Seminar „Software aus Komponenten“

Sommersemester 2005

Programmierern über die Schulter gesehen

Frederik Bässmann
fbaess@gmx.de
Advisor: Christopher Oezbek

27.08.2005

Zusammenfassung

Studien in der Softwareentwicklung stellen ein zentrales Thema in diesem Bereich dar. Speziell für die Entwicklung unterstützender Arbeitstools gewinnt diese Form von Recherchearbeit mit ihren unterschiedlichen Ansätzen zunehmend an Bedeutung.

Inhaltsverzeichnis

1 Einleitung - Motivation und Ziele von SE-Studien

Die Beobachtung von Softwareengineers ist ein relativ ausgeklügelter Bereich in der Softwareentwicklung, da durch sie erst eine effiziente Organisation und Anpassung dieser Entwicklungstätigkeit ermöglicht wird. Dies ist auch die Zielsetzung der Entwicklungsrecherchen, nämlich durch das Sammeln von Erfahrungen eine kontinuierliche und ökonomische Entwicklungsarbeit zu fördern.
Zu diesem Zweck werden Studien an der Arbeit von Softwareentwicklern durchgeführt, die in dieser Ausarbeitung vorgestellt und beschrieben werden sollen.
Um die Entwicklung von Software in der heutigen Zeit möglichst effizient und ökonomisch hinsichtlich Zeit- und Kostenaufwand zu gestalten, müssen die Entwicklungsprozesse ständig optimiert und durch geeignete Werkzeuge vereinfacht werden. Daher ist eines der Hauptziele dieser Studien die Entwicklung solcher Werkzeuge, im folgenden „Tools“ genannt, zur Unterstützung der Arbeit der Entwickler. Auch der Umgang der Entwickler mit diesen Tools ist Teil der Studien, da neben der eigentlichen Entwicklung der Projekte auch der Benutzerfreundlichkeit der Tools angemessene Beachtung zukommen muss. Ein zentraler Punkt dieser Studien und damit dieser Ausarbeitung besteht daher darin, Studien und Recherchen mit dem letztendlichen Ziel einer möglichst effizienten Toolentwicklung zu beschreiben “The goal of our research is to develop tools to help software engineers (SEs) more effectively maintain software.”[LA97].
Es ist vorteilhaft, solche Studien möglichst umfangreich zu gestalten, um umfassende Informationen über die fokussierten Arbeitsabläufe zu erhalten. Da sich die Studien allerdings zumeist als recht aufwändig erweisen, beschränkt man sich auf einzelne Unternehmen und beobachtet diese nach detailliert geplanten Richtlinien.
Bei der Durchführung dieser Recherchen gibt es unterschiedliche Ansätze. Es soll in dieser Ausarbeitung nach einer Beschreibung der Aufgaben von Softwareentwicklern und Programmierern ein Überblick über analytische, empirische und arbeitspraktische Studien und deren Organisation gegeben werden. Ferner soll in diesem Zusammenhang sowohl das Video-, das Notiz- als auch das „Shadowing“-Verfahren vorgestellt.
Schliesslich wird auf die Auswertung von aus Recherchen gewonnenen Daten und auf die Entwicklung eines Tools auf Grundlage solcher Informationen eingegangen werden.
Um die Studienarbeiten an Softwareprojekten anschaulich darzustellen, werden Beispiele aus den Papers „Studies of the Work Practices of Software Engineers“ und „An Examination of Software Engineering Work Practices“ hinzugezogen.

2 SE-Studien

Dieses Kapitel befasst sich sowohl mit den Tätigkeiten in der Softwareentwicklung als auch mit grundlegenden Sichtweisen in Bezug auf dortige Studien und deren Organisation.

2.1 Die Arbeit von Programmierern und Softwareentwicklern

Zunächst soll ein Einblick in die beruflichen Tätigkeiten von Softwareentwicklern und Programmierern gegeben und der Unterschied zwischen diesen beiden oft synonym verwendeten Begriffen hinsichtlich der Arbeitsbereiche aufgezeigt werden.
Ein Programmierer sieht sich im Laufe seiner Arbeit mit unterschiedlichen Arbeitsformen konfrontiert. Abgesehen davon, dass er sich mit verschiedenen Programmiersprachen auskennen und zumindest eine professionell beherrschen sollte, sind seine Tätig-

