T U M

I N S T I T U T F Ü R I N F O R M A T I K

Quantitative Untersuchung des Extreme Programming Prozesses

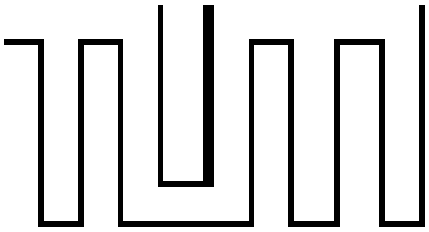
Bernhard Rumpe Astrid Schröder

TUM-I0110
Dezember 01

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-12-I0110-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

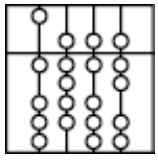


Technische Universität München
Fakultät für Informatik

Technischer Bericht

Quantitative Untersuchung des Extreme Programming Prozesses

Bernhard Rumpe
Astrid Schröder



Inhaltsverzeichnis

Zusammenfassung

Die Suche nach einer optimalen Vorgehensweise bei der Softwareentwicklung bringt gelegentlich neue Vorschläge für Vorgehensmodelle hervor. Extreme Programming ist eine radikale, aus der Praxis motivierte Variante, die versucht „best practices“ der Software-Entwicklung in möglichst schlanker Weise zu kombinieren und soweit wie möglich auf organisatorischen Ballast zu verzichten. Aufgrund der hohen Praxisrelevanz von XP scheint eine wissenschaftliche und kritische Aufarbeitung der Thematik geboten. Dieser Beitrag exploriert einige der Themenfelder von XP und stellt somit einen Baustein dieser Aufarbeitung dar. Dieser technische Bericht gibt zunächst eine Übersicht über die Extreme Programming-Vorgehensweise und beschreibt dann die Ergebnisse einer weltweiten Umfrage zu durchgeführten XP-Projekten. Dieser technische Bericht stellt damit einen ersten Baustein für eine objektive Aufarbeitung von Extreme Programming und verwandten Vorgehensmodellen dar.

1 Der Ursprung von Extreme Programming

Extreme Programming (in Kurzform: XP) ist eine sogenannte „leichgewichtige“ Softwareentwicklungs-Methode. Ihre Entwicklung und Verbreitung wird zumindest in der Anfangsphase von XP den drei Softwareentwicklungs-Gurus Kent Beck, Ron Jeffries und Ward Cunnigham zugeschrieben. Obwohl XP als Vorgehensweise u.a. bei Softwareprojekten einer Schweizer Bank definiert und verfeinert wurde, zeigt schon die Namensgebung eine starke Beeinflussung durch die nordamerikanische Softwareentwicklungs-Kultur. Denn trotz des gewählten Namens ist XP keineswegs nur eine Hacker-Technik, sondern besitzt einige sehr detailliert ausgearbeitete methodische Aspekte. Diese erlauben es ihren Befürwortern zu postulieren, dass durch XP mit verhältnismäßig wenig Aufwand qualitativ hochwertige Software unter Einhaltung der Budgets zur Zufriedenstellung der Kunden erstellt werden kann.

XP erfreut sich gerade in der Praxis einer stetig steigenden Popularität. Zahlreiche Bücher sind erschienen bzw. in Vorbereitung. Darunter sind [Bec99, JAH00, BF01] die teilweise unterschiedliche Aspekte der Thematik sowie jeweils aktuelle Wissensstände der sich noch stark in Weiterentwicklung befindlichen Methodik detailliert betrachten. Die Popularität von XP in der Praxis zeigt sich auch anhand zahlreicher Artikel in Computerzeitschriften wie [Weg99, Eck00, EH00, EH01], die einen oft lesenswerten Überblick über den jeweils aktuellen Stand von XP oder einen einführenden Vergleich mit alternativen Ansätzen beschreiben. Aktuelle Informationen lassen sich über mehrere im Internet eingerichtete Diskussionsgruppen und Informations-Sites

zum Thema XP erfahren. Diese Informationen sind allerdings nicht notwendigerweise konsistent und nur wenig aufbereitet [Wel01, Cun01, Jef01].

Obwohl derzeit nur wenige genaue Zahlen über XP-Projekte vorliegen, kann davon ausgegangen werden, dass XP heute vielen Softwareentwicklern, zumindest im nordamerikanischen und europäischen Raum, ein Begriff ist. Eine Reihe von Consulting-Firmen beginnen ihre Mitarbeiter und Kunden für zukünftige XP-Projekte vorzubereiten. Firmen mit Inhouse-Software-Entwicklung starten Pilot-Projekte, wobei dort aufgrund der teilweise radikal anderen Sichtweise von XP gegenüber klassischen Vorgehensweisen eine deutliche Verunsicherung über die Implikationen besteht. Die Techniken und Konzepte von XP führen zu einer relativ starken Polarisierung der Meinungsbildung. Auf der einen Seite glauben Softwareentwickler, XP erhebe ihre bisherige, informelle Vorgehensweise (Hacking) zur Ingenieurs-Disziplin. Auf der anderen Seite wird XP verdammt, weil es alles über Bord werfe, was in den letzten Jahrzehnten an Vorgehensmethodiken so mühsam erarbeitet worden ist.

Richtig ist, dass XP eine leichtgewichtige Softwaremethode ist, die explizit als Gegengewicht zu schwergewichtigen Methoden, wie dem Rational Unified Process [Kru00] oder dem V-Modell [Ver00] positioniert ist.

In Abschnitt 2 wird ein kurzer Überblick über die wesentlichen Elemente von XP gegeben. Dabei werden zwei interessante technische Aspekte, Testen und Refactoring, genauer beleuchtet. Der Abschnitt 3 behandelt die Frage einer wissenschaftlichen Aufarbeitung von XP und des Umgangs mit XP in der Lehre und zeigt einige Ansätze auf, die XP mit traditionellen Techniken verbinden. Im vierten Abschnitt wird die quantitative Studie zu XP vorgestellt und beschrieben, wie diese Studie durchgeführt wurde. Der fünfte Abschnitt beinhaltet die Ergebnisse dieser Studie.

2 XP- Ein Überblick

XP ist nicht einfach eine Neuauflage des Hacking, sondern eine leichgewichtige aber dennoch rigorose Softwareentwicklungs-Methode. Sie verzichtet auf eine Reihe von Elementen der klassischen Softwareentwicklung, um so eine schnellere und effizientere Kodierung zu erlauben. Die dabei entstehenden Defizite versucht sie durch stärkere Gewichtung anderer Konzepte (insbesondere der Testverfahren) zu kompensieren. Auch das Leichgewicht XP besteht aus einer Reihe von Einzelkonzepten, die alle vorzustellen den Rahmen dieses Überblicks sprengen würde. Stattdessen ist auf entsprechende Literatur (siehe eine Auswahl im Literaturverzeichnis) verwiesen.

XP versucht explizit nicht auf neue und damit nicht ausreichend erprob-

te methodische Konzepte zu setzen, sondern integriert weitgehend bewährte Techniken zu einem Vorgehensmodell, das auf Wesentliches fokussiert und auf organisatorischen Ballast soweit wie möglich verzichtet. Weil der Programmcode das ultimative Ziel einer Softwareentwicklung ist, fokussiert XP von Anfang an genau auf diesen Code. Im Gegensatz dazu wird Dokumentation als aufwendiger Ballast betrachtet. Eine Dokumentation ist aufwendig zu erstellen und oft sehr viel fehlerhafter als der Code, weil sie nicht automatisiert analysier- oder testbar ist. Zusätzlich verhindert sie eine flexible Weiterentwicklung des Systems eine schnelle Reaktion auf neue oder veränderte Anforderungen der Kunden, wie es in der Praxis häufig der Fall ist. Folgerichtig wird in XP-Projekten keine Dokumentation erstellt. Im Ausgleich dafür wird Wert auf eine gute Kommentierung des Quellcodes durch Coding Standards und eine umfangreiche Test-Sammlung gelegt.

2.1 Werte, Prinzipien und Aktivitäten in XP

Die primären Ziele von XP sind altbekannt: Effiziente Entwicklung qualitativ hochwertiger Software und unter Einhaltung von Zeit- und Kostenbudgets. Welche Mittel dazu eingesetzt werden, wird anhand der *Werte*, der *Prinzipien* und der *grundlegenden Aktivitäten* kurz vorgestellt. Die vier Werte sind:

Kommunikation: Permanente und intensive Kommunikation der Entwickler untereinander, sowie mit den Kunden erlaubt schnellstmögliches Feedback sicherzustellen, unnötige Funktionalität zu verhindern, entstehende Probleme so schnell wie möglich zu lösen und das Problem der fehlenden Dokumentation zu mildern.

Einfachheit: Die Software soll so einfach wie möglich gestaltet werden, keine Vorbereitung möglicher zukünftiger Erweiterungen, keine redundante oder unnötige Funktionalität und keine redundanten Strukturen sind geduldet. Dadurch bleibt das System einfach und wartbar. Dies basiert auf der XP-Annahme, dass es effizienter ist, heute etwas einfaches zu erstellen und morgen etwas mehr Aufwand zu investieren, um Änderungen einzubauen, als heute Komplexes zu entwickeln, das morgen nicht oder nicht in der antizipierten Form genutzt wird.

Feedback: Viele heutige Projekte scheitern an Missverständnissen zwischen Entwickler und Anwendern. Evolutionäre Entwicklung des Systems in möglichst kleinen Releases und eine permanente Verfügbarkeit der Kunden erlaubt schnelles Feedback und dadurch flexible Steuerung des Projektfortschritts. Eine weitere wichtige Quelle des Feedbacks ist die

Entwicklung von Tests auf verschiedenen Ebenen (Unit-Tests, Test-Stories), um zu prüfen, ob die realisierte Funktionalität korrekt und robust ist und gegebene Anforderungen erfüllt.

Eigenverantwortung: Die Entwickler sind angehalten, eigenverantwortlich zu handeln. Das impliziert, in Absprache mit dem Kunden Funktionalitäten anzupassen, Prioritäten zu aktualisieren und Pläne zu überdenken. Verantwortungsbewusstsein beinhaltet auch, dass jeder Programmierer den Überblick über das Gesamtsystem hat und sich zutraut, von Kollegen entwickelten Code zu modifizieren.

XP kennt drei wesentliche Rollen: den Projektleiter, den Kunden und den Entwickler. Der *Projektleiter* ist verantwortlich für das Management und die Koordination des Projekts, er verwaltet Ressourcen, Kosten und insbesondere Zeitpläne, ist aber im Normalfall ebenfalls als Entwickler tätig. Wenigstens ein *Kunde* ist permanent ansprechbar, um schnell aufkommende Fragen zu klären. Idealerweise sitzt er mit den Entwicklern im selben Raum und entwirft funktionale Tests für die Software (User-Stories). Die *Entwickler* tragen die Hauptlast des Projekts. Sie kodieren, testen, entwerfen und hören dem Kunden aufmerksam zu. Genaugenommen vereinigen dadurch die Entwickler mehrere Rollen in sich. Insbesondere die im nächsten Kapitel noch genauer besprochene Rolle des Testers ist für die erfolgreiche Anwendung der Extreme Programming-Vorgehensweise essentiell. Da XP noch eine relativ neue und im Ausbreitung befindliche Vorgehensweise ist, wird gerne zusätzlich ein *Coach* hinzu gezogen, der seine XP-Kenntnisse den übrigen Projektmitgliedern vermittelt und darauf achtet, dass das Team keine falsche Richtung einschlägt.

Durch die Besonderheiten von XP gibt es eine Reihe von notwendigen Rahmenbedingungen für XP-Projekte. So scheint nur eine Maximalzahl von 10 Programmierern in einem Projekt wirklich sinnvoll, obwohl bereits größere XP-Projekte erfolgreich abgeschlossen wurden. Es muss ein Kunde zur intensiven Einbindung in das Projekt zur Verfügung stehen. Die Räumlichkeiten erlauben intensive Kommunikation und Kontakte und der Kunde verzichtet auf eine ausführliche Dokumentation von Analyse und Entwurf.

XP in Reinkultur beschreibt *vier wesentliche Aktivitäten*:

Kodierung: Das System wird in kleinen Schritten inkrementell erweitert. Kodierungsstandards und regelmäßige Durchläufe aller Tests, die gemeinsam mit dem Code entwickelt werden, sind wesentlicher Bestandteil der Kodierung. Modifikationen am Code werden mit Hilfe von Refactoring-Techniken [Fow99] durchgeführt.

Testen: Jedes Programmelement besitzt automatisierte Tests. Komponententests entstehen gemeinsam mit dem Code. Kunden schreiben funktionale Tests, die die Geschäftslogik prüfen. Tests sind in XP methodisch besser verankert und integriert, als dies bei anderen Vorgehensweisen der Fall ist.

Zuhören: Kommunikation zwischen den Entwicklern, sowie mit dem Kunden ist essentiell.

Design: Ist Teil der täglichen Programmiertätigkeit. Während der Design-Aktivität wird u.a. die Systemlogik organisiert. Ein explizites Modell oder Design-Dokument wird aber nicht erstellt.

Folgende *fundamentale Prinzipien* werden propagiert:

Schnelles Feedback: Erlaubt eine kontinuierliche Projektsteuerung, u.a. durch Priorisierung der Anforderungen.

Einfachheit: Fördert die Klarheit und Eleganz des Codes.

Inkrementelle Änderungen: Verhindern die Problemstellungen eines Big-Bang und erlauben einen messbaren Fortschritt.

Änderbarkeit unterstützen: Um damit Flexibilität zu erhöhen.

Qualitativ hochwertige Ergebnisse: Wird angestrebt durch eine Reihe von geeigneten Maßnahmen.

Auf Basis des vorgeschlagenen Wertesystems, der Rollenverteilung und der groben Aktivitätsbeschreibung definiert XP eine Reihe von *Entwicklungspraktiken*, von denen ein Teil hier genannt wird:

Das Planspiel: Dient zur Erhebung von Anforderungen. Kunden entscheiden über die zu realisierende Funktionalität, ihre Priorisierung und die daraus erwachsende Release-Planung. Entwickler schätzen und entscheiden über die Aufwände, Konsequenzen und Vorgehensweise zur Implementierung der gewünschten Funktionalitäten.

Metaphern: Sind dazu gedacht Grundprinzipien im Code durch anschauliche Begrifflichkeiten zugänglich zu machen.

Pair-Programming: Je zwei Personen sitzen an einem Rechner: Abwechselnd tippt der eine, während der andere ihm über die Schulter schaut. Entwürfe werden so schneller und besser durchdacht, Fehler leichter erkannt, Neulinge können von Kennern effizient in den Code eingearbeitet werden, und die Kommunikation ist intensiv.

Testen, testen, testen: Wird im nächsten Abschnitt genauer behandelt.

Refactoring: Siehe ebenfalls nächster Abschnitt.

Gemeinsamer Codebesitz: Ist eine logische Konsequenz des Pair-Programming und erlaubt eine schnellere Anpassung der Software bei geänderten Anforderungen.

Kleine Releases: Erlauben schnelles Feedback mit dem Kunden und werden idealerweise im fast im Wochentakt durchgeführt.

Kontinuierliche Integration: Sichert, dass Systemteile sich nicht inkompatibel auseinander entwickeln.

Max. 40 Stunden Woche: Sichert die Motivation der Entwickler und optimiert so die Leistungsfähigkeit der Entwickler und die Qualität der Ergebnisse.

Ständig verfügbarer Kunde: Notwendig für ein intensives Feedback und zum Ausgleich der fehlenden Dokumentation.

Kodierungsstandards: Sind eine notwendige Vorbedingung für gemeinsamen Codebesitz und den Ersatz von Dokumentation durch den Code.

Unter den in XP genutzten Konzepten spielen gerade die Entwicklung automatisierter Tests und die Verbesserung der Software durch Refactoring eine wesentliche Rolle. Beide Konzepte greifen intensiv ineinander, weil die Prüfung der Korrektheit von Refactoring-Maßnahmen ohne eine existente Test-Sammlung nicht praktikabel ist.

2.2 Testen in XP

Testen ist eine der wichtigsten Tätigkeiten in XP-Entwicklungsprojekten. Ziel der Testaktivität ist die Erstellung von automatisierten Tests mit einer möglichst vollständigen Überdeckung der Funktionalität des entwickelten Programms. Tests sind auf mehreren Ebenen zu entwickeln. Beginnend mit einfachen Methoden-Tests über Unit-Tests bis hin zur Überprüfung der Korrektheit von Geschäftsabläufen durch auf User-Stories basierenden Tests muss alles vertreten sein.

In XP sind alle Tests automatisiert. Damit wird es möglich, jederzeit und damit auch nach kleinen Änderungen die Korrektheit des Systems durch Tests zu prüfen. Tests und Code werden parallel entwickelt, um so eine Testüberdeckung besser zu garantieren. Black-Box-Tests werden typischerweise vor der Methoden-Implementierung, White-Box-Test parallel oder nach

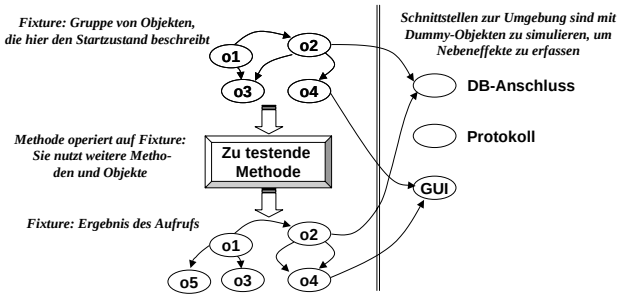


Abbildung 1: Typische Form eines automatisierten Tests

der Methoden-Implementierung realisiert. Der Kunde beschreibt User-Stories, die im Idealfall von Kunden selbst dazu genutzt werden, Test-Cases für Geschäftsanwendungen zu realisieren. Für effizientes Management und zur Definition von Tests stehen für verschiedene Sprachen Test-Frameworks zur Verfügung. JUnit für Java gilt heute als am bekanntesten [Jun01]. Auch bei der Verwendung von Test-Werkzeugen gilt in XP das Prinzip der Einfachheit. Dies impliziert den Verzicht auf komplexe Tools und die Benutzung des einfachen Frameworks JUnit, das in der Entwicklungssprache selbst geschrieben wurde.

Bieten Tests eine gewisse Überdeckung der Systemfunktionalität, so kann die Gesamtheit der Tests als ein Modell des Softwaresystems verstanden werden. Anders als explizite Modelle, die z.B. in UML repräsentiert werden, ist ein Test-Modell sehr implizit, hat aber den Vorteil der automatisierten und wiederholbaren Prüfbarkeit. Dies ist insbesondere bei der Weiterentwicklung des Systems von Vorteil. Der Verzicht auf die Zuordnung von Code zu Verantwortlichen (Besitzern) ist nur möglich, weil korrekte Testläufe einem Entwickler eine gewisse Sicherheit geben, dass er fremden Code nicht in unzulässiger Weise verwendet hat. Insbesondere bei der Verbesserung der Struktur eines Systems durch Refactoring-Techniken ist eine gute Sammlung von Tests zwingende Voraussetzung. Die Vorteile einer guten Test-Sammlung zeigen sich erst später im Projekt. Es erfordert aber einigen initialen Aufwand eine Testumgebung einzurichten (siehe Abb. 1), und daher einiges an Disziplin, auf interaktives Debugging z.B. durch Test-Ausgaben zu verzichten und stattdessen automatisierte Tests zu schreiben.

2.3 Refactoring

Der Wunsch nach Techniken zur inkrementellen Verbesserung und Modifikation von Programmcode ist fast so alt wie die Programmierung selbst

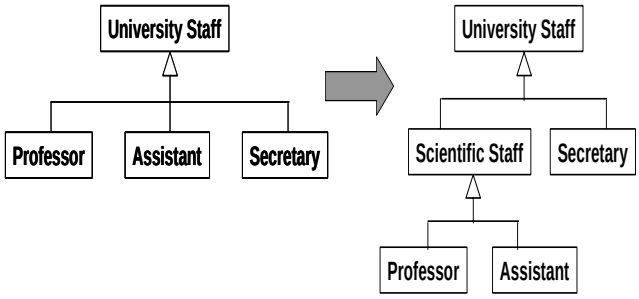


Abbildung 2: Restrukturierung einer Klassenhierarchie als korrektheitserhaltendes Refactoring

[BBB+85]. Ziel einer transformationellen Software-Entwicklung ist die Zerlegung des Software-Entwicklungs-Prozesses in kleine, systematisch durchführbare Schritte, die aufgrund lokaler Anwendung überschaubar werden. Das empfehlenswerte Buch [Fow99] beschreibt Transformationstechniken auf Java-Basis. Es erlaubt die Migration von Code entlang von Klassenhierarchien, die Zusammenlegung oder Teilung von Klassen, die Verschiebung von Attributen, das Expandieren oder Falten von Code-Teilen in eigene Methoden, und ähnliches mehr (siehe Beispiel in Abb. 2). Die Stärke der Refactoring-Techniken entsteht durch die Überschaubarkeit der einzelnen Schritte und durch die flexible Kombinierbarkeit zu großen, zielorientierten Verbesserungen der Software-Architektur.

Das Ziel des Refactoring-Konzepts sind semantikerhaltende Transformationen eines bereits existenten Programms. Refactoring dient nicht zur Erweiterung der Funktionalität, sondern zur Verbesserung der Qualität des Entwurfs unter Beibehaltung der Funktionalität. Es ergänzt damit die normale Programmiertätigkeit (siehe Abb. 3). Ziel ist die Implementierung der gewünschten Funktionalität unter Beibehaltung eines hohen Qualitätsstandards für Design und Architektur. Dadurch bleibt das System wartbar, einfach und seine Qualität hoch.

