegen, ist daran zu erkennen, dass die Tätigkeit des Systementwicklers im Vergleich zu Personen aus anderen Bereichen der Datenverarbeitung eine relativ hohe intrinsische Motivation enthält, während Personen, die mit dem Testen von Software beschäftigt sind, eine geringere intrinsische Motivation aufweisen (Lending & Chervany, 1997). Eine hohe intrinsische Motivation weisen in EDV-Abteilungen vor allen solche Tätigkeiten auf, die

über ein hohes Ausmaß an Autonomie verfügen, während Kontrolle durch den Vorgesetzten die extrinsische Zufriedenheit erhöht (Santana & Robey, 1994).

Während für Erwerbstätige mit der Lohnzahlung zumindest ein extrinsischer Anreiz existiert, sind Tätigkeiten im Bereich der Freiwilligenarbeit wesentlich stärker auf die intrinsische Motivation angewiesen (Mieg & Wehner, 2002). Schneidewind, Landsberger und Eggers (2002) bezeichnen die Entwicklung von Linux "als ein Paradebeispiel für die produktiven Potenziale eines intrinsisch motivierten und virtuell koordinierten Arbeitseinsatzes in der Internet-Ökonomie". Aus bisher vorliegenden Ergebnissen geht hervor, dass intrinsische Motive in jedem Fall eine große Bedeutung in der Open-Source-Softwareentwicklung haben. Hars und Ou (2000) identifizierten in ihrer empirischen Untersuchung intrinsische Motivation als einen der wichtigsten Gründe und Lakhani und Wolf (2003) identifizieren sie sogar als den wichtigsten Grund für das Engagement von Open-Source-Entwicklern. Von den TeilnehmerInnen in Open-Source-Projekten geben 41,8 % an, dass sie das Programmieren im Open-Source-Bereich stimulierend finden (Bosco, 2004). Luthiger (2005) konnte zeigen, dass mit dem Konstrukt Spaß zwischen 27 und 30 % der Varianz am Ausmaß des Engagements in Open-Source-Projekten erklärt werden konnte.

Auch in der Open-Source-Softwareentwicklung existieren extrinsische Anreize, wie z. B. der Bedarf an Software und die Möglichkeit, die Gemeinschaft oder potenzielle Arbeitgeber auf die eigenen Fähigkeiten aufmerksam zu machen. Im Gegensatz zu proprietärer Softwareentwicklung und anderen Bereichen von Erwerbstätigkeit entfällt mit der Entlohnung ein wichtiger extrinsischer Anreiz für einen großen Teil der Open-Source-Softwareentwickler. Zumindest für diese Gruppe kann man negative Auswirkungen durch extrinsische Anreize ausschließen. In über hundert Experimenten wurde untersucht, ob die intrinsische Motivation durch extrinsische Anreize korrumpiert wird. Auch wenn in vorliegenden Meta-Analysen nicht gezeigt werden konnte, dass extrinsische Anreize im Allgemeinen einen senkenden Einfluss auf die intrinsische Motivation haben (Cameron, Banko & Pierce, 2001), so gibt es eine Reihe von Anreizformen, bei denen das sehr wohl beobachtet wurde (vgl. Meta-Analysen von Deci et al., 1999; Rummel & Feinberg, 1988; Tang & Hall, 1995; Wiersma, 1992). Extrinsische Anreize zeigen bei erwarteten materiellen Anreizen negative Auswirkungen auf die intrinsische Motivation (Cameron et al., 2001; Deci et al., 1999). Insbesondere solche Belohnungen, die als Gegenleistung nur die Ausführung der Tätigkeit erfordern, zeigen einen signifikanten korrumpierenden Effekt (Cameron et al., 2001; Deci et al., 1999). In diese Kategorie fällt z. B. der Monatslohn der proprietären Softwareentwickler. Positive Auswirkungen von extrinsischen Anreizen wurden dagegen bei verbalen Belohnungen (Cameron et al., 2001; Deci et al., 1999) und bei wenig interessanten Tätigkeiten beobachtet, in welche die Tätigkeit der Softwareentwicklung nicht einzuordnen ist.

Einige Eigenschaften von Open-Source-Projekten (vgl. Kapitel 3.2) stimmen mit den Voraussetzungen für intrinsische Motivation gut überein: die Freiheit zu wählen (Zuckerman, Porac, Lathin & Deci, 1978), das Fehlen von Abgabeterminen (Amabile, DeJong & Lepper, 1976), keine Bewertungen (Smith, 1975 zitiert nach Deci et al., 1999), die Abwesenheit von Zielvorgaben (Mossholder, 1980), geringer externer Druck (Stukas et al., 1999) und keine kontrollierende Funktion durch Wettbewerb (Reeve & Deci, 1996). Durch die selbstständige Auswahl von Tätigkeiten in Open-Source-Projekten wird die Autonomie nicht eingeschränkt und ermöglicht so eine hohe intrinsische Motivation (Osterloh et al., in Druck). Ebenso sollte sich Rückmeldung durch das Peer-Review positiv auf die intrinsische Motivation auswirken (Deci et al., 1999). Durch eine freie Auswahl an Tätigkeiten kann man im Open-Source-Umfeld sicher auch von einer sehr vielfältigen Tätigkeit ausgehen. Vielfalt sollte für die intrinsische Motivation von besonderer Bedeutung sein, da es von den fünf Tätigkeitsmerkmalen des JCM den stärksten Zusammenhang mit der intrinsischen Motivation aufweist (Fried & Ferris, 1987). Insgesamt muss man damit in der Open-Source-Softwareentwicklung von einer höheren intrinsischen Motivation als in der proprietären Softwareentwicklung ausgehen.

4.5.2 Allgemeine Arbeitszufriedenheit

Mit der Motivation ist die Arbeitszufriedenheit eng verbunden (Schuler, 1995), jedoch gilt die Verbesserung der Arbeitszufriedenheit seit den Hawthorne-Studien als eigenständiges Zielkriterium der Arbeitspsychologie (Schuler, 1995). Neben der intrinsischen Motivation ist die Arbeitszufriedenheit eines der von Hackman und Oldham (1976) im JCM berücksichtigten organisationalen Kriterien. Dass Arbeitszufriedenheit ein wichtiges Konstrukt darstellt, sieht man nicht nur an der Vielzahl der hierzu durchgeführten Untersuchungen, sondern auch an der relativ hohen Korrelation mit Leistung und Fluktuation. So beträgt die Korrelation von Arbeitszufriedenheit mit Leistung 0,3 (vgl. Meta-Analyse von Judge, Thoresen, Bono & Patton, 2001).

Von den unterschiedlichen Faktoren, die auf die Arbeitszufriedenheit einwirken, hat insbesondere die Komplexität der Tätigkeit einen wichtigen Einfluss. So konnte Stone (1986) in einem Review empirischer Studien einen starken Zusammenhang zwischen der Komplexität von Tätigkeiten und der Arbeitszufriedenheit für Feld- (r=0,63) und Laborstudien (r=0,53) zeigen. In einer Meta-Analyse, die hauptsächlich Laboruntersuchungen umfasste, fanden Loher et al. (1985) etwas schwächere Zusammenhänge (r=0,53). Insbesondere Rückmeldung durch die Tätigkeit und Autonomie zeigen eine besonders hohe Korrelationen mit der Arbeitszufriedenheit (Fried & Ferris, 1987). Neben der Komplexität der Tätigkeit haben auch soziale Aspekte einen Einfluss auf die Zufriedenheit. In einer Studie von Campion et al. (1993) zu Tätigkeitsmerkmalen in Arbeitsgruppen stellte sich soziale Unterstützung als das Merkmal mit der stärksten Korrelation

zu Arbeitszufriedenheit (Mitarbeiterstichprobe r=0,48; Führungskräftestichprobe r=0,28) heraus. Aktuellere meta-analytische Ergebnisse können diesen Zusammenhang für soziale Unterstützung durch Vorgesetzte und Arbeitszufriedenheit (Korrelation von 0,52) und soziale Unterstützung durch Kollegen (Korrelation von 0,37) bestätigen (Ng & Sorensen, 2008). Weiterhin zeigen freundschaftliche Beziehungen einen positiven Einfluss auf die Arbeitszufriedenheit (Keller, 1983; Martin & Hunt, 1980; Mueller & Price, 1990; Nicholson, 1977; Price & Mueller, 1986b). Merkmale der Komplexität der Arbeit (wie Selbststeuerung, Vielfalt und Bedeutsamkeit der Aufgabe) wiesen in dieser Untersuchung eine geringere Korrelation zur Arbeitszufriedenheit auf. Nicht eindeutige Ergebnisse ergaben sich für die Interdependenz der Ziele und Aufgaben; es wurde nur für Mitarbeiter ein positiver Zusammenhang gefunden, jedoch kein Zusammenhang für Führungskräfte (Campion et al., 1996).

Insgesamt weisen Personen, die im Bereich Softwareentwicklung eine Tätigkeit ausüben, eine hohe Arbeitszufriedenheit auf (Couger & Zawacki, 1980). Die Tätigkeitsmerkmale Autonomie (Goldstein & Rockart, 1984; Thatcher, Stepina & Boyle, 2002b) sowie Vielfalt und Bedeutsamkeit der Aufgabe (Thatcher et al., 2002b) sind die einflussreichsten Prädiktoren. Im Vergleich zu den Analysten weist die Gruppe der Programmierer eine hohe Zufriedenheit auf (Couger & Zawacki, 1980). Die Zufriedenheit der TeilnehmerInnen in Open-Source-Projekten wurde für das Projekt Linux-Kernel von Hertel et al. (2003) untersucht. Dabei wurde die Zufriedenheit im eigenen Entwicklungsbereich mit den Kommunikationsprozessen, der Arbeitsatmosphäre, der Anerkennung der Leistung und den Ergebnissen des Entwicklungsbereichs zu einer Skala zur Zufriedenheit kombiniert. Bei Werten in einen Bereich von 1 bis 5 und einem Mittelwert von 4,4 kann die Zufriedenheit als hoch bewertet werden. Die Zufriedenheit mit der Gesamtentwicklung des Linux-Kernels war mit einen Mittelwert von 4,2 nicht wesentlich geringer. Mit nur einer Studie über ein einzelnes Projekt ist die empirische Basis unzureichend, aber ein hohes Maß an Zufriedenheit auch für andere Projekte scheint wahrscheinlich.

4.5.3 Leistung

Im Vergleich zu Einstellungsvariablen, wie der intrinsischen Motivation oder Arbeitszufriedenheit, kann bei Verhaltensvariablen, wie dem Leistungsverhalten, nur ein geringerer Anteil der Varianz durch die Tätigkeitsmerkmale erklärt werden (Fried & Ferris, 1987). Insgesamt sind die Zusammenhänge zwischen den Tätigkeitsmerkmalen und der Leistung schwach und inkonsistent (Arnold & House, 1980; Cherrington & England, 1980; Gorn & Kanungo, 1980; Griffin, Welsh & Moorhead, 1981). Wie bei der Arbeitszufriedenheit ist von den fünf Tätigkeitsmerkmalen von Hackman und Oldham (1976) auch bei der Leistung die Rückmeldung durch die Tätigkeit zwar ein einflussreiches, aber nur das zweiteinflussreichste Merkmal. Den stärksten Zusammenhang

weist die Aufgabengeschlossenheit auf. In der Meta-Analyse bestimmte Fried (1991) sowohl für die Aufgabengeschlossenheit (0,13) als auch die Rückmeldung durch die Tätigkeit (0,09) eine geringe Korrelation mit der Leistung. Einen positiven Einfluss von Komplexität konnte Stone (1986) bei seinem Review nur bei Feldexperimenten bestimmen (r=0,30), während die drei betrachteten Laborstudien eine negative Korrelation (r=-0,26) aufwiesen. In einer Untersuchung von Campion et al. (1993) zu Arbeitsgruppen wurden sowohl Merkmale der Komplexität als auch soziale Aspekte erfasst. Unabhängig davon, ob die Gruppenleistung in Form einer Selbsteinschätzung durch die Mitarbeiter oder durch die Führungskräfte erfasst wurde, zeigten Aspekte wie soziale Unterstützung, das Teilen von Arbeitsbelastung, Kommunikation und Kooperation deutlich höhere Korrelationen als Merkmale der Komplexität.

Sowohl in der Forschung als auch in der Praxis der Softwareproduktion werden zwei Aspekte der Leistung intensiv diskutiert: zum einen die Qualität des Endproduktes und zum anderen der betriebene Aufwand für die Erstellung. Mit dem Aspekt der Qualität ist die Frage verbunden, ob ein Produkt den Vorgaben des Herstellers und/oder den Kunden entspricht (Samoladas & Stamelos, 2005). Eine häufig verwendete Auflistung an Vorgaben entstammt der ISO 9216 (International Organization for Standardization, 1991) und beschreibt in einem hierarchischen Modell die sechs wichtigsten Aspekte von Softwarequalität: Funktionalität, Zuverlässigkeit, Benutzerfreundlichkeit, Wartbarkeit, Portabilität und Effizienz.

Die Frage nach der Qualität von Open-Source-Software im Vergleich zu proprietärer Software kann aufgrund des Fehlens von entsprechenden gesicherten Erkenntnissen nicht beantwortet werden. Die Selbsteinschätzung einer Mehrheit von Open-Source-Entwickler (63 %) ist, dass Open-Source-Software eine höhere Qualität als proprietäre Software aufweist (David et al., 2003). In einer weiteren Erhebung stuften 60 % der Open-Source-Entwickler die Qualität von Open-Source-Software als überlegen ein, während nur 2 % dies von proprietärer Software behaupteten (Ghosh et al., 2002). Bei dieser Befragung können Tendenzen zur positiveren Darstellung von Open-Source-Software nicht ausgeschlossen werden, jedoch hat zumindest ein großer Teil (mehr als 50 %) der TeilnehmerInnen eine gute Vergleichsmöglichkeit, da sie neben der Open-Source- auch in der proprietären Softwareentwicklung arbeiten. Eine objektivere Aussage erlaubt der Vergleich der Fehlerhäufigkeit in Open-Source-Projekten. Diese Untersuchungen bescheinigen Open-Source-Projekten eine geringere Fehlerhäufigkeit (vgl. Kapitel 3.2.3). Auch eine Untersuchung zur Wartbarkeit hat zumindest keine Nachteile von Open-Source-Software aufgedeckt (Stamelos, Angelis, Oikonomu & Bleris, 2002). Als mögliche Ursache für die höhere Qualität wird von Anhängern des Open-Source-Modells die große Anzahl von Testern (Raymond, 1999a) und die schnelle Korrektur von Fehlern (z. B. Kuan, 2001) genannt.

Neben der Qualität ist auch die Produktivität und die Verbesserung derselben ein bedeutendes Thema in der Softwareentwicklung. Die Schätzung von Boehm (1987), wonach bereits im Jahr 1995 eine Effizienzsteigerung von 20 % in der Softwareentwicklung eine weltweite Kostenreduktion von 90 Milliarden US-Dollar zur Folge gehabt hat, zeigt die Bedeutung dieses Themas. Ein wichtiger und regelmäßig untersuchter Einflussfaktor auf die Leistung in der Softwareentwicklung ist die Teamgröße. Bereits seit Ringelmann (1913) ist bekannt, dass sich ein größeres Team negativ auf die Leistung des Einzelnen auswirken kann. In der Softwareentwicklung wird dieses Thema seit der einflussreichen Publikation von Brooks (1982) über den "Mystischen Mann-Monat" diskutiert. Basierend auf praktischen Erfahrungen stellt Brooks heraus, dass der Abschluss eines Entwicklungsprojekts durch das Hinzufügen von zusätzlichen Teammitgliedern länger dauern kann als ohne diese Teammitglieder. Neben dem von Ringelmann gefundenen Abfall der individuellen Leistungsfähigkeit eines einzelnen Gruppenmitglieds, gibt es in der Softwareentwicklung die Situation, dass die Leistungsfähigkeit der Gesamtgruppe bei steigender Gruppengröße abnehmen kann. Eine Übersicht über den Einflussfaktor Teamgröße und andere Einflussfaktoren der Produktivität gibt Scacchi (1995). Die einflussreichsten Faktoren sind dabei die Komplexität der Software und Fähigkeiten der Entwickler (Boehm, 1981). Für eine rege Diskussion von Theoretikern und Praktikern aus der Softwareentwicklung sorgt die Frage des Einflusses von sozialer Interaktion auf die Leistung von Softwareentwicklern (Brodbeck, 2001). Eine Reihe von Autoren betrachtet die Zeit, die mit Kommunikation verbracht wird, als "nicht produktiv" und glaubt, dass diese damit zu geringerer Leistung führt (Brooks, 1982; Brooks, 1987; Sommerville, 1989). Dem widersprechen Untersuchungen aus dem durchaus vergleichbaren Bereich der Forschung und Entwicklung. Diese zeigen, dass Kommunikation einen positiven Einfluss auf die Leistung hat (Brown & Eisenhardt, 1995; Katz, 1982; Katz & Allen, 1985; Keller, 1994). In einer empirischen Untersuchung konnte Brodbeck (2001) die positiven Effekte von Kommunikation auch im Bereich der Softwareentwicklung bestätigen (Brodbeck, 2001). Dabei waren wichtige Moderatoren die Phase des Projektes im Lebenszyklus und der Einsatz von standardisierten Methoden und Werkzeugen. Projekte, die sich in späten Phasen des Lebenszyklus befinden und Projekte, die nur im geringen Umfang standardisiert vorgehen, profitieren am meisten von der Kommunikation innerhalb des Projektteams (Brodbeck, 2001). Die Bedeutung von Kommunikation in Open-Source-Projekten geht auch aus der Pfadanalyse von Stewart und Gosain (2003) hervor: Aufgabenbezogene Kommunikation führt zu größerem Vertrauen im Team, was wiederum eine erhöhte Effizienz zur Folge hat.

Die Frage nach der Effizienz von Open-Source-Softwareentwicklung hat zu kontroversen Diskussionen geführt (z. B. Bollinger et al., 1999; z. B. Wilson, 1999). Die Beurteilung der Effizienz von Open-Source-Softwareentwicklung wird durch den Charakter der Ent-

wicklung massiv erschwert, wenn nicht sogar unmöglich gemacht. Die Einsehbarkeit des Quellcodes und die automatische Dokumentation des Arbeitsfortschritts ermöglichen zwar eine Beurteilung des Endproduktes, jedoch bleibt der dafür betriebene Aufwand zum großen Teil nicht messbar. Durch die Selbstorganisation, die Dezentralität, die Parallelität und die Offenheit der Projekte ist es sehr aufwendig festzustellen, wie viele Personen mit welchem zeitlichen Aufwand zu einem Endprodukt beigetragen haben. Koch und Gonzalez-Barahona (2005) fragen hierzu:

> *"Is open source software development a way of producing better software more efficiently, or is an enormous effort just invisibly expended?" (Koch & Gonzalez-Barahona, 2005)*

Open-Source-Entwickler sind von der Produktivität ihrer Projekte durchaus überzeugt. In einer Befragung von David et al. (2003) gaben 59 % an, dass sie Open-Source-Entwicklung für effizienter als proprietäre Softwareentwicklung halten. In einer vergleichbaren Befragung von Ghosh et al. (2002) gaben 42 % an, dass Open-Source-Entwicklung effizienter ist. Ein Anteil von 12 % hielt die proprietäre Softwareentwicklung für effizienter. Für die Verteilung von Arbeitskräften im Open-Source-Projekt "Gnome" konnte gezeigt werden, dass diese einer Empfehlung für effiziente kommerzielle Projekte entsprach (Koch & Schneider, 2002). Ein objektiveres Maß für die Produktivität der Softwareentwickler stellt ein Vergleich der produzierten Menge an Quellcodes dar. Hierzu wurden von Mockus et al. zwei Studien vorgelegt (Mockus et al., 2000, 2002), die zeigen, dass zumindest die besten Entwickler des Open-Source-Projekts Apache eine vergleichbare Leistung wie Entwickler in proprietären Projekten aufweisen (vgl. Tabelle 7). Mit 4300 Zeilen pro Person und Jahr im Apache-Projekt würde dies der Produktivität in proprietären Projekten entsprechen, die in zahlreichen Studien mit einem Durchschnitt von 10 bis 20 Zeilen pro Tag und Person angegeben ist (MacCormack et al., 2005).

Tabelle 7: Vergleich bearbeitete Änderungsanforderungen und Anzahl Quellcodezeilen pro Jahr bei Open-Source- (beste Entwickler für Apache-Projekt) und proprietären Projekten (Mockus et al., 2000, 2002)

Projekt	Bearbeitete Änderungs-anfor-derungen*	Quellcode-zeilen*
Apache (Gesamt)	0.11	4.3
/layout	0.17	11
/js	0.13	16
/rdf	0.11	11
/network	0.13	8,4
/editor	0.09	8
/intl	0.08	7
/xpinstall	0.07	6
Proprietäres Projekt A	0.03	38.6
Proprietäres Projekt B	0.03	11.7
Proprietäres Projekt C	0.09	6.1
Proprietäres Projekt D	0.02	5.4
Proprietäres Projekt E	0.06	10.0

* In Tausend pro Entwickler pro Jahr

Eine überzeugende Untersuchung, die eine Vielzahl von Open-Source-Projekten einbezieht und die Frage löst, welche Entwickler in den Vergleich einbezogen werden, existiert nach Wissen des Autors noch nicht. Insbesondere die Auswirkungen von paralleler Entwicklung lassen Fragen nach der Effizienz aufkommen. So berichten nur 23 % der Entwickler des Linux-Kernel-Projekts, dass die von ihnen erstellten Beiträge für das Projekt auch in die offizielle Version aufgenommen wurden (Lee & Cole, 2003). Dies bedeutet im Umkehrschluss, dass die Bemühungen von 77 % der TeilnehmerInnen im schlechtesten Falle unnötig gewesen wären.[7] Neben der parallelen Entwicklung sollte auch die geografische Distanz zwischen den Entwicklern negative Auswirkungen auf die Produktivität haben (Espinosa et al., 2001). Zu den produktivitätsförderlichen Eigenschaften von Open-Source gehört die häufig zitierte Wiederverwertung von bereits erstellten Quellcodeteilen (Osterloh et al., in Druck). Studien aus dem Bereich proprietäre Softwareentwicklung berichten von einer Produktivitätssteigerung um 57 % und über eine um 25-57 % beschleunigte Produkteinführungszeit (Henry & Faller, 1995; Lim, 1994). Insgesamt geben verfügbare Daten Anlass zu der Vermutung, dass Open-Source-Entwickler mehr leisten als Entwickler in proprietären Projekten.

[7] Für den Beitrag von Parallelentwicklung für Qualität und Geschwindigkeit des Projektes vgl. Kapitel 3.2.6.

4.5.4 Organizational Citizenship Behavior

Bei einer zunehmend stärkeren Verbreitung von Organisationen, die weniger von hierarchischen Strukturen und mehr von den Entscheidungen von selbst gesteuerten Gruppen und Individuen abhängen, gewinnt das Organizational Citizenship Behavior (im Folgenden OCB) insgesamt zunehmend an Bedeutung (LePine, Erez & Johnson, 2002). Mit dem "Organisationsbürgerverhalten" sind solche Verhaltensweisen gemeint, die der Organisation nützlich sind, aber nicht direkt von dieser belohnt oder gefördert werden (Organ, 1988). Nach der überarbeiteten Definition von Organ ist das OCB solches Verhalten, das den sozialen und psychologischen Kontext fördert und aufrechterhält und damit die Arbeitsleistung unterstützt (Organ, 1997). Neben Prädiktoren für das OCB, die in der Persönlichkeit der Mitarbeiter liegen, wurden auch Eigenschaften der Führungspersönlichkeit, der Organisation und der Tätigkeit betrachtet. Trotz mäßiger Aufmerksamkeit und einer geringen Anzahl von Studien zeigten insbesondere Tätigkeitsmerkmale konsistente Zusammenhänge mit dem OCB und waren wichtige Determinanten des OCB (Podsakoff, MacKenzie, Paine & Bachrach, 2000). Dabei wurden signifikante Korrelationen zwischen verschiedenen Dimensionen des OCB und der Autonomie (Farh, Podsakoff & Organ, 1990; Todd, 2004), der Bedeutsamkeit der Aufgabe (Farh et al., 1990; Namm, 2004; Todd, 2004), der Rückmeldung durch die Aufgabe (Farh et al., 1990; Podsakoff & MacKenzie, 1995; Podsakoff, Niehoff, MacKenzie & Williams, 1993), der Vielfalt (Farh et al., 1990), der Anforderungsvielfalt (Namm, 2004), der Routinelastigkeit der Tätigkeit (Podsakoff & MacKenzie, 1995; Podsakoff et al., 1993; Todd, 2004) und den intrinsisch motivierenden Tätigkeiten (Podsakoff & MacKenzie, 1995; Podsakoff et al., 1993; Van Dyne, Graham & Dienesch, 1994) gefunden. Hierbei wiesen alle Merkmale – bis auf die Routinelastigkeit von Aufgaben – positive Zusammenhänge mit dem OCB auf. Für den Einfluss von Interdependenz liegen widersprüchliche Ergebnisse vor. So konnten von Smith, Organ und Near (1983) keine Zusammenhänge gefunden werden, während Pearce und Gregersen (1991) zeigten, dass Interdependenz einen Einfluss auf das Extra-Rollen-Verhalten ausübt, wenn sich der Mitarbeiter für die Aufgabe verantwortlich fühlt.

Im Vergleich zu anderen untersuchten Prädiktoren konnten Tätigkeitsmerkmale die OCB-Dimensionen Hilfsbereitschaft, Gewissenhaftigkeit und Eigeninitiative am besten prognostizieren (Podsakoff, MacKenzie & Bommer, 1996). Neben Merkmalen der Komplexität sind auch soziale Aspekte der Tätigkeit auf einen Zusammenhang mit OCB untersucht worden (Farh et al., 1990; Moorman, 1991; Moorman, Blakely & Niehoff, 1998; Settoon, Bennett & Liden, 1996; Shore & Wayne, 1993; Smith et al., 1983). Die Ergebnisse waren weniger eindeutig als für die Komplexitätsmerkmale: Es konnten keine konsistenten Zusammenhänge für soziale Unterstützung gefunden werden (Podsakoff et

al., 2000). Nur die Beziehung zwischen Vorgesetzten und Mitarbeitern zeigte einen starken Zusammenhang zum OCB (Podsakoff et al., 2000).

Ob Tätigkeitsmerkmale einen direkten Einfluss auf das OCB haben oder ob die Beziehung indirekter Natur ist, bleibt umstritten. Todd (2004) zeigt, dass ein Modell mit der Arbeitszufriedenheit als Mediator zwischen den Tätigkeitsmerkmalen und dem OCB eine gute Übereinstimmung mit seinen Daten zeigt. Cardona, Lawrence und Bentler (2004) finden für ein Modell mit Bindung (attachment) als Mediator eine bessere Übereinstimmung als für ein Modell mit direktem Zusammenhang zwischen Tätigkeitsmerkmalen und OCB. Allerdings wurden von ihnen die Eigenschaften der Tätigkeit mit nur einer Skala, die aus drei aus dem JDS entnommenen Items besteht, erhoben. Die Kürze und die geringe interne Konsistenz (α zwischen 0,52 und 0,58) könnten ursächlich dafür sein, dass durch Hinzunahme eines weiteren Merkmals (Bindung) mehr Varianz aufgeklärt wird. Für einen direkten Zusammenhang von Tätigkeitsmerkmalen sprechen die Ergebnisse von Farh et al. (1990). Diese stellten bei dem Modell mit Tätigkeitsmerkmalen als gemeinsame Ursache von Arbeitszufriedenheit und OCB die beste Übereinstimmung mit den erhobenen Daten fest. Nach Farh et al. (1990) vermittelt eine motivierende Tätigkeitsgestaltung zum einen das Gefühl der Verantwortlichkeit für ein Arbeitsergebnis, und zwar gleichgültig, ob es Teil der Aufgabenstellung ist oder nicht, und zum anderen führt das größere Verständnis der Abhängigkeiten zu einem umfassenden Einblick in die Bedürfnisse, Probleme und Perspektiven der anderen Akteure.

> *"In other words, an interesting task does not lead one to circumscribe more narrowly the sphere of personal accountability; rather, it heightens the importance of the broader context in which the task is performed." (Farh et al., 1990, S. 709)*

Dieses steht im Einklang mit den theoretischen Ansichten von Hackman und Oldham (1980). Für die Erklärung des Zusammenhangs von OCB und sozialen Faktoren wird das Konzept des sozialen Austausches (social exchange) von Blau (1964) angeführt. Dieses besagt, dass positive Interaktion (z. B. mit Führungskräften oder Mitarbeitern) zu unspezifischen Verpflichtungen führt, die durch das OCB beglichen werden können (vgl. Organ, 1988).

Erste Erkenntnisse über das OCB vom Mitarbeiter in der IT-Branche deuten darauf hin, dass von diesen weniger OCB gezeigt wird als in anderen Bereichen. Moore und Love (2005) zitieren die Daten einer Promotionsstudie, die von allen untersuchten Berufsgruppen in der IT-Branche die niedrigsten OCB-Werte gefunden hat. Bei der Erhebung waren die Wahrnehmung von wenig empfundener Fairness und geringes Vertrauen in den Vorgesetzten Auslöser für das geringe Ausmaß an OCB.

Aufgrund fehlender Vergleiche von OCB in unterschiedlichen Tätigkeitsfeldern sind nur begrenzte Aussagen über die Softwareentwicklung möglich. Moore und Love (2005) erwähnen eine Untersuchung, bei der Mitarbeiter in der IT-Branche weniger OCB zeigten als in anderen Unternehmensbereichen. Für die beiden in dieser Studie untersuchten Auslöser von OCB, Vertrauen in den Vorgesetzten und wahrgenommene Gerechtigkeit, wurden in der IT-Branche geringere Werte als in anderen Branchen gefunden. Als Ursachen vermuten Moore und Love (2005) die häufige Anwendung von modernen Arbeitsformen in der IT-Branche. Diese Arbeitsformen weisen eine größere Unabhängigkeit und weniger Kontakt von Mitarbeitern zu Vorgesetzten auf, was zu einer Reduktion des gegenseitigen Vertrauens führen soll. Insbesondere für das Hilfsverhalten existieren dokumentierte Beispiele, die zeigen, dass dieses von den Vorgesetzten in der IT-Branche ausgesprochen wenig gefördert wird (Perlow & Weeks, 2002).

Nach der Definition von Organ (1997) versteht man unter OCB Verhalten, das den sozialen und psychologischen Kontext fördert und aufrechterhält. Solches Verhalten ist auch für den Open-Source-Bereich relevant. Untersucht wurde das OCB im Open-Source-Bereich bisher nur in Form von "partizipativem Verhalten" (participative behavior), wovon ein substanzieller Anteil (R^2=0,50) von den Variablen Vergnügen, Identifikation, Verpflichtung und berufliche Vorteile vorhergesagt wird (Bo Xu, 2006). Constant, Sproull und Kiesler (1996) vermuten, dass Unterstützung in virtuellen Gemeinschaften auf OCB beruht, während Burroughs und Eby (1998) davon ausgehen, dass die virtuelle Verbundenheit OCB fördert. Auf den Teilaspekt von OCB, der Bereitschaft Hilfe zu leisten, sollte der freiwillige Charakter von Open-Source-Projekten einen positiven Einfluss haben (Stukas et al., 1999). Insgesamt betrachtet, kann man vermuten, dass in Open-Source-Gemeinschaften ein ausgeprägtes Organizational-Citizenship-Verhalten besteht. Da Open-Source-Gemeinschaften weniger formale Strukturen aufweisen und diese auch weniger systematisch zum direkten Belohnen und Fördern tendieren, sollte in Open-Source-Gemeinschaften ein größerer Teil des beobachtbaren Verhaltens auf Organizational-Citizenship-Verhalten beruhen, als dies in der proprietären Softwareentwicklung der Fall ist.

4.5.5 Flow-Erleben

Nach Csikszentmihalyi und Csikszentmihalyi (1991) beschreibt Flow einen Zustand des optimalen Erlebens mit vollständigem Aufgehen in der ausgeführten Handlung. Dieser Zustand ist mit ausgeprägter zielbezogener Aktivierung und Konzentration verbunden und führt zu Selbst- und Zeitvergessenheit. Die Aufmerksamkeit ist dabei völlig auf die Tätigkeit gerichtet, das Ziel der Handlungen und die Rückmeldungen sind deutlich und es besteht eine hohe Kompetenz und Kontrolle (Csikszentmihalyi, 1982). Als kennzeichnend für das Flow-Erlebnis gilt, dass ein Gleichgewicht zwischen hohen An-

forderungen und den eigenen Fähigkeiten wahrgenommen wird (Csikszentmihalyi & Csikszentmihalyi, 1991). Ebenso wie die intrinsische Motivation ist das Flow-Erleben ein Motivationskonzept, bei dem die Motivation in Tätigkeitsanreizen begründet ist. Die intrinsische Motivation betrachtet alle Tätigkeitsbereiche, während das Flow-Konzept auf einen Teilbereich fokussiert (Rheinberg, 2009). Damit stehen Flow-Konzept und intrinsische Motivation in einem engen Zusammenhang. Zu den Flow-förderlichen Tätigkeitsmerkmalen gehört neben der wahrgenommenen Herausforderung auch ein ausreichender Handlungsspielraum (Gail & Frese, 1994; Ghani & Desphande, 1994; Rau & Riedel, 2004). Auch die Tätigkeitsmerkmale Lernpotenzial, Verantwortung und Kooperation/Kommunikation haben einen positiven Einfluss auf das Flow-Erleben (Rau & Riedel, 2004).

Studien zu Flow-Zuständen bei der Nutzung von Computern (z. B. Konradt & Sulz, 2001; Webster, Trevino & Ryan, 1993), der Benutzung des Internets (z. B. Chen, Wigand & Nilan, 1999; Novak, Hoffman & Yung, 1999) oder bei der computervermittelten Kommunikation (z. B. Trevino & Webster, 1992) weisen darauf hin, dass die Tätigkeit der Softwareentwicklung ein häufiges Flow-Erlebnis erlaubt. Nach Schachtner (1994) erleben Programmierer immer wieder "lustvolle" Phasen des "spielerischen" Programmierens, die durch das Ausprobieren verschiedener Methoden geprägt sind. Dass diese Phasen mit dem Flow-Konzept gut charakterisiert werden, zeigen die Ergebnisse der Studie von Gail und Frese (1994) zur Softwareentwicklung. Über 80 % der Softwareentwickler hatten bereits ein Flow-Erlebnis während der Arbeitszeit, wobei 34 % Flow einige Male pro Woche, 5 % einmal pro Tag und 10 % mehrmals pro Tag aufwiesen. Die Häufigkeit der Flow-Erlebnisse in der Softwareentwicklung kann mit einem ausreichenden Handlungsspielraum und der hinreichenden Komplexität der Aufgaben erklärt werden (Gail & Frese, 1994). Dies wird durch die Ergebnisse von Rheinberg und Tramp (2006) bei Personen, die Computer intensiv in der Freizeit nutzen, bestätigt. Ein hohes Ausmaß an Flow-Erleben zeigte sich nur, wenn mit den Aktivitäten Herausforderungen und Kompetenzwachstum verbunden sind.

Das Flow-Konzept könnte auch einen wichtigen Erklärungsansatz für das freiwillige Engagement in Open-Source-Projekten darstellen: So fördert Flow das Experimentieren, das Explorationsverhalten und die freiwillige Beschäftigung mit der Flow-auslösenden Tätigkeit (Webster et al., 1993). Bei der Befragung von TeilnehmerInnen in Open-Source-Projekten gaben über 70 % an, dass sie "immer" oder "häufig" beim Programmieren die Zeit aus den Augen verlieren würden (Lakhani & Wolf, 2003). In einer Erhebung von Open-Source- und proprietären Softwareentwicklern mit selbst entwickelten Skalen zum Flow-Erleben hatten Open-Source-Entwickler auf allen Unterskalen höhere Werte, wobei dieser Unterschied ausschließlich für die Skala "Flow/Spaß" eine signifikante Abweichung aufwies (Luthiger, 2006). Dies deutet auf ein höheres Flow-Potenzial in Open-

Source-Projekten hin. Dass auch im Open-Source-Bereich die erlebte Herausforderung ein wichtiger Einflussfaktor ist, zeigt, dass von den untersuchten Merkmalen der Tätigkeit das Merkmal "optimale Herausforderung" die höchste Korrelation (0,41) zum Flow-Erlebnis aufwies (Luthiger, 2005). Das Ausmaß an Flow-Erleben in Open-Source-Projekten ist auch von der Art der Tätigkeit abhängig: Besonders ausgeprägt ist das Flow-Erleben bei Projektleitern und Entwicklern, während es bei Mitarbeitern, die hauptsächlich mit dem Korrigieren von Fehlern beschäftigt sind, weniger ausgeprägt ist (Luthiger, 2006). Als Ursache für das hohe Ausmaß an Flow-Erleben kommt neben der optimalen Herausforderung auch der intensive Einsatz von computervermittelter Kommunikation in Frage. So fördert die Nutzung von elektronischer Post (E-Mail) im Vergleich zu einem telefonbasierten Anrufbeantworter (Voicemail-System) das Flow-Erleben (Trevino & Webster, 1992). Auch flexiblere Software fördert das Flow-Erleben (Webster et al., 1993). Da in vielen Open-Source-Projekten als Werkzeuge hauptsächlich Open-Source-Produkte zum Einsatz kommen, welche möglicherweise flexibler als proprietäre Produkte sind (Franke & Hippel, 2003; Krishnamurthy, 2003), würde auch dies ein ausgeprägteres Flow-Erleben in Open-Source-Projekten erklären.

4.5.6 Organisationale Identifikation

Organisationale Identifikation kann durch die Schaffung von konvergenten Erwartungen die Koordination erleichtern (Kogut & Zander, 1996) und Organisationsmitglieder motivieren, gemeinsame Ziele anzustreben (Kramer & Brewer, 1984, 1986). Für die organisationale Identifikation gibt es verschiedene Definitionen, denen gemein ist, dass ein Organisationsmitglied seine Zugehörigkeit zu einer Organisation auf emotionale und/oder kognitive Art und Weise in sein Selbstkonzept aufgenommen hat (Riketta, 2005). Ashforth und Mael (1989) beschreiben dies als die Wahrnehmung der Einheit oder Zugehörigkeit mit einer Organisation. In virtuellen Organisationen kann die Identifikation einen wichtigen Faktor für den Zusammenhalt ausmachen (Wiesenfeld, Raghuram & Garud, 1999). Während in traditionellen Organisationen Eigenschaften wie z. B. die Kleiderordnung, gemeinsame Sprache oder Geschäftsabläufe, Organigramme, Gebäude oder benachbarte Mitarbeiter ein Gefühl von gemeinsamer Identität aufkommen lassen, sind solche Hinweise in virtuellen Organisationen weitaus weniger präsent (Wiesenfeld et al., 1999). Weil solche sichtbaren Gemeinsamkeiten weniger ausgeprägt sind, ist die soziale Kommunikation in virtuellen Organisationen eine wichtige Voraussetzung für die Entwicklung und Aufrechterhaltung einer gemeinsamen Identität (Stewart & Gosain, 2003). Beim Einsatz elektronischer Medien ist es insbesondere die Frequenz, die das Ausmaß an Identifikation stärkt (Cooney, 2003; Wiesenfeld et al., 1999).

Neben den Möglichkeiten zur Kommunikation sind auch andere Tätigkeitsmerkmale auf ihren Einfluss auf die organisationale Identifikation hin untersucht worden. In einer Meta-Analyse ergaben sich für das am häufigsten untersuchte Merkmal Handlungsspielraum/Herausforderung (job scope/challenge) mittlere Korrelationen (0,26 bis 0,33; je nach verwendetem Instrument) für den Zusammenhang mit der organisationalen Identifikation (Riketta, 2005). Besonders hohe Zusammenhänge ergaben sich für anspruchsvolle Tätigkeiten wie in Forschungs- und Entwicklungsabteilungen (Hall & Schneider, 1972). Für den Bereich Softwareentwicklung konnten Effekte in die gleiche Richtung nachgewiesen werden. Die Komplexität der Arbeit und der Mangel an Identifikation zeigen eine negative Korrelation (Sonnentag et al., 1994). Wenige Studien liegen für spezifischere Tätigkeitsmerkmale, wie Autonomie (z. B. Bell & Kozlowski, 2002; Brown, 1969; Hall & Schneider, 1972), zwischenmenschliche Interaktion (z. B. Brown, Condor, Mathews, Wade & Williams, 1986) und soziale Unterstützung (z. B. Benkhoff, 1997; Lee, 1971; z. B. Wiesenfeld, Raghuram & Garud, 2001), vor; diese weisen jedoch auf positive Zusammenhänge zur organisationalen Identifikation auf.

Sehr eng mit der organisationalen Identifikation ist das organisationale Commitment verwandt. Bei beiden handelt es sich um eine Form des "Attachment" zu einer Organisation (Van Dick, 2003). Im Konzept des organisationalen Commitments stehen die Akzeptanz von Zielen und Werten, der Wille zur besonderen Anstrengung sowie der starke Wunsch, ein Mitglied zu bleiben, im Vordergrund. Das Konzept der organisationalen Identifikation betont die Überlappung des eigenen Selbstbilds mit der Wahrnehmung der Organisation. Ergebnisse der Meta-Analyse von Riketta (2005) weisen darauf hin, dass beides distinkte Konstrukte mit eigenständigen Varianzanteilen sind, die hoch miteinander korrelieren. Aufgrund der Gemeinsamkeiten scheint es gerechtfertigt, auch die größere Anzahl von Untersuchungen zu beachten, die zum Zusammenhang von Tätigkeitsmerkmalen und dem organisationalen Commitment erstellt wurden. Aus ihnen geht hervor, dass Autonomie, Vielfalt, Rückmeldung (z. B. Hunt, Chonko & Wood, 1985; Ramaswami, Agarwal & Bhargava, 1993) und Aufgabengeschlossenheit (z. B. Hunt et al., 1985) einen direkten Einfluss auf die organisationale Bindung haben. Eine Meta-Analyse konnte für Autonomie (r=0,28) und Partizipation (r=0,43) einen Zusammenhang bestätigen (Spector, 1986). In ihrem Strukturgleichungsmodell zeigten Igbaria et al. (1994), dass die Tätigkeitsmerkmale den stärksten Einfluss aller untersuchten Variablen auf organisationales Commitment ausüben. Der Zusammenhang zwischen Autonomie und Commitment hat sich allerdings nicht für alle Berufsgruppen als stabil erwiesen: Für Fachleute scheint er ausgeprägter zu sein, da Autonomie eine wichtige Erwartung an ihre Tätigkeit darstellt (Cohen, 1992). Auch ein Übermaß an "positiver" Tätigkeitsgestaltung ist möglich. Dies wird zumindest von der invertiert U-förmigen Beziehung zwischen Aufgabengeschlossenheit und

organisationalem Commitment suggeriert (Lin & Hsieh, 2002). Als Erklärung für den Zusammenhang von Tätigkeitsmerkmalen mit der Identifikation bieten Forscher ein wechselseitiges Austauschmodell an: Die Organisation bietet erweiterte Aufgaben mit hoher Autonomie, Vielfalt, Aufgabengeschlossenheit und Rückmeldung, was von den Mitarbeitern durch eine stärkere Identifikation "entgolten" wird (Strauss, 1977; Tyagi & Wotruba, 1993).

Auch die Identifikation mit der Open-Source-Gemeinschaft stellt eine Form der sozialen Identifikation dar, die mit der organisationalen Identifikation vergleichbar ist (Bo Xu, 2006). Von besonderer Bedeutung für Open-Source-Gemeinschaften ist die Identifikation, weil sie einen Einfluss auf die Bereitschaft zur Übernahme von freiwilligen Aufgaben hat: je höher die soziale Identifikation ausgeprägt ist, desto mehr freiwillige Aufgaben werden übernommen (Kramer, 1993). Eine Reihe von Studien konnte zeigen, dass die Identifikation mit dem eigenen Open-Source-Projekt hoch ist (Ghosh et al., 2002; Hars & Ou, 2002; Hertel et al., 2003). Aus diesen Studien geht weiterhin hervor, dass die Identifikation ein wichtiger Antrieb für die Teilnahme an und das Engagement in Open-Source-Projekten ist. Nach Hertel et al. (2003) unterscheidet sich jedoch das Objekt der Identifikation in Abhängigkeit von der betrachteten Gruppe. Verglichen mit der Nutzergruppe identifizierten sich die Entwickler im Linux-Kernel Projekt etwas geringer mit der Linux-Gemeinschaft im Allgemeinen, dafür jedoch stärker mit spezifischeren Gruppierungen wie der Gemeinschaft der Entwickler und der Gemeinschaft des Teilbereichs, in dem sie aktiv waren. Einen interessanten Unterschied im Zusammenhang zwischen Identifikation und Leistung konnten Hars und Ou (2002) beobachten. Sie fanden nur für die Gruppe der Studenten- und Hobbyentwickler einen Zusammenhang zwischen der Identifikation und der Leistung, nicht aber für solche Entwickler, die für ihre Tätigkeit entlohnt werden.

Dezentrale Strukturen ermöglichen die Identifikationen mit multiplen Zielen, wie z. B. für die Forstwirtschaft (Bullis & Tompkins, 1989), Weiterbildungsanbieter (Scott, 1997) und soziale Bewegungen (Simon et al., 1998) gezeigt werden konnte. Auch im Open-Source-Bereich wurde die Identifikation auf verschiedenen Ebenen untersucht: Identifikation mit der Gemeinschaft der Open-Source-Entwickler ("Hacker-Community", Lakhani & Wolf, 2003), mit dem Gesamtprojekt (Hertel et al., 2003; Lakhani & Wolf, 2003) und dem eigenen Teilprojekt (Hertel et al., 2003). Aus der Studie von Hertel et al. (2003) geht hervor, dass die Identifikation mit dem Teilprojekt den besten Prädiktor für den betriebenen Aufwand darstellt. Dies steht in Übereinstimmung mit der Erkenntnis, dass lokale Attribute den größten Einfluss auf die Identifikation haben (Bullis & Tompkins, 1989) und dass besonders selbst gesteuerte Teams ein wichtiges Objekt der Identifikation sind (Barker & Tompkins, 1994).

Aus den vorhandenen Erkenntnissen zu Open-Source-Projekten und den theoretischen Implikationen kann man schließen, dass Open-Source-Projekte über ein hohes Ausmaß an organisationaler Identifikation verfügen.

4.5.7 Zusammenfassung der organisationalen Kriterien

Betrachtet man die organisationalen Kriterien im Kontext von Open-Source-Gemeinschaften, wird für intrinsische Motivation, allgemeine Arbeitszufriedenheit, Leistung, OCB, Flow-Erleben und organisationale Identifikation eine höhere Ausprägung erwartet, als dies in proprietären Entwicklungsprojekten der Fall ist.

Wie in den Kapiteln 4.3 und 4.4 dargelegt wurde, wird bei einer Reihe von Tätigkeitsmerkmalen für die Open-Source-Softwareentwicklung eine positivere Ausprägung erwartet als in der proprietären Softwareentwicklung. In diesem Kapitel wurde eine solche positivere Ausprägung auch für die organisationalen Kriterien postuliert. Dass für viele Tätigkeiten die Merkmale einen wichtigen Einfluss auf die organisationalen Kriterien haben, wurde bereits in vielen Untersuchungen bestätigt (vgl. oben). Nicht geklärt ist, ob in einem Kontext wie der Open-Source-Softwareentwicklung, die über viele freiwillige Entwickler verfügt, ein solcher Zusammenhang auch hergestellt werden kann. Die beste Unterstützung für eine solche Vermutung stellen die Ergebnisse aus anderen Tätigkeitsbereichen dar, die auch auf Freiwilligenarbeit basieren. Wie in den Untersuchungen zu Freiwilligenarbeit gezeigt, bestehen auch hier enge Zusammenhänge zwischen den Tätigkeitsmerkmalen und organisationalen Kriterien (Güntert & Wehner, 2005; Wehner & Güntert, 2005). In dieser Arbeit werden insgesamt sechs organisationale Kriterien betrachtet und die Zusammenänge sollen getrennt für jedes der Kriterien untersucht werden. Daher gliedert sich Hypothese 3 in insgesamt sechs Unterhypothesen:

> *Hypothese 3a: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf die allgemeine Arbeitszufriedenheit aus.*
>
> *Hypothese 3b: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf die intrinsische Motivation aus.*
>
> *Hypothese 3c: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf die Leistung aus.*
>
> *Hypothese 3d: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf das OCB aus.*

Hypothese 3e: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf die organisationale Identifikation mit dem Unternehmen aus.

Hypothese 3f: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf die organisationale Identifikation mit der Abteilung aus.

Hypothese 3g: Sowohl bei Open-Source-TeilnehmerInnen als auch bei Mitarbeitern in der proprietären Softwareentwicklung wirken sich Tätigkeitsmerkmale in gleicher Weise auf das Flow-Erleben aus.

Wie die aufgestellten Hypothesen überprüft werden, ist im folgenden Kapitel beschrieben.

4.6 Übersicht Gesamtmodell

Wie in den vorhergehenden Kapiteln beschrieben wurde, liegt dieser Untersuchung ein einfaches Gesamtmodell zugrunde. Es wird davon ausgegangen, dass die relevanten Tätigkeitsmerkmale einen direkten Einfluss auf die betrachteten organisationalen Kriterien haben. Bei den Tätigkeitsmerkmalen werden solche der Komplexität und soziale Aspekte unterschieden. Bei den Merkmalen der Komplexität sollen Aufgaben- und Anforderungsvielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit und Rückmeldung der Tätigkeit erhoben werden. Bei den sozialen Aspekten sollen die Merkmale Rückmeldung durch Personen, soziale Unterstützung, Interdependenz der Aufgabe und der Ziele, die Bedeutsamkeit der Aufgabe und das Qualifikationspotenzial erhoben werden. Im Fokus der organisationalen Kriterien stehen die allgemeine Arbeitszufriedenheit, die intrinsische Motivation, die Leistung, das Organizational Citizenship, die organisatorische Identifikation mit dem Unternehmen und der Abteilung sowie das Flow-Erleben. Abbildung 4 gibt einen Überblick über das Gesamtmodell.

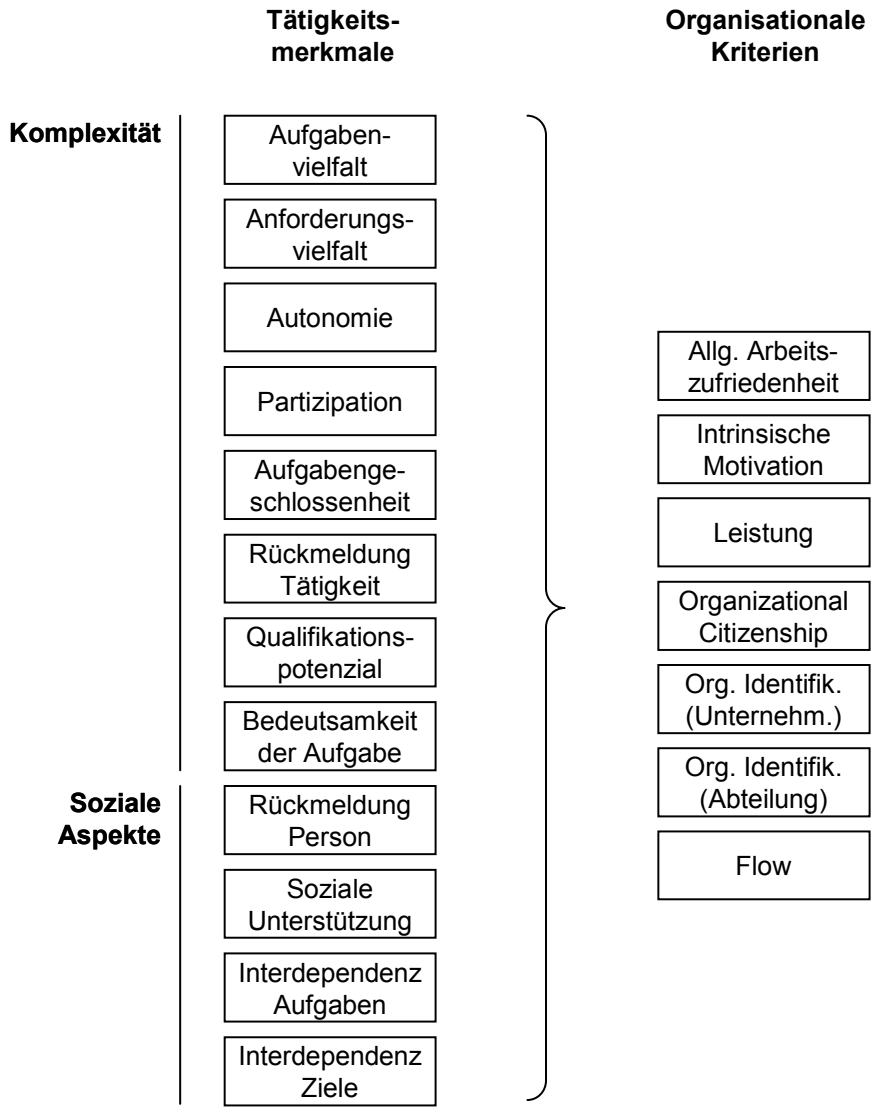

Abbildung 4: Übersicht Gesamtmodell

5 Methode

Für die Erhebung von Merkmalen der Tätigkeit bietet die psychologische Forschung ein breites Angebot an ausgereiften Arbeitsanalyseverfahren. Bisher wurden in Open-Source-Projekten keine systematischen Arbeitsanalysen durchgeführt, drei Aspekte bestärken jedoch den Autor darin, dass ein Einsatz möglich ist: Zum Ersten hat eine Reihe von Arbeitsanalysen in Softwareentwicklungsprojekten und -unternehmen gezeigt, dass gängige Verfahren die Inhalte der Tätigkeit eines Softwareentwicklers abbilden können (z. B. Brodbeck & Frese, 1994; z. B. Couger et al., 1989; Goldstein & Rockart, 1984; Wieland et al., 2004). Zum Zweiten konnte gezeigt werden, dass Arbeitsanalyseverfahren auch bei den besonderen Rahmenbedingungen, wie sie in der Freiwilligenarbeit zu finden sind, eingesetzt werden können (Burkolter, 2005; Wehner & Güntert, 2005). Zum Dritten wurden bereits einzelne Items aus Arbeitsanalyseverfahren in Untersuchungen von Open-Source-Projekten genutzt (Roberts, Hann & Slaughter, 2006). Die Auswahl der eingesetzten Verfahren wird in den folgenden Kapiteln beschrieben.