keiten nicht nur auf die Entwicklung von Sourcecode beschränkt, wenn man den Begriff Programmierer mit dem des Softwareentwicklers gleichgesetzt verstehen möchte. Ein „reiner Programmierer“ ist oftmals allerdings tatsächlich nur für das Verfassen von Code beziehungsweise Teilen davon verantwortlich und “meist wird vom Programmierer im Gegensatz zum Softwareentwickler keine universitäre Ausbildung verlangt”[ua]. Wenn man allerdings von einem Softwareentwickler spricht, versteht man unter dieser Bezeichnung jemanden, der nicht nur in der Lage ist, Programme in bestimmten Programmiersprachen zu schreiben, sondern darüber hinaus auch deren Entwicklung zu planen und zu organisieren.
Ein Team von Programmierern kann nur dann effizient zusammenarbeiten, wenn seitens des Managements eine gute Aufgabenverteilung und eine realistische Arbeitseinteilung gewährleistet wird. Außerdem sehen sich die Leiter eines Softwareentwicklungsprojekts auch mit den Aufgaben der Programmierer konfrontiert, auch wenn sie diese nicht bzw. nur teilweise selber bewältigen müssen. So muss auch ein Softwaremanager in der Lage sein, die Arbeit der ihm untergeordneten, als Programmierer beschäftigten Mitarbeiter nachzuvollziehen, um effizient weitere Arbeitsschritte planen und ihre Umsetzung gewährleisten zu können.
Bei der Umsetzung eines Softwareprojekts ist es zu Anfang, also noch bevor überhaupt mit der Implementierung begonnen wird, notwendig, einen genauen Zeitplan aufzustellen, welcher Meilensteine und selbstverständlich eine Deadline für die entgültige Fertigstellung des Projekts enthält. Es ist hierbei zu beachten, dass die Auftraggeber gleichzeitig oftmals die finanziellen Mittel zur Verfügung stellen, die ein solcher Arbeitsablauf fordert. Sollten also bestimmte Fristen nicht ordnungsgemäß eingehalten werden, so haben die Kunden durchaus die Möglichkeit und juristisch gesehen die Berechtigung, die Geldversorgung einzustellen, was zusätzlichen Druck und Stress bei der weiteren Projektumsetzung hervorruft und sich negativ auf das Arbeitsklima auswirken kann. Gleichzeitig ist es jedoch auch wichtig, den Mitarbeitern gewisse Freiheiten in der Wahl ihrer zu absolvierenden Aufgaben zu überlassen, um ihre Motivation und ihr Selbstvertrauen nicht negativ zu beeinflussen. So kann beispielsweise durch das Anlegen von „TODO“ - Listen eine Übersicht über die zu tätigenden Teilaufgaben angelegt werden, wodurch mehr Flexibilität in den Arbeitsalltag gebracht wird.
Um die Tätigkeit des Implementierens der geplanten Software nicht nur in Hinsicht auf die eigenen Arbeitsziele zufriedenstellend auszuführen, sondern auch im Sinne der Teamfähigkeit, sollten die Entwickler bei der Produktion von Quellcode nicht nur auf die Funktionalität, sondern auch auf die Nachvollziehbarkeit durch andere in den betreffenden Teilbereich des Projekts involvierte Mitarbeiter bedacht sein. In einem Team müssen die anderen Teammitglieder in der Lage sein, sich in die Arbeit ihrer Kollegen hineinzuversetzen, um effizient und ohne unnötigen erhöhten Einarbeitungsaufwand an bereits vorhandene Software anknüpfen bzw. diese an ihre eigene Arbeit anpassen zu können.
Nicht nur die Kenntnisse über die verwendete Programmiersprache, sondern auch die Berufserfahrung in einem bestimmten Arbeitsbereich sowie die Vertrautheit mit den jeweiligen Projekten, die sich über Jahre erstrecken können, spielen eine entscheidende Rolle. Daher ist es oftmals üblich, dass diejenigen Entwickler in den Bereich des Managements aufsteigen, die bereits langjährige Eindrücke im Bereich der Programmierung gesammelt haben. Auch ist Verständnis für den Umgang mit den unterstellten Teammitgliedern und Flexibilität hinsichtlich der Projektplanung vonnöten, da es selten vorkommt, das sich ein Projekt genau nach der ursprünglichen Planung und ohne unvorhergesehene Schwierigkeiten umsetzen lässt.
Somit wird ersichtlich, dass bei der Softwareentwicklung keineswegs nur die Implementierung, sondern noch eine Menge anderer Tätigkeiten in Betracht zu ziehen sind, um

anspruchsvolle Softwareprojekte zu realisieren.

