Freie Universität Berlin

Wissenstransfer in global verteilten Firmen

Unterstützung der Entwicklungsprozesse von virtuellen Teams

Masterarbeit an der
AG Software Engineering
Prof. Dr. Lutz Prechelt
Institut für Informatik
Freie Universität Berlin

von
Robert Kunze

Gutachter:
Prof. Dr. Lutz Prechelt

Zweitgutachter:
Prof. Carsten Schulte

Betreuer:
Stephan Salinger

Anmeldedatum: 1. September 2009
Abgabedatum: 8. März 2010

Inhaltsverzeichnis

Überblick **xi**

Danksagung **xiii**

1 Einleitung **1**

1.1 Globalisierung 1

1.2 Virtuelle Teams 4

1.3 Wissenstransfer 6

1.3.1 Lernpräferenzen 7

1.4 Inhalte 9

2 Optimierungsansätze **11**

2.1 Initialer Zustand 11

2.1.1 Grad der Verteilung 13

2.1.2 Kommunikationskanäle 13

2.1.3 Initiale Wissensprozesse 14

2.2 Aufgaben des Managements 18

2.2.1 Wissensweitergabe 18
2.2.2 Vertrauen und Präsenz 21
2.3 Pair Programming 25
2.3.1 Distributed Pair Programming 30

3 Iterationen 35

3.1 Versuchsreihen 35
3.1.1 Vorgehensweise 36
3.1.2 Lernpräferenzen und Vorkenntnisse . 38
3.2 Erwartete Ergebnisse 40
3.3 Erste Iteration: Einführung in die Materie . . 40
3.3.1 Vorbereitung 40
3.3.2 Durchführung 43
3.3.3 Auswertung 44
3.4 Zweite Iteration: Anwendung des Gelernten 47
3.4.1 Vorbereitung 47
3.4.2 Durchführung 50
3.4.3 Auswertung 51
3.5 Dritte Iteration: Selbstständiges Schlussfolgern 53
3.5.1 Vorbereitung 53
3.5.2 Durchführung 54
3.5.3 Auswertung 56
3.6 Zusammenfassung 57

4 Zusammenfassung und Ausblick **61**

4.1 Zusammenfassung und Schlussfolgerungen . 61

4.2 Ausblick 63

A INTERAKTIONSPROTOKOLLE **65**

A.1 Iteration 1 65

A.1.1 Anweisungs-eMail 65

A.1.2 Feedback-eMail 1 66

A.1.3 Skype-Interview mit [N] 67

A.1.4 eMail-Interview 69

A.1.5 eMail-Interview-Antworten 70

A.2 Iteration 2 71

A.2.1 Anweisungs-eMail 71

A.2.2 eMail-Interview 71

A.2.3 eMail-Interview-Antworten von [N] . 72

B SONSTIGE DOKUMENTE **75**

B.1 Fragebögen 75

B.1.1 Vorversuchspersonen 75

Fragebogen [I] 75

Fragebogen [T] 76

B.1.2 Fragebogen [N] 77

B.1.3 Fragebogen [S] 78

B.2 Skill Matrix 79

Literaturverzeichnis **83**

Stichwortverzeichnis **87**

Abbildungsverzeichnis

1.1 globales virtuelles Team 5

2.1 K-Organigramm 12

2.2 Entwicklungsprozess initial 15

2.3 Weiterbildungsprozess initial 17

2.4 DPP via VNC 32

3.1 finaler Weiterbildungsprozess 60

Tabellenverzeichnis

3.1 Modellierungsnotationen - Vergleich 42

3.2 Vergleich von Screencast-Recorder-Software 48

Überblick

In den letzten Jahren wurde die Software-Industrie beeinflusst durch die Globalisierung und dieser Trend setzt sich heute fort. Der Aufstieg von Internet, eMail und verbesserter Telekommunikation zu niedrigen Kosten hat zur Entwicklung von virtuellen Arbeitsgruppen und sogar virtuellen Firmen geführt, welches seinerseits die Softwareentwicklung zu einer globalen Angelegenheit werden ließ. Dennoch ist, durch die große Komplexität von heutiger Software, das (teilweise) Auslagern der Entwicklungskapazitäten alles andere als einfach. Einige der Schwierigkeiten, auf die man trifft, beinhalten Faktoren, wie das Verstehen von Anforderungen oder das Testen von Systemen. Diese Schwierigkeiten werden ergänzt durch Unterschiede in der Kultur, der Sprachen, unzureichender Kommunikation, unterschiedliche Prozessreifestufen in den beteiligten (Sub-)Unternehmen, Standards, technische Fähigkeiten und Erfahrungen. In dieser Masterarbeit nun wurde untersucht, wie der Wissenstransfer in einer solchen globalen und virtuellen Umgebung optimiert werden kann – sowohl in technischer Hinsicht, als auf auf Managementseite. Dazu wurden die Voraussetzungen für erfolgreichen Wissenstransfer ermittelt und in Form von Experimenten getestet. Es wurden darin auch Hindernisse für erfolgreichen Wissenstransfer aufgedeckt und Möglichkeiten der Behebung analysiert. Zum Schluß wurde ein solch optimierter Prozess auf die Ausgangssituation direkt angewendet und damit die Prozesse angepasst.

Danksagung

Ich möchte mich herzlich bei allen bedanken, die mich beim Schreiben dieser Arbeit unterstützt haben. Mein besonderer Dank geht an Professor Carsten Schulte für gute Literaturhinweise und ein interessantes Gespräch über Lerntheorie, Professor Dr. Lutz Prechelt, für die Aufdeckung vieler Stolpersteine und möglicher Wege, Christopher Özbek für viele Anregungen und Hilfestellung in der Anfangsphase des Projektes, meiner Familie für die aktive Durchführung des Korrekturlesens, und ganz besonders meinem Betreuer Stephan Salinger, für dessen Förderung, Anleitung und Unterstützung vom Anfang bis zum Ende der Arbeit, ohne die die Arbeit nie in die richtige Richtung gekommen wäre.

Robert Kunze

Kapitel 1

Einleitung

Diese Masterarbeit widmet sich der Frage ob und in welcher Weise im Kontext von globalen virtuellen Teams ein effektiver Wissenstransfer stattfinden kann. Die Effektivität wird hierbei vor allem durch die Qualität des Lernerfolgs definiert, mit welchem die Mitglieder des Teams in eine für sie neue Technologie eingeführt werden können. Zusätzlich steht jedoch auch im Fokus, wie Wissen, welches während der Softwareentwicklung entsteht, effizient im ganzen Team verbreitet werden kann.
Dazu wird unter anderem exemplarisch eine Untersuchung im Unternehmen des Autors durchgeführt. Vorhandene Probleme zu erkennen und Möglichkeiten zu entdecken, sowie diese zu beheben, ist, wie im Folgenden beschrieben wird, essentiell für die Arbeit mit virtuellen Teams. Um zu verstehen, was das im Detail bedeutet und wie so etwas funktionieren könnte, müssen zunächst einige wichtige Begriffe definiert werden, sodass ein möglichst einheitliches Verständnis derselben unter den Lesern und dem Autor existiert.

1.1 Globalisierung

Globalisierung beschreibt einen stetigen Prozess durch welchen lokale Gesellschaften, Ökonomien und Kulturen in ein global umspannendes Kommunikations- und Han-

delsnetzwerk integriert werden. Im Rahmen dieser Arbeit wird vor allem der ökonomische Aspekt der Globalisierung thematisiert. Dieser wird definiert als die Integration von nationalen Ökonomien in die internationale Wirtschaft durch Handel, ausländische Direktinvestitionen, Kapitalströme, Migration und das Wachstum von Technologie[1]. Insbesondere soll es hier um virtuelle Teams gehen, die in einer multikulturellen und multilingualen Umgebung welche staatliche und organisationelle Grenzen überschreiten kann, arbeiten.
Globaliserung bedeutet also auch *'pulling away' of power and influences from local communities and even nations, and the establishment of a new 'global arena'*.Hustad [2004]
Eine generelle Konsequenz von Modernisierung und Globalisierung ist das Auftauchen der *Risikogesellschaft*. Ein Wachstum des Risikos bedeutet sinkende Kontrolle. Anthony Giddens und Ulrich Beck, beide bekannte Soziologen, die im Bereich Modernisierung und Globalisierung viel geforscht haben, sehen die aktuellen Modernisierungs- und Globalisierungsprozesse als einen Bruch mit früheren Formen der Modernisierung. Traditionell implizierte die Modernisierung eine bessere Kontrolle durch ein Mehr an Wissen und besseren Technologien. Im Zeitalter von hoher Modernisierung und Globalisierung kann mehr Wissen jedoch auch zu einer Steigung der Unvorhersagbarkeit, Ungewissheit und damit zu einem Verlust der Kontrolle führen[2].

Zu den Faktoren, die zu einer derartigen Verteilung und Globalisierung von Unternehmen führen können, zählen vor allem in der Software Industrie, der Arbeitskräftemangel und die hohen Kosten für fähige IT Spezialisten. In diesem Umfeld wachsen Indien, Irland und Israel zu entfernten Entwicklungszentren für US-basierte Unternehmen heran, da dort die Lohnkosten niedrig sind und es eine große Menge an gut ausgebildeten, englisch sprechenden Arbeitern gibt.[3] Weitere Ursachen können die Akquise von internationalen Firmen sein, das Potenzial, die IT Entwicklungskosten zu senken, der Wunsch die globale Präsenz zu

[1] siehe Bhagwati [2004]
[2] siehe Braa et al. [2000], S. 48 – 50
[3] siehe Abramson [1999]

stärken.[4]

> *Another motivation is to locate IT operations and support close to operational centers or customer bases, to enhance local relationships, establish local presence, and to take advantage of localized areas of expertise.*[5]

Mit der Einführung von direkter und billiger Kommunikation über das Internet – genannt seien hier exemplarisch eMail, Internet-Telefonie und Datenbanken – wurde die Tür für die schnelle Globalisierung der Software-Industrie geöffnet.

Schon von Natur her ist die Softwareentwicklung ein komplexer, sozialer Prozess, der viel Kommunikation und Koordination bedarf. Sich als eine Firma in dem Bereich zu globalisieren erhöht die Komplexität weiter durch kulturelle, sprachliche und organisationelle Unterschiede. Carmel [1999] nennt fünf wesentliche Probleme, die es für Software-Entwicklungsteams im globalen Kontext gibt:

- Anpassung an die geografische Verteilung des IT Personals und der Nutzer
- Verlust der Kommunikationsvielfalt und -reichhaltigkeit durch weniger persönliche (*face-to-face*) Interaktion
- Störungen der Koordination im Projektmanagement
- Verlust des *Teamgefühls*
- mit kulturellen Unterschieden zwischen global verteilten Angestellten und Organisationen zurecht kommen

Auftretende Probleme liegen hierbei, wie man sieht, in allen möglichen Bereichen der Softwareentwicklung und

[4] siehe Carmel [1999]
[5] siehe Davidson and Tay [2003]

können von vorhandenen Sprachbarrieren bis hin zu unterschiedlichen Prozessreifegraden in den jeweiligen (Sub-) Unternehmen reichen. All das schlägt sich nieder auf die Qualität und die entstehenden Kosten der Entwicklung, wodurch eine Behebung und/oder Vermeidung dieser Probleme als äußerst lohnenswertes Ziel erscheint.

1.2 Virtuelle Teams

Ein virtuelles Team ist eine Weiterentwicklung der sogenannten *Network Organization*. Eine *Network Organization* ist eine Firma, oder eine Gruppe von Firmen, die nur ein Minimum an formellen Strukturen aufweist und sich im Ausgleich dafür auf die Formation und Auflösung von Teams verlässt, um ein spezifisches Ziel zu erreichen. Eine Network Organization benutzt dabei extensiv Informations- und Kommunikationstechnologien und greift dabei auf Wissen über Unternehmensgrenzen hinweg und innerhalb von Unternehmen der Wertschöpfungskette zurück.[6] Das virtuelle Team stellt insofern eine Weiterentwicklung der Network Organization dar, als dass es auf verbesserten Informations- und Kommunikationstechnologien basiert.[7] Ein virtuelles Team besteht weiterhin aus Individuen mit unterschiedlichen Kompetenzen, die verteilt über Zeit, Raum und Kulturen sind und sich formen, reorganisieren und wieder auflösen in Abhängigkeit von Änderungen am Markt.[8]

Virtuelle Teams ermöglichen eine hohe Flexibilität und Reaktionsgeschwindigkeit, niedrige Kosten und verbesserte Resourcennutzung, die vor allem in einer hochturbulenten und dynamisch-globalen Geschäftswelt mit sich ständig ändernden Anforderungen angepasst ist (siehe Fußnote 8). Doch es gibt auch Nachteile dieser Form: niedrige individuelle Verbindlichkeit, Überlastung an eingenommenen Aufgaben/Rollen, uneindeutige Aufgaben und Rollen. Abwesenheit und Abgelenktheit können in einem

[6] siehe BNET [zuletzt überprüft: Jan. 2010]
[7] siehe Davidow and Malone [1992]
[8] siehe Mowshowitz [1997]

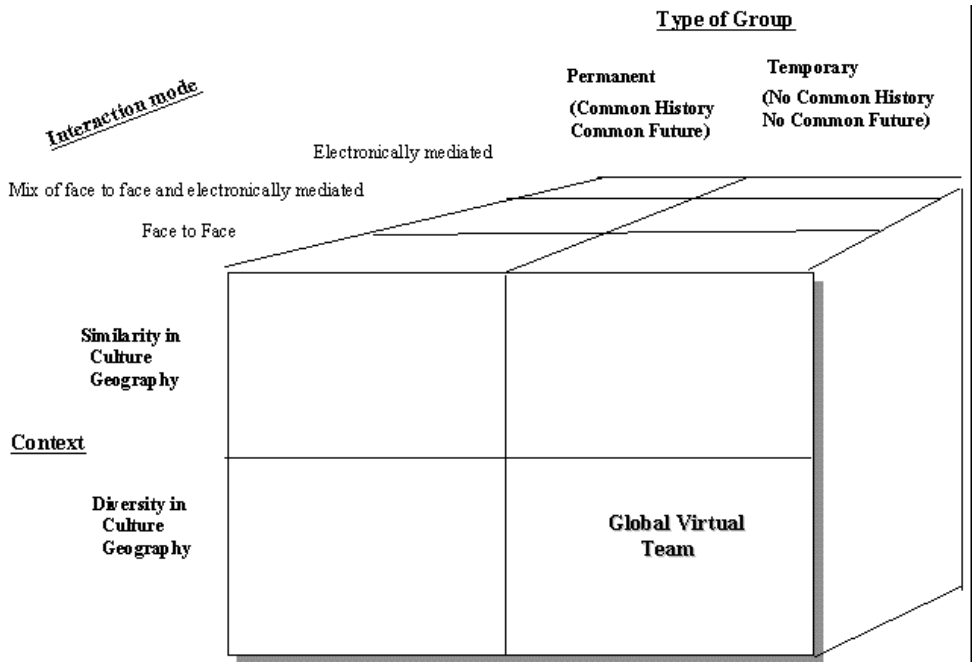

Abbildung 1.1: Definition: globales, virtuelles Team; entnommen aus Jarvenpaa and Leidner [June 1998]

virtuellen Team verstärkt auftreten – manchmal sogar ohne dass man es sofort bemerkt. Es ist außerdem möglich, dass Kunden den Eindruck von geringer Dauerhaftigkeit, Verlässlichkeit und Konsistenz in virtuellen Formationen gewinnen (siehe Fußnote 8).

In der Definition des virtuellen Teams ist es bereits angedeutet, doch mit der präziseren Benennung als *globales virtuelles Team*, wird hier deutlich, dass es, wie man in Abbildung 1.1 sehen kann, drei wichtige Dimensionen gibt, in denen man dieses einordnen muss: den **Kontext**, der durch den Grad der Unterschiedlichkeit von Kultur und Geografie bestimmt ist, den **Interaktionsmodus**, der von direkter, persönlicher Kommunikation (face-to-face) bis hin zu elektronisch vermittelter Kommunikation reichen kann und der **Art der Gruppe**, welche eine permanente Gruppe mit einer gemeinsamen Vergangenheit und einer gemeinsamen Zukunft oder eine temporäre Gruppe, in der es weder eine gemeinsame Vergangenheit, noch eine gemeinsame Zukunft geben muss. Hier zeichnet sich das globale

virtuelle Team durch einen globalisierten Kontext, oft eine elektronisch unterstützte Kommunikation und seine geringe Permanenz aus.

Entwickler verbringen in der Regel 70% ihrer Zeit mit der Zusammenarbeit mit anderen Projektbeteiligten. In global verteilten Teams wird die Koordination besonders schwierig, da die Team-Interaktionsmöglichkeiten auf weniger *reiche* Kommunikationskanäle, wie zum Beispiel Telefon, eMail und Datenbanken, begrenzt sind.

1.3 Wissenstransfer

Der Begriff *Wissen* unterscheidet sich vom Begriff der *Information* in der Hinsicht, dass das Wissen in den Menschen vorhanden ist und dadurch immer persönlich ist. Damit ist Wissen nicht nur etwas, das man versteht und an das man glaubt, sondern vor allem auch etwas, was man aktiv benutzt beziehungsweise anwendet. Der Aufbau und die Entwicklung des Wissens eines Unternehmens hängt damit sehr stark von der Interaktion der Menschen und damit von einer Kultur der Kommunikation und Zusammenarbeit im Unternehmen ab – und immer öfter auch zwischen Unternehmen.
Blackler [1995] unterteilt den Begriff *Wissen* in fünf Arten: *embrained*, *embodied*, *encultured*, *embedded* und *encoded*. Dabei bezeichnet er *embrained* als kognitive und individuelle, konzeptionelle Fähigkeiten. *Embodied* ("körperliches") Wissen bezeichnet die Fähigkeit, bestimmte Aktionen mit seinem Körper durchführen zu können. Unter *encultured* Wissen definiert er den Prozess, ein gemeinsames Verständnis in Gruppen, Organisationen oder Gesellschaften zu erreichen, wobei *embedded* (eingebettetes) Wissen eine weitere Art von gemeinsamem Wissen darstellt, welches sich im Alltag widerspiegelt. Schlussendlich benennt er als *encoded* Wissen explizites Wissen, welches durch eine schriftliche Form, wie in Büchern und digitalen Informationen repräsentiert wird.

Prozesse wie die Generierung, das Bereitstellen und der Transfer von Wissen werden in einem globalen Kontext wesentlich komplizierter, zum Beispiel durch unterschiedliche Kulturen. Das Konzept *Wissen* selbst kann in anderen Kulturen gänzlich anders aufgefasst werden, was nicht zuletzt auch die Einstellung der Menschen hinsichtlich der Bereitstellung von Wissen beeinflussen kann. Wissen über welches nicht gesprochen wird, kann nur schwerlich erfasst werden und ist damit fast unmöglich zu enkodieren und im Weiteren zu verbreiten.
Explizites Wissen kann in Prozessen, Dokumenten und Datenbanken umgesetzt (*embedded*) werden. Dafür benötigt es jedoch eine strenge, formale Kodierung um es in eine Information zu transformieren, da ohne einen solchen Formalismus die Genauigkeit des Wissensinhalts verloren gehen kann. Da die Informationstransformation zu Wissen immer nur rein subjektiv, und damit allein im jeweiligen Kontext des Lernenden ist, kann es außerdem passieren, dass Wissen durch die Fähigkeit des Lernenden, das enthaltene Wissen zu interpretieren falsch verstanden wird und damit zu Fehlinformation führt.
Es können also mannigfaltige Probleme beim Wissenstransfer auftreten.