5.1 Arbeitsanalyseverfahren

Es gibt eine Vielzahl unterschiedlicher Arbeitsanalyseverfahren (für einen Überblick deutschsprachiger Verfahren: Dunckel, 1999b), die auf unterschiedlichen theoretischen Grundlagen basieren, sich durch den abgedeckten Merkmalsbereich unterscheiden und verschiedene Methoden der Erhebung einsetzen. Die bei Arbeitsanalysen am häufigsten angewandte Methode ist die Befragungsmethode (Dunckel, 1999a); diese zeichnet sich dadurch aus, dass Instrumente leicht entwickelt und angewendet werden können. Ein Arbeitsanalyseverfahren, das auf der Befragungsmethode beruht, ist der JDS (Hackman & Oldham, 1976). Der JDS hat vor allem im angloamerikanischen Raum sowohl in der Wissenschaft als auch in der Praxis weite Verbreitung gefunden (Schmidt & Kleinbeck, 1999) und wird für eines der bekanntesten Verfahren der Arbeitsanalyse gehalten (Matern, 1984; Taber & Taylor, 1990). Neben seiner Durchführungsökonomie zeichnet sich der standardisierte Fragebogen des JDS durch seine Anwendbarkeit auf unterschiedliche Branchen und Tätigkeitsklassen aus. Allerdings sind die psychometrischen Eigenschaften des JDS Gegenstand vielfältiger Kritik. Die Reliabilität der Skalen der Tätigkeitsmerkmale ist mit 0,65 bis 0,71 nicht zufriedenstellend (Taber & Taylor, 1990) und die Faktorenstruktur der Tätigkeitsmerkmale hat sich als ausgesprochen instabil herausgestellt: Von 30 Studien wiesen in explorativen Faktorenanalysen nur 12 die prognostizierte Anzahl von 5 Faktoren auf (Taber & Taylor, 1990). Die restlichen Untersuchungen fanden zwischen einem (Dunham, 1976) und sechs Faktoren (Idaszak & Drasgow, 1987). Bei konfirmatorischen Faktorenanalysen erreichten nur solche Modelle eine gute Übereinstimmung mit den Daten, die das JCM durch Hinzufügen von

Methodenfaktoren oder zusätzlichen Items erweiterten (Taber & Taylor, 1990). Auf der Basis ihrer Analysen entwickelten Idaszak und Drasgow (1987) eine revidierte Fassung des JDS, die auf negativ formulierte Items verzichtete. Bei der revidierten Fassung konnte die Struktur der fünf Faktoren bestätigt werden (Cordery & Sevastos, 1993; Harvey, Billings & Nilan, 1985; Kulik, Oldham & Langner, 1988), allerdings erhöhte die Revision nicht die Prognosekraft für die Kriterien Arbeitszufriedenheit und Leistung (Kulik et al., 1988). Auch von Autoren aus der IT-Branche wurde die unzureichende Stabilität der Faktoren des JDS kritisiert (Sein & Bostrom, 1991).

Neben dem JDS ist auch das "Job Characteristics Inventory" (JCI) verbreitet. Die vorliegenden Studien weisen darauf hin, dass die psychometrischen Eigenschaften des JCI dem JDS überlegen sind (zusammenfassend Morgeson & Campion, 2003). Das JCI basiert zu großen Teilen auf Items aus dem JDS, verfügt aber neben vier Skalen aus dem JDS (Vielfalt, Autonomie, Rückmeldung durch die Tätigkeit und Aufgabengeschlossenheit) über die zwei zusätzlichen Skalen "Möglichkeiten zum Schließen von Freundschaften" und "Umgang mit Anderen" (Sims et al., 1976). Die Reliabilität der Skalen liegt mit 0,76 bis 0,84 (Cronbachs Alpha) höher als beim JDS (Aryee, Chay & Chew, 1996; Dodd & Ganster, 1996; Ganzach, 1998; Mathieu, Hofmann & Farr, 1993; Williams, Gavin & Williams, 1996). Die Faktorenstruktur erwies sich als stabil über die Zeit (Griffin, 1981) und verschiedene Stichproben (Griffin et al., 1980).

Gängige Arbeitsanalyseverfahren, insbesondere der JDS, wurden von psychologischen Forschern für die Beschränkung auf nur wenige Tätigkeitsmerkmale kritisiert (z. B. Parker & Wall, 2002; z. B. Stone & Gueutal, 1985; Taber & Taylor, 1990; Wall & Martin, 1994). Die von Hackman und Oldham (1975) als "Kerntätigkeitsmerkmale" (core job dimensions) bezeichnete Auswahl von fünf Tätigkeitsmerkmalen basiert auf den Arbeiten von Turner und Lawrence (1965), welche ebenfalls die Grundlage für den JCI darstellen (Sims et al., 1976). Nach der Veröffentlichung wurden "Angemessenheit" und "Vollständigkeit" der Dimensionen infrage gestellt (z. B. Roberts & Glick, 1981). Merkmale wie Arbeitsgeschwindigkeit, Zeitdruck, physische Gefährdung, Arbeitsbelastung, Zykluslänge, Anforderungen an die Aufmerksamkeit, Entwicklungsmöglichkeiten sowie soziale Interaktion und Unterstützung haben sich teilweise in Untersuchungen als wichtig oder als potenzielle Determinanten von Arbeitszufriedenheit und Leistung erwiesen, wurden jedoch noch nicht in die Theorie zur Arbeitsgestaltung einbezogen (Parker & Wall, 2002; Taber & Taylor, 1990; Wall & Martin, 1994). Ohne die Einbeziehung in vorhandene Theorien ist es nicht möglich, die Interaktionen und Redundanzen mit anderen Tätigkeitsmerkmalen abzuschätzen (Parker & Wall, 2002). Kritik an der beschränkten Anzahl von Tätigkeitsmerkmalen in gängigen Arbeitsanalyseverfahren kam auch aus der Forschung in der IT-Branche (Goldstein & Rockart, 1984). Den Unzulänglichkeiten wurde teilweise durch das Hinzufügen zusätzlicher

Merkmale wie Rollenkonflikt (role-conflict) und Rollenunklarheit (role ambiguity) (Baroudi, 1985; Bostrom, 1978; Goldstein & Rockart, 1984) oder Lernmöglichkeiten (Sein & Bostrom, 1991) begegnet.

Bei einer explorativen Bestimmung der Dimensionen von Tätigkeit wurden insgesamt 3 verschiedene Dimensionen identifiziert: Komplexität (job complexity), Nützlichkeit für die Öffentlichkeit (serves the public) und physische Beanspruchung (physical demand) (Stone & Gueutal, 1985). Die körperliche Beanspruchung macht neben motivationalen Eigenschaften der Tätigkeit einen großen Teil des "Multimethod Job Design Questionnaire" (MJDQ) aus (Campion, 1988; Campion & Thayer, 1985). Das postulierte Modell mit vier Faktoren konnte nicht bestätigt werden (Edwards et al., 1999). Fragen zur physischen Beanspruchung sind z. B. auch im "Job Content Questionnaire" (JCQ, Karasek et al., 1998) und in einigen deutschsprachigen Verfahren wie SALSA (Udris, 2001) und ISTA (Semmer et al., 1999) enthalten.

Neben physischen Merkmalen fehlen im JDS vor allem soziale Aspekte der Tätigkeiten. Mit den Tätigkeitsmerkmalen "Möglichkeiten zum Schließen von Freundschaften" und "Umgang mit Anderen" sind diese zumindest partiell im JCI vorhanden (Sims et al., 1976). Der Einfluss von sozialer Unterstützung wurde insbesondere in Arbeitsanalyseverfahren, die aus der Stressforschung stammen, untersucht (Karasek et al., 1998). Ein Beispiel hierfür ist der Job Content Questionnaire (JCQ). Auch deutschsprachige Verfahren, die soziale Aspekte erfassen, sind von den Ergebnissen der Stressforschung geleitet. Zusätzlich beruhen Verfahren wie der SynBA-GA (Wieland-Eckelmann, Saßmannshauen, Rose & Schwarz, 1999) oder ISTA (Zapf, 1991) auf handlungspsychologischen Konzepten. Wie die Stressforschung geht auch die "Salutogenetische Subjektive Arbeitsanalyse" (SALSA) davon aus, dass soziale Aspekte der Arbeit wie soziale Unterstützung Ressourcen darstellen, die die negativen Auswirkungen von Arbeitsbelastung mindern können (Rimann & Udris, 1997).

Auch wenn Instrumente für weitgehend alle bisher diskutieren Merkmale der Tätigkeit existieren, so fehlt ein Instrument, das sowohl über ausreichende psychometrische Qualität als auch Vollständigkeit verfügt. Diesem Umstand versuchen Morgeson und Humphrey (2003) mit der Entwicklung des "Work Design Questionnaire" (WDQ) zu begegnen. Dieser Fragebogen fasst Dimensionen aus unterschiedlichen theoretischen Richtungen zusammen, um eine vollständigere Beschreibung der Tätigkeit zu erreichen. Bei der Konstruktion wurde, soweit möglich, auf Items aus vorhandenen Verfahren zurückgegriffen. In einer ersten empirischen Überprüfung erwiesen sich sowohl Reliabilität, Validität als auch Dimensionalität als vielversprechend.

Tabelle 8: Interne Konsistenz für ausgewählte Skalen des WDQ (Morgeson & Humphrey, 2003)

Skala	Interne Konsistenz*
Aufgabenvielfalt	0,95
Anforderungsvielfalt	0,86
Autonomie (Einteilung der Arbeitsabfolge)	0,85
Autonomie (Treffen von Entscheidungen)	0,85
Autonomie (Auswahl der Methoden)	0,88
Aufgabengeschlossenheit	0,88
Bedeutsamkeit der Aufgabe	0,87
Rückmeldung durch die Tätigkeit	0,86
Rückmeldung durch andere Personen	0,88
Soziale Unterstützung	0,82

* Cronbachs Alpha

Die für den Vergleich von Open-Source- und proprietärer Softwareentwicklung als relevant betrachteten Tätigkeitsmerkmale werden weitgehend vom WDQ abgedeckt. Daher soll der WDQ, soweit möglich, für die Erhebung der Tätigkeitsmerkmale eingesetzt werden. Eine detaillierte Übersicht über die verwendeten Instrumente enthält das folgende Kapitel.

5.2 Definitionen und Operationalisierung

In der Forschung zu Arbeitsanalysen fehlen häufig Definitionen oder explizite Verweise auf Definitionen der untersuchten Tätigkeitsmerkmale. Da in dieser Arbeit auf Tätigkeitsmerkmale aus unterschiedlichen Forschungsbereichen zurückgegriffen wird, soll im folgenden Kapitel eine genaue Definition der Merkmale aufgeführt werden. Außerdem soll in diesem Kapitel eine Übersicht über die Operationalisierung der Tätigkeitsmerkmale gegeben werden. Dabei wurde, soweit möglich, auf bestehende Skalen zurückgegriffen. Um einen möglichst großen Einblick in das noch unerforschte Gebiet der Tätigkeitsgestaltung in Open-Source-Projekten zu ermöglichen, wurde eine große Anzahl an Tätigkeitsmerkmalen in die Untersuchung einbezogen. Damit die Ausfüllzeit des Fragebogens nicht ein vertretbares Maß übersteigt, wurden einige Skalen gekürzt. Bei der Auswahl der Fragen wurde die interne Konsistenz und die Anwendbarkeit im Open-Source-Kontext berücksichtigt. Beispielsweise konnten Items nicht angewendet werden, die zusätzliche Leistungen, die über das verpflichtende Maß hinausgehen, erheben, da im Open-Source-Bereich für viele Entwickler keine verpflichtenden Leistungen existieren und alle Leistungen freiwillig sind. Des Weiteren waren die Instrumente der

Arbeitsanalyse mit nur geringfügigen Umformulierungen (z. B. Ersetzen von "Führungskraft" durch "Projektleiter") im Open-Source-Bereich sinnvoll anwendbar.

5.2.1 Komplexität der Tätigkeit

In diesem Abschnitt sollen die Tätigkeitsmerkmale der Komplexität definiert und die Auswahl der entsprechenden Instrumente zur Erhebung dargestellt werden. Das Tätigkeitsmerkmal *Autonomie* wurde von Hackman und Oldham (1975) als Ausmaß der Freiheit bei der Ausführung der Tätigkeiten definiert. Von Wall et al. (1992) und Wall, Jackson und Mullarkey (1995) wurde Autonomie in drei Teilaspekte untergliedert: Arbeitsabfolge, Treffen von Entscheidung und Arbeitsmethoden. Die verwendete Definition von Autonomie basiert auf dieser Differenzierung (Morgeson & Humphrey, 2003):

> Die Skala "Autonomie" gibt das Ausmaß der Freiheit in der Einteilung der Arbeitsabfolge (work scheduling), im Treffen von Entscheidungen (decision-making) und in der Auswahl der Arbeitsmethoden (work methods) an (übersetzt nach Morgeson & Humphrey, 2003).

Für die Erhebung wird die Skala "Autonomy" aus dem WDQ genutzt. Aufgrund der hohen Anzahl von untersuchten Skalen wurde bei der Zusammenstellung der Skala Autonomie, wie auch bei allen anderen Skalen, auf eine möglichst geringe Anzahl von Items geachtet. Aus diesen Grund wurden von den 9 Items des WDQ insgesamt 3 Items entfernt. Es wurde aus den 3 Subskalen von Autonomie jeweils ein Item entfernt. Als Auswahlkriterium diente dabei der Korrelationskoeffizient des Items mit der Gesamtskala. Diese Daten sind unveröffentlicht, wurden jedoch von den Autoren zur Verfügung gestellt (Morgeson & Humphrey, 2003). Aus der Subskala "Work Scheduling Autonomy" wurde das 3. Item nicht verwendet (korrigierte Item-Skalen Korrelation von .69). Aus der Subskala "Decision-Making Autonomy" entfiel das Item Nummer 6 (korrigierte Item-Skalen Korrelation von .65) und bei "Work Methods Autonomy" wurde auf das Item Nummer 7 verzichtet (korrigierte Item-Skalen Korrelation von .72).

Die in der Literatur verwendeten Definitionen für Autonomie sind weitgehend konsistent: Bei der Erhebung von *Partizipation* wird häufig die persönliche Kontrolle über die Arbeitstätigkeit mit der Möglichkeit für Eingaben in die Entscheidungsfindungen vermischt (Spector, 1986). Partizipation soll in der vorliegenden Arbeit als Möglichkeit zur Einflussnahme auf den größeren Zusammenhang der Organisation verstanden werden. Die von Glew, O'Leary-Kelly, Griffin und Van Fleet (1995) herausgestellte und häufig verwendete Definition von Partizipation, das Teilen von Einfluss (Mitchell, 1973), das gemeinsame Treffen von Entscheidungen (Locke & Schweiger, 1979) und das Involvieren von Mitarbeitern (Miller & Monge, 1986) werden vom Autor als

wichtige Bestandteile von Partizipation verstanden. Aus diesen Grund wurde zur Definition auf die Festlegung von Udris und Rinmann (1999) in ihrem Fragebogen SALSA zurückgegriffen:

> Die Skala "Partizipation" gibt das Ausmaß an, zu "dem die Firmenleitung oder Vorgesetzte rechtzeitig über Änderungen der Arbeitsorganisation informieren und inwiefern bei Veränderungen auch Eigeninitiative, Mitsprache und Beteiligung (Partizipation) ermöglicht werden." (Udris & Rinmann, 1999, S. 409).

Das Tätigkeitsmerkmal *Partizipation* wird im WDQ nicht berücksichtigt, stattdessen wird die Skala "Information und Mitsprache" aus dem "Kurz-Fragebogen zur Arbeitsanalyse" (KFZA, Prümper, Hartmannsgruber & Frese, 1995) übernommen. Trotz anderer Benennung sind in dieser Skala beide Aspekte aus der Definition von Udris und Rinmann (1999), die Information und die Möglichkeit der Beteiligung bei der Entscheidung enthalten. Die Items der Skala "Information und Mitsprache" basieren auf dem "Erhebungsbogen zur Erfassung des Betriebsklimas" von Rosenstiel, Falkenberg, Hehn, Henschel und Warns (1982). Für diese Erhebung wurde die Skala vom Autor aus dem Deutschen ins Englische übersetzt.

Rückmeldung ist nach (Hackman & Oldham, 1976) das Ausmaß an direkter und klarer Information über die Effektivität des eigenen Handelns, die Personen als Ergebnis ihrer Arbeit erhalten. In dieser Konzeption beschränkt sich Rückmeldung ausschließlich auf die Rückmeldung durch die Arbeit, während Rückmeldung durch andere Personen nicht berücksichtigt wird. Die Definition "*Rückmeldung durch die Tätigkeit*" (feedback from job) von Morgeson und Humphrey (2003) entspricht weitgehend der Definition von Hackman und Oldham (1976).

> Die Skala "Rückmeldung durch die Tätigkeit" gibt das Ausmaß an, zu dem die Tätigkeit Informationen über die Erfüllung der Aufgabe vermittelt (Morgeson & Humphrey, 2003).

Die im WDQ verwendeten Items sind weitgehend identisch mit der Skala "feedback" aus dem JDS (Hackman & Oldham, 1975). Um den Fragebogen kurz zu halten, wurde auf das erste Item verzichtet. Dieses wies mit .63 die geringste korrigierte Korrelation mit der Skala auf.

Die *Aufgabengeschlossenheit* ist nach Turner und Lawrence (1965) die Deutlichkeit, mit der das Ergebnis der Tätigkeit als eigenständiger Beitrag im Endprodukt zu erkennen ist. In der Erhebung von Turner wurde die Geschlossenheit des Zyklus (cycle closure), der Wert der Transformation, die Sichtbarkeit der Veränderung durch den Mitarbeiter und die Sichtbarkeit dieser Transformation im Endprodukt berücksichtigt. Hackman und Oldham (1976) definieren Aufgabengeschlossenheit als das Ausmaß, zu dem eine Tätigkeit die Erfüllung einer vollständigen Aufgabe, mit eindeutigem Anfang und Ende, er-

fordert. Das Ergebnis dieser Arbeit muss erkennbar sein. Sims et al. (1976) verwenden eine fast identische Definition, an der sich auch (Morgeson & Humphrey, 2003) orientieren.

> Die Skala "Aufgabengeschlossenheit" gibt das Ausmaß an, zu dem eine Tätigkeit eine abgeschlossene Aufgabe beinhaltet, deren Ergebnisse leicht identifiziert werden können (Morgeson & Humphrey, 2003).

Die Skala Aufgabengeschlossenheit des WDQ orientiert sich an JCI und JDS. Von den 4 Items der Skala wurden die Items 1 und 2 nicht in den Fragebogen aufgenommen. Sie wiesen mit .64 (Item 1) und .76 (Item 2) die geringste Korrelation mit der Gesamtskala auf.

In der Konzeption sowohl von Turner und Lawrence (1965) als auch von Hackman und Oldham (1976) wird bei Vielfalt nicht zwischen Abwechselungsreichtum der Aufgabe und dem Einsatz von verschiedenen Fertigkeiten unterschieden. Für die Definition wird daher auf die Arbeit von Morgeson und Humphrey (2003) zurückgegriffen, die Anforderungsvielfalt und Aufgabenvielfalt unterscheidet. Dabei bezieht sich die Skala *Anforderungsvielfalt* auf die Unterschiedlichkeit der eingesetzten Fertigkeiten.

> Die Skala "Anforderungsvielfalt" misst das Ausmaß an unterschiedlichen Fertigkeiten, die zur Erfüllung der Aufgabe eingesetzt werden müssen (Morgeson & Humphrey, 2003).

Bei der verwendeten Skala des WDQ handelt es sich um eine Neuentwicklung, die an der Arbeit von Hackman und Oldham (1980) angelehnt ist. Aus dieser Skala entfielen die Items 1 und 3, da sie die niedrigste Korrelation mit der Gesamtskala aufwiesen (.76 für Item 1 und .63 für Item 3).

Im Unterschied zu der Ausführung unterschiedlicher Fähigkeiten erfasst die Skala *Aufgabenvielfalt* die Anzahl an verschiedenen Aufgaben, die für eine Tätigkeit ausgeführt werden müssen.

> Die Skala "Aufgabenvielfalt" ist das Ausmaß, zu dem eine Tätigkeit die Ausführung einer großen Auswahl von Aufgaben erforderlich macht (Morgeson & Humphrey, 2003).

Das von Morgeson und Humphrey (2003) verwendete Maß für die Aufgabenvielfalt entspricht weitgehend der Skala "Variety", wie sie im JCI verwendet wird (Sims et al., 1976). Bis auf die Items 1 und 4 wurde diese Skala für den Fragebogen übernommen. Dabei wies Item 1 mit einer Korrelation von .85 und Item 4 mit einer Korrelation von .87 die geringste Korrelation zur Gesamtskala auf.

Die *Bedeutsamkeit der Aufgabe* beschreibt die Auswirkungen der Tätigkeit für Außenstehende. Die Definition von Hackman und Oldham (1976) wurde von Morgeson und Humphrey (2003) übernommen.

> Die Skala "Bedeutsamkeit der Aufgabe" gibt das Ausmaß an, zu dem die Tätigkeit Einfluss auf das Leben und die Arbeit anderer Personen hat (Hackman & Oldham, 1980).

Die Skala, die im WDQ verwendet wird, entspricht der von Hackman und Oldham (1980); es wurde ein negativ gepoltes Item entfernt und die Antwortmöglichkeiten wurden vereinfacht. In diesen Fragebogen sind die Items 2 und 4 aufgrund der geringen Korrelation mit der Skala (Item 2 mit r=.64 und Item 4 mit r=.70) nicht übernommen worden.

Für die Skala *Qualifikationspotenzial* kann eine Definition für das Instrument SALSA von (Udris & Rinmann, 1999) herangezogen werden:

> Die Skala "Qualifikationspotenzial" gibt das Ausmaß des Fähigkeitszuwachses und -verlusts durch die Tätigkeit an (Udris & Rinmann, 1999, S. 409).

Die Skala "Qualifikationspotenzial der Arbeitstätigkeit" aus dem Instrument erschien für die Erhebung im Open-Source-Bereich weniger geeignet. Stattdessen wurde ein Item neu entwickelt und ein weiteres Item aus der Skala "growth/learning" des "Multimethod Job Design Questionnaire" (MJDQ, Edwards et al., 1999) entnommen.

Zusätzlich wurden noch Einzelfragen auf der Basis von Eigenentwicklungen zu der Arbeitsumgebung und der Qualität der eingesetzten Werkzeuge aufgenommen.

5.2.2 Soziale Aspekte der Tätigkeit

Neben den Tätigkeitsmerkmalen der Komplexität werden auch die sozialen Aspekte der Tätigkeit erhoben. Soziale Unterstützung integriert verschiedene Aspekte der Zuwendung: zum einen die direkte Unterstützung durch Kollegen und Vorgesetzte, zum anderen die Möglichkeit zur Schaffung von freundschaftlichen Beziehungen.

> Die Skala "Soziale Unterstützung" gibt das Ausmaß an, zu dem die Tätigkeit Unterstützung von anderen und Austausch mit anderen bietet. Dabei sind sowohl die Unterstützung durch Kollegen und Vorgesetzte als auch Möglichkeiten für Freundschaft eingeschlossen (Morgeson & Humphrey, 2003).

Dabei ist der Aspekt der sozialen Unterstützung durch Kollegen und Vorgesetzte durch die Arbeiten von Karasek et al. (1998) inspiriert, während der Aspekt "Möglichkeiten zum Schließen von Freundschaften" auf den JCI (Sims et al., 1976) zurückgeht. Für den Fragebogen wurde die Skala "social support" bis auf die Items 3 und 4 übernommen.

Diese beiden Items wurden ausgelassen, da sie geringste Korrelationen zur Skala aufwiesen (Item 3: r=.62 und Item 4: r=.42).

In frühen Untersuchungen von Hackman und Lawler (1971) wurde sowohl die Rückmeldung durch die Tätigkeit als auch die *Rückmeldung durch andere Personen* berücksichtigt. Die Rückmeldung durch Personen fand keine Aufnahme in den JDS (Hackman & Oldham, 1975), während im JCI zumindest der Aspekt der Rückmeldung durch den Vorgesetzten berücksichtigt wurde (Sims et al., 1976). Für die verwendete Skala wurden die Fragen aus dem JCI erweitert, um sowohl Rückmeldung von Vorgesetzten als auch Rückmeldung von Mitarbeitern einzuschließen (Morgeson & Humphrey, 2003).

> Die Skala "Rückmeldung durch andere Personen" gibt das Ausmaß an, zu dem andere Personen aus der Organisation Informationen über die Leistung vermitteln (Morgeson & Humphrey, 2003).

Aus der entsprechenden Skala des WDQ wurden alle Fragen bis auf das Item 1 entnommen. Dieses wurde aufgrund der geringsten Korrelation mit der Gesamtskala von .73 weggelassen.

Bei der Interdependenz sollten die beiden Teilaspekte Aufgaben- und Zielinterdependenz berücksichtigt werden. Im WDQ von Morgeson und Humphrey (2003) wird ausschließlich die Interdependenz der Aufgaben eingeschlossen.

> Die Skala "Aufgabeninterdependenz" gibt das Ausmaß an, zu dem die eigene Tätigkeit von den Resultaten anderer Tätigkeiten abhängt und andere Tätigkeiten von der eigenen Tätigkeit abhängig sind (Morgeson & Humphrey, 2003).

Für die Erhebung wurden sowohl die Items aus der Skala "Initiierte Interdependenz" (initiated interdependence) als auch aus der Skala "Erhaltene Interdependenz" (received interdependence) aufgenommen; die Items mit den Nummern 1 und 4 wurden aufgrund der geringen Korrelation mit der Gesamtskala ausgelassen (Item 1 mit r=.56 und Item 4 mit r=.65). Die Definition der *Zielinterdependenz* basiert auf Johnson und Johnson (1992):

> Die Skala "Zielinterdependenz" gibt das Ausmaß an, inwieweit die eigenen Ziele nur erreicht werden können, indem andere ihre Ziele erreichen (Johnson & Johnson, 1992, S. 181).

Da im WDQ der Aspekt der Zielinterdependenz nicht berücksichtigt wird, wurde zur Erhebung eine Skala aus dem Instrument von Campion et al. (1993) genutzt. In Ermangelung anderer Kriterien wurde das Item 20 ("My work activities on any given day are determined by my team's goals for that day.") auf Basis von geringerer Augenscheinvalidität ausgeschlossen.

5.2.3 Organisationale Kriterien

Als Ergebnisse bzw. Auswirkungen der Tätigkeit werden sechs verschiedene Aspekte im Fragebogen berücksichtigt: Intrinsische Motivation, Leistung, OCB, Flow-Erleben und organisationale Identifikation. Die *intrinsische Motivation* wird im Sinne von Deci (1975) als intrinsisch motiviertes Handeln verstanden:

> "Intrinsisch motivierte Verhaltensweisen können als interessenbestimmte Handlungen definiert werden, deren Aufrechterhaltung keine vom Handlungsgeschehen 'separierbaren' Konsequenzen erfordert, d. h. keine externen oder intrapsychischen Anstöße, Versprechungen oder Drohungen." (Deci 1974, 1992 zitiert nach Deci & Ryan, 1993)

Für die Erhebung der intrinsischen Motivation wird die Skala "internal motivation" aus dem JDS (Hackman & Oldham, 1975) verwendet.

Die *Leistung* der Entwickler wurde durch eine subjektive Selbsteinschätzung und durch eine Einschätzung der Anzahl erstellter Quellcodezeilen pro Zeiteinheit erhoben. Für die subjektive Selbsteinschätzung wurde eine Skala entwickelt, die sich an der Erhebungsmethode von Prüden und Reese (1972) orientiert und die Befragten auffordert, ihre Arbeitsleistung im Hinblick auf Qualität und Quantität in Relation zu ihren Mitentwicklern einzuschätzen. Eine objektivere Einschätzung der Leistung von Programmierern erlauben diverse Maße aus der Softwaretechnik (vgl. z. B. Shepperd & Ince, 1993). Mit geringem Aufwand kann die Größe und Komplexität einer Software über die Anzahl der Programmzeilen ("lines of code"; LOC) abgeschätzt werden (Mall, 2004; Shepperd & Ince, 1993). Als Produktivitätsmaß wurde LOC dafür kritisiert, dass es Anreize für längere, aber nicht effizientere Software schafft, und dass es manipulierbar sei (Jones, 1978). Im Rahmen dieser Untersuchung sollte dieses Problem nicht besonders schwer wiegen, da die Ergebnisse nicht für eine Bewertung der individuellen Leistung herangezogen werden. Unterstützung findet die Maßeinheit LOC darin, dass sie mit vielen, weitaus komplexeren Produktivitätsmaßen hoch korreliert ist (Boehm, 1987; Lawrence, 1981).

Das "Organizational Citizenship Behavior" (OCB) beschreibt das Verhalten eines "Bürgers" der Organisation.

> Als OCB wird solches Verhalten bezeichnet, das durch die Förderung und Aufrechterhaltung des sozialen und psychologischen Kontexts die Arbeitsleistung unterstützt (Organ, 1997, S. 91).

Organ (1988) postuliert fünf Dimensionen des OCB: Hilfsbereitschaft (Altruism), Gewissenhaftigkeit (Conscientiousness), Unkompliziertheit (Sportmanship), Rücksichtnahme (Courtesy) und Eigeninitiative (Civic Virtue). Dass es sich hierbei tatsächlich um

unterschiedliche Dimensionen handelt, wird durch die hohe Interkorrelation infrage gestellt (z. B. Podsakoff, MacKenzie, Moorman & Fetter, 1990). Nach der Analyse von LePine et al. (2002) überschreiten die meisten der Dimensionen die Schwelle, die für Items einer Skala noch als akzeptables Maß der internen Konsistenz angesehen wird. Aus diesem Grund scheint eine Zusammenfassung zu einer Skala gerechtfertigt. Um möglichst alle Aspekte abzudecken, wurden aus dem Instrument von Podsakoff et al. (1990) bzw. aus dem an Podsakoff et al. angelehnten Instrument von Verbeke, Belschak, Wuyts und Bagozzi (2004) Items aus allen Dimensionen entnommen und zu einer Gesamtskala "OCB" zusammengefasst. Dabei wurden insgesamt 3 Items aus der Skala Hilfsbereitschaft und jeweils 1 Item aus den Skalen Gewissenhaftigkeit, Unkompliziertheit, Rücksichtnahme und Eigeninitiative entnommen.

Der mit *Flow* bezeichnete Zustand wird folgendermaßen definiert:

> Flow ist ein Zustand des optimalen Erlebens mit vollständigem Aufgehen in der ausgeführten Handlung. Dieser Zustand ist mit ausgeprägter zielbezogener Aktivierung und Konzentration verbunden und führt zu Selbst- und Zeitvergessenheit. Die Aufmerksamkeit ist dabei völlig auf die Tätigkeit gerichtet, das Ziel der Handlungen und die Rückmeldungen sind deutlich und es besteht eine hohe Kompetenz und Kontrolle. (Csikszentmihalyi, 1982; Csikszentmihalyi & Csikszentmihalyi, 1991)

Zur Erhebung stehen unterschiedliche Verfahren zur Verfügung (für einen detaillierten Überblick siehe Rheinberg, Vollmeyer & Engeser, 2003). Da es rückblickend schwierig ist, einen Zustand des "vollständigen Aufgehens" in der Tätigkeit zu beurteilen, hat die Experience Sampling Method (ESM, Csikszentmihalyi & Larson, 1987) zum Ziel, die Messung möglichst zeitnah vorzunehmen. Zu diesem Zweck wurden Probanden wiederholt zu unvorhersagbaren Zeitpunkten von einem Signalgeber zur Beantwortung von einigen Fragen aufgefordert. Problematisch sind bei der ESM die Inhalte der gestellten Fragen: Statt des Flow-Erlebens werden auslösende Bedingungen abgefragt, die mit dem Flow-Erleben gleichgesetzt werden (vgl. auch Rheinberg et al., 2003). Neben dem ESM existiert eine Reihe von Fragebogenverfahren (z. B. Novak & Hoffman, 1997; z. B. Rheinberg, 1987). Für diese Untersuchung wurde ein Verfahren eingesetzt, das auf einer Stellungnahme zur Beschreibung von Flow-Erlebnissen beruht. Dieses Verfahren wurde von Chen et al. (1999) zur Erhebung bei Internet-NutzerInnen eingesetzt und basiert auf der Beschreibung von Flow-Erlebnissen, die bereits in vorherigen Studien verwendet wurden (Csikszentmihalyi, 1982; Han, 1988; McQuillan & Conde, 1996). Zu dieser Beschreibung sollen die TeilnehmerInnen angeben, ob sie vergleichbare Erlebnisse während ihrer Entwicklungstätigkeit hatten und wie häufig dies vorkam.

Es existiert eine Reihe von Definitionen für *organisationale Identifikation* (für eine Übersicht siehe Scott, 1997). Den Definitionen gemein ist, dass es sich bei Identifikation um einen Prozess handelt, in dem Individuen sich mit Objekten aus ihrer sozialen Umwelt verbinden bzw. bei dem gemeinsame Interessen angenommen werden (Burke, 1969; Cheney, 1983). Im Fall der organisationalen Identifikation ist die Organisation das Ziel dieser Verbindung.

> Organisationale Identifikation ist ein Prozess der internen und externen Überzeugung, bei dem sich Interessen des Individuums mit Interessen der Organisation verbinden (Johnson, Johnson & Heimberg, 1999, S. 160)

Um die Vergleichbarkeit mit einer parallel laufenden Studie zu gewährleisten, wurde ein Maß der Identifikation genutzt, das an den Fragebogen von Hertel et al. (2003) angelehnt ist. Es wird dabei sowohl die Identifikation der TeilehmerInnen mit der Gesamtorganisation als auch mit der jeweiligen Arbeitsgruppe (falls vorhanden) berücksichtigt.

5.3 Erhebungsmethode

Für die Erhebung der Daten wurden zwei unterschiedliche Varianten eines Onlinefragebogens erstellt. Die erste Variante diente zur Erhebung in Open-Source-Projekten und die zweite Variante wurde für den Einsatz bei proprietären Softwareentwicklern erstellt. Beide Varianten waren weitgehend identisch. Die aus der Literatur entnommenen Skalen wurden für die Open-Source-Variante leicht modifiziert (siehe Kapitel 5.2). Außerdem enthielt die Open-Source-Variante noch 5 zusätzliche Fragen, die in Tabelle 9 aufgeführt sind.

Tabelle 9: Zusätzliche Fragen Open-Source-Version des Fragebogens

Fragen
What percentage of your income is based on your activities in free/open-source projects?
How many submissions/check-ins do you make for your main project?
Please estimate the time you have spent on the development of your main software project during the last three months (average hours per week).
On average, what percentage of your leisure time you spend on open-source projects (all open-source projects you participate in)?
What is the name of your project (optional)?

Für die technische Realisierung der Varianten des Onlinefragebogens wurde die Software von Unipark genutzt. Beide Varianten bestehen aus jeweils einer Bildschirmseite mit Ausfüllanweisungen, 15 Bildschirmseiten mit Fragen und einer Schlussseite. Insgesamt enthält der Fragebogen 73 größtenteils geschlossene Items. Die Fragen verteilen sich auf

die Blöcke "task/job complexity", "task/job cooperation", "general working conditions", "task/job satisfaction", "contribution" und "statistical information". Für die Beantwortung des Fragebogens brauchten die Probanden im Mittel 21 Minuten.

Bei der Erstellung des Fragebogens wurden Grundsätze der Fragebogengestaltung wie ein Beginn mit leichten und interessanten Fragen berücksichtigt (Dillman, 2007). Das Format der Fragen entspricht solchen, die auch bei papierbasierten Fragebögen gebräuchlich sind, und sollte damit den Ausfüllern weitgehend bekannt sein. Auf Farben zur Hervorhebung wurde vollständig verzichtet (Bowker, 1999; Dillman, 2007). Die Darstellung des Fragebogens ist für Monitore mit einer Auflösung von 800 mal 600 Punkten optimiert. Diese geringe Anforderung sollte von der technischen Ausstattung der Stichprobe leicht erfüllt werden. Auf die Verwendung von automatischen Kontrollen der Eingaben wurde zugunsten einer besseren Ausfüllbarkeit verzichtet. Ein allgemeines Problem von Internetbefragungen sind unzureichende Fähigkeiten im Umgang mit dem Computer (Dillman, 2007); dieses ist bei der vorliegenden Stichprobe von im Umgang mit Computern geschulten Softwareentwicklern jedoch nicht zu erwarten.

5.4 Stichprobe und Durchführung der Studie

Für den Vergleich der Arbeitsgestaltung in Open-Source-Projekten mit der Arbeitsgestaltung in proprietären Softwareentwicklungsprojekten wurden zwei Stichproben erhoben. Die Open-Source-Entwickler wurden auf zwei Arten kontaktiert. Eine Methode war eine persönliche E-Mail an eine zufällige Stichprobe von Kontakten innerhalb von SourceForge.net. Dieses ist eine Entwicklerplattform, die für alle Open-Source-Projekte eine Infrastruktur zur Verwaltung und Distribution ihrer Software anbietet. Zum anderen wurde an große Projekte, die häufig nicht auf SourceForge.net präsent sind, eine Nachricht an die Verteilerliste der Entwickler verschickt.

Der Onlinefragebogen konnte in der Zeit vom 04.01.2006 bis 20.11.2006 ausgefüllt werden. Die Nachrichten an einzelne Entwickler und die Verteilerlisten wurden im Zeitraum Februar bis Ende April 2006 verschickt. Die TeilnehmerInnen aus der proprietären Softwareentwicklung wurden zum größten Teil von einem großen deutschen IT-Dienstleister rekrutiert. Das Produktportfolio des Dienstleisters umfasst standardisierte Unternehmenslösungen für den Finanzsektor und individuelle Softwareentwicklung im Kundenauftrag. Dabei erhielten die ca. 800 Softwareentwickler des Unternehmens eine Einladung zur Teilnahme an der Untersuchung. Der Befragungszeitraum lag in der Zeit vom 01.05.2006 bis 31.05.2006. Ergänzend wurden im Zeitraum April bis Oktober 2006 vereinzelte Entwickler aus verschiedenen anderen anonymen Unternehmen über einen E-Mail-Verteiler zur Teilnahme gebeten. Im Fragebogen für die Open-Source-Entwickler wurde abgefragt, ob diese neben ihrer Open-Source-Tätigkeit auch in der proprietären

Softwareentwicklung tätig sind. Soweit diese ihr Einverständnis gaben, wurden sie im September 2006 gebeten, zusätzlich den Fragebogen für die proprietäre Softwareentwicklung auszufüllen.

Die erste Seite des Open-Source-Fragebogens wurde im Befragungszeitraum insgesamt 2.011-mal aufgesucht. Die zweite Seite, die auch die erste Frage enthielt, wurde insgesamt von noch 811 Personen aufgesucht, ein großer Teil brach danach die Befragung ab. Nur 23 % der 2011 Besucher sind zur zweiten Seite mit Fragen fortgeschritten. Von den TeilnehmerInnen, die zumindest bis zur ersten Frage fortschritten sind, beendeten 170 (21 %) die gesamte Befragung.

Die Startseite des Fragebogens für die proprietäre Softwareentwicklung wurde insgesamt von 312 Personen besucht. Davon sind 237 bis auf die erste Seite mit Fragen fortgeschritten und 81 Personen haben den Fragebogen bis zum Ende durchlaufen. Dies bedeutet, dass 40 % von denen, die mit der Befragung begonnen haben, auch die letzte Seite betrachteten (vgl. Tabelle 10).

Tabelle 10: Fortschritte der TeilnehmerInnen bei der Beantwortung des Onlinefragebogens

	Open-Source-Softwareentwicklung	Proprietäre-Softwareentwicklung
Startseite aufgerufen	2011	312
Bis zur ersten Frage fortgeschritten	811	237
Bis zur letzten Seite fortgeschritten	170	81

Es wurden insgesamt 43 Open-Source-Entwickler angeschrieben, die neben ihrer Open-Source-Tätigkeit ebenfalls in der proprietären Softwareentwicklung beschäftigt sind. Der Rücklauf von Fragebögen zu ihrer Tätigkeit in der proprietären Softwareentwicklung belief sich bei dieser Teilstichprobe auf insgesamt 8 (19 %) vollständig ausgefüllte Fragebögen. Von diesen Personen wurde aufgrund der Abhängigkeit der Gruppen ausschließlich der Fragebogen zu ihrer Open-Source-Tätigkeit in die allgemeine Auswertung einbezogen.

Bei der Analyse der fehlenden Werte zeigte sich, dass eine Reihe derer, die bis zur letzten Seite fortgeschritten sind, den Fragebogen nur gelesen, jedoch keine Fragen beantwortet haben. Bei den TeilnehmerInnen aus dem Open-Source-Bereich traf dies auf 33 Fälle (19 %) und bei den Beantwortern des Fragebogens des Unternehmens auf 13 Fälle (14 %) zu. Die verbleibenden 137 Fälle (Open-Source-Stichprobe) und 81 Fälle (Stichprobe der proprietären Softwareentwickler) weisen insgesamt einen Anteil von weniger als 0,8 %

fehlender Werte auf. Aufgrund der relativ geringen Stichprobengröße wurde auf eine fallweise Löschung von Daten verzichtet. Um dennoch alle Datensätze nutzen zu können, sollte fehlende Werten ersetzt werden.

Ein häufig eingesetztes Imputationsverfahren ist das Ersetzen von fehlenden Werten durch das arithmetische Mittel. Dieses birgt jedoch den Nachteil, dass hierbei Varianzen und Kovarianzen systematisch unterschätzt werden (Spieß, 2008). Daher wurden die fehlenden Werte durch die Technik Multiple Imputation ergänzt. Bei dieser Technik werden für fehlende Werte eines Datensatzes "plausible Werte" generiert (Schafer, 1997). Angewendet wurde der "Expectation-Maximization" (EM)-Algorithmus. Bei diesem Algorithmus werden in einem iterativen Prozess die Parameter der Wahrscheinlichkeitsverteilung geschätzt und über ein Maximum-Likelihood-Verfahren geprüft. Die Berechnung der Multiplen Imputation wurde mit der Statistik-Software LISREL (Version 8.54) für Windows durchgeführt. Die Werte für die Skalen Interdependenz der Ziele und Identifikation mit dem Team wurden nicht ergänzt, da einige der TeilnehmerInnen kein Team mit gemeinsamen Zielen bzw. gar kein Team aufwiesen. Insgesamt wurde für 0,74 % der Werte die Multiple Imputation ergänzt. Der EM-Algorithmus konvergierte nach 6 Iterationen.

Sowohl in der Open-Source-Stichprobe als auch in der Stichprobe der proprietären Entwickler ist der überwiegende Teil der TeilnehmerInnen männlich. In der Open-Source-Softwareentwicklung trifft dies auf 91 % der TeilnehmerInnen zu, der Anteil in der proprietären Softwareentwicklung ist mit 86 % vergleichbar hoch (vgl. Tabelle 11).

Tabelle 11: Geschlecht der TeilnehmerInnen (Angaben in Prozent)

	Open-Source-Software-entwicklung	**Proprietäre Software-entwicklung**
Männlich	91 %	86 %
Weiblich	7 %	10 %
Keine Angaben	1 %	4 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Beim Alter ergeben sich zwischen den beiden Teilstichproben deutlichere Unterschiede als beim Geschlecht. Insgesamt sind die TeilnehmerInnen im Open-Source-Bereich jünger als Entwickler aus der proprietären Softwareentwicklung. Der Modus für die Open-Source-TeilnehmerInnen liegt bei der Altersgruppe von 20 bis 30 Jahren, während der Modus für die proprietären Softwareentwickler bei 31 bis 40 Jahren liegt (vgl. Tabelle 12).

Tabelle 12: Altersgruppen der TeilnehmerInnen (Angaben in Prozent)

	Open-Source-Software-entwicklung	Proprietäre Software-entwicklung
Unter 20 Jahren	1 %	0 %
20 bis 30 Jahre	56 %	22 %
31 bis 40 Jahre	27 %	30 %
41 bis 50 Jahre	10 %	37 %
Über 50 Jahre	6 %	10 %
Keine Angaben	1 %	1 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Weniger groß sind die Unterschiede in der Erfahrung mit der Programmiertätigkeit. Der Median der Programmiererfahrung der Open-Source-Entwickler liegt bei 10 Jahren, der der proprietären Softwareentwicklung bei 15 Jahren. Die Hälfte der Stichprobe der Open-Source-Entwickler hat 5 bis 17 Jahre Erfahrung, während 50 % der proprietären Softwareentwickler 8 bis 20 Jahre Erfahrung haben (vgl. Abbildung 5).

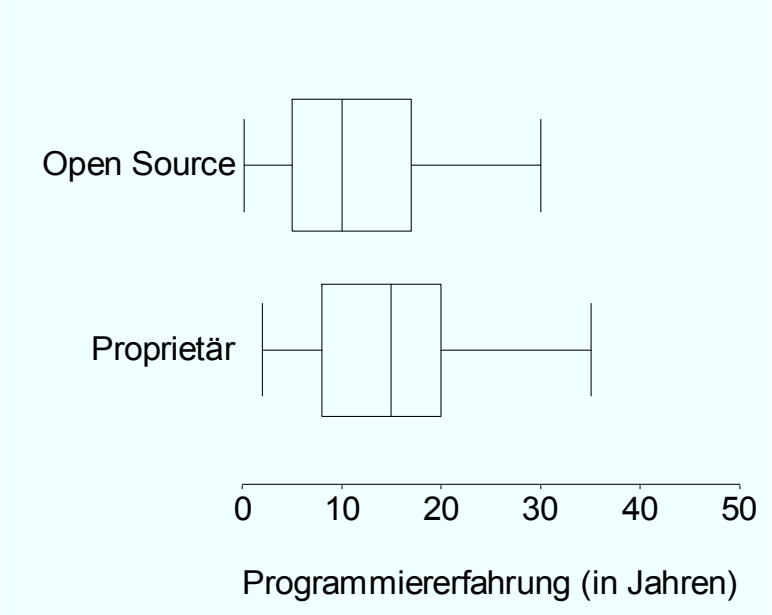

Abbildung 5: Programmiererfahrung in Jahren

Ein deutlicher Unterschied zeigt sich in der Dauer der Zugehörigkeit zu den Open-Source-Projekten im Vergleich mit der Zugehörigkeit zum aktuellen Arbeitgeber in der proprietären Softwareentwicklung. Die Open-Source-Entwickler sind erst seit einem deutlich kürzeren Zeitraum in ihren Projekten (Median von 18 Monaten) als die

proprietären Entwickler bei ihrem aktuellen Arbeitgeber (Median von 96 Monaten). Die Verteilung der Zugehörigkeitsdauer ist in Abbildung 6 dargestellt.

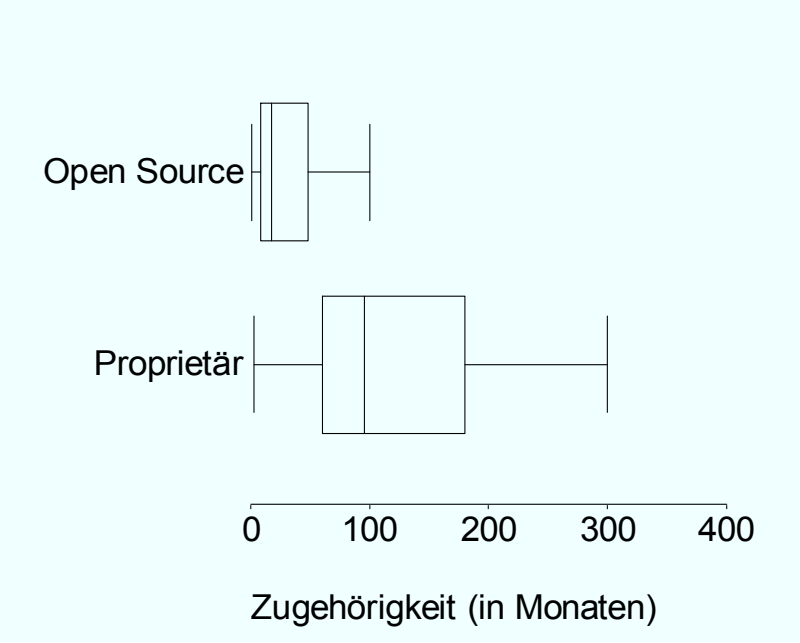

Abbildung 6: Zugehörigkeit zu Open-Source-Projekt bzw. Unternehmen (in Monaten)

Wie schon aus früheren Studien bekannt (Hertel et al., 2003; Luthiger, 2005), sind Open-Source-Projekte häufig international zusammengesetzt. Insgesamt kamen die TeilnehmerInnen aus 34 verschiedenen Ländern. Das Land mit den meisten TeilnehmerInnen ist die USA (22 %), danach folgen Deutschland (20 %) und Großbritannien (7 %). Mit mehr als der Hälfte der TeilnehmerInnen ist Europa der am stärksten repräsentierte Kontinent, gefolgt von Nordamerika mit 23 % (vgl. Tabelle 13).

Tabelle 13: Herkunftskontinente der Open-Source-TeilnehmerInnen (Angaben in absoluten Zahlen und Prozent)

Kontinent	Anzahl	Prozent
Afrika	1	1 %
Asien	14	10 %
Australien	1	1 %
Europa	74	54 %
Nordamerika	32	23 %
Südamerika	11	8 %
Keine Angaben	4	3 %
Gesamt	137	100 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Die Stichprobe der proprietären Softwareentwicklung rekrutiert sich zum größten Teil aus einem deutschen Unternehmen. Entsprechend stammen auch 88 % der TeilnehmerInnen aus Deutschland und insgesamt 95 % aus Europa. Die restlichen TeilnehmerInnen kommen aus Nordamerika.

Aufgrund der internationalen Ausrichtung der Open-Source-Gemeinschaft ist in vielen Projekten Englisch die Projektsprache. Von den Entwicklern, die den Fragebogen ausgefüllt haben, sind über 80 % Muttersprachler oder bezeichnen ihre Englischfähigkeiten als gut. Die IT-Branche ist insgesamt stark durch englische Vokabeln und Fachliteratur geprägt, sodass auch bei der Stichprobe der proprietären Softwareentwickler 45 % Muttersprachler sind oder über gute Englischkenntnisse verfügen (vgl. Tabelle 14).

Tabelle 14: Englischkenntnisse der TeilnehmerInnen (Angaben in Prozent)

	Open-Source-Software-entwicklung	Proprietäre Software-entwicklung
Muttersprache	31 %	5 %
Gut	55 %	40 %
Durchschnittlich	13 %	49 %
Schwach	1 %	5 %
Keine Angaben	0 %	1 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Von den Open-Source-Entwicklern in der Stichprobe befinden sich 22 % in der Ausbildung, die überwiegende Mehrheit (70 %) ist neben der Open-Source-Softwareentwicklung auch beruflich in der IT-Branche tätig. Vergleichbar mit der proprietären Stichprobe ist ein großer Teil als Softwareentwickler oder -architekt beschäftigt. Ein geringerer Teil sieht in beiden Stichproben seine Hauptfunktion in der IT-Beratung bzw. in Managementaufgaben in der IT-Branche (vgl. Tabelle 15).

Tabelle 15: Ausgeübte Berufsbereiche (Angaben in Prozent)

	Open-Source-Software-entwicklung	Proprietäre Software-entwicklung
Schüler/Student	22 %	0 %
IT-Manager	7 %	14 %
IT-Berater	10 %	9 %
Softwarearchitekt	18 %	32 %
Systementwickler	24 %	31 %
Datenbankarchitekt	3 %	2 %
Tester	0 %	2 %
IT-Support	0 %	1 %
Andere IT-Berufe	7 %	5 %
Sonstige Berufe	6 %	0 %
Arbeitslos/Rentner	1 %	0 %
Keine Angaben	3 %	4 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Für diese Untersuchung wurde eine Zusammenstellung an Tätigkeitsarten erfasst, die vom Autor zusammengestellt wurde und die sich an den unterschiedlichen Tätigkeiten des Softwarelebenszyklus und den bisher untersuchten Tätigkeitsarten bei Tätigkeitsanalysen in der Softwareentwicklung orientiert. Zwischen der Open-Source- und der proprietären Teilstichprobe gibt es nennenswerte Unterschiede in der Verteilung der Tätigkeitsarten (vgl. Tabelle 16). Ein besonders hoher Anteil der Open-Source-Entwickler gibt an, dass seine Haupttätigkeit im Bereich Codieren bzw. Fehlerbeseitigen (Debuggen) liegt. Dafür sind die Anteile Softwaredesign und Management bei den proprietären Entwicklern höher. Diese Unterscheidung bestätigt den in Kapitel 3.2 beschrieben Fokus von Open-Source-Entwicklern auf der tatsächlichen Entwicklung der Software und einen geringen Zeiteinsatz für Design- und Architekturtätigkeiten.

Tabelle 16: Verteilung der Tätigkeitsarten

Tätigkeitsart	Open-Source-Softwareentwicklung	Proprietäre Softwareentwicklung
Requirements Engineering	4 %	5 %
Software Design	19 %	28 %
User Interface Design	3 %	0 %
Codieren/Debuggen	47 %	25 %
Testen	4 %	6 %
Dokumentation	1 %	0 %
Systemadministration	3 %	5 %
Management/Koordination	6 %	23 %
User-Support	2 %	2 %
Übersetzung/Lokalisierung	4 %	0 %
Andere	7 %	5 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

In den bisher bekannten Untersuchungen (z. B. Hars & Ou, 2002; Hertel et al., 2003; Lakhani & Wolf, 2003) bekam ein großer Teil der Open-Source-Entwickler seine Tätigkeit in Open-Source-Projekten vergütet. Auch in dieser Studie trifft auf 44 % der TeilnehmerInnen von Open-Source-Projekten zu, dass ihre Tätigkeit ganz oder teilweise vergütet wird. Dies stimmt weitgehend mit Daten von Lakhani und Wolf (2003) überein. Die häufigste Form der Entlohnung ist die Anstellung als Vollzeitkraft (22 %) bzw. als Freiberufler (15 %). 8 % befinden sich in einem Teilzeitarbeitsverhältnis, während der überwiegende Teil (53 %) keine Bezahlung für seine Open-Source-Entwicklungstätigkeit erhält. Für ungefähr 20 % der TeilnehmerInnen stellt die Bezahlung für ihre Tätigkeit in Open-Source-Projekten ihre Haupteinkommensquelle dar und mehr als 27 % beziehen 40 % ihres Einkommens durch diese Aktivitäten (vgl. Abbildung 7).