2.2 Begutachtung von Programmierer-Arbeit: 'Empirical Studies' vs. 'Studies of Work Practices'

In diesem Abschnitt soll die Herangehensweise an die Begutachtung von Softwareentwicklung beschrieben werden, auf deren Grundlage später das Design und die Anforderungen für unterstützende Tools bei der Entwicklungsarbeit festgelegt werden. Die verwendeten Bezeichnungen „Empirical Studies“ und „Work Practices“ entstammen dem Paper „An Examination of Software Engineering Work Practices“.
Um die Arbeit von Programmierern durch entsprechende Tools effizient unterstützen zu können, ist es zunächst erforderlich, sich einen möglichst guten Überblick über die Aktivitäten von in der Softwareentwicklung tätigen Menschen und deren Arbeitsweise zu verschaffen. Dies stellt die Basis für eine spätere effiziente Entwicklung von Tools dar, um den Entwicklern bei ihrer Arbeit möglichst optimal entgegenkommen zu können. Hierfür gibt es verschiedene Verfahrensweisen. Neben analytischen Studien, die hauptsächlich auf logische Schlussfolgerungen anhand der gewonnen Daten ausgerichtet sind, stellen zwei andere Varianten die Durchführung von empirischen Studien („Empirical Studies“) und die Studie von Arbeitspraktiken („Work Practices“) dar. Die konventionelle und auch sehr häufig angewandte ist die empirische Vorgehensweise. Hierbei steht im Vordergrund, den mentalen Entwicklungsprozess von Programmiererarbeit zu analysieren und somit Rückschlüsse auf deren Bedürfnisse hinsichtlich unterstützender Arbeitstools zu ziehen. Diese Erkenntnisse sollen dann bei der Tool-Entwicklung herangezogen werden. Eine solche Vorgehensweise bei der Betrachtung von Programmier- und Softwareentwicklungstätigkeit ist durchaus sinnvoll und lässt ein gutes Grundverständnis bezüglich der Bedürfnisse der Entwickler zu. Nun hat dieses Verfahren allerdings einige Nachteile.
So werden oft experimentelle Studien an Programmierern durchgeführt, die nicht in jeder Hinsicht mit Entwicklern von realen Softwareprojekten vergleichbar sind, da es sich teilweise um erfahrene und gut ausgebildete Programmierer, teilweise aber auch um Anfänger wie zum Beispiel Studenten handelt, die bereits mit der Arbeit von größeren Projekten betraut werden. “It is not at all clear that these subjects accurately represent the population of industrial programmers.”[SLVA97] Es kann also nicht unbedingt von den Ergebnissen solcher Studien auf die Verhältnisse in der Softwareindustrie rückgeschlossen werden.
Sehr wichtig ist auch die Frage, wie viel Bedeutung dem Erfassen von Arbeitsschemen überhaupt für die sinnvolle Beschaffenheit von Arbeitshilfsmitteln beizumessen ist. “For instance, how does knowing that programmers will sometimes use a top-down strategy to understand code inform tool design?”[SLVA97] So ist nicht unbedingt davon auszugehen, dass die Kenntnis über die bloße Tatsache, dass bei der Programmierung von Entwicklern bestimmte Strategien und Arbeitsabläufe eingesetzt werden, automatisch zu einer Grundlage für die Erschließung der notwendigen Beschaffenheit des zu designenden Tools führt. Auch die Einbindung des neu zu erschaffenden Tools in die Arbeitsentwicklung steht hierbei sehr im Hintergrund der Studie. Der Programmierer soll nicht nur bei seinen gewohnten Arbeitsschritten unterstützt werden, sondern es soll ihm durch neue Tools auch die Angewöhnung neuer Methoden, die sich mit seinen Gewohnheiten verbinden lassen, ermöglicht werden. “The lack of tool adoption and use is a major problem in the area of tool design for software engineering.”[SLVA97]
Hinsichtlich dieser genannter Aspekte ist also über alternative Vorgehensweisen bei der Studie der Entwicklerarbeit nachzudenken. Eine solche Alternative stellt die Studie von

Arbeitsabläufen "Studies of Work Practices"[SLVA97] dar, bei der es sich um ein neueres Studienverfahren handelt. Hierbei wird nicht das mentale oder psychologische Prozessmodell von Arbeitsausführungen, sondern die Erfassung von simpleren Teilschritten bzw. ganz individuellen Teilbereichen, wie zum Beispiel das Lesen von Dokumentationen oder die Einarbeitung in Codestücke, in den Vordergrund gestellt. Somit lassen sich detaillierte Rückschlüsse über die Arten von Tätigkeiten, aus denen letzten Endes die o.g. Entwicklungsprozesse bestehen, über die Häufigkeit von deren Auftreten und die zeitlichen Faktoren bei deren Ausführung gewinnen. Prinzipielles Vorgehen bei dieser Studienvariante ist die Verfolgung der Programmiererarbeit und die detaillierte Erfassung der erwähnten Arten von Teilarbeiten, welche statistisch erfasst werden und später die Basis für die Toolentwicklung darstellen. Hiermit soll nicht nur zur Entwicklung eines bestimmten Tools beigetragen werden, sondern auch die Abwägung, was für ein Tool nach sorgfältiger Beobachtung und Interpretation der gewonnen Daten überhaupt benötigt wird, erleichtert werden.
Somit repräsentiert also die Durchführung von Studien nach dem Prinzip der Analyse von Arbeitspraktiken eine eher statistische Sichtweise bei SE-Studien, was die Informationsgewinnung für die spätere Verwertung angeht.

2.3 Organisation von Studien in der Softwareentwicklung

In diesem Abschnitt sollen die grundlegenden organisatorischen Ansätze bei Studien von Entwicklerarbeit behandelt werden.
Es muss zunächst Grundwissen über die Arbeitsweise der an einem Projekt beteiligten Softwareentwickler erlangt werden. Hierzu werden verschiedene Verfahren angewendet. Es sollen ein paar dieser Möglichkeiten anhand der im Paper „An Examination of Software Engineering Work Practices" beschriebenen Studie über die Arbeit von Programmierern, deren Hauptprojekt ein Telekommunikationssystem ist, vorgestellt und beschrieben werden.
Eine Methode besteht darin, einen Fragebogen aufzusetzen, anhand dessen die Mitarbeiter auf freiwilliger Basis erfassen können, wie viel zeitlichen Aufwand sie in einzelne Arbeitsprozesse investieren. Im Beispiel der o.g. Studie erwies sich die Einarbeit in Quellcode und das Lesen von Dokumentationen als die am meisten erfasste Aktivität, wobei allerdings das Korrigieren von Fehlern und die Weiterentwicklung des Systems als die beiden zeitintensivsten Arbeitsbereiche angesehen wurden. Da diese Erfassungen wie erwähnt völlig freiwillig von den Mitarbeitern getätigt wurden, kann davon ausgegangen werden, dass es sich um realistische Angaben handelt, da sich schließlich niemand unter zusätzlichen Arbeitsdruck gesetzt sehen konnte, was vielleicht zu unauthentischen Angaben geführt hätte. Nun liegt es auf der Hand, dass ein einzelner Fragebogen zur Repräsentation des Arbeitslebens nicht ausreichend sein kann. Aus diesem Grunde werden ebenso Studien an den Mitarbeitern in Form von Beobachtung durchgeführt.
So stellt eine weitere Methode die Beobachtung eines einzelnen Mitarbeiters dar, der über einen gewissen Zeitraum, beispielsweise etlichen Monaten, beobachtet wird. Dies geschieht anfänglich in geringen zeitlichen Abständen und später in größeren, also weniger häufig, um die Wiederholung von bestimmten Eindrücken möglichst gering zu halten. So können in den ersten Monaten wöchentliche, später alle paar Wochen Treffen mit den Reportern stattfinden, in denen sowohl die Geschehnisse der Zwischenzeit als auch der Entwicklungsstatus des Systems besprochen werden und zusätzlich noch eine kurze „Realtime" - Observation des Betreffenden bei der Ausführung seiner Tätigkeit stattfinden können.
Neben einzelnen Mitarbeitern ist es ebenso notwendig, auch Gruppen zu betrachten, was