1.3.1 Lernpräferenzen

Lernpräferenzen beschreiben die Art und Weise, mit der Menschen zu lernen bevorzugen. Es gibt eine ganze Reihe unterschiedlicher Lernpräferenzmodelle, wobei die Meisten davon ausgehen, dass jedem Menschen potenziell völlig individuelle Lernpräferenzen zugeschrieben werden können. Eines dieser Modelle soll hier kurz näher vorgestellt werden: der sogenannte Myers-Briggs-Typ-Indikator (MBTI). Der MBTI ermöglicht eine Messung der psychologischen Präferenzen, die die Menschen ihre Umwelt wahrnehmen und Entscheidungen treffen lässt. Er unterscheidet die Lernpräferenzen in insgesamt 4 Dimensionen:

- *extroversion* versus *introversion*,
- *sensing* versus *intuition*,

- *thinking* versus *feeling* und
- *judging* versus *perceiving*.

Diese Dimensionen führen zusammen zu insgesamt 16 unterschiedlichen Lernstilen oder -typen. Jeder Typ ist eine Kombination aus den vier Dimensionen.
Die erste Dimension, gibt die Quelle der persönlichen Antriebskraft an, also ob man sie aus der äußeren Welt, durch Aktivitäten, Belustigungen, Menschen und Dingen bezieht oder ob man sie aus der inneren Welt der Gedanken, Interessen, der eigenen Vorstellungskraft und Ideen bezieht.
Die zweite Dimension, *sensing* versus *intuition*, gibt an, wie das Gehirn die Realität kategorisiert, organisiert und speichert. Dabei spielt für den *sensing* Menschen das Hier und Jetzt die wichtigste Rolle, also vor allem sensorische Details. Die *intuitive* Menschen versuchen Konzepte und Muster in den gesammelten Informationen zu finden und zu interpretieren. Für sie spielen die zu entdeckenden Möglichkeiten und die Zukunft eine wichtige Rolle.
Thinking versus *feeling* beschreibt, wie Entscheidungen vom Menschen getroffen werden: der Denker entscheidet auf Grundlage von faktischen Prinzipien und formt Schlussfolgerungen systematisch. Die Logik steht hier im Vordergrund. Beim Fühlenden steht der subjektive Eindruck, also Gefühle, zwischenmenschliche Aspekte, ästhetische Ideale und Werte bei der Entscheidungsfindung im Vordergrund.
Die letzte Dimension beschreibt, welches die eigentlich ausgeführten Aktionen nach dem Treffen der Entscheidung sind. Der *judging* Mensch ist dabei organisiert und vorbereitet und hält an getroffenen Entscheidungen fest, während der *perceiving* Mensch bereit ist, seine Entscheidungen mit neuen Informationen zu überdenken und tendenziell wenig plant.

DeZure [2000], S. 83ff reduziert dabei die vier Dimensionen auf die ersten zwei, die für das Verständnis des Lernstils besonders hilfreich sind. Sie fassten diese in zwei Gruppen zusammen: Menschen, die das *abstract reflective* Lernmuster bevorzugen (entsprechen den *introverted intuitives*, oder kurz `IN` MBTI-Part) und Menschen, die das *concrete active* Lernmuster bevorzugen (entsprechen dem *extro-*

verted sensing, oder kurz ES MBTI-Part).
Für die *abstract reflective* Lerntypen gilt hiernach oft:

> *Their path to excellence is from theory to practice, and they prefer open-ended instruction to highly structured instruction. They usually demonstrate a high degree of autonomy in their learning and value knowledge for its own sake.*[9]

Das genaue Gegenteil findet man in den *concrete active* Lerntypen vor: Sie bevorzugen das Konkrete, das Praktische und das Unmittelbare. Sie haben oft Schwierigkeiten mit komplexen Konzepten und sind weniger unabhängig in dem was sie denken und wie sie bewerten. *The path to educational excellence [...] is usually a practice-to-theory approach.*[10].

1.4 Inhalte

Die Herausforderungen beim Wissenstransfer in global verteilten Firmen liegen also auf der Hand: zum Einen ist der Wissensaustausch und -transfer, der ohnehin keine triviale Aufgabe ist, durch die Globalisierung wesentlich erschwert, zum Anderen schaffen die Globalisierung und im Speziellen der zunehmende Einsatz von virtuellen Teams weitere Probleme, die nur durch geschicktes Management und neue Technologien gelöst werden können. Im Folgenden sollen mögliche Ansätze dargestellt und untersucht werden. Wie bereits initial erwähnt, soll hier sowohl auf die Management-Seite als auch die Technologie-Seite eingegangen werden. Im Anschluss daran wird versucht einen effektiven Weiterbildungsprozess zu implementieren und diesen in Form einer Reihe von Experimenten zu veranschaulichen.
Zum Ende hin werden die gewonnenen Erkenntnisse zusammen gefasst und ein Ausblick auf mögliche weitere Untersuchungen und Fragestellungen gegeben.

[9] siehe DeZure [2000], S. 83
[10] siehe Fußnote 9

Kapitel 2

Optimierungsansätze

2.1 Initialer Zustand

Da diese Arbeit das Ziel hat den Zustand in einer spezifischen Firma zu verbessern, soll hier kurz dargestellt werden, wie in dieser der initiale Zustand bezüglich der wichtigsten Faktoren – Grad der Verteilung, Kommunikationsschnittstellen, Wachstum und aktuelle Wissensprozesse – zum Zeitpunkt des Beginns der Arbeit war. Der Autor dieser Arbeit steht dabei in direktem Kontakt mit der Firma, ist sogar wesentlicher Teil davon (hat die Rolle des CTO[1] inne) und hat somit direkten Einfluss auf all diese Faktoren. Im Folgenden sei *K* die Bezeichnung für dieses Unternehmen.

Die Firma K ist ein noch recht junges Internet-Startup, welches sich seit 2008 im Bereich Web Publishing entwickelt. Initial von drei Gründern erdacht und aufgebaut, hat es sich von vornherein das Ziel gesetzt extrem flexibel in allen für die Erfüllung des Geschäftsziels relevanten Bereichen zu sein. Dieser Anspruch schafft nicht nur organisatorische Herausforderungen sondern auch eine Menge an technologischen Anforderungen an die gesamte Infrastruktur und das entwickelte Systemdesign.
K hat eine sehr schlanke und flache Unternehmensstruktur. Keiner der Mitarbeiter in der Firma ist angestellt –

[1] Chief Technology Officer (=*Technischer Direktor*)

nicht einmal die Unternehmensführung. Dennoch existiert natürlich eine Hierarchie in der Organisation. Wie man in Abbildung 2.1 sieht, ist diese Gliederung der einzelnen Aufgabenbereiche tatsächlich sehr ähnlich den Strukturen in anderen Unternehmen.

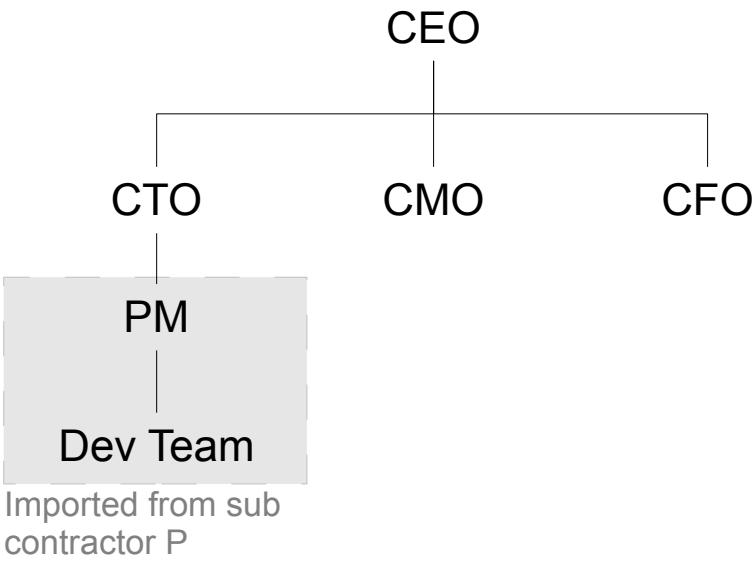

Abbildung 2.1: leicht vereinfachtes Organigramm von Firma K

Zu sehen ist, dass an oberster Spitze der Firmenleiter angeordnet ist, darunter die jeweiligen Ressort-Leiter für Technologie, Marketing und Finanzen. Dem Leiter der Technologie-Abteilung (CTO) untersteht ein durch den Zulieferer P bereitgestelltes Team inklusive entsprechendem Projektmanager. Letzterer hat vor allem die Funktion die Anforderungen entgegen zu nehmen und eventuell offene Punkte mit dem CTO zu klären, sie anschließend dem Entwicklungsteam zu übergeben und anschließend in der Qualitätssicherungsphase den entstandenen Code im Hinblick auf die Anforderungen zu validieren (welches nicht immer zuverlässig funktioniert). CTO und Projektmanager sind teilweise selbst in die eigentliche Implementierung eingebunden. Das bereitgestellte Team hat im Wesentlichen die Aufgabe, den CTO zu entlasten und die stetige Weiter-

entwicklung des Systems zu gewährleisten. Entsprechend sind die Entwicklungsaufgaben größtenteils das Verstehen und Anpassen bestehender Funktionalität an neue Anforderungen.

2.1.1 Grad der Verteilung

Das Unternehmen ist derzeitig über weit voneinander entfernte Teile der Welt verteilt. Das bedeutet im Einzelnen: der CTO hat seinen Sitz in Berlin, der CEO sitzt in Los Angeles, der CFO in New York, das Marketing in San Francisco und das Entwicklungsteam vom Unternehmen P in Chennai, Indien. Aufgrund dieser enormen Verteilung entfallen innerhalb der Firma die Kosten für Bereitstellung von Büros und Arbeitsräumen und schaffen somit weitestgehend Freiheit des Arbeitsortes. Durch dieses Minimum an Facility Infrastruktur ist der Aufwand für die Verwaltung derselben geringer und hält damit die interne Verwaltungsstruktur schlanker.
Das Team in Indien besitzt jedoch einen gemeinsamen Arbeitsplatz innerhalb eines Gebäudes; beziehungsweise sogar innerhalb einer Etage. Da diese Einrichtungen jedoch außerhalb der Firma K liegen, gibt es nur auf Seiten von Unternehmen P entsprechende Verwaltungsaufwände. Weiterhin ist der Autor aus Sichtweise des Unternehmens K betrachtet, zusätzlich Teil des Entwicklungsteams, da er ebenfalls stark mit in die Entwicklung involviert ist[2], sodass hier innerhalb des Teams keine wirkliche Kolozierung existiert.

2.1.2 Kommunikationskanäle

In Firma K werden zur Kommunikation unterschiedliche Technologien eingesetzt. E-Mail wird zur Planung von Meetings, Austausch von Briefing-relevanten Dokumenten und für interne Ankündigungen sowie als Fallback-Lösung, falls der gewünschte Gesprächspartner nicht online ist, benutzt. Eine bekannte Instant-Messaging-Software

[2] die Aufgabenteilung findet also nicht nur vertikal statt, sondern auch horizontal

wird in insgesamt 3 Varianten verwendet: Textnachrichten für den schnellen Austausch von Informationen und zur kurzfristigen gegenseitigen Abstimmung (Koordination), *(Video-)Anrufe* für Telefonkonferenzen und Meetings (durchschnittliche Dauer: 1 – 4 Stunden), wobei darin stets das strategische Vorgehen und eventuelle Probleme thematisiert sowie Briefings durchgeführt werden. Screensharing wird häufig dazu benutzt, um Funktionalität im Entwicklungsstadium zu demonstrieren und in Verbindung mit Anrufen zur detaillierteren Problembeschreibung.

Desweiteren ist ein Versionskontrollsystem (Subversion) im Einsatz, mittels welchem sowohl Dokumentation als auch Quellcode verwaltet und ausgetauscht werden. Zwei unterschiedliche Ticketing-Systeme kommen zur Ressourcen- und Aufgabenplanung sowie zur Beschreibung von internen Prozessen zum Einsatz. Dabei liegt ein Ticketing-System auf Seiten der Firma P und enthält dringende zu erledigende, sehr detailliert beschriebene Aufgabenpakete. Das andere Ticketing-System (ein Trac[3]-System) dient dem Bug-Tracking und der Zeitplanung von Features sowie als chronologische Übersicht über die Aktionen im Repositorium. Die bevorzugte Sprache für sämtliche Kommunikation ist Englisch, da es von allen im Unternehmen verständlich und verwendbar ist.

2.1.3 Initiale Wissensprozesse

Um die Wissenstransferprozesse im derzeitigen Zustand zu beschreiben, wird nun zunächst darauf eingegangen, wie die einzelnen Prozesse im Unternehmen im Rahmen der Softwareentwicklung aufgebaut sind. Dabei gibt es zwei wesentliche Unterteilungen:
Wissenstransfer zur Implementierung von neuen Features beziehungsweise der Änderung von bestehender Funktionalität und
der Wissenstransfer mit dem Ziel der Weiterbildung der Mitarbeiter. Zur Implementierung von neuen Features beziehungsweise der Änderung von bestehender Funktiona-

[3] siehe http://trac.edgewall.org/

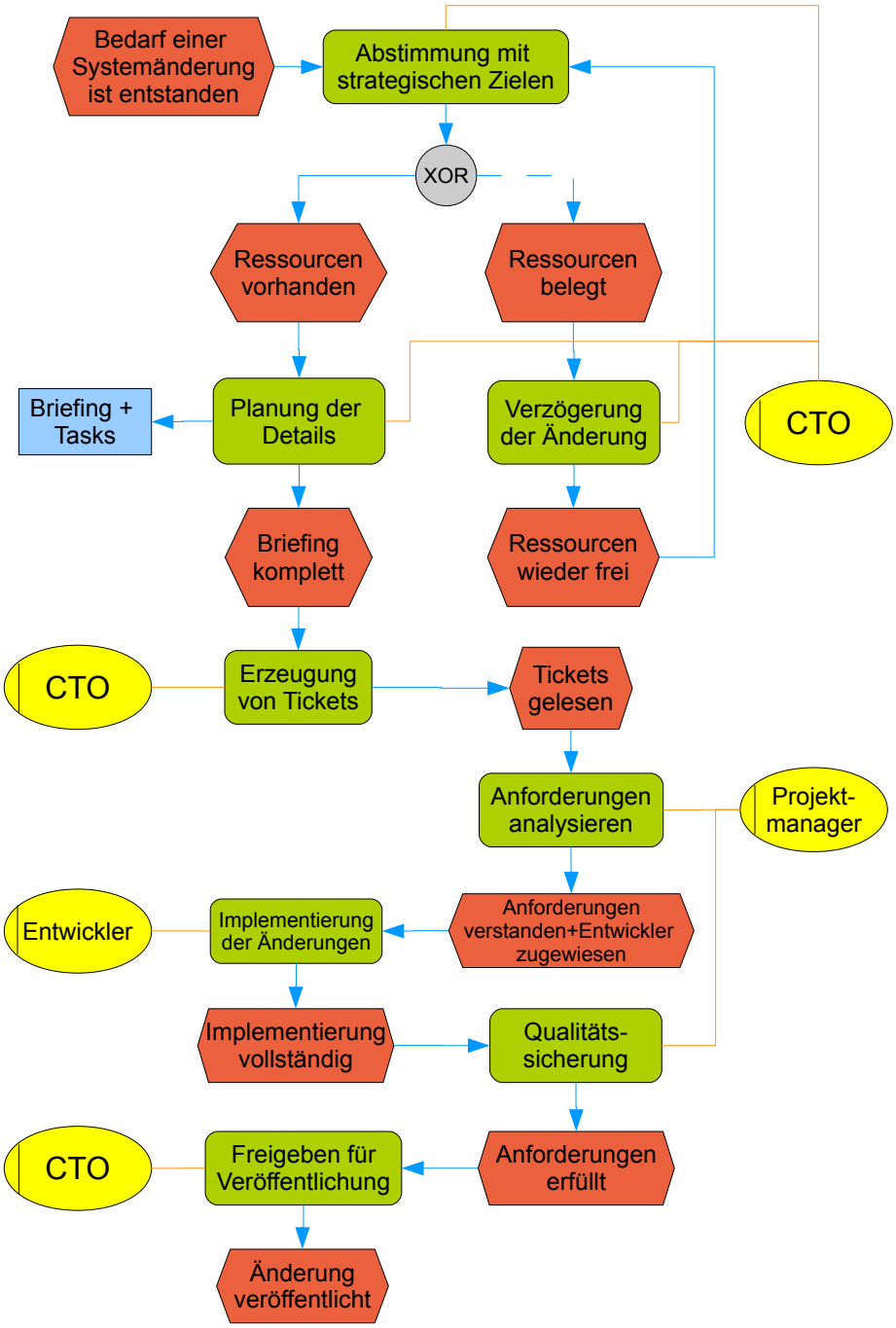

Abbildung 2.2: EPK-Modell Wissenstransferprozesse bei Feature-Implementierung/-Änderung

lität, wie in Abbildung 2.2 zu sehen ist, wird derzeitig zunächst die Änderung innerhalb der Geschäftsführung abgestimmt, sodass diese Entwicklungsaufgaben nicht mit strategischen Zielen kollidieren und somit wertvolle Ressourcen zu Zeitpunkten binden, wo diese an anderer Stelle dringend benötigt werden. Anschließend wird die Änderung an sich vom CTO bis zu einem gewissen Detailgrad geplant. Das bedeutet, dass Design-Mockups, sofern notwendig, erzeugt werden beziehungsweise zur Erzeugung an einen freischaffenden Designer gegeben werden. Außerdem werden Briefings mit einzelnen Tasks und deren geschätztem Zeitaufwand geschrieben. Diese werden in Aufgabenpakete (im Falle von umfangreicheren Änderungen) unterteilt und in das Ticketing-System der Firma P eingetragen. Dort werden diese dann zunächst vom Projektmanager analysiert und die Anforderungen interpretiert. Falls dabei Fragen auftauchen, was die Implementierung oder das Verständnis angeht, so werden diese in Rücksprache mit dem CTO geklärt. Anschließend werden bei der Firma P die Entwicklerressourcen alloziert und die Implementierung an diese übergeben.
Das bedeutet, dass zunächst neues Wissen durch detaillierten Entwurf der Inhalte von geplanten Änderungen vom CTO geschaffen wird. Dieses wird zu Informationen in Form von Briefings, Mockups und Aufgabenpaketen umgewandelt und damit in schriftlicher und modellhafter Art kodiert. Diese Informationen werden dann vom Projektmanager durch das Lesen und Verarbeiten wieder zu Wissen umgewandelt, um dieses an die Entwickler zu übertragen. Teil dieser Umwandlung ist also in einigen Fällen auch ein Dialog.