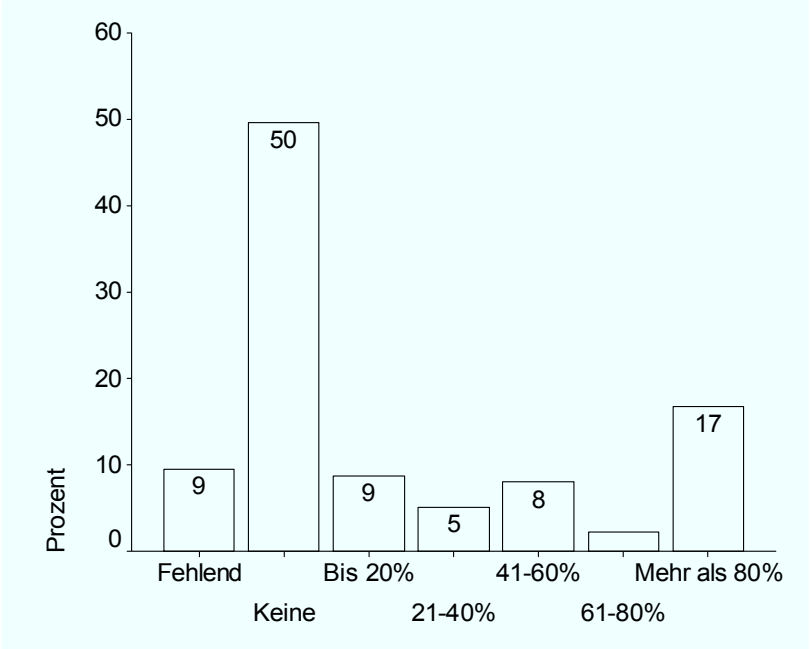

Abbildung 7: Relativer Anteil des Einkommens durch Open-Source-Aktivitäten am Gesamteinkommen

Die Softwareentwicklung in Open-Source-Projekten ist für viele TeilnehmerInnen ein wichtiger Bestandteil ihrer Freizeit. Mehr als 40 % ihrer Freizeit wird von fast 40 % der Entwickler investiert (vgl. Abbildung 8).

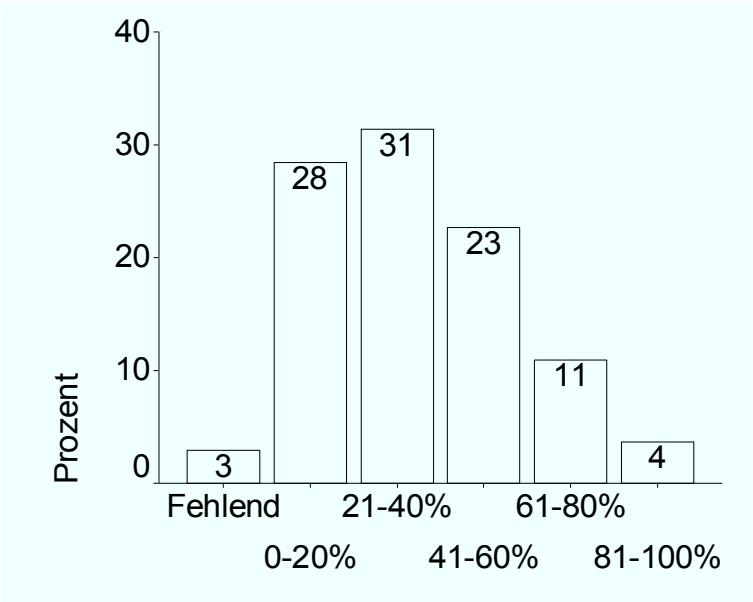

Abbildung 8: Mit Open-Source-Aktivitäten verbrachte Freizeit

Wenig überraschend arbeitet der größte Teil der Entwickler (96 %) in einem Unternehmen, während Open-Source-Softwareentwickler hauptsächlich zu Hause tätig sind (vgl. Tabelle 17). Betrachtet man nur die Open-Source-Entwickler, die als Vollzeitmitarbeiter angestellt sind, so befinden sich deren Arbeitsstätten zu fast drei Viertel (74 %) in einem Unternehmen. Es sind überwiegend Selbstständige und unbezahlte Entwickler, die ihren Arbeitsplatz zu Hause bzw. in der Universität oder Schule haben.

Tabelle 17: Arbeitsorte der Entwickler (Angaben in Prozent)

	Open-Source-Softwareentwicklung	Proprietäre Softwareentwicklung
Zu Hause	59 %	0 %
Universität/Schule	13 %	0 %
Unternehmen	23 %	96 %
Andere	4 %	4 %
Keine Angaben	2 %	0 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Der Vergleich der wichtigsten Kommunikationsmedien in der Open-Source- und der proprietären Softwareentwicklung zeigt eine stärkere Nutzung elektronischer Medien in Open-Source-Projekten. So geben insgesamt 82 % der Open-Source-Entwickler an, dass sie hauptsächlich elektronische textbasierte Medien (E-Mail, Chat, E-Mail-Verteiler oder Foren) nutzen, während diese in der proprietären Softwareentwicklung nur von 27 % der TeilnehmerInnen als wichtigstes Kommunikationsmedium angegeben werden. Dagegen nennen nur 68 % der proprietären Entwickler nicht-textbasierte Medien wie das Telefon oder das persönliche Gespräch (Face-to-Face) als hauptsächliche Kommunikationsform, was in der Open-Source-Entwicklung nur in 12 % der Fälle genannt wird (vgl. Tabelle 18).

Tabelle 18: Hauptsächlich genutztes Kommunikationsmedien in der Softwareentwicklung

Kommunikations-medium	Open-Source-Software-entwicklung	Proprietäre Software-entwicklung
Face-to-Face	12 %	51 %
Telefon	0 %	17 %
Fax/Brief	0 %	2 %
E-Mail	32 %	25 %
Chat	14 %	1 %
E-Mail-Verteiler	28 %	1 %
Foren	8 %	0 %
Andere	6 %	2 %
Keine Angaben	1 %	0 %

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Die Größe der untersuchten Open-Source-Projekte schwankt zwischen 1 und 1.000 TeilnehmerInnen. Der Median liegt bei 5 TeilnehmerInnen pro Projekt. Die Anzahl der Mitarbeiter der beteiligten proprietären Unternehmen ist mit einem Median von 3.200 deutlich größer. Durch die Dominanz eines Unternehmens mit ca. 3.000 Mitarbeitern ist die vorliegende Varianz sehr gering (Interquartilsdistanz liegt bei 3.500-3.000=500).

Die überwiegende Mehrheit der TeilnehmerInnen aus beiden Stichproben verbringt zumindest einen Teil der Arbeitszeit in Teamarbeit. Nur 16 TeilnehmerInnen aus den Open-Source-Projekten und 4 TeilnehmerInnen aus der proprietären Softwareentwicklung arbeiten überhaupt nicht in Teams. Obwohl im Open-Source-Bereich Projekte mit mehreren tausend Mitwirkenden existieren, erscheint in dieser Stichprobe die Anzahl der Personen, mit denen direkt zusammengearbeitet wird, überschaubar. In Open-Source-Projekten beträgt der Median der Gruppengröße 4,5 Personen, die Teamgröße im proprietären Umfeld liegt mit einem Median von 7,0 sogar etwas höher. Die Hälfte der Teams im Open-Source-Bereich ist zwischen 3,0 und 8,0 Personen groß (unteres und oberes Quartil), im proprietären Umfeld ist mit einem Quartilsabstand von 8,0 (unteres Quartil=4,0; oberes Quartil=12,0) auch die Streuung geringfügig größer (vgl. Abbildung 9).

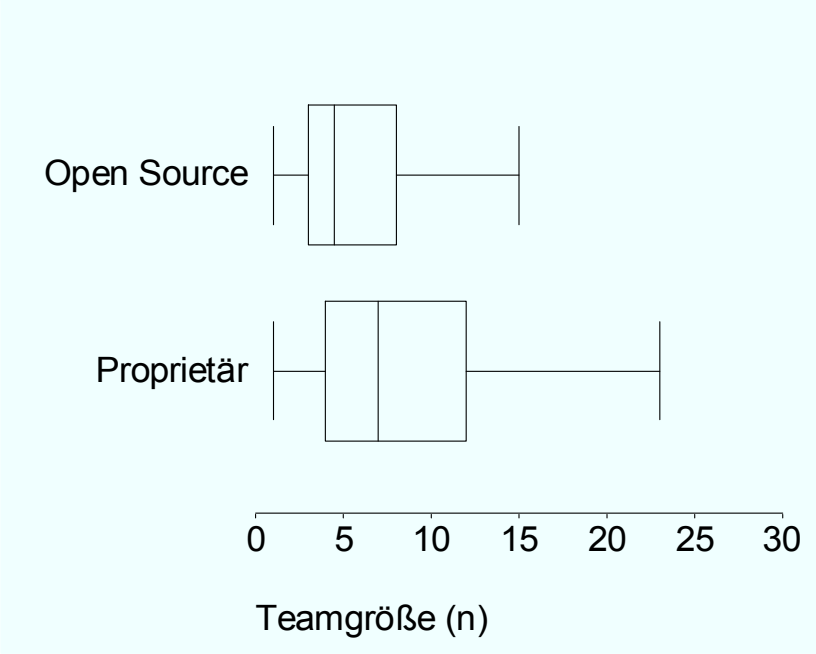

Abbildung 9: Teamgrößen (in Anzahl Personen)

Insgesamt zeigt sich eine Reihe von Übereinstimmungen zwischen den Eigenschaften der vorliegenden Stichprobe mit anderen Untersuchungen (z. B. Hars & Ou, 2002; Hertel et al., 2003; Krishnamurthy, 2002; Lakhani & Wolf, 2003). Im folgenden Kapitel werden die Ergebnisse der Datenanalyse beschrieben.

6 Ergebnisse

Im folgenden Kapitel 6.1 werden zunächst die psychometrischen Eigenschaften des Fragebogens geprüft. Anschließend wird die inferenzstatistische Prüfung der Hypothesen 1 und 2 (Kapitel 6.2) und der Hypothese 3 (Kapitel 6.3) vorgenommen. Hinzu kommt noch die Untersuchung des Moderators Entlohnung in Kapitel 6.4. In Kapitel 6.5 werden Unterschiede in der Produktivität der Programmierer detailliert untersucht und im abschließenden Kapitel 6.4 werden in einer explorativen Betrachtung die Zusammenhänge der sozialen Aspekte der Tätigkeit mit diversen Moderatoren bestimmt. Für die Auswertung wurde, soweit nicht anders vermerkt, die Software SPSS in der Version 10.0.7 verwendet. Vor der empirischen Analyse wurden invers formulierte Items entsprechend recodiert.

6.1 Überprüfung des Fragebogens

Zur Überprüfung des Fragebogens wird für die erhobenen Skalen die interne Konsistenz bestimmt (Kapitel 6.1.1). Die Übereinstimmung der Daten mit den aufgestellten Messmodellen wurde mithilfe einer konfirmatorischen Faktorenanalyse getestet (Kapitel 6.1.2).

6.1.1 Interne Konsistenz

Wie in Kapitel 5.2 dargestellt, orientierte sich die Konstruktion des Fragebogens an bereits bestehenden Instrumenten. Einige Skalen wurden jedoch verkürzt oder durch neue Items ergänzt. Zudem gibt es noch keinerlei Erfahrung mit der Anwendung von subjektiven Tätigkeitsanalysen im Open-Source-Kontext. Dies zusammengenommen macht eine Überprüfung der Reliabilität der Skalen notwendig. Ein hierfür notwendiger Schritt stellt die Bestimmung der internen Konsistenz der Skalen dar. Als Kenngröße der internen Konsistenz wird Cronbachs α-Koeffizient herangezogen (Bortz, 1999). Ein α-Koeffizient größer als 0,7 wird vor dem Hintergrund des explorativen Charakters dieser Studie im Allgemeinen als ausreichend angesehen (Nunnally, 1978). Wie Tabelle 19 zu entnehmen ist, trifft dieses auf einen Großteil der Skalen zu.

Tabelle 19: Deskriptive Statistik und interne Konsistenz der Skalen

Skala	Item-Anzahl	N	Mittelwert	Standardabw.	Interne Konsistenz*
Aufgabenvielfalt	2	218	4,24	0,57	0,70
Anforderungsvielfalt	2	218	4,08	0,65	0,67
Autonomie (Gesamt)	6	218	4,08	0,60	0,84
Autonomie (Einteilung der Arbeitsabfolge)	2	218	4,06	0,73	0,73
Autonomie (Treffen von Entscheidungen)	2	218	4,16	0,67	0,60
Autonomie (Auswahl der Methoden)	2	218	4,03	0,71	0,67
Partizipation	2	218	3,75	0,82	0,67
Aufgabengeschlossenheit	2	218	3,86	0,85	0,90
Bedeutsamkeit der Aufgabe	2	218	3,53	0,93	0,75
Rückmeldung durch die Tätigkeit	2	218	3,62	0,88	0,87
Interdependenz (Aufgabe)	4	218	3,45	0,75	0,72
Rückmeldung durch andere Personen	2	218	3,61	0,81	0,83
Soziale Unterstützung	4	218	3,64	0,67	0,76
Interdependenz (Ziele)	2	196	3,72	0,73	0,57
Qualifikationspotenzial	2	218	4,17	0,74	0,79
Allgemeine Arbeitszufriedenheit	3	218	3,92	0,73	0,74
Intrinsische Arbeitsmotivation	3	218	4,02	0,72	0,54
Organizational Citizenship	7	218	3,87	0,55	0,59
Organisationale Identifikation (Gesamt)	4	218	3,99	0,70	0,76
Organisationale Identifikation (Unternehm.)	2	218	3,91	0,78	0,76
Organisationale Identifikation (Abteilung)	2	207	4,07	0,94	0,93
Leistung (Allgemein)	2	218	1,93	0,84	0,79
Flow	2	218	3,51	0,82	0,74

* α-Koeffizient nach Cronbach

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Bei den Tätigkeitsmerkmalen weisen die beiden Autonomieunterskalen "Treffen von Entscheidungen" (α=0,60) und "Auswahl der Methoden" (α=0,67) eine leicht unbefriedigende interne Konsistenz auf. Aus diesem Grund werden die drei Unterskalen von Autonomie zusammengefasst und nur die Gesamtskala Autonomie (α=0,84) ausgewertet. Die Skala Partizipation befindet sich ebenfalls unterhalb des kritischen Wertes von 0,7. Mit einem Koeffizienten von 0,67 ist dieser nur geringfügig unterschritten und daher wird die Skala unverändert beibehalten. Gleiches gilt für die Skala des Tätigkeitsmerkmals Anforderungsvielfalt, deren Koeffizient ebenfalls 0,67 beträgt. Die geringe Konsistenz der Skala Interdependenz der Ziele ist auf eine uneinheitliche Beantwortung innerhalb der Stichprobe der proprietären Softwareentwicklung zurückzuführen. In dieser Teilstichprobe beträgt Cronbachs α nur inakzeptable 0,19, während er in der Open-Source-Stichprobe mit 0,68 nur geringfügig den Wert von 0,7 unterschreitet. Die Skala Interdependenz der Aufgaben wurde entsprechend theoretischen Überlegungen

(vgl. 4.4.3) in die beiden Subskalen "Initiierte Aufgabeninterdependenz" und "Erhaltene Aufgabeninterdependenz" aufgeteilt. Da sich für die Skala "Erhaltene Aufgabeninterdependenz" eine Konsistenz von nur 0,67 ergab, wurde aus Gründen der Reliabilität auf eine Aufteilung der Skalen für weitere Analysen verzichtet.

Mit Ausnahme von Organizational Citizenship und der intrinsischen Motivation liegen die Skalen zu organisationalen Kriterien über dem Wert von 0,7. Die geringe Konsistenz von α=0,59 der Organizational Citizenship Skala ist damit zu erklären, dass für die Konstruktion der vorliegenden Skala aus allen fünf von Organ (1988) postulierten Bereichen Items ausgewählt wurden. Da die OCB-Skala als formativ betrachtet werden muss, ist Cronbachs α kein geeignetes Kriterium zur Beurteilung der Skalengüte (Diamantopoulos & Winklhofer, 2001). Das Eliminieren von Items könnte zu einer Reduktion der Validität führen. Daher werden alle 7 Items in die endgültige Skala aufgenommen.

Eine Trennschärfenanalyse für die Skala zur intrinsischen Motivation ergab, dass die Aussage "I feel bad and unhappy when I discover that I have performed poorly on this job." insbesondere in der Stichprobe der Open-Source-Entwickler nur einen geringen Zusammenhang zur Gesamtskala aufweist. Durch die Entfernung dieses Items erhöht sich die interne Konsistenz von 0,54 auf den akzeptablen Wert von 0,71. Ein Vergleich der internen Konsistenz der Open-Source- und der Stichprobe der proprietären Entwickler zeigte weiter keine Unterschiede.

6.1.2 Überprüfung des Messmodells

Für die Bestätigung des Messmodells der Tätigkeitsmerkmale und den organisationalen Kriterien soll jeweils eine konfirmatorische Faktorenanalyse durchgeführt werden. Zu diesem Zweck wurde sowohl für die Tätigkeitsmerkmale als auch die organisationalen Kriterien ein Strukturgleichungsmodell in LISREL aufgebaut und die Qualität der Übereinstimmung dieses Modells mit den erhobenen Daten geprüft.

Im ersten Schritt wurde das Messmodell für die Tätigkeitsmerkmale geprüft (vgl. auch Modell 1 in Abbildung 10). Bei einer konfirmatorischen Faktorenanalyse für die Skalen Aufgabenvielfalt, Anforderungsvielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe, Rückmeldung durch die Tätigkeit, Interdependenz der Aufgabe, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz der Ziele und Qualifikationspotenzial (Messmodell Nr. 1) ergab sich ein signifikantes X^2 (398) von 621,8 ($p<0,001$). Dies spricht für eine fehlende Übereinstimmung der Daten mit dem Messmodell, wobei der X^2-Test nicht sehr aussagekräftig ist (Backhaus, Erichson, Plinke & Weiber, 1996). Aus diesem Grund wurden weitere Kennwerte zur Überprüfung des Messmodells herangezogen. Der Goodness of Fit Index (GFI) liegt mit

0,83 unter der erwünschten Grenze von 0,90 und der Root Mean Square Error of Approximation (RMSEA) ist mit 0,052 leicht über der Grenze von 0,05, die für ein gutes Modell angesetzt wird. Da der Index jedoch deutlich unter 0,08 bleibt, kann man zumindest von einem adäquaten Modell ausgehen (Fan, Thompson & Wang, 1999). Neben dem RMSEA wird insbesondere der Comparative Fit Index (CFI) nur im geringen Ausmaß durch die Stichprobengröße beeinflusst. Dieser liegt mit 0,97 deutlich über der Akzeptanzschwelle von 0,9 (Fan et al., 1999). Mit diesen Werten und in Anbetracht der geringen Stichprobengröße wird das Messmodell als bestätigt betrachtet.

In Kapitel 4 wurde ausgeführt, dass es große Unterschiede in der Auswahl der Skalen bei den gebräuchlichen Instrumenten zur Tätigkeitsanalyse gibt. In diesen Fragebogen wurden einige Skalen aufgenommen, die eine hohe inhaltliche Übereinstimmung aufweisen, und es wurde bisher nur in wenigen Untersuchungen überprüft, ob die vorgenommenen Differenzierungen sinnvoll sind. Aus diesem Grund soll geprüft werden, ob die vorgenommene Unterscheidung der Untermerkmale von Kontrolle, Vielfalt und Rückmeldung bei der vorliegenden Stichprobe gerechtfertigt ist. Hierfür wurden die Modelle 2 bis 9 aufgestellt. Die Modelle bestehen aus unterschiedlichen Kombinationen der Untermerkmale von Kontrolle, Vielfalt und Rückmeldung. Die Modelle 1 bis 4 sind in Abbildung 10 und die Modelle 5 bis 8 in Abbildung 11 dargestellt. Das Messmodell 9 ist weiter unten im Text dargelegt.

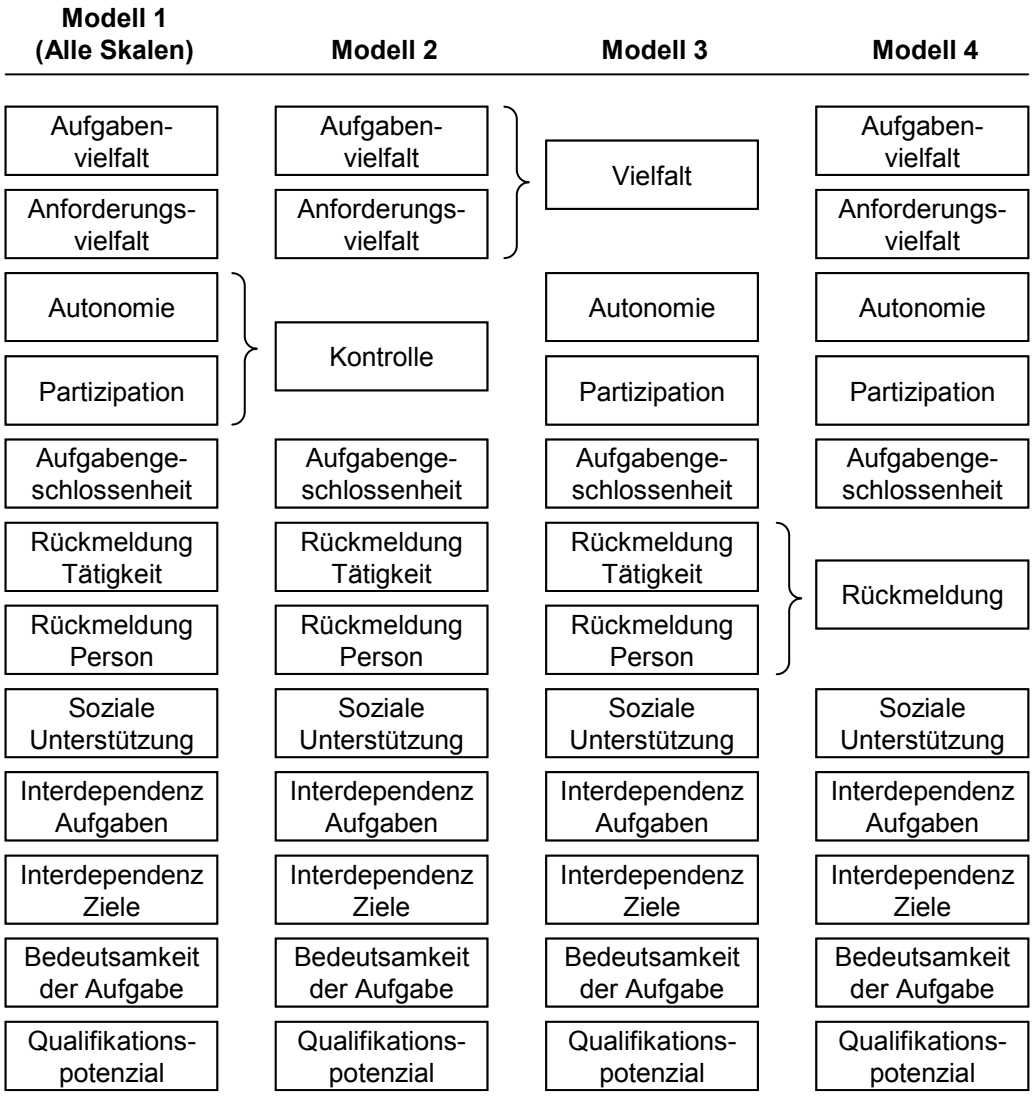

Abbildung 10: Geprüfte Messmodelle 1-4 für Skalen Tätigkeitsmerkmale

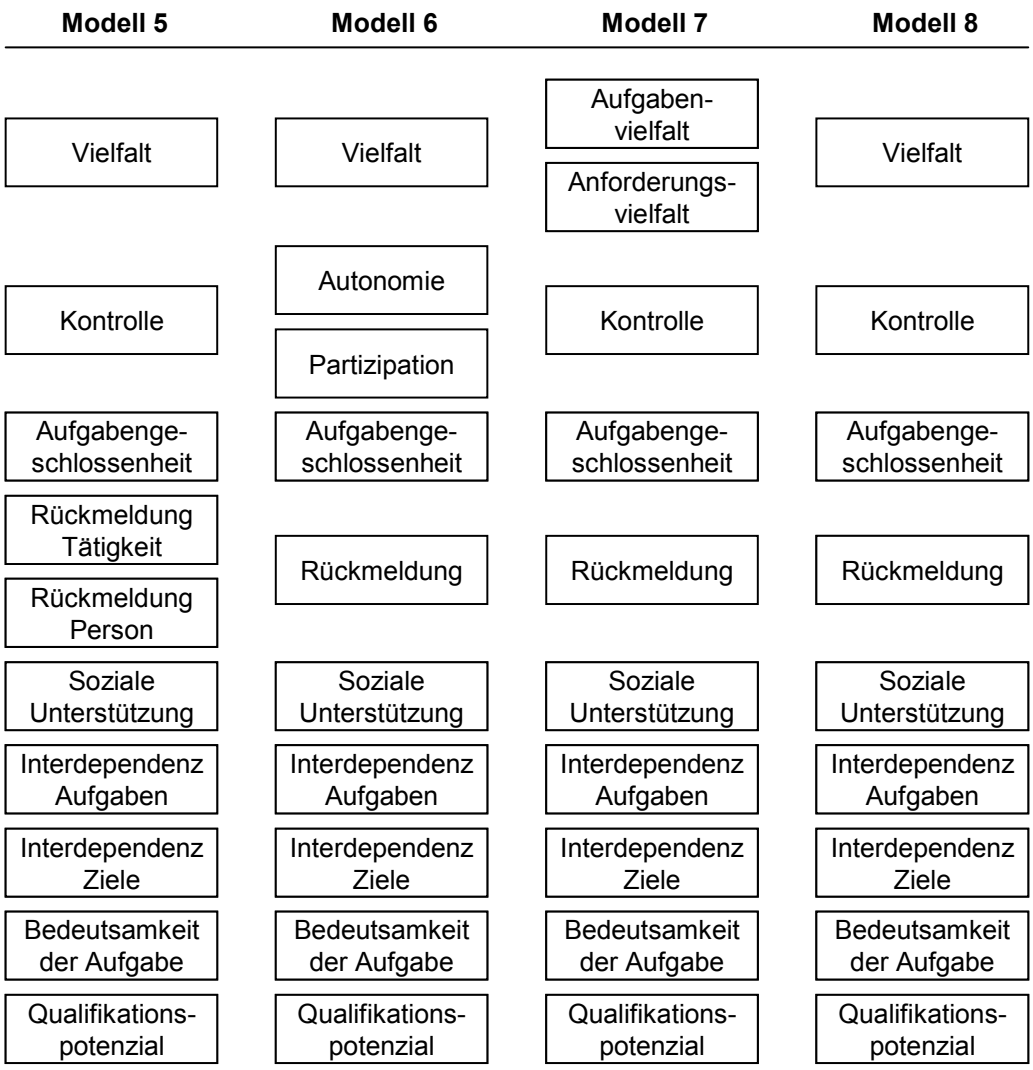

Abbildung 11: Geprüfte Messmodelle 5-8 für Skalen Tätigkeitsmerkmale

Das Modell Nummer 2 fasst die beiden Skalen Autonomie und Partizipation zu der Skala Kontrolle zusammen. Auch bei diesem Modell ist das X^2 (409) mit 694,4 hoch signifikant, der Wert für RMSEA entfernt sich mit 0,060 deutlich von den angestrebten 0,05 und auch GFI und CFI liegen niedriger als das Modell mit allen Skalen (vgl. Tabelle 20). Dieses Modell zeigt damit eine schlechtere Anpassung an die erhobenen Daten als Modell 1. Messmodell Nummer 3 fasst Anforderungsvielfalt und Aufgabenvielfalt zu Vielfalt und ist ebenfalls hoch signifikant (X^2=598,0; df=409). Der RMSEA liegt im Vergleich zum Gesamtmodell mit 0,051 etwas näher an 0,05 und der GFI ist mit 0,84 geringfügig höher, während der CFI mit 0,97 identisch ist. Damit zeigt dieses Modell einen höheren Fit als das Ausgangsmodell mit allen Skalen (Modell 1). Die Bildung einer allgemeinen Rückmeldungsskala aus Rückmeldung durch die Tätigkeit und Rückmeldung durch andere ist

in Modell 4 zusammengefasst und auch hoch signifikant (X^2=740,7; df=409), zeigt aber schlechtere Werte als das Modell 1 mit allen Skalen für alle Güteindizes (vgl. Tabelle 20).

Neben dem Zusammenfassen einzelner Skalen wurden auch die möglichen Kombinationen aus der Zusammenfassung der Skalen Kontrolle, Vielfalt und Rückmeldung permutiert (Modell 5 bis 8). Diese wiesen jedoch alle Güteindizes auf, die ausnahmslos schlechter waren als die Indizes des Modells 1 mit allen Skalen (vgl. Tabelle 20).

Basierend auf den Überlegungen von Morgeson und Campion (2003) soll überprüft werden, ob das Messmodell nicht am besten über die beiden übergeordneten Merkmale Komplexität und soziale Aspekte der Tätigkeit beschrieben werden kann. Für das Modell 9 werden dazu die Items für Aufgabenvielfalt, Anforderungsvielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit, Bedeutsamkeit der Aufgabe und Qualifikationspotenzial der Komplexitätsskala zugeordnet. Die Skala der sozialen Aspekte setzt sich aus den Items der Interdependenz der Aufgabe und der Ziele, der Rückmeldung durch Personen und der sozialen Unterstützung zusammen. Das Messmodell 9 ist mit X^2=1538,6 (df=463) ebenfalls hoch signifikant, weist aber nach den anderen Güteindizes die schlechteste Anpassung an die vorhandenen Werte auf (vgl. Tabelle 20).

Betrachtet man die Ergebnisse aller untersuchten 9 Modelle, so weist das Modell 3 mit 11 Skalen, bei dem die beiden Skalen Anforderungsvielfalt und Aufgabenvielfalt zusammengefasst sind, eine leicht höhere Übereinstimmung mit den vorhandenen Daten als das Modell 1 mit allen 12 Skalen auf. In den folgenden Abschnitten wird daher das Modell 3 mit den Skalen Vielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe, Rückmeldung durch die Tätigkeit, Interdependenz der Aufgabe, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz der Ziele und Qualifikationspotenzial verwendet.

Tabelle 20: Gütekriterien für Varianten des Messmodells

Nummer	Beschreibung	Anzahl Skalen	CFI	GFI	RMSEA	X^{2***}	df
1	Alle Skalen*	12	0,97	0,83	0,052	621,8	(398)
2	Zusammenfassung von Autonomie und Partizipation zu Kontrolle	11	0,96	0,82	0,060	694,4	(409)
3	Zusammenfassung von Anforderungsvielfalt und Aufgabenvielfalt zu Vielfalt	11	0,97	0,84	0,051	598,0	(409)
4	Zusammenfassung von Rückmeldung durch Tätigkeit und Andere	11	0,90	0,81	0,064	740,7	(409)
5	Zusammenfassung von Kontrolle und Vielfalt	10	0,96	0,81	0,060	712,8	(419)
6	Zusammenfassung von Rückmeldung und Vielfalt	10	0,95	0,80	0,065	762,2	(419)
7	Zusammenfassung von Rückmeldung und Kontrolle	10	0,94	0,79	0,072	840,6	(419)
8	Zusammenfassung von Kontrolle, Rückmeldung und Vielfalt	9	0,94	0,78	0,072	857,5	(428)
9	Zusammenfassung von Komplexität und sozialen Aspekten**	2	0,84	0,64	0,120	1538,6	(463)

* Aufgabenvielfalt, Anforderungsvielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe, Rückmeldung durch die Tätigkeit, Interdependenz der Aufgabe, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz der Ziele und Qualifikationspotenzial

** Zuordnung zu Komplexität: Aufgabenvielfalt, Anforderungsvielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit, Bedeutsamkeit der Aufgabe und Qualifikationspotenzial; Zuordnung zu sozialen Aspekten: Interdependenz der Aufgabe und der Ziele, Rückmeldung durch Personen, soziale Unterstützung.

*** Alle X^2-Tests ergaben ein hochsignifikantes Ergebnis ($p<0,001$)

Neben dem Messmodell der Tätigkeitsmerkmale wurde auch das Messmodell der organisationalen Kriterien überprüft. Bei diesem Modell wurden die erhobenen organisationalen Kriterien allgemeine Arbeitszufriedenheit, intrinsische Motivation, Leistung, Organizational Citizenship, organisatorische Identifikation mit dem Unter-

nehmen und der Abteilung sowie Flow eingeschlossen. Für dieses Modell ergibt sich ein signifikantes X^2=182,7 (df=83). Der RMSEA liegt bei akzeptablen 0,076, der GFI ist mit 0,90 auf der Grenze für gute Modelle und der CFI ist mit 0,93 ausreichend hoch. Mit diesen Werten kann das Messmodell für die organisationalen Kriterien als ausreichend bestätigt angesehen werden.

Durch die Zusammenfassung von Anforderungsvielfalt und Aufgabenvielfalt hat sich das Messmodell für die Tätigkeitsmerkmale leicht geändert. Das modifizierte Gesamtmodell für die weitere Untersuchung ist in Abbildung 12 dargestellt.

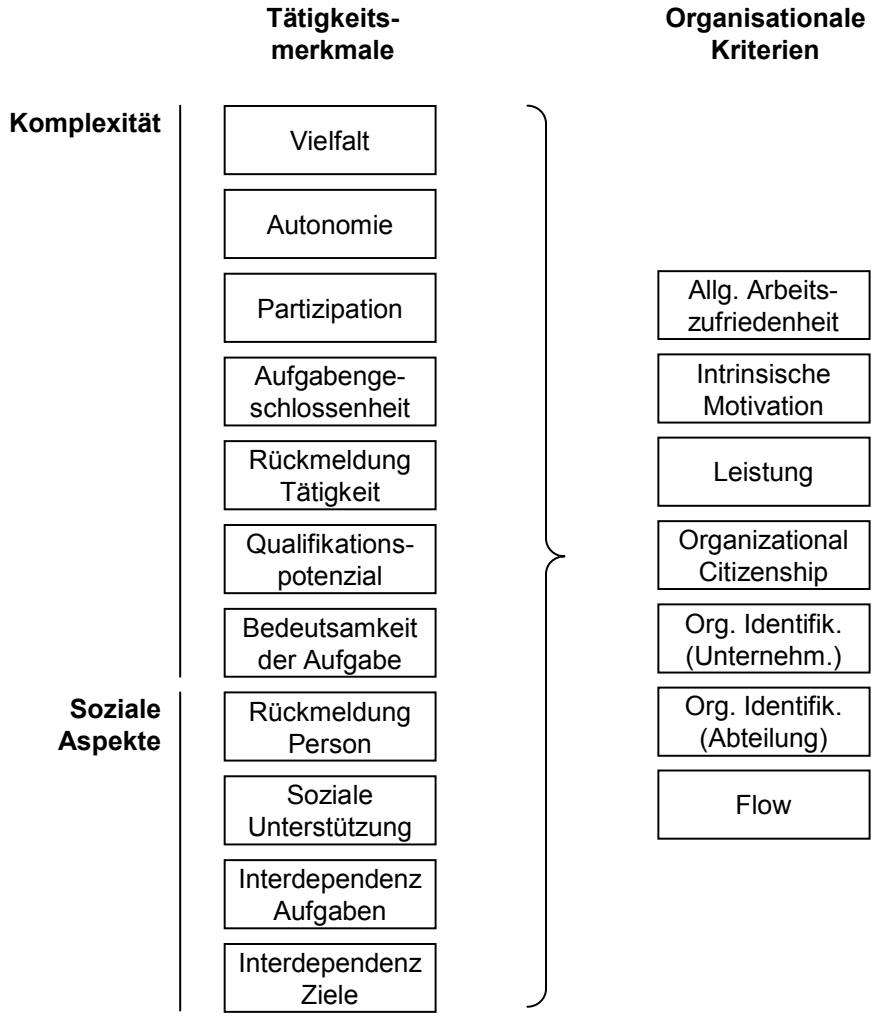

Abbildung 12: Modifiziertes Gesamtmodell

6.2 Tätigkeitsmerkmalen der Komplexität und soziale Aspekte

Im folgenden Abschnitt sollen die Hypothesen 1 und 2 geprüft werden. Diese beiden Hypothesen postulieren Unterschiede zwischen Open-Source- und proprietärer Softwareentwicklung. Hierbei muss berücksichtigt werden, dass eine Reihe von Studien

Unterschiede bei Tätigkeitsmerkmalen für verschiedene Arten von Tätigkeiten in der Software-Entwicklung gezeigt hat (vgl. Kapitel 4.1).

Hypothese 1 postuliert eine höhere Komplexität der Tätigkeitsmerkmale in der Open-Source-Softwareentwicklung im Vergleich zur proprietären Softwareentwicklung. Dies bedeutet, Open-Source-Entwickler verfügen über ein größeres Maß an Autonomie, Partizipation und Aufgabenvielfalt, Aufgabengeschlossenheit und Rückmeldung durch die Tätigkeit als Personen, die in proprietären Softwareprojekten arbeiten. Eine Betrachtung der Mittelwerte dieser Merkmale zeigt für alle Tätigkeiten von Open-Source-Entwicklern einen tendenziell höheren Mittelwert als die Tätigkeiten der proprietären Entwickler. Diese Tendenz kann interpretiert werden als eine positivere Gestaltung der Open-Source-Tätigkeiten im Vergleich zur Tätigkeit der proprietären Entwickler im Hinblick auf die Komplexität (vgl. Tabelle 21).

Zur Prüfung der Hypothese wurde eine multivariate Varianzanalyse durchgeführt mit dem "Modell der Software-Entwicklung" (Open-Source vs. proprietär) als Faktor. Da es sich hier um eine multivariate Varianzanalyse handelt, wurde als Statistik für die Prüfung auf einen signifikanten Unterschied Wilks-Lambda (U) genutzt (Backhaus, Erichson, Plinke & Weiber, 1996). Der Einfluss des Faktors "Entwicklungsmodell" ist hoch signifikant ($U=0,62$; $p=0,000$; $\eta^2=0,38$), womit die Hypothese 1 als bestätigt angesehen werden kann.

Bei einem Vergleich der aufgeklärten Varianz (η^2) für einzelne Tätigkeitsmerkmale zeigt sich, dass bei Autonomie und Partizipation mit jeweils $\eta^2=0,22$ und bei Qualifikationspotenzial mit $\eta^2=0,25$ ein besonders großer Anteil erklärt wird (vgl. Tabelle 21). Bei Effekten, die größer oder gleich 0,14 sind, ist bei einer Varianzanalyse nach der Konvention von Cohen (1969) von großen Effekten auszugehen. Die Merkmale Aufgabengeschlossenheit ($\eta^2=0,09$) und Rückmeldung durch die Tätigkeit ($\eta^2=0,06$) zeigen mittlere Effekte ($\eta^2\geq0,06$), während der Effekt auf die Vielfalt ($\eta^2=0,02$) und Bedeutsamkeit der Aufgabe ($\eta^2=0,03$) klein ist ($\eta^2\geq0,02$).

Tabelle 21: Multivariate Varianzanalyse für die Tätigkeitsmerkmale der Komplexität

Tätigkeitsmerkmale	Mittelwert		F	η^2
	Open-Source-Entwicklung	Proprietäre Entwicklung		
Vielfalt	4,2	4,1	4,67 *	0,02
Autonomie	4,3	3,7	57,31 ***	0,22
Partizipation	4,0	3,3	57,39 ***	0,22
Aufgabengeschlossenheit	4,1	3,5	21,11 ***	0,09
Rückmeldung durch Tätigkeit	3,8	3,4	13,46 ***	0,06
Bedeutsamkeit der Aufgabe	3,7	3,3	7,13 **	0,03
Qualifikationspotenzial	4,5	3,7	68,49 ***	0,25

* p<0,05; ** p<0,01; *** p<0,001

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung. Die Angaben zu F(7,201)-Wert und η^2 beziehen sich auf den Faktor "Entwicklungsmodell" in der Varianzanalyse.

Basis: 137 Open-Source- und 73 proprietäre Softwareentwickler

Hypothese 2 postuliert bei Tätigkeiten in Open-Source-Projekten eine geringere Ausprägung der sozialen Aspekte der Tätigkeit als bei Tätigkeiten in der proprietären Softwareentwicklung. Dies soll sich in einem geringeren Ausmaß an Rückmeldung durch Personen, sozialer Unterstützung sowie Interdependenz der Aufgaben und Ziele ausdrücken. Dieser Hypothese entsprechend ist der Mittelwert für das Tätigkeitsmerkmal Aufgabeninterdependenz bei Open-Source-Tätigkeiten geringer. Die Mittelwertsunterschiede der Merkmale Rückmeldung durch Personen, soziale Unterstützung und Interdependenz der Ziele stimmen nicht mit der Hypothese überein. Sie sind alle tendenziell größer für Open-Source-Tätigkeiten als für proprietäre Softwareentwicklungstätigkeiten (vgl. Tabelle 22).

Der zur Prüfung der Normalverteilung der abhängigen Variablen durchgeführte Levene-Test zeigt für die beiden Merkmale Interdependenz der Aufgaben und der Ziele ein signifikantes Ergebnis. Damit ist eine Normalverteilung der beiden Variablen nicht gesichert und eine Voraussetzung der Varianzanalyse nicht erfüllt (Bortz, 1999). Trotz fehlender Normalverteilung ist bei gleicher Stichprobengröße in den Gruppen die Varianzanalyse weitgehend robust. Da in diesen Fall unterschiedlich große Stichproben vorliegen (Open-Source=120; proprietär=70), wird auf die Teststatistik von Pillai-Bartlett (V) zurückgegriffen, die für diesen Fall robuster ist als Wilks-Lambda (Olson, 1976)[8]. Die multivariate Varianzanalyse zeigt keinen signifkanten Einfluss des Faktors "Entwicklungsmodell" (V=0,04; p=0,098). Von den vier Merkmalen wird nur für die soziale Unter-

[8] Das Wahrscheinlichkeitsniveau bei Anwendung von Wilks-Lambda ist sowohl für die Kovariate als auch den Faktor auf drei Nachkommastellen identisch.

stützung ein signifikanter Anteil der Varianz durch den Faktor "Entwicklungsmodell" aufgeklärt (F(4,185)=4,46; p=0,036). Der vorgefundene Effekt ist als klein zu bezeichnen (η^2=0,02). Die fehlende Signifikanz des Faktors "Entwicklungsmodell" und die teilweise entgegengesetzte Richtung der Mittelwertsdifferenzen führt zu einer Verwerfung der Hypothese 2.

Tabelle 22: Ausprägung Tätigkeitsmerkmale für Open-Source- und proprietäre Softwareentwicklung

Tätigkeitsmerkmale	**Mittelwert**		**F**	**η^2**
	Open-Source-Entwicklung	Proprietäre Entwicklung		
Rückmeldung durch Personen	3,7	3,6	1,04	0,01
Soziale Unterstützung	3,8	3,6	4,46 *	0,02
Interdependenz (Aufgabe)	3,5	3,6	0,55	0,00
Interdependenz (Ziele)	3,8	3,6	1,90	0,01

* p<0,05

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung. Die Angaben zu F-Wert und η^2 beziehen sich auf den Faktor "Entwicklungsmodell" in der Varianzanalyse.

Basis: 120 Open-Source- und 70 proprietäre Softwareentwickler

Sowohl bei der Untersuchung von Hypothese 1 als auch bei Hypothese 2 wurde die Gesamtstichprobe betrachtet. Allerdings unterscheiden sich Open-Source- und proprietäre Stichprobe im Anteil der sich in einem Angestelltenverhältnis befindlichen Personen (vgl. Kapitel 5.4). Da die Tätigkeit von Selbstständigen im Allgemeinen über mehr Freiheitsgrade verfügt (Gerlmaier, 2002), kann dieser Unterschied in den Stichproben zu einer Verfälschung der bisher aufgefundenen Unterschiede zwischen Open-Source- und proprietären Entwicklern geführt haben. Möglicherweise sind die Ergebnisse nicht auf das Entwicklungsmodell zurückzuführen, sondern beruhen nur auf der unterschiedlichen Verteilung von Angestellten und Freiberuflern in der Stichprobe. Daher soll eine zusätzliche Varianzanalyse ausschließlich für die Angestellten berechnet werden. In der Stichprobe der Open-Source- und der Stichprobe der proprietären Softwareentwickler werden nur diejenigen in diese Analyse eingeschlossen, die in einer Voll- oder Teilzeitanstellung stehen.

Für die Überprüfung wurde eine Varianzanalyse durchgeführt mit dem Faktor "Entwicklungsmodell". Es bestätigen sich die Ergebnisse der Gesamtstichprobe: der Faktor "Entwicklungsmodell" hat auch für die Angestellten einen signifikanten Einfluss auf die Merkmale der Komplexität (U=0,63; p=0,000; η^2=0,37). Dabei weisen sämtliche Tätigkeitsmerkmale der Komplexität für die Open-Source-Stichprobe einen höheren Mittelwert auf als die proprietäre Stichprobe (vgl. Tabelle 23). Für die sozialen Aspekte der

Tätigkeit zeigen sich auch für die Angestellten keine signifikanten Unterschiede (U=0,99; p=0,963; η^2=0,06).

Tabelle 23: Ausprägung Tätigkeitsmerkmale für Angestellte in Open-Source- und proprietärer Softwareentwicklung

Tätigkeitsmerkmale	**Mittelwert**		**F-Wert**	**η^2**
	Open-Source	Proprietär		
Komplexität der Tätigkeit				
Vielfalt	4,4	4,1	10,89 **	0,09
Autonomie	4,3	3,7	27,90 ***	0,21
Partizipation	4,0	3,3	21,10 ***	0,17
Aufgabengeschlossenheit	3,9	3,5	6,86 *	0,06
Rückmeldung durch Tätigkeit	3,8	3,4	7,76 **	0,07
Bedeutsamkeit der Aufgabe	4,0	3,4	12,64 ***	0,11
Qualifikationspotenzial	4,5	3,7	36,66 ***	0,26
Soziale Aspekte				
Rückmeldung durch Personen	3,7	3,6	0,27	0,00
Soziale Unterstützung	3,6	3,6	0,27	0,00
Interdependenz (Aufgabe)	3,5	3,6	0,00	0,00
Interdependenz (Ziele)	3,7	3,6	0,40	0,00

* p<0,05; ** p<0,01; *** p<0,001

Separate Varianzanalysen für Komplexität der Tätigkeit und soziale Aspekte. Angabe zu F-Wert und η^2 beziehen sich jeweils auf den Faktor "Entwicklungsmodell".

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung.

Basis: 38 Open-Source-Softwareentwickler (Angestellte) und 71 proprietäre Softwareentwickler (Angestellte)

Bei den bisher durchgeführten Analysen der Unterschiede zwischen Open-Source- und proprietären Entwicklern kann nicht ausgeschlossen werden, dass die vorgefundenen Differenzen zwischen den Tätigkeitsmerkmalen der Open-Source- und der proprietären Softwareentwicklung auf interpersonelle Unterschiede zurückzuführen sind. Durch den Selektionseffekt könnten z. B. nur solche Personen sich in Open-Source-Projekten engagieren, die eine besonders positive Wahrnehmung der Tätigkeitsmerkmale aufweisen. Die vorliegende Untersuchung bietet die Möglichkeit, interpersonelle Varianz zu eliminieren. Wie im Kapitel 5.4 beschrieben, wurden Open-Source-Entwickler gefragt, ob sie neben ihrer Tätigkeit in Open-Source-Projekten auch in der proprietären Softwareentwicklung beschäftigt sind. Entwickler, auf die das zutraf und die sich bereit erklärten, wurden kontaktiert, um zusätzlich zum bereits ausgefüllten Fragebogen über die Tätigkeit im Open-Source-Projekt auch den Fragebogen für die proprietäre Softwareentwicklung auszufüllen. Durch einen erhobenen eindeutigen Identifizierer konnten die Daten der Erhebungen eines Entwicklers für die Open-Source-Tätigkeit und die Tätigkeit in der proprietären Softwareentwicklung zusammengeführt werden. Eine Analyse der

beiden Tätigkeitsbereiche dieser abhängigen Stichprobe ist nicht durch interpersonelle Varianz konfundiert.

Aufgrund der sehr geringen Fallzahl (n=8) und der Abhängigkeit der Stichproben wurde keine Varianzanalyse, sondern ein t-Test für gepaarte Stichproben durchgeführt. Dabei wurden jeweils die Beschreibungen der Tätigkeitsmerkmale einer Person zu Open-Source- und proprietärer Softwareentwicklung gepaart. Die Ergebnisse sind in Tabelle 24 aufgeführt. Zur Prüfung der Angemessenheit des Stichprobenumfangs wurde eine Teststärkenanalyse durchgeführt (Cohen, 1969). Die Teststärke liegt bei allen nicht signifikanten Tätigkeitsmerkmalen unter 0,80. Bis auf Autonomie befindet sich die Teststärke im Bereich zwischen 0,07 und 0,44, für Autonomie beträgt sie 0,69.

Tabelle 24: Ausprägung der Tätigkeitsmerkmale für Personen, die Entwicklungsarbeiten sowohl in Open-Source- als auch in proprietären Softwareprojekten wahrnehmen.

Tätigkeitsmerkmal	**Mittelwert**		**t**
	Open-Source-Entwicklung	Proprietäre Entwicklung	
Komplexität der Tätigkeit			
Vielfalt	4,1	3,9	0,70
Autonomie	4,4	3,7	1,69†
Partizipation	4,1	3,1	2,01*
Aufgabengeschlossenheit	3,9	3,2	1,53†
Rückmeldung durch Tätigkeit	2,9	2,8	0,22
Bedeutsamkeit der Aufgabe	3,6	2,4	2,55*
Qualifikationspotenzial	4,4	3,6	3,13**
Soziale Aspekte			
Rückmeldung durch Personen	3,1	2,9	0,57
Soziale Unterstützung	3,3	3,2	0,54
Interdependenz (Aufgabe)	2,8	3,0	0,25
Interdependenz (Ziele)	3,3	3,9	0,97

† p<0,10; * p<0,05; ** p<0,01 (einseitiger t-Test für gepaarte Stichproben)

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung.

Basis: 8 Entwickler, die sowohl in der Open-Source- als auch in der proprietären Softwareentwicklung tätig sind.

Das Muster, welches sich bei dieser Analyse zeigt, ist mit den Ergebnissen für die Gesamtstichprobe vergleichbar. Die Mittelwerte für die Merkmale der Komplexität sind auch in dieser Substichprobe für die Tätigkeiten der Open-Source-Entwickler größer als die der proprietären Softwareentwickler. Trotz der kleinen Stichprobe sind die Unterschiede für Autonomie, Partizipation, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe und Qualifikationspotenzial signifikant. Mit Ausnahme der Interdependenz der Ziele sind bei den sozialen Aspekten der Tätigkeit keine großen Unterschiede beobachtbar. Die Interdependenz der Ziele ist im Gegensatz zur Gesamtstichprobe für die

proprietäre Softwareentwicklung größer als in der Open-Source-Softwareentwicklung; der Unterschied ist jedoch nicht signifikant.

6.3 Organisationale Kriterien in Abhängigkeit vom Entwicklungsmodell

Neben den Merkmalen der Tätigkeit wurden auch die organisationalen Kriterien auf Unterschiede zwischen Open-Source- und proprietärer Softwareentwicklung untersucht. Parallel zum Vorgehen für die Tätigkeitsmerkmale soll in diesem Kapitel im ersten Schritt überprüft werden, ob Unterschiede in den organisationalen Kriterien durch das Entwicklungsmodell aufgeklärt werden können. Im weiteren Verlauf werden dann die Hypothesen im Zusammenang mit den organisationalen Kritieren geprüft.

Für den Einfluss des Entwicklunsgmodells auf die Tätgkeitsmerkmale wurde parallel zur Analyse der Tätigkeitsmerkmale eine multivariate Varianzanalyse mit dem Faktor "Entwicklungsmodell" durchgeführt. Die Variablen Leistung, Organizational Citizenship Behavior, Identifikation mit dem Teilprojekt/Abteilung und das Flow-Erleben erfüllen nicht die Voraussetzung der Normalverteilung, deshalb wurde die Pillai-Bartlett-Teststatistik (V) genutzt. Der Faktor "Entwicklungsmodell" ist hoch signifikant und hat einen großen Effekt ($V=0{,}21$; $p=0{,}000$; $\eta^2=0{,}21$). Das Entwicklungsmodell hat allerdings nicht auf alle organisationalen Kriterien einen gleich großen Einfluss; signifikant sind nur die Merkmale allgemeine Arbeitszufriedenheit ($p=0{,}000$; $\eta^2=0{,}12$), intrinsische Arbeitsmotivation ($p=0{,}000$; $\eta^2=0{,}09$), die organisationale Identifikation mit dem Projekt bzw. Unternehmen ($p=0{,}003$; $\eta^2=0{,}04$) und das Flow-Erleben ($p=0{,}004$; $\eta^2=0{,}04$). Dabei geben Open-Source-Entwickler positivere Werte für all diese Merkmale an. Einen nicht signifikanten Unterschied findet sich beim Organizational Citizenship Behavior. Dieses Kriterium ist jedoch für Open-Source-Entwickler tendenziell niedriger ausgeprägt (vgl. Tabelle 25).

Tabelle 25: Ausprägung der organisationalen Kriterien für Open-Source- und proprietäre Softwareentwicklung

Organisationale Kriterien	**Mittelwert**		**F**	**η^2**
	Open-Source-Entwicklung	Proprietäre Entwicklung		
Allgemeine Arbeitszufriedenheit	4,1	3,6	26,08 ***	0,12
Intrinsische Arbeitsmotivation	4,2	3,7	20,27 ***	0,09
Leistung (Allgemein)	2,0	1,8	3,39	0,02
Organizational Citizenship	3,7	3,6	0,75	0,00
Organisationale Identifikation (Unternehm.)	4,1	3,7	8,95 **	0,04
Organisationale Identifikation (Abteilung)	4,1	4,1	0,03	0,00
Flow	3,6	3,3	8,44 **	0,04

** p<0,01; *** p<0,001 (einseitiger t-Test für unabhängige Stichproben)

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung; Leistung: vierstufige Skala von 1=hohe Leistung bis 7=geringe Leistung. Die Angaben zu F-Wert und η^2 beziehen sich auf den Faktor "Entwicklungsmodell" in der Varianzanalyse.

Basis: 126 Open-Source- und 73 proprietäre Softwareentwickler

Um auch für die organisationalen Kriterien eine Konfundierung durch interpersonelle Unterschiede in der Wahrnehmung auszuschließen, wurde auch hier die Substichprobe der Softwareentwickler, die sowohl in Open-Source- als auch in proprietären Softwareprojekten tätig sind, durchgeführt. Bei dem t-Test für gepaarte Stichproben erwiesen sich trotz des geringen Stichprobenumfangs 5 Mittelwertsunterschiede mindestens auf einem p<0,10-Niveau als signifikant (vgl. Tabelle 26). Dabei geben die Entwickler für den Open-Source-Bereich eine höhere Arbeitszufriedenheit und eine höhere organisationale Identifikation im Vergleich zu proprietären Softwareprojekten an. Dagegen wird die Leistung im Open-Source-Projekt geringer eingeschätzt als die Leistung während der Tätigkeit als proprietärer Entwickler. Zur Prüfung der Angemessenheit des Stichprobenumfangs wurde auch hier eine Teststärkeanalyse durchgeführt (Cohen, 1969). Die Teststärke für die nicht signifikanten organisationalen Kriterien liegt bei 0,34 bis 0,42 und damit deutlich unter den geforderten 0,80.