auch unter mehreren Auswertungskriterien ablaufen kann. Diese reichen von dem Verfassen von Diagrammen, die das Verständnis der Mitarbeiter hinsichtlich des Systems widerspiegeln, über Interviews bezüglich durchgeführter Problemlösungen bis hin zu Beobachtungen der Gruppe bei der Arbeit. Hierbei ist es sinnvoll, Gruppen von Entwicklern auszuwählen, die einen unterschiedlichen Grad an Erfahrung haben bzw. unterschiedlich lange Zeit bereits in ihrer Firma tätig sind, um vielfältigere Rückschlüsse auf die Abhängigkeit der Arbeitsqualität von diesen Faktoren ziehen zu können. "Their experience ranged from the most expert member of the group (8 years) to the least experienced (6 months - recent college graduate)"[SLVA97]
Es gibt auch die Möglichkeit, die Benutzung von manchen Tools automatisch vom System erfassen zu lassen. Bei der o.g. Studie stellte sich beispielsweise heraus, dass die am häufigsten verwendeten Tools Suchtools waren, allerdings wurde die zeitliche Dauer nicht miterfasst. Diese Form der automatischen Datenerfassung ist nicht in jeder Hinsicht verlässlich, da oftmals nur das Öffnen der Anwendungen erfasst wird, die Benutzung aber weitaus häufiger erfolgen kann, wie beispielsweise bei Editoren. Diese werden nicht jedes Mal erneut, sondern nur einmalig geöffnet und dann nach Bedarf ab und zu benutzt.
Bei der Beobachtung von Softwareentwicklern während einer realen Entwicklung ist darauf zu achten, dass die Mitarbeiter sich nicht während ihrer Arbeit eingeschränkt fühlen. Somit sollte es vermieden werden, Beobachtungen in Zeiträumen durchzuführen, in denen eine Deadline zur Fertigstellung eines Projektes ansteht, um zusätzlichen Stress zu vermeiden, der die Studienergebnisse verfälschen könnte.

3 Recherche und Auswertung

In diesem Kapitel werden die Durchführung von Recherchen und deren Auswertung sowie die Vorgehensweise bei der Entwicklung von Tools auf Basis gewonnener Studiendaten erläutert.

3.1 Durchführung einer Recherche

Nachdem in den vorangegangenen Abschnitten auf die Ansätze und die allgemeinen Abläufe bei verschiedenen Formen von Datenerfassungen und Beobachtungen eingegangen wurde, soll in diesem genauer auf die Planung und Vorbereitung von Entwicklungsstudien eingegangen werden, die die genannten Aspekte beinhalten.
Wer eine Studie durchführt, muss sich im Vorfeld Gedanken darüber machen, wie er dabei genau vorgehen will. Sollen beispielsweise einzelne Personen oder Gruppen beobachtet werden, muss dies schließlich nach wohlüberlegten Gesichtspunkten geschehen.
Es ist nicht erstaunlich, dass sich die Durchführung von Beobachtungen in einem realen Arbeitsumfeld als eine sehr anspruchsvolle Aufgabe erweisen kann, da sie Umständen unterliegt, die leicht zu einer Überforderung der Recherchierenden führen können.
Es kann sich als sehr schwierig erweisen, stets alle interessanten oder möglicherweise relevanten Arbeitsabläufe zu protokollieren, und gleichzeitig zu entscheiden, welches Gewicht auf die jeweils auftretenden Eindrücke hinsichtlich ihrer Relevanz für spätere Auswertungen zu legen ist. Daneben muss drauf geachtet werden, dass nicht andere wichtige Details bei der Fokussierung auf bestimmte Einzelheiten vernachlässigt werden.
Im laufenden Betrieb führen die Mitarbeiter viele Einzelhandlungen sehr schnell und routiniert aus, sodass es nicht immer möglich ist, bei der Recherche überhaupt Schritt zu halten. Dies führt zu lückenhaften Protokollierungen, was die gesamten durch die Studie