Zur Weiterbildung wurden bisher lediglich zwei Kommunikationskanäle benutzt: eMails und Skype-Textnachrichten. E-Mails wurden hier dazu benutzt, um die Konzepte zu erklären und enthielten Referenzen auf den Quellcode, sodass Implementierungsdetails im tatsächlichen System nachgeschlagen werden mussten. Diese eMails enthielten keinerlei graphische Hilfestellungen in Form von Modellen oder ähnlichem, zum Verstehen der Struktur oder komplexer (interner) Prozesse, sodass diese Art der Weiterbildung sich als sehr langwierig

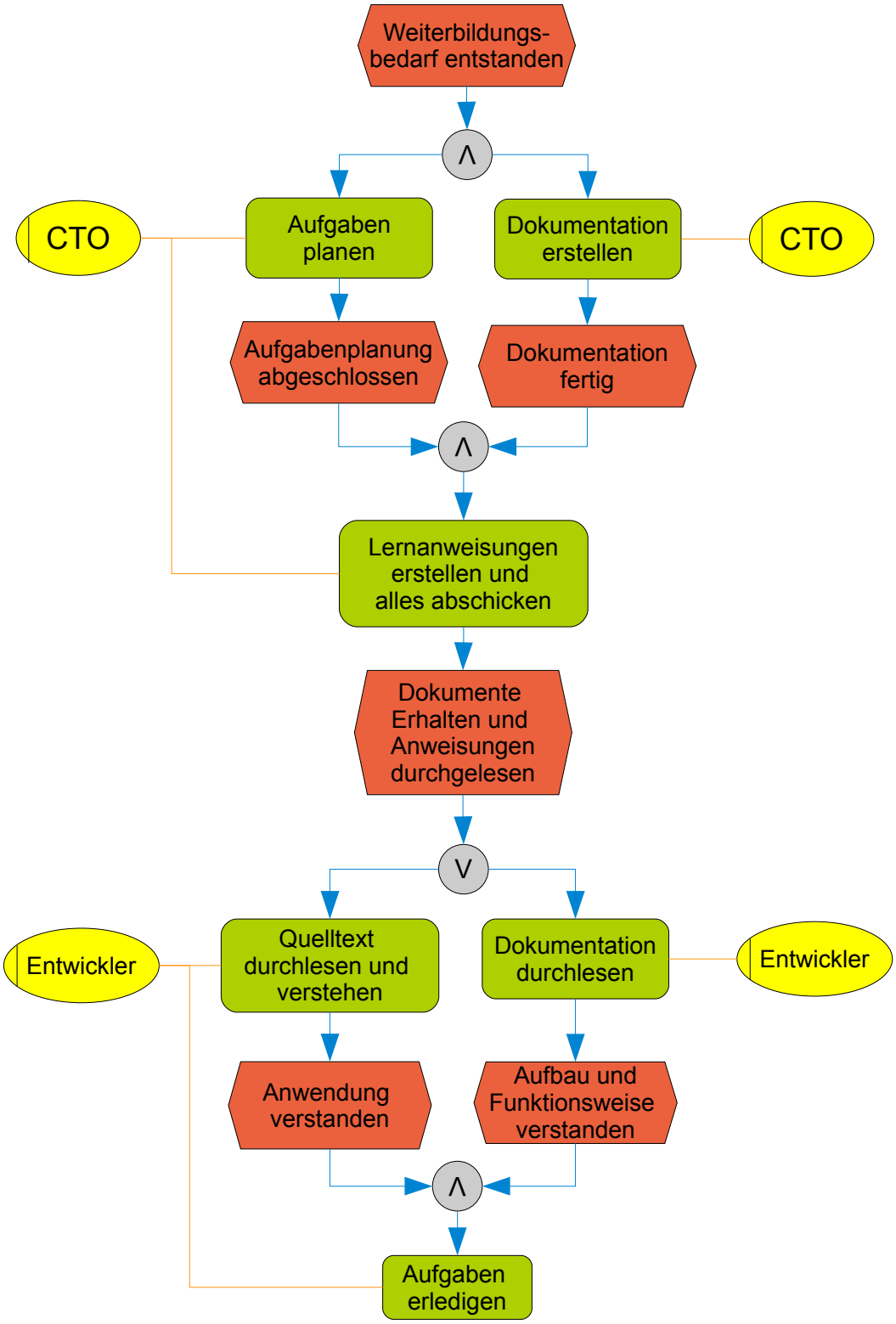

Abbildung 2.3: EPK-Modell des Wissenstransferprozesses bei Weiterbildungsmaßnahmen

darstellt. Bei der Neueinstellung von Entwicklern wurde so verfahren, dass diese zunächst vom Projektmanager in die Grundlagen der Entwicklung des Systems durch einfaches Erklären in einer gemeinsamen Sitzung vor dem PC und Zeigen im Quellcode eingeführt wurden. Anschließend hatten sie etwas Zeit, sich selbst mit dem System vertraut zu machen (sowohl aus Benutzersicht als auch aus Entwicklersicht). Nach einem Tag bekamen sie gezielt sehr einfache Aufgaben gestellt, die sie zu lösen hatten, wie zum Beispiel Anpassung von Oberflächenelementen. Erfahrenere Entwickler gaben hierbei anfänglich Unterstützung, später nur noch auf explizite Fragen von den unerfahreneren Entwicklern hin.
Das bedeutet, dass hier das Wissen über sehr *schmal-bandige* Kanäle transportiert wird, bei denen es zu einem höheren Maß an fehlverstandenem Wissen kommen kann. Durch Skype-Textnachrichten kann dies zwar etwas verringert werden, da Verständnisprobleme hier schneller bemerkt werden können, jedoch ist dies keine effiziente Form des Wissenstransfers, da im Nachhinein mittels eines Dialogs eventuelle Probleme und Fragen extra geklärt werden müssen, um ein korrekte(re)s Verständnis der Informationen zu gewährleisten.

2.2 Aufgaben des Managements

2.2.1 Wissensweitergabe

Wissen innerhalb eines Unternehmens zu verteilen hat einen intuitiven Anreiz: Wenn Mitarbeiter des Unternehmens ihre wertvollen Informationen frei mit anderen Mitarbeitern des Unternehmens teilen, so kann direkt verhindert werden, dass diese unnötigerweise wiederholt dieselben Probleme lösen müssen und damit Arbeit doppelt erledigt wird. Doch beim Teilen von Wissen gibt es einige ernsthafte Probleme.

> *[...] individuals often gain considerable power when they hold unique knowledge, especially when it is*

perceived to be of high value.[4]

Hierbei neigen die Menschen dazu, ihr Wissen nicht gerne zu teilen, da sie dadurch ihren Vorteil gegenüber anderen Mitarbeitern aufgeben und damit letztendlich leichter ersetzbar werden würden. Derartige Effekte können jedoch verringert oder sogar ganz verhindert werden.
Als Basis für eine solche Unternehmenskultur der Wissensverteilung gilt allgemein hin ein sogenanntes Wissensmanagementsystem. So installierte Microsoft ihre Knowledge Base einst für ihre eigenen Mitarbeiter, um diesen schneller Zugriff zu bekannten Problemen und deren (mögliche) Lösungen zu gestatten[5]. In derartigen Wissensmanagementsystemen kann das enthaltene Wissen als Allgemeingut angesehen werden, da es allen Mitarbeitern des Unternehmens (oder einer gewissen Gruppe) frei zugänglich ist. Und wie es bei allen Gütern ist, die der Allgemeinheit zur Verfügung stehen, zeigt sich – mehr oder minder ausgeprägt – der Trittbrettfahrer-Effekt, also Menschen, die auf diese Informationen zurückgreifen, sie nutzen, aber selbst ihr eigenes Wissen nicht teilen.
Marks et al. [February 2008] hat dazu einige Faktoren untersucht, die diese Effekte verringern sollen. Untersucht wurden hier das sogenannte *managerial prompting*, also eine direkte Aufforderung vom Management, das Wissen zu teilen, sowie die Gruppenidentifikation, also die Loyalität des Individuums zur Firma und den Einfluss der sozialen Werteorientierung.

Das *managerial prompting* soll den Mitarbeiter daran erinnern was von ihm erwartet wird und dadurch sein Verhalten in Hinsicht auf das Teilen von Wissen zu verändern. Diese (indirekten) Aufforderungen müssen allerdings nach bestimmten Kriterien angewendet werden, da die resultierenden Effekte auch mit den anderen beiden Faktoren eng zusammenhängen. Menschen, die sich sehr stark mit einer Gruppe identifizieren, sind eher bereit, Wissen innerhalb dieser zu teilen. Daher ist es als Manager gut, Mitarbeiter, die sehr egoistisch vorgehen, sich also wenig mit der Gruppe identifizieren, an ihre Gruppenzugehörigkeit

[4] siehe Fußnote 5
[5] siehe Marks et al. [February 2008]

zu erinnern, um so deren Bereitschaft das Wissen zu teilen zu vergrößern[6]. Obwohl in der Studie von Marks et al. [February 2008] keine statistisch signifikanten Unterschiede zwischen dem Teilverhalten von Menschen mit Gruppenzugehörigkeit und Menschen ohne eine solche gefunden werden konnten, fördern viele Organisationen dieses Gruppenzusammengehörigkeitsbild innerhalb der Organisation durch entsprechende Mottos an den Wänden oder jährliche Veranstaltungen mit Motivationsrednern oder regelmäßigen Hinweisen darauf, wie wichtig eine starke Positionierung des Unternehmens auf dem Markt ist.
Einen stärkeren Einfluss hingegen hatte in der Untersuchung die soziale Werteorientierung. Hierbei wurde in zwei wesentliche Gruppen unterteilt: *prosocial* und *proself*. Die Zufriedenheit der Ersteren steigt, wenn die Ziele der Gruppe als Ganzes erreicht werden, während die Zufriedenheit der Letzteren durch einen maximalen eigenen Vorteil in jeder Auseinandersetzung steigt. Die Ergebnisse hier waren, dass tatsächlich die *proself* signifikant weniger bereitwillig Informationen mit anderen teilten, als die *prosocial* Probanden. Durch *managerial prompting* konnte dies verbessert werden, wohingegen es bei *prosocial* Individuen keinen signifikanten Effekt hatte.

Als Manager eines virtuellen Teams, in welchem Wissen als elementares und doch allgemeines Gut gilt, sollte also darauf geachtet werden, dass, wenn man auf Mitarbeiter angewiesen ist, die ihr Wissen teilen, möglichst Individuen ins Team geholt werden, die *prosocial* sind, da diese eine starke Gruppenzugehörigkeit entwickeln und damit freiwillig eher bereit sind, anderen aus der Gruppe dieses Wissen bereitzustellen. Für die *proself* Individuen ist es hier wichtig, sie durch *managerial prompting* dazu zu bewegen, mehr zu teilen. Marks et al. [February 2008] stellten hierbei fest, dass nur einmaliges *managerial prompting* bei *proself* Individuen nur geringe Effekte hatte. Erst beim zweiten Mal stieg die Bereitschaft der *proself* Individuen ihr Wissen zu teilen deutlich an.

[6] siehe De Cremer and Van Vugt [1999]

Als weiterer wichtiger Punkt an dieser Stelle, muss das sogenannte *cognitive apprenticeship* angeführt werden. Dies beschreibt einen Prozess, bei welchem ein "Auszubildender" von einem "Meister" Fähigkeiten beigebracht bekommt. Durch die Nutzung von Prozessen wie Modellierung und Beratung unterstützt diese Theorie drei Stufen der Wissensvermittlung: die kognitive Stufe, die assoziative Stufe und die autonome Stufe[7]. Die erste Stufe dient hier dazu, sich eine integrierte Menge an Fähigkeiten anzueignen, welches vor allem durch Beobachtung und unter Anleitung erreicht werden soll. Die zweite, assoziative Stufe beinhaltet Methoden, um es dem Lernenden zu gestatten, sich auf die beobachteten Problemlösungsansätze zu konzentrieren und bewußten Zugang zu eigenen Problemlösungsstrategien zu entwickeln. Die dritte und letzte, die autonome Stufe hat das Ziel, den Lernenden auch darin zu fördern nicht nur selbst Probleme lösen zu können, sondern sie auch von sich aus zu erkennen, zu formulieren und letztendlich zu lösen.

2.2.2 Vertrauen und Präsenz

Virtuelle Teams unterscheiden sich von Teams an einem gemeinsamen Ort auch durch die Geschwindigkeit und die Art, mit welcher Vertrauen zwischen den einzelnen Mitglieder aufgebaut wird. Vertrauen wird sogar als grundlegend notwendig angesehen, um als Team zusammen arbeiten zu können. Vertrauen in einem virtuellen Team aufzubauen ist dabei eine große Herausforderung[8]. Handy [1995] bezweifelte ob virtuelle Teams tatsächlich ohne häufige direkte (face-to-face) Interaktion überhaupt effektiv arbeiten können. Er führte an, dass Vertrauen und Glaube auf gegenseitiger Gegenwart basieren (*trust needs touch*). O'Hara-Devereaux and Johansen [1994] fanden jedoch heraus, dass Vertrauen in einem global verteilten Team eine Grundlage dafür ist, dass die gegenseitige geographische Entfernung nicht zu einer psychologischen Barriere wird. Denn nur mit einem gewissen Maß an Vertrauen ist man bereit, risikoreiche Unternehmungen auf sich zu nehmen, die man weder

[7] siehe Collins et al. [1991]
[8] siehe Jang [2009]

überwachen noch steuern kann. Vertrauen ermöglicht also eine auf Intuition gestützte Entscheidung, die mangels ausreichender Informationen oder Zeit sonst nicht getroffen werden könnte[9].

> *Vertrauenswürdig ist, wer bei dem bleibt, was er bewußt oder unbewußt über sich selbst mitgeteilt hat.*[10]

Es sind also Glaubwürdigkeit, Verlässlichkeit und Authentizität nötig für die Vertrauensbildung. Gudykunst et al. [1996] legt zusätzlich nahe, dass Menschen aus individualistischen Kulturen eher bereit sind, anderen Menschen in einer Internet basierten Kommunikation zu vertrauen, als es bei Menschen aus kollektivistischen Kulturen der Fall ist. Das heißt, dass hier auch zusätzlich die soziale Prägung eine Rolle spielt. Individualistische Kulturen sind vorrangig in Ländern wie Australien, Österreich, Kanada (ohne Quebec), Dänemark, Finnland, Deutschland, Groß Britannien, Irland, Neuseeland, Norwegen, Schweiz oder die USA anzutreffen. Kollektivistische Kulturen sind vor allem in Ländern wie Brasilien, China, Indien, Frankreich, Indonesien, Italien, Mazedonien, Pakistan, Philippinen, Singapur, Spanien oder Vietnam vorzufinden[11].

Jarvenpaa and Leidner [June 1998] fanden in der Studie heraus, dass die Bildung von Vertrauen in einem virtuellen Team durch bestimmte Handlungsweisen begünstigt, bzw. behindert wird. Verhalten, welches zu Beginn der Arbeit als Team zu höherem Vertrauen geführt hat, war danach:

- Social Communication
- Communication of enthusiasm
- Coping with technical uncertainty
- Individual initiative

[9] siehe Luhmann [2000], S. 27 – 28
[10] siehe Luhmann [2000], S. 48
[11] siehe Jarvenpaa and Leidner [June 1998]

Verhalten, welches zu einer Beibehaltung des Vertrauens in der weiteren Zusammenarbeit geführt hat, war:

- predictable communication
- substantial and timely responses
- successful transition from social to procedural to task focus
- positive leadership
- phlegmatic response to crisis

Die Computer gestützte Kommunikation kann auch dazu führen, dass kulturelle Unterschiede weniger stark bemerkt werden können und somit die wahrgenommene Ähnlichkeit der Teammitglieder steigt.

> *the lack of nonverbal cues eliminates evidence of cultural differences, such as variations in dressing, gesticulating, and greeting. Likewise, the written medium eliminates the effect of accents which would again reduce the saliency of differences in cultural background.*[12]

Um also in einer solchen Umgebung Vertrauen aufzubauen und zu erhalten, ist es wichtig, dass der Manager eines virtuellen Teams klar die Verantwortlichkeiten definiert, da ein Mangel an Struktur zu Verwirrung, Frustration und Entmutigung führen kann. Kommunikationsrichtlinien, wann und wie kommuniziert wird, können helfen, die Kommunikation vorhersagbarer zu machen. Wichtig ist: nicht die Häufigkeit der Kommunikation trägt zum Erfolg bei, sondern die Qualität und die Vorhersagbarkeit. Weiterhin sollten Konflikte bereits so früh wie möglich geklärt werden damit diese nicht im späteren Verlauf der Arbeit negative Auswirkungen haben.

[12] siehe Jarvenpaa and Leidner [June 1998]

Auf Fragen sollte, sofern organisierbar, so schnell wie möglich reagiert werden – selbst wenn es erstmal nur der Hinweis ist, dass die Frage gelesen wurde und nun diskutiert wird. Soziale Gespräche, also *Small Talk*, helfen in der Kennenlernphase der Teammitglieder beim Aufbau von Vertrauen. Hobbies, Wochenendaktivitäten und das Thema Familie eignen sich dazu. Dieses muss jedoch im Verlauf der Zusammenarbeit um die Aspekte der eigentlichen Arbeit ergänzt werden, bleibt also zu geringeren Teilen bestehen, während die aufgabenrelevanten Dialoge zunehmen sollten. Hier ist wichtig, dass Small Talk nicht als Ersatz für Arbeitsfortschritt mißbraucht werden darf. Enthusiasmus und Optimismus, den Projektfortschritt betreffend zu äußern sowie Eigeninitiative zu zeigen muss gefördert werden, da diese den *Wohlfühleffekt* verstärken und damit die Bereitschaft der Teammitglieder erhöhen, Wissen innerhalb des Teams zu teilen.

Eine Möglichkeit, diese Mischung aus sozialer und arbeitsrelevanter Informationsvermittlung zu erreichen, wurde von Carroll [2008, zuletzt überprüft: Feb. 2010] vorgeschlagen: die Förderung von *ambient awareness*, also der Präsenz nach außen, wie es auf Plattformen wie Facebook, Twitter und Flickr in Form von persönlichen Statusmeldungen bereits in nicht kommerziellem Einsatz stark genutzt wird, kann über die Zeit verteilt zu einem sehr genauen Verständnis der jeweils anderen Person führen und damit eine empathische Beziehung zu diesem erzeugen. Im Geschäftsumfeld kann dies zu einer größeren Kohäsion, also einem Zusammenwachsen des virtuellen Teams und damit zu mehr Vertrauen untereinander führen:

> *Since virtual team members are typically dispersed, awareness tools might be very useful for leaders to help team members grow closer, building cohesion and trust.*[13]

Der Artikel empfiehlt Methoden, mit denen die Präsenz im virtuellen Team gefördert werden kann, ohne zu einer Belastung und Störung der Teammitglieder zu führen. So

[13] siehe Carroll [2008, zuletzt überprüft: Feb. 2010]

sollten derartige Statusmeldungen pro Person zwischen 3 – 4 am Tag ausreichend sein. Der Team-Leiter sollte seinem Team klar machen, warum derartig häufige Statusmeldungen hilfreich sein können und mit gutem Vorbild voran gehen. Auch sollte dieses Statusupdate von allen im Team durchgeführt werden. Bei Meldungen, die sich aufeinander beziehen, also zum Beispiel direkte Antworten oder andere Reaktionen auf vorhergehende Nachrichten, sollte der Tonfall unterstützend sein.

2.3 Pair Programming

Laut Salomon [1993] wird Wissen üblicherweise in einer Gemeinschaft produziert. Dabei kann es zum Einen durch eine Zusammenarbeit in Richtung eines gemeinsamen Zieles entstehen. Zum Anderen kann es aber auch Herausforderungen, die durch Dialoge, in denen die unterschiedlichen Perspektiven der Menschen zum Ziele führen entstehen. Daher kann es zur Optimierung des Wissenstransfers und der Gleichverteilung des Wissens innerhalb des Teams eine gute Idee sein, Pair Programming in der Entwicklung einzusetzen.