Tabelle 26: Ausprägung der organisationalen Kriterien für Personen, die in Open-Source- und proprietärer Softwareentwicklung Entwicklungsarbeit leisten

Organisationale Kriterien	**Mittelwert**		**t**
	Open-Source-Entwicklung	Proprietäre Entwicklung	
Allgemeine Arbeitszufriedenheit	4,3	3,1	2,65*
Intrinsische Arbeitsmotivation	4,0	3,5	1,53†
Leistung	2,6	1,7	2,31*
Organizational Citizenship	3,1	3,8	2,17*
Organisationale Identifikation (Unternehm.)	3,5	3,2	2,38*
Organisationale Identifikation (Abteilung)	3,8	3,6	0,53
Flow	3,8	3,7	0,13

† p<0,10; * p<0,05 (einseitiger t-Test für gepaarte Stichproben)

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung; Leistung: vierstufige Skala von 1=hohe Leistung bis 7=geringe Leistung.

Basis: 8 Entwickler, die sowohl in der Open-Source- als auch in der proprietären Softwareentwicklung tätig sind.

Betrachtet man die gefundenen Unterschiede zwischen der Tätigkeit von Open-Source- und proprietären Entwicklern zeigt sich eine positivere Wahrnehmung der Ergebnisse der Arbeit bei der allgemeinen Arbeitszufriedenheit, der intrinsischen Arbeitsmotivation, der Leistung, der organisationalen Identifikation mit dem Unternehmen/Projekt, dem Flow-Erleben und – zumindest der Tendenz nach – auch für die organisationale Identifikation mit der Abteilung/Teilprojekt. Ein niedrigerer Wert zeigt sich für die Open-Source-Gruppe nur im Organizational-Citizenship-Verhalten. Diese Unterschiede finden sich auch bei Ausschluss der Konfundierung durch interpersonelle Varianz.

Hypothese 3 besagt, dass sowohl bei Open-Source- als auch bei proprietären Software-Entwicklungstätigkeiten sich die untersuchten Merkmale in gleicher Weise auf die organisationalen Kriterien auswirken. Eine Übersicht über die Zusammenhänge zwischen Tätigkeitsmerkmalen und Ergebnisvariablen vermittelt Tabelle 27 (Open-Source-Stichprobe) und Tabelle 28 (proprietäre Stichprobe).

Tabelle 27: Bivariate Korrelationen für Tätigkeitsmerkmale und organisationale Kriterien (Stichprobe Open-Source-Entwickler)

	Vielseitigkeit	Autonomie	Partizipation	Aufgabengeschl.	Rückmeldung (Tätigkeit)	Rückmeldung (Personen)	Soziale Unterstützung	Interdependenz-Aufgabe	Interdependenz-Ziele	Bedeutsamkeit der Aufgabe	Qualifik. potenzial	Allgemeine Arbeitszufr.	Intrinsische Arbeitsmotiv.	Leistung	Organizational Citizenship	Org. Identifikation (Unternehm.)	Org. Identifikation (Abteilung)	Flow
Vielseitigkeit	1,	,435(**)	,330(**)	,180(*)	,345(**)	,13	,104	,177(*)	,201(*)	,293(**)	,217(*)	,197(*)	,391(**)	-,305(**)	,228(**)	,270(**)	,005	,281(**)
Autonomie	,435(**)	1,	,469(**)	,452(**)	,256(**)	,211(*)	,089	-,069	,05	,078	,326(**)	,345(**)	,314(**)	-,143	,106	,213(*)	-,052	,16
Partizipation	,330(**)	,469(**)	1,	,352(**)	,507(**)	,377(**)	,418(**)	,114	,207(*)	,359(**)	,280(**)	,514(**)	,237(**)	,068	,128	,405(**)	,084	,035
Aufgabengeschl.	,180(*)	,452(**)	,352(**)	1,	,410(**)	,148	,059	-,022	,014	,065	,239(**)	,203(*)	,128	,04	,123	,144	-,097	-,007
Rückmeldung (Tätigkeit)	,345(**)	,256(**)	,507(**)	,410(**)	1,	,578(**)	,438(**)	,268(**)	,237(**)	,304(**)	,410(**)	,302(**)	,369(**)	-,041	,244(**)	,470(**)	,088	,180(*)
Rückmeldung (Personen)	,13	,211(*)	,377(**)	,148	,578(**)	1,	,544(**)	,429(**)	,295(**)	,320(**)	,306(**)	,284(**)	,264(**)	,024	,286(**)	,440(**)	,211(*)	,165
Soziale Unterstützung	,104	,089	,418(**)	,059	,438(**)	,544(**)	1,	,445(**)	,324(**)	,297(**)	,308(**)	,413(**)	,364(**)	,033	,343(**)	,483(**)	,208(*)	,069
Interdependenz-Aufgabe	,177(*)	-,069	,114	-,022	,268(**)	,429(**)	,445(**)	1,	,329(**)	,227(**)	,084	-,006	,131	-,052	,310(**)	,250(**)	,177(*)	,148
Interdependenz-Ziele	,201(*)	,05	,207(*)	,014	,237(**)	,295(**)	,324(**)	,329(**)	1,	,250(**)	,219(*)	,181(*)	,373(**)	,07	,424(**)	,361(**)	-,1	,168
Bedeutsamkeit der Aufgabe	,293(**)	,078	,359(**)	,065	,304(**)	,320(**)	,297(**)	,227(**)	,250(**)	1,	,112	,166	,248(**)	-,178(*)	,266(**)	,325(**)	,147	,109
Qualifikationspotenzial	,217(*)	,326(**)	,280(**)	,239(**)	,410(**)	,306(**)	,308(**)	,084	,219(*)	,112	1,	,277(**)	,432(**)	-,03	,12	,351(**)	-,059	,063
Allgemeine Arbeitszufriedenheit	,197(*)	,345(**)	,514(**)	,203(*)	,302(**)	,284(**)	,413(**)	-,006	,181(*)	,166	,277(**)	1,	,210(*)	,052	,075	,396(**)	,268(**)	,086
Intrinsische Arbeitsmotivation	,391(**)	,314(**)	,237(**)	,128	,369(**)	,264(**)	,364(**)	,131	,373(**)	,248(**)	,432(**)	,210(*)	1,	-,201(*)	,426(**)	,488(**)	,11	,179(*)
Leistung	-,305(**)	-,143	,068	,04	-,041	,024	,033	-,052	,07	-,178(*)	-,03	,052	-,201(*)	1,	-,222(**)	-,046	-,065	-,232(**)
Organizational Citizenship	,228(**)	,106	,128	,123	,244(**)	,286(**)	,343(**)	,310(**)	,424(**)	,266(**)	,12	,075	,426(**)	-,222(**)	1,	,369(**)	,019	,105
Org. Identifikation (Unternehm.)	,270(**)	,213(*)	,405(**)	,144	,470(**)	,440(**)	,483(**)	,250(**)	,361(**)	,325(**)	,351(**)	,396(**)	,488(**)	-,046	,369(**)	1,	,331(**)	,202(*)
Org. Identifikation (Abteilung)	,005	-,052	,084	-,097	,088	,211(*)	,208(*)	,177(*)	-,1	,147	-,059	,268(**)	,11	-,065	,019	,331(**)	1,	,018
Flow	,281(**)	,16	,035	-,007	,180(*)	,165	,069	,148	,168	,109	,063	,086	,179(*)	-,232(**)	,105	,202(*)	,018	1,

* 0,05 signifikant (2 seitig)
** 0,10 signifikant (2 seitig)

Tabelle 28: Bivariate Korrelationen für Tätigkeitsmerkmale und organisationale Kriterien (Stichprobe proprietäre Entwickler)

	Vielseitigkeit	Autonomie	Partizipation	Aufgabengeschl.	Rückmeldung (Tätigkeit)	Rückmeldung (Personen)	Soziale Unterstützung	Interdependenz-Aufgabe	Interdependenz-Ziele	Bedeutsamkeit der Aufgabe	Qualifik. potenzial	Allgemeine Arbeitszufr.	Intrinsische Arbeitsmotiv.	Leistung	Organizational Citizenship	Org. Identifikation (Unternehm.)	Org. Identifikation (Abteilung)	Flow
Vielseitigkeit	1,	,553(**)	,458(**)	,146	,581(**)	,316(**)	,321(**)	,16	,119	,290(*)	,404(**)	,381(**)	,492(**)	-,469(**)	,111	,297(*)	,122	,244(*)
Autonomie	,553(**)	1,	,309(**)	,082	,419(**)	,545(**)	,291(*)	,092	,156	,318(**)	,355(**)	,365(**)	,325(**)	-,446(**)	,059	,338(**)	,083	,168
Partizipation	,458(**)	,309(**)	1,	,400(**)	,598(**)	,367(**)	,341(**)	,077	-,03	,233(*)	,354(**)	,351(**)	,507(**)	-,254(*)	,101	,501(**)	-,125	,023
Aufgabengeschl.	,146	,082	,400(**)	1,	,363(**)	,12	,259(*)	,054	,113	,132	,353(**)	,453(**)	,376(**)	-,073	-,014	,225	-,249(*)	,036
Rückmeldung (Tätigkeit)	,581(**)	,419(**)	,598(**)	,363(**)	1,	,327(**)	,365(**)	,187	,069	,397(**)	,406(**)	,433(**)	,650(**)	-,466(**)	,189	,295(*)	-,005	,174
Rückmeldung (Personen)	,316(**)	,545(**)	,367(**)	,12	,327(**)	1,	,345(**)	,082	,126	,15	,497(**)	,375(**)	,370(**)	-,215	,04	,397(**)	,101	,049
Soziale Unterstützung	,321(**)	,291(*)	,341(**)	,259(*)	,365(**)	,345(**)	1,	,098	,059	,086	,551(**)	,463(**)	,385(**)	-,124	,179	,257(*)	-,095	,176
Interdependenz-Aufgabe	,16	,092	,077	,054	,187	,082	,098	1,	-,082	,289(*)	,165	,11	,069	-,024	,141	,026	-,184	-,13
Interdependenz-Ziele	,119	,156	-,03	,113	,069	,126	,059	-,082	1,	,311(**)	,073	,096	,145	,011	,086	,058	,218	,124
Bedeutsamkeit der Aufgabe	,290(*)	,318(**)	,233(*)	,132	,397(**)	,15	,086	,289(*)	,311(**)	1,	,098	,203	,343(**)	-,414(**)	,254(*)	,202	,115	,115
Qualifikationspotenzial	,404(**)	,355(**)	,354(**)	,353(**)	,406(**)	,497(**)	,551(**)	,165	,073	,098	1,	,556(**)	,359(**)	-,081	,14	,249(*)	-,061	,166
Allgemeine Arbeitszufriedenheit	,381(**)	,365(**)	,351(**)	,453(**)	,433(**)	,375(**)	,463(**)	,11	,096	,203	,556(**)	1,	,404(**)	-,129	-,012	,518(**)	,118	,204
Intrinsische Arbeitsmotivation	,492(**)	,325(**)	,507(**)	,376(**)	,650(**)	,370(**)	,385(**)	,069	,145	,343(**)	,359(**)	,404(**)	1,	-,326(**)	,233(*)	,318(**)	,011	,337(**)
Leistung	-,469(**)	-,446(**)	-,254(*)	-,073	-,466(**)	-,215	-,124	-,024	,011	-,414(**)	-,081	-,129	-,326(**)	1,	,	-,117	,08	,004
Organizational Citizenship	,111	,059	,101	-,014	,189	,04	,179	,141	,086	,254(*)	,14	-,012	,233(*)	,	1,	,06	,135	-,038
Org. Identifikation (Unternehm.)	,297(*)	,338(**)	,501(**)	,225	,295(*)	,397(**)	,257(*)	,026	,058	,202	,249(*)	,518(**)	,318(**)	-,117	,06	1,	,295(*)	,057
Org. Identifikation (Abteilung)	,122	,083	-,125	-,249(*)	-,005	,101	-,095	-,184	,218	,115	-,061	,118	,011	,08	,135	,295(*)	1,	,118
Flow	,244(*)	,168	,023	,036	,174	,049	,176	-,13	,124	,115	,166	,204	,337(**)	,004	-,038	,057	,118	1,

* 0,05 signifikant (2 seitig)
** 0,10 signifikant (2 seitig)

Im ersten Schritt wurde für die Unterhypothesen 3a bis 3g jeweils eine lineare Regression mit den Ergebnisvariablen (intrinsische Motivation, allgemeine Arbeitszufriedenheit, Leistung, OCB, Flow-Erleben und organisationale Identifikation) als Kriterium und den Tätigkeitsmerkmalen als Prädiktoren aufgestellt. Dabei wurde jeweils ein Modell für die Gruppe der Open-Source- und ein Modell für die Gruppe der proprietären Softwareentwickler berechnet. Da in das Modell sämtliche Tätigkeitsmerkmale einbezogen werden, konnten nur solche Personen berücksichtigt werden, für die alle Tätigkeitsmerkmale zutreffen. Da insgesamt 17 Open-Source- und 3 proprietäre Softwareentwickler nicht in einem Team bzw. in einer Abteilung arbeiten, konnten diese auch keine Angaben zu Interdependenzen der Ziele bzw. der organisationalen Identifikation mit der Abteilung machen. Dies reduzierte den Stichprobenumfang bei den Open-Source-Entwicklern auf 120 und bei den proprietären Entwicklern auf 70 Personen.

Der Einfluss der Tätigkeitsmerkmale auf die organisationalen Kriterien wurde durch mehrere Regressionsmodelle überprüft. Da in dieser Arbeit neben den häufig genutzten Tätigkeitsmerkmalen des JDS eine Reihe zusätzlicher Merkmale aufgenommen wurde, wird auch untersucht, welche Änderungen sich im aufgeklärten Varianzanteil durch diese zusätzlichen Merkmale ergeben. Zu diesem Zweck wird jeweils eine hierarchische Regression mit den Tätigkeitsmerkmalen des JDS (Vielfalt, Autonomie, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit und Bedeutsamkeit der Aufgabe) im ersten Block und mit sämtlichen erhobenen Tätigkeitsmerkmalen (zusätzlich Partizipation, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz der Aufgabe, Interdependenz der Ziele, und Qualifikationspotenzial) durchgeführt. Die Ergebnisse dieser Analyse für die Open-Source-Stichprobe befinden sich in Tabelle 29, die Details für die Stichprobe der proprietären Softwareentwicklung in Tabelle 30.

Auffällig ist, dass insbesondere für die Open-Source-Stichprobe die zusätzlichen Merkmale einen großen Teil der Varianz aufklären. Dabei sind die Änderungen im aufgeklärten Varianzanteil bei der allgemeinen Arbeitszufriedenheit, der intrinsischen Arbeitsmotivation, dem OCB und der Identifikation mit dem Unternehmen signifikant. Für die Stichprobe der proprietären Entwickler ist der Beitrag der zusätzlichen Merkmale geringer und in keinem Fall signifikant.

Tabelle 29: Hierarchische Regression auf organisationalen Kriterien der Tätigkeit für JCM Tätigkeitsmerkmale und zusätzliche Tätigkeitsmerkmale (Open-Source-Stichprobe)

Organisationale Kriterien	**Durch Modell aufgeklärter Varianzanteil (R^2)**		**Delta aufgeklärter Varianzanteil (R^2)**
	Tätigkeitsmerkmale des JCM	Alle Tätigkeitsmerkmale	
Allgemeine Arbeitszufriedenheit	0,13	0,35	0,22***
Intrinsische Arbeitsmotivation	0,25	0,41	0,15***
Leistung (Allgemein)	0,13	0,19	0,06
Organizational Citizenship	0,12	0,30	0,18***
Organisationale Identifikation (Unternehm.)	0,22	0,34	0,12**
Organisationale Identifikation (Abteilung)	0,02	0,10	0,08
Flow	0,09	0,15	0,07

** p<0,01; *** p<0,001 (Signifikanzniveau der Änderung)

Hierarchische Regressionen mit jeweiligen organisationalen Kriterien als Kriterium. Prädiktoren des JCM sind Vielfalt, Autonomie, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit und Bedeutsamkeit der Aufgabe. Alle Tätigkeitsmerkmale enthalten zusätzlich: Partizipation, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz (Aufgabe), Interdependenz (Ziele) und Qualifikationspotenzial (Methode: Einschluss)

Basis: 120 Open-Source-Entwickler

Tabelle 30: Hierarchische Regression auf organisationalen Kriterien für JCM Tätigkeitsmerkmale und zusätzliche Tätigkeitsmerkmale (Stichprobe der proprietären Softwareentwickler)

Organisationale Kriterien	**Durch Modell aufgeklärter Varianzanteil (R^2)**		**Delta aufgeklärter Varianzanteil (R^2)**
	Tätigkeitsmerkmale des JCM	Alle Tätigkeitsmerkmale	
Allgemeine Arbeitszufriedenheit	0,38	0,49	0,11
Intrinsische Arbeitsmotivation	0,47	0,53	0,06
Leistung (Allgemein)	0,39	0,47	0,07
Organizational Citizenship	0,17	0,21	0,04
Organisationale Identifikation (Unternehm.)	0,21	0,34	0,13
Organisationale Identifikation (Abteilung)	0,08	0,21	0,13
Flow	0,07	0,14	0,07

** p<0,01; *** p<0,001 (Signifikanzniveau der Änderung)

Hierarchische Regressionen mit jeweiligen organisationalen Kriterien als Kriterium. Prädiktoren des JCM sind Vielfalt, Autonomie, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit und Bedeutsamkeit der Aufgabe. Alle Tätigkeitsmerkmale enthalten zusätzlich: Partizipation, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz (Aufgabe), Interdependenz (Ziele) und Qualifikationspotenzial (Methode: Einschluss)

Basis: 70 proprietäre Softwareentwickler

Ein besonders hoher Anteil an aufgeklärter Varianz weist die Gruppe der proprietären Softwareentwickler für die Merkmale allgemeine Arbeitszufriedenheit (49 %), in-

trinsische Arbeitsmotivation (44 %) und Leistung (47 %) auf. Für die Gruppe der Open-Source-Entwickler ist bei allgemeiner Arbeitszufriedenheit (35 %) und bei intrinsischer Arbeitsmotivation (35 %) dieser Anteil etwas geringer, für die Leistung ist er mit 19 % sogar deutlich geringer. Beide Gruppen weisen auch bei der organisationalen Identifikation mit dem Unternehmen mit 34 % einen relativ hohen Anteil an aufgeklärter Varianz auf. Die Prognosekraft der Tätigkeitsmerkmale für OCB, organisationale Identifikation mit der Abteilung und Flow sind deutlich geringer.

Zur Prüfung der Unterhypothesen 3a bis 3g sollen die Regressionsparameter der Open-Source- und der proprietären Stichprobe verglichen werden. Hierfür bietet sich die Anwendung eines Chow-Tests an. Dieses Testverfahren erlaubt den Vergleich eines Satzes von Regressionsparametern von unterschiedlichen Gruppen (Chow, 1960). Ein signifikantes Ergebnis des Chow-Tests besagt, dass bei den untersuchten Gruppen die Parameter nicht identisch sind. Da die Hypothesen 3a bis 3g gleiche Parameter für die beiden Gruppen postuliert, entspricht die Nullhypothese des Chow-Tests der inhaltlichen Hypothese, dass der Satz der Parameter identisch ist. Damit *kein* Einfluss des Faktors Open-Source- vs. proprietäre Softwareentwicklung vorliegt, ist es nötig, den β-Fehler möglichst klein zu halten. Dies wird über die Anpassung des α-Niveaus erreicht (Bortz, 1999). Da eine Annahme der Nullhypothese bei einem $\alpha<0{,}05$ nicht ausreichend konservativ wäre, wird das Signifikanzniveau auf 0,20 liberalisiert und damit die Annahme der Nullhypothese schwieriger gemacht (Sachs, 1999; Wirtz & Nachtigall, 2002).

Der Chow-Test kann nicht direkt durch gängige Statistiksoftware bestimmt werden. Aus diesem Grund wurde die Berechnung mithilfe von Microsoft Excel 2003 vorgenommen. Die für die Berechnung notwendige Quadratsumme der Fehler wurde der SPSS-Regressionsanalyse entnommen. Die verwendete Formel für die Berechnung ist in Abbildung 13 dargestellt.

$$F_{K+1,N_1+N_2-2K-2} = \frac{(SSE_c - SSE_u) * (N_1 + N_2 - 2K - 2)}{SSE_u * (K+1)}$$

Abbildung 13: Formel für F-Statistik des Chow-Tests (Chow, 1960)

Für die untersuchten Regressionsmodelle weist der Chow-Test für die Hypothese 3c (Leistung), Hypothese 3d (OCB), Hypothese 3e (organisationale Identifikation mit dem Unternehmen) und Hypothese 3g (Flow-Erleben) einen p-Wert über dem Signifikanzniveau von 0,20 auf. Bei diesen Werten ist die Nullhypothese anzunehmen, der Satz der Regressionsparameter ist nicht unterschiedlich. Damit können die Hypothesen 3c, 3d, 3e und 3g als bestätigt gelten. Bei der Hypothese 3a (allgemeine Arbeitszufriedenheit),

Hypothse 3b (intrinsische Arbeitsmotivation) und Hypothse 3f (organisationale Identifikation mit der Abteilung) liegt der p-Wert unter dem Signifikanzniveau von 0,20; damit kann nicht von einer strukturellen Gleichheit der Parameter bei der Prognose dieser organisationalen Kriterien ausgegangen werden (vgl. Tabelle 31). Eine Übersicht sämtlicher Regressionsparameter ist in Anhang A (Tabelle 37) aufgeführt.

Tabelle 31: Chow-Test für Modelle zur Prognose der organisationalen Kriterien

Hypothese	Organisationales Kriterium	F	p-Wert
3a	Allgemeine Arbeitszufriedenheit	1,64	0,084
3b	Intrinsische Arbeitsmotivation	2,39	0,007
3c	Leistung (Allgemein)	0,67	0,783[a]
3d	Organizational Citizenship	0,91	0,537[a]
3e	Organisationale Identifikation (Unternehm.)	0,69	0,755[a]
3f	Organisationale Identifikation (Abteilung)	1,38	0,179
3g	Flow	1,12	0,346[a]

[a] p>0,20

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung; Leistung: vierstufige Skala von 1=hohe Leistung bis 7=geringe Leistung.

Basis: 120 Open-Source- und 70 proprietäre Softwareentwickler

Bei den Hypothesen 3a (allgemeine Arbeitszufriedenheit), 3b (intrinsische Arbeitsmotivation) und 3f (organisationale Identifikation mit der Abteilung) konnte keine Gleichheit der Parameter bestätigt werden. Daher soll geprüft werden, welche Unterschiede in der Gewichtung der Tätigkeitsmerkmale zwischen der Open-Source- und der proprietären Stichprobe bei den organisationalen Tätikeitsmerkmalen dieser Kriterien vorliegen. Hierfür wurde eine explorative schrittweise Regression durchgeführt, die Tätigkeitsmerkmale in die Regression aufnimmt, wenn diese zu einer signifikanten Erhöhung (auf dem 5-%-Niveau) des aufgeklärten Varianzanteils beitragen (Methode: Einschluss). Die Ergebnisse dieser Regressionsanalyse zeigen deutliche Unterschiede zwischen Open-Source- und proprietären Softwareentwicklern (vgl. auch Tabelle 32).

Für die allgemeine Arbeitszufriedenheit wurden bei Open-Source-Entwicklern die Merkmale Partizipation (β=0,38) und soziale Unterstützung (β=0,24) aufgenommen, während bei den proprietären Entwicklern die Merkmale Qualifikation (β=0,37), Aufgabengeschlossenheit (β=0,31) und Autonomie (β=0,24) den größten Einfluss auf die allgemeine Arbeitszufriedenheit haben. Für das Modell der intrinsischen Motivation brachte die Qualifikation (β=0,33), die Vielfalt (β=0,26) und die Interdependenz der Ziele (β=0,25) in der Gruppe der Open-Source-Softwareentwickler einen signifikanten Zuwachs. In der Gruppe der proprietären Softwareentwickler wurde die intrinsische Motivation durch die Rückmeldung der Tätigkeit (β=0,57) und die Aufgabengeschlossenheit (β=0,20) am stärksten beeinflusst.

Für das Modell der organisationalen Identifikation mit der Abteilung konnte bei den Open-Source-Entwicklern aufgrund des geringen Anteils aufgeklärter Varianz (10 %) aller Parameter keiner die Schwelle der Signifikanz überschreiten, es wurde somit kein Parameter in das Modell aufgenommen. Für die proprietären Entwickler war zumindest ein Merkmal, die Interdependenz der Aufgabe (β=-0,24), signifikant.

Tabelle 32: Multiple lineare Regression auf organisationale Kriterien mit signifikanten Tätigkeitsmerkmalen

Organisationales Kriterium	**Open-Source-Entwickler**		**Proprietäre Entwickler**	
Arbeitszufriedenheit	Partizipation	(β=0,38)	Qualifikation	(β=0,37)
	Soziale Unterstützung	(β=0,24)	Aufgabengeschlossenheit	(β=0,31)
			Autonomie	(β=0,24)
Intrinsische Motivation	Qualifikation	(β=0,33)	Rückmeldung der Tätigk.	(β=0,57)
	Vielfalt	(β=0,26)	Aufgabengeschlossenheit	(β=0,20)
	Interdependenz der Ziele	(β=0,25)		
Organisationale Identifikation	N/A		N/A	

Die Ergebnisse zeigen, dass Leistung, Organizational-Citizenship-Verhalten, organisationale Identifikation mit dem Unternehmen und Flow sowohl bei Open-Source- als auch bei den proprietären Softwareentwicklern in gleicher Weise mit den Tätigkeitsmerkmalen in Zusammenhang stehen. Unterschiede finden sich bei den Parametern für die übrigen organisationalen Kriterien. Die explorative multiple Regression zeigte dabei kein einheitliches Muster: es gibt Unterschiede zwischen der Open-Source- und der proprietären Stichprobe, aber auch Unterschiede zwischen den Tätigkeitsmerkmalen.

Eine wichtige Referenz für das betrachtete Modell stellt eine weitere Gemeinschaft von Freiwilligen dar, die mittels computervermittelter Kommunikation ein gemeinsames Produkt erstellen: die Wikipedia. Bei dieser wurde von Schroer und Hertel (2009) der Einfluss von Tätigkeitsmerkmalen untersucht. Dabei stellten sich die Tätigkeitsmerkmale als wichtigster Prädiktor der Leistung heraus. Allerdings war der Zusammenhang zwischen Tätigkeitsmerkmalen und Leistung nicht direkt, sondern wurde durch die intrinsische Motivation vollständig mediert. Dieser Arbeit liegt ein Modell zugrunde, das von einem direkten Zusammenhang zwischen Tätikeitsmerkmalen und allen organisationalen Kriterien ausgeht. Ob dieses Modell noch ergänzt werden muss, soll in einer Mediationsanalyse basierend auf dem Modell von Schroer und Hertel (2009) untersucht werden.

Das Modell von Schroer und Hertel (2009) geht dabei von den Tätigkeitsmerkmalen als unabhängige Variable aus. Die betrachteten Tätigkeitsmerkmale sind Autonomie, Bedeutsamkeit der Tätigkeit und Rückmeldung durch die Tätigkeit. Diese gehören alle zu

den Tätigkeitsmerkmalen der Komplexität, welche im Modell von Hackman und Oldham (1974) in einem Zusammenhang zur intrinsischen Motivation stehen. Für die Prüfung des Mediators intrinsische Motivation mit einer Regressionsanalyse soll die Komplexität der Tätigkeit über den Mittelwert aller Tätigkeitsmerkmale der Komplexität (Autonomie, Partizipation, Rückmeldung durch die Tätigkeit, Aufgabengeschlossenheit, Vielfalt, Bedeutsamkeit der Aufgabe und Qualifkationspotenzial) operationalisiert werden. Als Mediator wird die intrinsische Motivation verwendet und die abhängige Variable ist die Leistung. Das komplette Modell und die ß-Gewichte sowohl für die Open-Source Stichprobe als auch für die proprietäre Stichprobe sind in Abbildung 14 zu finden.

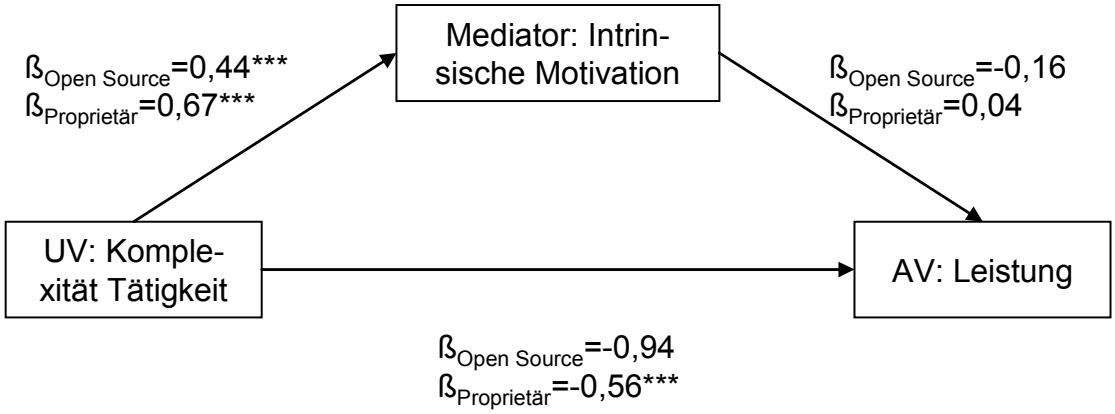

Abbildung 14: Mediatormodell für den Zusammenhang von Komplexität der Tätigkeit und Leistung

Insgesamt ist sowohl der direkte als auch der indirekte Zusammenhang zwischen der Komplexität der Tätigkeit und der Leistung für die Open-Source-Stichprobe sehr gering und nicht signfikant. Für die proprietäre Stichprobe ist zumindest der ß-Koeffizient für den direkten Zusammenhang signifikant, der indirekte Zusammenhang über den Mediator intrinsische Motivation ist nur sehr gering und nicht signifikant. Ein durchgeführter Sobel-Test auf die Mediation von intrinsischer Motivation ist sowohl für die Open-Source- (z=1,633, p=0,10) als auch für die proprietäre Stichprobe (z=0,311, p=0,75) nicht signifikant.

6.4 Einfluss des Moderators Entlohnung

Die Open-Source-Stichprobe besteht zu einer Hälfte (53 %) aus unbezahlten und zur anderen Hälfte aus bezahlten Entwicklern. Mit einer Bezahlung ist meist eine Weisungsbefugnis durch einen Auftraggeber oder Vorgesetzten verbunden, was zentrale Elemente des Open-Source-Entwicklungsmodells wie die Selbstselektion (vgl. Kapitel 3.2.5) ein-

schränken könnte. Viele der Eigenschaften des Open-Source-Modells sind von einer Bezahlung unabhängig, sodass davon ausgegangen werden kann, dass die Bezahlung nur einen untergeordneten Einfluss auf die Gestaltung der Entwicklertätigkeit hat (David et al., 2003; Ghosh et al., 2002; Hars & Ou, 2002; Hertel et al., 2003; Jorgensen, 2001; Lakhani & Wolf, 2003). Um dies zu überprüfen, wurden Open-Source-Entwickler ohne Bezahlung mit solchen verglichen, die einen substanziellen Anteil ihres Einkommens (mehr als 60 %) aus ihren Open-Source-Tätigkeiten beziehen. Insgesamt 68 Open-Source-Entwickler beziehen keine monetäre Belohnung und 26 Personen fallen in die Kategorie einer hohen Bezahlung.

Zur Prüfung des Einflusses der Bezahlung wurden separate Varianzanalysen für die Komplexität der Tätigkeit und die sozialen Aspekte durchgeführt. Beide Varianzanalysen verwendeten "Bezahlung" als Faktor. Bei den Varianzanalysen für den Faktor "Entlohnung" ist die Nullhypothese die inhaltliche Hypothese, dass der Faktor Entlohnung keinen Einfluss hat. Der Faktor "Bezahlung" hat für die Komplexität (U=0,78; p=0,540; η^2=0,05) und die sozialen Aspekte (U=0,83; p=0,536; η^2=0,05) einen p-Wert über der Grenze von 0,20 und damit keinen Einfluss auf die abhängige Variable. Damit kann für die Gesamtheit der Merkmale davon ausgegangen werden, dass der Faktor Bezahlung keinen signifikanten Einfluss auf die Gestaltung der Tätigkeit hat.

Ein detaillierter Vergleich der einzelnen Tätigkeitsmerkmale soll zeigen, ob sich hier einzelne Unterschiede zwischen Open-Source-Entwicklern mit und solchen ohne Bezahlung gibt. Dabei zeigte sich nur für das Merkmal Bedeutsamkeit der Aufgabe ein signifikanter Unterschied. Open-Source-Entwickler mit Bezahlung schätzten die Bedeutsamkeit ihrer Aufgabe geringer ein als solche, die Bezahlung erhielten (vgl. Tabelle 33).

Tabelle 33: Änderung des aufgeklärten Varianzanteils durch die Hinzunahme des Prädiktors Entlohnung

Tätigkeitsmerkmale	Mittelwert		F-Wert	p-Wert	η^2
	Open-Source (Keine Bezahlung)	Open-Source (Hohe Bezahlung			
Komplexität der Tätigkeit					
Vielfalt	4,1	4,3	2,68	0,105	0,03
Autonomie	4,3	4,3	0,36	0,849	0,00
Partizipation	4,0	4,1	0,51	0,478	0,01
Aufgabengeschlossenheit	4,1	4,1	0,00	0,995	0,00
Rückmeldung durch Tätigkeit	3,8	3,9	0,42	0,518	0,01
Bedeutsamkeit der Aufgabe	3,5	4,0	5,86	0,017	0,06
Qualifikationspotenzial	4,4	4,6	1,86	0,176	0,02
Soziale Aspekte					
Rückmeldung durch Personen	3,6	3,7	1,30	0,257	0,02
Soziale Unterstützung	3,7	3,9	3,21	0,077	0,04
Interdependenz (Aufgabe)	3,3	3,7	3,91	0,052	0,05
Interdependenz (Ziele)	3,7	3,8	0,07	0,794	0,00

Separate Varianzanalysen für Komplexität der Tätigkeit und soziale Aspekte. Angabe zu F-Wert, p-Wert und η^2 beziehen sich jeweils auf den Faktor Bezahlung.

Anmerkung: Die Mittelwerte basieren auf einer siebenstufigen Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung.

Basis: 68 Open-Source-Entwickler ohne Bezahlung und 26 Open-Source-Entwickler mit hoher Bezahlung (mehr als 61 % des Einkommens aus Open-Source-Aktivitäten)

Neben dem Einfluss der Bezahlung auf die Tätigkeitsmerkmale soll ebenfalls geprüft werden, ob die Bezahlung in direktem Zusammenhang mit den organisationalen Kriterien steht. Dazu wird für alle Open-Source-Entwickler, die eine Angabe darüber gemacht haben, dass sie für ihre Entwicklungstätigkeit eine Entlohnung erhalten (n=109), eine hierarchische Regression durchgeführt. Dabei wird die aufgeklärte Varianz durch die Tätigkeitsvariablen mit der zusätzlichen Varianz der Dummy-Variable Entlohnung (keine Entlohnung=0; Entlohnung=1) verglichen.

Das Ergebnis dieser Analyse zeigt für die Kriterien allgemeine Arbeitszufriedenheit, intrinsische Motivation, OCB, organisationale Identifikation mit dem Unternehmen bzw. der Abteilung und für das Flow-Erleben keine signifikanten Veränderungen des aufgeklärten Varianzanteils durch die Hinzunahme des Prädiktors Entlohnung (vgl. Tabelle 34). Für das Kriterium Leistung steigt durch das Hinzufügen der Entlohnung der aufgeklärte Varianzanteil signifikant an. Der Koeffizient des Prädiktors ist negativ (β=-0,26), was bedeutet, dass eine Entlohnung mit höherer Leistung einhergeht.

Tabelle 34: Änderung des aufgeklärten Varianzanteils durch die Hinzunahme des Prädiktors Entlohnung

Organisationale Kriterien	Aufgeklärte Var. (R^2)		F-Wert	p-Wert
	Ohne Variable Entlohnung	Mit Variable Entlohnung		
Allgemeine Arbeitszufriedenheit	0,33	0,33	0,32	0,575
Intrinsische Arbeitsmotivation	0,39	0,39	0,17	0,677
Leistung (Allgemein)	0,20	0,26	7,24 **	0,008
Organizational Citizenship	0,30	0,33	3,31	0,072
Organisationale Identifikation (Unternehm.)	0,33	0,34	1,35	0,249
Organisationale Identifikation (Abteilung)	0,12	0,12	0,50	0,482
Flow	0,14	0,15	0,99	0,322

** p<0,01

Multiple lineare Regression mit jeweiligen organisationalen Kriterien als Kriterium und der Vielfalt, Autonomie, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit und Bedeutsamkeit der Aufgabe, Partizipation, Rückmeldung durch Personen, soziale Unterstützung, Interdependenz (Aufgabe), Interdependenz (Ziele), Qualifikationspotenzial und zusätzlichen Prädiktor Entlohnung (Methode: Einschluss)

Basis: 109 Open-Source-Softwareentwickler

Sehr ausführlich wurde in der Literatur der Einfluss von finanziellen Anreizen auf die intrinsische Motivation diskutiert (z. B. Cameron et al., 2001; Deci et al., 1999). Aus diesem Grund soll der Zusammenhang von Entlohnung und intrinsischer Motivation bei Open-Source-Teilnehmern speziell untersucht werden. Dafür wird ein t-Test für die Open-Source-Entwickler durchgeführt. Für die Gruppe der Open-Source-Entwickler, die keine finanzielle Entlohnung für ihre Tätigkeit erhielten, ergab sich eine mittlere intrinsische Motivation von 4,18. Entwickler mit einer finanziellen Entlohnung wiesen einen geringfügig höheren Mittelwert von 4,22 für die intrinsische Motivation auf. Ein t-Test für unabhängige Stichproben (t=-0,374; df=131) ergab mit p=0,71 keinen signifikanten Unterschied zwischen der intrinsischen Motivation der Open-Source-Entwickler mit Bezahlung und der Gruppe der Open-Source-Entwickler ohne Bezahlung.

6.5 Produktivität auf Basis der Anzahl von Programmzeilen

Die Nutzung der Anzahl geschriebener Programmzeilen als Indikator für die Produktivität ist aus der Gruppe der Entwickler nur für die Subgruppe der Programmierer sinnvoll, da deren Tätigkeit hauptsächlich aus dem Schreiben von Programmen besteht. Für diese Analyse wurden diejenigen Entwickler ausgewählt, die als Haupttätigkeit Codieren/Debuggen angegeben haben. Von den insgesamt 64 Programmierern der Open-Source-Stichprobe hatten 19 % keine oder unvollständige Angaben zu der Anzahl erstellter Programmzeilen gemacht, sodass insgesamt 52 Fälle ausgewertet werden

konnten. In der Stichprobe der proprietären Entwickler sind insgesamt 15 mit Programmieren beschäftigt, von denen 9 in die Auswertung einbezogen werden konnten.

Wie auch in anderen Untersuchungen (Dempsey, Weiss, Jones & Greenberg, 1999; Koch & Schneider, 2000; Mockus et al., 2000) weist die Anzahl der erstellten Programmzeilen je Entwickler eine stark linksschiefe Verteilung auf: Wenige Programmierer erstellen einen großen Teil des Quellcodes. Dabei schwankt die Anzahl der erstellten Zeilen pro Monat zwischen 10 und 55.000. Der Median liegt bei 750 Programmzeilen (vgl. Abbildung 15).

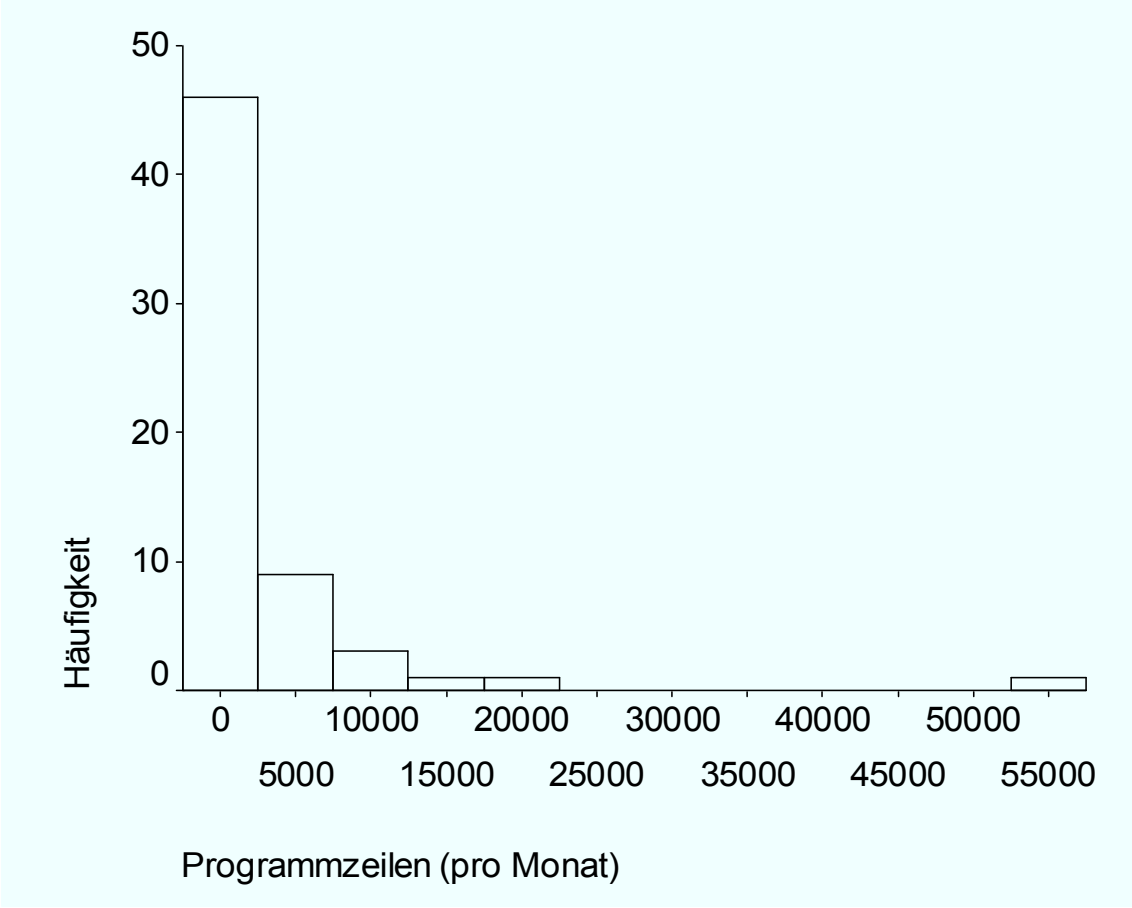

Abbildung 15: Programmzeilen pro Monat für alle Programmierer

Zur Überprüfung der Übereinstimmung zwischen der allgemeinen Selbsteinschätzung der Leistung und der Selbsteinschätzung des erzeugten Codes wurde eine Bivariate Korrelation berechnet. Da ein nicht-linearer Zusammenhang zwischen der Anzahl der Programmzeilen und der Leistung angenommen wird, soll zur Bestimmung ein Rangkorrelationskoeffizient nach Spearman genutzt werden (Bortz, 1999). Der Korrelationskoeffizient der beiden Variablen ist sehr gering (r_{SP}=0,14; N=61). Eine hohe Bewertung der allgemeinen Leistung korreliert zumindest in einem geringen Ausmaß mit einer hohen Anzahl erstellter Programmzeilen pro Monat.

Aufgrund der großen Varianz der mit der Entwicklung verbrachten Zeit bei Open-Source-Entwicklern ist für einen Produktivitätsvergleich auf Basis der Programmzeilen eine Normierung der Anzahl Zeilen pro Zeiteinheit notwendig. Für die Open-Source-Entwickler kann hierfür die Angabe der mit der Open-Source-Entwicklung verbrachten absoluten Zeit verwendet werden. Im Fall der proprietären Entwickler wurde keine Arbeitszeit erfasst, daher muss diese geschätzt werden. Damit die Schätzung möglichst genau werden kann, sollen nur Arbeitnehmer in Vollzeitbeschäftigung berücksichtigt werden. Da die überwiegende Mehrheit der proprietären Entwickler aus Deutschland kommt, kann nach IAT-Report für diese eine durchschnittliche Arbeitszeit von 1.760 Stunden pro Jahr angesetzt werden. Das entspricht 146,7 Stunden pro Monat (Schief, 2004). Bei einer durchschnittlichen Arbeitszeit von 146,7 Stunden errechnet sich für Open-Source-Entwickler ein größerer Durchschnitt an Programmzeilen pro Stunde ($\bar{x}$ =117; $\tilde{x}$ =48; N=51) als für proprietäre Entwickler ($\bar{x}$ =21; $\tilde{x}$ =14; N=9). Aufgrund der fehlenden Normalverteilung für die Stichprobe wurde die Signifikanz dieses Unterschieds mit dem nicht-parametrischen Mann-Whitney-Test bestimmt. Der Test ergab eine auf dem 5-%-Niveau signifikante Differenz (Mann-Whitney-U=120; Z=2,27).

Die bisherigen Vergleiche der Produktivität zwischen der Stichprobe der Open-Source- und der proprietären Entwickler beruhen auf einer Schätzung der relativen Zeit, die Open-Source-Entwickler für ihr Hauptprojekt aufwenden. Fehler in dieser nicht leicht verfügbaren Zahl können zu Über- oder Unterschätzung der Produktivität führen. Das Risiko dieser möglichen Fehlerquelle wird reduziert, wenn nur Angestellte in einer Vollzeitbeschäftigung betrachtet werden. Diese sollten eine weitgehend vergleichbare Arbeitszeit aufweisen. In der vorliegenden Stichprobe haben von den Open-Source-Programmierern in einer Vollzeitanstellung insgesamt 8 Angaben zur Anzahl der Programmzeilen gemacht. Bei den proprietären Softwareentwicklern können 9 Personen in einer Vollzeitbeschäftigung mit Angaben zur Anzahl der Programmzeilen in den Vergleich einbezogen werden. Betrachtet man die Mittelwert für diese beiden Gruppen, so haben Open-Source-Softwareentwickler ($\bar{x}$ =2824; $\tilde{x}$ =1250) eine geringere Anzahl von Programmzeilen pro Monat als die proprietären Entwickler ($\bar{x}$ =3100; $\tilde{x}$ =2000). Aufgrund der geringen Fallzahl wurde keine inferenzstatistische Prüfung dieser Differenz vorgenommen.

Diese bisher dargestellten Ergebnisse ergeben kein homogenes Bild zu Leistung und Produktivität von Open-Source-Entwicklern. Der Vergleich der Gesamtstichprobe zeigt für Open-Source-Entwickler sowohl bei der Anzahl an Zeilen pro Monat als auch bei der Anzahl an Zeilen pro Zeiteinheit einen größere Leistung bzw. Produktivität. Für die kleine Stichprobe der Angestellten in Vollzeitbeschäftigung konnte dieser Unterschied jedoch nicht bestätigt werden.

Die Frage, ob diese Produktivität mithilfe der Tätigkeitsmerkmale erklärt werden kann, soll im folgenden Abschnitt untersucht werden. Da die untersuchten Tätigkeitsmerkmale z. T. unterschiedliche Auswirkungen auf die organisationalen Kriterien von Open-Source- und proprietären Softwareentwicklern aufweisen, können beide Teilstichproben nicht gemeinsam untersucht werden. Die Teilstichprobe der proprietären Entwickler umfasst nur 9 Personen und es wird daher keine Regression durchgeführt. Aufgrund der fehlenden Normalverteilung der Anzahl der Programmzeilen ist die Prüfung, welche Tätigkeitsmerkmale diese am stärksten beeinflussen, mit einer linearen Regression nicht direkt möglich. Anhand der Häufigkeitsverteilung der Variable kann vermutet werden, dass es sich hierbei um eine logarithmische Verteilung handelt. Daher wurde eine logarithmische Transformation (ln[x]) der Anzahl der Programmzeilen durchgeführt. Die so gebildete Variable weist eine gute Annäherung an die Normalverteilung auf (vgl. Abbildung 16). Zur Prüfung der Normalverteilung wird der auch für kleinere Stichproben gut geeignete Kolmogorow-Smirnow-Test verwendet. Für den Vergleich mit einer Normalverteilung ergibt sich für die logarithmische Transformation der Programmzeilen eine Kolmogorov-Smirnow-Z-Statistik von 0,55. Die asymptotische Signifikanz ist größer 5 % und daher kann davon ausgegangen werden, dass die Variable annähernd normalverteilt ist.

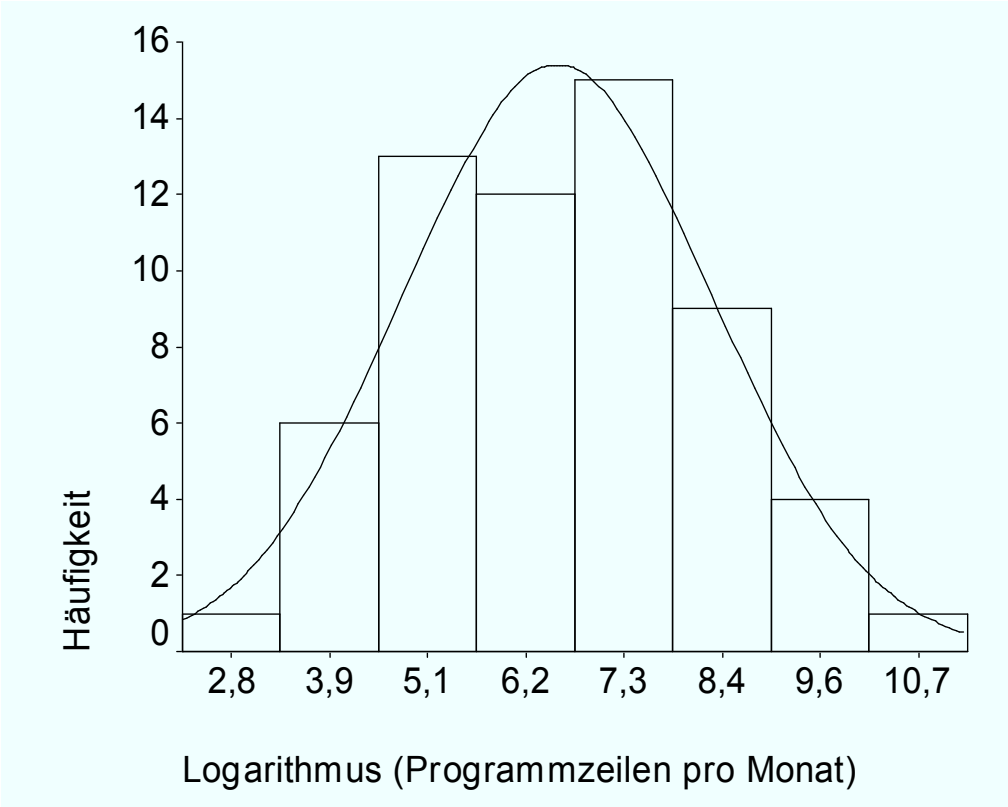

Abbildung 16: Logarithmus von Programmzeilen pro Monat für alle Programmierer

In der anschließenden linearen Regression wurde für die Open-Source-Tätigkeiten der Logarithmus der Programmzeilen durch sämtliche Tätigkeitsmerkmale vorhergesagt. Dabei konnten die Tätigkeitsmerkmale insgesamt 36 % der Gesamtvarianz an der logarithmierten Anzahl der Programmzeilen aufklären. In dem Modell weisen die Merkmale Interdependenz der Aufgabe, die Rückmeldung durch die Tätigkeit und die Autonomie den größten positiven Einfluss auf den Umfang an erstellter Software auf. Ein starker negativer Zusammenhang zeigt sich zwischen den Prädiktoren Aufgabengeschlossenheit und Rückmeldung durch Personen (vgl. Tabelle 35).

Tabelle 35: Multiple Regression von Tätigkeitsmerkmalen auf die log-transformierte Anzahl von Programmzeilen (nur Open-Sourceentwickler)

Tätigkeitsmerkmale	**β**
Komplexität der Tätigkeit	
Vielfalt	-0,06
Autonomie	0,21
Partizipation	0,18
Aufgabengeschlossenheit	-0,23
Rückmeldung durch die Tätigkeit	0,28
Bedeutsamkeit der Aufgabe	-0,07
Qualifikationspotenzial	-0,02
Soziale Aspekte	
Rückmeldung durch Personen	-0,16
Soziale Unterstützung	0,10
Interdependenz (Aufgabe)	0,36
Interdependenz (Ziele)	0,13

Anmerkung: Multiple lineare Regression der Tätigkeitsmerkmale auf Logarithmus der Programmzeilen, R^2=0,36.

N = 44 Open-Source-Softwareentwickler mit der Haupttätigkeit Programmieren

7 Diskussion

Während frühere Untersuchungen zu Open-Source-Projekten ihren Schwerpunkt auf die Motivation der TeilnehmerInnen gelegt haben, zielte diese Arbeit auf die Tätigkeit der Open-Source-Entwicklung. Aufgrund der Neuheit des Untersuchungsgegenstands wurde bewusst einem umfassenden Ansatz Priorität vor der Betrachtung von Details eingeräumt. Das sich hieraus ergebende Bild soll in diesem Kapitel diskutiert werden. Die Auswertung der Ergebnisse hat teilweise beträchtliche Unterschiede zwischen den Entwicklungsmodellen ergeben, die im anschließenden Kapitel 7.1 und 7.2 erörtert werden. Das Kapitel 7.3 beschäftigt sich mit den Unterschieden in den organisationalen Kriterien. Aus vorhergehenden Untersuchungen zur Open-Source-Softwareentwicklung war eine Reihe von potenziellen Moderatoren bekannt; deren Einflüsse werden im Kapitel 7.4 behandelt. Neben den Ergebnissen soll auch das methodische Vorgehen in dieser Arbeit betrachtet werden (Kapitel 7.5). Die beiden letzten Kapitel bewerten die Ergebnisse im Hinblick auf die Bedeutung für die Softwareentwicklung (Kapitel 7.6) und die psychologische Forschung (Kapitel 7.7).

7.1 Tätigkeitsmerkmale der Komplexität

Nach den vorliegenden Daten zu urteilen, kann die Hypothese 1, wonach Open-Source-Tätigkeiten im Vergleich zu Tätigkeiten der proprietären Softwareentwicklung eine höhere Komplexität aufweisen, als bestätigt angesehen werden. Besonders hervorzuheben ist, dass dieser Effekt sich als ausgesprochen stabil erwiesen hat. Durch die Replikation der Varianzanalysen in der Stichprobe der Angestellten konnte gezeigt werden, dass die gefundenen Effekte nicht auf dem unterschiedlichen Anteil Angestellter in beiden Teilstichproben beruht. Zusätzlich konnte die Auswertung der kleinen Stichprobe der Open-Source-Entwickler, die auch die Tätigkeit im Unternehmen bewertet haben, mögliche Konfundierungen durch interpersonelle Varianz ausschließen. Die vorliegenden Daten führen zu dem Schluss, dass die Tätigkeitsmerkmale der Komplexität in Open-Source-Entwicklungsumgebungen höher ausgeprägt sind und damit von Menschen als motivierendere Arbeitsbedingungen wahrgenommen werden (Hackman & Oldham, 1974).