gewonnen Eindrücke verfälschen kann.
Es muss bei einer Studie nicht nur festgehalten werden, welche Tätigkeiten ausgeübt werden, sondern es sollte prinzipiell auch nachvollziehbar sein, welche Ziele dabei anvisiert werden, denen die Arbeitschritte zugrunde liegen. Dies gilt besonders bei empirischen Studien, wo nicht nur Daten gesammelt, sondern vorrangig auch die jeweiligen Zusammenhänge erfasst werden müssen, um später noch ganze Arbeitsprozesse nachvollziehen zu können. Aber auch für die anderen genannten Verfahren, dem analytischen Ansatz und dem Betrachten von Arbeitspraktiken, ist eine konsistente Gewinnung von Studieninformationen ebenso von essentieller Bedeutung. Da die empirische Art der Studiendurchführung aber die bis heute am häufigsten verwendete ist, gewichten sich die folgenden Ausführungen daher auf diese Art der Recherche, sind aber grundlegend für alle Formen relevant.
Vor der eigentlichen Studie ist zunächst einmal zu entscheiden, in was für einem Arbeitsumfeld die Studie überhaupt durchgeführt werden soll. Es bestehen grundlegend zwei Möglichkeiten: ein künstliches oder ein reales Arbeitsumfeld, im Paper „Studies of the Work Practices of Software Engineers“ werden diese beiden Arbeitsumfelder als „laboratory environment“ und als „natural work environment“ bezeichnet, die Studie in einem natürlichen Umfeld wird als „field study“ bezeichnet.
Das Recherchieren in einem natürlichen Arbeitsumfeld hat den großen Vorteil, reale Arbeitssituationen studieren zu können “In order to improve software engineering tools and practice, it is therefore essential to conduct field studies, i.e. to study real practitioners as they solve real problems”[LSS05], birgt allerdings die Problematik, dass man mit einer Reihe nicht beeinflussbarer bzw. nicht kalkulierbarer Umstände konfrontiert wird, die die Qualität der Informationsgewinnung mitunter erheblich beeinflussen können. So ist zum Beispiel nicht unbedingt abzusehen, mit was für alltäglichen Hindernissen im laufenden Betrieb zu rechnen ist, schließlich darf dieser nicht durch die Recherchearbeiten behindert werden, die Recherchearbeiten sollten jedoch so effizient wie möglich durchgeführt werden können. Wenn beispielsweise nicht die Möglichkeit besteht, bestimmte Arbeitsabläufe zu dokumentieren, weil es aus Datenschutzgründen nicht erwünscht ist, würde dies eine Beeinträchtigung der Recherche bewirken. Es müssen im Vorfeld schon einzelne Personen ausgewählt werden, die bereit sind, mit den Reportern zu kooperieren, es muss genau abgesteckt werden, welche Entwickler für welchen Teil der Observierung geeignet erscheinen, wann sie welche Arbeiten tätigen und wann sie Zeit haben. Es muss außerdem geklärt werden, welche Projekte sich überhaupt für die Studie eignen, und welche Betriebe überhaupt interessant sind und für die Studie in Frage kommen.
Bei Studien in „laboratory environments“ hat man hingegen die Möglichkeit, sich die gewünschten Recherchebedingungen selber zu gestalten. Es kann ein individuelles Projekt ins Leben gerufen werden, dass speziell einer bestimmten Form von Studie angepasst ist. Man braucht sich keine Gedanken über die Gefährdung von realen und existentiell wichtigen Betriebsangelegenheiten zu machen, die mit der Recherche in Konflikt kommen könnten oder um die Verfügbarkeit von Personal, dass bereit ist, sich observieren zu lassen. Der große Nachteil ist allerdings, dass die Umgebung eben künstlich ist und somit nicht für Authentizität garantiert. “While the conclusions of laboratory studies are useful, they are not as likely to be relevant to industrial practice...”[LS02] Ferner spielt der Umfang und damit der Quellcode von verschiedenen Projekten eine entscheidende Rolle. Eine realistische Studie muss sich mit Größenordungen auseinandersetzen, die denen in der industriellen Softwareentwicklung angemessen sind, was allerdings bei kleineren Recherchen zum Beispiel im Rahmen von universitären Projekten nicht unbedingt zutrifft. Daher stellt sich bei solchen Studien die Frage, wie groß im Endeffekt der reale Nutzen für die Weiterverwertung bei der Toolentwicklung ist.

3.1.1 Methoden und Techniken der Informationsgewinnung

Es sollen im folgenden konkrete Verfahrensweisen der Informationserfassung innerhalb von Studien geschildert werden, unter anderem das spezielle 'synchronized shadowing'. Eine essentielle, bereits oft erwähnte und vielen Studienabschnitten zugrunde liegende Methode, bei einer Studie Informationen festzuhalten, ist die Observierung. Bei der Observierung werden einzelne Personen oder auch Gruppen direkt bei ihrer Arbeit beobachtet und permanent Informationen aufgezeichnet. Eine spezielle Form dieser Methode wird "shadowing"[LS02] genannt. Hierbei wird die zu observierende Person bei allen Tätigkeiten verfolgt und alles erfasst, was mit ihrer Arbeit zu tun hat. Dieses Methode birgt einige Schwierigkeiten für den Recherchierenden. Es ist zum einen nicht einfach, die Informationen überhaupt immer lückenlos zu erfassen, da ein permanenter Informationsfluss vorhanden ist, und zum anderen stellt die nachträgliche Auswertung der gewonnenen Daten eine weitere Schwierigkeit dar. Es ist also notwendig, die Daten vollständig aber auch möglichst übersichtlich und am besten vorstrukturiert aufzuzeichnen, um später auf einer guten Informationsgrundlage effizient weiterarbeiten zu können.
Zwei Techniken der Datenaufzeichnung sind das Führen von Notizen und die Aufzeichnung des Arbeitsprozesses auf Video. Beim Notieren stellt sich das Problem, dass permanent abgeschätzt werden muss, ob eine Information relevant ist oder nicht, da niemals alles manuell erfasst werden kann. Dies birgt die Gefahr, eigentlich nützliche Daten versehentlich zu vernachlässigen. Auch kann leicht etwas übersehen werden.
Beim späteren Auswerten eines Videos ist es wiederum umständlich, stundenlang ein viel zu großes Spektrum von Daten nachzuvollziehen."However, the process of coding can be very time consuming because one has much more data to work with."[LS02]
Um diese jeweiligen Nachteile auszugleichen, wurde ein Verfahren entwickelt, das es ermöglichen soll, die Vorzüge des Notiz und des Videoverfahrens gleichzeitig zu nutzen, ohne jedoch deren klassische Nachteile in Kauf nehmen zu müssen. Die Rede ist vom sog. "synchronized shadowing"[LS02].
Hierbei wird ein speziell entwickeltes Programm eingesetzt, dass eine gezielte Aufzeichnung von Notizen erleichtert. Das Programm läuft auf einem Notebook und ermöglicht die schnelle Notiz von als typisch berücksichtigten und eingepflegten Arbeitsschritten per Tastendruck, wobei auch der Zeitpunkt miterfasst wird. Um hierbei mehr Vollständigkeit und Flexibilität zu schaffen, wird die Observierung von zwei Beobachtern gleichzeitig mit synchronisierten Notebooks durchgeführt, daher auch die Bezeichnung „synchronized shadowing".
Wie auch die beiden konventionellen Verfahren der manuellen Notizsammlung und des Aufzeichnens auf Video ist auch das synchronized shadowing nicht vollkommen. Es lässt sich nicht immer genau festhalten, wann exakt welcher Arbeitsschritt des Programmierers auftrat, da immer etwas Zeit bis zur Protokollierung durch den Observierenden vergeht. Dies ist allerdings akzeptabel, da es sich meist nur um wenige Sekunden handelt.
„Synchronized shadowing" wurde von den Verfassern des Papers „Studies of the Work Practices of Software Engineers", Lethbridge und Singer, eigens entwickelt.