Pair Programming bezeichnet ein Entwicklungsverfahren, in dem zwei Entwickler vor einem einzigen Computer zusammen programmieren. Dabei haben beide Entwickler unterschiedliche Rollen inne: derjenige, der die Tastatur benutzt, ist der sogenannte *Driver*, was bedeutet, dass er derjenige ist, der den Quellcode nieder schreibt, während der andere Entwickler die Rolle des *Observers* (manchmal auf *Navigator* genannt) inne hat. Letzterer ist dafür verantwortlich, den Driver zu überwachen und darauf zu achten, dass dieser keine mehr oder weniger offensichtlichen Fehler macht (zum Beispiel sich vertippt ohne es zu merken) und den Überblick über das zu erstellende Resultat zu behalten. Damit obliegt ihm auch die Pflicht, sich über die Richtung, in die entwickelt werden soll, Gedanken zu machen und diese mit in die Arbeit des Drivers einfließen zu lassen.
Nach einer gewissen Zeit - sie kann je nach den Vorlie-

ben beziehungsweise der Aufmerksamkeit der beteiligten Entwickler variieren - werden die Rollen dann getauscht und der Driver wird zum Observer, während der Observer zum Driver wird und den Code weiter entwickelt. Obwohl dieser Rollentausch ein informeller Prozess ist, ist ein typisches Arbeitsinterval vor einem Rollentausch 20 Minuten.[14]

Lange Zeit wurde die Software von den jeweiligen Programmierern einzeln entwickelt. Tatsächlich ist es so, dass in den frühren Jahren der Softwareentwicklung ein Programm, später ein Stück Code, als ein privates Eigentum betrachtet wurde:

> *In the early years of programming, a program was regarded as the private property of the programmer. One would no more think of reading a colleague's program unbidden than of picking up a love letter and reading it. This is essentially what a program was, a love letter from the programmer to the hardware, full of the intimate details known only to partners in an affair.*[15]

Diese Sichtweise schafft durchaus viele Probleme, da das Wissen, was in einem solchen Stück Code steckt, nur der jeweilige Entwickler hat. Das bedeutet, dass sobald dieser Entwickler die Firma verlässt oder nicht erreichbar ist, niemand mehr etwas mit dem Code anfangen kann und er im schlimmsten Fall noch einmal komplett neu geschrieben werden muss. Da beim Pair Programming – und noch viel stärker auch beim Extreme Programming – immer mindestens zwei Entwickler über den Code Bescheid wissen, ist diese Gefahr geringer. Wenn zusätzlich, wie beim Extreme Programming, die Teams immer neu gebildet werden, dann kann so mit der Zeit das Wissen um den Code in dem gesamten Entwicklungsteam propagiert werden und wird somit zu einem Allgemeingut.

[14] siehe Hanks [2004]
[15] siehe McConnell [2004], S. 842

Die weiteren Vorteile der Paarprogrammierung sind in allen möglichen Bereichen des Softwareentwicklungsprozesses zu finden. So steigert sich die Design-Qualität des Programms, da es von beiden Entwicklern gelesen werden können muss und gemeinsam mehr Implementierungsalternativen diskutiert werden, als es für einen einzelnen Programmierer der Fall wäre.[16]. Damit sinkt auch die Fehlerrate, da diese frühzeitig erkannt werden:

> *With pair programming, 'four eyeballs are better than two,' and a momentous number of defects are prevented, removed right from the start.*[17]

Durch die Zusammenarbeit fühlen sich die einzelnen Entwickler auch zufriedener, legen erhöhte Disziplin, zum Beispiel beim Schreiben von Unit-Tests, an den Tag und verwenden weniger Zeit darauf, eMails zu lesen oder im Web zu surfen.[18] Durch die geringere Arbeit, die der einzelne Entwickler des Teams am PC verbringen muss, verringert sich auch die Zeit, die er mit der Bedienung von Tastatur und Maus beschäftigt ist – idealerweise um die Hälfte, da dann beide Teampartner sich immer nach derselben Zeit gegenseitig ablösen – was auch dazu führt, dass die Wahrscheinlichkeit am RSI-Syndrom[19] zu erkranken sich für beide verringert.

Als großer Vorteil kann beim Pair Programming auch die Möglichkeit angesehen werden, neue Entwickler (NE) auf ein System einzulernen, indem ihnen ein erfahrenerer Programmierer (EE) zur Seite gestellt wird. Auf diese Weise kann der Neue nicht nur den Code besser kennenlernen, sondern auch auf anderes nützliche Wissen des Erfahreneren zurückgreifen und neue Programmiertechniken kennenlernen (siehe Fußnote 16):

[16] siehe Cockburn and Williams [2000]
[17] siehe Williams [2003], S. 27 – 28
[18] siehe dazu auch Williams [2003], S. 23
[19] oder im nicht-chronischen Fall eine Sehnenscheidenentzündung; mehr dazu siehe Petersen [2006]

> *Knowledge is constantly being passed between partners, from tool usage tips to design and programming idioms. The partners take turns being the teacher and the student. Even unspoken skills and habits cross partners.*[20]

Hierfür gibt es jedoch einige Grundbedingungen, so zum Beispiel, dass die Aufgabe, die beide zusammen bearbeiten für den EE weder zu schwierig und komplex, noch zu einfach sein darf, da dieser sonst mehr damit beschäftigt ist das Problem selbst zu lösen und dabei (möglicherweise unbeabsichtigt) den NE abhängt. Ist die Aufgabe jedoch zu einfach, so wird der Driver im ungünstigsten Fall (also wenn er kein echtes Interesse an der Weiterbildung des Team-Partners hat) den Code einfach herunter tippen und damit zu schnell für den NE sein, um ausreichend Zeit für Fragen zu lassen.

Üblicherweise, so zeigt die Erfahrung, sollten beide Entwickler weitestgehend auf dem gleichen Programmierniveau sein, damit sie bei der Paarprogrammierung möglichst produktiv sein können. Zusätzlich müssen beide die Eigenschaft mitbringen, dass sie offen für die Ideen des jeweils Anderen sind. Da beim Pairprogramming die Zusammenarbeit im Vordergrund steht, führt das Fehlen dieser Eigenschaft oft zu Problemen.
Auch kann nicht jeder Entwickler ohne weiteres zu einem guten Paarprogrammierer werden, da einige Entwickler lieber alleine arbeiten, was dazu führt, dass wenn man ihn zu Paarprogrammierung zwingt, viele der Vorteile nicht mehr zutreffen. Die Zufriedenheit dieses Entwicklers wird sinken, er wird wahrscheinlich weniger bereit sein, auf Ideen seines Partners einzugehen und damit ein gutes Design des Codes verhindern. Ähnlich sieht es bei den Programmierniveaus aus: Oft sind diese nicht auf einem ähnlichen Level und so hat die Paarprogrammierung in einigen Bereichen bereits einen schlechten Ruf als unpraktikabel und "Ressourcenverschwendung"(besonders im Managementbereich)[21].

[20] siehe Williams [2003], S. 29

[21] siehe Williams et al. [July/Aug. 2000] oder Cockburn and Williams [2000]

Da bei Einsatz der Paarprogrammierung im Rahmen dieser Masterarbeit vor allem in Hinsicht auf die Weiterbildung bzw. das Einlernen von Mitarbeitern im Fokus steht, kann man zumindest das Problem der unterschiedlichen Niveaus (welches meistens auch als das Schwerwiegendste benannt wird) umgehen. Denn da hierbei beide Entwickler nicht unter dem Druck stehen, so produktiv wie möglich sein zu müssen, kann der EE sich direkt auf seine Mentor-Rolle konzentrieren und so den NE wesentlich effektiver einlernen.
Doch auch hierfür gibt es einige Vorbedingungen, denn hier muss der Mentor sich auch seiner Rolle als solcher bewusst sein.

Arten des Pair Programming zeichnen sich durch Entwicklungsmuster ab. Ein Beispiel dafür ist das sogenannte Ping Pong Pair Programming (pattern). Dieses wird vor allem im Rahmen des *Test-driven development*s eingesetzt. Es wird bei diesem Verfahren zunächst ein Testfall von Entwickler A geschrieben, welcher mangels Implementierung der zu testenden Funktionalität fehlschlägt. Anschließend wird Entwickler B der Driver (und damit A wieder zu Observer) und implementiert die fehlende Funktionalität, sodass der Test nicht mehr fehlschlägt. Daraufhin schreibt der den nächsten Testfall, welcher dann wiederum mangels einer Implementierung nicht erfolgreich abschließt. Dann übergibt er die Rolle des Drivers zurück an den Entwickler A, welcher dann die fehlende Funktionalität implementiert, sodass der Testfall erfolgreich endet. Im Anschluss daran schreibt A den nächsten Testfall, welcher mangels Implementierung fehlschlägt, und so weiter...[22]
Hierbei entsteht also nicht nur Code, der durch das 4-Augenprinzip subjektiv als sicherer wahrgenommen wird, sondern zusätzlich wird durch entsprechende Testfälle diese Sicherheit auch auf programmatischem Wege zu einem gewissen Grad gewährleistet.
Eine weitere Art des Pair Programmings ist das sogenannte Distributed Pair Programming, oder auch Remote Pair Programming. Dieses wird im Folgenden näher dargestellt.

[22] siehe http://www.c2.com/cgi/wiki?PairProgrammingPingPongPattern

2.3.1 Distributed Pair Programming

Pair Programming hat für virtuelle Teams den offensichtlichen Nachteil, dass beide Team-Mitglieder sich am selben Ort befinden müssen. Das heißt, es ist oft schwierig und kann sehr kostspielig werden, die einzelnen Mitarbeiter zusammen zu bringen, damit diese die Möglichkeit einer Weiterbildung durch die Anwendung von Pair Programming erhalten können. Es müssen in einem Unternehmen dafür im schlimmsten Fall die Mitarbeiter erst eingeflogen werden. Man würde auch für die Dauer der Weiterbildung zusätzliche Räumlichkeiten benötigen, da in einem virtuellen Team nicht zwangsläufig bereits welche existieren. Außerdem müssten die Kosten für die Unterbringung erbracht werden.

Da dies keine sehr praktikable, geschweige denn ökonomische Lösung ist, eignet sich hier das sogenannte Distributed Pair Programming (DPP). Keiner der beiden Teampartner muss dafür extra eingeflogen werden, da die Möglichkeiten des Internets für den Einsatz eines sogenannten Collaborative Real-time Editors, einer Screensharing Software oder auch extra für diesen Zweck geschaffene IDE Plugins benutzt werden können.

Dass Distributed Pair Programming ähnlich effektiv sein kann wie Pair Programming mit Team-Mitgliedern die nebeneinander sitzen, wurde bereits von Baheti et al. [2002] angedeutet. In dem Experiment lieferten die mittels NetMeeting virtuell zusammen arbeitenden Studenten ähnlich gute Ergebnisse wie die real gemeinsam Arbeitenden. Ein direkter Vergleich der Gruppen fand hier jedoch nicht statt. Weitere Untersuchungen auf dem Gebiet wurden von Canfora et al. [2003], Stotts et al. [2003] und Hanks [2004] durchgeführt. Die letzteren beiden hatten ähnliche Resultate zum Ergebnis. Canfora et al. [2003] gestattete den Studenten keine auditive Kommunikation, sodass die dort beschriebenen Ergebnisse stark von den andere Experimenten abweichen: Die Studenten neigten nach initialem Austausch mittels eines Text basierten Chats dazu, wie zwei Einzelprogram-

mierer zu arbeiten, womit wichtige Paarprogrammierungsaspekte fehlten.

Der Verteilungsaspekt des DPP bringt also auch Nachteile mit sich, die teilweise nur schwer erfassbar sind. Neben der Tatsache, dass sich die Entwickler miteinander unterhalten können müssen, was zum Beispiel durch Telefon oder VoIP[23] gewährleistet werden kann, werden noch weitere Anforderungen an die benötigte Software, die zum DPP benutzt wird gestellt. In den genannten Studien kam somit auch zutage, dass den Studenten dort die Möglichkeit fehlte, direkt auf die betreffende Stelle im Quellcode zu zeigen und zu gestrikulieren[24]. Erst Hanks [2004] führte die Möglichkeit des *Zeigens* und *Gestikulierens* mittels eines zweiten Mauscursors, den der Observer bei Bedarf bedienen konnte, ein. Dabei kam eine von ihm modifizierte VNC-Anwendung zum Einsatz (siehe Abbildung 2.4).
Der Einfluss von gegenseitiger Präsenz in virtuellen Umgebungen versus in realer Umgebung und deren Gewährleistung im Rahmen des DPP ist ebenfalls ein noch nicht vollständig erforschter Einfluss. Denn während man in einer realen Umgebung sieht, wenn der Teampartner nicht anwesend ist (sei es physisch oder geistig), kann es in einer virtuellen Umgebung durchaus ein wesentliches Problem sein und damit die Codequalität mindern. Im Projekt *Saros*[25], einem Eclipse-Plugin, welches zur verteilten Entwicklung und Paarprogrammierung verwendet wird, sind bereits erste Ansätze zur Verbesserung der Präsenz enthalten. Diese sind die aktuelle (Tastatur-)Cursorposition sowie der aktuell sichtbare Scrollbereich und offene/gerade betrachtete Dateien des jeweils anderen Partners in Form von farbigen Markierungen an den Dateisymbolen, dem Scrollrand und dem Editorhintergrund darzustellen. Außerdem wird derzeitig an einer weiteren Verbesserung durch das Hinzufügen einer Screensharing-Funktion gearbeitet. Diese zusätzlichen Informationen bereichern die Wahrnehmung der voneinander entfernten Teampartner bezüglich der Aktionen. Dies steigert damit die gegenseitige Präsenz.

[23] so geschehen in Hanks [2004]
[24] siehe Stotts et al. [2003]
[25] siehe https://www.inf.fu-berlin.de/w/SE/DPP

Abbildung 2.4: Eingebettetes, modifiziertes VNC-Programm zum Distributed Pair Programming an der Santa Cruz University of California

Zur Weiterbildung eingesetztes Distributed Pair Programming schafft weitere Probleme: Um Sachverhalte einfacher und leichter verständlich zu erklären, hat es sich bewährt, als Hilfsmittel Skizzen und Schaubilder zu erzeugen, um daran die Funktionsweisen, Aufbau oder Zusammenhänge von Informationen zu verdeutlichen. Dies ist im verteilten Fall jedoch weniger leicht möglich, da hierfür extra Software benötigt wird, um diese zu erstellen und letztendlich zu übertragen. Screensharing, Benutzung von Webcams und Whiteboard Software können mögliche Ansätze sein, dieses Problem anzugehen.

Für die Firma [K] wäre die Anwendung von Disitributed Pair Programming direkt durch einen alternativen Entwicklungsprozess, ähnlich dem, wie er in Abbildung 2.2 zu sehen ist, umsetzbar. Darin wären allerdings dann der Schritt *Implementierung der Änderungen* und die *Zuweisung der Entwickler* entsprechend angepasst. Hier würde DPP direkt in der Implementierungsphase eingesetzt werden können und in der Zuweisungsphase könnte zusätzlich eine entsprechende manuelle Zuweisung der Entwickler zu den einzelnen DPP-Teams vorgenommen werden.
Für größere Entwicklerzahlen und im Rahmen von normaler (nicht Weiterbildungs-)Arbeit würde sich an der Stelle allerdings anbieten, diese Zuweisung auf entweder automatischem oder zumindest semi-automatischem Wege zu bewerkstelligen, sodass das entstandene und entstehende Wissen unter den Entwicklern entsprechend gleichverteilt wird. Zu diesem Zweck könnte eine Ressource-Management-Software entwickelt werden, oder es auf organisatorische Weise, also durch entsprechende Kommunikation des Teampartner-Tauschprozesses umgesetzt werden. Der Entwicklungsprozess aus obiger Abbildung bleibt demnach von der Struktur her so bestehen, wird jedoch um die Wissensaustauschaspekte ergänzt.

Kapitel 3

Iterationen

3.1 Versuchsreihen

Es wurde eine Exploration der unterschiedlichen Lehransätze durchgeführt, welche im verteilten Fall als geeignet befunden wurden. Um herauszufinden, welche diese Eigenschaft aufweisen, wurde zunächst untersucht, mit welchen Möglichkeiten bisher im Internet Weiterbildungen, beispielsweise für die Java-Technologien oder auch für die Arbeit mit Webframeworks, wie CakePHP durchgeführt werden.
Untersucht wurden hierbei exemplarisch insgesamt drei verschiedene und sehr bekannte Technologien: das Ilias Lernmanagementsystem, die JavaServer Pages Technologie und das bereits erwähnte CakePHP Framework. Allen drei Technologien ist gemeinsam, dass sie eine umfangreiche Dokumentation zum Verständnis des Systems bereit stellen[1], welche im Falle von CakePHP und Ilias außer der API-Dokumentation auch noch einen (schriftlichen) Architekturüberblick enthält, sodass man sich über den Aufbau und die Funktionsweisen der Systeme aufklären lassen kann. Beide verwenden in diesem Rahmen auch Architekturmodelle und Diagramme (subjektiv gesehen, sind die-

[1] siehe http://book.cakephp.org/view/13/Basic-Principles-of-CakePHP, http://java.sun.com/javaee/6/docs/api/, http://www.ilias.de/docu/goto_docu_lm_42.html

se im Falle von CakePHP allerdings zu wenig konkret und im Falle von Ilias wenig verständlich). Weiterhin gibt es für JavaServer Pages und CakePHP Tutorials, mit denen man sich direkt mit der Entwicklung in/am System einlernen kann. Diese Tutorials sind beispielgetrieben und (im Falle von CakePHP) als Screencast aufbereitet.

Da diese Versuchsreichen auch einen Nutzen für das Unternehmen selbst haben sollten, bedeutete dies, dass die Erfahrungen direkt praxisbezogen und nutzenorientiert sein sollten. Für diese Versuche wurden folglich Mitarbeiter des Unternehmens herangezogen. Da knapp ein Jahr zuvor bereits eine Weiterbildungsmaßnahme für diese Mitarbeiter durchgeführt worden war, konnte hierbei auf einige Erfahrungen aus dieser Zeit zurückgegriffen werden (insbesondere in Hinsicht auf Formulierung und Detailgrad von Inhalten). Initial sollten sieben Mitarbeiter an dieser Weiterbildungsmäßnahme beteiligt werden, um die Ergebnisse auf eine möglichst breite Ansatzbasis zu stellen. Da jedoch das Unternehmen zum tatsächlichen Startzeitpunkt der Arbeit nicht ausreichend Kapazitäten dazu zur Verfügung hatte, musste die Anzahl auf zwei Mitarbeiter reduziert werden. Zwischen den einzelnen Versuchen wurden sie nicht von ihren sonstigen Aufgaben entbunden und waren Teil des Subunternehmens P.

3.1.1 Vorgehensweise

Es wurde ein drei stufiges Iterationsmodell verwendet, entsprechend des *cognitive apprenticeships*, um zu untersuchen, inwieweit der Lernprozess durch Festlegung der Lernmaterialreihenfolge und -art in Abhängigkeit von den Lernpräferenzen unterstützt, bzw. inwiefern optimiert werden kann. Dafür wurden zunächst die Lernpräferenzen und Vorkenntnisse in Form von Fähigkeitsmatrizen und Fragebögen ermittelt. Mithilfe dieser Daten wurden die einzelnen Iterationen definiert und entsprechend aufbereitet. Die Einzelheiten können Annex B entnommen werden.

Anschließend wurden die drei Iterationen durchgeführt. Zu Beginn jeder Iteration wurden die Anforderungen und das Ziel den Probanden gegenüber definiert, dann die Iteration tatsächlich durchgeführt und anschließend eine Auswertung durch Fragebögen, Interviews oder zu lösende Aufgaben angeschlossen. Für alle drei Iterationen wurden bewusst unterschiedliche Medien und Kommunikationskanäle benutzt, um diese auf Effektivität und Effizienz zu prüfen.

Dabei diente die erste Iteration der Bildung von Grundlagen und der Gewinnung eines generellen Überblicks über die Materie. Dazu wurden entsprechend der zuvor stattgefundenen Untersuchung der Weiterbildungsmöglichkeiten den Probanden Architekturdiagramme und -beschreibungen, sowie eine API-Dokumentation zum Lesen vorgelegt. Am Ende der ersten Iteration wurde mithilfe eines Fragebogens und eines anschließenden Interviews das Verständnis der Inhalte geprüft.