Das Ergebnis einer höheren Komplexität stimmt mit anderen Studien der Freiwilligenarbeit überein (Burkolter, Güntert & Wehner, 2006). In der Untersuchung von Burkolter et al. (2006) waren die Tätigkeitsmerkmale der Freiwilligenarbeit besonders hoch ausgeprägt. Da in dieser Untersuchung ausschließlich die Skalen des JDS zum Einsatz kamen (Anforderungsvielfalt, Aufgabengeschlossenheit, Bedeutsamkeit der Aufgabe, Rückmeldung aus der Tätigkeit), wurde hier de facto die Komplexität der Tätigkeiten erhoben (Morgeson & Campion, 2003). Vergleicht man die Werte der Tätigkeits-

merkmale für die Freiwilligenarbeit, so liegen diese z. B. über den Werten für die Automobilindustrie, der Wissenschaft und der Technologie und sind größer als die von Hackman und Oldham (1980) ermittelten Normen. Allerdings gibt es auch Tätigkeiten im Bereich Sozialwesen und Dienstleistung, die höhere Werte bei der Komplexität aufweisen (Burkolter et al., 2006).

Betrachtet man die einzelnen Tätigkeitsmerkmale, so zeigen Autonomie und Partizipation die größten Effekte. Der große Unterschied zwischen Open-Source- und proprietären Arbeitsumgebungen bei diesen beiden Merkmalen ist nachvollziehbar. Entwickler in Open-Source-Projekten können eigenständig arbeiten und haben in vielen Projekten die Möglichkeit, bei wichtigen Entscheidungen mitzuwirken (Raymond, 1999a). Nach den Erkenntnissen von Schachtner (1994) werden die Gestaltungsspielräume von Softwareentwicklern durch das Fehlen von Beschränkungen bei der Definition von Aufgaben durch das Management und durch den Verzicht auf standardisierte Programmierwerkzeuge gefördert. Ebenfalls förderlich wirkt sich die Souveränität der Entwickler im Hinblick auf die eigene Arbeitszeit, die Transparenz und das eigenständige Handeln aus. Gerade in innovativen Tätigkeitsumfeldern erscheint Handlungsspielraum notwendig und förderlich, um verschiedene Optionen auszuprobieren und damit besser das Wissen und die kognitiven Fähigkeiten der Mitarbeiter zu nutzen. Jedoch gerade in Situationen mit hoher Unsicherheit reagieren Vorgesetzte, wie am Beispiel der Wissensverarbeitung ("knowledge engineering") gezeigt, häufig mit einer Beschränkung der Autonomie ihrer MitarbeiterInnen (Couger, 1986b). Auch in anderen Tätigkeitsfeldern von Freiwilligenarbeit ist die Autonomie ein wichtiges Merkmal. So berichtet ein großer Teil (74 %) der freiwillig Tätigen, keine Einschränkungen ihrer Autonomie in ihrer Tätigkeit zu erleben (Gensicke, Picot & Geiss, 2005; Vondernach, 1980).

Die größeren Wahlmöglichkeiten führen dazu, dass sich Open-Source-Entwickler für Tätigkeiten entscheiden, die zwar nur geringfügig vielfältiger sind, die aber über eine deutlich größere Aufgabengeschlossenheit verfügen. Solche Aufgaben weisen eine größere "Ganzheitlichkeit" (Ulich, 1998) auf. Entwickler in Open-Source-Projekten können sich ein genaueres Bild von dem Einfluss ihrer Arbeit auf das Endprodukt machen. Eine Ursache für die erhöhte Aufgabengeschlossenheit könnte in einem geringeren Maß an Arbeitsteilung liegen, das möglicherweise in Open-Source-Projekten gegeben ist. Damit würde jeder einzelne Entwickler einen größeren Teil des Ganzen bearbeiten. Ein Indiz hierfür wäre die leicht geringere Interdependenz der Aufgaben in Open-Source-Projekten, wobei die Differenz nicht signifikant ist. Eine geringere Interdependenz könnte auch durch eine stärkere Modularisierung der Aufgaben in Open-Source-Projekten im Vergleich zu proprietären Projekten hervorgerufen werden (Hertel, 2007; Moon & Sproull, 2002). Die stärkere Modularisierung erlaubt es dem Entwickler,

innerhalb seines Moduls den gesamten Herstellungsprozess vom technischen Design über die Programmierung bis zum Debuggen und Veröffentlichen zu übernehmen.

Eine weitere Erklärung für eine größere Aufgabengeschlossenheit besteht in der größeren Transparenz des Herstellungsprozesses. Durch mehr Informationen als bei proprietären Tätigkeiten können Open-Source-Entwickler besser den Einfluss und die Relevanz ihrer Tätigkeit für das Gesamtprodukt abschätzen. Das Ausmaß der Transparenz kann man an der Skala Partizipation ablesen, da diese nicht nur die Möglichkeit zur Mitbestimmung, sondern auch die Voraussetzung zur Mitbestimmung, nämlich die Information über Vorgänge in der Organisation beinhaltet. Wie auch bei Autonomie hat für Partizipation der Faktor "Entwicklungsmodell" einen großen Effekt; in Open-Source-Projekten wird die Partizipation von den TeilnehmerInnen deutlich höher bewertet als in der proprietären Softwareentwicklung. Für die Relevanz von Partizipationsmöglichkeiten sprechen auch die Ergebnisse aus einem Laborexperiment, das die Möglichkeit zur Partizipation an der Gestaltung der Arbeitstätigkeit als Einflussfaktor auf die anderen erhobenen Tätigkeitsmerkmale identifiziert (Streker-Seeborg, 1978).

Auch für die Bedeutsamkeit der Aufgabe wurde ein signifikanter Effekt gefunden: Open-Source-Entwickler nehmen die eigene Tätigkeit als bedeutsamer wahr, als dies bei proprietären Entwicklern der Fall ist. In Anbetracht der häufig in Zusammenhang mit Open-Source geführten Debatte über die "Freiheit" von Software und ideologischen "Grabenkämpfen" mit konkurrierenden kommerziellen Softwareangeboten (vgl. Stallman, 2002) erscheint es verwunderlich, dass der vorgefundene Effekt nur klein ist. Dies ist aber konsistent mit Untersuchungen zur Motivation von Open-Source-Entwicklern, die bei der Mehrheit der TeilnehmerInnen nicht ideologische, sondern andere Motive als wichtiger identifizierten. So wurde in vielen Untersuchungen (z. B. Ghosh et al., 2002; Hars & Ou, 2002; z. B. Lakhani et al., 2002; Lakhani & Wolf, 2003; Long, 2003) das Qualifikationspotenzial als eines der wichtigsten oder als wichtigstes Motiv identifiziert. Auch in dieser Arbeit hat von allen Einzelmerkmalen die Zugehörigkeit zu einem Open-Source-Projekt den größten Effekt auf das Qualifikationspotenzial. Open-Source-Entwickler haben den Eindruck, dass sie durch ihre Tätigkeit in Open-Source-Projekten deutlich besser qualifiziert werden, als dies bei Entwicklern in proprietären Projekten der Fall ist. Die Ursache für das niedrigere Qualifikationspotenzial in der proprietären Softwareentwicklung könnte darin liegen, dass viele Unternehmen die Möglichkeiten des gegenseitigen Lernens in Softwareentwicklungsprojekten nicht nutzen und damit die Qualifikationsmöglichkeiten der Mitarbeiter einschränken (Walz et al., 1993). Weiterbildung geschieht häufig in der Form von Seminaren und nicht, wie es das Prinzip der vollständigen Arbeitsaufgabe vorsehen würde, durch Integration der Wissensakquisition in die Gesamtaufgabe des Entwicklers (Schachtner, 1994). Möglicherweise bietet die Situation in Open-Source-Projekten nicht nur mehr

Möglichkeiten zur Qualifikation, sondern erleichtert auch die Annahme von Hilfestellungen. Bei Untersuchungen in IT-Unternehmen wurde Hilfe nur dann geleistet, wenn diese als nützlich für die eigene Karriere wahrgenommen wurde (Perlow & Weeks, 2002). Open-Source-Projekte können als ein Beispiel für die Überwindung von Problemen bei der Wissensweitergabe angesehen werden (Zimmer & Wegener, 2006). Diese Ergebnisse sind ebenfalls konsistent mit den Daten zum Qualifizierungspotenzial der Freiwilligentätigkeit. So geben 44 % Freiwilligen an, dass sie im "hohen" oder im "sehr hohen Maße" Fähigkeiten erwerben (Gensicke et al., 2005).

Insgesamt betrachtet, wurde für die Tätigkeiten in Open-Source-Projekten eine deutlich größere Komplexität gefunden als für den proprietären Softwareentwicklungsbereich. Diese Merkmale haben sich als konsistent herausgestellt und alle durchgeführten Tests von möglichen Konfundierungen haben zu keinen neuen Erklärungsansätzen geführt. Hypothese 1 muss daher als bestätigt betrachtet werden.

7.2 Soziale Aspekte der Tätigkeit

Hypothese 2 zur niedrigeren Ausprägung von sozialen Aspekten der Tätigkeit in Open-Source-Projekten konnte nicht bestätigt werden. Das sich ergebende Bild ist uneinheitlich: bei einigen Merkmalen liegt bei Open-Source-Projekten eine höhere, bei anderen eine niedrigere Ausprägung sozialer Aspekte der Tätigkeitsgestaltung vor. Bis auf das Merkmal "Soziale Unterstützung" ist keiner der Unterschiede signifikant. Diese Ergebnisse würden dafür sprechen, dass die Tätigkeit in Open-Source-Projekten Ähnlichkeiten mit der Tätigkeit von Selbstständigen hat. Zumindest finden sich bei einem Vergleich von Angestellten und Selbstständigen in der IT-Branche die gleichen Muster von Unterschieden. Vergleichbar mit Open-Source-Entwicklern weisen auch Selbstständige einen hohen Grad an Handlungs- und Zeitspielraum auf (und damit eine höhere Komplexität der Tätigkeit), jedoch unterscheiden auch sie sich nicht von den Angestellten in Bezug auf die untersuchten sozialen Aspekte (Gerlmaier, 2002).

Für soziale Unterstützung weist die Tätigkeit in Open-Source-Projekten eine höhere Ausprägung auf. Inwieweit diese höhere Ausprägung auf eine allgemein höhere Qualität der interpersonalen Kontakte hindeutet, ist nicht eindeutig zu sagen. Die einfache Möglichkeit, im Open-Source-Bereich das Projekt zu wechseln, könnte einen positiven Einfluss auf die sozialen Merkmale der Tätigkeit haben. Das Verbleiben in einem Projekt könnte z. B. auch von der erlebten sozialen Unterstützung abhängen. Projekte mit virtuellen Gemeinschaften, die ihre Mitglieder gut unterstützen, könnten mehr Entwickler an sich binden. Projekte, die nur über ein geringes Ausmaß an sozialer Unterstützung verfügen, würden auf eine geringe Größe schrumpfen oder ganz beendet werden. Der Wegfall dieser Opfer der "natürlichen Auslese" könnte dazu führen, dass

der Mittelwert für soziale Unterstützung bei den verbleibenden Projekten steigen würde. Nachfolgende Studien müssten jedoch klären, ob tatsächlich die soziale Unterstützung ein wichtiges Merkmal für den Verbleib in Projekten darstellt. Dass soziale Unterstüzung in Open-Source-Projekten höher ist als in Projekten der proprietären Softwareentwicklung ist auch im Hinblick auf die Theorie der computervermittelten Kommunikation (vgl. Kapitel 3.2.2) ein interessantes Ergebnis. Die größere soziale Unterstützung wird nämlich erreicht, obwohl in Open-Source-Projekten hauptsächlich computervermittelte Komunikation eingesetzt wird, während in proprietären Projekten die am häufigsten genutzte Kommunikationsform die Face-to-Face-Kommunikation ist. Auch wenn die hier beobachteten Unterschiede nicht verschiedene Ursachen haben können, so bestätigen die Ergebnisse, dass sich computervermittelte Kommunikation und soziale Unterstützung nicht ausschließen.

Dass sich keine deutlichen Unterschiede zwischen der Aufgabeninterdependenz in Open-Source- und proprietären Projekten zeigen, lässt die Frage aufkommen, inwieweit tatsächlich für die Tätigkeit relevante Unterschiede in der Organisation von Kooperation vorliegen. Ein häufig zitierter Einflussfaktor auf die Kooperation in der Softwareentwicklung ist die Modularität. Diese soll in Open-Source-Projekten deutlich höher liegen als in der proprietären Softwareentwicklung (Raymond, 1999a). Diese Aussage wird zwar von Einzelfallstudien an Vorzeigeprojekten gestützt (MacCormack et al., 2005), jedoch zeigen Untersuchungen, die mehrere Projekte einbeziehen, dass eine größere Modularität nicht als gesichert angesehen werden kann (Paulson et al., 2004). Wenn kein Unterschied in der Modularität vorliegt, ist vermutlich auch der gleiche Aufwand an Koordination und Kooperation notwendig.

Betrachtet man die möglichen Ursachen für die geringere Ausprägung sozialer Aspekte der Tätigkeit in Open-Source-Projekten, erweisen sich vor allem weniger Kontaktmöglichkeiten und eine geringere Intensität der Teamarbeit als Korrelate. Im Gegensatz zu dem verbreiteten Bild von einer sehr hohen Anzahl beteiligter Entwickler in Open-Source-Projekten fanden sich in dieser Stichprobe nur wenige Projekte, auf die diese Charakterisierung zutrifft. Die Mehrheit der untersuchten Open-Source-Projekte bestand nur aus wenigen Entwicklern. Dieses Ergebnis scheint nicht an einer außergewöhnlichen Zusammenstellung der Stichprobe zu liegen, sondern den tatsächlichen Gegebenheiten in Open-Source-Projekten zu entsprechen. So findet Görling (2003) bei seiner Auswertung der größten Plattform für Open-Source-Projekte, Sourceforge, dass ca. 60 % aller registrierten Projekte nur aus einem Entwickler bestehen und mehr als 75 % der Projekte aus weniger als 3 Entwicklern bestehen. Auch die Anzahl der Personen, die im Team kooperieren, war in Open-Source-Projekten kleiner als in den untersuchten proprietären Teams. Durch Beobachtungen von Holck und Jorgensen (2004) wird bestätigt, dass die meiste Arbeit in der Entwicklung von FreeBSD und Mozilla (zwei sehr großen Projekten

mit vielen Hundert Entwicklern) in Ein-Mann-Projekten stattfindet. Die geringere Größe der Projekte und der Teams könnte eine Ursache für die geringere Ausprägung der sozialen Aspekte der Tätigkeit sein. Bei Open-Source-Projekten werden die sozialen Tätigkeitsmerkmale besonders durch die Größe des Gesamtprojektes beeinflusst, während in der proprietären Entwicklung die Personen im Team wichtiger sind. Dies könnte ein Hinweis darauf sein, dass Teamarbeit in Open-Source-Projekten eine geringere Bedeutung hat. Die hochsignifikanten Zusammenhänge in Open-Source-Projekten zwischen einer hohen Intensität der Teamarbeit und hohen Ausprägungen sozialer Aspekte der Tätigkeit unterstützen diese Vermutung.

Ein Zusammenhang zwischen der räumlichen Distanz der Beteiligten und den sozialen Aspekten konnte für diese Stichprobe nicht bestätigt werden. Dabei sind die Unterschiede in der räumlichen Distanz zwischen Open-Source-Entwicklern, die einen großen Teil der Arbeit allein erledigen, und den proprietären Entwicklern, die zumeist über Kontaktmöglichkeiten an ihrem Arbeitsplatz verfügen, groß. Eine mögliche Erklärung für das Ausbleiben von negativen Auswirkungen könnte die hohe technische Kompetenz der erhobenen Softwareentwickler sein. Sie könnten in der Lage gewesen sein, durch den Einsatz von elektronischen Kommunikationsmedien die nachteilige Auswirkung der räumlichen Distanz zu kompensieren, was den Prognosen der Theorie der sozialen Informationsverarbeitung entsprechen würde (Walther, 1992; Walther, 1996). Unterstützt wird diese These dadurch, dass auch das eingesetzte Medium keine signifikanten Zusammenhänge zu den sozialen Aspekten der Tätigkeit aufweist.

Die weniger ausgeprägten sozialen Aspekte der Tätigkeit könnten auch darauf zurückzuführen sein, dass in Open-Source-Projekten ein geringerer Kommunikationsbedarf besteht. Ein Grund hierfür könnte ein geringerer Standardisierungsgrad und die Verwendung weniger fortgeschrittener Entwicklungswerkzeuge sein. Diese Aussage scheint der gängigen Vermutung zu widersprechen, dass in der Softwareentwicklung die Standardisierung von Routineaufgaben (z. B. durch Spezifikationen, Prozesse und Werkzeuge) zu einem geringeren Kommunikationsbedarf führt (Grinter, Herbsleb & D., 1999; Herbsleb & Grinter, 1999; Herbsleb, Mockus, Finholt & Grinter, 2000). Andere Untersuchungen an Softwareentwicklungsprojekten haben bei hohem Standardisierungsgrad und umfangreicher Unterstützung durch fortschrittliche Werkzeuge aber eine Steigerung der Kommunikationsanforderungen nachgewiesen (Brodbeck & Frese, 1994). Eine eindeutige Aussage ist auf Basis der vorliegenden Ergebnisse damit nicht zu treffen.

Zusammenfassend muss zu den sozialen Aspekten der Tätigkeit festgestellt werden, dass die in Hypothese 2 postulierten niedrigen Ausprägungen von sozialen Aspekten bei Open-Source-Tätigkeiten nicht gefunden werden konnten. Hypothese 2 muss damit verworfen werden. Die teilweise großen Unterschiede in Bezug auf den Arbeitsort und die Nutzung von Kommunikationsmitteln zwischen der Stichprobe der Open-Source-

Entwickler und den proprietären Entwicklern führen zu zwei unterschiedlichen Erklärungsmöglichkeiten für die geringe Differenz bei den sozialen Aspekten der Tätigkeit. Eine mögliche Erklärung ist, dass für die vorliegende technisch affine und kompetente Stichprobe die Unterschiede in den Kommunikationsvoraussetzungen nur eine untergeordnete Rolle spielen. Dies würde aber im Widerspruch zu einigen verbreiteten Theorien der computervermittelten Kommunikation stehen (vgl. Kap. 3.2.2) und sich nur durch spezifische Ausprägungen von Moderatoren in der untersuchten Stichprobe erklären lassen. Eine alternative Erklärung würde darin liegen, dass sich die Wirkung von unterschiedlichen Einflussfaktoren aufhebt. So könnten sich die große Distanz, die geringere Anzahl an Face-to-Face-Treffen und der Einsatz von wenig reichhaltigen Kommunikationsmitteln negativ auf die sozialen Aspekte in Open-Source-Entwicklungsgemeinschaften auswirken. Zur selben Zeit könnte die wahrgenommene soziale Unterstützung in Open-Source-Projekten höher als in proprietären Projekten sein, da diese Projekte vielleicht ein Zusammentreffen von Gleichgesinnten ermöglichen und nicht durch Konkurrenzsituation und Leistungsdruck an effektiver gegenseitiger Unterstützung gehindert werden.

Das Fehlen von Unterschieden bei den sozialen Aspekten ist auch ein weiteres Indiz dafür, dass die signifikanten Differenzen bei den Tätigkeitsmerkmalen der Komplexität nicht auf einer generellen positiveren Wahrnehmung von Open-Source-Projekten basieren. Die TeilnehmerInnen an der Untersuchung scheinen differenziert die Tätigkeitsmerkmale bewertet zu haben: ein "Halo-Effekt" kann nicht identifiziert werden.

7.3 Organisationale Kriterien und der Zusammenhang zu Tätigkeitsmerkmalen

In diesem Abschnitt sollen die Ergebnisse zur Hypothese 3 und die organisationalen Kriterien im Allgemeinen diskutiert werden. Hypothese 3 postulierte, dass der Zusammenhang zwischen Tätigkeitsmerkmalen und den untersuchten organisationalen Kriterien für Open-Source-Tätigkeiten und Tätigkeiten in der proprietären Softwareentwicklung identisch ist. Diese Hypothese unterteilt sich in insgesamt sieben Unterhypothesen für die jeweiligen organisationalen Kriterien. Die Gleichheit der Parameter konnte dabei für die Zusammenhänge zwischen den Tätigkeitsmerkmalen und den organisationalen Kriterien Leistung (Hypothese 3c), Organizational Citizenship Verhalten (Hypothese 3d), organisationale Identifikation mit dem Unternehmen (Hypothese 3e) und für Flow (Hypothese 3g) bestätigt werden. Keine Bestätigungen fand die Gleichheit der Zusammenhänge für die Kriterien allgemeine Arbeitszufriedenheit (Hypothese 3a), intrinsische Arbeitsmotivation (Hypothese 3b) und organisationale Identifikation mit der Abteilung (Hypothese 3f). Für die nicht bestätigten Hypothesen konnten auf Basis

einer explorativen Regressionsanalyse die Unterschiede zwischen der Open-Source- und der proprietären Stichprobe identifiziert werden. So haben TeilnehmerInnen in Open-Source-Projekten eine höhere Arbeitszufriedenheit, wenn sie die Möglichkeit haben, an Entscheidungen, die das ganze Projekt betreffen, mitzuwirken und dies in einem angenehmen sozialen Kontext möglich ist. Für proprietäre Softwareentwickler sind es dagegen vor allem Merkmale der Komplexität der Tätigkeit, die zur Zufriedenheit beitragen. So sollte die eigene Aufgabe in sich abgeschlossen sein, die Möglichkeit bieten, im Rahmen der eigenen Tätigkeit zu entscheiden und sich positiv auf die eigene Qualifikation auswirken. Für die intrinsische Motivation sind in der Open-Source-Entwicklung Merkmale der eigenen Tätigkeit wie das Qualifikationspotenzial und die Vielfalt besonders wichtig. Ebenfalls wichtig ist das Verfolgen gemeinsamer Ziele (Interdependenz der Ziele). Bei der Stichprobe der proprietären Softwareentwickler sind es wiederum Merkmale der Komplexität (Rückmeldung durch die Tätigkeit und Aufgabengeschlossenheit), die im besonders engen Zusammenhang mit der intrinsischen Motivation stehen.

Wie in anderen Untersuchungen zum Einfluss von Tätigkeitsmerkmalen gezeigt, sind diese in der Lage, Varianzanteile bei einstellungsbasierten organisationalen Kriterien gut aufzuklären, bei verhaltensbasierten Kriterien wie der Leistung gelingt dies in der Regel weniger zufriedenstellend (siehe z. B. Fried & Ferris, 1987). Es ist daher überraschend, dass für die Stichprobe der proprietären Entwickler der Anteil aufgeklärter Varianz für die Leistung mit 47 % vergleichbar hoch ist wie für Arbeitszufriedenheit und intrinsische Motivation. Eine mögliche Erklärung für die starken Zusammenhänge könnte die Erhebung der Leistung in Form eines Selbstberichts sein. Für die Open-Source-Softwareentwicklung ist der Anteil aufgeklärter Varianz mit 19 % deutlich niedriger. Auch für das objektivere Maß (das jedoch auch auf einen Selbstbericht beruhte) der Leistung über die Anzahl der Programmzeilen kann für die Open-Source-Stichprobe mit 36 % ein hoher Anteil an Varianz durch die Tätigkeitsmerkmale aufgeklärt werden. Insgesamt kann mit diesen Ergebnissen eine gute Prognose der organisationalen Kriterien auch für die Open-Source-Stichprobe festgestellt werden, allerdings sind die Zusammenhänge nicht so eng, wie das in anderen Stichproben zu freiwilligen Tätigkeiten gefunden wurde (Güntert & Wehner, 2005; Wehner & Güntert, 2005).

Vergleicht man für die Open-Source- und die proprietäre Stichprobe den Anteil an Varianz, den die Tätigkeitsmerkmale aufklären, so ergibt sich ein größerer Anteil aufgeklärter Varianz für proprietäre Projekte. Über die Ursachen kann bisher nur spekuliert werden. Zum einen ist es möglich, dass für die Open-Source-Softwareentwicklung wichtige Tätigkeitsmerkmale nicht erfasst wurden. Zum anderen ist es möglich, dass in der Open-Source-Softwareentwicklung einige Tätigkeitsmerkmale eine zu hohe Ausprägung haben: Obwohl im Allgemeinen davon ausgegangen wird, dass die Tätigkeits-

merkmale in einem positiven linearen Zusammenhang zu den organisationalen Kriterien stehen (zusammenfassend DeJonge & Schaufeli, 1998), deutet eine Reihe von Untersuchungen an, dass die Beziehung komplexer ist. So finden Kluger und DeNisi (1996) z. B. in mehr als einem Drittel (38 %) aller Studien als Folge von Rückmeldung einen Rückgang der Leistung. Der Zusammenhang zwischen Handlungsspielraum und Stress soll am besten durch eine umgekehrte U-Funktion abgebildet werden (Xie & Johns, 1995), womit nach dem Überschreiten des optimalen Ausmaßes an Handlungsspielraum der Stress nicht weiter sinken, sondern wieder steigen würde. Auch das Vitamin-Modell von Warr (1990) postuliert einen nicht-linearen Zusammenhang zwischen dem Tätigkeitsspielraum und verschiedenen organisationalen Kriterien. Zudem existieren Moderatoren, die einen Einfluss auf die Beziehung haben. So ist der Zusammenhang zwischen Autonomie und Zufriedenheit nur für solche Personen positiv, die schon länger (mehr als 3 Monate) in dem Unternehmen angestellt sind (Katz, 1978).

Neben der Analyse der Zusammenhänge ist auch ein Vergleich der Höhe der Ausprägung der organisationalen Kriterien zwischen der Open-Source- und der proprietären Stichprobe interessant. So weisen Open-Source-Projekte im Vergleich zu der proprietären Stichprobe eine höhere Zufriedenheit mit der Arbeit, eine größere intrinsische Arbeitsmotivation, eine stärkere Identifikation mit dem Open-Source-Projekt/Unternehmen und ein stärker ausgeprägtes Flow-Erleben auf. Für die Identifikation mit der Abteilung, das OCB und die Leistung zeigen sich keine signifikanten Unterschiede. Allerdings ist eine nicht-signifikante Tendenz zu höherer Leistung und stärker ausgeprägtem OCB bei der proprietären Entwicklung erkennbar. Insgesamt kann man damit feststellen, dass die TeilnehmerInnen der Studie die organisationalen Kriterien in Open-Source-Projekte positiver beurteilten als die TeilnehmerInnen aus proprietären Tätigkeiten. Da ein Großteil der TeilnehmerInnen in Open-Source-Projekten nicht durch wirtschaftliche Erwägungen an ein Projekt gebunden ist, ist es wenig überraschend, dass solche Projekte besucht werden, die zu einer hohen allgemeinen Arbeitszufriedenheit führen und die intrinsisch motivierend sind. Da das Verlassen der Projekte mit geringen Kosten verbunden ist, könnte in Open-Source-Projekten eine deutlich höhere Fluktuation vorherrschen. Diese wurde jedoch in dieser Untersuchung nicht erhoben.

Auch wenn für das OCB keine signifikanten Unterschiede vorliegen, ist eine Tendenz zu höherem OCB in der proprietären Softwareentwicklung zu erkennen. Dies widerspricht der Ansicht von Aktivitäten in Open-Source-Gemeinschaften als OCB per se. So fördert eine autonome Arbeitsgestaltung die Bildung eines pro-aktiven Rollenverhaltens und führt weg von der "that’s not my job"-Perspektive (Parker, Wall & Jackson, 1997). Hoher Druck und enge Zeitrahmen, wie sie sicherlich stärker in der proprietären Softwareentwicklung anzutreffen sind, sollen gerade in der IT-Branche zu einer Überlastung führen, die OCB verhindert (Moore & Love, 2005). Ein tendenziell geringer ausgeprägtes

altruistisches Verhalten im Open-Source-Bereich erscheint in einer Gemeinschaft, die ihre Arbeitskraft kostenlos zur Verfügung stellt, um Software für die Allgemeinheit zu erstellen, auf den ersten Blick ebenfalls kontraintuitiv. Untersuchungen legen nahe, dass altruistische Motive nicht die Bedeutung für Open-Source-TeilnehmerInnen haben, die zuvor häufig unterstellt wurde (z. B. Ghosh et al., 2002; Hars & Ou, 2002; Hertel et al., 2003; z. B. Lakhani et al., 2002; Lakhani & Wolf, 2003; Long, 2003). In dieser Studie haben die Tätigkeitsmerkmale für die Open-Source-Stichprobe zumindest mit 31 % zur Aufklärung des OCB-Varianzanteils beigetragen, während sie in der proprietären Softwareentwicklung mit 10 % einen deutlich geringeren Beitrag leisten.

Im Gegensatz zur Untersuchung von Hertel et al. (2003) konnten in der vorliegenden Studie für Open-Source-Entwickler keine Unterschiede zwischen der Identifikation mit dem Gesamtprojekt und mit dem Teilprojekt gefunden werden. Eine mögliche Erklärung besteht in den unterschiedlichen Zusammensetzungen der Stichproben. Während die Stichprobe von Hertel et al. ausschließlich aus Entwicklern eines großen Projekts (Linux Kernel) bestand, sind in der vorliegenden Untersuchung auch viele Projekte mit geringer Größe enthalten. Bei Projekten mit nur wenigen TeilnehmerInnen können Teilprojekte z. B. aus nur einer Person bestehen. Solche Teilprojekte sind als Objekt für eine Identifikation wenig geeignet. Dies macht eine Interpretation der nicht vorgefundenen Unterschiede zur organisationalen Identifikation mit der Abteilung in der proprietären Softwareentwicklung nicht sinnvoll.

Ein Vergleich der organisationalen Identifikation auf Projekt- bzw. Unternehmensebene ergibt eine signifikant größere Identifikation der Open-Source-TeilnehmerInnen gegenüber den proprietären Softwareentwicklern. In Anbetracht des höheren Grades an Virtualität von Open-Source-Projekten ist diese auch für die Koordination notwendig. So ist bei virtuellen Tätigkeiten organisationale Identifikation für die Leistung von größerer Bedeutung als in traditionellen Organisationsformen (Wiesenfeld et al., 1999, 2001). Die Virtualität ist möglicherweise auch eine Erklärung für die größere organisationale Identifikation. So könnte bei isolierten Gruppenteilnehmern das Zugehörigkeitsgefühl ausgeprägter sein, weil dadurch das idealisierte Bild der Gruppe nicht zerstört wird (Spears, Lea & Lee, 1990). Für beide Stichproben besteht ein enger Zusammenhang zwischen den Tätigkeitsmerkmalen und der organisationalen Identifikation, wie ein aufgeklärter Varianzanteil von 34 % für beide Stichproben zeigt.

Dass die erhobenen Tätigkeitsmerkmale nur bedingt in der Lage sind, Flow-Erleben von Softwareentwicklung zu erklären, zeigt die mit 15 % bzw. 14 % relativ geringe aufgeklärte Varianz für Open-Source- bzw. proprietäre Softwareentwickler. Der in anderen Studien gefundene Einflussfaktor "Herausforderung" (Rau & Riedel, 2004) konnte nur zum Teil bestätigt werden, da die in der Komplexität der Aufgabe enthaltenen Merkmale Aufgabengeschlossenheit und Partizipation tendenziell zu einem geringen Ausmaß an

Flow-Erleben führen. Die positive Wirkung von Handlungsspielraum (z. B. Gail & Frese, 1994; z. B. Ghani & Desphande, 1994) wurde durch das positive β-Gewicht bei Autonomie bestätigt. Beim Vergleich aller Softwareentwickler haben die Open-Source-Entwickler signifikant häufigere und intensivere Flow-Erlebnisse. Für die kleine Stichprobe der Entwickler, die in beiden Bereichen tätig sind, lässt sich dieser Unterschied zumindest tendenziell bestätigen.

Häufig wird die geringere Benutzerfreundlichkeit (Usability) von Open-Source-Software beklagt (Eklund, Feldman, Trombley & Sinha, 2002; Lerner & Tirole, 2002b; Nichols & Twidale, 2003). Einen Grund sehen Hars und Ou (2000) darin, dass Open-Source-Software nicht an den Bedürfnissen der Nutzer, sondern der Entwickler ausgerichtet ist. In Projekten, bei denen Nutzer und Entwickler identisch sind, führt dies jedoch zu keinen Konflikten. Da immer mehr Open-Source-Projekte eine große Verbreitung finden und auch von weniger technisch versierten Personen genutzt werden, könnte dies zu einem Nachteil von Open-Source-Software werden. Eine Orientierung an den Bedürfnissen der Nutzer würde jedoch eine extrinsische Anforderung darstellen, die möglicherweise nicht mit den intrinsischen Bedürfnissen der TeilnehmerInnen übereinstimmt und damit keine Beachtung findet. Die Bedeutung der intrinsischen Motivation lässt sich an der signifikanten Korrelation mit der Anzahl der erstellten Programmzeilen ablesen. Die Ergebnisse anderer Untersuchungen (Bosco, 2004; Hars & Ou, 2000; Lakhani & Wolf, 2003) bestätigen, dass intrinsische Motivation ein wichtiger Grund für die Teilnahme an Open-Source-Projekten ist. Möglicherweise müsste das in dieser Untersuchung genutzte einfache Modell zum Zusammenhang zwischen Tätigkeitsmerkmalen und organisationalen Kriterien erweitert werden. Die Überprüfung des Modells von Schroer und Hertel (2009) bzgl. der Mediation von intrinsischer Motivation konnte zwar nicht bestätigt werden, jedoch könnte eine vollständige Einbeziehung der Varianz der einzelnen Tätigkeitsmerkmale in ein pfadanalytisches Modell hier eine bessere Aufklärung bieten. Voraussetzung für die pfadanalytische Auswertung wäre jedoch bei der gegebenen Anzahl an Variablen ein größerer Stichprobenumfang. Auch für die Qualifikation ist die Frage nach einem komplexeren Modell gerechtfertigt (siehe Kapitel 4.3.6). So ist es nicht eindeutig, ob Qualifikation bzw. Qualifikationspotenzial als Tätigkeitsmerkmal oder als ein organisationales Kriterium zu betrachten ist.

Die Bestimmung von Leistung und Produktivität ist ein wichtiges Thema der Softwareentwicklung. So wendet z. B. das IT-Unternehmen IBM in den 1980er Jahren 5 % seiner Entwicklungskosten für die Messung und Analyse des Entwicklungsprozesses auf (Jones, 1986). Auch aktuell sollte Leistung noch einen hohen Stellenwert in der Softwareentwicklung haben. Die Leistung in der Open-Source-Entwicklung wird in der Literatur sehr unterschiedlich bewertet. Autoren, die keine Kosten für die Arbeitskraft in Open-Source-Projekten in Betracht ziehen, kommen zu der Auffassung, dass "Developing open

source software is faster, better and cheaper" (Potdar & Chang, 2004), während in der proprietären Softwareentwicklung nicht alle Anforderungen gleichzeitig erfüllt werden können. Andere Autoren vermuten, dass das Anwerben einer Open-Source-Gemeinschaft zu weniger Fehlern und größerer Produktivität führt (z. B. Mockus et al., 2002). Im Gegensatz dazu betrachtet z. B. McConnell (1999) die Open-Source-Entwicklung als schnell und effektiv, jedoch sieht er einen Mangel an Effizienz.

In dieser Untersuchung wurde die Anzahl individuell produzierter Programmzeilen von den Entwicklern selbst eingeschätzt. Vergleicht man diese Zahlen mit Untersuchungen, die ein Auszählen der Programmzeilen vorgenommen haben, zeigt sich ein beträchtlicher Unterschied. Die besten Entwickler des Apache-Projekts erstellen 4.300 Zeilen pro Person und Jahr (Mockus et al., 2000, 2002). Dies entspricht dem Durchschnitt von 10 bis 20 Zeilen pro Tag und Person, der auch in vielen Studien zur proprietären Softwareentwicklung gefunden wurde (MacCormack et al., 2005). In der vorliegenden Untersuchung liegen sowohl proprietäre als auch Open-Source-Softwareentwicklung deutlich über diesem Wert. Neben der Frage nach der Validität von Programmzeilen als Produktivitätskriterium ergibt sich hieraus auch die Frage nach der Reliabilität der Erhebung der Programmzeilen durch Selbstauskunft.

Für einen tatsächlichen Vergleich der Effizienz von Open-Source- und proprietärer Entwicklung müsste es gelingen, den Gesamtaufwand von Open-Source-Softwareentwicklung zu quantifizieren. Dies würde z. B. auch den Aufwand für Doppelentwicklung einschließen, die durch zeitliche Überschneidung oder durch Alternativvorschläge zustande gekommen ist. Viele Beiträge werden und können nicht zentral erfasst werden, weil z. B. Entwickler ihre Beiträge eigenständig verwerfen und die Gemeinschaft nie etwas von diesen Tätigkeiten erfährt. Der Vergleich der Produktivität ist eine Aufgabe, die diese Arbeit nicht erfüllen kann. Die vorgenommenen Selbsteinschätzungen ihrer eigenen Produktivität und der erstellten Programmzeilen können nur ein erster Hinweis auf Produktivitätsunterschiede sein, da noch keine anderen Zahlen oder Methoden der Bestimmung vorliegen. Zu beachten ist, dass in dieser Untersuchung die Anzahl der Programmzeilen vorwiegend zur Validierung der anderen organisationalen Kriterien diente. Die vergleichsweise geringe Korrelation zwischen der selbst eingeschätzten Leistung und der selbst eingeschätzten Anzahl der Programmzeilen gestattet jedoch nur eine vorsichtige Interpretation.

Die vorliegenden Zahlen zeigen, dass die Leistung in Open-Source-Projekten niedriger eingeschätzt wird als in der proprietären Softwareentwicklung. Der Unterschied zeigt sich tendenziell bei der allgemeinen Einschätzung der Leistung der Gesamtstichprobe und ist für die gut vergleichbaren Daten der gepaarten Stichprobe von Entwicklern, die im Open-Source- und proprietären Bereich arbeiten, besonders hoch und signifikant. Für die Anzahl der Programmzeilen, die pro Stunde erstellt werden, sind die Ergebnisse ge-

mischt. Für die Gesamtstichprobe zeigt sich ein Vorsprung für die Open-Source-Entwickler; bei der gepaarten Stichprobe der Entwickler, die in beiden Bereichen tätig sind, ist die Anzahl der Programmzeilen während ihrer Open-Source-Tätigkeit geringer. Aufgrund der Abschätzung der Stundenzahl für die Stichprobe der proprietären Entwickler sind diese Angaben nur bedingt verlässlich.

Dass große Unterschiede in der Anzahl der erstellten Programmzeilen zwischen einzelnen Entwicklern gefunden wurden, entspricht den Ergebnissen anderer Studien im Open-Source-Bereich. So stellt Mockus et al. (2000) fest, dass die obersten 15 Entwickler für mehr als 88 % der erstellten Programmzeilen verantwortlich waren. In der Studie von Koch und Schneider (2000) betrug die Standardabweichung aller im Projekt erstellter Zeilen 67.000 – bei einem Mittelwert von 21.000. Vergleichbare Unterschiede lassen sich auch in weiteren Studien finden (Dempsey et al., 1999; Hertel et al., 2003). Die Validität der erstellten Programmzeilen als Indikator für Produktivität ist weiterhin umstritten. Häufige Kritikpunkte sind Unterschiede zwischen verschiedenen Programmiersprachen in der Anzahl der zur Problemlösung benötigten Zeilen und die fehlenden Berücksichtigung der Aufgabenschwierigkeit sowie der Qualität der Arbeit (z. B. Caban, Cimino, Swencionis, Ginsberg & Wylie-Rosett, 2001; Rauscher & Smith, 1995). Die geringe Korrelation zwischen der allgemeinen Leistung und der Anzahl der erstellten Programmzeilen könnte ein Indikator dafür sein, dass die Anzahl der erstellten Zeilen nicht als alleiniges Produktivitätsmaß dienen kann. Dass es zumindest möglich ist, auch so große Unterschiede in der Anzahl der erstellten Programmzeilen auf Unterschiede in der Produktivität zurückzuführen, zeigen Untersuchungen zum Verhältnis zwischen dem am wenigsten produktiven und dem produktivsten Mitarbeiter. Dieses liegt bei Produktionstätigkeiten bei 1:2 bis 1:3, für Manager zwischen 1:3 bis 1:6, für Versicherungsverkäufer bei 1:14 und für Rechtsanwälte bei 1:20 (McCormick & Tiffin, 1974). Die Schätzungen für Programmierer sind noch extremer (Schuler, 1995, zitiert nach Bergmann, 2000b).

Insgesamt kann die Hypothese 3 als teilweise bestätigt angesehen werden. Dass Unterschiede für die Zusammenhänge von allgemeiner Arbeitszufriedenheit, intrinsischer Motivation und organisationaler Identifikation mit der Abteilung mit den Tätigkeitsmerkmalen gefunden wurden, bleibt jedoch ein bedeutender Befund. Die auf explorativem Wege gesuchten Erklärungen für diese Ergebnisse können so noch nicht vollständig zufriedenstellen. Bis diese Befunde ausreichend erklärt sind, bleibt die Frage offen, ob die Wirkungszusammenhänge in Open-Source-Gemeinschaften mit den Wirkungszusammenhängen mit anderen Tätigkeiten in Unternehmen identisch sind. Es kann nicht ausgeschlossen werden, dass ein eigenes, möglicherweise komplexeres Modell für den Zusammenhang von Tätigkeitsvariablen und organisationalen Kriterien aufgebaut werden muss. Die geringe aufgeklärte Varianz für die organisationalen Kriterien

könnte ein Hinweis auf das unzureichende Modell oder aber auch ein Indikator für nicht-lineare Zusammenhänge zwischen Tätigkeitsmerkmalen und organisationalen Kriterien sein. Ingesamt kann eine positivere Einschätzung von organisationalen Kriterien in Open-Source-Projekten als in proprietären Entwicklungsumgebungen beobachtet werden.

7.4 Der Einfluss von Moderatoren

Für die Entscheidung, ob die vorgefundenen positiven Merkmale der Open-Source-Entwicklung auf das Entwicklungsmodell oder auf den Charakter von unbezahlter Freiwilligenarbeit zurückzuführen sind, spielen die Tätigkeiten von bezahlten Open-Source-Entwicklern eine entscheidende Rolle. Diese Ergebnisse implizieren, dass die Entlohnung keinen Einfluss auf die Gestaltung der Tätigkeitsmerkmale hat. Sowohl Open-Source-Entwickler, die Geld für ihre Arbeit erhielten, als auch solche, die kein Geld erhielten, berichteten in gleicher Weise positiv über die Gestaltung ihrer Tätigkeit. Der einzige signifikante Unterschied, der gefunden wurde, betrifft die Bedeutsamkeit der Tätigkeit. Dieser Unterschied in der Einschätzung der Wirkung der eigenen Arbeit auf die Außenwelt könnte sich damit erklären lassen, dass nur Entwickler in wichtigen Projekten einen Sponsor finden, der ihre Tätigkeit entlohnt. Es scheint plausibel anzunehmen, dass Open-Source-Entwickler, die keine Entlohnung erhalten, in Projekten arbeiten, die eine geringere wirtschaftliche Bedeutung haben, weil sie vielleicht nur für einen Nutzerkreis Relevanz haben oder im Schatten von erfolgreichen Konkurrenzprodukten stehen. Eine abschließende Aussage hierzu ist nicht möglich, da keine objektiven Hinweise auf die Bedeutsamkeit, wie die Anzahl der Nutzer, erhoben wurde. Ein zusätzlicher und wichtiger Indikator für den tatsächlichen Unterschied zwischen dem Open-Source-Entwicklungsmodell und den Entwicklungsmodellen der proprietären Softwareentwicklung sind die Ergebnisse aus dem Vergleich der Angestellten in beiden Stichproben. Berücksichtigt man, dass hier beide Gruppen vergleichbare äußere Arbeitsbedingungen aufweisen (immerhin über 60 % der Open-Source-Entwickler arbeiten im Büro), bestätigt sich eindrücklich, dass das Open-Source-Entwicklungsmodell deutliche höhere Werte in den Merkmalen der Komplexität aufweist.

Wie auch in anderen Studien zur Bezahlung in der Open-Source-Softwareentwicklung (z. B. zu Motiven von bezahlten und unbezahlten Entwicklern Lakhani & Wolf, 2003) konnte nur ein geringer Zusammenhang zwischen Bezahlung und dem Verhalten der Entwickler nachgewiesen werden. Für die untersuchten organisationalen Kriterien konnte ausschließlich bei der Leistung ein signifikant größerer Anteil an aufgeklärter Varianz festgestellt werden. Dabei zeigten solche Entwickler eine größere Leistung, die eine Bezahlung für ihre Tätigkeit erhielten. In anderen Studien zum Einfluss von Be-

zahlung konnte ein Zusammenhang zur Quantität, jedoch nicht zur Qualität der Arbeit gefunden werden (Jenkins, Mitra, Gupta & Shaw, 1998). Der bei Luthiger (2006) gefundene Unterschied im Flow-Erleben bei bezahlten und unbezahlten Open-Source-Entwicklern konnte nicht bestätigt werden.

Auch der gesondert untersuchte Einfluss von Entlohnung auf die intrinsische Motivation konnte wie schon in einer Reihe von anderen Studien (zusammenfassend Cameron et al., 2001; Deci et al., 1999) nicht bestätigt werden.

7.5 Diskussion der angewandten Methoden

Im vorliegenden Fall wurden Instrumente und Methoden der Arbeits- und Organisationsdiagnose genutzt, um Aspekte der Gestaltung einer teilweise unbezahlten, räumlich verteilten Form intellektueller Tätigkeit im Bereich der Softwareentwicklung zu analysieren. Im Hinblick auf diese Abweichung vom normalen Einsatzbereich erscheint die Frage angebracht, inwieweit eine solche Anwendung zulässig und sinnvoll ist.

Ein Argument für die Anwendbarkeit von Tätigkeitsanalysen in Open-Source-Projekten leitet sich aus dem Umstand ab, dass in Open-Source-Projekten Software entwickelt wird. Aus vorangegangenen Untersuchungen ist bekannt, dass Arbeitsanalysen für Tätigkeiten in der Softwareentwicklung bereits häufig angewendet wurden und auch für diesen Bereich sinnvolle Ergebnisse liefern (z. B. Couger & Zawacki, 1980; Goldstein, 1989; Lending & Chervany, 1997; McKnight & Chervany, 1998).

Ein weiteres Argument basiert auf dem Umstand, dass Open-Source-Softwareentwicklung für einige Entwickler eine Form der Freiwilligenarbeit darstellt. Zumindest bei kleineren Projekten wird die Arbeitsleistung unentgeltlich erbracht. Dies macht Open-Source vergleichbar mit anderen Formen der Freiwilligenarbeit. Die größte Verbreitung findet die Freiwilligenarbeit im sozialen Bereich. Hier konnte gezeigt werden, dass Arbeitsanalysen für die Bewertung von freiwilligen Tätigkeiten durchaus geeignet sind und die Instrumente in diesem Bereich angewendet werden können (Burkolter et al., 2006; Güntert & Wehner, 2005; Wehner & Güntert, 2005). Die Ergebnisse der vorliegenden Arbeit bekräftigen die Anwendbarkeit für freiwillige Tätigkeiten in der Softwareentwicklung. Eine Auswertung der freien Kommentarfelder des Fragebogens ergab keine Bemerkung zu einer grundsätzlichen oder teilweisen Unangemessenheit des Fragebogens für die Open-Source-Entwicklung. Zum anderen konnte gezeigt werden, dass Tätigkeitsmerkmale auch im Open-Source-Umfeld einen bedeutenden Beitrag zur Aufklärung der Varianz der organisationalen Kriterien beitragen können.

Neben dem ungewöhnlichen Anwendungsumfeld der Open-Source-Softwareentwicklung zeichnet sich diese Untersuchung durch ein breiteres Spektrum an erfassten Tätigkeitsmerkmalen aus. Die Beschränkung der Forschung zur Arbeitsgestaltung auf die begrenzte Auswahl der Merkmale, die von Turner und Lawrence (1965) eingeführt wurde, ist mehrfach kritisiert worden (z. B. Aldag et al., 1981; z. B. Roberts & Glick, 1981; Stone & Gueutal, 1985). Die Hinzunahme von Merkmalen hat sich in dieser Untersuchung als gewinnbringend erwiesen. Die Tätigkeitsmerkmale Rückmeldung durch Personen, soziale Unterstützung sowie die Interdependenz der Aufgaben und Ziele haben einen signifikanten zusätzlichen Anteil an aufgeklärter Varianz erbracht; wobei insbesondere bei Tätigkeiten im Open-Source-Umfeld eine höhere Prognosekraft durch die Hinzunahme der neuen Merkmale zu beobachten war.

Insgesamt hat sich das Messmodell der Untersuchung bestätigt. Dass die beiden Skalen Anforderungsvielfalt und Aufgabenvielfalt zu einer zusammengefasst werden mussten, deckt sich durchaus mit anderen Ergebnissen. So weisen die Vielfaltskalen des JCI (Aufgabenvielfalt) und des JDS (Anforderungsvielfalt) eine hohe Interkorrelation von 0,72 (Pierce & Dunham, 1978a) auf. Auch haben beide Skalen z. B. eine vergleichbare Beziehung zur allgemeinen Arbeitszufriedenheit, jedoch leicht unterschiedliche Beziehungen zur Leistung (Fried, 1991). Dass teilweise hohe Interkorrelationen zwischen einzelnen Tätigkeitsmerkmalen vorzufinden sind, muss kein Hinweis auf eine mangelnde diskriminante Validität der eingesetzten Skalen sein, sondern kann möglicherweise auch ein Hinweis auf einen echten Zusammenhang zwischen einzelnen Tätigkeitsmerkmalen sein (Taber & Taylor, 1990). So sollte auch eine hohe Autonomie bei der Gestaltung der eigenen Tätigkeit einen Einfluss auf die Vielfalt der Tätigkeit haben.

Die in dieser Arbeit angewandten Verfahren zur Arbeitsanalyse basieren auf einer Befragungsmethode, die zu subjektiven Aussagen über die Tätigkeitsmerkmale führt. Im Gegensatz zu objektiven Arbeitsanalyseverfahren können hierdurch keine von den Beurteilenden unabhängigen Aussagen über die Tätigkeit getroffen werden (Oesterreich & Volpert, 1987). Meta-Analysen z. B. von Fried und Ferris (1987) zeigen, dass die subjektive Wahrnehmung durch den Stelleninhaber teilweise nur mittelmäßig mit der Beurteilung durch Außenstehende (Experten oder Vorgesetzte) übereinstimmt. Allerdings kann man davon ausgehen, dass mögliche Verfälschungen durch Selbsteinschätzung nicht gravierend ausfallen. So konnten die TeilnehmerInnen von beiden Stichproben von der Anonymität der Ergebnisse ausgehen, was für eine höhere Korrelation zwischen Selbstbewertung und objektiven Maßen spricht (Pym & Auld, 1965). Auch sollten TeilnehmerInnen davon ausgegangen sein, dass die Maße nicht für eine Entscheidung über die Anreize oder die Ressourcenzuteilung genutzt werden, was eine stärkere Verzerrung der Selbsteinschätzung hätte zur Folge haben können (Scacchi,

1995). Insgesamt war es das Ziel der Untersuchung, keine objektive Beschreibung der Tätigkeitsmerkmale, sondern das subjektive Ergebnis der Auseinandersetzung mit den Tätigkeitsmerkmalen zu erfassen (Hackman, 1969; Hackman, 1970).

Eine mögliche Quelle der systematischen Verzerrung könnten Unterschiede in der Persönlichkeit von Open-Source- und proprietären Softwareentwicklern sein. Die Persönlichkeitsstruktur könnte mit unterschiedlichen Antworttendenzen einhergehen. Die Stichprobengröße der TeilnehmerInnen, die sowohl den Fragebogen für Open-Source- als auch den für proprietäre Entwickler beantwortet haben, ist bedauerlicherweise nur gering. Eine größere Stichprobe hätte die Anwendung von geeigneteren Methoden erlaubt und damit die Zuversicht vergrößert, dass die vorgefundenen Effekte nicht auf interpersonellen Unterschieden beruhen. Nichtsdestotrotz ist die Konsistenz der vorgefundenen Effekte ein Indiz dafür, dass solche systematischen Verzerrungen nicht existieren. Eine Verzerrung der Ergebnisse, die wissentlich geschieht, z. B. um aus ideologischen Gründen kostenlose Software zu fördern, oder unwissentlich, weil sie kognitive Dissonanzen in Bezug auf ihr Engagement im Open-Source-Bereich vermeiden wollen, kann jedoch nicht ausgeschlossen werden.

Durch die ausschließliche Verwendung eines Fragebogens als Datenquelle können außerdem Common Method Biases nicht ausgeschlossen werden. Ein Common Method Bias liegt vor, wenn ein Teil der vorgefundenen Varianz nicht auf das erhobene Konstrukt, sondern auf die verwendete Erhebungsmethode zurückzuführen ist. Zum Beispiel konnte Espinosa et al. (2002) bei einer Untersuchung zum Einfluss von Distanz auf die Koordination von Softwareentwicklungsteams für Interviews und Archivauswertungen einen Effekt zeigen, der mit dem Befragungsinstrument nicht nachgewiesen werden konnte. In Abhängigkeit von dem zu erhebenden Konstrukt kann bei Selbstauskünften ein solcher Effekt vorliegen. Soll z. B. Arbeitszufriedenheit erhoben werden, ist ein Selbstauskunftsfragebogen nicht zu vermeiden, jedoch für die erhobenen Tätigkeitsmerkmale wären prinzipiell auch Beobachtungsverfahren möglich. Subjektive Verzerrungen und Bewertungen der Tätigkeitsmerkmale durch die Arbeitnehmer bzw. ProjektteilnehmerInnen ließen sich durch die Anwendung von alternativen Verfahren, z. B. nicht-reaktiven Verfahren vermeiden. So wurden in Open-Source-Projekten bereits Untersuchungen auf Basis von Dokumentenanalysen wie z. B. Diskussionsforen, Protokolldateien oder Quellcodeanalysen durchgeführt (z. B. Crowston & Scozzi, 2002b; z. B. Dempsey et al., 1999; Mockus et al., 2002). Solche Dokumente reichen als Datenquelle nicht aus, um eine umfassende Tätigkeitsanalyse durchzuführen, da sie wenige Informationen darüber enthalten, wie einzelne Entwickler ihre Tätigkeit tatsächlich gestalten. Der Anwendung von Beobachtungsverfahren stehen im Open-Source-Umfeld logistische Probleme entgegen: Zum einen ist der Aufwand für die Durchführung von Beobachtungen in weltweit verteilten Projekten, bei denen zumindest ein Teil der Arbeit

im privaten Umfeld erledigt wird, erheblich. Zum anderen nimmt die Qualität der Beobachtung bei komplexen Tätigkeiten (wie sie bei der Softwareentwicklung vorliegen) ab und auch Beobachter unterliegen Bewertungs-, Interpretations- und "Verzerrungsprozessen" (Dunckel, 1999a). Die Beobachtungsmethode erscheint insgesamt für den Open-Source-Bereich nur bedingt anwendbar zu sein. Von Schmitt (1994) wird die Anwendung von Forschungsdesigns, die grundsätzlich der Gefahr eines Methodenfehlers unterliegen, für Forschungen in neuen Bereichen, wie sie bei Open-Source-Tätigkeiten vorliegen, als akzeptabel bezeichnet.