Zur Sicherung der Korrektheit semantikerhaltender Transformationen werden keine Verifikations-Techniken eingesetzt, sondern die vorhandene Test-Sammlung genutzt. Unter der Annahme der Existenz einer qualitativ hochwertigen Test-Überdeckung ist die Wahrscheinlichkeit hoch, dass fehlerhafte Modifikationen erkannt werden. Natürlich müssen oft Tests gemeinsam mit dem Code refaktorisiert werden, wodurch das Risiko entsteht, dass tatsächlich fehlerhafte Refactoring-Anwendungen nicht erkannt werden.

Refactoring-Techniken zielen auf verschiedene Ebenen des Systems. Manche Refactoring-Regeln wirken im Kleinen, andere sind eher für Modifikation

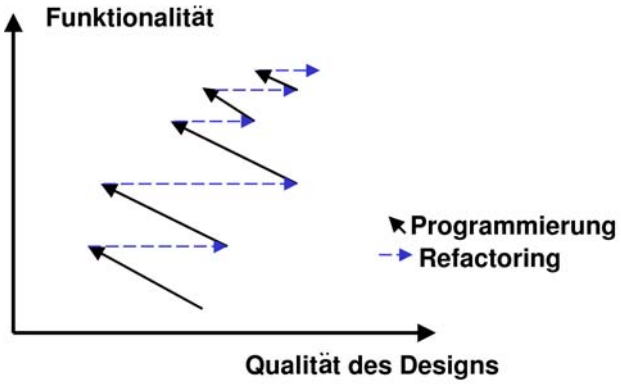


Abbildung 3: Refactoring und normale Programmierung ergänzen sich

einer System-Architektur geeignet. Durch die Möglichkeit, eine bereits im Code realisierte System-Architektur zu verbessern, verliert die Notwendigkeit a priori eine korrekte und stabile System-Architektur zur definieren an Brisanz. Natürlich sind Modifikationen an der System-Architektur kostenintensiv. In XP wird jedoch angenommen, dass die Wartung ungenutzter System-Funktionalität auf Dauer kostenintensiver ist. XP nimmt hier sehr deutlich den Standpunkt ein, die System-Architektur einfach zu halten und nur bei Bedarf durch geeignete Refactoring-Techniken zu modifizieren oder zu erweitern.

2.4 Bekannte XP Projekte

WyCash

Das erste kommerzielle „XP“ Projekt findet sich bereits 1991. Es handelt sich um das WyCash Projekt, an dem Ward Cunningham, ein amerikanischer Softwareentwickler und Entwurfsmusterspezialist, beteiligt war. Im WyCash Projekt wurde ein Portfolio Management System in der Sprache Smalltalk entwickelt; es ist heute noch unter dem Namen MaPS erhältlich (siehe `http://www.InvesSys.com`). In dem Projekt wurden laut Aussagen der Entwickler bereits alle XP Konzepte verwendet, allerdings waren diese damals noch nicht unter dem Namen XP als Vorgehensmodell definiert. Kent Beck gab der von ihm in Zusammenarbeit mit Ward Cunningham und Ron Jeffries [1] entwickelten Methode 1995 den Namen XP. (Quelle: [Web01a])

[1] ebenfalls ein amerikanischer Softwareentwickler

Daimler Chrysler: C3

Das C3 Payroll Projekt bei Daimler Chrysler ist *das* XP Referenzprojekt. Es war das erste tatsächliche kommerzielle XP Projekt im grossen Stil und wurde von Kent Beck gecoached. Es startete 1996 mit einem Team von 10 Entwicklern. Die Sprache war Smalltalk, die erstellte Software ein Gehaltsabrechnungssystem von über 10.000 Mitarbeitern, das ein System von 10 - 14 Payroll-Programmen ablöste. Das Projekt lief vier Jahre. (Quelle: [Web01a]).

Bei der Referenzierung dieses Projekts wird jedoch gerne verschwiegen, dass in der ersten Phase dieses Projekt fast fehlgeschlagen wäre. In dieser Not wurde unter anderem Kent Beck als Coach hinzugezogen und der bereits existente Code durch Tests verstehbar gemacht und mittels Refactoring verbessert. Die schwierige Ausgangssituation dieses Projekts führt hinzu den Einführung der radikal neuen Methode „Xtreme Programming“

Ford Motor Company: VCAPS

Das *Vehicle Cost and Profit System (VCAPS)* Projekt bei der Ford Motor Company startete vor fünf Jahren. Das System wurde anfänglich unter Verwendung des Wasserfall Modells entwickelt, seit 1999 wurde XP benutzt. Das Team bestand aus 7 Personen. Das Projekt wurde kurz vor seiner Fertigstellung aus externen Gründen gekippt, da ein System, mit dem es zusammenarbeitete, ersetzt wurde. (Quelle: [Web01a])

Weitere Projekte

Die Bayerische Landesbank entwickelt ein Credit Risk Managment Tool in Smalltalk unter Verwendung von XP. Pick 'n Pay, die grösste Supermarkt-Kette in Südafrika, lässt Software unter Verwendung von XP entwickeln. (Quelle: [Web01a]) BASF migrierte in Zusammenarbeit mit Andrena Objects eine Anwendungsentwicklung für zugeschnittene Lösungen im Bereich Supply Chain Management mittels XP Techniken (siehe [GE01]).

Die Banca IMI S.p.A., eine der gössten italienischen Investmentbanken, setzt XP ebenso ein wie Alcatel und die belgische Europeloan Bank. (Quelle: [Ser01]) SUN Microsystems setzt XP bei der Entwicklung von Java 1.4 ein, im speziellen bei der Verbesserung des Security API. Gründe für den Einsatz von XP waren Fehlerreduzierung aufgrund von Pair Programming und Common CodeOwnership. (Quelle: Vortrag der Java User Group im Dezember 2000 in München)

3 XP in Lehre und Forschung

Die aktuelle Lehrmeinung im Software-Engineering fordert eine saubere Vorgehensweise von der Analyse der Geschäftsvorfälle bis zur Wartung. Es wird eine systematische, teilweise bürokratisierte Vorgehensweise mit entsprechender Dokumentationsleistung erwartet. XP bricht also mit dieser insbesondere in Europa vorherrschenden Lehrmeinung. Weil aber XP eine deutliche Praxisrelevanz bekommen hat, ist es für die akademische Welt geboten, sich auch mit dieser Methodik kritisch auseinander zu setzen. Bei genauerer Analyse wird zu erwarten sein, dass die akademische Lehrmeinung und Extreme Programming nicht unversöhnlich sind. So sind z. B. die Techniken des Refactoring durchaus eine pragmatische und gleichzeitig rigorose Umsetzung eines uralten akademischen Wunsches nach transformationeller, systematischer Softwareentwicklung. Es ist Aufgabe der akademischen Welt, die einzelnen Techniken von XP kritisch zu durchleuchten, seine Annahmen und Schlussfolgerungen mit systematisch erhobenem Zahlenmaterial zu be- oder widerlegen und auf dieser Basis konstruktive Verbesserungsvorschläge vorzunehmen. Die Erhebung in diesem Bericht beschreibt dazu einige Ansatzpunkte. Zuvor jedoch soll aufgezeigt werde, wie sich XP-Elemente für die Lehre einsetzen lassen.

3.1 XP in der Lehre

Während eines Studiums lösen Studenten zahlreiche kleine Programmieraufgaben. Darüber hinaus führen sie typischerweise zwei bis drei halbjährige Systementwicklungsprojekte durch, die ernsthafteren Projektcharakter simulieren. Wird ein solches Projekt in Teamarbeit durchgeführt, so können einige Elemente von XP genutzt werden, um die Effizienz der Beteiligten und die Qualität des Ergebnisses zu steigern. So erlaubt etwa die Forderung nach automatisierten Tests eine verbesserte Übergabe von Ergebnissen. Ein permanent anwesender Kunde (Aufgabensteller, Betreuer), der aktiv an der Systementwicklung mitwirkt, kann als Experte und Vorbild für die Studenten wirken. Die Verinnerlichung fundamentaler Prinzipien, wie Einfachheit, inkrementelle Änderungen oder schnelles Feedback hilft den Studenten in späteren Projekten.

Um die Notwendigkeit von Tests zu motivieren, können diese bereits im ersten Semester anhand kleiner Programmieraufgaben eingesetzt werden. So kann eine Test-Sammlung vorgegeben sein, die den Studenten die selbständige Überprüfbarkeit ihrer Lösung ermöglicht. (eine ausführlichere Test-Suite kann dann für die Betreuer zur Korrektur eingesetzt werden). Alternativ

kann eine leicht fehlerhafte Lösung vorgegeben sein, die durch eine Test-Suite zu korrigieren ist. Danach sollten die Studenten motiviert sein, Programmierlösungen selbständig mit Test-Suite abzuliefern.
Ein weiteres lehrreiches Element von XP ist die sogenannte Extreme Hour (XH). In sieben Phasen zu je 10 Minuten werden die Rollen des XP-Prozesses, Iterationsplanung, Release-Planung, Priorisierung der Anforderungen, Aufwandsschätzungen und ähnliches mehr geübt. Der spielerische Zugang zur Vorgehensweise von XP ist bei Studenten sehr beliebt und gleichzeitig lehrreich.

3.2 Wissenschaftliche Aufarbeitung von XP

Ziel einer Auseinandersetzung mit XP ist sicher die konstruktive Verbesserung von XP, die Integration mit anderen existierenden Techniken oder deren fundierte Ablehnung. XP wurde erstmals Mitte des letzten Jahrzehntes publiziert, ist aber erst Ende 1999 populär geworden. Entsprechend ist eine fundierte Basis von Zahlenmaterial zum Thema XP noch nicht vorhanden. Die spezifische Problematik der Erhebung von Zahlenmaterial bei Software-Entwicklungsprojekten ist bei XP in der gleichen Weise gegeben. Es wird also einige Zeit dauern, bis über gefühlsbasierte Indikatoren hinaus ein besseres Verständnis über Vor- und Nachteile von XP entwickelt sein wird. Eine gesicherte Basis an erfolgreichen und an erfolglosen XP-Projekten existiert derzeit nicht. Indikatoren für die Ursachen gescheiterter XP-Projekte gibt es nicht. Untersuchungen weshalb XP-Projekte erfolgreich abgelaufen sind gibt es ebenfalls nicht. Insbesondere wäre zu erwarten, dass alleine aufgrund hoher Motivation, die die Verwendung eines neuen Softwareentwicklungs-Prozesses mit sich bringt, die Erfolgs-Wahrscheinlichkeit eines Projekts steigt. Dennoch gibt es zu einzelnen Aspekten von XP Arbeiten und Zahlenmaterial, die nicht direkt aus dem XP-Umfeld erhoben wurden. Ohne Anspruch auf Vollständigkeit sollen nachfolgend einige Aspekte von XP diskutiert werden.

Annahme von XP: Lineare Kostenkurve bei Fehlererkennung

Eine der Annahmen in XP stellt wesentliche Erkenntnisse klassischen Software-Engineerings in Frage: nämlich dass Kosten für Änderungen oder Fehlerbehebung exponentiell über die Projektlaufzeit steigen. Innerhalb der XP-Gemeinde wird die Kostenkurve von Entwicklungsfehlern derzeit kontrovers diskutiert. Durch die Nutzung besserer Sprachen (Java), besserer Datenbanktechnologie und die Verbesserung der Programmierpraktiken, insbesondere auf Basis von Kodierungs-Standards, sowie bessere Werkzeuge und Entwicklungsumgebungen, wird behauptet, dass Fehler-Kosten nicht mehr exponen-

tiell über die Zeit steigen [Bec99]. Obwohl gewisse Argumente zumindest partiell für diese These sprechen, muss ein Beleg dieser Aussage durch Zahlenmaterial erst erfolgen. Die Validierung dieser These ist jedoch essentiell für XP.

Annahme von XP: Viele Änderungswünsche der Kunden

Des weiteren nimmt XP an, dass Software generell stark häufig Änderungswünschen der Kunden unterliegt. Die dadurch ausgelöste permanente Erosion der in normalen Softwareentwicklungsprozessen entstehenden Dokumentation ist dann konsequenterweise durch XP-Techniken zu entgegnen. Bei betrieblichen Software-Systemen, sowie insbesondere internetbasierter Software (e-Business) kann diese Annahme sicherlich als richtig erachtet werden. Speziell im e-Business ist Time-to-Market genauso wichtig wie die Qualität der Software. Es gibt jedoch auch andere Projektumfelder: z. B. im Bereich eingebetteter Systeme, bei Projektaufträgen der öffentlichen Hand, verteilt realisierten Projekten oder Projekten mit deutlich mehr als 10 Entwicklern.

Annahme von XP: Einarbeitung in dokumentierten Code ist wenig aufwendig

Eine weitere Annahme von XP ist, dass die Einarbeitung in gut dokumentierten Code, gegebenenfalls unter Zuhilfenahme eines geeigneten Reengineering-Werkzeugs nicht wesentlich aufwendiger ist, als der Zugang über die Dokumentation. Zusätzlich bietet ein Zugang über den Code die Sicherheit, keine falsche Information zu nutzen. Dies ist heute vor allem ein Problem nicht genügend ausgereifter Software-Engineering-Werkzeuge, die die Konsistenz zwischen Modellen und Code nicht ausreichend sichern können. Dem verbreiteten Ansatz der Zuordnung von Modulen an einzelne Entwickler wird in XP eine Absage erteilt. Stattdessen dürfen und sollen alle Entwickler von allen Modulen Kenntnisse und das Recht zur Modifikation haben. Ist schlechter Code erkannt, so soll er verbessert werden. Dies führt leider häufig zu Befindlichkeiten zwischen den Projektbeteiligten, kann aber durch paarweise Programmierung deutlich reduziert werden. Kritisch ist auch zu bewerten, dass Individualisten mit unterschiedlichen Auffassungen sich gegenseitig die Code-Struktur zerstören können. Mit detaillierten Kodierungs-Standards und einer ausgeprägten teamorientierten Projekt-Kultur lässt sich dem begegnen.

Pair-Programming

Die paarweise Programmierung (Pair-Programming) greift stark in die Projekt-Kultur ein. Pair-Programming bedeutet, dass zwei Personen an einem Rech-

ner abwechselnd die Tastatur bedienen, während der jeweils andere über die Schulter sieht. Rein rechnerisch gibt das zunächst doppelten Personalaufwand. Durch geeignete Projekte, speziell auch durch Studenten, können jedoch die Auswirkungen des Pair-Programmings relativ gut untersucht werden. Bei solchen Untersuchungen wurde bereits festgestellt, dass nach einer relativ kurzen Einarbeitungszeit in den neuen Programmierstil zwar der Gesamtaufwand gegenüber Einzelprogrammierung erhöht war, sich aber eine deutliche Reduktion der Projektdauer und insbesondere eine signifikante Erhöhung der Softwarequalität ergeben hat [WKCJ00]. Um die Korrektheit dieser Ergebnisse zu validieren sind weitere Versuche durchzuführen.

Keine objektive Berichterstattung über XP-Projekte, Wartungsproblematik

Eine ganze Reihe von XP-Projekten haben mittlerweile Erfolgsmeldungen hervor gebracht. Die Dunkelziffer erfolgloser XP-Projekte ist unbekannt. Des weiteren ist unklar, wie sich XP-Projekte auf die Wartungsproblematik auswirken. Ein System das längere Zeit ruht „erodiert“. Erosion geschieht aus zwei Gründen: zum einen verändern sich die Anforderungen der Anwender, teilweise getrieben durch veränderte Geschäftsprozesse, teilweise durch Verbesserung des Verständnis des Machbaren. Zum anderen erodiert das Wissen über die Software in den Köpfen der Entwickler. Dadurch wird die Wartungsproblematik und die Wiederaufnahme einer Weiterentwicklung immens erschwert. Aufgrund der Neuheit des XP-Prozesses ist derzeit kein Zahlenmaterial zur Frage der Wartungsfreundlichkeit von XP-Ergebnissen vorhanden. Hier werden Langzeit-Untersuchungen notwendig sein.

Testmethodik für XP

Obwohl Tests in XP eine außerordentliche Rolle spielen, wird nicht auf fundierte Testmethodiken und Testwerkzeuge zurück gegriffen. Stattdessen wird explizit vorgeschlagen, das einfachste vorhandene Testwerkzeug zu verwenden [JAH00, S. 105]. Eine verstärkte Integration der vorhandenen Testkonzepte, insbesondere von Testmetriken und Testgenerierungs-Werkzeugen ist sicherlich wünschenswert. Techniken zur Messung der Systemdefekte auf Basis vorhandener Testsammlungen wären ideale XP-Analyse-Werkzeuge.

Refactoring

Refactoring ist eine der interessantesten Techniken, die mit XP populär werden. Die Evolution eines Softwaresystems durch kleine, systematisch angewendete Transformationstechniken auf Code-Basis bietet ein weites For-

schungsfeld. Neben der Entwicklung geeigneter Werkzeuge zur automatischen oder halbautomatischen Transformation von Code verschiedener Programmiersprachen sind insbesondere Werkzeuge zur automatisierten oder halbautomatischen Verifikation der Korrektheit von Transformationen interessant. Im Compilerbau werden seit langem Transformationen zur Codeoptimierung eingesetzt. Techniken zur Optimierung der Code-Struktur (Ausbalancierung von Aufruf- oder Klassen-Hierarchien, Faktorisierung gemeinsamer Codestücke in eigenständige Methoden oder Entfernung nicht genützten Codes) könnten sich in diesem Kontext als hilfreich erweisen. Eine Ausweitung der Transformationstechniken auf weitere Notationen, wie z. B. UML Klassendiagramme, und deren Integration mit dem Code in geeigneten Werkzeugen dürfte ebenfalls sinnvoll sein.

3.3 Studien zu XP

Studien zu XP und seinen Konzepten konnten nur im Bereich des Pair Programming gefunden werden. Pair Programming oder Collaborative Programming war bereits vor XP bekannt und Studien diskutierten die Frage, ob zwei Programmier an einem Codestück nicht Geld- und Zeitverschwendung bedeuten. Mehrere Studien beweisen Gegenteiliges. Ein Programmierpaar arbeitet zwar nicht doppelt so schnell wie zwei einzelne Programmierer, produziert jedoch qualitativ höherwertigen Code und hat mehr Spass bei der Arbeit.

Feldstudie Collaborative Programming

Nosek beschreibt in [Nos98] eine Feldstudie zu Collaborative Programming. Unter Collaborative Programming wurde hier verstanden, dass zwei Programmierer zusammen an dem selben Algorithmus und Code arbeiten. Erfahrene Programmierer einer IT-Firma sollten zu einem für das Unternehmen wichtigen Problem eine Lösung erarbeiten. Sie arbeiteten hierbei in ihrer bekannten Umgebung mit ihrer eigenen Arbeitsausrüstung. Die Ergebnisse von fünf einzelnen Programmierern wurden mit den Ergebnissen von fünf Programmierpaaren verglichen.

Bewertet wurden einerseits Lesbarkeit und Funktionalität des erstellten Codes, sowie andererseits das Vertrauen der Programmierer in ihre entwickelte Lösung und der Spass bei der Arbeit. Ausserdem wurde die Zeit verglichen, die die Paare und die Einzel-Programmierer jeweils gebraucht hatten, um das Problem zu bearbeiten. Zur Bewertung der Ergebnisse wurde der zweiseitige t-Test verwendet, eine statistische Technik zum Vergleich von zwei kleinen Stichproben.

Die Paare übertrafen die einzelnen Programmierer in allen Kategorien, besonders bei Vertrauen und Spass. Sie entwickelten die Lösung schneller als die einzelnen Programmierer, brauchten jedoch insgesamt mehr Programmierstunden als diese.

Pair Programming Studien

Williams et al. haben an der University of Utah Forschungen zu Pair Programming durchgeführt. Deren Ergebnisse sind zusammengefasst in ([WC01]). Hier zeigen sie unter anderem, dass die leicht erhöhte Entwicklungszeit ausgeglichen wird durch eingesparte Zeit bei der Fehlerbehebung (Qualitätssicherung und Customer Support), da von Programmierpaaren erstellter Code weniger Fehler aufweist als von einzelnen Programmierern erstellter.

Iher statistisch signifikanten Studien zeigen: Nach einer Eingewöhnungsphase braucht ein Programmierpaar ca. 15% mehr Programmierstunden als zwei einzelne Programmierer, um ein Problem zu lösen. Der erstellte Code eines Programmierpaares enthält jedoch ca. 15% weniger Fehler. Da die Fehlerbehebung umso mehr koste, je später der Fehler aufgedeckt würde (für einen durch die Qualitätssicherung entdeckten Fehler zwischen 4 und 16 Stunden, für einen erst durch Anwender im Betrieb aufgedeckten Bug zwischen 33 und 88 Stunden), sparten die 15% weniger Fehler im Code eines Programmierpaares deutlich mehr Kosten ein, als durch die zusätzliche Entwicklungszeit von 15% entstünden.

Auch Williams stellte in ihren Studien eine höhere Zufriedenheit der Programmierer beim Einsatz von Pair Programming fest. Zur eventuellen Kosteneinsparung durch die höhere Zufriedenheit enthält [WC01] keine Aussagen (beispielsweise zu Kosten, die eine Kündigung eines Entwicklers und eine darauf hin nötige Neueinstellung verursachen).

3.4 Erweiterungen von XP

XP und Frameworks

Wesentliche Elemente von XP sind die Einfachheit des Entwurfs, die stete Entfernung nicht genützten Codes und die Realisierung genau der Funktionalität, die von den Kunden gewünscht ist. Dementsprechend ist XP für die Entwicklung eines Frameworks ungeeignet. Das heißt aber nicht, dass in XP-Projekten keine existenten Frameworks eingesetzt werden sollten oder dass nicht „zufällig“ durch mehrere verwandte XP-Projekte ein Framework entstehen kann. Interessant ist XP vor allem beim Einsatz von Frameworks wenn diese geeignet vorbereitet sind. Ein Framework definiert sich durch eine

Sammlung von Klassen mit starker Interaktion, eigenem Kontrollfluss und eine Menge von Klassen bzw. Methoden, die dazu gedacht sind, durch applikationsspezifische Implementierungen redefiniert zu werden. Die Verwendung eines Frameworks ist im Sinne von XP erwünscht, wenn es das Projekt schneller zu einem qualitativ hochwertigeren Ergebnis bringt. Schwierig jedoch wird es, wenn einzelne Verhaltensweisen des Frameworks nicht bekannt sind. Durch die Entwicklung von Tests können diese Verhaltensmuster in Erfahrung gebracht werden. Umgekehrt kann und sollte ein Framework selbst Testklassen mitliefern, die dazu geeignet sind, Applikationsklassen auf ihre Korrektheit und Robustheit im Bezug auf die Redefinition von Hot-Spots zu prüfen. Solche Framework-Tests können für die Applikations-Entwicklung sehr hilfreich sein. Leider ist bis dato kein Framework bekannt, das eine derartige Test-Sammlung mitliefern würde.