Die zweite Iteration führte in die Anwendung der zuvor gelernten Sachverhalte ein. Dazu wurden den Probanden drei aufeinander aufbauende, rund 20- bis 30-minütige Screencasts bereitgestellt, die alle zur Benutzung/Anwendung der in der ersten Iteration vorgestellten Technologien wichtigen Aspekte erläutern und das Ganze an einem einfachen Beispiel erklären sollten. Am Ende der zweiten Iteration wurde mithilfe eines Interviews und einer zu lösenden Aufgabe das Verständnis der Inhalte geprüft.

Die dritte Iteration diente der Übertragung der gelernten Konzepte und deren Anwendung auf andere Sachverhalte und bestand darin, dass ein Code-Review eines bestehenden (den Probanden bekannten) Systems mittels Screensharing und einer Sprachkonferenzsoftware durchgeführt wurde. Diese Iteration hatte im Wesentlichen die Detail-Planung einer vollständige Portierung des bekannten Systems in das in den vorigen Iterationen vorgestellte System zum Ziel.

In den ersten beiden Iterationen wurden für die jeweiligen Lernmaterialien Vor-Tests mit unabhängigen Personen durchgeführt, um sowohl das generelle Verständnis der Inhalte zu überprüfen als auch die Zeit abschätzen zu können, die für die Durcharbeitung der Materialien zur Verfügung gestellt werden sollte, abschätzen zu können. Mangels Lernmaterialien wurde darauf in der dritten Iteration verzichtet.
Im Folgenden wird nun detaillierter dargelegt, wie die Datenerhebung in den einzelnen Schritten in den jeweiligen Iterationen durchgeführt und wie diese Daten ausgewertet wurden. Anschließend wird eine mögliche Interpretation der erhobenen Informationen präsentiert und mögliche Schlussfolgerungen angeführt.

3.1.2 Lernpräferenzen und Vorkenntnisse

Zur Vorbereitung der eigentlichen Iterationen wurden zunächst die Vorkenntnisse in möglichst vielen Wissensbereichen abgefragt. Dabei waren diese unterteilt in gesprochene Sprachen, Programmiersprachenkenntnisse, Wissen um verschiedene Softwareentwicklungsprozesse, Internet-Technologien, Software-Modellierung, Entwicklungsframeworks, Entwicklungsumgebungen, Datenbanksysteme, Betriebssysteme, Content-Management Systeme und sonstige Anwendungen. Es wurde darauf hingewiesen, dass diese Bereiche durch weitere Einträge ergänzt werden könnten, sofern etwas nicht in der Liste enthalten war.
Für jeden Eintrag in diesen Bereichen sollten die Probanden dann sowohl ihr subjektiv vorhandenes Wissen (wie gut kennen sie sich damit aus), als auch die Erfahrung (wie lange haben sie sich damit auseinander gesetzt) angeben. Die Werte 1 – 5 standen dabei für das subjektiv vorhandene Wissen, wobei 1 *sehr gut*, 2 *gut*, 3 *zufriedenstellend*, 4 *Grundlagen* und 5 *keine Kenntnisse* bedeutete. Die Werte 10 – 50 sind entsprechend für die Erfahrung äquivalent aufgeteilt gewesen: 10 bedeutete *seit 5 Jahren oder mehr*, 20 entsprach *zwei bis vier Jahre*, 30 *ein paar Monate bis zu einem Jahr*, 40 *einige Wochen*, während 50 *noch nie damit beschäftigt* bedeutete. Beides zusammenaddiert (also subjektiv vorhandenes Wissen + Erfahrung) diente als Indikator für den tatsächlichen Wis-

sensstand in den enthaltenen Einträgen. Die ausgefüllten Listen wurden dann anschließend vom Autor in eine sogenannte Skillmatrix überführt. Die Struktur der Liste, sowie die der anschließenden Skillmatrix ist den offiziellen Sourceforge.net-*Skill Profiles*[2] angelehnt.

Weiterhin wurde den Probanden als Vorbereitung für die eigentlichen Iterationen ein Fragebogen zur Beantwortung vorlegt. Dieser diente der Bestimmung der Lernpräferenzen der Teilnehmer. So wurden sowohl der soziale Hintergrund als auch das generelle Lerninteresse abgefragt.
Der soziale Hintergrund fragte nach dem Grad der Internationalisierung des Teilnehmers, dem Bildungsgrad, Introvertiertheit/Extrovertiertheit. Das generelle Lerninteresse ging darauf ein, ob sich der Teilnehmer für neue Technologien interessiert und wie oft er üblicherweise neue kennenlernt. Wichtig hierbei ist auch, wie lange er bereits in dem Bereich arbeitet, da dies das Interesse an neuen Technologien beeinflussen kann. Zuletzt ist auch Teil dieses Fragebereichs, ob der Proband intuitives Lernen, also eine Neigung zu *abstract reflective* Lernmustern hat, oder experimentelles Lernen bevorzugt, also eine Neigung zu *concrete active* Lernmustern hat.

Beide Probanden gaben in ihren Fragebögen an, dass sie keinen besonders großen Internationalisierungsgrad haben und seit vier Jahren im IT Sektor tätig seien. Proband [N] gab bei der Frage nach seinem Ausbildungsort eine grosse, indische Universität an, während Proband [S] lediglich ein Ingenieurs-College besuchte. Beide Probanden stammen aus einem kollektivistischen Land (Indien) und bevorzugen das intuitive Lernen, also zuerst theoretische Grundlagen und anschließend erst praktische Arbeit. Nach dieser Auswertung stand die letztendliche Reihenfolge der Iterationen fest.

[2] mit vorhandenem Sourceforge.net Account kann man in eingeloggtem Zustand über https://sourceforge.net/people/editprofile.php seine perönliche Skill-Liste anlegen und verwalten

3.2 Erwartete Ergebnisse

Da diese Experimente als eine Exploration der Möglichkeiten der Weiterbildung ausgelegt sind, wird erwartet, dass die Probanden im Anschluss ein gutes Verständnis des zu benutzenden Systems haben und selbstständig in der Lage sind, auf Grundlage von vordefinierten Aufgabenpaketen das vorhandene System in das Neue zu portieren. Die Iterationen dienen zunächst dazu, diese Grundlagen zu schaffen und zu verinnerlichen, aber letztendlich auch dazu, die benötigten Fähigkeiten zu überprüfen. Trotz unterschiedlicher Ausbildung und eines unterschiedlichen Erfahrungsreichtums im Umgang mit Modellierung und Modellen wird erwartet, dass die verwendeten Techniken und Technologien bei beiden Probanden ähnlich effektiv sind.

3.3 Erste Iteration: Einführung in die Materie

3.3.1 Vorbereitung

Zunächst wurde die Dokumentation selbst inklusive der nötigen Architekturdiagramme entwickelt. Diese wurden zu zwei getrennten Dokumenten geformt, welche von den Probanden in der für sie angenehmsten Reihenfolge gelesen werden sollten.

Dokument 1 beinhaltete die generelle Architektur des Systems, welches den Probanden näher gebracht werden sollte. Dabei sollte dieses Dokument den Aufbau und die Zusammenhänge der Systemkomponenten grafisch in Form von FMC-Diagrammen und textuell in Form von Beschreibungen der Diagramme erläutern. Das Dokument war so aufgebaut, dass es zunächst einen Systemüberblick bot und beschrieb, welcher die relevanten Komponenten aufführte und grob deren Sinn und Zweck darstellte. Dem folgend wurden die zuvor genannten Komponenten ein-

zeln im Detail erklärt, wobei Wert darauf gelegt wurde, sowohl die Funktionsweise, als auch die Interaktion mit den anderen Komponenten zu beschreiben. Neue Begriffe wurden eingeführt und sukzessive strenger umrissen. Deren Verwendung im Kontext der Komponenten, die sie betreffen, wurde ebenso beschrieben. Die Ziele der Modellierung der Architektur waren dabei die folgenden:
Die Architekturdiagramme sollten...

- mit minimaler Zeit selbst für Laien der Modellierungsmethode erlernbar/lesbar und erzeugbar sein (um das Lesen, Verstehen und Entwickeln der Dokumente zu vereinfachen und zu beschleunigen)
- präzise die wesentlichen Strukturen des Systems beschreiben und unwesentliche Merkmale, wie Implementierungsdetails, nicht beinhalten (um dadurch vorhandene Komplexität zu reduzieren und damit das Systemverständnis zu steigern)
- eine einheitliche Terminologie für möglichst viele Aspekte der Systemmodellierung verwenden (um die Kommunikation über das System zu vereinfachen)

Dazu wurden unterschiedliche Modellierungsnotationen betrachtet und anhand der oben genannten Ziele als geeignet/ungeeignet eingestuft. Die betrachteten Modellierungsnotationen sind dabei: BPMN[3], UML[4], FMC[5] und "Powerpoint"(also eine eigene, auf Powerpoint-typischen Symbolen und Elementen basierende Modellierung). Die Auswahl der Notationen für die Einstufung wurde aufgrund des Bekanntheitsgrades und der Erfahrungen in den dem Autor bekannten Notationen durchgeführt. Vor- und Nachteile der jeweiligen Notationen sind in Tabelle 3.1 aufgelistet.

Die Wahl fiel auf FMC, da ein geeignetes Tool vorlag (OpenOffice Draw Templates und Microsoft Visio Stencils)

[3]Business Process Modeling Notation
[4]Unified Modeling Language
[5]Fundamental Modeling Concepts

Notation	Pro	Kontra
BPMN	einfach zu verstehen / lesen, einheitliche Terminologie möglich	bescheibt weniger die Struktur, als vielmehr die Abläufe ⇒ falscher Fokus
UML	präzise Beschreibung der Systemstruktur, Komplexität reduzierbar, hoher Bekanntheitsgrad	viele verschiedene Diagramm-Typen zur Modellierung ⇒ lange Einarbeitungszeit für Leser; uneinheitliche Terminologie
FMC	wenige Diagramm-Typen, einfach zu verstehen, präzise Systemstruktur beschreibbar, einheitliche Terminologie	geringer Bekanntheitsgrad, wenig geeignete Tools vorhanden
Power -point	einfach zu verstehen / lesen und zu erzeugen, präzise Beschreibung möglich, einheitliche Terminologie möglich	zeitaufwändig zu produzieren, da Konsistenz und Präzision "manuell"gewährleistet werden müssen

Tabelle 3.1: Modellierungsnotationen - Vergleich

und die geringe Bekanntheit durch die Einfachheit der Notation ausgeglichen werden konnte.
Im Anhang an das erste Dokument war eine Kurzeinführung in die wichtigsten Konzepte von FMC, auf welche in der Einleitung des Dokuments verwiesen wurde. Das Dokument hatte insgesamt einen Umfang von 15 DIN A4-Seiten.

Dokument 2 beinhaltete die Beschreibung der (internen) API, welches die Details für die Arbeit in dem System durch Erläuterung von Syntax und Semantik von Befehlen

und deren Verwendung erklärte. Dabei wurde die Struktur des Dokuments entsprechend der Struktur einer weit verbreiteten API-Dokumentation (PHP API Dokumentation) übernommen. Diese ist derart aufgebaut, dass zunächst der Name des zu beschreibenden Befehls/Themas genannt wird, anschließend die Funktionsweise beschrieben wird, gefolgt von der erlaubten Syntax (in erweiterter Bacchus-Naur-Form). Dem folgt in tabellarischer Form die Erklärung der jeweiligen Syntax-Elemente (Parameter, default-Werte, etc...). Im Anschluss daran kommen ein oder mehrere Beispiele für dessen Verwendung. Das zweite Dokument hatte insgesamt einen Umfang von 25 DIN A4-Seiten.

Die Vortests mit zwei unabhängigen (firmenfremden) Versuchspersonen wurden ebenfalls durchgeführt. Die Hinweise und kleinere Umformulierungen, die sich daraus ergaben, wurden entsprechend mit in die Dokumente eingearbeitet.

3.3.2 Durchführung

Die Dokumente sollten ursprünglich in dynamischer Art und Weise per Website zur Verfügung gestellt werden, sodass die einzelnen Inhalte via Hyperlinks miteinander vernetzt sind. Dadurch sollte unter anderem eine Messung der Aufenthaltsdauer der Probanden auf den jeweiligen Inhalten ermöglicht werden und somit detailliertere Hinweise auf das Nutzungsverhalten für diese Inhalte ermittelt werden.

Aufgrund eines zu knapp bemessenen Zeitrahmens für die Umsetzung der Dokumente in diese HTML-Form konnte dieses nicht direkt so durchgeführt werden, sondern musste ein "konventionellerer" Weg gegangen werden: Die Inhalte wurden als zwei PDF-Dokumente per eMail an die Probanden übertragen. Als Teil davon wurde ebenfalls eine Erklärung der gewünschten und erwarteten Vorgehensweise beim Durcharbeiten der Dokumen-

te hinzugefügt. Dies beinhaltete eine frei definierbare Lese-Reihenfolge als auch das bewusste Auslassen/nicht-Lesen von Inhalten, sofern die Probanden diese als *uninteressant* befanden[6].

Zum Durcharbeiten der Dokumente hatten die Probanden zwei Tage Zeit. Nachdem jeder der Teilnehmer beide Dokumente durchgelesen hatte, haben sie sich zusammen gesetzt und darüber diskutiert, um ein Feedback zu erstellen, welches sie dem Autoren anschließend per eMail zurück sandten. Dieses gesammelte Feedback beinhaltete Kommentare zu den Dokumenten (vor allem inhaltlich) und einige Fragen, die auf Verständnisschwierigkeiten schließen ließen[7].

Nachdem das Feedback vom Autor analysiert worden war, ist ein Interview mit dem ersten der beiden Probanden durchgeführt worden. Aufgrund von technischen/organisatorischen Schwierigkeiten auf Seiten der Probanden konnte ein zeitnahes Interview mit dem zweiten Teilnehmer nicht direkt über einen geeigneten Nachrichtendienst durchgeführt werden, sodass das zweite Interview[8] via eMail geführt wurde (die Antworten auf diese eMail [9] wurden fünf Tage später empfangen).

3.3.3 Auswertung

Die Vortests mit den Versuchspersonen ergaben, dass diese Probleme beim Verständnis der Arbeitsweise des Systems hatten. Die Reihenfolge, in der beide Dokumente gelesen wurden, war in beiden Fällen zuerst die Architektur-Beschreibung und anschließend die API-Dokumentation. Die Interviews konnten hier direkt im Anschluss durchgeführt werden und zeigten, dass trotz der Probleme beim Verständnis der generelle Systemaufbau verstanden worden war – ebenso die wesentlichen Aspekte der API. Dass

[6] siehe Appendix A, Abschnitt A.1.1
[7] siehe Sektion A.1.2
[8] siehe Sektion A.1.4
[9] siehe Sektion A.1.5

die Arbeitsweise des Systems nicht verstanden werden konnte, kann darauf zurück geführt werden, dass beide Vorversuchspersonen - ganz im Gegensatz zu den eigentlichen Probanden - die Lernpräferenzen *experimental learning / sensing learning*[10] inne hatten, was den Fragebögen in B.1.1 entnommen werden kann.

Das erste Feedback[11] (bereits vor dem Interview erhalten) belegt, dass selbst nach Zusammenlegung der Erkenntnisse der beiden Probanden nicht alle Aspekte im Detail verstanden wurden. Proband $[N]$ gab an, dass vor allem innerhalb der API-Dokumentation einige Inhalte nur unzureichend erklärt seien: *But Its better if we get an example of defining the custom Tags with detailed explanation*. Im Hinblick auf die anderen Fragen, die innerhalb dieses ersten Feedbacks auftauchten (zum Beispiel *In which scenario simpleCheckSum will be useful?* und *Is it possible to have Limits and Group By Clause here?*) kann davon ausgegangen werden, dass einige Inhalte mit zu geringem Detail-Grad erklärt wurden.
Die Modelle und Architekturbeschreibung des ersten Dokuments wurden von Proband $[N]$ als adäquat und einfach zu verstehen charakterisiert: *Overall the Architecture is really simple to understand* [...] *The Agents* [...] *explains clearly the role they play in the system.*
Proband $[N]$ gab dabei an, die Dokumente in der Reihenfolge, zuerst Architektur-Überblick, dann API-Dokumentation gelesen zu haben: *I started with Architecture and Proceeded with API*. Diese Versuchsperson gab zuvor in seiner Skill-Matrix an, bereits seit langer Zeit sich mit komplexeren Modellierungsnotationen (vor allem UML) zu beschäftigen und darin gute und stabile Kenntnisse zu besitzen (Wert 12). Der Proband räumte jedoch ein, dass für einen neuen Entwickler einige weitere lebensnahe Beispiele innerhalb des Architekturüberblicks benötigt würden, um diese weitestgehend zu verstehen: *On Understanding Level this Document is more than enough for us. But for a New User [of this framework] / Developer, still it requires some real time examples*.

[10] siehe DeZure [2000], S. 83 – 84
[11] siehe Sektion A.1.2

Proband [S] gab an, er hätte sich in zufälliger Art durch die Dokumente gearbeitet (*Initially i just traversed both the docs in a random fashion.*). Dadurch scheint der Zusammenhang zwischen den Modellen und dem Erklärten nicht erkannt worden zu sein. Der Proband gab an, dass die Modelle von der Struktur des Systems unabhängig seien.

> *The Models (FMC) and Structure of Framework currently seems to be independent. Its better of the Document explains the Framework much more detail in conjunction with the specified model*[12]

Eine weitere Erklärung dafür kann sein, dass die Modellierungsnotation unzureichend verstanden worden war. Laut dessen Einträgen in der Skillmatrix (siehe B.2) sind dem Probanden (komplexe) Modelle mit definierter Notation (z.B. UML) nicht geläufig (fast überall Wert 55 in Bereich Modellierung).
Weiterhin ist durch den größeren zeitlichen Abstand zwischen Durchführung der Iteration und Beantwortung der eMail die Wahrscheinlichkeit, dass diese Person sich an alle Details, nach denen gefragt wurde noch erinnern konnte, gering. Durch Vorkenntnisse aus anderen (ähnlichen) Systemen (in diesem Feedback von der Versuchsperson wurden zusätzliche Kenntnisse darin erwähnt), scheint das generelle Verständnis leichter gefallen zu sein: *Regaring the framework structure it seems it can address any type of complex scenarios.*
Bezüglich der API-Dokumentation wurden, trotz direkter Frage danach, von dem Probanden keinerlei direkte Aussagen gemacht.

Zusammenfassend kann man also schließen, dass das neue System im Großen und Ganzen von den Probanden verstanden wurde, vor allem aus architektureller Sicht ist das Zusammenspiel der einzelnen Komponenten klar geworden und damit die generelle Arbeitsweise des Systems. Die Modelle hatten dabei allerdings nur eine unterstützende Rolle, da teilweise nicht ausreichende Kenntnisse vorhanden waren, um diese zu interpretieren.

[12] siehe Sektion A.1.5

Durch die textuelle Beschreibung ist dieses Manko allerdings teilweise ausgeglichen worden. Das bedeutet, dass in künftigen derartigen Dokumenten eine bessere Beschreibung der Modellnotation nötig ist, da dies für ungeübte Personen, die wenig Erfahrung mit komplexeren Modellnotationen haben, insgesamt zu einem tieferen Systemverständis führen sollte.
Die API-Dokumentation sollte weitere, realitätsnahe Beispiele und detailliertere Beschreibungen zu den einzelnen Elementen und deren Funktionsweise beinhalten. An dieser Stelle traten die meisten Probleme auf.

3.4 Zweite Iteration: Anwendung des Gelernten

3.4.1 Vorbereitung

Zur Vorbereitung auf die zweite Iteration - welche für die Probanden das Ziel hatte, sie mit konkreter Entwicklungsarbeit im neuen System vertraut zu machen - wurden eine Reihe von Screencasts aufgenommen und den Versuchspersonen zur Verfügung gestellt. Die Screencasts sollten dabei die Interaktion und Vorgehensweise, sowie die wesentlichen Eigenschaften der für das zuvor in Iteration eins vorgestellte System entwickelten IDE dienen. Für die Aufnahme der Screencasts wurde zunächst eine Evaluierung für eine geeignete Screen recording software durchgeführt. Untersucht wurden hierbei das Camtasia Studio 6[13] und QuickTime X[14] . Diese wurden in einer Gegenüberstellung beide betrachtet und auf ihre Funktionsweisen und die ideale Art des Einsatzes hin getestet.