Ein weiteres Problem von subjektiven Verfahren, das unabhängig von dem untersuchten Bereich auftritt, ist das der Kausalität der Zusammenhänge. Arbeitsanalysemodellen wie dem JCM von Hackman und Oldham (1976) liegt die Annahme zugrunde, dass objektive Tätigkeitsmerkmale die Wahrnehmung der Tätigkeitsmerkmale beeinflussen, welche dann zu affektiven Reaktionen führen. Ein alternatives Kausalitätsmodell unterstellt die Theorie der sozialen Informationsverarbeitung. Als wichtigste Determinanten der wahrgenommenen Tätigkeitsmerkmale gilt die soziale Umwelt des Arbeitenden und nicht die objektiven Eigenschaften der Tätigkeit (Salancik & Pfeffer, 1978). Das soziale Umfeld, insbesondere in der Form von Kollegen, beeinflusst die Einstellungen und die kognitiven Strukturen. In dieser sozial-konstruierten Umwelt werden der Tätigkeit Eigenschaften zugeschrieben, die in Übereinstimmung mit den Erwartungen des Einzelnen stehen. Den objektiven Merkmalen der Tätigkeit werden dabei allenfalls moderierende Einflüsse zugesprochen (Thomas & Griffin, 1983). Dass objektive Merkmale einen Einfluss auf die subjektive Wahrnehmung haben, belegen eine Reihe von Feld- und Laborexperimenten, die objektive Veränderungen an Tätigkeiten vorgenommen haben und Varianz in den subjektiven Tätigkeitsmerkmalen beobachten konnten. Ein Zusammenhang zwischen vertikaler Aufgabenerweiterung (task enrichment) und den Tätigkeitsmerkmalen des JDS konnte in Laborexperimenten (z. B. Griffin, Bateman, Wayne & Head, 1987; O'Reilly & Caldwell, 1979; z. B. Umstot, Mitchell & Bell Jr, 1978; White & Mitchell, 1979) und Feldexperimenten (z. B. Griffeth, 1985; Orpen, 1979) gezeigt werden. Zumindest in Laborexperimenten konnten auch Effekte von Manipulationen an einzelnen objektiven Tätigkeitsmerkmalen auf die korrespondieren Skalen des JDS beobachtet werden. Beispiele hierfür sind die Veränderung der Autonomie (Farh & Scott, 1983), der Vielfalt (Jackson & Zedeck, 1983) und der Bedeutsamkeit (Ferris, Fedor, Rowland & Porac, 1985). Bei diesen Studien zeigten schon geringfügige Änderungen der Tätigkeitsmerkmale deutliche Effekte auf den Skalen des JDS (zusammenfassend Taber & Taylor, 1990). In einer Reihe von Experimenten konnten ebenfalls Einflüsse durch soziale Hinweisreize beobachtet werden; diese waren in der Regel geringer als die Effekte durch objektive Tätigkeitsmodifikationen (zusammenfassend Taber & Taylor, 1990). Ebenfalls gegen eine große Bedeutung von sozialen Hinweisreizen sprechen die Untersuchungen von Vance

und Biddle (1985). In ihren Ergebnissen konnte ein Einfluss von sozialen Reizen bestätigt werden, der aber mit zunehmender Ausführungsdauer der Tätigkeit abnahm. Dies spricht für die Hypothese, dass Hinweisreize nur so lange von Bedeutung sind, wie nicht ausreichend Informationen durch die Ausführung der Tätigkeit selbst vorliegen. Auch scheint fraglich, ob soziale Informationen einen Einfluss auf das Verhalten oder nur die Wahrnehmung der Tätigkeit haben (Kilduff & Regan, 1988). Trotz der aufgetretenen Fragen über den Zusammenhang zwischen wahrgenommenen und objektiven Tätigkeitsmerkmalen kommen aktuelle Arbeiten zu dem Schluss, dass Veränderungen von subjektiven Merkmalen der Tätigkeit auf Veränderungen von objektiven Tätigkeitsmerkmalen beruhen (Glick, Jenkins Jr & Gupta, 1986; Taber & Taylor, 1990).

Rückschlüsse von der Stichprobe auf die Verhältnisse in der Grundgesamtheit sind nur möglich, wenn die Stichprobe repräsentativ ist. Es gibt einige Hinweise, um an der Repräsentativität der vorliegenden Stichprobe zu zweifeln. Eine Gefährdung der Repräsentativität wurde durch die gewählte Sprache des Fragebogens erzeugt. Dass dieser nur in Englisch vorlag, könnte eine systematische Selektion in der Stichprobe der proprietären Softwareentwicklung zur Folge gehabt haben. Da die proprietäre Stichprobe von Entwicklern aus deutschen Unternehmen dominiert wurde, kann trotz allgemein hoher Ausbildungsstandards in der Softwareentwicklung und der Tatsache, dass Englisch die "lingua franca" der IT-Branche ist, dennoch davon ausgegangen werden, dass besonders gut ausgebildete Mitarbeiter den Fragebogen häufiger beantwortet haben als weniger gut ausgebildete. Ein Hinweis auf die Selektivität könnte der hohe Anteil von Mitarbeitern mit Führungsaufgaben in der Stichprobe sein. Da Mitarbeiter mit besserer Ausbildung und höherer hierarchischer Position eine positivere Tätigkeitsgestaltung aufweisen (Brodbeck & Frese, 1994; Couger & Zawacki, 1980), sollte eine Selektion in dieser Richtung tendenziell zu einer zu positiven Einschätzung der Tätigkeitsgestaltung in der proprietären Softwareentwicklung führen. Dies würde bedeuten, dass die vorgefundenen Unterschiede bei der Komplexität der Tätigkeitsmerkmale zwischen Open-Source- und proprietärer Softwareentwicklung eher unterschätzt würden. Neben der Sprache können auch länderspezifische Effekte in der Wahrnehmung des Motivationspotenzials einer Tätigkeit in der Softwareentwicklung nicht ausgeschlossen werden (Couger et al., 1989).

Im Allgemeinen sind bei einer Erhebung über das Internet Selektionseffekte durch unterschiedliche Computerkompetenz zu erwarten (Dillman, 2007). In einer Stichprobe von Softwareentwicklern sollte ausreichendes Wissen bei allen Teilnehmern vorauszusetzen sein und eine Selektion auf Basis dieses Kriteriums unwahrscheinlich machen. Eine positive Konsequenz aus der Verwendung eines Internet-Fragebogens sollte die größere Offenheit der Antwortenden sein (Turner et al., 1998).

Mit der Erhebung über das Internet steht auch die Form der Kontaktierung der Softwareentwicklung im Zusammenhang. Die Kontaktierung der Open-Source-TeilnehmerInnen durch eine Veröffentlichung des Aufrufs zur Teilnahme in Foren führt ähnlich wie nicht personalisierte Teilnahmeaufforderungen in Zeitungen oder Zeitschriften zu einer höheren Zahl nicht Antwortender und damit zu einer geringeren Repräsentativität als bei Verfahren mit einem persönlichen Kontakt (Dillman, 2007). Aufgrund der öffentlichen Erreichbarkeit der Befragung kann nicht ausgeschlossen werden, dass auch Personen an der Befragung teilgenommen haben, die nicht zur beabsichtigten Zielgruppe gehören. Zudem lässt dieses Verfahren keine genaue Kontrolle der Rücklaufquoten zu, die in diesem Fall nur abgeschätzt werden konnten und bei der Open-Source-Stichprobe sehr niedrig erscheinen. Negativ auf die Rücklaufquote könnte sich ausgewirkt haben, dass aufgrund der Kontaktierung der Personen die Anrede nicht persönlich gestaltet werden konnte (Dillman, 2007). Unabhängig vom eingesetzten Medium sind mehrere Kontakte nötig, um eine befriedigende Rücklaufquote zu erhalten (Dillman, 1991; Heberlein & Baumgartner, 1978; Scott, 1961). Auch die Festlegung eines zeitnahen Endtermins hätte sich förderlich auf die Quote auswirken können (Petrie, Moore & Dillman, 1998). Da nicht bekannt war, wann einzelne Interessenten die Seite betrachten, war dieses nicht möglich. Zur Steigerung der Rücklaufquote hätte u. U. auch die Gewinnung eines bekannten Sponsors aus der Open-Source-Gemeinschaft beitragen können; eine höhere Aufmerksamkeit und größeres Vertrauen wären die Folgen gewesen (Dillman, 2007). Möglicherweise war das Thema der Befragung für eine technisch orientierte Gemeinschaft wie Softwareentwickler nur von geringem Interesse. Zudem wurden einige der Fragen von den Ausfüllern als "wiederholend" empfunden, was das Interesse weiter gesenkt haben und damit zu einer niedrigeren Quote beigetragen haben könnte (Heberlein & Baumgartner, 1978). Auch die relative Länge des Fragebogens könnte die Rücklaufquote negativ beeinflusst haben, da längere Fragebögen tendenziell von weniger Personen beantwortet werden (Heberlein & Baumgartner, 1978). In direktem Zusammenhang mit der Länge des Fragebogens steht auch die teilweise geringe interne Konsistenz einiger Skalen. Da eine relativ hohe Anzahl von Konstrukten erfasst werden sollte, musste die Länge der Skalen begrenzt werden. Eine niedrige Anzahl von Items wirkt sich jedoch negativ auf Cronbachs Alpha aus (Lienert & Raatz, 1994). Es kann davon ausgegangen werden, dass bei längeren Skalen die Konsistenz größer gewesen wäre.

Bei der Auswahl der anzuwendenden statistischen Testverfahren musste aufgrund der Fragestellung auch auf die Gleichheit von Regressionsgewichten geachtet bzw. musste der Nicht-Einfluss von Faktoren getestet werden. Beides machte die Auswahl der Nullhypothese als inhaltliche Hypothese notwendig. Dies ist eigentlich nach der Logik von R. A. Fisher beim Testen von Hypothesen nicht möglich (z. B. Blackwelder, 1982; Hager,

2000). Dennoch wird diese Möglichkeit in statistischen Lehrbüchern aufgeführt (Bortz, 1999) und ist aus erkenntnistheoretischen Gründen akzeptabel (Bredenkamp, 1980).

Zusammenfassend zu den angewandten Methoden und dem Untersuchungsvorgehen kann man sagen, dass eine Einsetzbarkeit des verwendeten Instruments im Open-Source-Umfeld gerechtfertigt erscheint. Die Erweiterung des Fragebogens um Skalen, die nicht im JCM berücksichtigt wurden, hat insbesondere in der Open-Source-Stichprobe die Prognosekraft erhört. Eine Beschränkung der Aussagekraft der Ergebnisse durch eine Beschränkung auf ein Verfahren der Selbstauskunft ist ebenso wenig auszuschließen wie eine bewusste oder unbewusste Beeinflussung der Ergebnisse durch eine positive Darstellung der Tätigkeit im Open-Source-Bereich. Alternative Methoden wie Beobachtungsverfahren oder die Nutzung von verfügbaren Datenquellen scheinen keine realistische Alternative zu sein, um den Common Method Bias zu reduzieren. Eine Unterschätzung der gefundenen Unterschiede zwischen Open-Source- und proprietären Softwareentwicklern durch den englischsprachigen Fragebogen kann nicht ausgeschlossen werden. Eine Generalisierbarkeit der Ergebnisse wird durch die geringere Rücklaufquote gefährdet.

Nach der Diskussion der methodischen Einschränkungen soll im folgenden Abschnitt ein Ausblick darauf gegeben werden, wie die Ergebnisse die zukünftige Softwareentwicklung beeinflussen könnten, welche Konsequenzen sich für die Arbeitsgestaltung außerhalb der Softwareentwicklung ergeben und was dies für die psychologische Forschung bedeutet.

7.6 Gestaltungsansätze für die Softwareentwicklung

Betrachtet man den relativ hohen Anteil an bezahlten Open-Source-Entwicklern und die Tatsache, dass in der vorliegenden Untersuchung kein Einfluss durch den Moderator Bezahlung gefunden wurde, so kann man nicht davon sprechen, dass es sich bei Open-Source nur um die Tätigkeiten von freiwilligen Hobbyentwicklern handelt. Es stellt sich vielmehr die Frage, was einer Übertragung der Erkenntnisse aus dem Open-Source-Entwicklungsmodell auf Softwareentwicklung im Unternehmen im Wege steht. Aktuelle Trends im Bereich der Softwareentwicklung orientieren sich eher in eine gegenteilige Richtung. Sie beschäftigen sich mit der "Industrialisierung der Softwareentwicklung" (z. B. Wirtschaftsinformatik, geplantes Sonderheft Heft 3/2007). Diskutiert wird in diesem Rahmen darüber, ob eine Abkehr vom praktizierten "handwerklichen" Modell in der Softwareentwicklung hin zu einem industriellen Fertigungsprozess die Lösung für bestehende Qualitätsprobleme darstellen könnte. Auch der zunehmende Einsatz von CASE-Werkzeugen und die Betonung von standardisierten Methoden der Entwicklung könnten zu einer verstärkten Taylorisierung der Softwareentwicklung führen. Die Frage

ist, ob das Open-Source-Entwicklungsmodell einen Gegenpol zu diesen Bestrebungen darstellen könnte.

Wenn in Unternehmen Open-Source eine Rolle spielt, so geschieht dieses bei der Nutzung von Open-Source-Produkten für den eigenen Bedarf, Herstellung und Vertrieb von kostenpflichtigen Distributionen oder in der Weiterentwicklung eigener proprietärer Produkte unter einer freien Lizenz. Wenn Unternehmen Open-Source-Entwickler einbeziehen, ist die Möglichkeit der Kosteneinsparung durch einen verringerten Aufwand für das Testen der Software sicher ein nicht unwesentliches Argument. Immerhin werden nach Schätzungen 50 % oder mehr der gesamten Ressourcen der Softwareentwicklung für das Testen aufgewendet (Shepard, Lamb & Kelly, 2001). Neben dem Testen kann auch die Integration von Nutzern in den Entwicklungsprozess sehr aufwendig sein. So sind in einigen Studien negative Zusammenhänge zwischen einer starken Nutzerbeteiligung und der Effizienz, Qualität und Flexibilität des Produktes beobachtet worden (Heinbokel, Sonnentag, Frese & Stolte, 1996). Auch wenn ein überwiegender Teil an Studien positive Auswirkungen von Nutzerbeteiligung gezeigt hat (Kujala, 2003), so ist dies doch ein Indikator dafür, dass die Nutzerbeteiligung eine Herausforderung darstellen kann. In Open-Source-Projekten scheint diese sehr erfolgreich zu gelingen. In Open-Source-Projekten sind Nutzer sehr stark in den Entwicklungsprozess z. B. durch das Vorschlagen von neuen Funktionen, das Testen der Software oder das Melden von Fehlern integriert. Bei vielen Projekten sind die Entwickler gleichzeitig auch Nutzer der Software (Sillitti & Succi, 2005).

Damit der Open-Source-Entwicklungsprozess eine Alternative zur Taylorisierung der Softwareentwicklung bilden könnte, müsste dieser durch eine Integration in bestehende proprietäre Entwicklungsmodelle eine erfolgsversprechende Perspektive bieten können. So könnte eine Übernahme der sozialen und sozio-technischen Praktiken der Open-Source-Softwareentwicklung eine Möglichkeit zur Steigerung der Effektivität von Softwareentwicklung bieten (Crowston et al., 2004).

Die Ergebnisse dieser Arbeit deuten an, dass es auch für Unternehmen, die eine Offenlegung des Quellcodes ihrer Software nicht wünschen, möglich ist, von dem Open-Source-Entwicklungsmodell zu profitieren. Der Vergleich mit Open-Source-Projekten ist ein Hinweis darauf, dass in der proprietären Softwareentwicklung Möglichkeiten zur Verbesserung der Arbeitsgestaltung bestehen. Insbesondere zeigt die Gestaltung in der Open-Source-Softwareentwicklung, dass es möglich ist, die Komplexität der Aufgabe zu steigern und das Qualifikationspotenzial der Tätigkeit zu erhöhen. Welche Maßnahmen nun genau zu einer Verbesserung führen, kann auf Basis einer Tätigkeitsanalyse mit den eingesetzten Instrumenten nicht abgeleitet werden. Befragungsmethoden wie der JDS sind für die Planung von Gestaltungsmaßnahmen "zu grob" (Matern, 1983). Niedrige

Werte sind ein Hinweis auf die Notwendigkeit von Gestaltung, eine konkrete Maßnahme kann daraus nicht abgeleitet werden.

Hinweise auf konkrete Verbesserungsvorschläge können aber bei dieser Untersuchung aus den bekannten Eigenschaften des Open-Source-Entwicklungsmodells abgeleitet werden (vgl. Kapitel 3.2). So sollte etwa eine häufigere Veröffentlichung der fertigen Programmteile mehr Rückmeldung für die Entwickler bieten und den Eindruck der Geschlossenheit der Tätigkeit vergrößern. Mit einer stärkeren Modularisierung der Software ergibt sich eine größere Aufgabengeschlossenheit und mehr Autonomie für die Entwickler. Die Ermöglichung von Prozessen der Selbstorganisation, wie sie in der Open-Source-Entwicklung anzutreffen ist, sollte eine größere Autonomie und mehr Partizipationsmöglichkeiten für die Mitarbeiter ermöglichen, eine größere Aufgabengeschlossenheit bieten und das Potenzial der Tätigkeit zur Qualifikation vergrößern. Das Einrichten von virtuellen Gemeinschaften innerhalb von Unternehmen sollte sich förderlich auf die erlebten Partizipationsmöglichkeiten und das Qualifikationspotenzial auswirken. Selbstselektion könnte zu einer größeren Vielfalt und genauso wie Peer-Review zu einem größeren Qualifikationspotenzial führen.

Nicht alle diese Maßnahmen sind in der proprietären Softwareentwicklung gleich gut zu realisieren. Ein geringes Problem stellt z. B. eine stärkere Modularisierung oder eine häufigere Veröffentlichung von Quellcodes dar, da dieses mit der Änderung des Designs der Software oder der Schaffung bzw. Nutzung von entsprechender Infrastruktur möglich ist. In stärkerem Konflikt mit den Zielen des Unternehmens kann die Selbstorganisation oder -selektion stehen. Beides schränkt die Möglichkeit der zentralen Steuerung und damit der Ausrichtung der Organisation auf bestimmte Ziele ein. Unter Umständen bietet z. B. eine begrenzte Selbstselektion durch die Schaffung eines internen Arbeitsmarkts Möglichkeiten, um die Freiheit bei der Wahl der Aufgaben zu vergrößern und gleichzeitig eine Organisation an ihren Zielen auszurichten.

7.7 Implikationen für zukünftige Forschung

Die Open-Source-Softwareentwicklung zeigt ein funktionsfähiges Beispiel für eine Arbeits- und Organisationsgestaltung, die ohne formelle Macht und im hohen Maße selbstorganisiert in der Lage ist, eine koordinierte Leistung zu erbringen. Die Gestaltung der Tätigkeit wird dabei von den TeilnehmerInnen in vielen Belangen positiv erlebt. Auch wenn die in dieser Untersuchung vorgefundenen Ergebnisse nicht für alle Interpretationen ausreichend belastbar sind, so rechtfertigen sie zusätzliche Aufmerksamkeit für das Phänomen Open-Source auch aus der Perspektive der Arbeits- und Organisationspsychologie. Open-Source-Projekte könnten Feldstudien zu spezifischen

Fragestellungen ermöglichen, die für einen Einsatz in Unternehmen noch nicht geeignet sind und die in Laborexperimenten nur schlecht zu realisieren sind.

Eine der möglichen Fragestellungen betrifft den Einfluss der Organisationsstruktur auf die Arbeitsgestaltung. Die Überlegungen von Pierce und Dunham (1978b) und die Ergebnisse von Rousseau (1978) deuten an, dass ein negativer Zusammenhang zwischen Formalisierung und Zentralisierung und diversen Tätigkeitsmerkmalen vorliegt (z. B. Autonomie, Vielfalt, Rückmeldung und Aufgabengeschlossenheit). Anhand von Open-Source-Projekten, die unterschiedliche Formalisierungs- und Zentralisierungsgrade aufweisen, wäre eine genauere Untersuchung des Einflusses auf die Tätigkeitsmerkmale möglich.

Besonders auffällig ist das hohe Qualifizierungspotenzial, das Open-Source-Projekte im Vergleich zur proprietären Softwareentwicklung aufweisen. Da aus Open-Source-Projekten keine Qualifizierungen durch Weiterbildungsangebote in Form von Kursen oder Seminaren bekannt sind, findet das gesamte Lernen während der Tätigkeit selbst statt. Solches Erfahrungslernen ("Learning by Doing") wird auch als "Action Learning" bezeichnet. Die vorliegenden Ergebnisse scheinen ein Hinweis auf die Effizienz des Lernens anhand einer konkreten Herausforderung zu sein, wie sie z. B. auch im Umfeld der Softwareentwicklung untersucht wurde (Vat, 2000).

Der Fokus der Arbeitspsychologie liegt eindeutig auf der Erwerbsarbeit. Der Sektor der Freiwilligenarbeit, aber auch der Haus- oder Familienarbeit, erhalten nur wenig Aufmerksamkeit (Resch, Bamberg & Mohr, 1997). Dies erscheint nicht gerechtfertigt, denn zum einen hat Freiwilligenarbeit eine hohe gesellschaftliche Bedeutung, und zum anderen könnte die Erforschung hilfreich sein, um freiwilliges Engagement innerhalb von Unternehmen besser zu verstehen. Freiwillige Aktivitäten bekommen in Unternehmen eine zunehmend größere Bedeutung (Hertel et al., 2000).

Die Open-Source-Softwareentwicklung kann dabei als besondere Form der Freiwilligenarbeit angesehen werden. Sie ist zwischen institutionalisierten Formen, wie sie z. B. in Vereinen, Interessenvereinigungen und Institutionen stattfindet, und informeller Freiwilligenarbeit, die man z. B. in Form von Nachbarschaftshilfen und Hausarbeit antrifft, einzuordnen. Unternehmen können von der Freiwilligenarbeit insbesondere die Ausrichtung der Arbeit an den Motiven der TeilnehmerInnen lernen. So ist die individuelle Motivation der entscheidende Grund für das Zustandekommen und den Fortbestand von Freiwilligenarbeit (Gensicke et al., 2005; Yeung, 2004). Daneben zählen Spaß an der Tätigkeit, die sozialen Kontakte und die Möglichkeit, etwas für das Gemeinwohl und die Unterstützung anderer Menschen zu unternehmen, zu den Gründen für das Engagement (Gensicke et al., 2005; Rosenbladt, 2000). Im Verlauf der freiwilligen Tätigkeit treten altruistische Motive in den Hintergrund und es kommt zu einer verstärkten Bedeutung

von egoistischen Motiven (Bierhoff, 2001). Einer der auffälligsten Unterschiede zur Erwerbstätigkeit ist, dass sogar in großen Vereinen die Mitbestimmungsmöglichkeiten von fast drei Viertel der Mitglieder als günstig eingeschätzt werden (Gensicke et al., 2005). Möglicherweise resultiert auch daraus, dass trotz der teilweise großen Herausforderungen, die z. B. ein freiwilliges Engagement im sportlichen oder sozialen Bereich mit sich bringt, der größte Teil der TeilnehmerInnen sich nicht überfordert fühlt (Gensicke et al., 2005).

In Forschungsarbeiten zur Motivation wird häufig der Einfluss von Entlohnung auf die intrinsische Motivation der TeilnehmerInnen diskutiert (Akin-Little, Eckert, Lovett & Little, 2004; Deci et al., 1999; Eisenberger et al., 1999). Die vorliegenden Ergebnisse zeigen tendenziell eine positive Auswirkung von Entlohnung auf die Motivation. So scheint in der Freiwilligenarbeit Entlohnung eine Signalwirkung für die erlebte Bedeutung einer Tätigkeit zu haben (siehe auch Güntert & Wehner, 2005). Ungeklärt bleibt, ob ein solcher Effekt auch im Rahmen von normaler Erwerbstätigkeit, bei der alle Beschäftigten eine Entlohnung erhalten, eine positive Auswirkung zeigt. Ebenfalls ungeklärt bleibt z. B. die Frage, ob das Ausmaß der Belohnung diesen Effekt verstärken kann.

Open-Source Projekte können auch einen interessanten Untersuchungsgegenstand für die Person-Job-Fit oder die Berufsinteressenforschung darstellen. In Open-Source-Projekten kann beobachtet werden, welche Auswirkungen die Möglichkeit, Arbeit frei zu wählen, auf die Zufriedenheit mit der Tätigkeit und auf die Wahrnehmung der Tätigkeitsmerkmale haben. In Zeiten, da in Unternehmen zunehmend Marktmechanismen als Mittel der Koordination genutzt werden und Unternehmensbereiche miteinander oder zu externen Wettbewerbern in Konkurrenz treten, werden auch Fragen zu den Auswirkungen eines internen Arbeitsmarktes auf die Mitarbeiter und die Organisation für die psychologische Forschung zunehmend wichtiger. Eine Partizipation an der Entscheidung, welche Tätigkeit ausgeübt werden soll, könnte mehr Mitarbeitern die Möglichkeit bieten, eine für ihre Fähigkeiten und Bedürfnisse optimale Tätigkeit zu finden. In weiteren Untersuchungen zu Open-Source-Projekten sollte der Prozess von Projektauswahl und Wechsel zwischen Projekten eine besondere Berücksichtigung finden. Bisher gibt es nur Hypothesen über die Selbstselektion in Open-Source-Projekten, aber keine empirischen Ergebnisse zum tatsächlichen Verhalten von Entwicklern. Noch kann keine Aussage darüber getroffen werden, wie mobil sie wirklich sind und wie hoch bzw. niedrig die Kosten für z. B. einen Projektwechsel sind.

Unterstellt man Open-Source-Projekten, dass sie den TeilnehmerInnen Freiheit bei der Auswahl und Gestaltung ihrer Tätigkeiten bieten, sollten sich die Eigenschaften der Tätigkeit einem für die TeilnehmerInnen optimalen Wert annähern. Auch in der vorliegenden Untersuchung erreichen Open-Source-Entwickler selbst bei den positiven Ausprägungen der Merkmale der Komplexität keine maximalen Werte. Dies könnte ein

möglicher Hinweis darauf sein, dass für viele Tätigkeitsmerkmale der optimale Wert unterhalb des Maximalen liegt und damit für einen nicht-linearen Zusammenhang zwischen Tätigkeitsmerkmalen und organisationalen Kriterien sprechen. Unterstützt würde eine solche Interpretation durch die bereits vorgefundenen umgekehrt U-förmigen Zusammenhänge (z. B. Xie & Johns, 1995). Eine andere Erklärung wäre, dass durch die Interaktionen zwischen einzelnen Tätigkeitsmerkmalen auch bei freien Gestaltungsmöglichkeiten das Erreichen eines maximalen Wertes für alle Merkmale nicht möglich ist.

Für die weitere Erforschung der Tätigkeiten in Open-Source-Projekten wären weitere Studien an Tätigkeitsmerkmalen mit umfangreichen und repräsentativen Stichproben wünschenswert. Insbesondere die heterogenen Ergebnisse für soziale Merkmale demonstrieren, dass die Zusammenhänge in diesem Bereich noch nicht vollständig verstanden sind und weitere Theorien wünschenswert wären. Insbesondere eine genauere Betrachtung der interpersonellen Kommunikation wäre wünschenswert. Aus vorliegenden Studien zur Softwareentwicklung ist bekannt, dass Kommunikation einen erheblichen Einfluss auf den Entwicklungsprozess haben kann. Einerseits wird postuliert, dass Kommunikation die Produktivität in der Softwareentwicklung senkt (u. a. Sommerville, 1989) und andererseits gehen Forscher von einer Steigerung der Produktivität durch Kommunikation aus (Brodbeck, 1994a; Campbell & Gingrich, 1986). Dies zeigt, dass die Auswirkungen der Kommunikation im Bereich der Softwareentwicklung noch nicht vollständig verstanden worden sind.

Bisher unerforscht ist die Frage nach Merkmalen, die in der Person des Open-Source-Entwicklers liegen. So könnte nicht nur die Gestaltung der Tätigkeit ein Grund für die Teilnahme sein, sondern Open-Source-Entwickler könnten sich durch spezielle Persönlichkeitsmerkmale von anderen Softwareentwicklern unterscheiden. Solange keine Untersuchungen vorliegen, können auch Interaktionseffekte zwischen der Persönlichkeit und dem Erleben der Tätigkeitsmerkmale nicht ausgeschlossen werden. Insgesamt kann nicht ausgeschlossen werden, dass die Übernahme einer Open-Source-Kultur für Organisationen mit unerwarteten Kosten (wie z. B. erhöhter Koordinationsaufwand oder Verlust von Mitarbeitern, die hierarchische Strukturen bevorzugen) verbunden ist (Hertel, 2007).

Möglicherweise ist für Softwareunternehmen nicht die Frage entscheidend, ob sie Open-Source-Konzepte in ihr Entwicklungsmodell integrieren sollen, sondern welches Entwicklungsmodell für welches Projekt am besten geeignet ist. So geht aus der Untersuchung von Mantei (1981) hervor, dass ein dezentralisiertes demokratisches Vorgehen, vergleichbar mit der Open-Source-Entwicklung, insbesondere bei solchen Aufgaben angebracht ist, die relativ überschaubar sind und einen hohen Schwierigkeitsgrad haben. Weiterhin sollten diese Aufgaben lange andauern und nur eine geringe Modularität auf-

weisen. Der Fokus sollte bei diesen Aufgaben auf der Qualität liegen und der interpersonelle Austausch sollte wichtig für die Lösung sein. Im Gegensatz dazu ist bei Projekten, die weniger schwierig, größer, kürzer und höher modular sind, die Anwendung von kontrollierten dezentralisierten oder zentralisierten Teams zu empfehlen.

Die abschließende Frage ist, ob die Erkenntnisse zur Tätigkeitsgestaltung in der Softwareentwicklung auf die Softwareentwicklung beschränkt bleiben müssen oder ob sie übertragen werden können. Teile der präsentierten Ergebnisse, wie z. B. die häufige Veröffentlichung von Ergebnissen, sind nicht auf einen anderen Arbeitskontext zu übertragen, während die meisten Ergebnisse allgemeine Aspekte wie Selbstorganisation und -selektion betreffen und damit durchaus übertragbar sind. Eventuelle Veränderungen in der IT-Branche sollten nicht ohne Auswirkungen bleiben:

> *Die IT-Branche stellt eine Art Leitbranche für Arbeit in der Wissensgesellschaft dar, die für die ökonomische Entwicklung in Deutschland eine wichtige Rolle spielt. Die in dieser Branche üblichen Anforderungen an die Beschäftigten hinsichtlich Flexibilität, Eigenverantwortung, Arbeitseinsatz und Entlohnung sind Orientierungen, die auf andere Wirtschaftsbereiche ausstrahlen und zunehmend das Verständnis von Arbeit prägen. (Latniak & Gerlmaier, 2006, S. 1)*

Viele Eigenschaften von Open-Source-Projeken muten heute utopisch an. Es ist schwer vorstellbar, dass eine an wirtschaftlichen Interessen ausgerichtete Organisation sich wie ein Open-Source-Projekt aufstellt und handelt. Eine Prognose, wie sich Arbeit in Zukunft entwickeln wird, ist schwierig. Möglicherweise stellen jedoch Open-Source-Projekte die Avantgarde der Arbeitsgestaltung dar.

Anhang A: Tabellen

Tabelle 36: Varianzanalyse für unterschiedliche Tätigkeitsfelder in der Softwareentwicklung

	Mittelwert							F	
	Requirements Engineering	Software Design	Kodieren/ Debuggen	Testen	System-Administration	Management/ Koordination	Nutzer-Support		
Vielfalt	4,5	4,2	4,1	4,2	4,1	4,2	4,4	0,81	
Autonomie	4,1	4,0	4,1	4,1	4,0	4,1	3,9	0,17	
Partizipation	4,2	3,6	3,8	3,9	3,8	3,5	3,7	0,93	
Aufgabengeschlossenheit	4,2	3,8	4,0	4,0	3,7	3,4	4,4	2,27	*
Bedeutsamkeit der Aufgabe	4,1	3,4	3,4	3,6	3,9	3,3	4,4	2,14	
Rückmeldung durch die Tätigkeit	4,1	3,6	3,6	3,7	3,7	3,4	4,0	1,03	
Interdependenz (Aufgabe)	4,1	3,5	3,3	3,6	3,4	3,8	2,9	3,63	*
Rückmeldung durch Personen	4,1	3,7	3,5	3,5	3,3	3,6	3,4	1,08	
Soziale Unterstützung	4,2	3,6	3,6	3,4	3,5	3,7	3,7	1,61	
Interdependenz (Ziele)	4,0	3,8	3,6	3,5	4,1	3,6	3,7	0,75	
Qualifikationspotenzial	4,4	3,9	4,3	4,0	4,1	4,2	4,1	1,73	

* p<0,05 (Varianzanalyse)

Anmerkung: Mittelwerte basieren auf siebenstufiger Skala, von 1=geringe Ausprägung bis 7=hohe Ausprägung

Basis: 194 Softwareentwickler (Open-Source- & proprietäre Softwareentwickler)

Tabelle 37: Beta-Gewichte für multiple Regressionen von Tätigkeitsmerkmalen auf organisationalen Kriterien für Open-Source- und proprietären Entwickler

	Allgemeine Arbeitszufriedenheit		Intrinsische Arbeitsmotivation		Organisationale Identifikation (Abteilung)		Job Performance	
	Open-Source	Proprietär	Open-Source	Proprietär	Open-Source	Proprietär	Open-Source	Proprietär
Komplexität der Tätigkeit								
Vielfalt	0,00	0,11	0,16	0,15	0,07	0,10	-0,32**	-0,30*
Autonomie	0,09	0,11	0,25*	-0,12	0,04	0,12	-0,15	-0,22
Partizipation	0,38*	-0,05	-0,21*	0,10	0,09	0,36	0,24*	0,13
Aufgabengeschlossenheit	0,00	0,29**	-0,04	0,18	-0,01	0,03	0,09	-0,02
Rückmeldung durch die Tätigkeit	-0,21	0,03	0,21	0,38**	0,07	-0,14	-0,11	-0,23
Bedeutsamkeit der Aufgabe	-0,06	0,08	0,08	0,10	0,06	0,13*	-0,20*	-0,33**
Qualifikationspotenzial	0,15	0,24	0,23	-0,08	0,20	-0,04	0,09	0,20
Soziale Aspekte								
Rückmeldung durch Personen	0,05	0,11	-0,15	0,18	0,08	0,21	0,04	-0,09
Soziale Unterstützung	0,29**	0,16	0,16	0,13	0,13	0,07	-0,01	0,00
Interdependenz (Aufgabe)	-0,12	0,00	-0,03	-0,10	0,03	0,00	0,05	0,13
Interdependenz (Ziele)	0,06	-0,04	0,25**	0,04	0,18	-0,02	0,13	0,21

* p<0,05 ** p<0,01 *** p<0,001

Beta-Gewichte aus multipler linearer Regression mit jeweiligen organisationalen Kriterien als abhängige Variable und den Tätigkeitsmerkmalen Vielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit, Rückmeldung durch Personen, Soziale Unterstützung, Interdependenz (Aufgabe), Interdependenz (Ziele), Bedeutsamkeit der Aufgabe und Qualifikationspotenzial als unabhängige Variablen (Methode: Einschluß)

Tabelle 37: Beta-Gewichte für multiple Regressionen von Tätigkeitsmerkmalen auf organisationalen Kriterien für Open-Source- und proprietären Entwickler (Fortsetzung)

	Organizational Citizenship Behavior		Organisationale Identifikation (Unternehmen)		Flow-Erleben	
	Open-Source	Proprietär	Open-Source	Proprietär	Open-Source	Proprietär
Komplexität der Tätigkeit						
Vielfalt	0,10	-0,08	0,07	0,10	0,19	0,19
Autonomie	0,07	-0,10	0,04	0,12	0,19	0,07
Partizipation	-0,24*	0,08	0,09	0,36*	-0,23	-0,14
Aufgabengeschlossenheit	0,17	-0,12	-0,01	0,03	-0,12	0,03
Rückmeldung durch die Tätigkeit	0,03	0,09	0,07	-0,14	0,16	0,07
Bedeutsamkeit der Aufgabe	0,12	0,21	0,06	0,13	-0,03	0,08
Qualifikationspotenzial	-0,09	0,08	0,20*	-0,04	-0,04	0,09
Soziale Aspekte						
Rückmeldung durch Personen	0,08	-0,03	0,08	0,21	0,11	-0,08
Soziale Unterstützung	0,20	0,14	0,13	0,07	-0,09	0,11
Interdependenz (Aufgabe)	0,05	0,01	0,03	0,00	0,06	-0,21
Interdependenz (Ziele)	0,32***	0,04	0,18*	-0,02	0,13	0,03

* p<0,05 ** p<0,01 *** p<0,001
Beta-Gewichte aus multipler linearer Regression mit jeweiligen organisationalen Kriterien als abhängige Variable und den Tätigkeitsmerkmalen Vielfalt, Autonomie, Partizipation, Aufgabengeschlossenheit, Rückmeldung durch die Tätigkeit, Rückmeldung durch Personen, Soziale Unterstützung, Interdependenz (Aufgabe), Interdependenz (Ziele), Bedeutsamkeit der Aufgabe und Qualifikationspotenzial als unabhängige Variablen (Methode: Einschluß)
Basis: 137 Open-Source und 73 proprietäre Software-Entwickler

Literaturverzeichnis

Abrahamsson, B. (1977). *Bureaucracy or participation: the logic of organization.* Beverley Hills: Sage.

Ackermann, D. (1986). Untersuchung zum individualisierten Computerdialog: Einfluß des Operativen Abbildsystems auf Handlungs- und Gestaltungsspielraum und die Arbeitseffizienz. In G. Dirlich, C. Freska, U. Schwalto & K. Wimmer (Hrsg.), *Kognitive Aspekte der Mensch-Computer-Interaktion* (S. 95-110). Berlin: Springer.

Adam, F., Feller, J. & Fitzgerald, B. (2003). Libre Software: Implications for Organisations. *Systemes d'Information et Management, 8*(1).

Adolph, S. & Kruchten, P. (2008). *Summary for scrutinizing agile practices or shoot-out at process corral!* Paper presented at the Companion of the 30th international conference on Software engineering, Leipzig, Germany.

Agho, A. O., Price, J. L. & Mueller, C. W. (1992). Discriminant validity of measures of job satisfaction, positive affectivity and negative affectivity. *Journal of Occupational and Organizational Psychology, 65*, 185-196.

Akin-Little, K. A., Eckert, T. L., Lovett, B. J. & Little, S. G. (2004). Extrinsic Reinforcement in the Classroom: Bribery or Best Practice. *School Psychology Review, 33*(3), 344-363.

Aldag, R. J., Barr, S. H. & Brief, A. P. (1981). Measurement of Perceived Task Characteristics. *Psychological Bulletin, 90*(3), 415-431.

Algera, J. A. (1990). Feedback Systems in Organizations. In C. L. Cooper & I. Robertson (Hrsg.), *International Review of Industrial and Organizational Psychology* (Bd. 5, S. 169-193). Chichester: John Wiley and Sons.

Algera, J. A. (1998). Task characteristics. In H. Thierry & P. J. D. Drenth (Hrsg.), *Handbook of work and organizational psychology (2nd ed.)* (Bd. Vol. 3: Personnel psychology, S. 123-139). Hove, England: Psychology Press/Erlbaum (UK) Taylor & Francis.

Allen, N. J. (1987). The role of social and organizational factors in the evaluation of volunteer programs. *Evaluation and Program Planning, 10*, 257-262.

Allen, T. J. (1977). *Managing the Flow of Technology.* Cambridge, MA: MIT Press.

Amabile, T. M., DeJong, W. & Lepper, M. R. (1976). Effects of externally imposed deadlines on subsequent intrinsic motivation. *Journal of Personality and Social Psychology, 34*(1), 92-98.

Ambati, V. & Kishore, S. P. (2004). *How can Academic Software Research and Open Source Software Development help each other?* Paper presented at the ICSE'04 International Conference on Software Engineering - 4th Workshop on Open Source Software Engineering, Edinburgh, Scotland.

Ammons, R. B. (1956). Effects of knowledge of performance: A survey and tentative theoretical formulation. *Journal of General Psychology, 54*, 279-299.

Antoni, C. H. (1999). Konzepte der Mitarbeiterbeteiligung: Delegation und Partizipation. In C. Hoyos & D. Frey (Hrsg.), *Arbeits- und Organisationspsychologie* (S. 569-583). Weinheim: Psychologie Verlags Union.

Antonucci, T. C. & Akiyama, H. (1987). An examination of sex differences in social support among older men and women. *Sex Roles, 17*(11), 737-749.

Apache HTTP Server. (2006, 15. Februar). Verfügbar unter: http://en.wikipedia.org/wiki/Apache_HTTP_Server 2006].

Arnold, H. J. & House, R. J. (1980). Methodological and substantive extensions to the job characteristics model of motivation. *Organizational Behavior and Human Performance, 25*, 161-183.

Aryee, S., Chay, Y. W. & Chew, J. (1996). The Motivation to Mentor among Managerial Employees: An Interactionist Approach. *Group & Organization Management, 21*(3), 261-278.

Asay, M. (2009, 4. Oktober). *Intel claims No. 2 Linux contributor spot as hedge against Microsoft*. Verfügbar unter: http://news.cnet.com/8301-13505_3-10288910-16.html

Ashforth, B. E. & Mael, F. (1989). Social Identity Theory and the Organization. *The Academy of Management Review, 14*(1), 20-39.

Backhaus, K., Erichson, B., Plinke, W. & Weiber, R. (1996). *Multivariate Analysemethoden*. Berlin: Springer.

Baldwin, C. Y. & Clark, K. B. (2003). The Architecture of Cooperation: How Code Architecture Mitigates Free Riding in the Open Source Development Model: Harvard Business School, Working Paper Series, No. 03-209.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavior change. *Psychological Review, 84*, 191-215.

Barker, J. R. & Tompkins, P. K. (1994). Identification in the self-managing organization. Characteristics of Target and Tenure. *Human Communication Research, 21*(2), 223-240.

Barnes, J. (2003). *Open Source Software as a computer-specific organisational Technology*. Verfügbar unter: opensource.mit.edu/papers/barnes.pdf

Barnett, L. & Schwaber, C. E. (2004). *Applying Open Source Processes In Corporate Development Organizations*.

Baronas, A.-M. K. & Louis, M. R. (1988). Restoring a sense of control during implementation: How user involvement leads to system acceptance. *MIS Quarterly, 12*, 111-124.

Baroudi, J. J. (1985). The impact of role variables on IS personnel work attitudes and intentions. *MIS Quarterly, 9*(4), 341-356.

Baroudi, J. J. & Michael, J. G. (1986). Impact of the technological environment on programmer/analyst job outcomes. *Communications of the ACM, 29*(6), 546-555.

Becherer, R. C., Morgan, F. W. & Richard, L. M. (1982). The job characteristics of industrial salespersons: Relationship to motivation and satisfaction. *Journal of Marketing, 46*(4), 125-135.

Beck, K. (2004). *Extreme Programming Explained: Embracing Change* (2nd). Boston: Addison Wesley.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M.et al. (2001). *Manifesto for Agile Software Development*. Verfügbar unter: http://www.agilemanifesto.org/ [09.03.2006 2006].

Behson, S. J., Eddy, E. R. & Lorenzet, S. J. (2000). The importance of the critical psychological states in the job characteristic model: A meta-analytic and structural equations modeling examination. *Current Research in Social Psychology (CRISP), 5*(12).

Bell, B. S. & Kozlowski, S. W. J. (2002). A typology of virtual teams: Implications for effective leadership. *Group and Organization Management, 27*, 14-49.

Benkhoff, B. (1997). Better performance through organisational identification: A test of outcomes and antecedents based on social identity theory. In J. Wickham (Hrsg.), *The search for competitiveness and its implications for employment* (S. 159-179). Dublin: Oak Tree.

Benkler, Y. (2002). Coase's Penguin, or, Linux and the Nature of the Firm. *The Yale Law Journal, 112*(3), 369-446.

Bergmann, B. (2000a). Arbeitsimmanente Kompetenzentwicklung. In B. Bergmann, A. Fritsch, P. Göpfert, F. Richter, B. Wardanjan & S. Wilcek (Hrsg.), *Kompetenzentwicklung und Berufsarbeit* (S. 11-40). Münster: Waxmann.

Bergmann, B. (2000b). Kompetenzentwicklung im Arbeitsprozess. *Zeitschrift für Arbeitswissenschaften, 2*, 138-144.

Berkman, L. F. (1984). Assesssing the physical health effects of social networks and social support. *Annual Review of Public Health, 5*, 413-432.

Berlew, D. E. & Hall, D. T. (1966). The socialization of managers: Effects of expectations on performance. *Administrative Science Quarterly, 11*, 207-223.

Bhuian, S. N. & Menguc, B. (2002). An extension and evaluation of job characteristics, organizational commitment and job satisfaction in an expatriate, guest worker, sales setting. *Journal of Personal Selling & Sales Management, 22*(1), 1-11.

Bierhoff, H. W. (2001). Ehrenamtliche Hilfe. In G. Wenninger (Hrsg.), *Lexikon der Psychologie: In fünf Bänden* (S. 354). Heidelberg: Spektrum Verlag.

Bjerknes, G., Ehn, P. & Kyng, M. (1987). *Computers and Democracy: A Scandinavian Challenge*. Aldershot:: Avebury.

Blackwelder, W. C. (1982). Proving the null hypothesis in clinical trials. *Controlled Clinical Trials, 3*, 345-353.

Blau, P. M. (1964). *Exchange and power in social life*. New York: Wiley.

Bo Xu, B. E. (2006). *Volunteers' participative behaviors in open souce software development: the role of extrinsic incentive, intrinsic motivation and relational social capital.* Texas Tech University.

Boehm, B. W. (1981). *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice Hall.

Boehm, B. W. (1987). Improving software productivity. *Computer, 20*(9), 43-57.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer, 21*(5), 61-72.

Böhler, M. & Schatz, C. (1999). Project Management in Virtual Organizations. In F. Liebl (Hrsg.), *e-conomy. Management und Ökonomie in digitalen Kontexten* (S. 125-170). Marburg: Metropolis-Verlag.

Boland, R. J., Jr. & Tenkasi, R. V. (1995). Perspective making and perspective taking in communities of knowing. *Organization Science, 6*(4), 350-372.

Bollier, D. (1999). *The power of openness: why citizens, education, goverment and business should care about the coming revolution in open source code software*. Verfügbar unter: http://cyber.law.harvard.edu/home/uploads/194/1999-02.pdf [02.08.2004 2004].

Bollinger, T., Nelson, R., Self, K. M. & Turnbull, S. J. (1999). Open-Source Methods: Peering Through the Clutter. *IEEE Software, 16*(4), 8-11.

Bonaccorsi, A. & Rossi, C. (2003). Why open source software can succeed. *Research Policy, 32*(7), 1243-1258.

Bonaccorsi, A. & Rossi, C. (2004). Altruistic individuals, selfish firms? The structure of motivation in Open Source software. *First Monday, 9*(1).

Bortz, J. (1999). *Statistik für Sozialwissenschaftler* (5. vollst. üb. und akt. Aufl.). Berlin: Springer.

Bosco, G. (2004). *Implicit theories of "good leadership" in the open-source community.* Master Thesis, Technical University of Denmark, Kgs. Lyngby.

Bostrom, R. P. (1978). *Conflict-Handling and Power in the Redesign Process: A Field studv Investimtion of the Relationship between MIS Users and Svstem Maintenance Personnel.* UnpubIished Doctoral Dissertation, University of Minnesota.

Bowker, D. (1999). *Constructing the client-computer interface: Guidelines for design and implementation of web-based surveys*. Pullman, WA: Washington State University, Social & Economic Sciences Research Center.

Bredenkamp, J. (1980). *Theorie und Planung psychologischer Experimente*. Darmstadt: Steinkopff.

Brodbeck, F. C. (1994a). Intensive Kommunikation lohnt sich für SE-Projekte. In F. C. Brodbeck & M. Frese (Hrsg.), *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung* (S. 51-67). München: Oldenbourg.

Brodbeck, F. C. (1994b). Software-Entwicklung: Ein Tätigkeitsspektrum mit vielfältigen Kommunikations- und Lernanforderungen. In F. C. Brodbeck & M. Frese (Hrsg.), *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung* (S. 13-35). München: Oldenbourg.

Brodbeck, F. C. (2001). Communication and performance in software development projects. *European Journal of Work and Organizational Psychology, 10*(1), 73-94.

Brodbeck, F. C. & Frese, M. (Hrsg.). (1994). *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung.* . München: Oldenbourg.

Brooks, F. P. (1982). *The Mythical Man-Month, Essays on Software Engineering*: Reading.

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer, 20*(4), 10-19.

Brown, J. S. & Duguid, P. (1991). Organizational Learning and Communities of Practice: Towards a Unified view of Working, Learning and Innovation. *Organization Science, 2*(1), 40-57.

Brown, J. S. & Duguid, P. (1998). Organizing knowledge. *California Management Review, 40*(3), 90-111.

Brown, M. E. (1969). Identification and Some Conditions of Organizational Involvement. *Administrative Science Quarterly, 14*(3), 346-355.

Brown, R., Condor, S., Mathews, A., Wade, G. & Williams, J. (1986). Explaining intergroup differentiation in an industrial organization. *Journal of Occupational psychology, 59*(4), 273-286.

Brown, S. L. & Eisenhardt, K. M. (1995). Product Development: Past Research, Present Findings, and Future Directions. *The Academy of Management Review, 20*(2), 343-378.

Bullis, C. A. & Tompkins, P. K. (1989). The Forest Ranger Revisited: A Study of Control Practices and Identification. *Communication Monographs, 56*(4), 287-306.

Bungard, W. & Antoni, C. H. (1995). Gruppenorientierte Interventionstechniken. In H. Schuler (Hrsg.), *Lehrbuch Organisationspsychologie*. Bern: Huber.

Burke, K. (1969). *A Rhetoric of Motives*. Berkeley: University of California Press.

Burkolter, D. (2005). *Tätigkeitsmerkmale der Freiwilligenarbeit und ihr Zusammenhang mit intrinsischer Motivation sowie Arbeitszufriedenheit.* Lizentiatsarbeit, Universität Zürich.

Burkolter, D., Güntert, S. & Wehner, T. (2006). *Tätigkeitsmerkmale der Freiwilligenarbeit und ihr Zusammenhang mit Arbeitszufriedenheit.* Paper presented at the 45. Kongress der Deutschen Gesellschaft für Psychologie, Nürnberg.

Burn, J. M., Ma, L. C. K. & Ng Tye, E. M. W. (1995). Managing IT professionals in a global environment. *SIGCPR Comput. Pers., 16*(3), 11-19.

Burroughs, R. E. (1991). *Improving programmer job-based feedback: results from a longitudinal field experiment.* Paper presented at the Special Interest Group on Computer Personnel Research Annual Conference. Proceedings of the 1991 conference on SIGCPR, Athens, Georgia, United States.

Burroughs, S. M. & Eby, L. T. (1998). Psychological sense of community at work: A measurement system and explanatory framework. *Journal of Community Psychology, 26*(6), 509-532.

Burt, R. (1987). A Note on Strangers, Friends, and Happiness. *Social Networks, 9*, 311-331.

Burt, R. S. (1983). Range. In R. S. Burt & M. J. Minor (Hrsg.), *Applied Network Analysis* (S. 176-194). Beverly Hills, CA: Sage.

Butler, B. S. (2001). Membership Size, Communication Activity, and Sustainability: A Resource-Based Model of Online Social Structures. *Information Systems Research, 12*(4), 346-362.

Caban, A., Cimino, C., Swencionis, C., Ginsberg, M. & Wylie-Rosett, J. (2001). Estimating software development costs for a patient multimedia education project. *Journal of American Medical Informatics Association, 8*(2), 185-188.

Cameron, J., Banko, K. M. & Pierce, W. D. (2001). Pervasive Negative Effects of Rewards on Intrinsic Motivation: The Myth Continues. *The Behavior Analyst, 24*(1), 1-44.

Campbell, D. J. (1988). Task complexity: A review and analysis. *Academy of Management Review, 13*, 40–52.

Campbell, R. J. & Gingrich, K. F. (1986). The interactive effects of task complexity and participation on task performance: a field experiment. *Organizational behavior and human decision processes, 38*(2), 162-180.

Campion, M. A. (1988). Interdisciplinary approaches to job design: A constructive replication with extensions. *Journal of Applied Psychology, 73*(3), 467-481.

Campion, M. A., Medsker, G. J. & Higgs, A. C. (1993). Relations between work group characteristics and effectiveness: Implications for designing effective work groups. *Personnel Psychology, 46*(4), 823-850.

Campion, M. A., Papper, E. M. & Medsker, G. J. (1996). Relations between work team characteristics and effectiveness: A replication and extension. *Personnel psychology, 49*, 429-452.

Campion, M. A. & Thayer, P. W. (1985). Development and field evaluation of an interdisciplinary measure of job design. *Journal of Applied Psychology, 70*(1), 29-43.

Carayon, P., Hoonakker, P., Marchand, S. & Schwarz, J. (2003). *Job characteristics and quality of working life in the IT workforce: the role of gender.* Paper presented at the Special Interest Group on Computer Personnel Research Annual Conference. Proceedings of the 2003 SIGMIS conference on Computer personnel research, Philadelphia, Pennsylvania.

Cardona, P., Lawrence, B. S. & Bentler, P. M. (2004). The Influence of Social and Work Exchange Relationships on Organizational Citizenship Behavior. *Group and Organization Management, 29*(2), 219-247.

Carmel, E. (1999). *Global software teams: collaborating across borders and time zones.* Upper Saddle River, NJ: Prentice-Hall.

Chelf, B. (2006). *Measuring software quality: A Study of Open Source Software*: Coverity, Inc., 185 Berry St. Suite 3600, San Francisco, CA 94107, USA.

Chen, H., Wigand, R. T. & Nilan, M. S. (1999). Optimal experience of Web activities. *Computers in Human Behavior, 15*(5), 585-608.

Cheney, G. (1983). On the various and changing meanings of organizational membership: Field study of organizational identification. *Communication Monographs, 50*, 342-362.