Extreme Modeling

Eine Weiterentwicklung des Extreme Programming, ist das sogenannte „Extreme Modeling“. Es basiert auf der Überlegung, dass die Entwicklung noch effizienter und in noch kürzeren Zyklen ablaufen kann, wenn statt einer klassischen Programmiersprache eine ausführbare Teilsprache der UML verwendet wird. Mehrere Werkzeug-Hersteller, wie z.B. GentleWare.de entwickeln derzeit solche Code- und Testgeneratoren. Ziel ist es, Klassen-, Sequenz-, Statechart- und Use-Case-Diagramme teilweise zur konstruktiven Generierung von Code und teilweise zur Generierung von automatisierten Tests einzusetzen. Des weiteren wird über UML-basiertes Refactoring wieder nachgedacht, denn Refactoring wurde ursprünglich für Klassendiagramme eingeführt. Es ist vorstellbar, dass in wenigen Jahren eine konsolidierte, ausführbare Teilsprache der UML existiert, die in Kombination mit textuellen Anteilen aus Java oder einer anderen Programmiersprache ein ideales Programmierwerkzeug für XP-basierte Projekte darstellt. Bei all den Vorteilen die Extreme Modeling auf UML-Basis haben kann, sollte jedoch darauf geachtet werden, dass UML nicht nur als Programmiersprache, sondern auch als abstrakte Modellierungssprache für frühe Projektaktivitäten entwickelt wurde. Dies widerspricht aber keineswegs der Nutzung einer ausgezeichneten Teilsprache zur Code-Generierung.

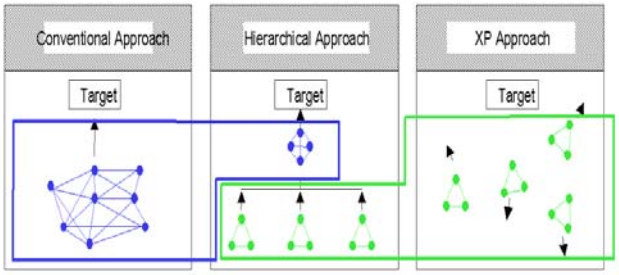


Abbildung 4: Hierarchische XP-Organisation mit projektübergreifendem Steuerkreis (mitte) im Vergleich zu konventionellem Ansatz (links) und reinem XP-Ansatz (rechts)

Hierarchische Organisation mehrerer XP-Projekte

Aus der betriebswirtschaftlichen Projektorganisation kommt ein Vorschlag, um XP-Projekte mit deutlich mehr als 10 Mitarbeitern umsetzen zu können [JR01]. Wesentliche Idee dabei ist, die Strukturierung großer Projekte in viele kleine XP-Projekte und einen Steuerkreis (siehe Abb. 4). Der Steuerkreis setzt sich zusammen aus wenigstens einem Mitglied jedes Teilprojekts, einem Werkzeug-Verantwortlichen, einem Versions-Manager, dem Gesamtprojektleiter und mindestens einem Kunden. Unter anderem identifiziert und bearbeitet er auftretende Risiken, entscheidet über Technologiefragen, identifiziert und rearrangiert Teilprojekte, sichert den Kommunkationsfluss zwischen den Teilprojekten und definiert deren softwaretechnische Schnittstellen. Gerade in der Schnittstellen-Problematik liegt aber die große Schwierigkeit der Zerteilung eines großen Projekts in XP-Teilprojekte. Hier müssen im Vorfeld bestimmte Aktivitäten zur Definition von Schnittstellen und meist der dahinterliegenden Software-Architektur getroffen werden. Dies ist z. B. in den Bereichen eingebettete Systeme und Telekommunikation zumeist der Fall. Projektorganisation kann dann eine wie in Abb. 4, Mitte dargestellte Form annehmen.

4 Die Studie: XP Projekte in der Praxis

Ein wichtiger Fokus dieser Arbeit ist, den tatsächlichen Einsatz von XP transparent zu machen. Deshalb wurde eine Befragung von Unternehmen und Beratern, die XP Projekte durchführen, vorgenommen. Ziel dieser Befragung war es, Daten zu sammeln, durch deren Auswertung Voraussetzungen geschaffen werden können, um XP noch gewinnbringender im Hinblick auf eine erfolgreiche Softwareentwicklung einzusetzen. Notwendige Basis hierfür ist die Erhebung von objektiven Zahlenmaterial zum Einsatz von XP in der Industrie.

In [Rum01] wurde dazu ausgeführt, dass derzeit nur wenige genaue Zahlen über XP Projekte vorliegen. Es sei Aufgabe der akademischen Welt, die Annahmen und Schlussfolgerungen von XP mit systematisch erhobenem Zahlenmaterial zu be- oder widerlegen und auf dieser Basis konstruktive Verbesserungsvorschläge vorzunehmen. XP in seiner jetzigen Form stelle nur einen Zwischenstand dar und werde sich weiterentwickeln, wozu auch die Entwicklung eines besseren Verständnisses über die Vor- und Nachteile von XP auf Basis wissenschaftlich gesicherter Untersuchungen gehöre. In einer IBM-Umfrage ([IBM01]) heisst es:

> *„Even the advocates of XP agree that it's too early to know how effective the methodology is.“*

Hier setzt die Studie an. Sie legt Schwerpunkte auf den Umgang mit dem XP Prozess und seinen methodischen Konzepten sowie auf spezielle Vorteile, aber auch Schwierigkeiten und Gefahren bei XP Projekten. XP besteht aus zwölf Elementen, die in Industrie und universitärem Umfeld teilweise kontrovers diskutiert werden.

- Wie bewähren sich diese Elemente im täglichen Einsatz?
- „Funktioniert“ XP wirklich oder bestätigt sich die Kritik an XP?
- Kann durch den Einsatz von XP tatsächlich erfolgreichere Softwareentwicklung erreicht werden?

Soll sich XP als ernstzunehmende Softwareentwicklungsmethode durchsetzen, ist auch wichtig, Risiken zu erkennen, die speziell bei XP Projekten auftreten können oder durch die Besonderheiten von XP bedingt sind. So kann vermieden werden, bei der Durchführung eines Projekts in typische Fallen zu tappen. Die Studie will hier einen Beitrag zur Aufklärungs- und Verbesserungsarbeit leisten und gleichzeitig eine erste statistische Erhebung über XP Projekte geben.

Wie in Abschnitt 4.3 noch detaillierter diskutiert werden wird, ist zur momentanen Zeit die Erhebung von aussagekräftigem, statistisch signifikanten Zahlenmaterial mit gewissen Schwierigkeiten verbunden. Zum einen ist XP eine noch junge Disziplin, so dass die Anzahl der weltweit durchgeführten Projekte begrenzt ist. Der Rücklauf unter den im Rahmen der Befragung angesprochenen Personen kann aus unterschiedlichen Gründen weiter beschränkt sein.

In Abschnitt 4.1 wird der zur Erhebung genutzte Fragebogen vorgestellt. Abschnitt 4.2 beschreibt die Durchführung der Datenerhebung. In Abschnitt 4.3 wird das Antwortverhalten der angeschriebenen Personen diskutiert. Abschnitt 4.4 beschreibt das Vorgehen bei der Auswertung der gesammelten Daten.

4.1 Fragebogen

Als Grundlage für die Studie und die Datenerhebung wurde ein Fragebogen entwickelt, der im Ganzen in Anhang A zu finden ist. Er besteht aus 27 Fragen, die in neun Kategorien aufgeteilt sind. Die Auswertungen zu den einzelnen Fragengruppen finden sich in den entsprechenden Abschnitten des Kapitels 5. Der Fragebogen enthält ausserdem einige weitere Fragen, die nicht in Kategorien eingeteilt wurden, sondern dazu benutzt wurden, ein noch besseres Gesamtverständnis für die beschriebenen XP Projekte zu entwickeln. Die Fragen sind im Fragebogen nicht nach Kategorien geordnet, sondern in lockerer Reihenfolge gemischt.

Da Softwareentwickler und Projektmanager im Allgemeinen wenig Zeit zur Verfügung haben, wurde im Fragebogen auf eine Reihe tiefergehender Fragen verzichtet. Dadurch sollte die Hürde einer vollständigen und korrekten

Ausfüllung des Fragebogens niedrig gehalten werden. Statt also alle möglichen Fragestellungen im Fragebogen bereits zu antizipieren, wurde den Teilnehmern der Studie durch relativ allgemein gestellte Fragen die Möglichkeit gegeben, eigene Eindrücke zu formulieren.

Der erste Fragenblock (Abbildung 5) sammelt firmenbezogene Daten zu den Unternehmen, in denen XP Projekte durchgeführt werden. Interessant ist hier die Branche, der das Unternehmen angehört, die Verteilung der Projekte auf Kontinente und weiterhin auf Länder, sowie Größe und Alter der Firma. Hier ist die Motivation, zu sehen, ob junge Firmen der New Economy eher XP einsetzen als alteingessesene Grossunternehmen; ob XP hauptsächlich von Software- und Technologiefirmen eingesetzt wird, oder ob es auch Einzug gehalten hat in klassische, eher konservative Branchen wie den Finanzsektor; ob eine Anhäufung von Projekten in den USA festzustellen ist, der Wiege von XP, und es dagegen vielleicht einen Kontinent gibt, der kaum XP Aktivitäten aufweist. Ausserdem ist die Sicht auf das Projekt wichtig. Wurde der Fragebogen von dem technischen Projektleiter ausgefüllt, von einem Entwickler oder einem XP Berater? Motivation ist hier, die Perspektive aus der heraus das Projekt beschrieben wurde, in die Bewertung miteinzubeziehen.

- *City and country where company is located:*
- *Some information about the company (how big, founded in, what line of business is it in, ...):*
- *Role of person who filled in this questionnaire:*

Abbildung 5: Block 1 des Fragebogens

Der nächste Fragenblock (Abbildung 6) gilt etwaiger XP-Vorerfahrung des Unternehmens und der Entwickler im Projekt. Hier ist interessant, ob das Projekt das erste XP Projekt war, das durchgeführt wurde, oder ob das Unternehmen bereits jahrelang XP einsetzt. Auf Basis der Anzahl der bisher durchgeführten XP-Projekte eines Unternehmens kann sich eine Abschätzung dafür finden lassen, wie lange XP bereits eingeführt ist, bzw. welche Expansionsrate die Verwendung des XP-Prozesses hat.Weiterhin wird untersucht, ob XP Wissen im Unternehmen selbst vorhanden ist und inwieweit bei XP Projekten typischerweise ein externer XP-Berater dazugezogen wird. Eine weitere Frage bezieht sich auf die generelle Erfahrung der Entwickler. Hier ist

- *How many other XP projects where carried out before?*
- *How good was the general software engineering training/knowledge of the team members initially?*
- *How many team members had made experiences in XP previously?*
- *How many development companies/independent consultants where involved?*

Abbildung 6: Block 2 des Fragebogens

interessant, in welcher Korrelation unterschiedlicher Wissensstand der Teammitglieder mit dem Projekterfolg steht.

Weiterhin wurden Daten zu den Projekten selbst gesammelt. Interessant ist, inwieweit objektorientierte Sprachen eingesetzt werden – wie die Teamgrösse bei XP-Projekten tatsächlich aussieht – inwieweit eventuell eine Korrelation zwischen Teamgrösse und Projekterfolg besteht – ob XP eher für kürzere, "einfache" Projekte eingesetzt wird oder auch für grössere Projekte, in denen kritische Systeme entwickelt werden – ob die Systeme komplett neu entwickelt werden oder ob bestehende, auf herkömmliche Art entwickelte Systeme mit XP weiterentwickelt werden. Die Fragen hierzu zeigt Abbildung 7.

Ein wichtiger Punkt ist, warum entschieden wird, Projekte mit XP zu entwickeln. Wird der Einsatz von XP motiviert durch Unzufriedenheit mit herkömmlichen Softwareentwicklungsmethoden? Die Frage hierzu zeigt Abbildung 8.

Mit einem Kernaspekt der Studie beschäftigt sich der nächste Fragenblock (Abbildung 9).

Wie wird XP im Projekt eingesetzt? Interessant ist, mit wievielen Teammitgliedern die einzelnen XP-Rollen besetzt sind, wieviele Kunden beteiligt sind, welche XP Konzepte tatsächlich benutzt werden und in welchem Umfang. Die Motivation ist hier, den tatsächlichen Umgang mit XP und seinen Konzepten zu erfahren. Wie massiv werden die einzelnen Konzepte eingesetzt? Werden Sie von Entwicklern als hilfreich empfunden, als verbesserungswürdig, oder gar als erschwerend für die Entwicklung? Wie wird XP in seinem tatsächlichen Einsatz modifiziert? Erweitern die Teams XP,

- *Duration of project (from when till when):*
- *Teamsize:*
- *What kind of software was developed?*
- *Has it been a development from scratch (new system), legacy maintenance or adding new functionality on an existing system?*
- *Programming languages used:*
- *Technologies used:*

Abbildung 7: Block 3 des Fragebogens

- *Why did you decide to develop this project with XP?*

Abbildung 8: Block 4 des Fragebogens

kombinieren Sie es mit anderen Methodiken; wie können nicht als hilfreich empfundene Elemente verbessert werden? Und schliesslich: inwieweit können durch den Einsatz von XP die Ziele einer erfolgreichen Softwareentwicklung erreicht werden?

Der nächste Block (Abbildung 10) behandelt eine ebenfalls sehr wichtige Fragestellung: Warum werden XP Konzepte nicht benutzt?

Den entscheidenden Aspekt des Projekterfolgs behandelt der nächste Fragenblock (Abbildung 11). Inwieweit XP seinem Anspruch gerecht wird und sein Einsatz schliesslich in ein erfolgreiches Projekt mündet, wird hier erfragt. Dazu soll der Projekterfolg auf einer Zahlenskala bewertet werden. Ausserdem wird nach Faktoren für den Projekterfolg und nach Risiken gefragt, die den Erfolg gefährdeten. Was war besonders entscheidend für den Erfolg des Projekts? Gibt es spezielle Gefahren, die bei mit XP durchgeführten Projekten auftreten können?

Der anschliessende Fragenblock befasst sich dem weiteren Einsatz von XP (Abbildung 12). Sind die Vorteile, die XP von anderen Softwareentwick-

- *What was the project structure (how many people where there for each role)?*
  Programmers (writing production code and code for component tests):
  Customers:
  Testers (helps customer developing functional tests):
  Coach:
  Further roles (consultant, big boss, tracker...):

- *How many customers with different stakes (requirements, forms of usage for the system) where involved?*

- *Which XP-Elements did you use in the Project (9=fully used, 0=not at all) ? Please say for every element how strong you used it (9-0) and if you consider it helpful(h), improvable(i), or even making-difficult(m) for success of development.*

- *XP was invented to make software development more successful. Some of its main goals are listed below. In your XP Project, could these goals be reached? If not, explain the obstacles? (5=fully achieved, 0=as always, -5=much worse)*
  Deliver software in time:
  Let developers have fun with their work:
  Develop software with a high quality (less bugs):
  Let changes don't cause big costs, because one can react fast to changes:

Abbildung 9: Block 5 des Fragebogens

lungsmethoden abheben soll, so signifikant und in der Praxis auch tatsächlich erreichbar, dass XP als Methode empfunden wird, mit deren Hilfe wirklicher Projekterfolg erreicht werden kann, so dass entschieden wird, es weiter einzusetzen? Inwieweit auch Unternehmen und Entwickler, deren XP Projekt nicht oder nur teilweise erfolgreich verlaufen ist, XP trotzdem erneut einsetzen möchten, ist ebenfalls stark von Interesse.

Ein letzter Fragenblock (Abbildung 13) enthält Fragen, die das Interesse an Extreme Modelling (XM) untersuchen, einer noch sehr im Entstehen be-

- *Please give reasons for the three least used concepts, why you didn't use them? Did you explicitly decide not to, or had there been other obstacles?*

Abbildung 10: Block 6 des Fragebogens

- *Did the project terminate successfully? (9=very successfull, 0=not at all successful)*
- *What where the major reasons for its success / failure? Can you priorize them?*
- *If it was a success what where the main obstacles? How dangerous have they been?*

Abbildung 11: Block 7 des Fragebogens

griffenenen Weiterentwicklung von XP, die die Unified Modeling Language (UML) als grundlegende "Programmiersprache" nützt. Hier soll festgestellt werden, inwieweit XP einsetzende Teams mit XP in seiner jetzigen Form zufrieden sind oder eine Weiterentwicklung Richtung XM befürworten.

Wie oben bereits erwähnt, wäre es möglich und für die Beantwortung tiefergehender Fragestellungen teilweise auch nötig gewesen, den Fragebogen noch detaillierter zu gestalten. Dies hätte durch Aufnahme von weiteren Frageblöcken, zum Beispiel zum Verhalten beim Schreiben von Komponententests, oder stärkere Untergliederung einzelner Fragen und Blöcke geschehen können. Davon wurde abgesehen, da ein Ziel bei der Entwicklung des Bogens war, ihn eher kurz zu halten, um möglichst viele Personen zum korrekten und möglichst vollständigen Ausfüllen zu motivieren. Deshalb konzentriert sich der Fragebogen auf wesentliche Aspekte und Fragestellungen.

Weitere Überlegungen beeinflussten die Vorgabe von Antwortmöglichkeiten. Hier wurde entschieden, nur bei wenigen Fragen konkrete Zahlenskalen zur Beantwortung vorzugeben. Vorgefertigte Antwortmöglichkeiten wurden

- *Will you use XP again?*
- *Are you actively advocating XP in the future?*

Abbildung 12: Block 8 des Fragebogens

- *Are you trained in UML or a similar modelling language?*
- *If you know UML, did you miss it?*
- *Would you like to use UML combined with an XP approach, e.g. for generation of code or tests?*

Abbildung 13: Block 9 des Fragebogens

ebenfalls nicht erstellt; bei den meisten Fragen wurde freier Text als Antwort erwartet. Mit der Studie soll neben statistischem Zahlenmaterial auch ein Gefühl für den Einsatz von XP gewonnen werden; dies ist oft nicht durch konkrete Messwerte oder vorgefertigte Antworten möglich. Es wurde entschieden, dass z.B. bei Fragen wie *Why did you decide to develop this project with XP?* vorgefertigte Antwortvarianten die Bandbreite möglicher Antworten zu stark einschränken würden.

Als Sprache für den Bogen wurde Englisch gewählt, da davon ausgegegangen wurde, dass diese Sprache von den meisten im Softwareentwicklungsbereich tätigen Personen verstanden wird. Auch hier war wieder die Motivation, möglichst viele Bögen zurückzuerhalten und damit möglichst viele Daten für die Studie zu sammeln.

4.2 Datenerhebung

Die Studie soll trotz einer eventuellen Beschränktheit durch eine relativ geringe Anzahl von weltweit durchgeführten XP-Projekten einen möglichst umgreifenden Überblick über laufende und abgeschlossene Projekte bieten.

Aus diesem Grund wurde durch Verteilung des Fragebogens über verschiedene Verbreitungskanäle versucht, bei der Datenerhebung möglichst viele und einen möglichst repräsentativen Ausschnitt von XP Projekten zu erfassen. Dabei kamen vor allem Mailing-Listen, persönliche Kontakte und im Internet recherchierte Ansprechpartner von Projekten zum Einsatz.

Eine grosse Sammlung von Berichten laufender und abgeschlossener XP Projekte findet sich im Wiki Web ([Web01b]). Die Berichte sind teils nur kurze Ankündigungen, dass ein XP Projekt durchgeführt wird oder wurde. Oft wird auch der Name des Unternehmens nicht genannt. Teilweise sind email-Adressen von Kontaktpersonen angegeben. Hier wurde der Fragebogen per email an die Ansprechpartner geschickt; waren für ein Projekt mehrere Ansprechpartner angegeben, wurde der Bogen an alle gemailt, um die Wahrscheinlichkeit des Ausfüllens zu erhöhen. Außerdem war es erwünscht, von verschiedenen Beteiligten desselben Projekts Fragebögen ausfüllen zu lassen, um gegebenenfalls verschiedene Sichtweisen auf ein Projekt vergleichen zu können.

Unter `www.extremeprogramming.org` existiert eine Liste von Unternehmen, die Entwickler mit XP Erfahrung suchen. Es wurde davon ausgegegangen, dass diese Firmen XP Projekte durchführen. Eine (weniger umfangreiche) ebensolche Liste findet sich im Wiki Web. Hier waren als Kontakt meist nur allgemeine Adressen wie `info@` oder `jobs@` angegeben. Der Fragebogen wurde an diese Adressen gemailt mit der Bitte, ihn an die zuständige Person in der Entwicklungsabteilung weiterzuleiten. Auch hier wurde der Bogen in einigen Fällen an mehrere Kontaktadressen desselben Unternehmens geschickt.

Eine weitere Möglichkeit wäre gewesen, Job-Suchmaschinen nach dem Stichwort Extreme Programming zu durchsuchen und diese Firmen ebenfalls anzusprechen; hiervon wurde abgesehen, da der zu erwartende Output zu gering im Verhältnis zum Aufwand eingeschätzt wurde. Auch wurde angenommen, dass Firmen, die unter einschlägigen Webseiten XP Jobs anbieten, stärker an diesem Thema interessiert sind als Firmen, die eher unspezifisch inserieren.