Da die zu entwickelnden Filme via Webbrowser direkt aufrufbar sein sollten (sodass die Erstellung von zusätzlichen HTML-Seiten, in welche die Videos eingebettet hätten werden müssen entfiel), war eine wichtige An-

[13] http://www.techsmith.com/camtasia.asp
[14] http://www.apple.com/macosx/what-is-macosx/quicktime.html

forderung, dass sie in einem entsprechend unterstützten Format produziert werden sollten. Da sowohl der Opera[15] Browser als auch der Google Chrome[16] und Mozilla Firefox[17] Browser nativ eine solche Unterstützung für das OGG-Theora-Format besitzen und für andere Browser wie Safari[18] und dem Internet Explorer[19] Erweiterungen existieren, um dieses Format abspielen zu können, wurde dieses als das Zielformat definiert. Weiterhin ist die Qualität bei gleicher Datenrate höher im Vergleich zu Flash-Videos, die ansonsten hätten benutzt werden müssen, um sie möglichst einfach anschaubar zu machen.

Produkt	Vorteile	Nachteile
Camtasa Studio	ist sehr umfangreich, bietet sehr viel Funktionalität ("Smart Focus" [auto zoom], Audio-Spur und Video-Spur unabhängig voneinander im Nachhinein bearbeitbar, Klicks hervorheben, Untertitel hinzufügen)	kompliziert(er) zu bedienen, kostenpflichtig (außer in 30-tägiger Testversion), kein Speichern als OGG-Theora möglich, lief bei den Tests nur ruckelnd
QuickTime X	bietet grundlegende Funktionen zum Aufnehmen von Screencasts mit gleichzeitiger Stimmaufnahme, nachträgliches Schneiden möglich, sehr einfach zu bedienen, kostenlos	kein Speichern von OGG-Theora möglich, nur grundlegende Funktionalität gegeben

Tabelle 3.2: Vergleich von Screencast-Recorder-Software

[15] http://dev.opera.com/articles/view/a-call-for-video-on-the-web-opera-vid/
[16] http://www.softsailor.com/news/3787-3787.html
[17] http://mozillalinks.org/wp/2008/07/native-ogg-vorbis-and-theora-support-added-for-firefox-31/
[18] http://xiph.org/quicktime/
[19] http://www.theora.org/cortado/

Entsprechend dieser Gegenüberstellung (siehe Tabelle 3.2) ist das Ergebnis, dass Camtasia Studio für einen effektiven Einsatz einige größere Konfigurationsmaßnahmen benötigt, während QuickTime direkt ohne weitere Einstellungen sofort gute Ergebnisse lieferte. Allerdings konnte bei beiden Produkten keine direkte Export-Funktion in das OGG-Theora-Format gefunden werden.
Die Entscheidung fiel folglich auf QuickTime X, auch, da die erweiterten Features von Camtasia Studio nicht benötigt wurden. Die Screencasts sollten in einem Stück aufgenommen werden und stets den gesamten Bildschirmbereich zeigen. Nachbearbeitung war nur in geringem Maße nötig.
Die aufgenommenen Videos wurden mittels ffmpeg, einem Konsolendienstprogramm, in das gewünschte OGG-Theora-Format überspielt.

Insgesamt wurden 3 Screencasts aufgenommen, welche jeweils eine Länge von 20 - 28 Minuten hatten. Inhaltlich sind sie aufeinander aufbauend entwickelt worden, sodass der erste Teil als eine Art Einführung angesehen werden kann, der zweite Teil etwas komplexere Themen dazu behandelt und der letzte Teil die fortgeschrittenen Inhalte vorstellte.
Diese Aufteilung der Inhalte ist abgeleitet von unterschiedlichen Tutorialen, die im Netz zu finden sind und ebenfalls komplexere Systeme zum Thema haben[20].
Anschließend wurden diese Videos auf einen Firmenserver übertragen und waren dort entsprechend für die Probanden über ihre Webbrowser abrufbar.

Damit der Grad des Systemverständnisses und der Benutzbarkeit/Klarheit der Screencasts untersucht werden konnte, wurde entsprechend eine Testaufgabe entwickelt, die inhaltlich mit dem in den Screencasts erklärten Vorgehen lösbar sein sollte.
Um diese Aufgabe schlussendlich lösbar zu machen, wurde zuletzt noch das von den Screencasts erklärte System inklusive der IDE online gestellt und verfügbar gemacht.

[20] siehe zum Beispiel http://java.sun.com/j2ee/1.4/docs/tutorial/doc/About.html

Wie zuvor in der ersten Iteration wurden auch hier die Lernmaterialien (also in diesem Fall die Screencasts) vor dem eigentlichen Versuchsdurchlauf mit den tatsächlichen Probanden an den zwei "Vorversuchspersonen"getestet. Auch wurde die Durchführbarkeit der Testaufgabe sowie die Gestalt der Aufgabenstellung geprüft. Die Aufgabenstellung wurde den Ergebnissen entsprechend angepasst, bevor die entsprechenden Daten zur Verfügbarkeit und die Aufgabenstellung an die Probanden weitergegeben wurden.

3.4.2 Durchführung

An die Probanden wurde im Zuge der Durchführung der zweiten Iteration anschließend eine eMail gesandt. Diese enthielt präzise Anweisungen, wo die neuen Lernmaterialien zu finden seien, was damit zu tun sei und was das Ziel der Iteration ist. Zusätzlich wurde dies noch ergänzt durch die Aufgabenstellung inklusive der nötigen Anforderungen, die mithilfe der Screencasts gelöst werden sollten, den persönlichen Zugangsdaten für das System und dessen Adresse (letztere in einer anschließenden Extra-eMail).
Die Einrichtung einer Zugangsbeschränkung gestattete es, das Verhalten der einzelnen Probanden innerhalb des Systems zu verfolgen und damit sowohl den aktuellen Fortschritt als auch die benötigte Zeit besser messen zu können, als es zuvor in der ersten Iteration möglich gewesen wäre. So wurde bei jeder Änderung, die ein Proband vornahm in der Datenbank der Ort der Änderung, das Identifikationsmerkmal des Probanden sowie der Zeitpunkt vermerkt.

Die Teilnehmer sich dann ein erstes Mal gemeinsam die Screencast an und verschafften sich einen Eindruck von dem Arbeiten im Zielsystem[21]. Nach dem ersten Betrachten der Lernmaterialien wurden die einzelnen Screencasts nach Aussagen der Probanden für die Entwicklung der Testaufgabe 3 – 4 Mal angesehen[22]. Innerhalb der Screencasts wurde die Seek-Funktion des Videoplayers benutzt, um beim

[21] siehe Abschnitt A.2.3
[22] siehe A.2.3

erneuten Anschauen die ersten Schritte überspringen zu können.
Es wurde also bereits während des Betrachtens der Screencasts damit begonnen, die Aufgabe zu lösen. Die Lernmaterialien aus der ersten Iteration wurden dabei nicht explizit berücksichtigt/zurate gezogen.

Da während der Zeit der zweiten Iteration ein Provider-Umzug stattfand - und sich dementsprechend der darunterliegende Server änderte, mussten sämtliche Materialien auf dem neuen Server ebenfalls eingespielt und schließlich den Probanden bekannt gemacht werden. Ihre bis dahin getätigte Arbeit ging durch die Migration der Datenbank nicht verloren, jedoch kam es durch diesen Umstand zu einer zeitlichen Verzögerung der Entwicklung der Testaufgabe.

3.4.3 Auswertung

Die Probanden benötigten insgesamt 5 Tage für die geforderte Aufgabe. Dabei wurde am ersten Tag nur knapp eine Stunde mit Entwicklungsarbeit verbracht. Dies steigerte sich bis einschließlich des dritten Tages auf mehrere Stunden (der gesamte dritte Tag wurde zur Entwicklung verwendet). Der vierte Tag war dann der Umzug, sodass an diesem kein Fortschritt erzielt werden konnte. Am letzten Tag wurde erneut eine weitere Stunde investiert und die Testaufgabe abschließend von den Probanden als fertig deklariert.

Es lässt sich also aufgrund des Zeitaufwandes feststellen, dass reine Videos als Tutoriale für reale Entwicklungen nur bedingt geeignet sind. Diese geben zwar einen guten und vollständigen Überblick über die Materie, gestatteten es den Teilnehmern sich einfacher und schneller zu orientieren und die Arbeitsweise besser zu verstehen, haben jedoch auch ein Manko: Das Nachschlagen von Informationen und Schritten, die im Screencast vorgestellt wurden, gestaltet sich als langwierig, da die gesuchten Schritte und

Informationen erst im Video *aufgespürt* werden müssen und anschließend der Screencast pausiert werden muss, um den Folgeschritt nicht zu verpassen bzw. ihn ungewollt zu überspringen.

Eine Möglichkeit das Problem zu umgehen ist jeden Screencast in Unterkapitel zu teilen, sodass der Lernende von Schritt zu Schritt springen kann ohne diese erst suchen zu müssen. Dieser Teilvorgang kann unter anderem durch *physische* Teilung der Datei in mehrere kleinere Dateien (jede Datei entspräche hierbei einem Unterkapitel) erfolgen, alternativ allerdings auch durch Annotation von Meta-Informationen, die in einen entsprechend mit dazu passenden Verarbeitungsmöglichkeiten ausgestatteten Videoplayer geladen werden. Letzterer Ansatz ist sicherlich der kompliziertere, bietet jedoch auch bessere Erweiterungsvarianten[23].

Eine weitere Möglichkeit für eine bessere Navigation innerhalb der Informationen ist eine zusätzliche Bereitstellung derselben in Form einer textuellen Beschreibung der Schritte und des Vorgehens, welche dann entsprechend die genannten Informationen mit enthalten sollten. Diese kann dann mittels der vom Browser oder auch von Internet-Suchmaschinen zur Verfügung gestellten Navigations-/Suchfunktion entsprechend herangezogen werden, ohne dass im Video die jeweilige Position heraus gesucht werden muss.

Weiterhin bemerkenswert ist der Unterschied der Umsetzungsgeschwindigkeit der Vortest-Personen im Vergleich zu den eigentlichen Probanden. Hier fällt auf, dass die Umsetzungsgeschwindigkeit insgesamt deutlich unter einem Tag lag (bei 4 Stunden). Da die in der Testaufgabe betroffenen Bereiche der Skillmatrix nahezu identisch sind, jedoch die Lernpräferenzen sich deutlich unterschieden, können letztere als Ursache dieser Diskrepanz verantwortlich sein.

[23] mehr Informationen dazu (bereits im Produktiveinsatz an mindestens zwei Universitäten) findet man unter http://www.tele-task.de/

3.5 Dritte Iteration: Selbstständiges Schlussfolgern

3.5.1 Vorbereitung

Die Vorbereitung auf die Durchführung der dritten Iteration sah so aus, dass zunächst verschiedene Möglichkeiten evaluiert wurden, um in verteilter Weise ein Code-Reviewing durchzuführen. Als erste Wahl wurde das vom Lehrstuhl im Rahmen zahlloser Diplom- und Bachelorarbeiten entwickelte Tool *Saros*, einem Plugin für die Entwicklungsumgebung Eclipse, welches zum Zwecke der verteilten Paarprogrammierung erschaffen wurde, identifiziert. Die ersten beiden Versuche dieses Tool einzusetzen, schlugen – trotz Unterstützung und Intervention des Entwickler-Teams – jedoch fehl, sodass kurzfristig weitere Möglichkeiten evaluiert werden mussten.

Da in der Firma bereits zur Kommunikation unter anderem die Software *Skype*[24] eingesetzt wurde, war eine mögliche Wahl das Screensharing-Feature von Skype zu diesem Zweck zu verwenden. Eine andere Möglichkeit war, einen VNC-Server[25] einzurichten und entsprechend über einen VNC-Client den Inhalt des Bildschirms und die Kontrolle, sofern benötigt, an die Probanden zu übergeben. Da zusätzlich zum Bildschirminhalt auch eine direkte Kommunikation stattfinden sollte, würde man hierbei jedoch außerdem noch auf ein geeignetes Kommunikationsprogramm zurückgreifen müssen.

Skype vereint also beide Technologien: Kommunikation und Screensharing. Jedoch ermöglicht es im Gegensatz zu VNC lediglich Screensharing, ohne den Zustand des entfernten Systems verändern zu können. Die Bandbreite ist hingegen optimal ausgenutzt, da der Applikation bekannt ist, wie hoch die Datenrate für Screensharing und die für

[24] http://www.skype.com/

[25] für mehr Informationen zu VNC, siehe http://en.wikipedia.org/wiki/Virtual_Network_Computing

die Kommunikation sein darf.
VNC hingegen hat den Nachteil, dass man, sofern man nicht einen extra VNC-Server installieren will[26], erst einen geeigneter Client auf der entfernten Seite installieren und eingerichten muss. Zudem ist keine direkte und einfache Kommunikation allein mittels VNC möglich. Die Wahl fiel also letztendlich auf Skype als Medium zur Durchführung.

Die weitere Vorbereitung bestand in einer Planung der durchzugehenden Systemkomponenten des Altsystems – ähnlich einem Code-Review, jedoch mit dem Ziel, den Probanden eine kurze Einleitung in die zu erledigenden Aufgaben zu geben – sodass die Probanden die aus den voran gegangenen Iterationen gewonnenen Erkenntnisse und das erlangte Wissen auf das Altsystem anwenden konnten, um so die Übertragung der Technologie in das neue System anzustoßen. Es wurden hierbei nur diejenigen Systemkomponenten ausgewählt, die besondere Aufmerksamkeit bei der Portierung bedüften, da diese entweder den Einsatz von neuen Technologien, wie sie nur im Zielsystem existieren, benötigen, oder restrukturiert werden müssten.

3.5.2 Durchführung

Der erste weitere Versuch diese Iteration durchzuführen, schlug erneut fehl, da zu dem Zeitpunkt am Standort der Probanden ein Übermaß an Regen einen weiträumigen Stromausfall verursacht hatte, und der eingesetzte Stromgenerator bei Beginn des Versuchs nur noch eine knappe halbe Stunde laufen konnte. Daher wurde diese Zeit dazu genutzt, das weitere Vorgehen zu kommunizieren und ein neuer Zeitpunkt für diese Sitzung anberaumt.

Nachdem geklärt war, wie der Ablauf aussehen wird, haben sich beide Probanden, wie vereinbart, zwei Tage später am Rechner von Proband [N] versammelt und die Verbindung kam endlich zustande. Während der gesamten Sit-

[26] auf dem Host war Mac OS X installiert, welches bereits einen einsatzbereiten VNC-Server beinhaltet

zung war der Bildschirminhalt des Autoren mit geöffneter Entwicklungsumgebung und dem jeweiligen Quellcode-Bereich für die Probanden sichtbar. Zudem wurde die Kommunikation vorrangig über eine Audio-Verbindung durchgeführt.

Der Vorgang dann war intuitiv in unterschiedliche Phasen gegliedert. Zunächst ging der Autor einen zu portierenden Systemteil durch und beschrieb und zeigte ihn anhand des Quelltextes. Während dieser Erklärungsphasen schrieb Proband [N] die für die Portierung notwendigen Informationen auf einem Blatt Papier mit.
Nach einer solchen Phase begann die zweite, die Dialog-Phase. Hier wurden von den Probanden [N] und [S] Fragen gestellt und Verständnisprobleme geklärt, die den zu portierenden Systemteil betrafen. Außerdem fragte der Autor die Probanden nach der Art, wie diese die Portierung vornehmen würden. Die entsprechenden Antworten wurden vom Autoren anschließend bewertet und Optimierungen und Vereinfachungsmöglichkeiten genannt, welche wiederum von den Probanden niedergeschrieben wurden.
In Phase drei wurde für die Portierung dieses Systemteil dann in entsprechende Aufgabenpakete unterteilt, sodass hier der Arbeitsumfang für jedes umrissen wurde. Dies geschah in direkter Abstimmung zwischen Autor und Probanden.
Nach diesen drei Phasen begann erneut die Phase eins, jedoch mit einem anderen Systemteil, sodass zum Schluß alle zuvor ausgewählten Systemkomponenten durchlaufen, erklärt und analysiert wurden.

Während der Durchführung traten wiederholt Unterbrechungen im Kommunikationsfluß auf, was auf eine instabile Internetverbindung zurückgeführt werden kann. So verschlechterte sich zeitweise die Qualität der Screensharing-Verbindung auf Seiten der Probanden [N] und [S] dermaßen, dass die Verbindung manuell unterbrochen und wieder neu initiiert werden musste. Auch die Audio-Verbindung konnte nicht durchgängig aufrecht erhalten werden, da die komplette Verbindung teilweise unterbrochen wurde. Daher wurde solange, bis die Audio-

Verbindung wieder zuverlässiger funktionierte, der Austausch von Textnachrichten genutzt, um mit den Inhalten voran zu kommen. Die Dauer der Sitzung wurde davon jedoch nur geringfügig beeinflußt. Insgesamt ging diese über einen Zeitraum von zweieinhalb Stunden.

3.5.3 Auswertung

Innerhalb der Dauer der zweieinhalb Stunden, die diese Iteration in der Durchführung in Anspruch nahm, gab es – abgesehen von den Verbindungsproblemen – wenig auditorische Verständnisprobleme auf beiden Seiten. Trotz merkbarer Akzente beider Probanden. Anders herum gab es ebenfalls wenig entsprechende Nachfragen, wie *Can you repeat it again, please?*. Von inhaltlicher Seite traten fast ausschließlich "zwischen" der ersten und zweiten Phase einige, wenige Verständnisprobleme auf, was durch Fragen wie *Did you want to say that...?* oder *What do you mean by...?* deutlich wurde. Hierbei ging es fast ausschließlich um das Verständnis von Details des alten Systems.

In der zweiten Phase nannten die Probanden zunächst nur sehr zögerlich entsprechende Portierungsmöglichkeiten und dies auch nur auf direkte Nachfrage hin. Es fiel auf, dass die von ihnen vorgeschlagenenen Übertragungskonzepte nur in Grundzügen den in den vorangegangenen Iterationen vermittelten entsprachen. Es wurde in keinem Punkt eines der erweiterten Features des neuen Systems, welches die Übertragung wesentlich vereinfacht hätte vorgeschlagen, sondern fast ausschließlich Basiskonzepte genannt. Auch gab es nach wie vor einige Verständnisprobleme bei den erweiterten Konzepten und Techniken. Entsprechende Fragen waren, bis auf in einem Fall, stets zu den vom Autoren genannten Optimierungen gestellt worden.

Es liegt nahe, anzunehmen, dass durch mehr Übung mit dem neuen System mehr erweiterte Konzepte vorgestellt worden wären. Die Verständnisprobleme hinsichtlich eini-

ger neuer Verfahren und Techniken, vor allem der komplexeren, sind auf den zeitlichen Abstand zwischen erster und dritter Iteration sowie den geringen Grad, mit dem diese verinnerlicht wurden zurückzuführen. Dies hätte ebenfalls durch mehr Übung im System verhindert werden können. Eine andere Möglichkeit wäre die Hinzunahme der Dokumentation während der dritten Iteration gewesen, sodass die entsprechenden Information nachgeschlagen hätten werden können und auf diese Weise auch die Art der Anwendung davon sich besser eingeprägt hätte, da der Zusammenhang für die Probanden stärker bewußt geworden wäre.