3.2 Auswertung von Studiendaten

Mindestens ebenso anspruchsvoll und aufwändig wie eine korrekte Recherche ist die Auswertung der gewonnen Daten hinterher. "It is very difficult to analyze data that result from observational studies."[LS02] Dieser Abschnitt behandelt diesen Bereich von Softwarestudien.
Nach der Sammlung von Daten während der Studie geht es danach daran, die gewon-

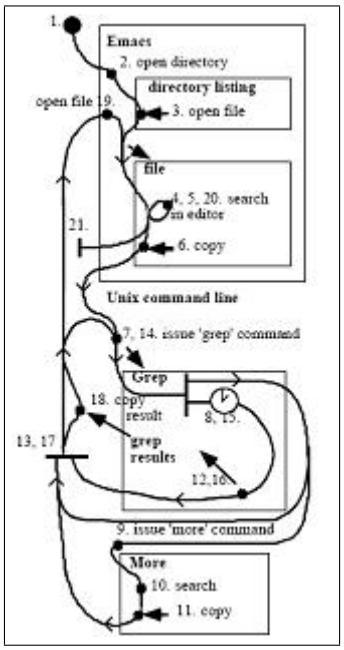

Abbildung 1: UCM-Beispiel, Quelle: Paper 'Studies of the Work Practices...'

nen Daten so durchzustrukturieren, dass sie auswertbar werden. Dies geschieht, indem man die Informationen vorinterpretiert und in bestimmte Bereiche einteilt, also nach Tätigkeiten kategorisiert. Dann ist es möglich, nach weiteren Methoden Auswertungen vorzunehmen. Eine Möglichkeit ist, Einzeltätigkeiten bestimmter Kategorien zu zählen bzw. ihre Gesamtzeit zu errechnen, eine andere besteht darin, unterschiedliche Arbeitsmuster heraus zu kristallisieren. Bei der ersten Vorgehensweise wird zunächst ein Überblick hinsichtlich der aufgewendeten Zeit und der ausgeübten Tätigkeiten heraus gearbeitet. Je mehr Personen bzw. Gruppen in unterschiedlichen Arbeitsfeldern beobachtet werden desto leichter lassen sich aus den angesammelten Daten Rückschlüsse über prinzipielle Unterschiede in diesen jeweiligen Arbeitsteilbereichen ziehen, sofern genug Daten gesammelt wurden.
Bei der Suche nach Arbeitsmustern wird nach ganz bestimmten Typen von Arbeitsschritten gesucht und diese mit den Zeitpunkten bzw. -abständen ihres Auftretens abgeglichen. Auch wird untersucht, inwiefern regelmäßig bestimmte Arbeitsschritte aufeinander folgen, aus denen dann Arbeitsprozesse abgeleitet werden können. Die beiden Varianten können getrennt eingesetzt werden oder sich auch gegenseitig ergänzen, je nachdem, ob eine gezielte Klassifizierung von Mitarbeitergruppen und ihren jeweiligen Spezialisierungen hinsichtlich ihrer Arbeitsweise oder eine Erschließung von Arbeitsmustern aus diesen Tätigleiten innerhalb von bestimmten Zeitspannen erschlossen werden soll.
Eine Möglichkeit dies geschickt durchzuführen besteht darin, bekannte Arbeitsprozesse in kleinere von beispielsweise etwa nur je 3 Schritten zu zerlegen und zielgerichtet nach diesen in den auszuwertenden Daten zu suchen. Diese Teilprozesse werden n-Gramme genannt, n bezeichnet die Zahl der Schritte.