Unter obigen URLs finden sich auch Listen mit Entwicklern, die daran interessiert sind, in Unternehmen zu arbeiten, die XP einsetzen; auch hier wurde davon ausgegangen, dass diese Personen eventuell derzeit bereits in Firmen arbeiten, in denen XP Projekte durchgeführt werden. Auch an diese Personen wurde der Bogen verschickt.

Weiterhin wurden alle Unternehmen und Universitäten angemailt, die zur XP 2001 Konferenz ein Paper mit einem oder mehreren XP-Projektberichten eingereicht hatten. Die Konferenzleitung, hier besonders Professor Michele Marchesi, haben sich als sehr hilfreich erwiesen diese Adressaten anzuspre-

chen.

Zusätzlich wurde der Bogen an persönlich bekannte Ansprechpartner in Unternehmen verschickt, von denen bekannt war, dass sie XP Projekte durchführen.

Der Bogen wurde in obigen Fällen als Attachement per email versandt, insgesamt an 215 Kontaktadressen in 182 Unternehmen bzw. Projekten.

Ein weiterer Verbreitungskanal neben diesen Direktmailings war das Posting in Mailinglisten. Jeweils zweimal im Abstand von einer Woche wurde ein Hinweis auf den Fragebogen mit Angabe einer URL, unter der er abgelegt war, in vier Mailinglisten gepostet. Es handelte sich hierbei um folgende Listen:

- `xp2001@yahoogroups.com`
- `extremeprogramming@yahoogroups.com`
- `xp-forum@yahoogroups.com`
- `ecoop-info@ecoop.org`

Die ersten drei Listen sind spezielle XP Listen, die letzte ist eine Liste zur Objektorientierung. Bis auf xp-forum, die eine deutschsprachige Liste ist, handelt es sich um englischsprachige Listen. Einige weitere Listen, die speziell für Softwareentwickler interessant sind, wie SEWORLD oder ISWORLD lehnten das Posting leider ab, da Postings mit Antwortgesuch nicht ihren internen Richtlinien entsprach.

Insgesamt wurden Fragebögen zu 47[2] XP-Projekten zurückgeschickt. Hierbei wurden von einzelnen Unternehmen und Beratern teilweise Bögen zu mehreren Projekten zurückgeschickt. Projekte, für die mehrere Bögen, ausgefüllt von verschiedenen Projektbeteiligten, eingingen, gab es nicht.

Das Verhältnis von verschickten Bögen zu Antworten auf den einzelnen Verbreitungskänalen zeigt Tabelle 1. Hierbei ist jedoch in Betracht zu ziehen, dass eine Anzahl der direkt angeschriebenen Personen gleichzeitig über die Mailing-Listen angesprochen wurde. Deshalb ist eine 100%-ige Zuordnung nicht möglich.

4.3 Antwortverhalten

In Tabelle 1 werden die Unterschiede in der Effizienz der Verbreitung auf den einzelnen Kanälen deutlich. Relativ uneffizient war die Verbreitung über Mailinglisten. Auf das Posting in vier Mailinglisten kamen insgesamt nur sieben

[2]Zwei dieser Bögen gingen jedoch erst nach Auswertungsschluss ein; diese wurden in die Auszählung nicht mehr mit aufgenommen, aber teilweise in den Abschnitten 5.2 bis 5.10 zitiert. Auch in Abschnitt 5.11 wird auf sie Bezug genommen. In die Auszählung wurden nur die 45 rechtzeitig eingesandten Bögen einbezogen.

Verbreitungskanal	versandte Bögen	Antworten	Antwortrate%
Projektberichte (Wiki Web)	42	13	31.0%
Firmen	48	5	10.4%
XP-Entwickler/Berater	63	11	17.5%
XP 2001 Projekte	23	9	39.1%
persönliche Kontakte	6	2	33.3%
gesamt Direktmailings	182	40	22.0%
Mailinglisten	-	7	-
gesamt	-	47	-

Tabelle 1: Verhältnis Verbreitungskanal - Response

Fragebögen zurück. Eine sehr geringe Response, wenn man bedenkt, dass über Mailinglisten doch sehr viele an einem Thema interessierte Personen erreicht werden können.

Sehr viel effizienter waren die Direktmailings. Hier ergab sich insgesamt eine Response von 22%.

Am höchsten lag hier die Antwortrate bei den Unternehmen und Universitäten, die ein Paper zur XP 2001 eingereicht haben - fast 40% antworteten. Die hohe Response könnte daraus resultieren, dass es sich hierbei um eher wissenschaftlich motivierte XP Projekte handelt und deshalb eine Bereitschaft und Interesse bestand, auch an einer wissenschaftlich motivierten Studie teilzunehmen.

Ebenfalls gut war das Feedback bei den Projektberichten im Wiki Web. 31% der angeschriebenen Kontakte schickten einen ausgefüllten Fragebogen zurück. Hierzu ist zu bemerken, dass die im Wiki Web beschriebenen Projekte teilweise schon länger abgeschlossen sind, weshalb die Bögen nicht notwendigerweise für das im Wiki Web stehende Projekt, sondern gegebenenfalls für ein laufendes oder erst vor kurzem abgeschlossenes Projekt ausgefüllt wurden.

Bei den persönlich angesprochenen Kontakten lag die Antwortrate bei 33.3%, bei durch Recherchen bekannt gewordenen XP-Entwicklern bei 17.5%. Hier antworteten viele XP Berater, die in Unternehmen XP Projekte durchführen. Diese füllten teilweise Fragebögen für bis zu drei Projekte aus, die sie gecoacht hatten.

Die Antwortrate bei den Firmen, die XP Entwickler suchen, lag etwas geringer - 10.4 %. Ein möglicher Grund ist, dass der Fragebogen hier grösstenteils an unspezifische Adressen wie `info@` geschickt wurde mit der Bitte um Weiterleitung; nur selten waren email- Adressen von konkreten Personen

angegeben. Auch hier konnte festgestellt werden, dass persönliche/direkte Mailings wesentlich erfolgreicher sind als unspezifische: obwohl der Bogen zum Grossteil an unspezifische Adressen gesandt wurde, kamen drei der fünf zurückgesendeten Fragebögen von Firmen, bei denen ein konkreter Ansprechpartner angegeben war.

Da der Bogen über die bekanntesten XP Mailinglisten verbreitet wurde, die grössten XP Websites nach Projektberichten und XP einsetzenden Unternehmen durchsucht wurden, bekannte XP Berater sowie die Verfasser von Papers zu einer der zwei wichtigsten XP Konferenzen angesprochen wurden, wird davon ausgegegangen, dass ein signifikanter Anteil von XP Projekten weltweit erreicht wurde. Jedoch ist auch davon auszugehen, dass eine Anzahl von XP-Projekten insbesondere innerhalb konservativer, größerer Firmen abläuft, dies aber aus firmenpolitischen Gründen nach außen hin nicht bekannt gegeben werden darf. Einige unspezifische Anfragen bei einem Elektronik-Konzern, einem Automobil- Hersteller und einer Bank haben derartige Signale vermittelt. Abschnitt 5.8 diskutiert einige weitere interessante Aussagen in diesem Zusammenhang. Eventuelle Projekte in diesen Unternehmen konnten nicht erreicht werden, auch die seit längerem abgeschlossenen Projekte im Wiki Web konnten teilweise nicht erreicht werden. Zieht man zusätzlich die Gesamtantwortrate von 22%[3] in Betracht, besitzt diese Studie damit zwar nur einen limitierten statistischen Wert. Dennoch kann sie wichtige Aussagen über den Einsatz des XP-Prozesses in Softwareentwicklungsprojekten machen, die Trends und Tendenzen aufzeigen. Als erste Studie dieser Art legt sie damit einen Boden für weitere, detailliertere Studien, die sich in den nächsten Jahren mit der Eignung von XP für die Softwareentwicklung beschäftigen werden.

4.4 Vorgehen bei der Auswertung

Wie bereits oben erwähnt, bestand der Fragebogen sowohl aus durch Freitext beantwortbaren Fragen als auch aus Fragen, die einen konkreten Zahlenwert als Antwort erwarten.

Bei Fragen, die mittels Zahlenskala beantwortet werden konnten, wurde für jeden möglichen Antwortwert der prozentuale Anteil an der Gesamtzahl der Antworten ermittelt. Die Gesamtzahl der Antworten variiert je nach Frage, da beispielsweise bei einigen Fragen Antworten von Unternehmen, die zu mehreren Projekten Bögen eingereicht hatten, nur einmal gezählt wurden. Wie dies im jeweiligen Fall gehandhabt wurde, ist bei der Auswertung der einzelnen Fragen angegeben. Fehlende Angaben wurden unter dem Punkt

[3] ohne Mailinglisten

“keine Angabe“ gesammelt. Für diese Fragen wurden jeweils Balkendiagramme erstellt, die die Verteilung auf die einzelnen Antwortwerte visualisieren.

In einer Interpretation der einzelnen Ergebnisse, bei der Antworten teilweise gruppiert wurden und diesen Gruppierungen jeweils eine eigenständige Bewertung zugewiesen wurde, wurden die Ergebnisse bewertet.

Bei Fragen, die mittels Freitext beantwortet werden konnten, wurden die Antworten abhängig vom Frage- und Ergebniskontext ausgewertet. War eine Klassifizierung der Antworten in mehrere Gruppen möglich, so wurde diese Gruppierung vorgenommen und auf dieser Basis %-Punkte berechnet. Auch hier wurden in einigen Fälle Diagramme zur besseren Veranschaulichung erstellt. Ansonsten wurden die Antworten nur in einer Bewertung interpretiert.

5 Auswertung der Studie

Die Auswertung der im Rahmen der Studie zurückgesendeten Fragebögen zeigt eine klare Richtung auf: die Zufriedenheit mit XP bei den befragten Unternehmen und Personen ist sehr hoch. Alle Projekte bis auf eines wurden als erfolgreich bewertet, eines wurde als teilweise erfolgreich bewertet.

Die Hauptziele von XP, wie Software termingerecht fertigzustellen und Spass bei der Arbeit zu haben, wurden durchgehend als erreicht angesehen, der Grossteil der XP Elemente wurde intensiv benutzt und als hilfreich angesehen. Probleme gab es hauptsächlich mit Elementen, die Beteiligung von Personen ausserhalb des Enwicklungsteams erfordern, wie dem On-Site Customer. Alle Befragten, auch in dem nur teilweise als erfolgreich bewerteten Projekt, wollen XP wieder benutzen.

Die durchweg positive Einstellung gegenüber XP, die in den Fragebogen zum Ausdruck kommt, wirft die Frage auf, ob nur Unternehmen und Personen, deren XP Projekt erfolgreich war, an der Studie teilgenommen und den Bogen ausgefüllt haben. Dies kann nicht abschliessend festgestellt werden. Es ist vorstellbar, dass auch gerade bei einem erfolglosen Projektverlauf der Wunsch besteht, an der Studie teilzunehmen, um öffentlich zu machen, dass XP als Methode versagt hat. (Da die Unternehmensdaten anonym bleiben, würde das erfolglose Projekt weder dem Unternehmen noch den Entwicklern schaden). Über zwei Drittel der Projekte wurden von dem Entwicklungsleiter oder einem XP Coach bewertet, also Personen, denen der Projekterfolg besonders wichtig ist. Bei diesen kann eher davon ausgegangen werden, dass sie XP als Methode gesamtheitlich und objektiver beurteilen als vielleicht Entwickler, bei denen eine „technische“ Begeisterung für XP zusammen mit dem Faktor, keine Verantwortung für den Gesamterfolg des Projekts zu haben, zu einer eventuell leicht einseitigen Bewertung führen könnte.

Andererseits ist gerade für den Personenkreis der Entwicklungsleiter und der Coaches, die positive Darstellung des Projektergebnisses Teil des normalen Geschäfts. So kann aus dem Wunsch nach einem positiven Ergebnis eine unterschwellige Überzeugung entstehen, die das Gesamtergebnis als positiver einschätzen lässt, als dies der Kunde sehen wurde. Es ist daher ein Defizit, in dieser Form der Studie nicht gleichzeitig auch die Einschätzung des Projekterfolgs durch den Kunden einzubeziehen. Deshalb muss das Ergebnis kritisch gesehen werden, da eine Erfolgsquote von über 90 % doch sehr hoch scheint. Die Fragebögen wurden, wie in Kapitel 4 ausgeführt, an Unternehmen, Universitäten und Personen verschickt, die im Web über ein XP Projekt berichten, einen Job im XP Bereich annehmen wollen, XP Entwicker suchen oder ein Paper zur XP 2001 eingereicht haben. Es ist hier vorstellbar, dass Unternehmen und Personen, die mit einem XP Projekt gescheitert

sind, nicht weiter an XP interessiert sind und demzufolge auch nicht im Web darüber berichten oder Entwickler für weitere XP Projekte suchen.

Aber: auch wenn es eine „Dunkelziffer“ von gescheiterten Projekten geben mag, zeigt die Anzahl von hier 44 sehr erfolgreichen sowie einem teilweise erfolgreichen XP Projekt aus allen Branchen und mit verschiedensten Arten dabei entwickelter Systeme, dass XP als Softwareentwicklungsmethode gewinnbringend eingesetzt werden kann.

5.1 Kernaussagen der Studie

Bei der Auswertung der Studie fiel folgendes als besonders signifikant auf:

- Über 90% der Projekte endeten erfolgreich.
- 100% wollen XP wieder benutzen. (*„At the moment, I don't ever want to do anything else.[than XP]“*)[4]
- 40% benutzten das Element Metaphor gar nicht.
- Das häufige Fehlen eines On-Site Customers wurde von fast 30% der befragten Unternehmen als Risiko für den Projekterfolg angesehen.
- Der Einsatz von Testing und Pair Programming wurde in 18% der Fälle als starker Faktor für den Projekterfolg angegeben.
- Probleme mit XP rührten meist von “Barrieren in den Köpfen“ - das Management war skeptisch, die Unternehmensphilosphie der Kundenfirma erlaubte es nicht, einen Mitarbeiter als On-Site Customer abzustellen, Entwickler hatten Vorbehalte gegenüber Pair Programming.

5.2 Die Unternehmen

Von den 45[5] XP Projekten, die an der Studie teilnahmen, waren 43 Projekte kommerzielle Projekte, die in Unternehmen durchgeführt wurden, 2 waren private Hobbyprojekte.

Wie Abbildung 14 zeigt, wurden 25% der teilnehmenden Projekte in den USA durchgeführt, 20% in Deutschland. Jeweils gut 13% entfielen auf die Schweiz und Grossbrittanien. Frankreich, Italien, Belgien, die Niederlande, Kanada, Indien und Australien waren jeweils nur mit ein oder zwei Projekten vertreten.

[4] Alle Zitate in den folgenden Abschnitten stammen aus den von den Unternehmen ausgefüllten Fragebögen.

[5] Die beiden zu spät eingegangenen Fragebögen wurde nicht mehr mit ausgezählt.

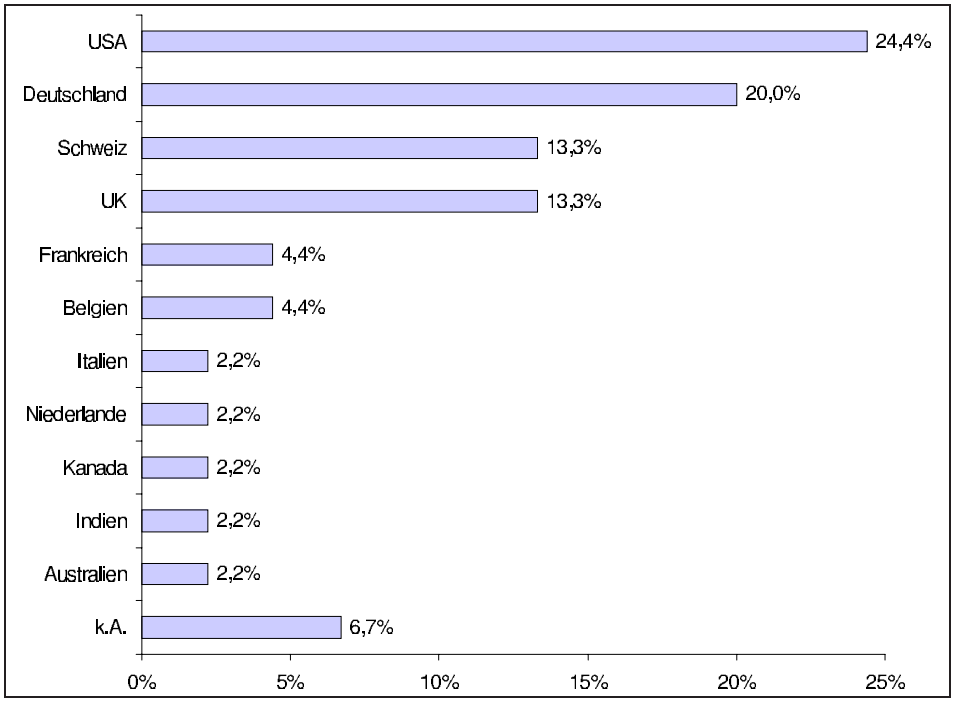


Abbildung 14: Länderverteilung (n=45)

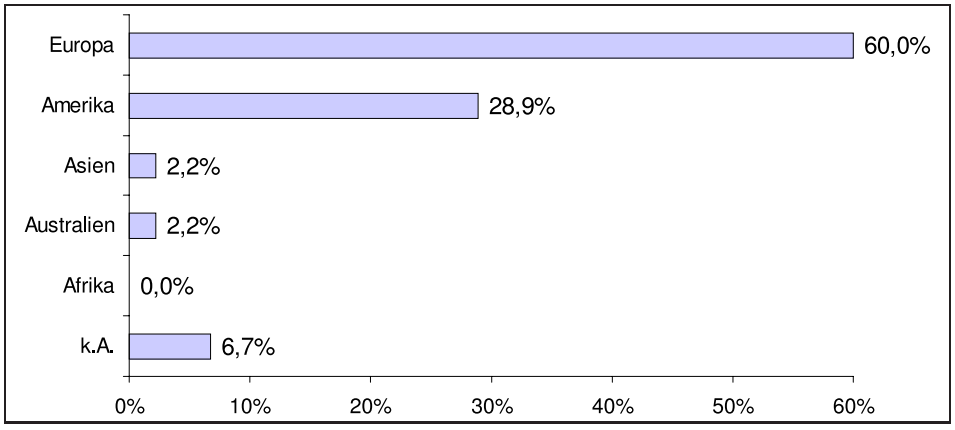


Abbildung 15: Verteilung auf Kontinente (n=45)

Knapp zwei Drittel der Projekte wurden also im europäischen Raum durchgeführt, knapp ein Drittel in Nordamerika (Abbildung 15). Asien und Australien sind mit jeweils einem Projekt vertreten, Afrika gar nicht. Was auffällt, ist die geringe Präsenz von XP Projekten im asiatischen Raum, in

dem doch viel Computerindustrie vertreten ist. Hier konnten bereits bei der Recherche so gut wie keine Firmen und Projekte gefunden werden, denen der Bogen geschickt werden konnte.

Abbildung 16 visualisiert die Verteilung auf Branchen. Hier liegt die ITK-Industrie klar vorn. Knapp zwei Drittel der Unternehmen stammen aus den Bereichen Softwareentwicklung, Internet, ITK sowie IT-Consulting. Doch auch klassische, eher konservative Branchen wie Banken und Versicherungen sind mit einem Anteil von über 7% vertreten, ebenfalls Biotechnologie mit knapp 5%. Hier sowie bei Grösse und Mitarbeiterzahl wurden Unternehmen, die mehrere Projekte beschrieben hatte, nur einmal gezählt, die beiden Hobbyprojekte gar nicht.

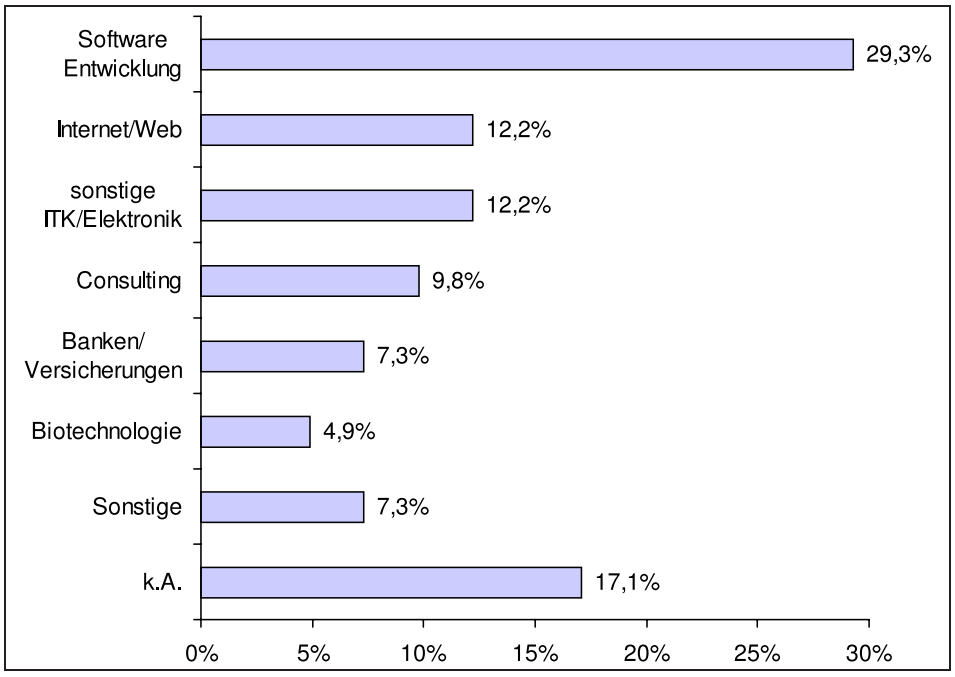


Abbildung 16: Branchen (n=41)

In Bezug auf Grösse und Alter der Unternehmen ist interessant, dass XP bereits Einzug in Firmen aller Grössen und Gründungsjahre gefunden hat (Abbildungen 18 und 17). Die Vermutung, dass XP hauptsächlich in kleinen und jungen Firmen angewendet wird, wurde nicht bestätigt. Knapp 37% der Projekte stammt zwar aus Firmen, die nicht älter als fünf Jahre sind. Über 14% der Projekte jedoch wurde in Firmen mit einem Alter zwischen 10 und 50 Jahren durchgeführt, weitere 12% in alteingesessenen Firmen, deren Gründung teilweise bis 150 Jahre zurückreicht. Old und New Economy sind

also gleichermassen vertreten.