Zudem entstand im Laufe der Iteration der Eindruck beim Autoren, dass einer der beiden Probanden ([S]) teilweise abwesend war. [S] hat vor allem im späteren Verlauf der Iteration nur sehr wenig zum Vorhaben beigetragen und insgesamt lediglich zwei Vorschläge für die Portierung von Systemteilen mitgeteilt. Daher war für den Autoren dessen Präsenz oft nicht wahrnehmbar. Eine Webcam hätte hier dabei geholfen die Präsenz zu stärken, jedoch die Übertragung wesentlich stärker belastet, sodass die Qualität des Screensharings als auch der Sprachkommunikation stark darunter gelitten hätte. Auch wäre die geringe Anzahl an Vorschlägen damit wahrscheinlich nicht signifikant gestiegen.

Am Ende der dritten Iteration merkte Proband [N] abschließend an, dass er das Code-Review als sehr hilfreich und nützlich ansah und schlug vor, dieses häufiger zu wiederholen.

3.6 Zusammenfassung

Insgesamt, so zeigen die Ergebnisse der drei Iterationen, war es wichtig und richtig, zuvor die Lernpräferenzen zu erfragen. Die Unterschiede in den Lernerfolgen zwischen den Vorversuchspersonen und den eigentlichen Probanden – vor allem in der ersten Iteration – spiegeln das wider.

Die Modelle dort halfen einem der beiden Probanden dabei, den generellen Systemaufbau zu verstehen und entsprechend die in der Beschreibung enthaltenen Informationen einzuordnen. Bei einem erneuten Einsatz von Modellen zur Weiterbildung, sollte darauf geachtet werden, dass entweder die Grundlagen bei allen Lernenden für das Lesen und Verstehen von Modellen vorliegen oder die Modelle lediglich unterstützenden, nicht notwendigen, Charakter für das Begreifen des Beschriebenen haben. Die Dokumentation der API hätte in Teilen etwas detaillierter sein und mehr Beispiele enthalten können.
Die Screencasts der zweiten Iteration machten für sie dieses zuvor gewonnene Wissen anwendbar und verfehlten somit ihren Zweck nicht. Die Vorversuchspersonen kamen hiermit deutlich besser zurecht. Optimierungen hier wären jedoch textuelle Beschreibungen / eine Transkription der im Video erklärten und beschriebenen Handlungen, sodass das Nachschlagen von Informationen einfacher und schneller werden kann. Diese Tutorials sollten ebenfalls im Zusammenspiel mit (zumindest) der API-Dokumentation betrachtet werden, da diese bei einer entsprechenden Umsetzung von eigenen Inhalten mithilfe des Tutoriums Frustration vermeiden helfen, da wichtige Zusatzinformationen, die nicht in den Tutorials enthalten sind, direkt nachgeschlagen werden können.
Das Code-Review mit Portierungsansätzen hat die erwarteten Resultate nicht im vollen Umfang geliefert – dazu hätte es entsprechend mehr Übungszeit in der zweiten Iteration geben müssen. Das zuvor gesammelte Wissen sollte also bereits zu diesem Zeitpunkt gefestigt sein, um es dem Lernenden von sich aus zu gestatten, Probleme zu erkennen und zu lösen. Es hat sich jedoch gezeigt, dass eine entsprechende Nutzung dieses Verfahrens aufgrund der zuvor vermittelten Inhalte durchaus möglich ist und die Probanden insgesamt in der Lage waren, Probleme zu erkennen und Lösungsansätze zu entwickeln.

Es wurde für alle betrachteten Kommunikationskanäle ermittelt, ob diese effektiv sind und Möglichkeiten zur weiteren Optimierung dieser vorgeschlagen. Es wurde deutlich, dass die Effizienz des Einsatzes der Kanäle stark von den Lernpräferenzen der Lernenden abhängen. Durch ent-

sprechende zeitliche Anordnung der Anwendung dieser Kommunikationswege, kann hier auch die Effizienz optimiert werden. Das bedeutet, dass, um Wissen besser zu transferieren, zunächst das bevorzugte Lernmuster der einzelnen Teilnehmer ermittelt werden sollte. Ist jemand *abstract reflective*, so sind hier zunächst theoretische Grundlagen zu legen, bevor mit praktischer Anwendung fortgesetzt werden kann. Bei *concrete actives*, sollte stets der Bezug zur Anwendung, zum Beispiel in Form von (realistischen) Beispielen, vorhanden sein. Nach und nach können hier dann die theoretischen Grundlagen hinzugefügt werden. Sind die Grundlagen für beide Gruppen gelegt und die entsprechenden neuen Fähigkeiten gefestigt, so kann der letzte Schritt zur selbstständigen Arbeit durchgeführt werden: das Training zur Problemerkennung.
Die Geschwindigkeit des Wissenstransfers der einzelnen Kanäle variierte. Während die erste Iteration sehr schnell dazu geführt hat, dass Konzepte erkannt und erlernt wurden, waren diese durch den hohen Grad an Abstraktion wenig leicht anwendbar aber blieben dennoch größtenteils gut in Erinnerung. Die Besonderheiten der Entwicklung, und damit der Anwendung der Konzepte, wie sie in der zweiten Iteration vorgestellt wurde war hingegen verhältnismäßig flüchtig – das Wissen war innerhalb der dritten Iteration nur noch sporadisch präsent.

Insgesamt sähe damit ein entsprechend neuer, angepasster Weiterbildungsprozess im Unternehmen [K] so aus, dass zunächst geprürft werden muss, ob von allen Lernenden die Unterlagen vorliegen (also Skill-Listen und Fragebögen). Sofern dies der Fall ist, werden die Lernenden in zwei Gruppen geteilt: eine Gruppe bestehend aus *abstract reflectives* und die andere aus *concrete actives*. Die ersten zwei Iterationen werden dann geplant, angefertigt und den jeweiligen Gruppen bereitsgestellt (die erste Iteration, bestehend aus theoretischen Grundlagen der *abstract reflective*-Gruppe und die zweite Iteration, bestehend aus beispielsgetriebenen Tutorials der anderen Gruppe). Sobald diese fertig sind, werden die Materialien zwischen den Gruppen "getauscht". So werden für die eine Gruppe die theoretischen Grundlagen die zweite Iteration, während es für die andere die Tutorials die zweite Iteration wird.

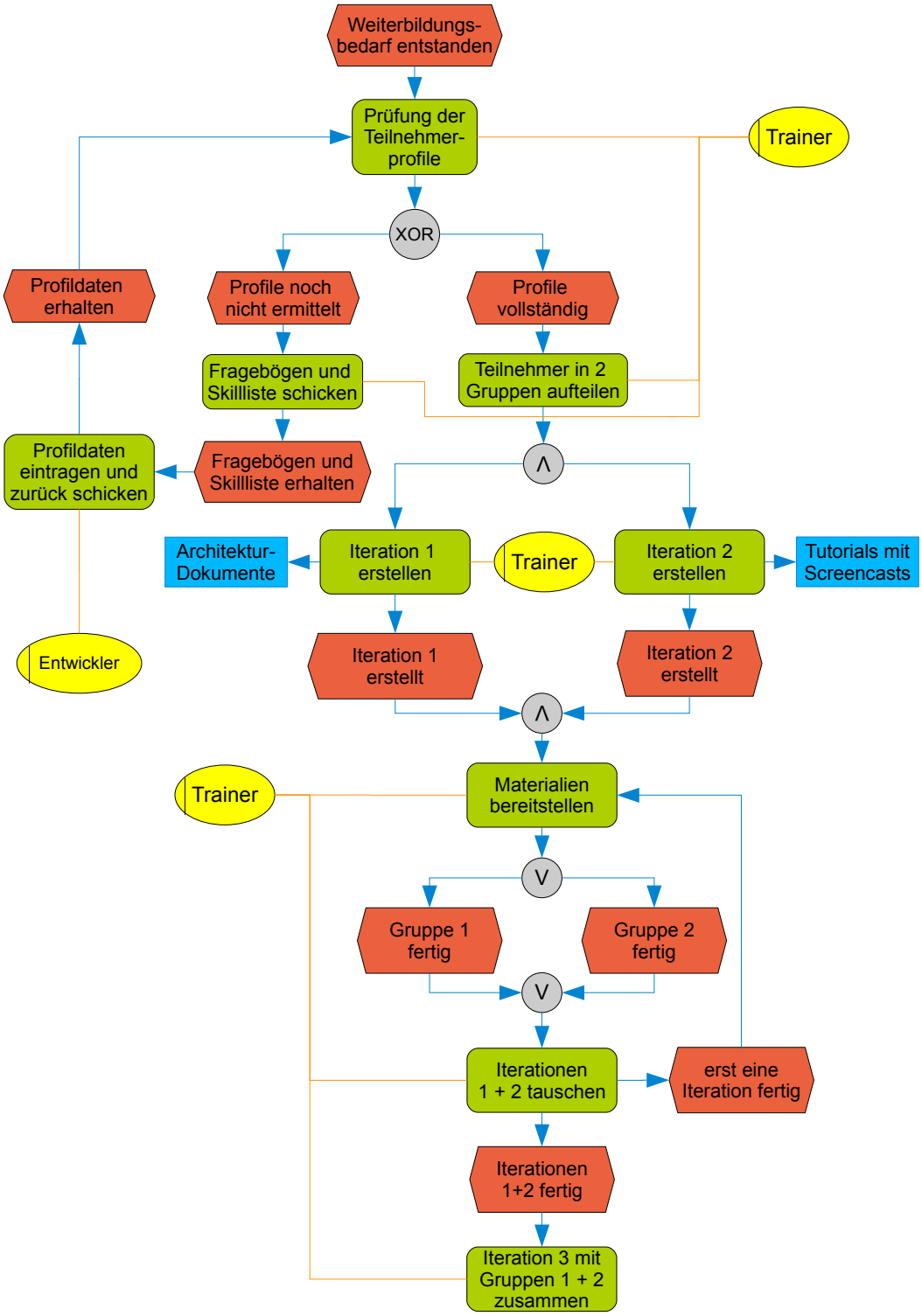

Abbildung 3.1: Angepasstes EPK-Modell zur Weiterbildung

Kapitel 4

Zusammenfassung und Ausblick

Im Folgenden wird das Ergebnis der Arbeit als Ganzes zusammengefasst und die daraus entstehenden Schlussfolgerungen vorgestellt. Was weitere interessante Forschungsbereiche zu dem Thema sind und was man dort in Zukunft an Forschungsarbeit leisten könnte, wird im Anschluss daran kurz diskutiert.

4.1 Zusammenfassung und Schlussfolgerungen

Die Nutzung eines Wissensmanagementsystems (z.B. das vorhandene Trac) sollte innerhalb des Unternehmens weiter forciert und durch das Management viel stärker eine Wissensteilungsmentalität etabliert und gefördert werden, sodass auch tatsächlich entsprechende Informationen in dieses System eingepflegt werden. Dazu sollten von den Mitarbeitern zunächst die Persönlichkeitsprofile extrapoliert und analysiert werden, sodass sinnvolle operationelle Maßnahmen dazu für die jeweiligen Mitarbeiter ausgewählt und durchgeführt werden können. Möglichkeiten, im verteilten Team ein Gruppengefühl zu verstärken, kann beispielsweise eine entsprechende eMail-Signatur anstelle

von Plakaten an den Wänden, wie sie bei kolozierten Unternehmen benutzt werden, sein, welche vom Management eingesetzt wird. Teil davon muss jedoch auch sein, den Aufbau bzw. die Vertiefung des gegenseitigen Vertrauens aktiv voran zu treiben. Das kann durch Konzepte wie dem Einsatz von Kurznachrichten via Twitter oder ähnlichen Technologien zur Förderung des *ambient trusts* geschehen.
Da das Pair Programming und hier insbesondere das Distributed Pair Programming wesentliche Unterstützung zum Wissenstransfer ist, muss auch dieses, zum Beispiel in Form von regelmäßigen, geplanten Distributed Pair Programming Sitzungen als Teil des Entwicklungsprozesses etabliert werden. Hier muss vor allem darauf geachtet werden, dass geeignete Werkzeuge für die verteilte Entwicklung selbst, aber auch für Erklärungen bereitgestellt und benutzt werden. Eine entsprechende Infrastruktur ist bereits teilweise vorhanden (zum Beispiel für die Kommunikation), sodass auf dieser aufbauend, der Prozess weiter verbessert werden kann.

In der Versuchsreihe, bestehend aus drei Iterationen, wurde eine Exploration der Möglichkeiten für den Wissenstransfer und dessen Geschwindigkeit mithilfe einiger Probanden aus dem Unternehmen durchgeführt. Es wurden dafür drei unterschiedliche Medien bzw. Kommunikationskanäle genutzt, um zu untersuchen, welches den größten Effekt hat. Ergebnis dieser Iterationen war, dass alle zusammen eine effektive und stabile Weiterbildung ermöglichen. Zusätzlich sollte, nachdem die Grundkenntnisse, wie in den ersten zwei Iterationen beschrieben, vorhanden sind, auch das (Distributed) Pair Programming verstärkt eingesetzt werden, wenn neue Mitarbeiter eingelernt werden müssen. Effektiv kann das für sehr viele neue Mitarbeiter so funktionieren, dass wichtige Aspekte des Extreme Programmings, so zum Beispiel die Rotation der Pair Programming Partner mit in den Einarbeitungsprozess übernommen werden. Zur Minimierung des Organisationsaufwandes sollte hier eine entsprechende automatische oder zumindest semi-automatische Zuteilung der Teampartner statt finden.

Die im Rahmen dieser Arbeit erarbeiteten Optimierungsmöglichkeiten, sowohl zum Wissenstransfer allgemein als auch zur Weiterbildung insbesondere, werden im Unternehmen in den nächsten Monaten eingeführt und umgesetzt werden.

4.2 Ausblick

Themen, die an dieser Stelle noch weiter erforscht werden sollten, sind beispielsweise die Effektivität der hier vorgeschlagenen Optimierungsmöglichkeiten zu evaluieren. Aber auch Vergleiche, wie Weiterbildungsmaßnahmen in kleinen Gruppen, wie sie in dieser Arbeit beschrieben wurden, im Gegensatz zu großen Gruppen funktionieren können bzw. welche Änderungen am Weiterbildungsprozess dann besonders wichtig wären. Beispielsweise könnte sich bei großen Gruppen die Ermittlung und Analyse der Lernpräferenzen der Teilnehmer als sehr zeitaufwändig darstellen und würde entsprechend die Weiterbildung selbst verzögern. Fragen, die hier von Interesse sein könnten, wären beispielsweise, inwiefern durch Reduktion der Kontrolle, welche durch das Mehr an Fortbildungsteilnehmern dann ohne Zweifel auftreten würde, trotzdem eine effektive Weiterbildung etabliert werden könnte.
Eine andere Forschungsfrage betrifft die Struktur von global verteilten Unternehmen. Inwiefern kann ein großes Maß an Verteilung trotz vieler Mitarbeiter durch ein Minimum an Hierarchie beibehalten werden?

Die Relevanz einiger dieser Fragen, wenn nicht sogar aller, wird im Unternehmen [K] in jedem Fall mit der Zeit steigen. Durch die aktuellen Trends hin zu global verteilten Unternehmen sind diese Fragen jedoch auch von größter Wichtigkeit für andere (oft junge) Firmen. Die Zukunft wird zeigen, auf welche Weise und mit welchen Mitteln mit solchen Problemen umgegangen werden kann.

Anhang A

INTERAKTIONSPROTOKOLLE

A.1 Iteration 1

A.1.1 Anweisungs-eMail

Hello $[N]$.

Here is the set of documents for the first training iteration. It's an architecture overview and then the API documentation. While the first one is rather short, it shows the general components and explains how everything works together. There is not much new about this, but a few things are, though. One small hint before: at the very end of the first document you'll find a description of the modeling language I chose.

When you're done with both documents (you may skip parts of both documents if you find them too uninteresting), tell me. I will then ask you a few general questions about this (these mainly focus on how much the concepts were understandable).

Please also forward these documents to $[S]$, so that he can also read through them. Also he should tell me as soon as he is done with reading them, so that I can ask him too. He can also skype me, directly (same for you, of course).

[...]

Regards,
Robert

A.1.2 Feedback-eMail 1

Hello Robert,

Today we have done with the Review of the Document.

Here the feedback of the Review.

Overall the Architecture is really simple to understand and easier to build complex applications rapidly with minimum efforts.

1.[Q] IDE : Where can I get this IDE?. Is the [Q] Name given by us or already existing?

2.The Agents (Request,Generator,Module, DB Manager Section etc) explains clearly the role they play in the system. In earlier framework we have been using same function (redundant) in 2 modules to avoid (memory consumption) parent inheritance module reference. In the New framework how the Module going to interact between them?

3.Some of the features are really admiring which makes the readability of template file even better..
I mean TagLibs. But Its better if we get an example of defining the custom Tags with detailed explanation. (Page 15 of API Doc)

4.SelectQuery : This feature is also remarkable one. Is it possible to have Limits and Group By Clause here?

5.checkFields : This is meant for only NULL checks or Email Validation, Zip Code Validation etc.
FYI : We really need a utility module to handle all the Input Validations either Client Side or Server Side.

6.Javascript Library :
Third Party Library Info : What we going to use. (Prototype)
Custom Library Info and its Usages (Usage of FlashUpload, Java Upload etc..)

7.As discussed earlier we need a separate utility class to handle all session variables in a more sophisticated way.

8.In which scenario simpleCheckSum will be useful?

I understand this framework is designed not only for $[onesingle]$ Application, this is meant for all future Web Based Application which makes the Development Process easier and easily customizable.

[...]

Regards
$[N]$

A.1.3 Skype-Interview mit [N]

Robert Kunze: Hello [N].
$[N]$: Hope you got My Email..
Robert Kunze: Good afternoon.
Robert Kunze: Yes, I got it.
$[N]$: Good..
[...]
Robert Kunze: Okay. So, some questions to how you read the documents. 'Just to see if structure and everything can still be optimized.
Robert Kunze: What did you read first?
Robert Kunze: Did you have the feeling that the models themselves did have an impact on understanding the structure (positive or negative)?
$[N]$: We reviewed the doc individually and had a Discussion wih the Overall Review.
$[N]$: Initially it doesn't seems to be new for us..
$[N]$: As we already have some of the features in the existing framework

Robert Kunze: And did you first read the architecture overview and then the API or some more mixed?
[N]: I started with Architecture and Proceeded with API
Robert Kunze: Okay.
[N]: Initially we had tough time to understand the runlet
Robert Kunze: 'Yes. That's somehow a new concept - but only partly.
[N]: After reviewing subsequent topics, it made clear..
Robert Kunze: Okay. So how would you describe the process of evaluating/rendering runlets?
[N]: The Current Process is ok.
Robert Kunze: 'I meant actually the process from calling an event until the final page is sent to the user.
Robert Kunze: You don't need to describe everything in detail.
Robert Kunze: Just what gets evaluated when.
[N]: Is it Possible to specify in the handler to avoid caching in the runlet even if caching is enabled
Robert Kunze: Caching is always event-wise. So a handler can activate it and deactivate it as it wishes.
Robert Kunze: That's a good point! I forgot to mention this...
[N]: Anothery Query
Robert Kunze: Yes?
[N]: After re-evaluating runlet, does this again stored in cache?
Robert Kunze: Runlets are evaluated at runtime. And after this point, the result of it is not stored again in a cache, but directly sent to the user.
[N]: ok.
Robert Kunze: What would you say: did the architecture overview itself give you any new insight? And what about the models? Were these understandable and did they have an impact on understanding?
[N]: On Understanding Level this Document is more than enough for us.
[N]: But for a New User/Developer, still it requires some real time examples
Robert Kunze: Okay. Do you think that the API documentation can fill this gap for new developers?
Robert Kunze: (at least on the theoretical level)
[...]
[N]: API Document is ok
[N]: ! thing needs to be added in the doc

Robert Kunze: which thing?
[*N*]: How the index.php main page handles the reuqest
[*N*]: including friendly URLS etc
[*N*]: and how its delegating accordingly
Robert Kunze: what do you mean by "friendly URLs"?
[*N*]: to the respective events
[*N*]: i mean /Articles/1234
Robert Kunze: Okay, thanks.
[*N*]: index.php?event=articles:show& id=1234
Robert Kunze: Aah, okay.
Robert Kunze: Yes. That's actually not part of the framework, but of the environment it runs in. But better mention it, that's right.
Robert Kunze: Anything in the API documentation that you felt was missing or described in too few detail?
[*N*]: As specified in the Email
Robert Kunze: Okay. great.
Robert Kunze: Good. Then that's all of my questions.
Robert Kunze: Thanks for the answers and insights.