3.2.1 Veranschaulichung von Arbeitsmustern - „Use Case Maps“ (UCMs)

Um die gefundenen Arbeitsmuster anschauliche darstellen zu können, wurde eine Technik entwickelt, diese Auswertungsergebnisse graphisch zu gestalten: die sog. „Use Case

Maps“, abgekürzt 'UCMs', die in diesem Teil behandelt werden sollen.
Diese Form von graphischer Notation wurde von Buhr und Casselmann 1996 ursprünglich zur Veranschaulichung von real-time-Abläufen von Programmen ersonnen, ist allerdings auch sehr geeignet, um humane Arbeitsprozesse darzustellen. So lässt sich ein typischer Arbeitsablauf, also ein Prozess, als Use Case Map widerspiegeln.
Es sollen kurz der grundlegende Aufbau und die Struktur eines typischen UCM-Diagramms anhand von Abbildung 11 beschrieben werden, in der „copy and paste“-Schritte eines Anwenders unter Verwendung eines Editors in UCM-Notation modelliert wurden: Der Arbeitsprozess verläuft an aufsteigend nummerierten Punkten von einem Start- zu einem Endpunkt, die im Modell durch eine durchlaufende Linie verbunden sind. Diese kann sich gabeln, Schleifen für wiederholte Schritte beinhalten oder auch separate bzw. untergeordnete Teilprozesse repräsentieren. Jeder einzelne Arbeitsschritt tritt an einem Punkt und in einem bestimmten Kontext auf, der als viereckiges Feld dargestellt wird. Kontexte mit untergeordneten Bereichen werden als Feld mit inneren Feldern symbolisiert. Informationen, die durch den Arbeitsschritt vom Anwender aufgenommen bzw. ans System übergeben werden, sind durch entsprechend gerichtete Pfeile gekennzeichnet. Zur Erstellung des Modells werden gewonnene und kategorisierte Studiendaten durchgearbeitet und fortlaufend die erkannten Kontexte, Schritte usw. eingezeichnet. Die Uhr kennzeichnet eine Verzögerung des Prozesses. Je komplexer ein Arbeitsablauf ist, der durch ein UCM-Diagramm modelliert wird, desto komplizierter und unübersichtlicher wird naheliegender Weise auch das Diagramm. Sich regelmäßig wiederholende Schritte, die immer in denselben Kontexten auftreten oder ganze bestimmte Konstellationen von Schritten deuten auf ein separates Arbeitsmuster ('work pattern') hin. Deshalb lassen sich zu komplex gewordene Use Case Maps auch wiederum in mehrere unterteilen.

3.3 Von der Analyse zum Tool-Entwurf

In diesem Abschnitt soll genauer auf die eigentliche Entwicklung von Tools eingegangen werden.
Es ist nicht nur zu untersuchen, inwiefern die Arbeit durch Tools unterstützt werden kann, sondern auch, wie attraktiv diese Tools selber für die Mitarbeiter gestaltet werden können. Es ist vom psychologischen Standpunkt aus zu beachten, dass dem Entwickler ein Zurückgreifen auf neue Tools als sinnvoll erscheinen sollte, um nicht weiterhin nach den gewohnten ineffizienteren Arbeitsmethoden vorzugehen. Ein Programmierer muss schließlich auch selber Einarbeitungszeit für diese Werkzeuge aufwänden.
Neben den in den vorangegangenen Abschnitten beschriebenen grundlegenden Ansätzen zur Erfassung von Aktivitäten muss im weiteren Verlauf einer Recherche Wert auf die detaillierte Beobachtung von bestimmten Tätigkeiten gelegt werden. So kann z.B. nach der im Paper „An Examination of Software Engineering Work Practices“ beschriebenen Studie beobachtet werden, dass starke Unterschiede zwischen den Arbeitsweisen von erfahrenen Entwicklern und denen von unerfahreneren bestehen. Wenn etwa Veränderungen am System durchgeführt bzw. Fehler korrigiert werden, müssen sich die betreffenden Entwickler zunächst in bestimmte Bereiche des Systems genau einarbeiten. Zumeist ist also die Analyse und das Verständnis von Quellcode vonnöten. Erfahrene Entwickler gehen hierbei nach einer ganz bestimmten Methode vor, welche im genannten Paper als “Just in Time Comprehension”[SLVA97] (JITC) bezeichnet wird. Hiermit ist das gezielte und ökonomische Abschätzen von Prioritäten bei der Einarbeitung gemeint, da niemand ein Projekt mit teilweise Millionen von Lines of Code, an dem die Mitarbeiter einer ganzen Firma arbeiten, alleine immer vollständig im Kopf haben und in allen