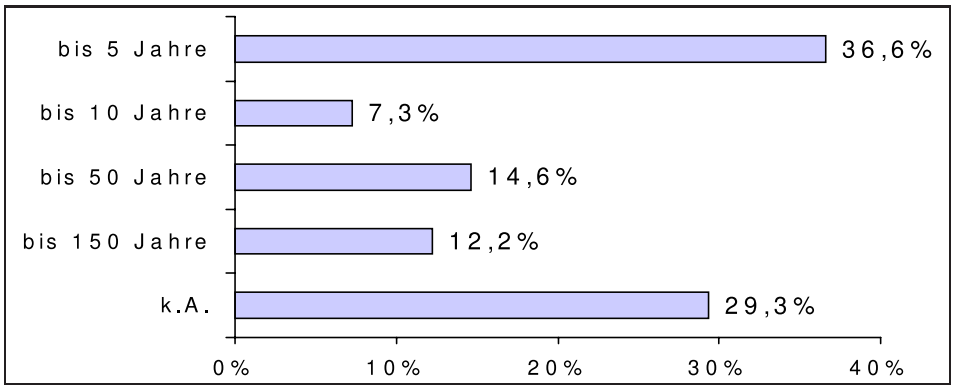


Abbildung 17: Alter der Firmen (n=41)

Bei der Mitarbeiterzahl findet sich eine Konzentration bei Firmen mit 10 bis 1000 Mitarbeitern. Über die Hälfte der Projekte wurden in Unternehmen dieser Grösse durchgeführt. Sehr kleine Firmen unter 10 Mitarbeitern sind mit einem Anteil von knapp 10%, sehr grosse Firmen zwischen 100.000 und 500.000 Mitarbeitern immerhin noch mit einem Anteil von knapp 5% vertreten.

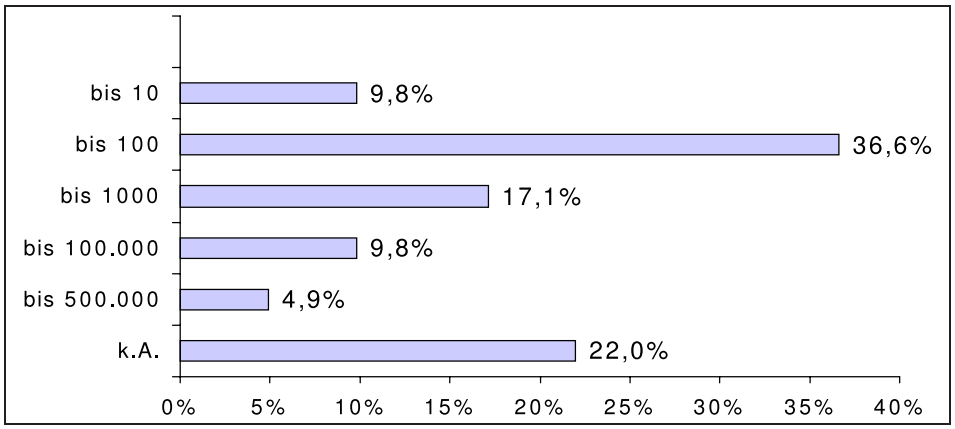


Abbildung 18: Grösse der Firmen (n=41)

Wie in Abbildung 19 zu sehen, wurden die XP Projekte meist aus der Sicht des Entwicklungs- oder Teamleiters beschrieben. Knapp 42% der Bögen wurden von diesen ausgefüllt. 25% der Projekte wurden aus der Sicht des XP Coach gesehen, gut 18% aus der Sicht eines Entwicklers. Die Hobbyprojekte wurden hier nicht mitgezählt.

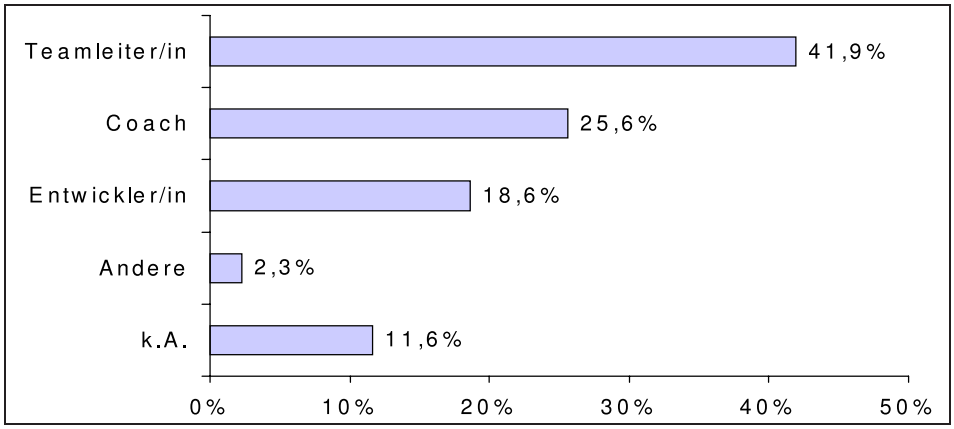


Abbildung 19: Rolle der ausfüllenden Person (n=43)

Zusammenfassend kann gesagt werden, dass XP bereits einen breiten Einzug in Unternehmen aller wichtiger Branchen, Grössen und Alters gehalten hat. Es wird sowohl in reinen "Softwarehäusern" eingesetzt als auch in Unternehmen, deren Kerngeschäft nicht die Softwareentwicklung ist. Findet sich momentan noch eine gewisse Konzentration auf mittelgrosse Firmen aus dem ITK-Bereich, so ist zu erwarten, dass es sich auch in anderen Branchen weiter ausbreiten wird. Leicht auffällig ist die Konzentration auf Europa und Nordamerika bzw. das Fehlen von Projekten aus dem asiatischen Raum.

5.3 Die Vorerfahrung

Abbildung 20 zeigt, dass das beschriebene Projekt für 51% der Unternehmen das erste XP Projekt war. In gut 18% der Unternehmen wurden bereits vorher XP Projekte durchgeführt (bis zu fünf Projekten).

Ein Unternehmen, eine US-amerikanische Bank, fällt hier besonders auf, da hier bereits ca. 20 Projekte mit XP durchgeführt wurden. 30% machten keine Angabe, ob es sich um das erste XP Projekt handelt oder nicht. Die Hobbyprojekte wurden nicht mitgewertet.

Die Teamzusammensetzung bei den teilnehmenden Projekten war typischerweise die, dass alle Teammitglieder hervorragende Kentnisse im Softwareengineering hatten (33% der Projekte) oder aber eine gemischte Struktur (42%), wobei meist ein bis zwei Teammitglieder sehr gute Kenntnisse hatten und der Rest oft Hochschulabsolventen oder Personen mit wenig Erfahrung waren. Die genaue Verteilung zeigt Abbildung 21.

Die Vorerfahrungen mit XP visualisiert Abbildung 22. In 49% der Projekte hatte niemand vorher Erfahrungen mit XP gemacht, in 20% der Projekte

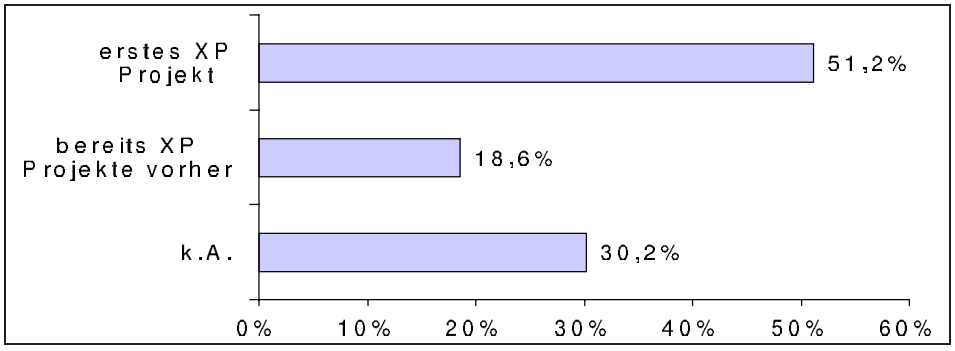


Abbildung 20: Vorherige XP Projekte im Unternehmen (n=43)

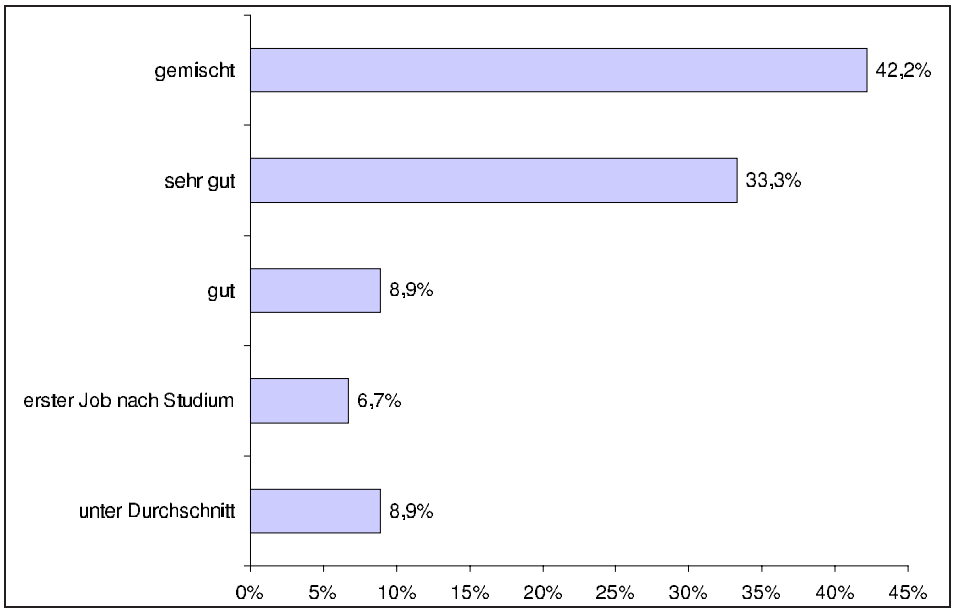


Abbildung 21: Allgemeine Erfahrung der Teammitglieder im Softwareengineering (n=45)

hatte nur eine Person im Team XP Erfahrung, in knapp 30% mehrere Teammitglieder. In einem Fall hatten alle Teammitglieder bereits XP-Erfahrung.

In über 50% der Projekte wurde kein Berater dazugezogen (Abbildung 23). 21% der Unternehmen zog einen Berater hinzu, gut 23% sogar mehrere. Knapp 5% machten keine Angabe zu dieser Frage, die Hobbyprojekte wurden nicht mitgewertet.

Interessant hier ist vor allem, dass es sowohl für die Hälfte der Unternehmen das erste XP Projekt war als auch in der Hälfte der Projekte noch

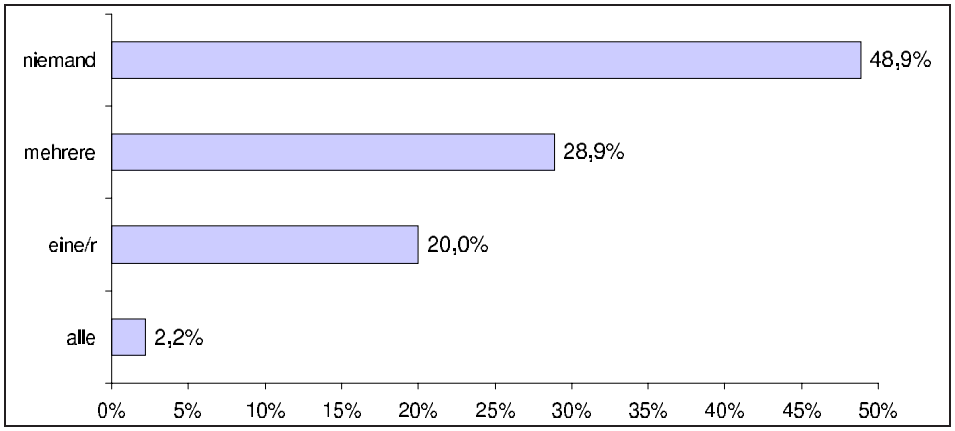


Abbildung 22: Wieviele Teammitglieder hatten Erfahrung in XP? (n=45)

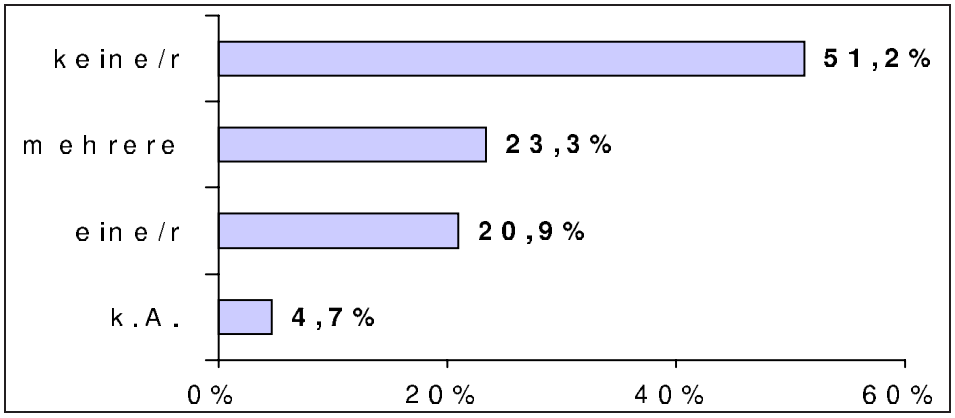


Abbildung 23: Beteiligung von Consultants/Development Companies (n=43)

niemand Erfahrungen mit XP hatte. Dies deutet darauf hin, dass viele Firmen und Entwickler, die bis jetzt Software mit klassischen Methoden entwickelt haben, nun anfangen, sich mit XP zu beschäftigen und dass XP dabei ist, sich auszubreiten und eventuell herkömmliche Softwareengineering-Methoden abzulösen. In mehreren der länger andauernden Projekte wurde dann auch angegeben, das Projekt wäre mit einer „traditionellen“ Methode angefangen worden und später wäre ein Umstieg auf XP erfolgt:

- *„The project itself started about two years ago using a standard development methodology. The decision to transition to XP was taken because of all the usual difficulties of managing development projects.“*

5.4 Die Projekte

Abgeschlossene und noch laufende Projekte halten sich knapp die Waage, wie in Abbildung 24 zu sehen.

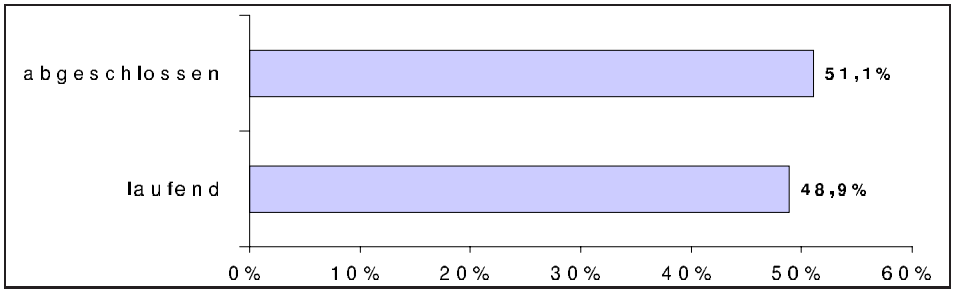


Abbildung 24: Verteilung laufende - abgeschlossene Projekte (n=45)

Hier ist für die Auswertung noch folgender Fragen zu bemerken, dass bei den noch laufenden Projekten meist bereits Releases fertiggestellt und im Einsatz befindlich waren.

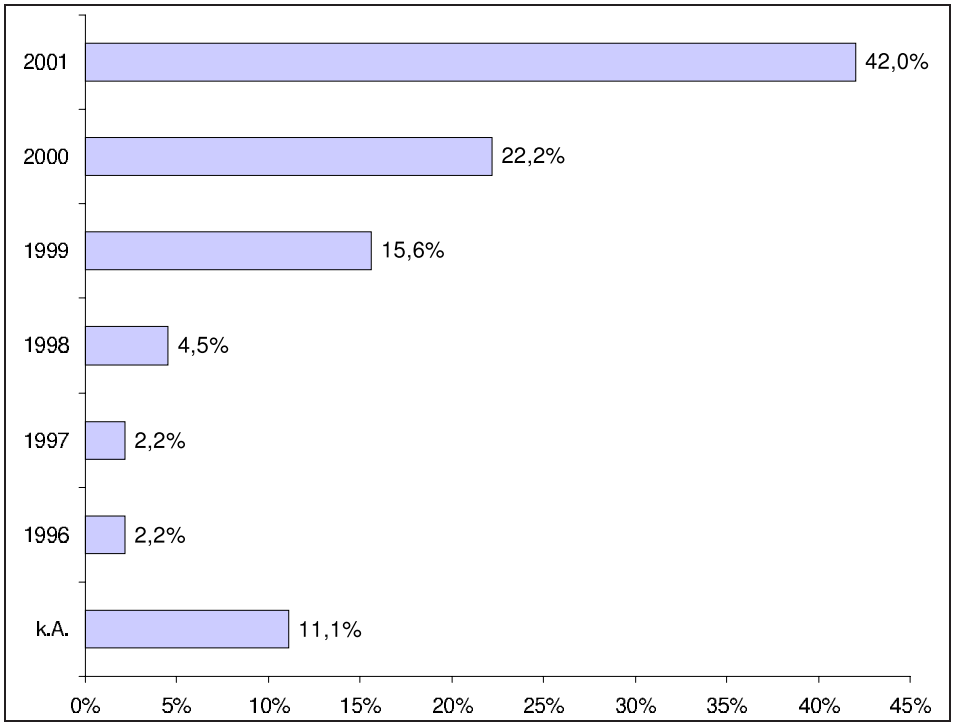


Abbildung 25: Start der Projekte (n=45)

Die abgeschlossenen Projekte wurde in den meisten Fällen (knapp 70%) erst vor kurzem abgeschlossen; knapp die Hälfte wurde hierbei in 2001 fertig, ein gutes Fünftel in 2000, der Rest machte keine Angabe.

Wann die Projekte gestartet wurden, zeigt Abbildung 25. Hier ist eine klare und starke Zunahme in den letzten Jahren erkennbar. Wurde gerade jeweils eines der Projekte in 1996 und 1997 gestartet, erfolgte von 2000 auf 2001 fast eine Verdoppelung. Zu beachten hierbei ist jedoch, dass mit einer gewissen Wahrscheinlichkeit eventuell eher Unternehmen an der Studie teilnahmen, die sich gerade oder vor kurzer Zeit mit einem XP Projekt beschäftigt haben als Unternehmen, die vor längerer Zeit ein solches Projekt gemacht haben. Ein gewisser Trend zur stetigen Zunahme von XP Projekten in der letzten Zeit kann jedoch auch hier abgeleitet werden.

Die Dauer der Projekte verteilt sich eher gleichmässig auf längere und kürzere Projekte (Abbildung 26). 28% der Projekte sind eher kurz, Laufzeit bis zu einem halben Jahr. Jeweils 21% der Projekte laufen bis zu einem Jahr bzw. bis zu drei Jahren. 30% machten keine Angaben zur Projektlänge.

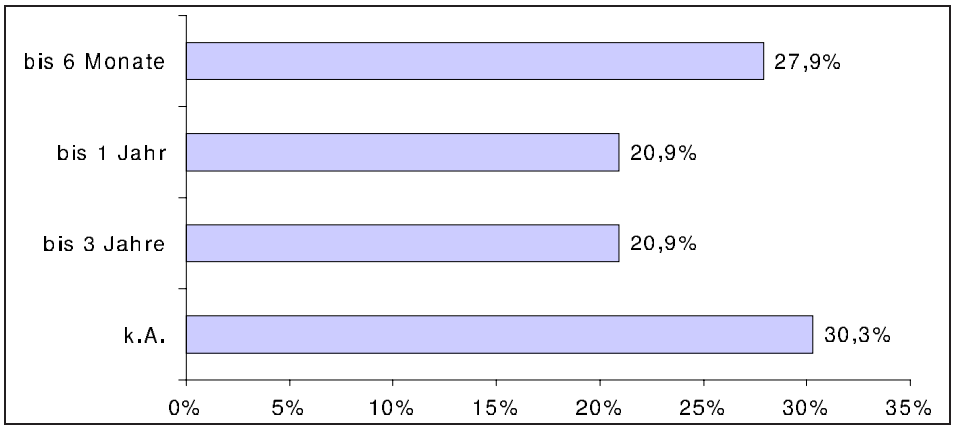


Abbildung 26: Projektdauer (n=45)

Abbildung 27 zeigt, dass die bei XP empfohlene Teamgrösse von zwei bis zehn Teammitgliedern meist eingehalten wurde. In knapp 36% der Projekte betrug die Teamgrösse bis zu fünf Personen, in 49% der Fälle lag sie zwischen fünf und zehn. In einem Unternehmen wurde ein XP Projekt nur von einer Person durchgeführt. Es gab jedoch auch zwei Projekte, die die empfohlene Teamgrösse stark überschritten und jeweils ca. 40 Mitglieder im Team hatten. Dabei handelte es sich jedoch in einem Fall nicht um ein „pures“ XP Projekt:

- *„It was an iterative approach using XP practices, but not explicitly XP.“*

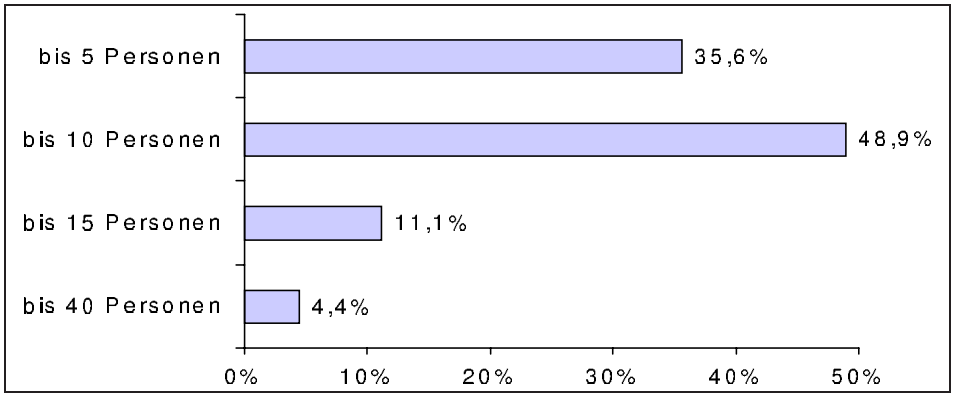


Abbildung 27: Teamgrösse (n=45)

Was für Software in den jeweiligen XP Projekten entwickelt wurde, wird in Abbildung 28 ersichtlich. Bei 29% der mit XP entwickelten Software handelt es sich um Internet-Software wie Web-Portale und E-Sales Lösungen. Weiterhin wurde Finanz- und Versicherungssoftware entwickelt sowie Tools und Frameworks (jeweils über 15%). In einem Fall wurde eine verteiltes soft real-time System zur Kontrolle und Überwachung von Zügen erstellt.