Cherrington, D. J. & England, J. L. (1980). The desire for an enriched job as a moderator of the enrichment-satisfaction relationship. *Organizational Behavior and Human Performance, 25*, 139-159.

Chow, G. (1960). Tests of Equality Between Sets of Coefficients in Two Linear Regressions. *Econometrica, 28*(3), 591-605.

Cockburn, A. & Highsmith, J. (2001). Agile Software Development: The People Factor. *IEEE Computer, 34*(11), 131-133.

Cohen, A. (1992). Antecedents of organizational commitment across occupational groups: A meta-analysis. *Journal of Organizational Behavior, 13*(6), 539-558.

Cohen, J. (1969). *Statistical Power Analysis for the Behavior Science*. Hillsdale, NJ: Erlbaum.

Computerprogramm. (2009, 11. Oktober). Verfügbar unter: http://de.wikipedia.org/wiki/Computerprogramm 2009].

Constant, D., Sproull, L. & Kiesler, S. (1996). The kindness of strangers: The usefulness of electronic weak ties for technical advice. *Organization Science, 7*(2), 119-135.

Cooney, M., A. (2003). *The Effects of Telecommuting on Worker Profile.* Master Thesis, B.A., Rutgers, The State University of New Jersey.

Copeland, L. (2001, 18. Oktober 2009). *Extreme Programming*. Verfügbar unter: http://www.computerworld.com/s/article/66192/Extreme_Programming?taxonomyId=63&pageNumber=2

Corbett, J. M., Martin, R., Wall, T. D. & Clegg, C. W. (1989). Technological coupling as a predictor of intrinsic job satisfaction: A replication study. *Journal of Organizational Behavior, 10*, 91-95.

Cordery, J. L., Mueller, W. S. & Smith, L. M. (1991). Attitudinal and Behavioral Effects of Autonomous Group Working: A Longitudinal Field Study. *Academy of Management Journal, 34*(2), 464-476.

Cordery, J. L. & Sevastos, P. P. (1993). Responses to the original and revised Job Diagnostic Survey: Is education a factor in responses to negatively worded items? *Journal of Applied Psychology, 78*(1), 141-143.

Couger, J. D. (1986a). Effect of Cultural Differences on Motivation of Analysts and Programmers: Singapore vs. the United States. *MIS Quarterly, 10*(2), 188-197.

Couger, J. D. (1986b). Motivation Norms of Knowledge Engineers Compared to Those of Software Engineers. *Journal of Management Information Systems, 4*(3), 82-94.

Couger, J. D., Borovits, I. & Zviran, M. (1989). *Comparison of Motivating Environments for Programmer/Analysts and Programmers in the U. S., Israel, and Singapore.* Paper presented at the International Conference on System Sciences, Hawaii.

Couger, J. D. & Zawacki, R. A. (1978). What motivates DP professionals? *Datamation, 24*(9), 116-123.

Couger, J. D. & Zawacki, R. A. (1980). *Motivating and Managing Computer Personnel.* New York: Wiley.

Cramton, C. D. (2001). The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. *Organization Science, 12*(3), 346-371.

Crowston, K., Annabi, H., Howison, J. & Masango, C. (2004). *Effective work practices for software engineering: free/libre open source software development.* Paper presented at the Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research, Newport Beach, CA, USA.

Crowston, K. & Howison, J. (2004). *The social structure of Free and Open Source software development. Syracuse FLOSS research working paper*. Verfügbar unter: http://opensource.mit.edu/papers/crowstonhowison.pdf [08.03.2005

Crowston, K. & Scozzi, B. (2002a). *Exploring the Strengths and Limits of Open Source Software Engineering Processes: A Research Agenda.* Paper presented at the

ICSE'02 International Conference on Software Engineering - 2nd Workshop on Open Source Software Engineering, Orlando, FL, USA.

Crowston, K. & Scozzi, B. (2002b). Open source software projects as virtual organizations: Competency rallying for software development. *IEE Proceedings Software, 149*(1), 3-17.

Crowston, K., Scozzi, B. & Buonocore, S. (2002). *An exploratory study of Open Source Software development team structure*. Verfügbar unter: http://crowston.syr.edu/floss/ecis2003.pdf [19.01.2004 2004].

Csikszentmihalyi, M. (1982). Towards a psychology of optimal experience. In L. Wheeler (Hrsg.), *Annual review of personality and social psychology* (Bd. 3, S. 13-36). Beverly Hills, CA: Sage.

Csikszentmihalyi, M. & Csikszentmihalyi, I. S. (1991). *Die außergewöhnliche Erfahrung im Alltag*. Stuttgart: Klett-Cotta.

Csikszentmihalyi, M. & Larson, R. (1987). Validity and reliability of the Experience-Sampling Method. *Journal of Nervous and Mental Disease, 175*(9), 526-536.

Cubranic, D. (1999a). *Coordinating Open Source Software development.* Paper presented at the 7th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, CA, USA.

Cubranic, D. (1999b). *Open-source software development.* Paper presented at the 2nd Workshop on Software Engineering over the Internet, Los Angeles.

Culnan, M. J. & Markus, M. L. (1987). Information technologies. In F. M. Jablin, L. L. Putnam & L. W. Porter (Hrsg.), *Handbook of Organizational Communication: An Interdisciplinary Perspective*. Newbury Park, CA: Sage.

Cummings, J., Kiesler, S. B. & Sproull, L. (2002a). Beyond Hearing: Where Real-World and Online Support Meet. *Group Dynamics, 6*(1), 78-88.

Cummings, J. N., Butler, B. & Kraut, R. (2002b). The Quality of Online Social Relationships. *Communications of the ACM, 45*, 103-108.

Cummings, T. G. & Blumberg, M. (1987). Advanced manufacturing technology and work design. In T. D. Wall, C. W. Clegg & N. J. Kemp (Hrsg.), *The Human Side of Advanced Manufacturing Technology*. Chichester, Sussex: Wiley.

Curtis, B., Krasner, H. & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31*(11), 1268-1287.

Cusumano, M., MacCormack, A., Kemerer, C. F. & Crandall, B. (2003). Software Development Worldwide: The State of the Practice. *IEEE Software, 20*(6), 28-34.

Dafermos, G., N. (2001). *Management & Virtual Decentralised Networks: The Linux Project.* Master, Durham Business School.

Daft, R. L. & Lengel, R. H. (1984). Information Richness: A new approach to managerial behavior and organization design. In B. Staw & L. L. Cummings (Hrsg.), *Research in Organizational Behavior* (Bd. Vol. 6, S. 191-233). Greenwich: JAI.

Daft, R. L. & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Management Science, 32*(5), 554-571.

Davenport, T. H. & Prusak, L. (1998). *Working Knowledge*. Boston: Harvard Business School Press.

David, P. M., Waterman, A. & Arora, S. (2003). *FLOSS-US. The Free/Libre/Open Source Software Survey*. Verfügbar unter: http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf [06.08.2004 2004].

Davis, A. M., Bersoff, E. H. & Cromer, E. R. (1988). A Strategy for Comparing Alternative Software Development Life Cycle Models. *IEEE Transactions on Software Engineering, 14*(10), 1453-1461.

Davison, K., Pennebaker, J. & Dickerson, S. (2000). Who talks? The social psychology of illness support groups. *American Psychologist, 55*, 205–217.

Dawes, R. M. (1980). Social dilemmas. *Annual Review of Psychology, 31*, 169-193.

Deci, E. L. (1971). Effects of externally mediated rewards on intrinsic motivation. *Journal of personality and social psychology, 22*, 105 - 115.

Deci, E. L. (1975). *Intrinsic Motivation*. New York: Plenum Press.

Deci, E. L., Koestner, R. & Ryan, R. M. (1999). A Meta-Analytic Review of Experiments Examining the Effects of Extrinsic Rewards on Intrinsic Motivation. *Psychological Bulletin, 125*(6), 627-668.

Deci, E. L. & Ryan, R. M. (1993). Die Selbstbestimmungstheorie der Motivation und ihre Bedeutung für die Pädagogik. *Zeitschrift für Pädagogik, 2*, 223 - 238.

DeJonge, J. & Schaufeli, W. B. (1998). Job characteristics and employee well-being: a test of Warr's Vitamin Model in health care workers using structural equation modelling. *Journal of Organizational Behavior, 19*(4), 387-407.

Demil, B. & Lecocq, X. (2003). *Neither Market nor Hierarchy or Network: The Emerging Bazaar Governance*. [PDF]. Verfügbar unter: http://opensource.mit.edu/papers/demillecocq.pdf [19.01.2004 2004].

Dempsey, B. J., Weiss, D., Jones, P. & Greenberg, J. (1999). *A Quantitative Profile of a Community of Open Source Linux Developers*: School of Information and Library Science, University of North Carolina at Chapel Hill.

Deng, J., Seifert, T. & Vogel, S. (2003). *Towards a Product Model of Open Source Software in a Commercial Environment.* Paper presented at the ICSE'03 International Conference on Software Engineering - 3rd Workshop on Open Source Software Engineering, Portland, Oregan.

DeNisi, A. S. & Kluger, A. N. (2000). Feedback effectiveness: Can 360-degree appraisals be improved. *Academy of Management Executive, 14*(1), 129-139.

Diamantopoulos, A. & Winklhofer, H. M. (2001). Index construction with formative indicators: an alternative to scale development. *Journal of Marketing Research, 38*(2), 269-277.

Dietze, S. (2003a). *Improvement Opportunities for the Open Source*

Software Development Approach and How to utilize them. Paper presented at the OSSIE 2003 - 1st Workshop on Open Source Software in an Industrial Environment, Erfurt, Deutschland.

Dietze, S. (2003b). *Quality Assurance, Project Maintenance and Requirements Documentation in Open Source Software Development Processes.* Paper presented at the Fourth International Conference on eXtreme Programming and Agile Processes

in Software Engineering 2003-Workshop: Making Free/Open Source Software Development Work Better, Genova, Italy.

Dillman, D. A. (1991). The design and administration of mail surveys. *Annual Review of Sociology, 17*, 225-249.

Dillman, D. A. (2007). *Mail and Internet Surveys: The Tailored Design Method* (2). Chinchester: Wiley.

Dodd, N. G. & Ganster, D. C. (1996). The interactive effects of variety, autonomy, and feedback on attitudes and performance. *Journal of Organizational Behavior, 17*(4), 329-347.

Donaldson, L. & Warner, M. (1979). Bureaucratic and democratic structure and occupational interest associations. In D. S. Pugh & C. R. Hinings (Hrsg.), *Organizational structure: extensions and replications* (S. 67-86). Westmead: Saxon House.

Döring, N. (2000a). Identitäten, soziale Beziehungen und Gemeinschaften im Internet. In B. Batinic (Hrsg.), *Internet für Psychologen* (2nd ed., S. 379-415). Göttingen: Hogrefe.

Döring, N. (2000b). Kommunikation im Internet: Neun theoretische Ansätze. In B. Batinic (Hrsg.), *Internet für Psychologen* (2nd ed., S. 345-377). Göttingen: Hogrefe.

Drasgow, F. & Miller, H. E. (1982). Psychometric and substantive issues in scale construction and validation. *Journal of Applied Psychology, 67*, 268—279.

Dunckel, H. (1999a). Psychologische Arbeitsanalyse: Verfahrensüberblick und Auswahlkriterien. In H. Dunckel (Hrsg.), *Handbuch psychologischer Arbeitsanalyseverfahren* (S. 9-30). Zürich: vdf-Hochschulverlag.

Dunckel, H. (Hrsg.). (1999b). *Handbuch psychologischer Arbeitsanalyseverfahren.* Zürich: vdf-Hochschulverlag.

Dunham, P. J., Hurshman, A., Litwin, E., Gusella, J., Ellsworth, C. & Dodd, P. W. D. (1998). Computer-Mediated Social Support: Single Young Mothers as a Model System. *American Journal of Community Psychology, 26*(2), 281-306.

Dunham, R. B. (1976). The Measurement and Dimensionality of Job Characteristics. *Journal of Applied Psychology, 4*, 404-409.

Dunham, R. B., Aldag, R. J. & Brief, A. P. (1977). Dimensionality of Task Design as Measured by the Job Diagnostic Survey. *Academy of Management Journal, 20*(2), 209-223.

Edwards, J. R., Scully, J. A. & Brtek, M. D. (1999). The measurement of work: Hierarchical representation of the multimethod job design questionnaire. *Personnel psychology, 52*(2), 305-334.

Eisenberger, R., Pierce, W. D. & Cameron, J. (1999). Effects of reward on intrinsic motivation-Negative, neutral, and positive: Comment on Deci, Koestner, and Ryan (1999). *Psychological Bulletin, 125*, 677-691.

Eklund, S., Feldman, M., Trombley, M. & Sinha, R. (2002). *Improving the Usability of Open Source Software: Usability Testing of StarOffice Calc.* Paper presented at the Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems (CHI 2002), Minneapolis.

Elliott, M. S. & Scacchi, W. (2002). *Communicating and Mitigating Conflict in Open Source Software Development Projects*. Verfügbar unter: http://www.ics.uci.edu/~melliott/commossd.pdf

Elliott, M. S. & Scacchi, W. (2004). Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture. In S. Koch (Hrsg.), *Free/Open Source Software Development*: IDEA Publishing.

Emery, F. E. (1959). *Characteristics of socio-technical systems (Document No. 527)*. London: Tavistock Institute of Human Relations.

Erenkrantz, J. R. (2003). *Release Management Within Open Source Projects*. Paper presented at the ICSE'03 International Conference on Software Engineering - 3rd Workshop on Open Source

Software Engineering, Portland, Oregan.

Espinosa, J. A., Kraut, R. E., Lerch, J. F., Slaughter, S. A., Herbsleb, J. D. & Mockus, A. (2001). *Shared mental models and coordination in large-scale, distributed software development*. Paper presented at the Twenty-Second International Conference on Information Systems.

Espinosa, J. A., Kraut, R. E., Slaughter, S. A., Lerch, J. F., Herbsleb, J. D. & Mockus, A. (2002). *Shared Mental Models, Familiarity, and Coordination: A Multi-Method Study of Distributed Software Teams*. Paper presented at the International Conference on Information Systems, Barcelona.

Espinosa, J. G. & Zimbalist, A. S. (1978). *Economic democracy: workers' participation in Chilean industry 1970-1973*. New York: Academic Press.

Ettrich, M. (2004). Koordination und Kommunikation in Open-Source-

Projekten. In R. A. Gehring & B. Lutterbeck (Hrsg.), *Open Source Jahrbuch 2004* (S. 179-192). Berlin: Lehmanns Media.

Etzioni, A. (1971). The fallacy of decentralization. In T. E. Cook & M. P. M. (Hrsg.), *Participatory democracy* (S. 63-68). San Francisco: Canfield Press.

Eunice, J. (1998, May 11, 1998). *Beyond the Cathedral, Beyond the Bazaar*. [HTML]. Verfügbar unter: http://www.illuminata.com/cgi-local/pub.cgi?docid=cathedral&section=cathedral1 [20.04.2005 2005].

Eysenbach, G., Powell, J., Englesakis, M., Rizo, C. & Stern, A. (2004). Health related virtual communities and electronic support groups: systematic review of the effects of online peer to peer interactions. *British Medical Journal, 328*, 1166-1170.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal, 15*(3), 182-211.

Fairhurst, G. T. (2004). Dualisms in Leadership Research. In L. L. Putnam & F. M. Jablin (Hrsg.), *The New Handbook of Organizational Communication*. Thousand Oaks: Sage.

Fan, X., Thompson, B. & Wang, L. (1999). Effects of sample size, estimation method, and model specification on structural equation modeling fit indexes. *Structural Equation Modeling, 6*, 56-83.

Faraj, S. & Wasko, M. M. (2001). *The Web of Knowledge: An Investigation of Knowledge Exchange in Networks of Practice*. Tallahassee, FL: Florida State University.

Farh, J. L., Podsakoff, P. M. & Organ, D. W. (1990). Accounting for Organizational Citizenship Behavior: Leader Fairness and Task Scope versus Satisfaction. *Journal of Management, 16*(4), 705-721.

Farh, J. L. & Scott, W. E., Jr. (1983). The experimental effects of "autonomy" on performance and self-reports of satisfaction. *Organization Behavior and Human Performance, 31*(2), 203-222.

Feller, J. & Fitzgerald, B. (2000). *A Framework Analysis of the Open Source Software Development Paradigm.* Paper presented at the Proceedings of the 21st Annual International Conference on Information Systems (ICIS 2000), Brisbane, Australia.

Feller, J. & Fitzgerald, B. (2002). *Understanding Open Source Software Development.* London: Addison-Wesley.

Ferratt, T. W. & Short, L. E. (1986). Are Information Systems People Different: An Investigation of Motivational Differences. *MIS Quarterly, 10*(4), 377-387.

Ferris, G. R., Fedor, D. B., Rowland, K. M. & Porac, J. F. (1985). Social influence and sex effects on task performance and task perception. *Journal of Vocational Behavior, 26*, 66-76.

Festinger, L. (1957). *A theory ofcognitive dissonance.* New York: Harper and Row.

Fink, M. (2002). *The Business and Economics of Linux and Open Source*: Prentice Hall PTR.

Fischer, J. L., Sollie, D. L., Sorell, G. T. & Green, S. (1989). Marital status and career stage influences on social networks of young adults. *Journal of Marriage and the Family, 51*, 521-534.

Fitz-enz, J. (1978). Who is the Data Processing Professional? *Datamation, 24*(9), 125-128.

Fox, S. & Feldman, G. (1988). Attention state and critical psychological states as mediators between job dimensions and outcomes. *Human Relations, 41*, 229-245.

Franck, E. & Jungwirth, C. (2002a). Das Open-Source-Phänomen jenseits des Gift-Society-Mythos. *Wirtschaftswissenschaftliches Studium, 31*, 124-129.

Franck, E. & Jungwirth, C. (2002b). *Reconciling investors and donators: The governance structure of open source.* Zürich: Universität Zürich.

Franke, N. & Hippel, E. v. (2003). Satisfying Heterogenous User Needs via Innovation Toolkits: The Case of Apache Security Software. *Research Policy, 32*(7), 1199-1215.

The Free Software Definition. (2009). Verfügbar unter: http://www.gnu.org/philosophy/free-sw.html [29.09.2009

Freedman, D. P. & Weinberg, G. M. (1990). Procedures and checklists for conducting reviews. In (3. ed.). New York: Dorset House.

Frese, M. (1989). Theoretical models of control and health. In S. L. Sauter, J. J. Hurrel & C. L. Cooper (Hrsg.), *Job control an worker health.* New York: John Wiley.

Frese, M., & Zapf, D. (1994). Action as the core of work psychology: A German approach. In H. C. Triandis, M. D. Dunnette & L. M. Hough (Hrsg.), *Handbook of industrial and organizational psychology* (S. 271-340). Palo Alto, CA: Consulting Psychologists.

Frese, M. & Altmann, A. (1989). The treatment of errors in learning and training. In L. Bainbridge & S. A. Ruiz Quintanilla (Hrsg.), *Developing skills with information technology* (S. 65-86). New York: Wiley.

Frese, M. & Hesse, W. (1993). The work situation in software development. Results of an empirical study. *Software Engineering Notes, 18*(3), 65-72.

Fried, Y. (1991). Meta-analytic comparison of the JDS and Job Characteristics Inventory as Correlates for Work Satisfaction and Performance. *Journal of Applied Psychology, 76*(5), 690-697.

Fried, Y. & Ferris, G. R. (1986). The dimensionality of job characteristics: Some neglected issues. *Journal of Applied Psychology, 71*, 419 - 426.

Fried, Y. & Ferris, G. R. (1987). The Validity of the Job-Characteristics Model: A Review and Meta-Analysis. *Personnel Psychology, 40*, 287-322.

Fujigaki, Y. (1990). A study of mental workload of software engineers. In L. Berlinguer & D. Bertheietre (Hrsg.), *Work with Display Units 89* (S. 395-402). Amsterdam: Elsevier.

Fulk, J., Schmilz, J. & Steinfield, C. W. (1990). A Social Influence Model of Technology Use (A. 6934, Trans.). In J. Fulk & C. W. Steinfield (Hrsg.), *Organizations and Communication Technology* (S. 117-140). Newbury Park: Sage.

Gail, K. & Frese, M. (1994). Positive Gefühle in der Arbeit. In F. C. Brodbeck & M. Frese (Hrsg.), *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung* (S. 87-101). München: Oldenbourg.

Galegher, J., Sproull, L. & Kiesler, S. (1998). Legitimacy, authority, and community in electronic support groups. *Written Communication, 15*(4), 493-530.

Ganzach, Y. (1998). Intelligence and Job Satisfaction. *The Academy of Management Journal, 41*(5), 526-539.

Garzarelli, G. (2002, 06.04.2003). *Open Source Software and the Economics of Organization.* [PDF]. Verfügbar unter: [06.04.2003 2003].

Gensicke, T., Picot, S. & Geiss, S. (2005). *Freiwilliges Engagement in Deutschland 1999-2004.* München: Bundesministeriums für Familie, Senioren, Frauen und Jugend.

Gerlmaier, A. (2002). *Anforderungen, Belastungen und Ressourcenpotenziale von Freelancern und Intrapreneuren im IT-Bereich: Ergebnisse einer empirischen Studie. Vorläufiger Zwischenbericht des NestO-Teilprojektes "Ressourcen- und Kompetenzmanagement" für den IT-Bereich*: Universität Dortmund.

Ghani, J. A. & Desphande, S. P. (1994). Task characteristics and the experience of optimal flow in human-computer interaction. *Journal of Psychology, 128*(4), 381-391.

Ghosh, R. A. (2002). *Free/Libre and Open Source Software: Survey and Study. FLOSS. Workshop on Advancing the Research on Free/Open Source Software.* Brussels: European Commission.

Ghosh, R. A., Glott, R., Krieger, B. & Robles, G. (2002). *Free/Libre and Open Source Software: Survey and Study - FLOSS - Part 4: Survey of Developers.* Maastricht, The Netherlands: International Institute of Infonomics, University of Maastricht, The Netherlands.

Gladstein, D. L. (1984). Groups in context: A model of task group effectiveness. *Administrative Science Quarterly, 29*, 499-517.

Glass, R. (1997). *Software runaways. Lessons learned from massive software project failures*. Upper Saddle River, NJ: Prentice Hall.

Glass, R. L., Vessey, I. & Conger, S. A. (1992). Software tasks: Intellectual or clerical. *Information and Management, 23*, 183-191.

Glew, D. J., O'Leary-Kelly, A. M., Griffin, R. W. & Van Fleet, D. D. (1995). Participation in Organizations: A Review of the Issues and Proposed Framework for Future Analysis. *Journal of Management, 21*(3), 395-421.

Glick, W. H., Jenkins Jr, G. D. & Gupta, N. (1986). Method versus Substance: How Strong Are Underlying Relationships between Job Characteristics and Attitudinal Outcomes? *Academy of Management Journal, 29*(3), 441-464.

Goldstein, D. K. (1982). *The effects of structured development methods on the job satisfaction of programmer/analysts: A theoretical model. Working Paper CISF-90. Alfred P. Sloan School of Management, Massachusetts Institute of Technology. Cambridge, Mass.*

Goldstein, D. K. (1989). The effects of task differences on the work satisfaction, job characteristics, and role perceptions of programmers/analysts. *Journal of Management Information Systems, 6*(1), 41-58.

Goldstein, D. K. & Rockart, J. F. (1984). An Examination of Work-Related Correlates of Job Satisfaction in Programmer/Analysts. *MIS Quarterly, 8*(2), 103-115.

González-Barahona, J. M. & Robles, G. (2003). Free Software Engineering: A Field to Explore. *Upgrade - The European Journal of the Informatics Professional, 4*(4), 49-54.

Goodman, P. S. (1979). *Assessing organizational change: The Rushton quality of work experiment*. New York: John Wiley.

Goodman, P. S., Devadas, R. & Griffith-Hughson, T. L. (1988). Groups and productivity: Analysing the effectiveness of self-managing teams. In J. P. Campbell & R. J. Campbell (Hrsg.), *Productivity in Organizations* (S. 295-327). San Francisco: Jossey-Bass.

Görling, S. (2003). *A critical approach to Open Source software*. Verfügbar unter: http://opensource.mit.edu/papers/gorling.pdf [08.03.2005

Gorn, G. J. & Kanungo, R. N. (1980). Job involvement and motivation: Are intrinsically motivated managers more job involved. *Organizational Behavior and Human Performance, 26*, 277-265.

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology, 91*, 481-510.

Granovetter, M. S. (1982). The strength of weak ties: A network theory revisisted. In P. V. Marsden & N. Lin (Hrsg.), *Social Structure and Network Analysis* (S. 105-130). New York: John Wiley and Sons.

Green, S. B., Armenakis, A. A., Marbert, L. D. & Bedeian, A. G. (1979). An Evaluation of the Response Format and Scale Structure of the Job Diagnostic Survey. *Human Relations, 32*(2), 181-188.

Griffeth, R. W. (1985). Moderation of the effects of job enrichment by participation: A longitudinal field experiment. *Organizational Behavior and Human Decision Processes, 35*(1), 73-93.

Griffin, R. W. (1981). A longitudinal investigation of task characteristics relationships. *Academy of Management Journal, 24*, 99-113.

Griffin, R. W. (1983). Objective and social sources of information in task redesign: A field experiment. *Administrative Science Quarterly, 28*(2), 184-200.

Griffin, R. W., Bateman, T. S., Wayne, S. J. & Head, T. C. (1987). Objective and social factors as determinants of task perceptions and responses: An integrated perspective and empirical investigation. *Academy of Management Journal, 30*(501-523).

Griffin, R. W., Moorhead, G., Johnson, B. H. & Chonko, L. B. (1980). The empirical dimensionality of the Job Characteristic Inventory. *Academy of Management Journal, 23*(4), 772-777.

Griffin, R. W., Welsh, A. & Moorhead, G. (1981). Perceived Task Characteristics and Employee Performance: A Literature Review. *The Academy of Management Review, 6*(4), 655-664.

Griffith, T. L. & Northcraft, G. B. (1993). Promises, pitfalls, and pardox: Cognitive elements in the implementation of new technology. *Journal of Managerial Issues, 5*(465-482).

Griffith, T. L. & Northcraft, G. B. (1996). Cognitive Elements in the implementation of new technology: Can less information provide more benefits? *MIS Quarterly, 20*, 99-110.

Grinter, R. E., Herbsleb, J. D. & D., P. (1999). *The geography of coordination: dealing with distance in R&D work.* Paper presented at the International ACM SIGGROUP Conference on Supporting Group Work, New York.

Güntert, S. T. & Wehner, T. (2005). *Wie motiviert ist frei-gemeinnützige Arbeit?* Paper presented at the Gesellschaft für Arbeitswissenschaft (2005). Personalmanagement und Arbeitsgestaltung. Bericht zum 51. Kongress der Gesellschaft für Arbeitswissenschaft, Heidelberg.

Guzzo, R. A., Jette, R. D. & Katzell, R. A. (1985). The effects of psychologically based intervention programs on worker productivity: A meta analysis. *Personnel psychology, 38*, 275-292.

Hacker, W. (1986). *Arbeitspsychologie. Psychische Regulation von Arbeitstätigkeiten.* Bern: Huber.

Hacker, W. & Iwanowa, A. (1984). Zur psychologischen Bewertung von Tätigkeitsmerkmalen. *Zeitschrift für Psychologie, 192*(2), 103-121.

Hacker, W. & Schoenfelder, E. (1986). Analyse und Bewertung von Arbeitstaetigkeiten mit Bildschirmtechnik. *Psychologie für die Praxis, Ergaenzungsheft*(34-41).

Hackman, J. R. (1969). Nature of the task as a determiner of job behavior. *Personnel Psychology, 22*, 435-444.

Hackman, J. R. (1970). Tasks and tasks performance in research on stress. In J. E. McGrath (Hrsg.), *Social and phsycological factors in stress* (S. 202-237). New York: Holt, Rinehart & Winston.

Hackman, J. R. & Lawler, E. E. (1971). Employee reactions to job characteristics. *Journal of applied psychology, 3*, 259-286.

Hackman, J. R. & Oldham, G. R. (1974). *The Job diagnostic survey: an instrument for the diagnostic of jobs and the evaluation of job redesign projects*. New Haven: Yale University.

Hackman, J. R. & Oldham, G. R. (1975). Development of the Job Diagnostic Survey. *Journal of Applied Psychology, 60*(2), 159-170.

Hackman, J. R. & Oldham, G. R. (1976). Motivation through the Design of Work: Test of a Theory. *Organizational Behavior and Human Performance, 16*, 250-279.

Hackman, J. R. & Oldham, G. R. (1980). *Work redesign*. London: Addison-Wesley Publishing Company.

Hage, J. & Aiken, M. (1967). Relationship of centralization to other structural properties. *Administrative Science Quarterly, 12*, 72-92.

Hager, W. (2000). About some misconceptions and the discontent with statistical tests in psychology. *Methods of Psychological Research Online, 5*(1).

Haim, A. B. (2002). *Participation Programs in Work Organizations: Past, Present, and Scenarios for the Future*. Westport, Conn: Quorum Books.

Hall, D. T. & Schneider, B. (1972). Correlates of Organizational Identification as a Function of Career Pattern and Organizational Type. *Administrative Science Quarterly, 17*(3), 340-350.

Han, S. (1988). The relationship between life satisfaction and flow in elderly Korean immigrants. In M. Csikszentmihalyi & I.-S. Csikszentmihalyi (Hrsg.), *Optimal experience: Psychological studies of flow in consciousness* (S. 138-149). New York: Cambridge University Press.

Hardin, G. (1968). The tragedy of the commons. *Science, 162*(3859), 1243-1248.

Harrington, T. F. (1995). *Assessment of Abilities*: ERIC Clearinghouse on Counseling and Student Services.

Hars, A. & Ou, S. (2000). *Why is Open Source Software viable? - A Study of Intrinsic Motivation, Personal Needs, and Future Returns*. Paper presented at the Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000), Long Beach, California.

Hars, A. & Ou, S. (2002). Working for free? - Motivations for participating in Open Source projects. *International Journal of Electronic Commerce, 6*, 25-39.

Harvey, R. J., Billings, R. S. & Nilan, K. J. (1985). Confirmatory Factor Analysis of the Job Diagnostic Survey: Good News and Bad News. *Journal of Applied Psychology, 70*(3), 461-468.

Hatcher, L., Ross, T. L. & Collins, D. (1989). Prosocial behavior, Job complexity, and Suggestion contribution under gainsharing plans. *Journal of Applied Behavioral Science, 25*, 231-240.

Hauptman, O. (1986). The influence of task type on the relation between communication and performance: The case of software development. *R & D Management, 16*(2), 127-139.

Heberlein, T. A. & Baumgartner, R. (1978). Factors Affecting Response Rates to Mailed Questionnaires: A Quantitative Analysis of the Published Literature. *American Sociological Review, 43*(4), 447-462.

Hecker, F. (1999). Setting Up Shop: The Business of Open-Source Software. *IEEE Software, 16*(1), 45-51.

Heinbokel, T., Sonnentag, S., Frese, M. & Stolte, W. (1996). Don't underestimate the problems of user centredness in software development projects--there are many! *Behaviour & information technology, 15*(4), 226-236.

Helfen, M. & Krüger, L. (2002). Informationstechnologie, neue Organisationskonzepte und Mitbestimmung. *WSI-Mitteilungen, 11*, 670-677.

Helmke, H., Höppner, F. & Isernhagen, R. (2007). *Einführung in die Software-Entwicklung: Vom Programmieren zur erfolgreichen Software-Projektarbeit.* München: Hanser Fachbuchverlag.

Henry, E. & Faller, B. (1995). Large-scale industrial reuse to reduce cost and cycle time. *Software, IEEE, 12*(5), 47-53.

Herbsleb, J. D. & Grinter, R. E. (1999). *Splitting the organization and integrating the code: Conway's law revisited.* Paper presented at the International Conference on Software Engineering (ICSE '99), Los Angeles, CA, US.

Herbsleb, J. D., Klein, H., Olson, G. M., Brunner, H., Olson, J. S. & Harding, J. (1995). Object-Oriented Analysis and Design in Software Project Teams. *Human-Computer Interaction, 10*(2 & 3), 249-292.

Herbsleb, J. D. & Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering, 29*(6), 481-494.

Herbsleb, J. D., Mockus, A., Finholt, T. A. & Grinter, R. E. (2000). *Distance, dependencies, and delay in a global collaboration.* Paper presented at the ACM Conference on Computer Supported Cooperative Work, New York.

Herbsleb, J. D., Mockus, A., Finholt, T. A. & Grinter, R. E. (2001). *An empirical study of global software development: Distance and speed.* Paper presented at the International Conference on Software Engineering (ICSE 2001), Toronto, Canada.

Herring, S. C. (1996). Two variants of an electronic message schema. In S. C. Herring (Hrsg.), *Computermediated communication: Linguistic, social and crosscultural perspectives* (S. 81-106). Philadelphia: John Benjamins.

Hertel, G. (2007). Motivating job design as a factor in open source governance. *Journal of Manage Governance, 11*, 129-137.

Hertel, G., Bretz, E. & Moser, K. (2000). Freiwilliges Arbeitsengagement: Begriffsklärung und Forschungsstand. *Gruppendynamik und Organisationsberatung, 2*, 121-140.

Hertel, G., Geister, S. & Konradt, U. (2005a). Managing virtual teams: A review of current empirical research. *Human Resource Management Review, 15*(1), 69-95.

Hertel, G., Niedner, S. & Herrmann, S. (2003). Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel. *Research Policy, 32*(7), 1159-1177.

Hertel, G., Schroer, J., Batinic, B., Konradt, U. & Naumann, S. (2005b). Kommunizieren schüchterne Menschen lieber per E-mail? Einflüsse der Persönlichkeit auf die Präferenz von Kommunikationsmedien. In K.-H. Renner, A. Schütz & F. Machilek (Hrsg.), *Internet und Persönlichkeit* (S. 134-147). Göttingen: Hogrefe.

Herzberg, F. (1968). One More Time: How Do You Motivate Employees? *Harvard Business Review, 46*(2), 53-62.

Hibbs, C., Jewett, S. C. & Sullivan, M. (2009). *The Art of Lean Software Development: A Practical and Incremental Approach*: O'Reilly Media, Inc.

Highsmith, J. (2002). *Agile software development ecosystems*: Addison-Wesley.

Hill, B. M. (2001, 15.04.2002). *Free Software Project Management HOWTO*. Verfügbar unter: http://www.linux.com/howtos/Software-Proj-Mgmt-HOWTO/index.shtml [21.03.2006 2006].

Hirschheim, R. & Miller, J. (1993). *Implementing empowerment through teams: the case of Texaco's information technology division.* Paper presented at the Conference on Computer personnel research, St Louis, Missouri, United States.

History of free software. (2009, 29. September). Verfügbar unter: http://en.wikipedia.org/wiki/Open_Source_history 2009].

History of the OSI. (2009). Verfügbar unter: http://www.opensource.org/history [29.09.2009

Holck, J. & Jorgensen, N. (2004). Do Not Check in on Red: Control Meets Anarchy in Two Open Source Projects. In S. Koch (Hrsg.), *Free/Open Source Software Development* (S. 1-26). Hershey: Idea Group Publishing.

Hulin, C. L. (1971). Individual Differences and Job Enrichment: The Case Against General Treatments. In J. R. Maker (Hrsg.), *New Perspectives in Job Enrichment*. New York: Van Nostrand Reinhold.

Hunt, S. D., Chonko, L. B. & Wood, V. R. (1985). Organizational Commitment and Marketing. *Journal of Marketing, 49*(1), 112-126.

Iannacci, F. (2003). The Linux Managing Model. *First Monday, 8*(12).

Iannacci, F. (2005). *Coordination Processes in Open Source Software Development: The Linux Case Study*. Verfügbar unter: http://opensource.mit.edu/papers/iannacci3.pdf

Idaszak, J. R. & Drasgow, F. (1987). A revision of the job diagnostiv survey: elimination of a measurement artifact. *Journal of Applied Psychology, 1*, 69-74.

Igbaria, M., Parasuraman, S. & Badawy, M. K. (1994). Work experiences, job involvement, and quality of work life among information systems personnel. *MIS Quarterly, 18*, 175-201.

Inmon, W. H. (1988). *Information Engineering for the Practitioner*. Englewood Cliffs, NJ: Yourdon.

Israel, B. A. & Antonucci, T. C. (1987). Social network characteristics and psychological well-being: a replication and extension. *Health education quarterly, 14*(4), 461-481.

Ivancevich, J. M., Napier, H. A. & Wetherbe, J. C. (1983). Occupational stress, attitudes, and health problems in the information systems professional. *Communications of the ACM, 26*(10), 800-806.

Ivancevich, J. M., Napier, H. A. & Wetherbe, J. C. (1985). An empirical study of occupational stress, attitudes and health among information systems personnel. *Information and Management, 9*(2), 77-85.

Ives, B. & Olson, M. H. (1984). User involvement and MIS success: A review of research. *Management Science, 30*(5), 586-603.

Jackson, S. E. & Zedeck, S. (1983). Explaining performance variability: contributions of goal setting, task characteristics, and evaluative contexts. *Journal of Applied Psychology, 67*(6), 759-768.

Jain, H. C. (1978). Information, training and effective participation. *Industrial Relations Journal, 9*, 48-60.

Janz, B. D. (1998). *The best and worst of teams: self-directed work teams as an information systems development workforce strategy*. Paper presented at the Proceedings of the 1998 ACM SIGCPR conference on Computer personnel research, Boston, Massachusetts, United States.

Jarvenpaa, S. L. & Ives, B. (1991). Executive involvement and participation in the management of information technology. *MIS Quarterly, 15*, 205-227.

Jenkins, G. D. J., Mitra, A., Gupta, N. & Shaw, J. D. (1998). Are financial incentives related to performance? A meta-analytic review of empirical research. *Journal of Applied Psychology, 83*, 777-787.

Johns, G. J., Xie, J. L. & Fang, Y. (1992). Mediating and moderating effects in job design. *Journal of Management, 18*, 657-676.

Johnson, D. W. & Johnson, R. T. (1992). Positive Interdependence: Key to Effective Cooperation. In R. H. Herz-Lazarowitz & N. Miller (Hrsg.), *Interaction in Cooperative Groups: The Theoretical Anatomy of Group Learning* (S. 174-199). Cambridge, UK: Cambridge University Press.

Johnson, J. P. (2001a, 06.04.2003). *Economics of open source software*. Verfügbar unter: [06.04.2003 2003].

Johnson, K. (2001b). *A Descriptive Process Model for Open-Source Software Development.* Masters thesis, University of Calgary, Calgary.

Johnson, W. L., Johnson, A. M. & Heimberg, F. (1999). A Primary-and Second-Order Component Analysis of the Organizational Identification Questionnaire. *Educational and Psychological Measurement, 59*(1), 159-170.

Jones, C. (1986). *Programming productivity*. New York: McGraw-Hill.

Jones, T. C. (1978). Measuring programming quality and productivity. *IBM Journal of Research and Development, 17*(1), 39-63.

Jorgensen, N. (2001). Putting it All in the Trunk: Incremental Software Engineering in the FreeBSD Open Source Project. *Information Systems Journal, 11*(4), 321-336.

Judge, T. A., Thoresen, C. J., Bono, J. E. & Patton, G. K. (2001). The Job Satisfaction-Job Performance Relationship: A Qualitative and Quantitative Review. *Psychological Bulletin, 127*(3), 376-407.

Kahn, H. & Robertson, I. T. (1992). Training and experience as predictors of job satisfaction and work motivation when using computers: A correlational study. *Behaviour & Information Technology, 11*(1), 53-60.

Kaisla, J. (2001). *Constitutional dynamics of the open source software development*. Verfügbar unter: http://www.cbs.dk/departments/ivs/wp/wp01-12.pdf [21.04.2003 2003].

Karasek, R., Brisson, C., Kawakami, N., Houtman, I., Bongers, P. & Amick, B. (1998). The job content questionnaire (JCQ): An instrument for internationally comparative assessment of psychosocial job characteristics. *Journal of Occupational Health Psychology, 3*(4), 322-355.

Karasek, R. A. (1991). The political implications of psychosocial work redesign: A model of the psychosocial class structure. In J. V. Johnson & G. Johansson (Hrsg.), *The psychosocial work environment: Work organization, democratization, and health* (S. 163-190). Baywood, NY: Amityville.

Karasek, R. A. (1998). Demand-control model: A social, emotional, and physiological approach to stress risk and active behaviour development. In J. M. Stellmann (Hrsg.), *Encyclopaedia of occupational health and safety* (Bd. 4, S. 34.36–34.14). Geneva: International Labour Office.

Katz, H. C., Kochan, T. A. & Weber, M. R. (1985). Assessing the effects of industrial relations systems and efforts to improve the quality of working life on organizational effectiveness. *Academy of Management Journal, 3*(28), 509-526.

Katz, R. (1978). Job Longevity as a Situational Factor in Job Satisfaction. *Administrative science quarterly, 23*, 204-223.

Katz, R. (1982). The Effects of Group Longevity on Project Communication and Performance. *Administrative Science Quarterly, 27*(1), 81-104.

Katz, R. & Allen, T. J. (1985). Project Performance and the Locus of Influence in the R&D Matrix. *The Academy of Management Journal, 28*(1), 67-87.

Keller, R. T. (1983). Predicting absenteeism from prior absenteeism, attitudinal factors, and nonattitudinal factors. *Journal of applied psychology, 68*(3), 536-540.

Keller, R. T. (1994). Technology-Information Processing Fit and the Performance of R&D Project Groups: A Test of Contingency Theory. *The Academy of Management Journal, 37*(1), 167-179.

Kiesler, S. & Cummings, J. N. (2002). What do we know about proximity and distance in work groups? A legacy of research. In P. Hinds & S. Kiesler (Hrsg.), *Distributed Work* (S. 57-80). Cambridge, MA: MIT Press.

Kiesler, S. B., Siegel, J. & McGuire, T. W. (1984). Social psychological aspects of computer-mediated communication. *American Psychologist, 39*, 1123-1134.

Kiggundu, M. N. (1983). Task interdependence and job design: Test of a theory. *Organizational Behavior and Human Performance, 31*(2), 145-172.

Kilduff, M. & Regan, D. T. (1988). What people say and what they do: the differential effects of informational cues and task design. *Organizational behavior and human decision processes, 41*(1), 83-97.

Kinney, S. T. & Dennis, A. R. (1994). *Reevaluating media richness: cues feedback and task.* Paper presented at the Twenty-Seventh Annual Hawaii International Conference on System Science.

Kinney, S. T. & Watson, R. T. (1992). *The Effect of Medium and Task on Dyadic Communication.* Paper presented at the International Conference on Information Systems.

Klemens, S. (2003). Individuelle und organisationale Ressourcen für gesunde und effektive IT-Arbeit. In T. K. U. Wuppertal (Hrsg.), *Tagungsband zur Fachtagung "Gesund in die Zukunft - Moderne IT-Arbeitswelt gestalten, 01./02. Juli in Berlin.* Hamburg/Wuppertal.

Klepper, R., Litecky, C. & Jones, T. W. (1989). Self managed teams and MIS productivity: literature, model and field study. *SIGMIS Database, 20*(1), 36-38.

Kluger, A. N. & DeNisi, A. (1996). The effects of feedback interventions on performance: An historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin, 119*(2), 254-284.

Koch, S. (2004, June 6-10, 2004). *Agile Principles and Open Source Software Development: A Theoretical and Empirical Discussion.* Paper presented at the Extreme Programming and Agile Processes in Software Engineering: 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany.

Koch, S. & Gonzalez-Barahona, J. M. (2005). Open Source software engineering - The state of research. *First Monday, Special Issue #2.*

Koch, S. & Schneider, G. (2000). *Results from Software Engineering Research into Open Source Development Projects Using Public Data.* Wien: Wirtschaftsuniversität Wien.

Koch, S. & Schneider, G. (2002). Effort, Cooperation and Coordination in an Open Source Software Project: GNOME. *Information Systems Journal, 12*(1), 27-42.

Kogut, B. & Metiu, A. (2001). Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy, 17*(2), 248-264.

Kogut, B. & Zander, R. (1996). What firms do? Coordination, identity and learning. *Organization Science, 7*, 502-518.

Kollock, P. & Smith, M. A. (1999). Communities in Cyberspace. In M. A. Smith & P. Kollock (Hrsg.), *Communities in Cyberspace* (S. 3-25). New York, NY: Routledge.

Konradt, U. & Sulz, K. (2001). The experience of flow in interacting with a hypermedia learning environment. *Journal of Educational Multimedia and Hypermedia, 10*(1), 69-84.

Köppen, A. & Nüttgens, M. (2000). Open Source: Strategien für die Beratung. In A.-W. Scheer & A. Köppen (Hrsg.), *Consulting - Wissen für die Strategie-, Prozess- und IT-Beratung.* Berlin: Springer.

Kraft, P. (1977). *Programmers and Managers: The Routinization of Computer Programming in the United States.* New York: Springer.

Kramer, R. M. (1993). Cooperation and Organizational Identification. In J. K. Murnighan (Hrsg.), *Social Psychology in Organizations: Advances in Theory and Research* (S. 244-268). Englewood Cliffs, NJ: Prentice Hall.

Kramer, R. M. & Brewer, M. B. (1984). Effects of group identity on resource use in a simulated commons dilemma. *Journal of Personality and Social Psychology, 46*, 1044-1057.

Kramer, R. M. & Brewer, M. B. (1986). Social group identity and the emergence of cooperation in resource conservation dilemmas. In H. Wilke, C. Rutte & D. M. Messick (Hrsg.), *Experimental Studies of Social Dilemmas* (S. 205-234). Frankfurt: Peter Lang.

Krasner, H., Curtis, B. & Iscoe, N. (1987). Communication breakdowns and boundary spanning activities on large programming projects. In G. Olson, S. Shepard & E. Soloway (Hrsg.), *Empirical Studies of Programmers - Second Workshop* (S. 47-64). Washington, DC.

Kraut, R., Egido, C. & Galegher, J. (1988). *Patterns of contact and communication in scientific research collaboration.* Paper presented at the 1988 ACM conference on Computer-supported cooperative work, Portland, Oregon.

Kraut, R. E. & Streeter, L. A. (1995). Coordination in large scale software development. *Communications of the ACM, 38*(7), 69–81.

Krishnamurthy, S. (2002). Cave or Community? An Empirical Investigation of 100 Mature Open Source Projects. *First Monday, 7*(6).

Krishnamurthy, S. (2003). A managerial overview of open source software. *Business Horizons*(September-October), 47-56.

Krishnamurthy, S. (2005). About Closed-door Free/Libre/Open Source (FLOSS) Projects: Lessons from the Mozilla Firefox Developer Recruitment Approach. *Upgrade, 6*(3), 28-32.

Kroah-Hartman, G. (2007). *Linux Kernel Development.* Paper presented at the Proceedings of the Linux Symposium 2007, Ottawa, Ontario, Canada.

Kroah-Hartman, G., Corbet, J. & McPherson, A. (2009, 5. November). *Linux Kernel Development.* Verfügbar unter: http://www.linuxfoundation.org/publications/whowriteslinux.pdf

Krogh, G. v., Haefliger, S. & Spaeth, S. (2003a). *Collective Action and Communal Resources in Open Source Software Development: The Case of Freenet.* Verfügbar unter: http://www.alexandria.unisg.ch/EXPORT/PDF/Publikation/30993.pdf

Krogh, G. v., Späth, S. & Lakhani, K. R. (2003b). Community, Joining and Specialization in Open Source Software Innovation: A Case Study. *Research Policy, 32*(7), 1217-1241.

Kuan, J. W. (2001). *Open Source Software as Consumer Integration Into Production.* Verfügbar unter: http://ssrn.com/abstract=259648 [23.10.2006 2006].

Kühl, S. & Kullmann, G. (1999). *Gruppenarbeit, einführen, bewerten und weiterentwickeln.* München: Hanser.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour and Information Technology, 22*(1), 1-16.

Kulik, C. T., Oldham, G. R. & Langner, P. H. (1988). Measurement of Job Characteristics: Comparison of the Original and the Revised Job Diagnostic... *Journal of Applied Psychology, 73*(3), 462-466.

Lakhani, K., R., Wolf, R. G., Bates, J. & DiBona, C. (2002). *The Boston Consulting Group Hacker Survey*. Verfügbar unter: http://www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf [18.09.2006 2006].

Lakhani, K. R. & Hippel, E. v. (2000). *How Open Source Software works: 'Free' user-to-user assistance - Working Paper*: MIT Sloan School of Management.

Lakhani, K. R. & Wolf, R. G. (2003). *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*. [PDF]. Verfügbar unter: http://ssrn.com/abstract=443040 [19.01.2004 2004].

Lankenau, K. (1984). Determinanten der Qualifizierungsbereitschaft von Industriearbeiterinnen. *Soziale Welt, 35*(1/2), 236-264.

Lanzara, G. F. & Morner, M. (2003). *The Knowledge Ecology of Open-Source Software Projects*. Paper presented at the 19th EGOS Colloquium, Copenhagen.

Latniak, E. & Gerlmaier, A. (2006). *Zwischen Innovation und alltäglichem Kleinkrieg: Zur Belastungssituation von IT-Beschäftigten*. Gelsenkirchen: Institut Arbeit und Technik.

Lawrence, M. J. (1981). Programming methodology, organizational environment, and programming productivity. *Journal of Systems and Software, 2*, 257-269.

Lee, G. K. & Cole, R. E. (2003). *From a firm-based to a community-based model of knowledge creation: the case of the linux kernel development*. Verfügbar unter: http://faculty.haas.berkeley.edu/glee/OS00-1246RR_glee.pdf [22.03.2003 2003].

Lee, R. & Graham, W. K. (1986). Self-actualization need strength: Moderator of relationships between job characteristics and job outcomes. *Journal of Employment Counseling, 23*(1), 38-47.

Lee, S. M. (1971). An Empirical Analysis of Organizational Identification. *Academy of Management Journal, 14*, 213-226.

Lehman, J. H. (1979). How Software Projects are Really Managed. *Datamation, 25*, 119-129.

Lending, D. (1997). *CASE technology and systems development job characteristics*. Dissertation, University of Minnesota.

Lending, D. & Chervany, N. L. (1997). *The changing systems development job: a job characteristics approach.* Paper presented at the Proceedings of the 1997 ACM SIGCPR conference on Computer personnel research, San Francisco, California, United States.

Lending, D. & Chervany, N. L. (1998a). CASE tools: understanding the reasons for non-use. *ACM SIGCPR Computer Personnel, 19*(2), 13-26.

Lending, D. & Chervany, N. L. (1998b). *The use of CASE tools*. Paper presented at the Proceedings of the 1998 ACM SIGCPR conference on Computer personnel research, Boston, Massachusetts, United States.

LePine, J. A., Erez, A. & Johnson, D. E. (2002). The Nature and Dimensionality of Organizational Citizenship Behavior: A Critical Review and Meta-Analysis. *Journal of Applied Psychology, 87*(1), 52-65.

Lerner, J. & Tirole, J. (2001). The open source movement: Key research questions. *European Economic Review, 45*(4-6), 819-826.

Lerner, J. & Tirole, J. (2002a). *The scope of open source licensing*. Verfügbar unter: [22.04.2003 2003].

Lerner, J. & Tirole, J. (2002b). Some Simple Economics of Open Source. *Journal of Industrial Economics, 50*(2), 197-234.

Lienert, G. A. & Raatz, U. (1994). *Testaufbau und Testanalyse*. Weinheim: Beltz.

Lim, W. C. (1994). Effects of reuse on quality, productivity, and economics. *Software, IEEE, 11*(5), 23-30.

Lin, N. (2001). *Social Capital*. Cambridge, UK: Cambridge University Press.

Lin, S. L. & Hsieh, A. T. (2002). Constraints of task identity on organizational commitment. *International Journal of Manpower, 23*(2), 151-165.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F.et al. (2002). *Empirical Findings in Agile Methods.* Paper presented at the XP/Agile Universe 2002, Second XP Universe and First Agile Universe Conference, Chicago, IL, USA.

Linus Torvalds. (2009, 5. November). Verfügbar unter: http://de.wikipedia.org/wiki/Linus_Torvalds 2009].

Ljungberg, J. (2000). Open Source Movements as a Model for Organising. *European Journal of Information Systems, 9*(4), 208-216.

Locke, E. A. (1966). The Relationship of Intentions to Level of Performance. *Journal of Applied Psychology, 50*, 60-66.

Locke, E. A. & Latham, G. P. (1990). *A theory of goal-setting and task performance*. Englewood Cliffs, NJ: Prentice Hall.

Locke, E. A. & Schweiger, D. M. (1979). Participation in decision-making: One more look. In B. Staw (Hrsg.), *Reseach in organizational behavior* (S. 265-339). Greenwich: JAI Press.

Locke, E. A., Sirota, D. & Wolfson, A. D. (1976). An experimental case study of the successes and failures of job enrichment in a government agency. *Journal of Applied Psychology, 61*(6), 701-711.

Loher, B. T., Noe, R. A., Moeller, N. L. & Fitzgerald, M. P. (1985). A Meta-Analysis of the Relation of Job Characteristics to Job Satisfaction. *Journal of Applied Psychology, 70*(2), 280-289.

Lohmann, A. & Prümper, J. (2002). *CCall - Report 7. Partizipation im Call-Center*. Hamburg: VBG. Verwaltungs-Berufsgenossenschaft.

Long, A. (2003). *How Firm Initiation and Control of Projects Affects Open-Source Development*. Verfügbar unter: opensource.mit.edu/papers/long.pdf [15.09.2006 2006].

Lowin, A. (1968). Participation in decision making: A model, literature critique and prescriptions for research. *Human Performance, 3*(68-106).

Luthiger, B. (2004). Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern. In R. A. Gehring & B. Lutterbeck (Hrsg.), *Open Source Jahrbuch 2004* (S. 93-106). Berlin: Lehmanns Media.

Luthiger, B. (2005). *Fun and Software Development.* Paper presented at the The First International Conference on Open Source Systems, Genova, Italy.

Luthiger, B. (2006). *Spass und Software-Entwicklung: Zur Motivation von Open-Source-Programmierern.* Universität Zürich, Zürich.

Lyons, D. (2006, 4. Oktober). *Linux Licensing.* Verfügbar unter: http://www.forbes.com/2006/03/09/torvalds-linux-licensing-cz_dl_0309torvalds1.html

MacCormack, A. (2001). How Internet Companies Build Software. *Sloan Management Review, 42*(2), 75-84.

MacCormack, A., Rusnak, J. & Baldwin, C. (2005, 09.06.2005). *Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code.* [PDF]. Verfügbar unter: http://opensource.mit.edu/papers/maccormackrusnakbaldwin2.pdf [15.07.2005 2005].

Madey, G., Freeh, V. & Tynan, R. (2002). *Understanding OSS as a Self-Organizing Process.* Paper presented at the ICSE'02 International Conference on Software Engineering - 2nd Workshop on Open Source Software Engineering, Orlando, FL, USA.

Mall, R. (2004). *Fundamentals of Software Engineering.* New Delhi: Prentice-Hall of India.