Einzelheiten nachvollziehen kann. Besteht also die Notwendigkeit einer Bearbeitung bestimmter Systemteile, muss immer zunächst eine zielgerichtete Einarbeitung erfolgen, bevor Veränderungen vorgenommen werden können. Diese Arbeitsweise beherrschen Anfänger meist nicht bzw. nur unzureichend, da sie den Fehler machen, das gesamte System kennen zu wollen, wodurch sie sich mit Angelegenheiten auseinandersetzen, die für das konkrete Problem nicht relevant sind, was unnötig Zeit und Aufwand kostet.
Diese Kenntnis von Vorgehensweisen wie „JITC“ ist interessant für die Entwicklung von Tools. Anhand solcher Beobachtungen lassen sich Rückschlüsse darauf ziehen, welche Aspekte beim Entwurf von einzelnen Tool-Funktionen am meisten ins Gewicht fallen, um gezielte Hilfestellungen bei der Entwicklungsarbeit bieten zu können. So ist nun eine Ausarbeitung der genauen Schwerpunkte für die Entwicklung eines entsprechenden Tools gewährleistet, in diesem Falle ein Suchtool.
Zunächst einmal sind gewisse allgemeine Erfordernisse zu berücksichtigen. Das Tool muss in der Lage sein, mit der Systemumgebung zu interagieren, muss also individuell den technischen Entwicklungsbedingungen angepasst werden. Außerdem ist natürlich eine benutzerfreundliche Bedienung zu gewährleisten. Ferner sollte das Tool auch mit anderen Tools zusammenarbeiten können, um ein unkompliziertes Arbeiten ohne unnötigen Zeitaufwand wie etwa für ständiges Wechseln zwischen verschiedenen Anwendungen zu ermöglichen; außerdem dürfen keine Vorzüge von anderen Tools durch das neue beeinträchtigt werden.
Hierzu werden spezielle Anforderungen an das Tool festgelegt; es wird definiert, welche Funktionen erfüllt werden müssen. Für die Erkundung von Source Code nach dem JITC-Schema (s.o.) sollte es möglich sein, sowohl anhand von Stichworten bestimmte Elemente zu suchen und die Suchergebnisse in einem sinnvollen Zusammenhang miteinander übersichtlich zurückzuliefern, die Informationen sollten stets vollständig und übersichtlich sein. Ferner sollten frühere Suchprozesse auch später noch nachvollziehbar sein, um die eigene Arbeit leichter zurückverfolgen zu können.
Neben diesen funktionalen Aspekten spielt die Leistungsfähigkeit eine wichtige Rolle: Je mehr Code mit möglichst wenig Verzögerungszeit bei Anfragen das Tool bewältigen kann, desto nützlicher erweist es sich für den Entwickler, der Wert auf Flexibilität bezüglich der zu bearbeitenden Art von Quellcode legt. Auch sollten verschiedene Programmiersprachen unterstützt werden, da bei größeren Projekten oftmals in mehreren verschiedenen Sprachen entwickelt wird, deren Sourcecode vom Tool erkannt werden sollte.
Da es wohl immer gewisse Limite bei der Performance von Anwendungen geben wird, sollten auch diese schon von vornherein abgesteckt und dargelegt werden. Es ist ausreichend, wenn bestimmte Grundkomponenten nur auf bestimmten Maschinen laufen, da sie voraussichtlich ohnehin permanent dort laufen werden. Im JITC - Beispiel wäre es akzeptabel, wenn Objektorientierung innerhalb von Sourcecode nicht durch das Tool erkannt wird, auch laufzeitabhängige Ereignisse brauchen nicht analysiert werden zu können, da es sich hier um ein Suchtool handelt.
Diese Art der Entwicklungsplanung empfiehlt sich auch in Bezug auf die Erhöhung der Leistungsfähigkeit neuer Tools für ein bestimmtes Anwendungsgebiet, auf dem bereits äquivalente Tools existieren. Es kann somit gezielt auf die Schwächen dieser älteren Vertreter für dieselben Bedürfnisse eingegangen werden und diese in der eigenen Entwicklung vermieden und verbessert werden. So ist jede Produktion eines Tools stets individuell auf zuvor abgewogene und herauskristallisierte Anforderungen ausgelegt.

4 Zusammenfassung und Schlussfolgerung

Es wurde die Arbeit von Softwareentwicklern in ihren Grundzügen und ihren verschiedenen Arbeitsbereichen beschrieben, wobei auf unterschiedliche Bereiche dieser Tätigkeit und ihre jeweiligen Aufgaben eingegangen wurde, sowie die Begriffe „Programmierer“ und „Softwareentwickler“ gegeneinander abgewägt wurden. Die Herangehensweise an Studien und Recherchen in dieser Branche wurde anhand der vorgestellten empirischen, analytischen und arbeitspraktischen Studien dargelegt.
Es wurde auf die Organisation von Studien und auf die Umstände und Probleme beim Erfassen von Informationen während einer Recherche in unterschiedlichen Umgebungen eingegangen, grundlegende Methoden hierzu erläutert und speziell das „synchronized shadowing“ genauer beschrieben.
Schließlich wurde auf die Auswertung der gewonnenen Daten nach der Recherche eingegangen und als Beispiel für eine effiziente Vorgehensweise dabei die „Use Case Map-Notation“ gezeigt. Anschließend wurde die Entwicklung eines Tools auf Basis von gewonnenen Daten beschrieben.
Es ist davon auszugehen, dass die Softwareentwicklung auch zukünftig einen interessanten Bereich für Studien darstellen wird, da es im Zuge steigender Konkurrenz in der Softwareindustrie stets Bedarf an effizienten und optimierten Arbeitsabläufen geben wird. Daher stellt sicher auch die Thematik solcher Studien selber einen ausbaufähigen Teilbereich in der Softwareproduktion dar. Zum Beispiel würde hier die Automatisierung von Auswertungsprozessen, die bis heute noch weitgehend manuell ablaufen, einen interessanten Ansatz darstellen, wie etwa bei der Erstellung von Use Case Maps.

Literatur

[LA97] T. Lethbridge and N. Anquetil. Architecture of a source code exploration tool: A software engineering case study. 1997.

[LS02] Timothy Lethbridge and Janice Singer. Studies of the work practices of software engineers. pages 51–72, 2002.

[LSS05] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical Softw. Engg.*, 10(3):311–341, 2005.

[SLVA97] Janice Singer, Timothy C. Lethbridge, N. Vinson, and Nicolas Anquetil. An examination of software engineering work practices. In *Proc. of CASCON '97*, pages 209–223, 1997.

[ua] unknown author. Programmierer (http://de.wikipedia.org/wiki/programmierer).