Über 73% der Systeme wurden komplett neu entwickelt, d.h. der gesamte Code war von Anfang an XP konform (Abbildung 29). In einem guten Viertel der Fälle wurde ein nicht mit XP entwickeltes, bestehendes System mit XP weiterentwickelt oder überholt. 9% waren neu entwickelte Systeme , die mit einem Altsystem interagieren. Bei dieser Frage gab es in einigen Fällen Mehrfachnennungen, die angaben, in der Entwicklung ihres Systems würden alle Antwortmöglichkeiten vorkommen.

Die vorherrschende Sprache der XP Projekte ist Java (Abbildung 30). 73% der XP Projekte wurden in Java geschrieben. 18% entfallen auf C++, 10% auf Smalltalk, die Sprache, auf die XP bei seiner „Erfindung“ zugeschnitten wurde. Ein Projekt wurde in Common Lisp geschrieben. Bei den restlichen 11%, die unter „Andere“ zusammengefasst sind, ist in allen Fällen auch jeweils Java mitangegeben. XP kommt beim Einsatz von objektorientierte Sprachen zu seiner grössten Effizienz; dies wurde in den Projekten beachtet. Bei dieser Frage waren ebenfalls Mehrfachnennungen möglich.

Bei den verwendeten Technologien herrschten XML/ HTML/ WML sowie die Verwendung von relationalen Datenbanken und SQL vor (Abbildung 31). Weiterhin wurden JSP/ ASP, COM/ CORBA und EJB Technologie verwendet.

Auch bei diesen Fragen sind deutliche Trends erkennbar. XP wird in kürzeren ebenso wie in längeren Projekten eingesetzt, die Teamgrösse ist

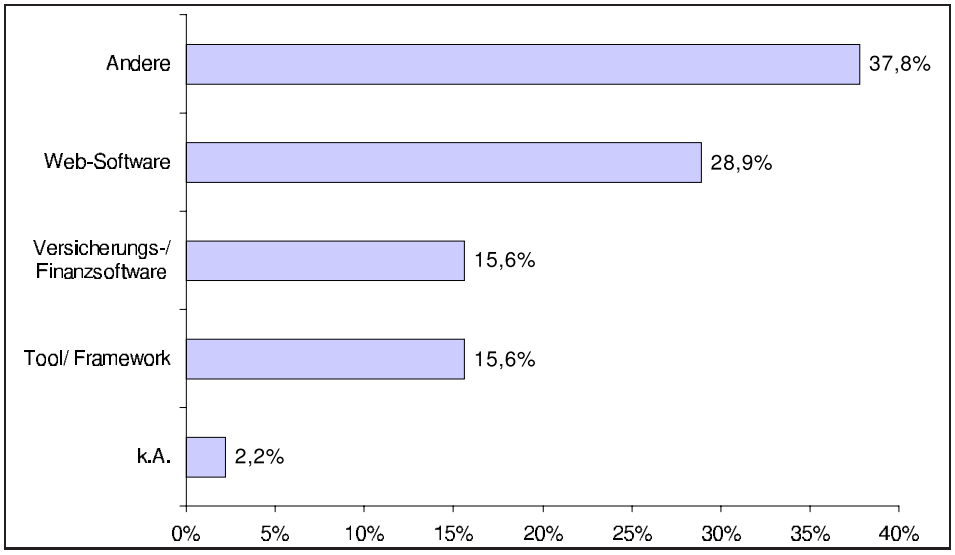


Abbildung 28: Art der entwickelten Software (n=45)

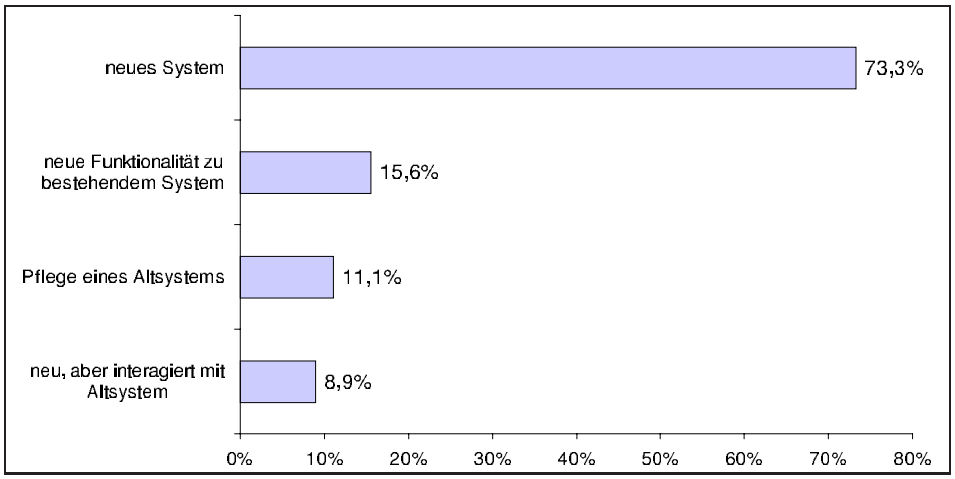


Abbildung 29: War es ein neues Projekt oder war alter Code involviert? (n=45, Mehrfachnennungen möglich)

meist eher klein, nicht mehr als zehn Leute, wie empfohlen für XP Projekte. Die fast immer in objektorientierten Sprachen geschriebenen Systeme reichen von einfachen Webseiten bis zu kritischen Systemen wie Finanzsoftware. Wenn XP benutzt wird, wird das Projekt meist von Anfang an mit XP entwickelt, aber es gibt auch Projekte, die unter Verwendung herkömmlicher Methoden gestartet und mit XP weiterentwickelt wurden. Mehrere

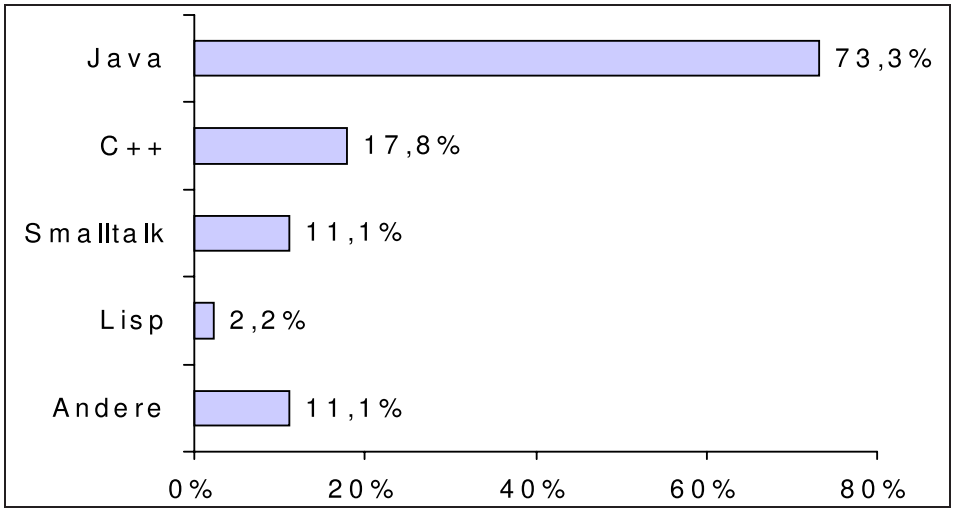


Abbildung 30: Benutzte Programmiersprache (n=45, Mehrfachnennungen möglich)

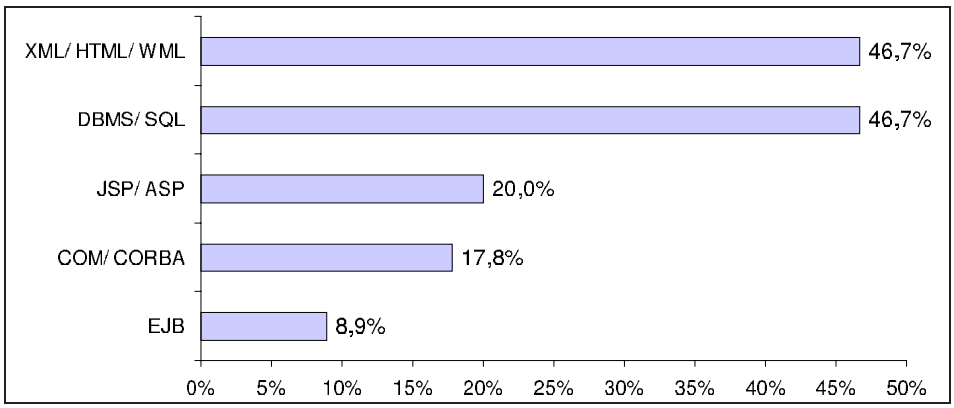


Abbildung 31: Benutzte Technologien (n=45, Mehrfachnennungen möglich)

Tendenzen deuten darauf hin, dass XP momentan dabei ist, sich stärker auszubreiten.

- *„As far as I know we are the only XP team in the company. However, there has been a gradual increase in interest in the rest of the company.“*

5.5 Die Entscheidung für XP

Warum wurde XP in den Projekten verwendet? In gut der Hälfte der Fälle handelte es sich um das erste XP Projekt im Unternehmen, wie also kamen

diese Unternehmen weg von ihrer herkömmlichen Entwicklungsart und hin zu XP? Abbildung 32 gibt Aufschluss über einige häufig genannte Beweggründe. Auch hier waren Mehrfachnennungen möglich.

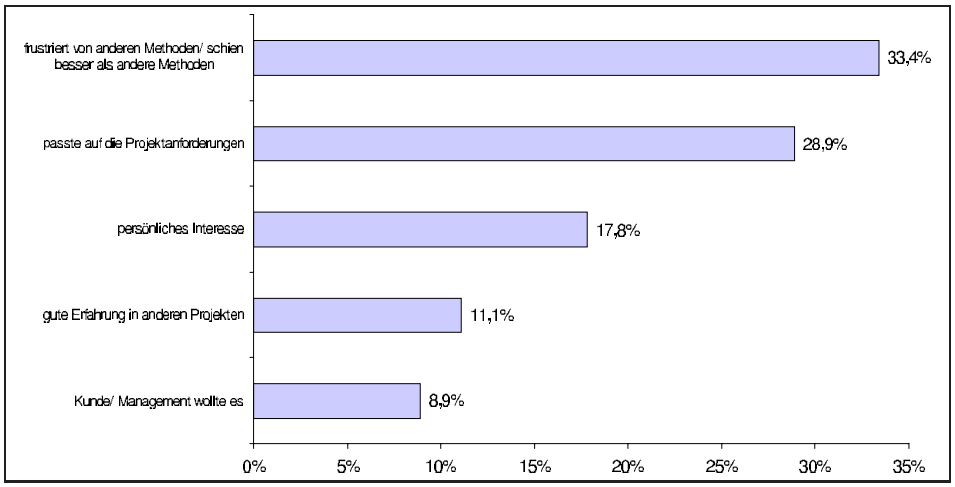


Abbildung 32: Warum wurde für XP entschieden? (n=45, Mehrfachnennungen möglich)

In über 33% der Fälle schien XP die sinnvollste Methode zu sein. Hier wurde vor allem nach einer einfachen, leichtgewichtigen Methode verlangt:

- *„Basically, after reading and thinking and talking about it a lot, I thought that it made more sense then any other methodology I'd read about. I didn't agree with all of it but I decided we should give it a try...“*
- *„Need of a light weight methodology.“*
- *„We felt that the XP is simple & better Process.“*

Mehrere waren konkret frustriert von herkömmlichen Methoden wie Waterfall oder CMM; vorher angewandte Techniken hatten zu unbefriedigenden Ergebnissen geführt:

- *„The project commenced in March 2000 using CMM Level 5 outsourced developers using Unified method. Code deliverred unsattisfactory. Development brought in-house February 2001, and project re started.“*
- *„Seemed a good way to get it working (we tried the same project with a waterfall-like method first)“*

- *„Need to fix a development process that did not work was costing allot of money.“*
- *„Frustrated with heavy CMM methodology, found XP info on the web“*
- *„Because we felt that other processes are too complex.“*
- *„Development process of first phase was not sufficient and lead to poor quality results. XP seemed a perfect fit.“*
- *„Because nothing else really had worked.“*

In 29% der Fälle war das Projekt ein “typisches XP Projekt“, die Ziele von XP passten auf die Anforderungen, die das Projekt stellte:

- *„...the project itself matches closely to the typical XP project: high time pressure, close team cooperation, changing and unclear requirements“*
- *„Besides the fact that we believe that XP is the approach for a team of this size, the requirements for the system to be developed were undefined at the beginning and rapidly changing due to developments in the marketplace.“*
- *„Changing Requirements, no clear design objective, because we were in the process of getting paying customers, and wanted to modify our product according to the wishes of (future) customers.“*
- *„Reduce costs and time to market.“*
- *„Needed results fast, but with high quality.“*

In 18% der Fälle spielte persönliches Interesse am XP Prozess eine Rolle bei der Entscheidung:

- *„Personal conviction“*
- *„Read about it, liked it, tried it.“*
- *„Because XP is convincing.“*

Hierbei gefällt die Flexibilität an XP; vielleicht holt es auch „Hacker“ da ab, wo sie stehen:

- *„We choose to use no methodology, and XP is a good methodology when you're not using a methodology! I.e., the good thing about XP is that you can pick and choose what you want and even extend later on.“*

- *„XP was close to our own development process but had some additional interesting techniques like PP etc.“*

11% hatten bereits konkrete, gute Erfahrungen mit XP in anderen Projekten gemacht.

- *„XP proved successfull in some of our other projects.“*

In einigen Fällen wurde XP direkt vom Kunden oder vom Management nachgefragt; dass dies im Moment noch eher ungewöhnlich ist und aus diesen Richtungen derzeit eher Widerstand gegen XP kommt, wird später noch ausgeführt.

- *„...initial management enthusiasm.“*

XP wurde in den meisten Fällen eingesetzt, da es als eine passende, erfolgversprechende und persönlich überzeugende Methode angesehen wurde. Oft wurden zusätzlich schlechte Erfahrungen mit anderen Methoden gemacht, die bekannten Probleme beim Softwareengineering waren aufgetreten und hatten Projekterfolg verhindert. XP als ein neuer, leichtgewichtiger Ansatz erschien vielversprechend, eine angenehmere und zielführendere Softwareentwicklung zu ermöglichen.

5.6 XP im Projekt

Wie wurde XP während des Projekts konkret umgesetzt? Wie war die Projektstruktur, wieviele Kunden waren beteiligt, in welcher Form wurden die Elemente von XP benutzt?

An 56% der Projekte waren mehrere Kunden(-gruppen) beteiligt (Abbildung 33). Über 25% der Projekte hatten einen beteiligten Kunden, nur bei knapp 5 % war gar kein Kunde beteiligt. 14% machten zu der Frage keine Angabe. Zu beachten ist hier, dass sich bei diesen beteiligten Kunden nicht immer um On-Site Kunden handelte. Die Hobbyprojekte wurden hier nicht mit gewertet.

Die typische Projektstruktur ist in Abbildung 34 zu sehen. In knapp 56% der Fälle war das Entwicklerteam klein, es bestand aus bis zu fünf Programmierern. In 29% der Fälle waren es zwischen fünf und zehn Entwicklern. In über 80% der Fälle waren ein oder mehrere Kunden beteiligt. Diese waren allerdings meist nicht die ganze Zeit vor Ort, teilweise handelte es sich auch um “Customer Substitutes“, deren Rolle von dem Projektmanager, Personen aus dem Sales Bereich oder den Programmierern selbst übernommen wurde. Näheres dazu wird weiter unten bei der Diskussion der Nutzung des Elements On-Site Customer ausgeführt.

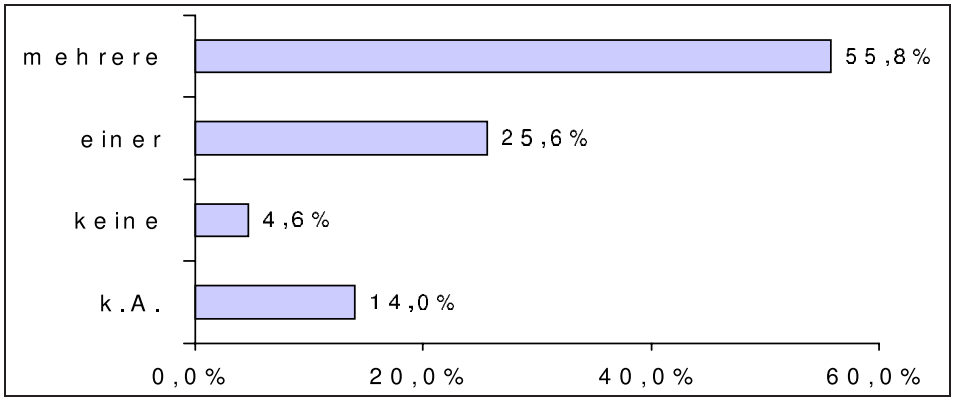


Abbildung 33: Involvierte Kunden(-gruppen) (n=43)

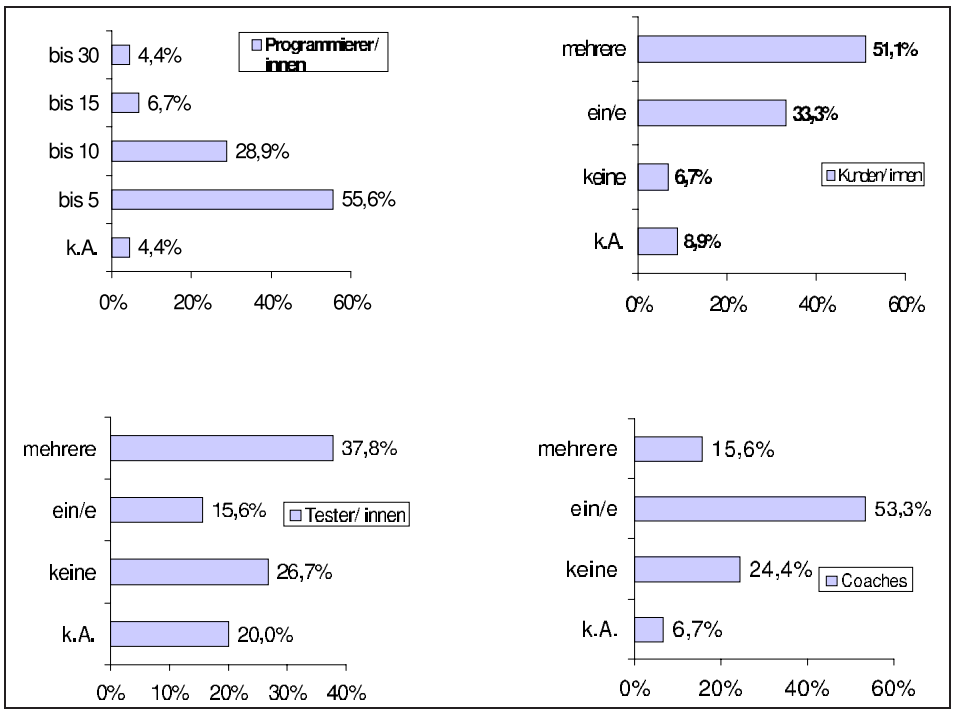


Abbildung 34: Projektstruktur (n=45)

- *„As we are the main users of our system, we played that role ourselves.“*

Die Tester waren ebenfalls öfter die Programmierer selbst (gemeint waren hier Tester, die dem Kunden helfen, Akzeptanztests zu schreiben), teilweise waren es auch explizite Tester:

- *„Our customer provided us with a tester.“*

53% der Projekte hatte einen Coach, gut 15% sogar mehrere; bei knapp 25% der Projekte war kein Coach dabei. In über 60% der Fälle gab es noch eine oder mehrere weitere Rollen, wie Projektmanager, oder Termintracker.

Die folgenden zwölf Abbildungen zeigen die Nutzung der XP Elemente im Projekt. Die Intensität der Nutzung war mit einem Wert zwischen 0 und 9 zu bemessen; ausserdem sollte angegeben werden, ob das Element als hilfreich (H), verbesserungswürdig (V) oder erschwerend (E) für die Entwicklung empfunden wurde.