Mantei, M. (1981). The Effect of Programming Team Structures on Programming Tasks. *Communications of the ACM, 24*(3), 106-113.

Manville, B., Markus, M. L. & Agres, C. E. (1999). *The Open Source Movement - A Prototype of Future Organizational Governance?* (White Paper).

Markus, L., M., Manville, B. & Agres, C. E. (2000). What Makes a Virtual Organization Work. *Sloan Management Review, 42*(1), 13-26.

Martin, T. N. & Hunt, J. G. (1980). Social Influence and Intent to Leave: A Path-Analytic Process Model. *Personnel Psychology, 33*(3), 505-528.

Matern, B. (1984). *Psychologische Arbeitsanalyse.* Heidelberg: Springer.

Matern, B. (Hrsg.). (1983). *Psychologische Arbeitsanalyse. Spezielle Arbeits- und Ingenieurspsychologie.* Berlin: Deutscher Verlag der Wissenschaften.

Mathieu, J. E., Hofmann, D. A. & Farr, J. L. (1993). Job perception-job satisfaction relations: an empirical comparison of three competing theories. *Organizational Behavior and Human Decision Processes, 56*, 370-387.

McCauley, C. D., Ruderman, M. N., Ohlott, P. J. & Morrow, J. E. (1994). Assessing the developmental components of managerial jobs. *Journal of Applied Psychology, 79*, 544–560.

McConnell, S. (1999). Open-Source Methodology: Ready for Prime Time? *IEEE Software, 16*(4), 6-8.

McConnell, S. (2004). *Code complete.* Redmond, WA, USA: Microsoft Press.

McCormick, F. & Tiffin, J. (1974). *Industrial Psychology* (6). Englewood cliffs: Prentice Hill.

McKenna, K. & Bargh, J. (1998). Coming out in the age of the Internet: "Demarginalization" through virtual group participation. *Journal of Personality and Social Psychology, 75*, 681–694.

McKnight, D. H. & Chervany, N. L. (1998). *The motivational nature of the critical systems operator job: expanding the job characteristics model with relationships.* Paper presented at the ACM SIGCPR conference on Computer personnel research, Boston, Massachusetts, United States.

McMahon, A. & Camilieri, S. F. (1975). Organizational structure and voluntary participation in collective-goods decisions. *American Sociological Review, 40*, 616-644.

McQuillan, J. & Conde, G. (1996). The conditions of flow in reading: two studies of optimal experience. *Reading Psychology, 17*, 109-135.

Meares, C. A. & Sargent, J. F. (1999). *The digital workforce: Building infotech skills at the speed of innovation.* Washington, DC: U.S. Department of Commerce, Technology Administration, Office of Technology Policy.

Mickelson, K. D. (1997). Seeking social support: Parents in electronic support groups. In S. Kiesler (Hrsg.), *Culture of the Internet* (S. 157-178). Mahawah, NJ: Lawrence Erlbaum Associates.

Mieg, H. A. & Wehner, T. (2002). Frei-gemeinnützige Arbeit. Eine Analyse aus Sicht der Arbeits- und Organisationspsychologie. *Harburger Beiträge zur Psychologie und Soziologie der Arbeit, 33.*

Miller, K. I. & Monge, P. R. (1986). Participation, satisfaction and productivity: a metaanalytic review. *Academy of Management Journal, 29*(4), 727-753.

Mills, H. D. (1971). Top-down programming in large systems. In R. Rustin (Hrsg.), *Debugging Techniques in Large Systems.* Upper Saddle River, NJ: Prentice Hall.

Miner, J. B. (1980). *Theories of organizational behavior.* Hinsdale, IL: Dryden.

Mitchell, T. R. (1973). Motivation and Participation: An Integration. *The Academy of Management Journal, 16*(4), 670-679.

Mockus, A., Fielding, R. T. & Herbsleb, J. D. (2000). A Case Study of Open Source Software Development: The Apache Server. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)* (S. 263-272). Limerick, Ireland.

Mockus, A., Fielding, R. T. & Herbsleb, J. D. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346.

Moody, G. (2001). *Rebel code - Inside Linux and the open source movement.* Cambridge, MA: Perseus Publishing.

Moon, J. Y. & Sproull, L. (2002). Essence of distributed work: The case of the Linux kernel. In P. Hinds & S. Kiesler (Hrsg.), *Distributed work* (S. 381-404). Cambridge, MA, US: MIT Press.

Moore, J. E. & Love, M. S. (2005). IT professionals as organizational citizens. *Communications of the ACM, 48*(6), 88-93.

Moorman, R. H. (1991). Relationship between organizational justice and organizational citizenship behaviors: do fairness perceptions influence employee citizenship? *Journal of applied psychology, 76*(6), 845-855.

Moorman, R. H., Blakely, G. L. & Niehoff, B. P. (1998). Does Perceived Organizational Support Mediate the Relationship between Procedural Justice and Organizational Citizenship Behavior? *The Academy of Management Journal, 41*(3), 351-357.

Morgeson, F. P. & Campion, M. A. (2003). Work Design. In W. C. Borman, D. R. Ilgen & R. J. Klimoski (Hrsg.), *Handbook of Psychology, Volume 12, Industrial and Organizational Psychology* (S. 423-452). Chichester: John Wiley & Sons.

Morgeson, F. P. & Humphrey, S. E. (2003). *The Work Design Questionnaire (WDQ): Developing and validating a comprehensive measure of work design.* Paper presented at the Interactive paper session presented at the 62nd Annual Meeting of the Academy of Management, Seattle, WA, USA.

Mossholder, K. W. (1980). Effects of externally mediated goal setting on intrinsic motivation: A laboratory experiment. *Journal of Applied Psychology, 65*(2), 202–210.

Mueller, C. W. & Price, J. L. (1990). Economic, psychological and sociological determinants of voluntary turnover. *Journal of Behavioral Economics, 19*(3), 321–335.

Muffatto, M. & Faldani, M. (2003). Open Source as a Complex Adaptive System. *Emergence, 5*(3), 83-100.

Nakakoji, K. & Yamamoto, Y. (2001). Taxonomy of open source software development. In J. Feller, B. Fitzgerald & A. v. d. Hoek (Hrsg.), *Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering.*

Namm, S. (2004). *The job characteristics - organizational citizenship behavior relationship: A test of competing models.* Temple U., US.

Narduzzo, A. & Rossi, A. (2003). *Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed.* Verfügbar unter: http://opensource.mit.edu/papers/narduzzorossi.pdf

Naur, P. & Randell, B. (1968). *Software Engineering: Report on a conference sponsored by the Nato Science Committee.* Garmisch: Nato Science Commitee.

Neider, L. L. (1980). An experimental field investigation utilizing an expectancy theory view of participation. *Organizational Behavior and Human Performance, 26*, 425-442.

Nelson, A. C. (1996). *Employee-job fit in MIS: research in progress.* Paper presented at the Proceedings of the 1996 ACM SIGCPR/SIGMIS conference on Computer personnel research, Denver, Colorado, United States.

Nelson, D. L. & Quick, J. C. (1991). Social support and newcomer adjustment in organizations: Attachment theory at work? *Journal of Organizational Behavior, 12*, 543-554.

Neus, A. (2001). *Managing Information Quality in Virtual Communities of Practice - Lessons learned from a decade's experience with exploding Internet communication.* Paper presented at the IQ 2001: The 6th International Conference on Information Quality at MIT.

Ng, T. W. H. & Sorensen, K. L. (2008). Toward a Further Understanding of the Relationship Between Perceptions of Support and Work Attitudes: A Meta-Analysis. *Group & Organization Management, 33*(3), 243-268.

Nichols, D. M. & Twidale, M. B. (2003). The Usability of Open Source Software. *First Monday, 8*(1).

Nicholson, N. (1977). Absence behavior and attendance motivation: A conceptual synthesis. *Journal of Management Studies, 14*, 231-252.

Noll, J. & Scacchi, W. (1999). Supporting Software Development in Virtual Enterprises. *Journal of Digital Information, 1*(4).

Norin, L. A. & Stöckel, F. (1998). *Open-source software development methodology.* Master of Science in System Development and Software Engineering, Luleå university of technology, Luleå, Sweden.

Novak, M. A. & Hoffman, D. L. (1997). *Measuring the Flow Experience Among Web Users.* Paper presented at the Interval Research Corporation.

Novak, M. A., Hoffman, D. L. & Yung, Y.-F. (1999). Measuring the Customer Experience in Online Environments: A Structural Modeling Approach. *Marketing Science, 19*(1), 22-44.

Nunnally, J. C. (1978). *Psychometric Theory* (2nd). New York: McGraw-Hill.

O'Mahony, S. & Ferraro, F. (2004, 01.09.2004). *Hacking Alone? The Effects of Online and Offline Participation on Open Source Community Leadership.* Verfügbar unter: http://opensource.mit.edu/papers/omahonyferraro2.pdf [04.11.2004

O'Reilly, C. A. & Caldwell, D. F. (1979). Information Influence as a Determinant of Perceived Task Characteristics and Job Satisfaction. *Journal of Applied Psychology, 64*(2), 157-165.

O'Reilly, T. (1999). Lessons from Open-Source Software Development. *Communications of the ACM, 42*(4), 32-73.

O'Reilly, C. A. & Caldwell, D. F. (1979). Informational influence as a determinant of perceived task characteristics and job satisfaction. *Journal of Applied Psychology, 64*(2), 157-165.

Oesterreich, R. & Volpert, W. (1987). Handlungstheoretisch orientierte Arbeitsanalyse. In U. Kleinbeck & J. Rutenfranz (Hrsg.), *Enzyklopädie der Psychologie* (S. 43-73). Göttingen: Hogrefe.

Oldham, G. R. (1996). Job design. In C. L. Cooper & I. T. Robertson (Hrsg.), *International review of industrial and organizational psychology* (Bd. 11, S. 34-60). New York: Wiley & Sons.

Olson, C. L. (1976). On choosing a test statistic in multivariate analyses of variance. *Psychological Bulletin, 83*, 579-586.

Open Source Initiative. (2006, 09.02.2006). *The Open Source Definition Version 1.9.* Verfügbar unter: http://www.opensource.org/docs/definition.php

Organ, D. W. (1988). *Organizational citizenship behavior: the good soldier syndrome.* Lexington, MA: Lexington Books.

Organ, D. W. (1997). Organizational Citizenship Behavior: It's Construct Clean-Up Time. *Human Performance, 10*(2), 85-98.

Orpen, C. (1979). The Effects of Job Enrichment on Employee Satisfaction, Motivation, Involvement, and Performance: A Field Experiment. *Human Relations, 32*(3), 189-217.

Osterloh, M., Rota, S. & Kuster, B. (2004). Open Source Software Produktion: Ein neues Innovationsmodell? In R. A. Gehring & B. Lutterbeck (Hrsg.), *Open Source Jahrbuch 2004.* Berlin: Lehmanns Media.

Osterloh, M., Rota, S. & Kuster, B. (in Druck). Open Source Software Production: Climbing on the Shoulders of Giants. In *Vertrauen in der vernetzten Wirtschaft*: Springer.

Osterloh, M., Rota, S. & Kuster, B. (in Druck). Trust and Commerce in Open Source - a Contradiction? In *Trust in the Network Economy*. Wien: Springer.

Page, R. C. & Caskey, D. T. (1988). Computer Programmer. In S. Gael (Hrsg.), *Job Analysis Part I.* Chinchester: Wiley.

Parker, S. & Wall, T. (1998). *Job and Work Design: Organizing Work to Promote Well-Being and Effectiveness*. Thousand Oaks: Sage Publications.

Parker, S. K. (1998). Enhancing Role Breadth Self-Efficacy: The Roles of Job Enrichment and Other Organizational Interventions. *Journal of Applied Psychology, 83*(6), 835-852.

Parker, S. K. & Wall, T. D. (2002). Work design: Learning from the past and mapping a new terrain. In N. Anderson, D. S. Ones & H. K. Sinangil (Hrsg.), *Handbook of industrial, work and organizational psychology* (Bd. Volume 1: Personnel psychology, S. 90-109). London, England: Sage.

Parker, S. K., Wall, T. D. & Jackson, P. R. (1997). That's not my job: Developing flexible employee work orientations. *Academy of Management Journal, 40*(4), 899-929.

Parkes, K. R. (1982). Occupational stress among student nurses: A natural experiment. *Journal of Applied Psychology, 67*, 784-796.

Parks, M. R. & Roberts, L. D. (1997). Making MOOsic: The Development of Personal Relationships On-line and a Comparison to their Offline Counterparts. *Journal of Social and Personal Relationships, 15*(4), 517-537

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM, 15*(12), 1053-1058.

Pasmore, W. A. (1978). The comperative impacts of sociotechnical systems, job redesign and survey-feedback interventions. In W. A. Pasmore & J. Sherwoord (Hrsg.), *Sociotechnical systems: A source book*. San Diego: University Associates.

Paulson, J. W., Succi, G. & Eberlein, A. (2004). An empirical study of open-source and closed-source software products. *IEEE Transactions on Software Engineering, 30*(4), 246-256.

Pearce, J. L. & Gregersen, H. B. (1991). Task interdependence and extrarole behavior: a test of the mediating effects of felt responsibility. *Journal of applied psychology, 76*(6), 838-844.

Pelley, L., Kappelman, L. & Vanacek, M. (1992). *The impact of computer aided systems engineering on employee attitudes, job commitment and turnover.* Paper presented at the ACM SIGCPR conference on Computer personnel research, Cincinnati, Ohio, United States.

Pennings, J. (1973). Measures of organizational structure: a methodological note. *American Journal of Sociology, 79*, 686-704.

Perkins, G. (1999). Culture Clash and the Road to World Domination. *IEEE Software, 16*(1), 80-84.

Perlow, L. & Weeks, J. (2002). Who's helping whom? Layers of culture and workplace behavior. *Journal of Organizational Behavior, 23*(4), 345-361.

Perry, D. E., Staudenmayer, N. A. & Votta, L. G. (1994). People, organizations, and process improvement. *Software, IEEE, 11*(4), 36-45.

Petrie, R., Moore, D. L. & Dillman, D. A. (1998). *Establishment of surveys: The effect of multi-mode sequence on response rates.* Paper presented at the American Statistical Association, Alexandria, VA.

Peyrache, E., Cremer, J. & Tirole, J. (2000). Some Reflections on Open Source Software. *Communications and Strategies, Special Issue 4th Quarter 2000*(40), 139-159.

Pickering, J. M. & King, J. L. (1995). Hardwiring weak ties Interorganizational computer-mediated communication, occupational communities, and organizational change. *Organization Science, 6*(4), 479-486.

Pierce, J. L. & Dunham, R. B. (1978a). The Measurement of perceived Job Characteristics: The Job Diagnostic Survey versus the Job Characteristics Inventory. *Academy of Management Journal, 21*(1), 123-128.

Pierce, J. L. & Dunham, R. B. (1978b). Research notes. *Academy of Management Journal, 21*(1), 123-128.

Podsakoff, P. M. & MacKenzie, S. B. (1995). An examination of substitutes for leadership within a levels of analysis framework. *Leadership Quarterly, 6*(3), 289-328.

Podsakoff, P. M., MacKenzie, S. B. & Bommer, W. H. (1996). Meta-analysis of the relationships between Kerr and Jermier's substitutes for leadership and employee job attitudes, role perceptions, and performance. *Journal of Applied Psychology, 81*(4), 380-399.

Podsakoff, P. M., MacKenzie, S. B., Moorman, R. H. & Fetter, R. (1990). Transformational leader behaviors and their effects on followers' trust in leader, satisfaction, and organizational citizenship behaviors. *Leadership Quarterly, 1*(2), 107-142.

Podsakoff, P. M., MacKenzie, S. B., Paine, J. B. & Bachrach, D. G. (2000). Organizational Citizenship Behaviors: A Critical Review of the Theoretical and Empirical Literature and Suggestions for Future Research. *Journal of Management, 26*(3), 513-564.

Podsakoff, P. M., Niehoff, B. P., MacKenzie, S. B. & Williams, M. L. (1993). Do substitutes for leadership really substitute for leadership? An empirical examiniation of Kerr and Jermier's situational leadership model. *Organizational Behavior and Human Decision Processes, 54*(1), 1-44.

Potdar, V. & Chang, E. (2004). *Open Source and Closed Source Software Development Methodologies.* Paper presented at the ICSE'04 International Conference on Software Engineering - 4th Workshop on Open Source Software Engineering, Edinburgh, Scotland.

Pressman, R. (2000). *Software engineering: A practitioner's approach* (5). Boston, MA: McGraw-Hill.

Price, J. L. & Mueller, C. W. (1986a). *Absenteeism and Turnover among Hospital Employees.* Greenwich: JAI.

Price, J. L. & Mueller, C. W. (1986b). *Absenteeism and Turnover of Hospital Employees.* Greenwich: JAI Press.

Probst, G. J. B. (1987a). *Selbst-Organisation - Ordnungsprozesse in sozialen Systemen aus ganzheitlicher Sicht*. Berlin: Paul Parey.

Probst, G. J. B. (1987b). Selbstorganisation und Entwicklung. *Die Unternehmung, 4*, 242-255.

Probst, G. J. B. (1992). Selbstorganisation. In E. Frese (Hrsg.), *Handwörterbuch der Organisation* (S. 2255-2269.). Stuttgart: Poeschel.

Prüden, H. O. & Reese, R. M. (1972). Interorganizational Role-Set Relations and the Performance and Satisfaction of Industrial Salesmen. *Administrative Science Quarterly, 17*(4), 601-609.

Prümper, J., Hartmannsgruber, K. & Frese, M. (1995). KFZA. Kurz-Fragebogen zur Arbeitsanalyse. *Zeitschrift für Arbeits- und Organisationspsychologie, 39*, 125-131.

Pym, D. L. A. & Auld, H. D. (1965). The Self-Rating as a Measure of Employee Satisfaction. *Occupational Psychology, 39*(2), 103-113.

Rafaeli, S. & La Rose, R. J. (1993). Electronic Bulletin Boards and "Public Goods" Explanations of Collaborative Mass Media. *Communication Research, 20*(2), 277-297.

Ramaswami, S. N., Agarwal, S. & Bhargava, M. (1993). Work Alienation of Marketing Employees: Influence of Task, Supervisory, and Organizational Structure Factors. *Journal of the Academy of Marketing Science, 21*(3), 179-193.

Rau, R. & Riedel, S. (2004). Besteht ein Zusammenhang zwischen dem Auftreten von positiven Arbeitserleben unter Flow-Bedingungen und Merkmalen der Arbeitstätigkeit? *Zeitschrift für Arbeits- und Organisationspsychologie, 48*(2), 55-66.

Rauscher, T. G. & Smith, P. G. (1995). Time-driven development of software in manufactured goods. *Journal of Product Innovation Management, 12*(3), 186-199.

Raymond, E. S. (1999a). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, California: O'Reilly and Associates.

Raymond, E. S. (1999b). Linux and Open-Source Success. *IEEE Software, 16*(1), 85-89.

Raymond, E. S. (2000, 11.09.2000). *The Cathedral and the Bazaar*. [PS]. Verfügbar unter: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/cathedral-bazaar.ps [13.05.2004 2004].

Reeve, J. & Deci, E. L. (1996). Elements of the Competitive Situation That Affect Intrinsic Motivation. *Personality and Social Psychology Bulletin, 22*(1), 24-33.

Reichers, A. E. (1987). An Interactionist Perspective on Newcomer Socialization Rates. *The Academy of Management Review, 12*(2), 278-287.

Renn, R. W. & Vandenberg, R. J. (1995). The critical psychological states: An underrepresented component in job characteristics model research. *Journal of Management, 21*(2), 279-303.

Rentsch, J. R. & Steel, R. P. (1998). Testing the durability of job characteristics as predictors of absenteeism over a six-year period. *Personnel Psychology, 51*, 165-190.

Resch, M., Bamberg, E. & Mohr, G. (1997). Von der Erwerbsarbeitspsychologie zur Arbeitspsychologie. In I. Udris (Hrsg.), *Arbeitspsychologie für morgen. Herausforderungen und Perspektiven* (S. 37-52). Heidelberg: Roland Asagener Verlag.

Rheinberg, F. (1987). *Fragen zum Erleben von Tätigkeiten. (Ein Fragebogen zur Erfassung des Flow-Erlebens im Alltag.)*: Psychologisches Institut der Universität Heidelberg.

Rheinberg, F. (2009). Intrinsische Motivation und Flow Erleben. In H. Heckhausen (Hrsg.), *Motivation und Handeln* (S. 331-354). Heidelberg: Springer.

Rheinberg, F. & Tramp, N. (2006). Anreizanalyse intensiver Nutzung von Computern in der Freizeit. *Zeitschrift für Psychologie, 214*(2), 97-107.

Rheinberg, F., Vollmeyer, R. & Engeser, S. (2003). Die Erfassung des Flow-Erlebens. In J. Stiensmeier-Pelster & F. Rheinberg (Hrsg.), *Diagnostik von Motivation und Selbstkonzept (Tests und Trends N.F. 2)* (S. S. 261-279). Göttingen: Hogrefe.

Rice, R. E. (1993). Media Appropriateness. Using Social Presence Theory to Compare Traditional and New Organizational Media. *Human Communication Research, 19*(4), 451-484.

Rice, R. E. & Love, G. (1987). Electronic emotion: Socioemotional content in a computer mediated network. *Communication Research, 14*, 85-108.

Riketta, M. (2005). Organizational identification: A meta-analysis. *Journal of Vocational Behavior, 66*(2), 358-384.

Rimann, M. & Udris, I. (1993). *Belastungen und Gesundheitsressourcen im Berufs- und Privatbereich. Eine quantitative Studie (Forschungsprojekt SALUTE: Personale und organisationale Ressourcen der Salutogenese, Bericht Nr. 3).* Zürich: Eidgenössische Technische Hochschule, Institut für Arbeitspsychologie.

Rimann, M. & Udris, I. (1997). Subjektive Arbeitsanalyse: der Fragebogen SALSA. In O. Strohm & E. Ulich (Hrsg.), *Unternehmen arbeitspsychologisch bewerten. Ein Mehr-Ebenen-Ansatz unter besonderer Berücksichtigung von Mensch, Technik und Organisation* (S. 281-298). Zürich: vdf-Hochschulverlag an der ETH.

Ringelmann, M. (1913). Recherches sur les moteurs animés. Travail de l'homme. *Annales de l'Institut National Agrovomique, 2e série-tome XII* 1-40.

Robbins, J. (2004). Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools. In J. Feller, B. Fitzgerald, S. Hissam & K. R. Lakhani (Hrsg.), *Making Sense of the Bazaar: Perspectives on Open Source and Free Software*. Cambridge, MA, US: MIT Press.

Roberts, J., Hann, I.-H. & Slaughter, S. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science, 52*, 984-999.

Roberts, K. H. & Glick, W. (1981). The Job Characteristics Approach to Task Design: A critical Review. *Journal of Applied Psychology, 66*(2), 193-217.

Robles, G. (2004). A Software Engineering approach to Libre Software. In R. A. Gehring & B. Lutterbeck (Hrsg.), *Open Source Jahrbuch 2004* (S. 193-208). Berlin: Lehmanns Media.

Robles, G. (2007, 16. Februar). *Debian Counting.* Verfügbar unter: http://libresoft.dat.escet.urjc.es/debian-counting/

Rosenberg, R. D. & Rosenstein, E. (1980). Participation and productivity: An empirical study. *Industrial and Labor Relations Review, 33*, 355-367.

Rosenbladt, B. v. (2000). *Freiwilliges Engagement in Deutschland. Ergebnisse der Repräsentativerhebung 1999 zu Ehrenamt, Freiwilligenarbeit und bürgerschaftlichem Engagement*. Stuttgart: Kohlhammer.

Rosenstein, E. (1977). Worker participation in Israel: experience and lessons. *The Annals, 431*, 113-112.

Rosenstiel, L., Falkenberg, T., Hehn, W., Henschel, E. & Warns, I. (1982). *Betriebsklima heute*. München: Bayerisches Staatsministerium für Arbeit und Sozialordnung.

Rossi, M. A. (2004). *Decoding the "Free/Open Source (F/OSS) Software Puzzle"a survey of theoretical and empirical contributions*. Verfügbar unter: http://opensource.mit.edu/papers/rossi.pdf [08.05.2004 2004].

Rothfuss, G. J. (2002). *A Framework for Open Source Projects*. Master Thesis, Universität Zürich, Zürich.

Rothschild-Whitt, J. (1979). The collectivist organization: an alternative to rational-bureaucratic models. *American Sociological Review, 44*, 509-527.

Rousseau, D. M. (1978). Measures of technology as predictors of employee attitude. *Journal of Applied Psychology, 63*, 213-218.

Royce, W. W. (1970, August 1970). *Managing the Development of Large Software Systems.* Paper presented at the Proceedings of IEEE WESCON, Los Angeles, Reprinted in Proceedings of the Ninth International Conference on Software Engineering, Pittsburgh, PA, USA, ACM Press, 1989, pp. 328-338.

Ruf, W. & Fittkau, T. (2007). *Ganzheitliches It-projektmanagement: Wissen-Praxis-Anwendungen*. München: Oldenbourg Wissenschaftsverlag.

Rummel, A. & Feinberg, R. (1988). Cognitive Evaluation Theory: A Meta-Analytic Review of the Literature. *Social Behavior and Personality, 16*(2), 147-164.

Ryan, R. M. & Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology, 25*, 54-67.

Sachs, L. (1999). *Angewandte Statistik. Anwendung statistischer Methoden* (9.). Berlin: Springer.

Salancik, G. R. & Pfeffer, J. (1978). A Social Information Processing Approach to Job Attitudes and Task Design. *Administrative Science Quaterly, 23*, 224-253.

Samoladas, I. & Stamelos, I. (2005). *Assessing Free/Open Source Software Quality*. Verfügbar unter: http://opensource.mit.edu/papers/samoladasstamelos.pdf

Santana, M. & Robey, D. (1994). *Controlling systems development: effects on job satisfaction of systems professionals.* Paper presented at the Proceedings of the 1994 computer personnel research conference on Reinventing IS Alexandria, Virginia, United States.

Scacchi, W. (1984). Managing Software Engineering Projects: A Social Analysis. *IEEE Transactions on Software Engineering, 10*(1), 49-59.

Scacchi, W. (1995). Understanding Software Productivity. In W. D. Hurley (Hrsg.), *Advances in Software Engineering and Knowledge Engineering* (Bd. 4, S. 273-316). Singapore: World Scientific.

Scacchi, W. (2002). *Is Open Source Software Development Faster, Better, and Cheaper than Software Engineering?* Paper presented at the ICSE'02 International Confer-

ence on Software Engineering - 2nd Workshop on Open Source Software Engineering, Orlando, FL, USA.

Schachtner, C. (1994). Von der Notwendigkeit einer subjektorientierten Arbeitsgestaltung in der Softwareproduktion. *Zeitschrift für Arbeitswissenschaft, 48*(4), 193-197.

Schaefer, C., Coyne, J. C. & Lazarus, R. S. (1981). The health-related functions of social support. *Journal of Behavioral Medicine, 4*(4), 381-406.

Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. London.: Chapman & Hall.

Schief, S. (2004). *Jahresarbeitszeiten als Standortindikator? Hintergründe zur fragwürdigen Nutzung internationaler Vergleiche* (No. IAT-Report, Nr. 2004-03). Gelsenkirchen: Inst. Arbeit und Technik.

Schmidt, K.-H. & Kleinbeck, U. (1999). Job Diagnostic Survey (JDS - deutsche Fassung). In H. Dunckel (Hrsg.), *Handbuch psychologischer Arbeitsanalyseverfahren* (S. 205-227). Zürich: vdf-Hochschulverlag.

Schmitt, N. (1994). Method Bias: The Importance of Theory and Measurement. *Journal of Organizational Behavior, 15*(5), 393-398.

Schmitz, J. & Fulk, J. (1991). Organizational Colleagues, Media Richness, and Electronic Mail. A Test of the Social Influence Model of Technology Use. *Communication Research, 18*(4), 487-523.

Schneidewind, U., Landsberger, M. & Eggers, H. (2002). Mythos Linux? Zur Übertragbarkeit der Koordinations- und Anreizmechanismen der Linux-Entwicklung auf Unternehmen. *Zeitschrift Führung + Organisation, 71*(4), 226-233.

Schroer, J. & Hertel, G. (2009). Voluntary engagement in an open web-based encyclopedia: Wikipedians, and why they do it. *Media Psychology, 12*(1), 96-120.

Schuler, H. (1995). *Lehrbuch Organisationspsychologie*. Göttingen: Huber.

Schumann, M. (1998). New Concepts of Production and Productivity. *Economic and Industrial Democracy, 19*(1), 17-32.

Scott, C. (1961). Research on Mail Surveys. *Journal of the Royal Statistical Society. Series A (General), 124*(2), 143-205.

Scott, C. R. (1997). Identification with Multiple Targets in a Geographically Dispersed Organization. *Management Communication Quarterly, 10*(4), 491-522.

Scott, W. E., Jr. (1966). Activation Theory and Task Design. *Organization Behavior and Human Performance, 1*, 3-30.

Searfoss, D. G. & Monczka, R. (1973). Perceived Participation in the Budget Process and Motivation to Achieve the Budget. *Academy of Management Journal, 16*(541-554).

Seeman, T. & Berkman, L. (1988). Structural Characterisitics of Social Networks and Their Relationsships with Social Support in the Elderly. *Social Science & Medicine, 26*(7), 737-749.

Seers, A. & Graen, G. B. (1984). The dual attachment concept: A longitudinal investigation of the combination of task characteristics and leader-member exchange. *Organizational Behavior and Human Performance, 33*, 283-306.

Sein, M. K. & Bostrom, R. P. (1991). *A psychometric study of the job characteristics scale of the job diagnostic survey in an MIS setting.* Paper presented at the Special Interest Group on Computer Personnel Research Annual Conference. Proceedings of the 1991 conference on SIGCPR, Athens, Georgia, United States.

Selig, J. (1986). *EDV-Management: Eine empirische Untersuchung der Entwicklung von Anwendungssystemen in deutschen Unternehmen*: Berlin: Springer.

Semmer, N., Zapf, D. & Dunckel, H. (1999). Instrument zur Stressbezogenen Tätigkeitsanalyse. In H. Dunckel (Hrsg.), *Handbuch psychologischer Arbeitsanalyseverfahren* (S. 179-204). Zürich: vdf-Hochschulverlag.

Settoon, R. P., Bennett, N. & Liden, R. C. (1996). Social exchange in organizations: Perceived organizational support, leader-member exchange, and employee reciprocity. *Journal of Applied Psychology, 81*(3), 219-227.

Shah, S. (2003). *Understanding the Nature of Participation & Coordination in Open and Gated Source Software Development Communities.* Verfügbar unter: http://opensource.mit.edu/papers/shah3.pdf

Shaikh, M. & Cornford, T. (2003). *Version Management Tools: CVS to BK in the Linux Kernel.* Paper presented at the ICSE'03 International Conference on Software Engineering - 3rd Workshop on Open Source

Software Engineering, Portland, Oregan.

Shalley, C. E., Gilson, L. L. & Blum, T. C. (2000). Matching creativity requirements and the work environment: Effects on satisfaction and intentions to leave. *Academy of Management Journal, 43*(2), 215-223.

Shepard, T., Lamb, M. & Kelly, D. (2001). More testing should be taught. *Communications of the ACM, 44*(6), 103-108.

Shepperd, M. & Ince, D. (1993). *Derivation and Validation of Software Metrics.* Oxford, USA: Oxford University Press.

Shneiderman, B. (1980). Group Processes in Programming. *Datamation, 26*, 138-140.

Shore, L. M. & Wayne, S. J. (1993). Commitment and employee behavior: comparison of affective commitment and continuance commitment with perceived organizational support. *Journal of Applied Psychology, 78*(5), 774-780.

Sillitti, A. & Succi, G. (2005). Agility and Libre Software Development. *Upgrade - The European Journal of the Informatics Professional, 6*(3), 33-37.

Simon, B., Loewy, M., Stürmer, S., Weber, U., Freytag, P., Habig, C.et al. (1998). Collective identification and social movement participation. *Journal of Personality and Social Psychology, 74*, 646-658.

Sims, H.-P., Szilagyi, A. D. & Keller, R. T. (1976). The measurement of Job Characteristics. *Academy of Management Journal, 19*(2), 195-212.

Sisson, K. (2000). *Direct Participation and the Modernisation of Work Organisation*: European Foundation for the Improvement of Living and Working Conditions.

Smith, C. A., Organ, D. W. & Near, J. P. (1983). Organizational citizenship behavior: Its nature and antecedents. *Journal of Applied Psychology, 68*, 653-663.

Smits, S. J., Tanner, J., R. & McLean, E., R. (1993). *Job characteristic preference-reality discrepancies and the job and career attitudes of I/S professionals.* Paper pre-

sented at the Proceedings of the 1993 conference on Computer personnel research, St Louis, Missouri, United States.

Software-Engineering – Auch freie Projekte folgen festen Regeln. Open Source: Chaos oder strukturierte Entwicklung? (2001). *Computerwoche, 10,* 120-121.

Software-Entwickler. (2006, 21. November 2006). Verfügbar unter: http://de.wikipedia.org/w/index.php?title=Software-Entwickler&oldid=23565317 2006].

Solheim, J. & Rowland, J. (1993). An empirical study of testing and integration strategies using artifical software systems. *IEEE Transactions on Software Engineering, 19*(10), 941-949.

Sommerville, I. (1989). *Software Engineering*. Boston: Addison-Wesley.

Sonnentag, S. (1994). Streß in SE-Projekten. In F. C. Brodbeck & M. Frese (Hrsg.), *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung* (S. 51-67). München: Oldenbourg.

Sonnentag, S., Brodbeck, F. C., Heinbokel, T. & Stolte, W. (1994). Stressor-burnout relationship in software development teams. *Journal of Occupational and Organizational Psychology, 67*(4), 327-341.

Spaeth, S. (2003, 01.04.2003). *Decision-Making in Open Source Projects*. Verfügbar unter: http://sspaeth.org/paper/Vorstudie.pdf [08.05.2004 2004].

Spears, R. M., Lea, M. & Lee, S. (1990). De-individuation and group polarization in computer-mediated comminication. *British Journal of Social Psychology, 29*, 121-134.

Spector, P. E. (1985). Higher-order need strength as a moderator of the job scope-employee outcome relationship: A meta-analysis. *Journal of Occupational psychology, 58*(2), 119-127.

Spector, P. E. (1986). Perceived control by employees: A meta-analysis of studies concerning autonomy and participation at work. *Human Relations, 39*, 1005-1016.

Spieß, M. (2008). *Missing Data: Analyse von Daten mit fehlenden Werten*. Berlin-Hamburg-Münster: LIT Verlag.

Spolsky, J. (2005, 17.08.2005). *The Project Aardvark Spec*. [HTML]. Verfügbar unter: http://www.joelonsoftware.com/articles/AardvarkSpec.html [08.03.2006 2006].

Stallman, R. (1999). The GNU Operating System and the Free Software Movement. In C. DiBona, S. Ockman & M. Stone (Hrsg.), *Open Sources: Voices from the Open Source Revolution*. Cambridge, Massachusetts: O'Reilly and Associates.

Stallman, R. M. (2002). *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston, Massachusetts: GNU Press.

Stamelos, I., Angelis, L., Oikonomu, A. & Bleris, G. L. (2002). Code Quality Analysis in Open-Source Software Development. *Information Systems Journal, 12*(1), 43-60.

Stark, J. (2002). Peer reviews as a quality management technique in open-source software development projects. In J. Kontio & R. Conradi (Hrsg.), *Software Quality - ECSQ 2002* (S. 340–350.). Heidelberg: Springer.

Stewart, K. J. & Gosain, S. (2003). *Impacts of Ideology, Trust, and Communication on effectiveness in Open Source Software Development Teams*. Verfügbar unter: opensource.mit.edu/papers/stewartgosain2.pdf

Stolte, W. & Heinbokel, T. (1993). Gibt es einen Zusammenhang zwischen der Werkzeugunterstuetzung und Merkmalen der Arbeitssituation in der Softwareentwicklung? In A. Gebert & U. Winterfeld (Hrsg.), *Arbeits-, Betriebs- und Organisationspsychologie vor Ort. Bad Lauterberg 1992* (S. 83-90). Bonn: Deutscher Psychologen Verlag.

Stone, E. F. (1986). Job scope-job satisfaction and job scope-job performance relationships. In E. A. Locke (Hrsg.), *Generalizing from laboratory to field settings* (S. 189-206). Lexington, MA: Lexington Books.

Stone, E. F. & Gueutal, H. G. (1985). An empirical derivation of the dimensions along which characteristics of jobs are perceived. *Academy of Management Journal, 28*(2), 376-396.

Strauss, G. (1977). Managerial Practices. In J. R. Hackman & J. L. Suttle (Hrsg.), *Improving Life at Work* (S. 297-363). Santa Monica, CA: Good-Year.

Streker-Seeborg, I. (1978). The Influence of Employee Participation in Job Redesign. *Journal of Applied Behavioral Science, 14*(1), 87-98.

Stukas, A. A., Snyder, M. & Clary, E. G. (1999). The effects of "mandatory volunteerism" on intentions to volunteer. *Psychological Science, 10*(1), 59-64.

Suh, K. S. (1999). Impact of communication medium on task performance and satisfaction: an examination of media-richness theory. *Information & Management, 35*, 295-312.

Taber, T. D. & Taylor, E. (1990). A review and evaluation of the psychometric properties of the job diagnostic survey. *Personnel Psychology, 43*(3), 467-500.

Tang, S. & Hall, V. C. (1995). The overjustification effect: a meta-analysis. *Applied Cognitive Psychology, 9*(5), 365-404.

Tannenbaum, A. S., Kavcic, B., Rosner, M., Vianello, M. & Wieser, G. (1974). *Hierarchy in organizations: an international comparison*. San Francisco: Jossey-Bass.

TeX. (2006, 15. Februar). Verfügbar unter: http://en.wikipedia.org/wiki/Tex 2006].

Thatcher, J. B., Liu, Y. & Stepina, L. P. (2002a). *The role of the work itself: an empirical examination of intrinsic motivation's influence on IT workers attitudes and intentions.* Paper presented at the ACM SIGCPR conference on Computer personnel research, Kristiansand, Norway.

Thatcher, J. B., Stepina, L. P. & Boyle, R. J. (2002b). Turnover of information technology workers: Examining empirically the influence of attitudes, job characteristics, and external markets. *Journal of Management Information Systems, 19*(3), 231-261.

Thomas, J. & Griffin, R. (1983). The social information processing model of task design: A review of the literature. *Academy of Management Review, 8*, 672-682.

Thomas, K. W. & Velthouse, B. A. (1990). Cognitive elements of empowerment: An "interpretive" model of intrinsic task motivation. *Academy of Management Review, 15*(4), 666-681.

Thompson, J. D. (1967). *Organizations in action*. New York: McGraw-Hill.

Tiegs, R. B., Tetrick, L. E. & Fried, Y. (1992). Growth Need Strength and Context Satisfactions as Moderators of the Relations of the Job Characteristics Model. *Journal of Management, 18*(3), 575-593.

Todd, S. Y. (2004). *A causal model depicting the influence of selected task and employee variables on organizational citizenship behavior*. The Florida State U., US.

Torvalds, L. (1999). The Linux edge. *Communications of the ACM, 42*(4), 38-39.

Torvalds, L. & Diamond, D. (2001). *Just for Fun: The Story of an Accidental Revolutionary*. New York: HarperCollins.

Tosi, H. (1970). A Re-Examination of Personality as a Determinant of the Effects of Participation. *Personnel psychology, 23*, 91-99.

Trevino, L. K. & Webster, J. (1992). Flow in Computer-Mediated Communication. Electronic Mail and Voice Mail Evaluation and Impacts. *Communication Research, 19*(5), 539-573.

Triebe, J. K. (1980). *Untersuchung zum Lernprozess während des Erwerbs der Grundqualifikation (Montage eines kompletten Motors). Arbeits- und sozialpsychologische Untersuchung von Arbeitsstrukturen im Bereich der Aggregatefertigung der Volkswagen AG*. Bonn: BMFT 1980, HA 80-019.

Triebe, J. K. (1981). *Aspekte beruflichen Handelns und Lernens*. Phil. Diss.

Triebe, J. K. & Ulich, E. (1977). Eignungsdiagnostische Zukunftsperspektiven: Möglichkeiten einer Neuorientierung. In J. K. Triebe & E. Ulich (Hrsg.), *Beiträge zur Eignungsdiagnostik* (Bd. 19, S. S. 241-273). Bern: Huber.

Trist, E. L. & Bamforth, K. W. (1951). Some social and psychological consequences of the longwall method of coal-getting. *Human Relations, 4*(1), 3-38.

Trist, E. L., Higgins, G. W., Murray, H. & Pollock, A. B. (1963). *Organizational choice*. London: Tavistock.

Trist, E. L., Susman, G. & Brown, G. W. (1977). An experiment in autonomous group working in an American coal mine. *Human Relations, 30*, 201-236.

Tuomi, I. (2000). Internet, innovation, and open source: Actors in the network. *Firstmonday, 6*(1).

Tuomi, I. (2004, 24.10.2009). *Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research*. Verfügbar unter: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1151/1071 [6 9].

Turner, A. N. & Lawrence, P. R. (1965). *Industrial jobs and the worker: An investigation of response to task attributes*. Boston: Harvard Graduate School of Business Administration.

Turner, C. F., Ku, L., Rogers, S. M., Lindberg, L. D., Pleck, J. H. & Sonenstein, F. L. (1998). Adolescent Sexual Behavior, Drug Use, and Violence: Increased Reporting with Computer Survey Technology. *Science, 280*(5365), 867-873.

Tyagi, P. K. & Wotruba, T. R. (1993). An exploratory study of reverse causality relationships among sales force turnover variables. *Journal of the Academy of Marketing Science, 21*(2), 143-153.

Udris, I. (1987). Soziale Unterstützung, Stress in der Arbeit und Gesundheit. In H. Keupp & B. Röhrle (Hrsg.), *Soziale Netzwerke* (S. 123-138). Frankfurt a. M.: Campus Verlag.

Udris, I. (2001). *Mitarbeiterbefragung und betriebliches Gesundheitsmanagement – SALSA, ein Instrument für die Praxis.* Paper presented at the Tagungsbericht des 6. Informationstages zur BGF, St. Pölten.

Udris, I. & Rinmann, M. (1999). SAA und SALSA: Zwei Fragebogen zur subjektiven Arbeitsanalyse. In H. Dunckel (Hrsg.), *Handbuch psychologischer Arbeitsanalyseverfahren* (S. 397-419). Zürich: vdf-Hochschulverlag.

Ulich, E. (1978). Über das Prinzip der differentiellen Arbeitsgestaltung. *Industrielle Organisation, 47*(12), 566-568.

Ulich, E. (1983). Differentielle Arbeitsgestaltung - ein Diskussionsbeitrag. *Zeitschrift für Arbeitswissenschaft, 37*, 12-15.

Ulich, E. (1987). Zur Frage der Individualisierung von Arbeitstätigkeiten unter Berückichtigung der Mensch-Computer-Interaktion. *Zeitschrift für Arbeits- und Organisationspsychologie, 31*(3), 86-93.

Ulich, E. (1990). Individualisierung und differentielle Arbeitsgestaltung. In C. Hoyos & B. Zimolong (Hrsg.), *Ingenieurpsychologie* (S. 511-535). Göttingen: Hogrefe.

Ulich, E. (1998). *Arbeitspsychologie* (4). Stuttgart: Schäffer-Poeschel.

Ulich, E., Groskurth, P. & Bruggemann, A. (1973). *Neue Formen der Arbeitsgestaltung. Möglichkeiten und Probleme einer Verbesserung der Qualität des Arbeitslebens.* Frankfurt/Main: Europäische Verlagsanstalt.

Umstot, D. D., Mitchell, T. R. & Bell Jr, C. H. (1978). Goal Setting and Job Enrichment: An Integrated Approach to Job Design. *The Academy of Management Review, 3*(4), 867-879.

V-Modell. (2009, 11. Oktober). Verfügbar unter: http://de.wikipedia.org/wiki/V-Modell 2009].

Valacich, J. S., Mennecke, B. E., Wachter, R. M. & Wheeler, B. C. (1994). *Extensions to media richness theory: a test of task-media fit hypothesis.* Paper presented at the Twenty-Seventh Annual Hawaii International Conference on System Science.

Van der Zwaan, A. H. & Molleman, E. (1998). Self-organizing groups: conditions and constraints in a sociotechnical perspective. *International Journal of Manpower, 19*(5), 301-318.

Van Dick, R. (2003). *Commitment und Identifikation mit Organisationen.* Göttingen: Hogrefe.

Van Dyne, L., Graham, J. W. & Dienesch, R. M. (1994). Organizational Citizenship Behavior: Construct Redefinition, Measurement, and Validation. *Academy of Management Journal, 37*(4), 765-802.

Vance, R. J. & Biddle, T. F. (1985). Task experience and social cues: interactive effects on attitudinal reactions. *Organizational behavior and human decision processes, 35*(2), 252-265.

Vat, K. H. (2000). *Training e-commerce support personnel for enterprises through action learning.* Paper presented at the Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research, Chicago.

Verbeke, W., Belschak, F., Wuyts, S. & Bagozzi, R. P. (2004). Account Managers Creation of Social Capital: Communal and Instrumental Investments and Performance Implications. *ERIM Report Series Research in Management, January*, 42.

Vixie, P. (1999). Software Engineering. In C. DiBona, S. Ockman & M. Stone (Hrsg.), *Open Sources: Voices from the Open Source Revolution.* Cambridge, Massachusetts: O'Reilly and Associates.

Vogt, W., Hofmann, W. & Zülch, G. (2000). Differenzielle Arbeitsgestaltung. *Zeitschrift für Unternehmensentwicklung und Industrial Engineering, 49*(5), 257-262.

Vondernach, G. (1980). Die "neuen Selbständigen" - Zehn Thesen zur Soziologie einers unvermuteten Phänomens. *Miteilungen aus der Arbeitsmarkt- und Berufsforschung, 2*, 153-169.

Vroom, V. H. (1959). Some personality determinants of the effects of participation. *Journal of Abnormal and Social Psychology, 59*, 322-327.

Wageman, R. (2001). The meaning of interdependence. In M. E. Turner (Hrsg.), *Groups at work: Theory and research* (S. 197-218). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Wall, T. D., Jackson, P. R. & Davids, K. (1992). Operator work design and robotics system performance: A serendipitous field study. *Journal of Applied Psychology, 77*(3), 353-362.

Wall, T. D., Jackson, P. R. & Mullarkey, S. (1995). Further evidence on some new measures of job control, cognitive demand and production responsibility. *Journal of Organizational Behavior, 16*(5), 431-455.

Wall, T. D. & Martin, R. (1994). Job and work design. In I. T. Robertson & C. L. Cooper (Hrsg.), *Key reviews in managerial psychology: Concepts and research for practice* (S. 158-188). Oxford, England: John Wiley & Sons.

Walther, J. B. (1992). Interpersonal Effects in Computer-Mediated Interaction: A Relational Perspective. *Communication Research, 19*, 52-90.

Walther, J. B. (1996). Computer-mediated communication: Impersonal, interpersonal, and hyperpersonal interaction. *Communication Research, 23(1)*, Mrz 44.

Walther, J. B., Anderson, J. F. & Park, D. W. (1994). Interpersonal Effects in Computer-Mediated Interaction. A Meta-Analysis of Social and Antisocial Communication. *Communication Research, 21*, 460-487.

Walton, R. E. (1977). Work innovations at Topeka: After six years. *Journal of Applied Behavioral Science, 13*, 422-433.

Walz, D. B., Elam, J. J. & Curtis, B. (1993). Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of the ACM, 36*(10), 63-77.

Warner, W. K. (1964). Attendance and division of labor in voluntary associations. *Rural Sociology, 29*, 396-407.

Warner, W. K. & Heffernan, W. D. (1967). The benefit-participation contingency in voluntary farm organizations. *Rural Sociology, 32*, 139-154.

Warr, P. (1987). *Work, Unemployment, and Mental Health*. Oxford: Clarendon Press.

Warr, P. B. (1990). The measurement of well-being and other aspects of mental health. *Journal of Occupational Psychology, 63*, 193-210.

Wasko, M. M. & Faraj, S. (2000). "It Is What One Does": Why People Participate and Help Others in Electronic Communities of Practice. *Journal of Strategic Information Systems, 9*(2-3), 155-173.

Wayner, P. (2000). *Free For All: How Linux and the Free Software Movement Undercuts the High-Tech Titans*. New York: HarperBusiness.

Webster, J., Trevino, L. K. & Ryan, L. (1993). The dimensionality and correlates of flow in human-computer interactions. *Computers in Human Behavior, 9*(4), 411-426.

Wehner, T. & Güntert, S. T. (2005). *Commitment und Involvement in der Freiwilligenarbeit.* Paper presented at the Gesellschaft für Arbeitswissenschaft (2005). Personalmanagement und Arbeitsgestaltung. Bericht zum 51. Kongress der Gesellschaft für Arbeitswissenschaft, Heidelberg.

Weinberg, G. M. (2004). *Die Psychologie des Programmierers*. Bonn: Mitp-Verlag.

Wellman, B. (1992). Which Types of Ties and Networks Give What Kinds of Social Support? *Advances in Group Processes, 9*, 207-235.

Wellman, B. & Gulia, M. (1999). The network basis of social support: A network is more-than the sum of its ties. In B. Wellman (Hrsg.), *Networks in the global village: Life incontemporary communities* (S. 83-118). Boulder, CO: Westview.

Weltz, F. & Ortmann, R. G. (1992). *Das Softwareprojekt: Projektmanagement in der Praxis*. Frankfurt/M: Campus.

Wenger, E. (1998). *Communities of Practice: Learning Meaning, and Identity*. Cambridge, Massachusetts: Cambridge University Press.

White, J. K. & Ruh, R. A. (1973). Effects of Personal Values on the Relationship Between Participation and Job Attitudes. *Administrative science quarterly, 18*(4), 506-514.

White, S. E. & Mitchell, T. R. (1979). Job enrichment versus social cues: A comparison and competitive test. *Journal of Applied Psychology, 64*(1), 1-9.

Wiegers, K. (2002). *Peer Reviews in Software: A Practical Guide*. Boston, MA: Addison-Wesley.

Wieland-Eckelmann, R., Baggen, R., Saßmannshausen, A., Schmitz, U., Schwarz, R., Ademmer, C.et al. (1996). *Gestaltung beanspruchungsoptimaler Bildschirmarbeit. Grundlagen und Verfahren für die Praxis*. Bremerhaven: Wirtschaftsverlag NW.

Wieland-Eckelmann, R., Saßmannshauen, A., Rose, M. & Schwarz, R. (1999). Synthetische Beanspruchungsanalyse SynBA-GA. In H. Dunckel (Hrsg.), *Handbuch psychologischer Arbeitsanalyseverfahren* (S. 421-463). Zürich: vdf.

Wieland, R., Klemens, S., Scherrer, K., Timm, E. & Krajewski, J. (2004). Moderne IT-Arbeitswelt gestalten - Anforderungen, Belastungen und Ressourcen in der IT-Branche. *Veröffentlichungen zum Betrieblichen Gesundheitsmanagement der TK, 4*.

Wieland, R., Metz, A.-M. & Richter, P. (2002). *Call Center auf dem arbeitspsychologischen Prüfstand. Teil 2: Arbeitsgestaltung im Call Center – Belastung, Beanspruchung und Ressourcen*. Hamburg: Verwaltungs-Berufsgenossenschaft.

Wiersma, U. J. (1992). The effects of extrinsic rewards in intrinsic motivation: a meta-analysis. *Journal of occupational and organizational psychology, 65*(2), 101-114.

Wiertz, C., de Ruyter, K. & Streukens, S. (2003). On The Role Of Normative Influences In Commercial Virtual Communities. *Research Memoranda from Maastricht: METEOR, 38*.

Wiesenfeld, B. M., Raghuram, S. & Garud, R. (1999). Communication patterns as determinants of organizational identification in a virtual organization. *Organization Science, 10*(6), 777-790.

Wiesenfeld, B. M., Raghuram, S. & Garud, R. (2001). Organizational identification among virtual workers: the role of need for affiliation and perceived work-based social support. *Journal of Management, 27*, 213-229.

Williams, L. J., Gavin, M. B. & Williams, M. L. (1996). Measurement and nonmeasurement processes with negative affectivity and employee attitudes. *Journal of applied psychology, 81*(1), 88-101.

Wilson, G. (1999). Is the Open-Source Community Setting a Bad Example? *IEEE Software, 16*(1), 23-25.

Wirtz, M. & Nachtigall, C. (2002). *Wahrscheinlichkeitsrechnung und Inferenzstatistik. Statistische Methoden für Psychologen* (4 Bd. 2). Weinheim: Juventa.

Wong, C. & Campion, M. A. (1991). Development and test of a task level model of motivational job design. *Journal of Applied Psy¬chology, 76*, 825-837.

Wright, B. M. & Cordery, J. L. (1999). Production uncertainty as a contextual moderator of employee reactions to job design. *Journal of Applied Psychology, 84*, 456-463.

Xie, J. L. & Johns, G. (1995). Job Scope and Stress: Can Job Scope Be Too High? *The Academy of Management Journal, 38*(5), 1288-1309.

Ye, Y. & Kishida, K. (2003, May 3-10). *Toward an Understanding of the Motivation of Open Source Software Developers.* Paper presented at the International Conference on Software Engineering (ICSE2003), Portland, OR.

Yeung, A. B. (2004). The octagon model of volunteer motivation: Results of a phenomenological analysis. *Voluntas: International Journal of Voluntary and Nonprofit Organization, 15*, 21-46.

Zapf, D. (1991). Stressbezogene Arbeitsanalyse bei der Arbeit mit unterschiedlichen Bürosoftwaresystemen. *Zeitschrift für Arbeits- und Organisationspsychologie, 35*, 2-14.

Zapf, D. (1998). *Psychische Belastungen in der Arbeitswelt-ein Überblick.* Paper presented at the Symposium „Psychische Belastungen in der Arbeitswelt“ 11.November 1998.

Zimmer, M. & Wegener, J. (2006). Zuviel Wissen?! Überlegungen zu Stigmatisierung in Organisationen. In G. Krell & H. Wächter (Hrsg.), *Diversity Management: Impulse aus der Personalforschung* (S. 167-200). München/Mering: Hampp.

Zink, K. (1978). Zur Begründung einer zielgruppenspezifischen Organisationsentwicklung. *Zeitschrift für Arbeitswissenschaft, 32*, 42-48.

Zuckerman, M., Porac, J., Lathin, D. & Deci, E. L. (1978). On the Importance of Self-Determination for Intrinsically-Motivated Behavior. *Personality and Social Psychology Bulletin, 4*(3), 443-446.

Zülch, G. & Starringer, M. (1984). Differentielle Arbeitsgestaltung in Fertigungen für elektronische Flachbaugruppen. *Zeitschrift für Arbeitswissenschaft, 38*(4), 211-216.