A.1.4 eMail-Interview

Hello [*S*].

As already for [*N*], I have some questions on how you read the documents. Just to see if (and in what aspects) structure and everything can still be optimized. What did you read first?

Did you have the feeling that the models themselves did have an impact on understanding the structure (positive or negative)?

Anything in the API documentation that you felt was missing or described in too few detail (except the things already said in the eMail)?

Regards,
Robert

A.1.5 eMail-Interview-Antworten

Hi Robert,

I just reviewed this email. Sorry for the delayed response.

Actually [N] & Me both of them reviewed those documents and had a discussion and thus finalized the feedback.

Here the answers for your queries.

1.Initially i just traversed both the docs in a random fashion.

2.Regarding the framework structure it seems it can addresses any type of complex scenarios.

3.Since I have been working in Cake framework earlier, I feel still we might need some optimization & Organizing compare to Cake

4.The Models (FMC) and Structure of Framework currently seems to be independent. Its better if the Document explains the Framework much more detail in conjunction with the specified model

Few days back i tried the [Q] IDE. It seems it was done in ExtJs. The IDE definitely a right tool for avoiding the data/environment based issues. But as a developer I feel it needs some enhancement at editor level. I mean Full Editor View, which can give full window space to review large set of codes and logics without scrolling back and forth.

Hope this feedback might helped you..

Thanks
[S]

A.2 Iteration 2

A.2.1 Anweisungs-eMail

Here now the instructions for the new iteration. First of all: you'll find the Screencasts at the following page:

; [Site URL]

The IDE, which is a web-based IDE, is explained within these videos by example. Your task will be first to watch these videos and then doing some practical work, by creating a small blog-like application. For this, the requirements are: create a new table, to hold the blog posts. This should contain fields for identifying the user, a title and a content field. Then there should be an interface, the user can create a blog post (enters his name, the blog's title and it's content - latter one can be a simple textarea - no need for WYSIWYG editors here and finally a button to publish it) and view what has been published before (no pagination, just a list of all the blog posts - directly below the create blog form). It does not need to look any nice, but is just to see if the main concepts of the IDE are understood.

When you are done with it, write me. Also write me, if you experience problems with the IDE itself.

A.2.2 eMail-Interview

Thanks for the information. I just had a look at the mini blog. Looks really good. Seems you got along with it pretty well, so far.

Here some questions to you and [S] concerning the learning method itself:

1) What was your impression of the screencasts?
2) Were they understandable?
3) How often did you watch each screencast?

4) Did you jump/fast-forward/backward within the movies?
5) Did you notice any difficulties in transforming what you heard in the screencasts and then doing it yourself?
6) Do you felt the screencasts like being too long or too short?
7) Did it explain everything you needed to begin or was something missing?

A.2.3 eMail-Interview-Antworten von [N]

Here the Summary of the feedback of the Web IDE

1) What was your impression of the screencasts?
Initially we amazed ([M], [S] and Me) at the concept of the Online development.

2) Were they understandable?: Yes

3) How often did you watch each screencast?
3 – 4 times. Sometimes it hanged (might be my machine problem)

4) Did you jump/fast-forward/backward within the movies?
Yes. (To ignore the Initial steps again and again)

5) Did you notice any difficulties in transforming what you heard in the screencasts and then doing it yourself?
The Language is very clear. (Voice). Not tough as US persons

6) Do you felt the screencasts like being too long or too short?
Normal

7) Did it explain everything you needed to begin or was something missing?
See below

Feedback :

1.The IDE expects more usage of mouse. Need options to maximum utilize the keyboard shortcuts to navigate ,run,switch between panels etc
2.Currently we don't have a SQL Editor to frame and execute queries.
3.When running, the errors are seems to be different line No. We understand these errors are coming from compiled PHP page. Can we have a option to view the compiled page so that we can exactly identify the error.?

Anhang B

SONSTIGE DOKUMENTE

B.1 Fragebögen

B.1.1 Vorversuchspersonen

Fragebogen [I]

Questions:

1.Do you sometimes browse the web/read magazines/attend workshops just to see what new technologies exist/emerge?
[−] yes [X] no

2.What do you prefer?
[X] learning by doing [−] first theoretical background, then practical work

3.How often do you normally learn using new technologies?
[−] weekly [−] monthly [X] quarterly [−] once a year [−] once a decade

4.What kind of education institutions did you attend?
.[Fachhochschule Brandenburg].

5.How long have you been working in a non-governmental company in the IT area?
[0] months/years

6.How would your friends characterize you?
[−] often rather quiet [−] normal [X] rather talkative [−] party booster

7.Have you ever lived for at least one year out of your home country?
[−] yes [X] no

Fragebogen [T]

Questions:

1.Do you sometimes browse the web/read magazines/attend workshops just to see what new technologies exist/emerge?
[X] yes [−] no

2.What do you prefer?
[X] learning by doing [−] first theoretical background, then practical work

3.How often do you normally learn using new technologies?
[−] weekly [−] monthly [X] quarterly [−] once a year [−] once a decade

4.What kind of education institutions did you attend?
.[Freie Universität Berlin].

5.How long have you been working in a non-governmental company in the IT area?
.[3 years]. months/years

6.How would your friends characterize you?
[−] often rather quiet [X] normal [−] rather talkative [−] party booster

7.Have you ever lived for at least one year out of your home country?
[−] yes [X] no

B.1.2 Fragebogen [N]

Questions:

1.Do you sometimes browse the web/read magazines/attend workshops just to see what new technologies exist/emerge?
[Y] yes [−] no

2.What do you prefer?
[−] learning by doing [Y] first theoretical background, then practical work

3.How often do you normally learn using new technologies?
[−] weekly [Y] monthly [−] quarterly [−] once a year [−] once a decade

4.What kind of education institutions did you attend?
.[NIIT].

5.How long have you been working in a non-governmental company in the IT area?
.[4 Years 3 Months]. months/years

6.How would your friends characterize you?
$[-]$ often rather quiet $[NORMAL]$ normal $[-]$ rather talkative $[-]$ party booster

7.Have you ever lived for at least one year out of your home country?
$[-]$ yes $[N]$ no

B.1.3 Fragebogen [S]

Questions:

1.Do you sometimes browse the web/read magazines/attend workshops just to see what new technologies exist/emerge?
$[Y]$ yes $[-]$ no

2.What do you prefer?
$[-]$ learning by doing $[Y]$ first theoretical background, then practical work

3.How often do you normally learn using new technologies?
$[Y]$ weekly $[-]$ monthly $[-]$ quarterly $[-]$ once a year $[-]$ once a decade

4.What kind of education institutions did you attend?
.[Engineering College].

5.How long have you been working in a non-governmental company in the IT area?
.[4 Years]. months/years

6.How would your friends characterize you?
[−] often rather quiet [−] normal [−] rather talkative [$PARTY\,BOOSTER$] party booster

7.Have you ever lived for at least one year out of your home country?
[−] yes [N] no

B.2 Skill Matrix

	[N]	[S]	[I]	[T]
Languages				
English	11	11	31	22
German	55	55	11	11
French	55	55	24	23
Hindi	23	55	55	55
Tamil	11	11	55	55
Programming Languages				
Java	32	34	44	12
PHP	22	21	55	12
Typoscript	55	55	55	55
C++/C	12	33	34	23
C#	22	22	55	22
Scheme	55	55	55	55
Python	55	55	55	44
Bash	33	55	44	33
Perl	55	55	55	33
Opal	55	55	55	32
Object Teams	55	55	55	55
Ruby	55	55	55	55
XSL	43	22	55	23
XSLT	43	22	55	23
Scala	55	55	55	55
Erlang	55	55	55	55
Prova	55	55	55	55
Prolog	55	55	55	44
Visual Basic	22	55	55	33
Assembler	55	55	55	34

Groovy	55	55	55	55
Development Process				
Bugtracking	22	22	44	22
SVN	22	22	55	23
Unit Testing (JUnit, JSUnit)	55	55	55	44
Code Documentation	22	23	33	22
Trac	32	44	55	55
Firebug	32	22	44	22
Xtreme Programming	55	55	55	55
V-Model XT	55	55	55	55
Waterfall model	23	34	55	22
Scrum	55	55	55	55
Crystal Clear	55	55	55	55
CMMI	23	55	55	55
Rational ClearCase	55	55	55	55
Internet				
XHTML / HTML	22	22	23	11
CSS	32	22	43	12
Javascript	32	21	33	12
AJAX	22	22	43	22
DOM	32	22	55	33
XML	32	22	55	23
JSON	22	22	55	43
Web-Service Integration	21	23	55	33
TCP/IP	33	44	44	23
SOAP	34	44	55	55
FTP	22	22	55	23
RPC / RMI	34	55	55	55
REST	32	22	55	55
CORBA	55	55	55	55
Flash	54	43	55	55
ActionScript	54	43	55	55
RDF + RDF Schema	55	55	55	55
OWL	55	55	55	55
RuleML + RIF	55	55	55	55
Haxe	55	55	55	55
Modeling				
ER - Database	12	22	45	22
Object-Oriented	12	22	55	22
XML DTD/Schema	12	23	55	23
UML	12	55	44	33
Petri Net	55	55	54	55
FMC Block Diagram	55	55	55	55

BPMN	55	55	55	55
EPC	55	55	55	55
Executable UML	55	55	55	55
GUI	12	22	43	23
Frameworks				
ExtJS	22	22	33	44
Prototype JS	32	42	44	44
jQuery	54	33	55	22
Script.aculo.us	54	55	55	44
Eclipse Modeling Framework	55	55	55	44
Selenium	55	55	55	55
IDEs				
Eclipse	44	43	43	13
MS Visual Studio	12	22	55	23
NetBeans IDE	55	22	55	55
Vim	55	55	54	33
XCode	32	55	32	55
Visual Slick Edit	55	55	55	55
Magic Draw	55	55	55	55
IBM Rational Software Architect	55	55	55	55
Zend Studio	33	44	55	44
Database Systems				
SQL	12	22	33	12
MySQL	22	22	34	12
PostgreSQL	55	55	55	55
Oracle	54	55	55	55
DB2	55	55	55	55
TSQL Sybase ASE 12.5	55	55	55	55
SPARQL	55	55	55	55
SchemaSQL	55	55	55	55
Operating Systems				
Windows	11	22	21	12
Linux	54	23	33	23
Mac OS X	33	44	22	44
Content Management Systems				
TYPO3	55	55	55	44
Imperia	55	55	55	55
Communiqué	55	55	55	55
MediaWiki	55	55	55	55
DokuWiki	55	55	55	55
phpBB	32	55	55	44
Moodle	55	55	44	55
Ilias	55	55	55	55

WordPress	32	55	55	32
Applications				
Microsoft Office	12	22	22	12
OpenOffice.org	55	22	22	22
GIMP	33	55	43	23
Latex	55	55	44	32
Adobe Photoshop	33	32	44	44
Paint.NET	55	55	55	55
Inkscape	33	55	55	44
	$[N]$	$[S]$	$[I]$	$[T]$

Literaturverzeichnis

Gary Abramson. IT, Phone Home. *CIO Enterprise Magazine*, Feb. 15:http://web.archive.org/web/20020125105703/http://www.cio.com/archive/enterprise/021599_afar.html, 1999.

Prashant Baheti, Edward Gehringer, and David Stotts. Exploring the efficacy of distributed pair programming. *Extreme Programming and Agile Methods - XP/Agile Universe 2002*, vol. 2418:pp. 208 – 220, 2002.

Jagdish Bhagwati. *In Defense of Globalization*. Oxford University Press, 2004.

F. Blackler. Knowledge, knowledge work and organizations: An overview and interpretation. *Organization Studies*, vol. 16:pp. 1021 – 1046, 1995.

BNET. Business definition for: Network organization. http://dictionary.bnet.com/definition/network+organization.html, zuletzt überprüft: Jan. 2010.

Kristin Braa, Ole Hanseth, Claudio U. Ciborra, Antonio Cordella, Bo Dahlbom, Angelo Failla, Vidar Hepsø, Jan Ljungberg, and Eric Monteiro. *From control to drift: the dynamics of corporate information infrastructures*. Oxford University Press, 2000.

Gerardo Canfora, Aniello Cimitile, and Corrado Aaron Visaggio. Lessons learned about distributed pair programming: What are the knowledge needs to address? *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE03:pp. 314–319, 2003.

E. Carmel. *Global Software Teams: Collaboration across Borders and Time Zones*. Prentice Hall, 1999.

Betsy Carroll. Fostering ambient awareness in virtual teams. http://www.leadingvirtually.com/?p=90, 2008, zuletzt überprüft: Feb. 2010.

Alistair Cockburn and Laurie Williams. The costs and benefits of pair programming. *Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering*, XP2000, 2000.

Allan Collins, John Seely Brown, and Ann Holum. Cognitive apprenticeship: making thinking visible. *American Educator*, vol. 6:pp. 38 – 46, 1991.

W. H. Davidow and W. S. Malone. *The virtual corporation*. Harper Collins Publishers, 1992.

Elizabeth Davidson and Albert S. M. Tay. Studying teamwork in global it support. In *HICSS*, page 19, 2003.

D. De Cremer and M. Van Vugt. Social identification effects in social dilemmas: A transformation of motives. *European Journal of Social Psychology*, vol. 29:pp. 871 – 893, 1999.

Deborah DeZure. *Learning from Change: landmarks in teaching and learning in higher education from Change magazine, 1969 – 1999*. Stylus Publishing, LLC, 2000.

W. B. Gudykunst, Y. Matsumoto, S. Ting-Toomey, T. Nishida, K. W. Linda, and S. Heyman. The influence of cultural individualism-collectivism, self construals, and individual values of communication styles across cultures. *Human Communication Research*, vol. 22 (4):pp. 510 – 543, 1996.

C. Handy. Trust and the virtual organization. *Harvard Business Review*, vol. 73 (3):pp. 40 – 50, 1995.

Brian F. Hanks. Distributed pair programming: An empirical study. *Lecture Notes in Computer Science*, vol. 3134:ch. 19, 2004.

Eli Hustad. Knowledge networking in global organizations: The transfer of knowledge. *Proceedings of the 2004 SIGMIS conference on Computer personnel research: Careers,*

culture, and ethics in a networked environment, April 22 – 24: 55 – 64, 2004.

Chyng-Yang Jang. Facilitating trust in virtual teams: The role of awareness. *Competition Forum*, vol. 7 (2):pp. 399 – 407, 2009.

Sirkka L Jarvenpaa and Dorothy E. Leidner. Communication and trust in global virtual teams. *JCMC*, vol. 3 (4):http://jcmc.indiana.edu/vol3/issue4/jarvenpaa.html, June 1998.

Niklas Luhmann. *Vertrauen: ein Mechanismus der Reduktion sozialer Komplexität*. Lucius und Lucius, 2000.

Peter Marks, Peter Polak, Scott McCoy, and Dennis Galletta. Sharing knowledge: How managerial prompting, group identification, and social value orientation affect knowledge-sharing behaviour. *Communications of the ACM*, vol. 51 (2):pp. 60 – 65, February 2008.

Steve McConnell. *Code Complete: A Practical Handbook of Software Construction, 2nd ed.* Microsoft Press, 2004.

A. Mowshowitz. Virtual organization. *Communications of the ACM*, vol. 40 (9):pp. 30 – 37, 1997.

M. O'Hara-Devereaux and R. Johansen. *Global work: Bridging distance, culture, and time*. Jossey-Bass Publishers, 1994.

J. Petersen. Bildschirmarbeitsplätze – eine arbeitsmedizinische Bewertung. *Deutsches Ärzteblatt*, vol. 103(30):pp. A 2047 – 52, 2006.

G. Salomon. *Distributed Cognitions: Psychological and educational considerations. Learning in doing: Social, cognitive, and computational perspectives*. Cambridge University Press, 1993.

David Stotts, Laurie Williams, Nachiappan Nagappan, Preshant Baheti, Dennis Jen, and Anne Jackson. Virtual teaming: Experiments and experiences with distributed pair programming. *Extreme Programming and Agile Methods - XP/Agile Universe 2003*, vol. 2753:pp. 129–141, 2003.

Laurie Williams. *Pair Programming Illumniated*. Addison Wesley, 2003.

Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. Strengthening the case for pair programming. *IEEE Software*, vol. 17, no. 4:pp. 19 – 25, July/Aug. 2000.

Stichwortverzeichnis

Briefing ist eine Einsatzbesprechung mit detaillierten Aufgabenbeschreibungen und enthält essentielle Informationen für deren Bearbeitung. 16

Business Process Modeling Notation (BPMN) ist eine graphische Repräsentation, um Geschäftsprozesse in einem Workflow zu spezifizieren. 39

Code-Review ist eine systematische Untersuchung von Computer Source Code, meist dazu verwendet um Fehler, die in der initialen Entwicklungsphase übersehen wurden zu finden und zu beheben. 35, 51, 52, 55, 56

Ereignisgesteuerte Prozessketten (EPK) sind Diagramme, die eine leichtverständliche Darstellung der Abhängigkeiten in einem Funktionsfluß bieten. 15

Fundamental Modeling Concepts (FMC) stellen ein Framework zur Kommunikation über komplexe Softwaresysteme dar. Unterschiedliche Modellarten werden für unterschiedliche Sichtweisen auf das zu beschreibende System benutzt. 39

Integrated Development Environment (IDE) ist eine Softwareanwendung, die umfangreiche Funktionen für Softwareentwickler bereitstellt. Sie enthält üblicherweise einen Quellcode-Editor, einen Compiler oder Interpreter, einen Debugger und automatische Buildtools. 29, 45, 47

Mockup ist eine Art Benutzerinterface-Modell in Originalgröße, das zum Erklären von Funktionsbausteinen, Bewerten von Design-Entscheidungen und für Briefings benutzt wird. 16

Screencast ist eine digitale Videoaufnahme des Computerbildschirminhalts, oft begleitet durch eine auditive Beschreibung. 34, 35

Ticketing-System ist eine Computer-Software, die Tickets (eine Arbeitseinheit zur Verbesserung des Systems) verwaltet und organisiert. Die enthaltenen Daten können Fehler, neue Features, einfache Aufgaben, fehlende Dokumentation und so weiter sein. 14, 16

Unified Modeling Language (UML) ist eine standardisierte Modellierungssprache für Softwareentwicklung. Es enthält eine Reihe von Notationen für unterschiedliche Zwecke und ist in der Regel sehr nah an der eigentlichen Implementierung ausgerichtet. 39, 43

Virtual Network Computing (VNC) ist ein graphisches Screensharing-System, welches es erlaubt, einen anderen, entfernen Computer zu steuern und dessen Bildschirminhalt zu sehen. VNC benötigt mindestens zwei Komponenten: einen Server und einen Viewer. 30, 51, 52

Voice over Internet Protocol (VoIP) ist ein genereller Ausdruck für eine Reihe von Übertragungstechnologien, bei denen die Sprachkommunikation über IP Netzwerke, wie das Internet statt findet. VoIP wird manchmal auch Internet Telefonie genannt. 30

Typeset 22. Februar 2010

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Berlin, February, 2010
ROBERT KUNZE