Bis auf die Metapher, On-Site Customer und teilweise Planning Game wurden alle Elemente stark benutzt und vorwiegend als hilfreich empfunden. Hierbei wird ein Wert zwischen 7 und 9 als starke Nutzung, ein Wert zwischen 3 und 6 als mittlere Nutzung und ein Wert zwischen 0 und 2 als geringe Nutzung angesehen. Zu den wenig benutzten Elementen und den Gründen dafür wird noch näheres in 5.7 ausgeführt, zu dem Einfluss, den die Nutzung der Elemente auf den Projekterfolg hatten, in 5.8. Es fällt hier auf, dass in jeweils 30 - 40% der Fälle leider keine Angaben erfolgten, ob ein Element als hilfreich, verbesserungswürdig oder erschwerend empfunden wurde; es ist davon auszugehen, dass diese Frage teilweise übersehen wurde, da sie mit der Frage nach der Intensität der Nutzung verknüpft war.

Das Planning Game wurde von 60% der Teams stark genutzt, gut 13% benutzten es allerdings gar nicht (Abbildung 35). Über 55% der Befragten empfand dieses Element als hilfreich, nur knapp 7% als verbesserungswürdig; knapp 40% machten keine Angabe. Das Problem hier war meist, dass das Planning Game aufgrund des nicht oder nicht häufig genug vorhandenen On-Site Customers nicht zufriedenstellend durchgeführt werden konnte; näheres wird weiter unten diskutiert.

Short Releases wurden in knapp 65% Projekte intensiv verwendet (Abbildung 36). 60% empfanden dieses Element als hilfreich; gut 35% machten keine Angabe.

- *„The development of the first version of the system was do be done in about 4 months. This was release one. Within that release the iterations were neglected. I would say that the first release of the system had only one iteration, which was a mistake.“*

Die Metapher war das am kritischten gesehene Element (Abbildung 37). 49% der befragten Teams sah sie als verbesserungswürdig an; eines empfand sie sogar als erschwerend. 40% benutzten sie infolgedessen auch überhaupt nicht. Das Problem mit der Metapher bestand vorwiegend darin, dass vielen

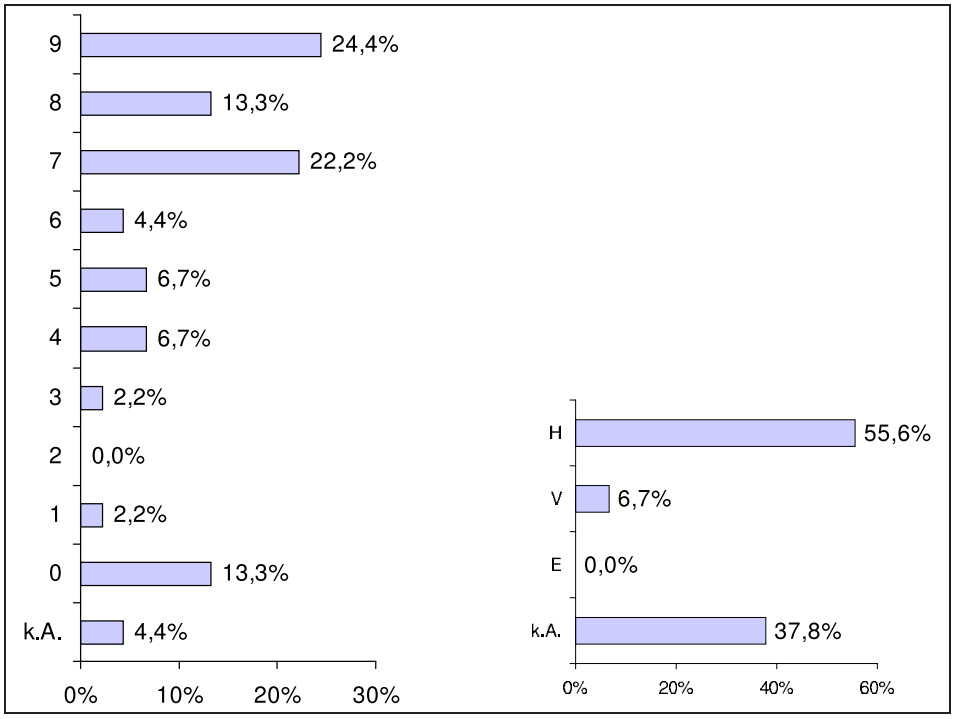


Abbildung 35: Nutzung des Planning Game (n=45)

nicht klar war, wie sie anzuwenden sei und was für einen Vorteil sie bringen solle. Auch dies wird weiter unten noch genauer diskutiert.

Simple Design wurde in 60% der Fälle stark genutzt, in gut 35% der Fälle erfolgte eine mittlere Nutzung (Abbildung 38). Knapp 60% sahen auch dieses Element als hilfreich an, 35% machten keine Angabe hierzu.

Testing war sehr beliebt (Abbildung 39). 80% nutzten dieses Element intensiv, knapp 18% nutzten es mittelstark. 60% empfanden es als hilfreich, gut 30% machten keine Angabe. Wie später noch ausgeführt wird, wurde es häufig als entscheidender Faktor für den Projekterfolg angesehen.

- *„I will never ever develop software without Test First and Automatic Testing anymore.“*

Auch Refactoring wurde gerne gemacht (Abbildung 40). Knapp 70% nutzten dieses Element stark, knapp 30% nutzten es mittelstark. Über die Hälfte empfand es als hilfreich, gut 10% sahen es allerdings als verbesserungswürdig an. Über 33% machten keine Angabe.

Ein weiterer „Renner“ war das Pair Programming (Abbildung 41). Über 75% der Teams programmierten zu zweit. Knapp 10% benutzten es allerdings

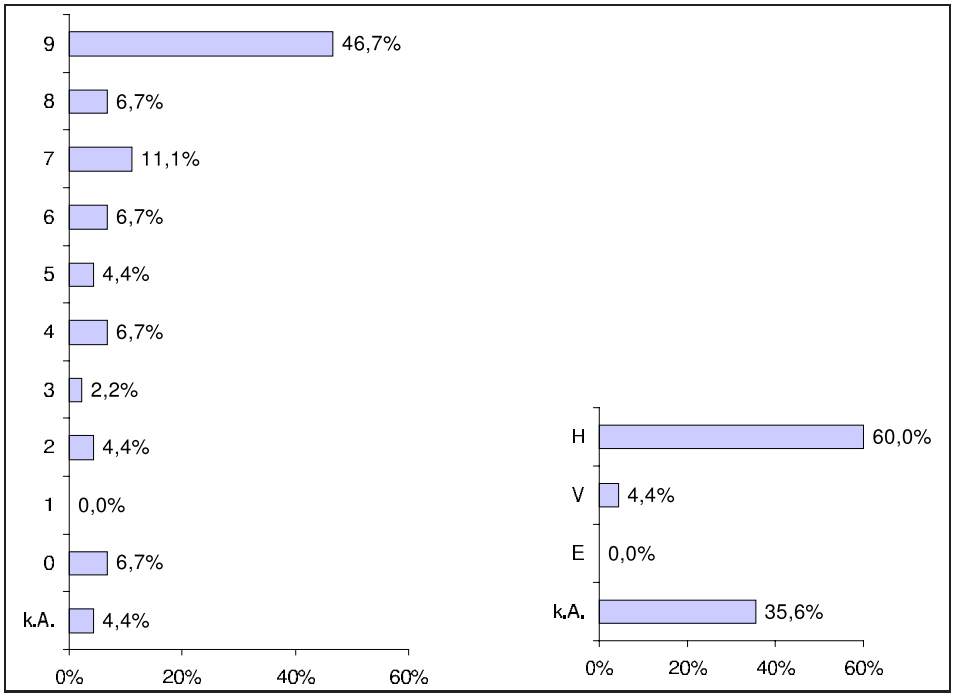


Abbildung 36: Nutzung der Short Releases (n=45)

kaum, bei gut 13% erfolgte eine mittlere Nutzung. Über 53% empfanden es als hilfreich, knapp 10% als verbesserungswürdig. Fast 40% machten keine Angabe. Auch Pair Programming wurde häufig als entscheidender Erfolgsfaktor angesehen; allerdings hatten auch einige Teams Probleme damit, die unten noch genauer erläutert werden.

Das am stärksten genutzte Element war die Common Code Ownership (Abbildung 42). Über 80% nutzten diese, über 70% gaben hierbei sogar den Wert 9 - volle Nutzung - an. Knapp 60% empfanden die Common Code Ownership als hilfreich, 35% machten keine Angabe. Diese Werte erstaunen vielleicht am meisten, da viele Entwickler normalerweise grosse Bedenken haben, einen anderen Entwickler an ihrem Code arbeiten zu lassen; anscheinend sind die Vorteile der Common Code Ownership – jeder und jede hat einen Überblick über den gesamten Code und kann so auch leicht Fehler finden sowie Verbesserungen durchführen – gross, so dass dieses Element stark genutzt wird.

Auch die Continuous Integration wurde viel genutzt(Abbildung 43); über 73%der Teams gaben hier eine starke Nutzung an. 60% sahen das dauernde Integrieren als hilfreich an, knapp 36% machte wieder keine Angabe.

Sehr interessant auch die Nutzung der 40-Hour Week (Abbildung 44).

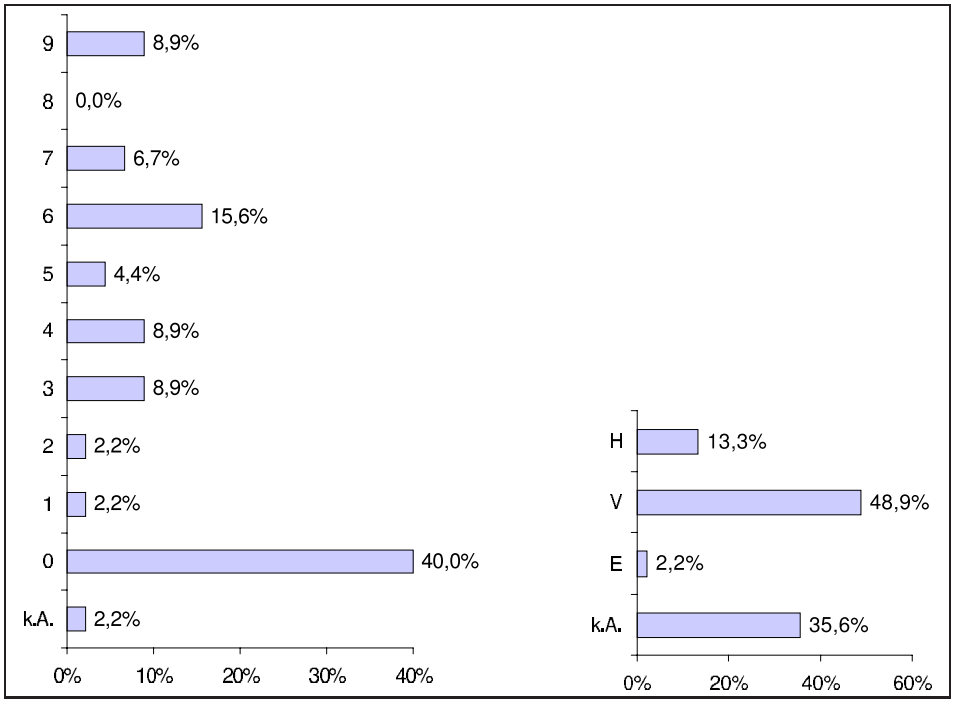


Abbildung 37: Nutzung der Metaphor (n=45)

Knapp 70% gaben an, im Projekt tatsächlich keine Überstunden zu machen. Knapp 10% nutzten dieses Element wenig, 13% mittel, knapp 18% sahen es als verbesserungswürdig an, wobei meistens nicht angegeben wurde, inwiefern die 40-Hour Week verebssert werden sollte. Lediglich einer gab an *„Also advocate the „chill-out“ area: in our case we've got a „Tischfussball.“*, ein anderes Mal wurden statt 40 Stunden nur 37.25 Stunden in der Woche gearbeitet. Demgegenüber sah knapp die Hälfte der Teams die 40-Hour Week als hilfreich an. Die doch starke Nutzung der 40-Hour Week ist ein erfreuliches Zeichen, da diesem Element nicht nur XP Kritiker, sondern auch viele weitere Angehörige der Softwareentwicklungs-Szene skeptisch gegenüberstehen. Gerade wenn Software termingerecht fertiggestellt werden muss, glauben viele, dass eine 40-Stunden Woche nicht mehr als eine schöne, aber unrealistische Wunschvorstellung ist. In Zusammenhang mit der Tatsache, dass die meisten der hier untersuchten XP Projekte zum vereinbarten Termin fertig wurden, wie weiter unten noch zu sehen sein wird, scheint es doch möglich zu sein, Software ohne dauernde Überstunden in der Zeit zu liefern.

Beim On-Site Customer verteilt sich die Nutzungsintensität (Abbildung 45). Knapp 30% benutzten dieses Element stark, 20 % gaben eine mittlere Nutzung an und über 40% benutzten dieses Element kaum. So empfanden

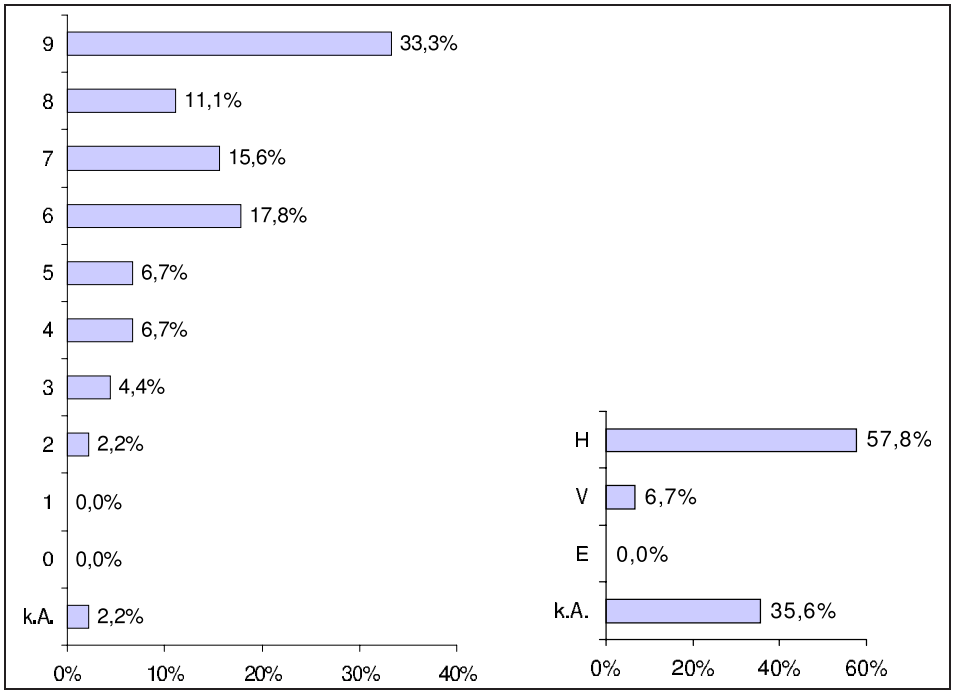


Abbildung 38: Nutzung von Simple Design (n=45)

es auch über 20% als verbesserungswürdig und 35% als hilfreich. Auch hier in knapp 40% der Fälle keine Angabe. Wie später noch zu sehen sein wird, bestand hier durchaus der Wunsch nach einer stärkeren Nutzung; das Problem mit diesem Element war meist das Nicht- Vorhandensein eines On-Site Customers.

Coding Standards schliesslich wurden in über 60% der Projekte stark benutzt, jedoch gaben auch knapp 30% nur eine mittlere Nutzung an und gut 10% empfanden dieses Element als verbesserungswürdig (Abbildung 46). Für knapp die Hälfte war es ein hilfreiches Element, 40% machten keine Angabe.

- *„I expect these [Coding Standards] to develop over time. And so they have not been explicit.“*

Wurden die Ziele einer erfolgreichen Softwareentwicklung durch XP und die Nutzung seiner Konzepte erreicht? Die Abbildungen 47 bis 50 geben hierüber Aufschluss. Die Auswertung dieser Frage unterscheidet zwischen abgeschlossenen und laufenden Projekten. Die laufenden Projekte wurden mit einbezogen, da auch in diesen Fällen oft Angaben zum (voraussichtlichen) Erreichen der Ziele gemacht wurden - das Erreichen eines Ziels wie

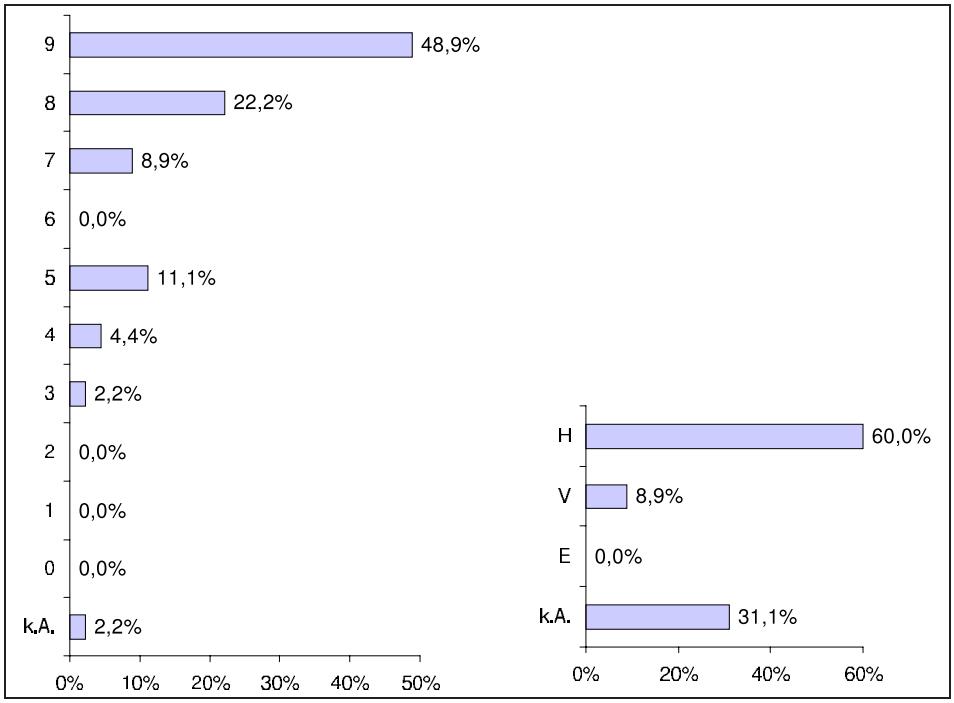


Abbildung 39: Nutzung von Testing (n=45)

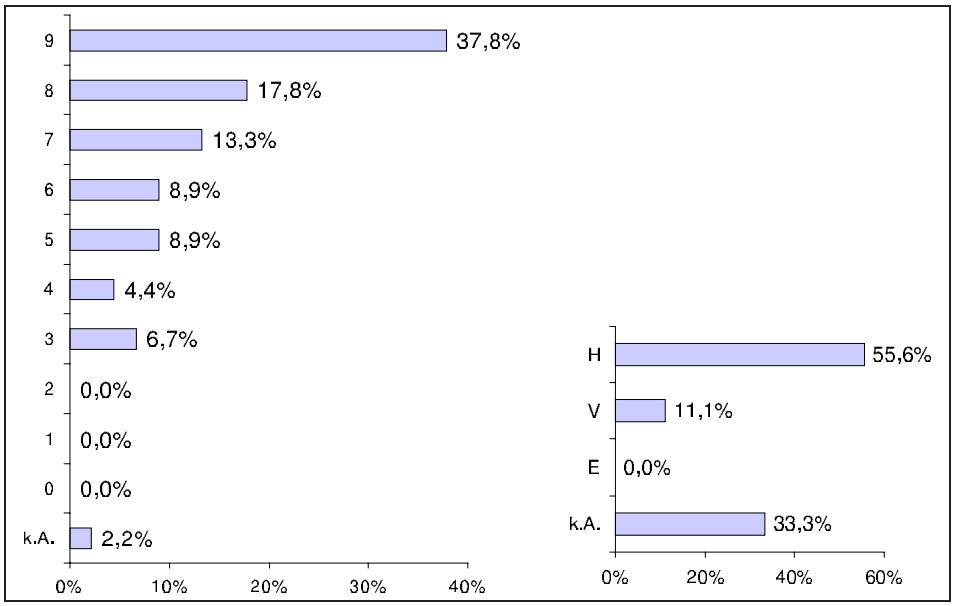


Abbildung 40: Nutzung von Refactoring (n=45)

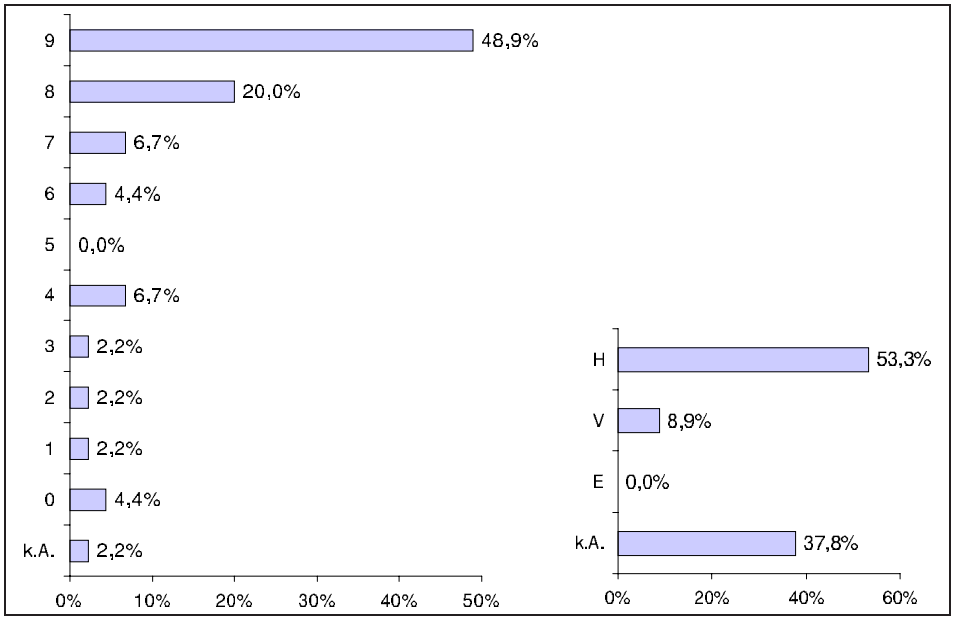


Abbildung 41: Nutzung von Pair Programming (n=45)

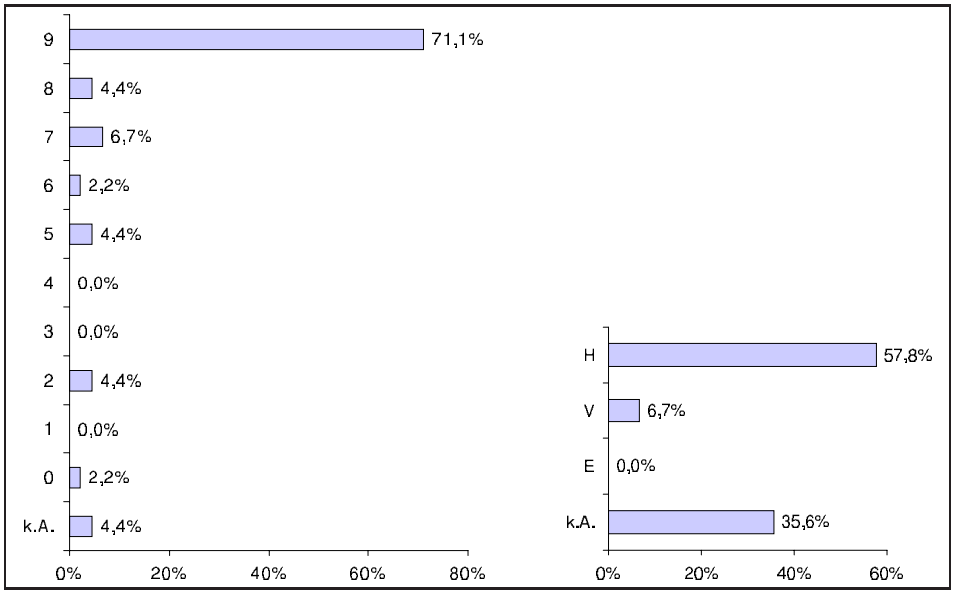


Abbildung 42: Nutzung der Common Code Ownership (n=45)

„Let developers have fun with their work“ kann bereits während des Projekts gesehen werden und anhand erster Iterationen kann auch schon abgeschätzt

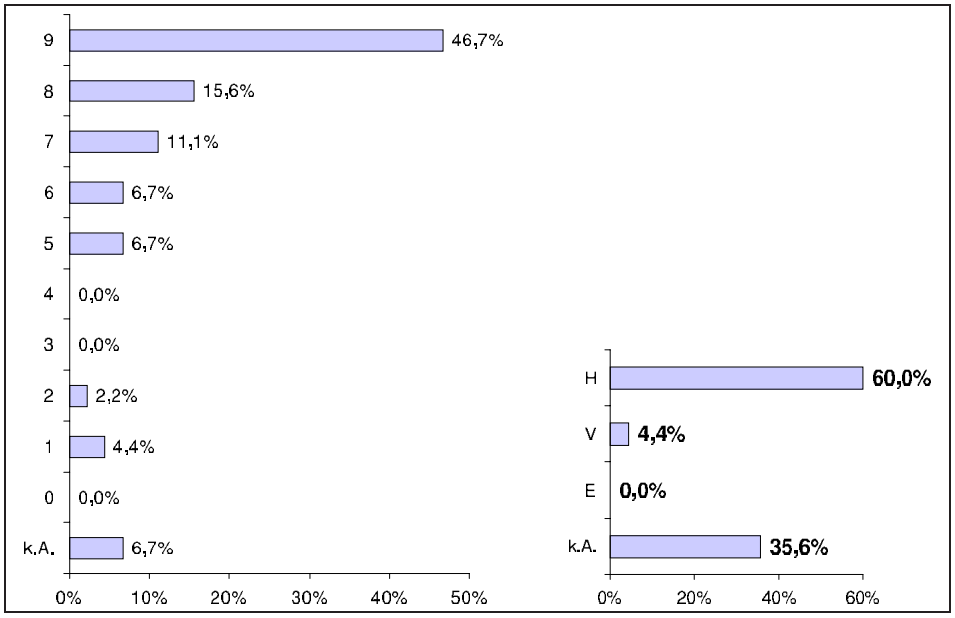


Abbildung 43: Nutzung von Continuous Integration (n=45)

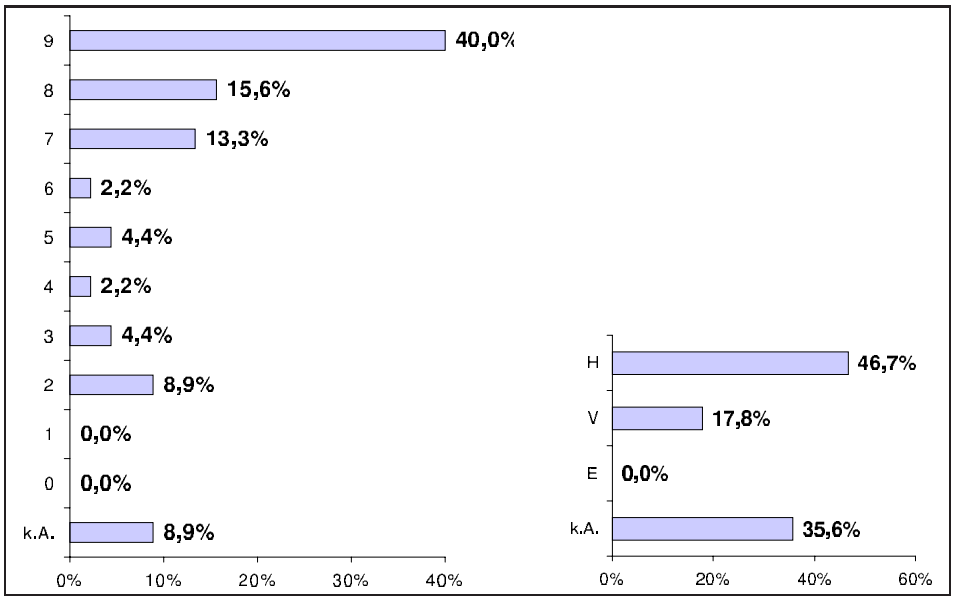


Abbildung 44: Nutzung der 40-Hour Week (n=45)

werden, inwieweit voraussichtlich eine termingerechte Fertigstellung möglich sein wird.

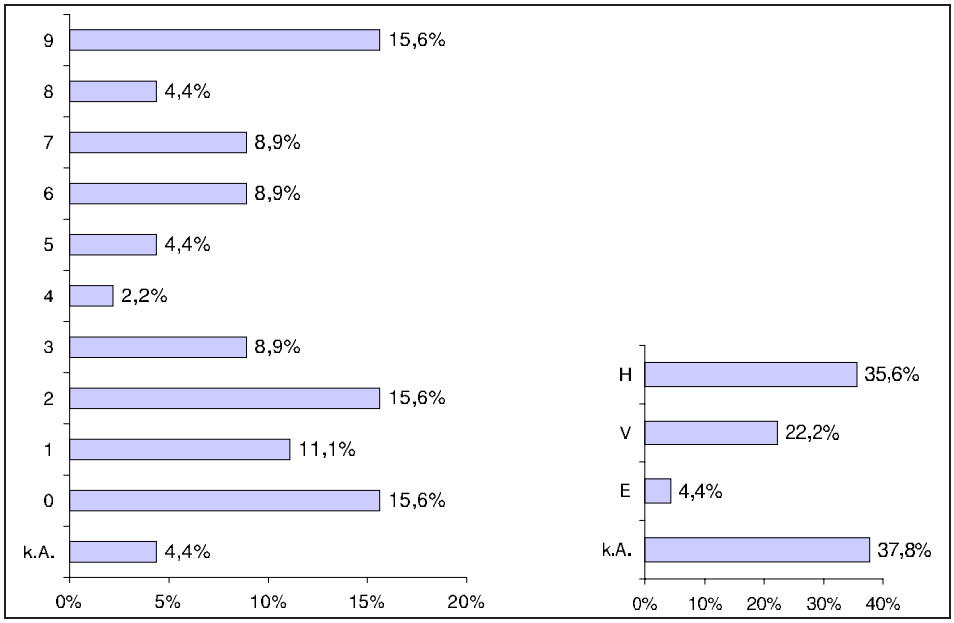


Abbildung 45: Nutzung des On-Site Customer (n=45)

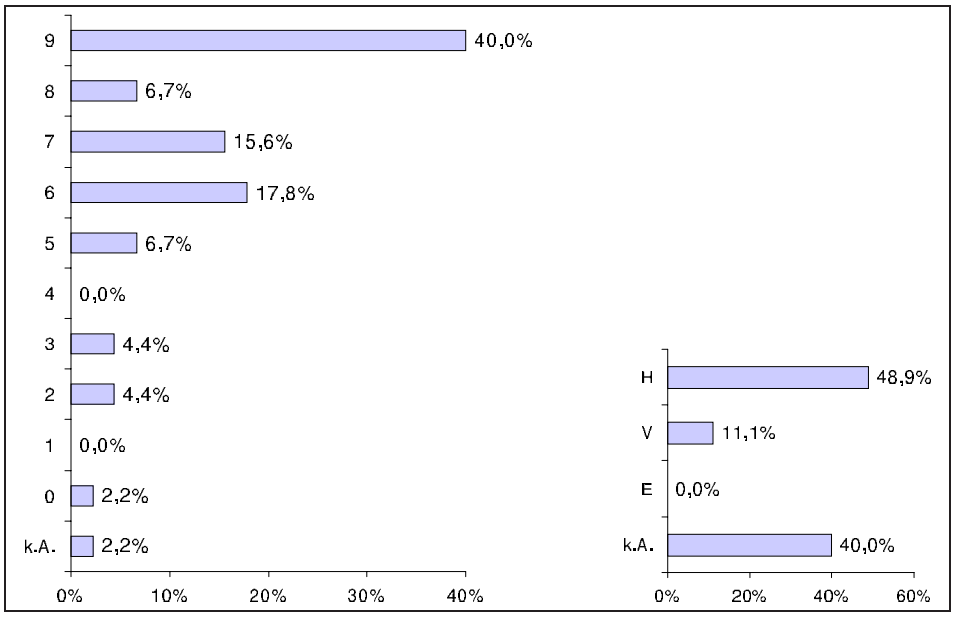


Abbildung 46: Nutzung von Coding Standards(n=45)

Der Wertebereich für die Erreichung der Ziele lief von -5 bis +5, wobei +5 bedeutete „Ziel voll erreicht“, 0 bedeutete „wie gewöhnlich erreicht“ und

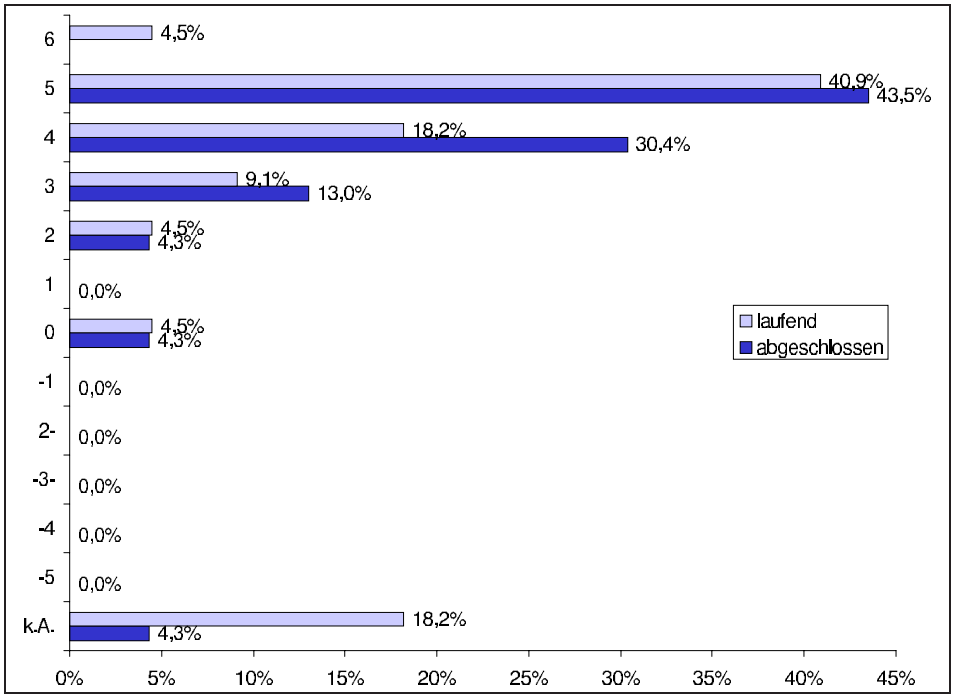


Abbildung 47: Termingerechte Fertigstellung der Software (n=45)

-5 bedeutete „schlechter als gewöhnlich erreicht“. Bemerkenswert ist, dass in keinem Fall eine Bewertung unter 0 vergeben wurde. In einem Fall wurde sogar für alle Ziele der Wert +6 vergeben, obwohl dieser gar nicht im Antwortwertebereich lag. Aufgrund dessen werden Werte unter 0 im folgenden ausser Betracht gelassen und folgende Gruppierung für die Interpretation vorgenommen: die Werte 5 und 4 werden als Ziel voll erreicht angesehen, 3 bis 0 werden als Ziel teilweise erreicht angesehen.

Die termingerechte Fertigstellung der Software, ein sehr heikles Thema bei jedem Softwareprojekt, war in den befragten XP Projekten sehr zufriedenstellend. Abbildung 47 veranschaulicht das eindrucksvoll. In knapp drei Viertel der abgeschlossenen Projekte konnte dieses Ziel voll erreicht werden. 17% hatten leichte Probleme mit der termingerechten Fertigstellung, bei einem Projekt klappte die Fertigstellung mit XP nicht schneller als gewöhnlich.

- *„The project is progressing fast.“*
- *„We completed the project in time with good quality.“*

Auch bei den laufenden Projekten wurde in über 60% der Fälle angegeben, die termingerechte Fertigstellung voll zu erreichen. Diese Annahmen stützten

sich meist auf die termingerechte Fertigstellung der ersten Releases. 18% machten keine Angaben. Auch hier gab es ein Projekt, das mit XP nicht schneller als gewöhnlich fertiggestellt werden konnte, wobei hier als Grund für die nicht termingerechte Fertigstellung nicht beeinflussbare Abhängigkeiten von Dritten angegeben wurden.

Für das Ziel, die Arbeitsqualität und den Arbeitszufriedenheit der Entwickler zu erhöhen, sah das Ergebnis etwas gemischter aus (Abbildung 48).

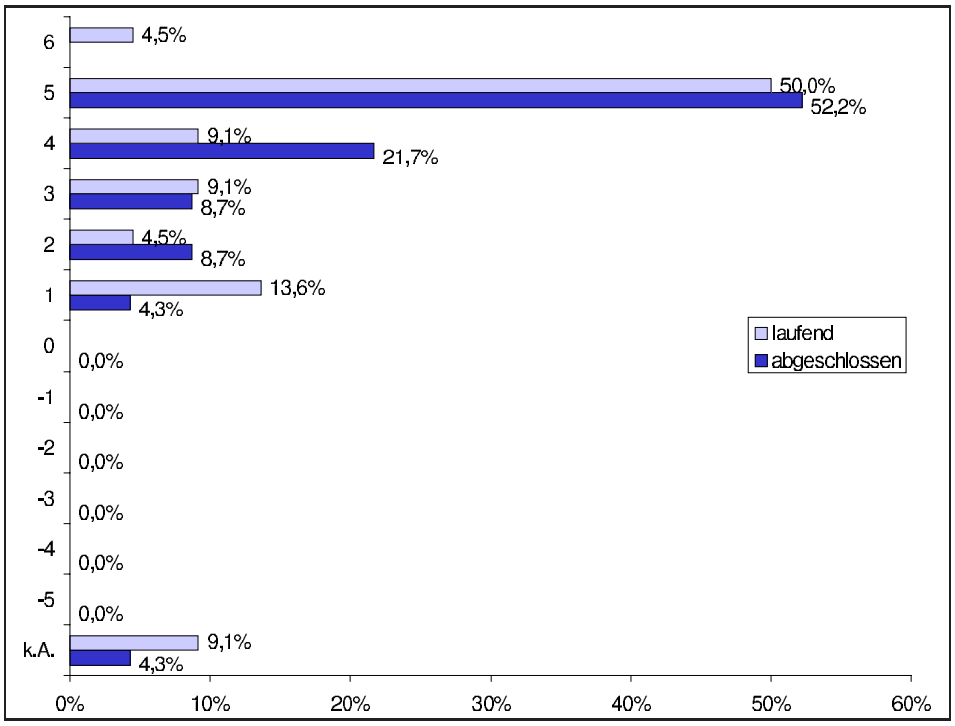


Abbildung 48: Arbeitsqualität, Spass (n=45)

Knapp drei Viertel in den abgeschlossenen Projekten gaben dieses Ziel als voll erreicht an. Gut 20% gaben jedoch keine deutlich bessere Arbeitsqualität als gewöhnlich an. Bei den laufenden Projekten ist dies noch signifikanter, hier waren es knapp 30%. Doch auch hier stehen demgegenüber 60%, die dieses Ziel als voll erreicht ansehen. Die Teilnehmer, die dieses Element schlechter bewertet hatten, gaben in einem Fall Probleme der Entwickler mit Pair Programming an, ein anderes Mal wurde angegeben, der geringe Spass hätte weniger mit XP als mit generellen Spannungen im Team zu tun.

- *„Extreme“ teambuilding power through xp. personal relationsships grew very quickly and still hold.“*

- *„It's fun, our code rocks and it's going faster than we thought."*

Das Ergebnis des Zieles „High Quality" sah ebenfalls leicht gemischt aus. (Abbildung 49)

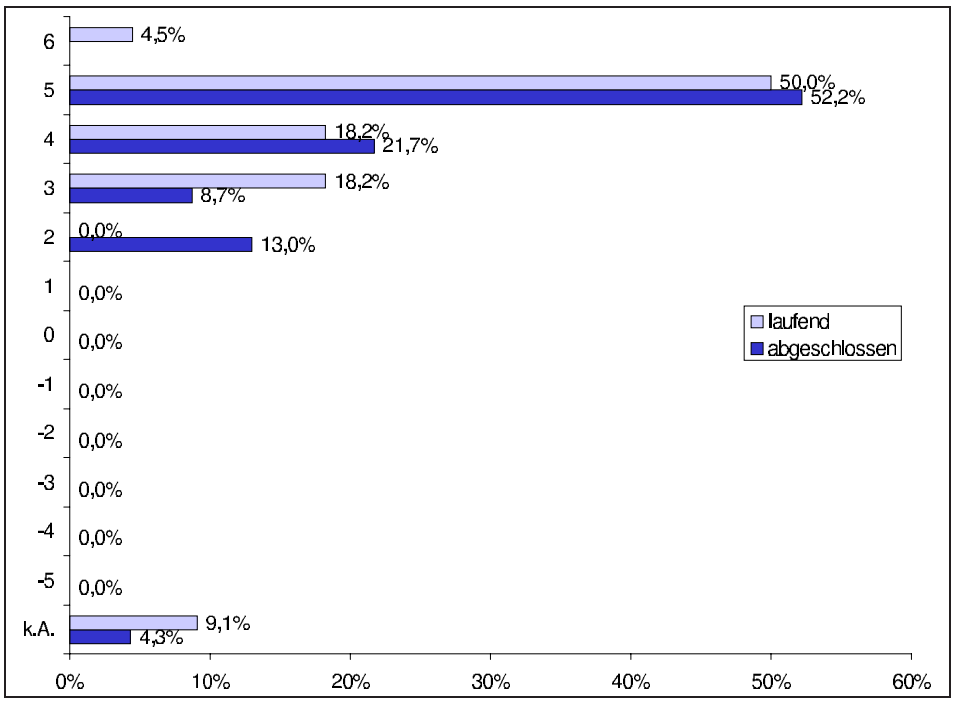


Abbildung 49: Hohe Qualität der Software (n=45)

An dieser Stelle muss jedoch nochmals betont werden, dass es hier nur um Werte zwischen 0 und +5 geht und die negativen Werte bis -5 in keinem Fall genannt wurden. Auch hier gaben knapp drei Viertel in den abgeschlossenen Projekten dieses Ziel als voll erreicht an. 20% waren jedoch nicht ganz zufrieden mit dem Ergebnis. Hier waren keine näheren Gründe angegeben. Bei den laufenden Projekten sah das Ergebnis ähnlich aus.

- *„External users have noted the high quality of the system."*
- *„The Client was delighted by quality..."*

Auch das Ziel, schnell auf Änderungen reagieren zu können, wurde bei den abgeschlossenen Projekten in 60% der Fälle als voll erreicht gesehen (Abbildung 50). Bei den laufenden Projekten waren es sogar über 75%. 30% der abgeschlossenen Projekte gaben das Ziel als nicht voll erreicht an, wobei

hierzu bemerkt werden, dass öfters angegeben wurde, es wären nicht viele Änderungen gefordert worden, so dass eventuell auch aus diesem Grund niedrigere Werte vergeben wurden. Eimal wurde als Grund für die Bewertung mit Wert 2 angegeben, der Kunde hätte sehr viele Anforderungen gehabt und wäre nicht bereit gewesen, irgendeine davon aufzugeben.

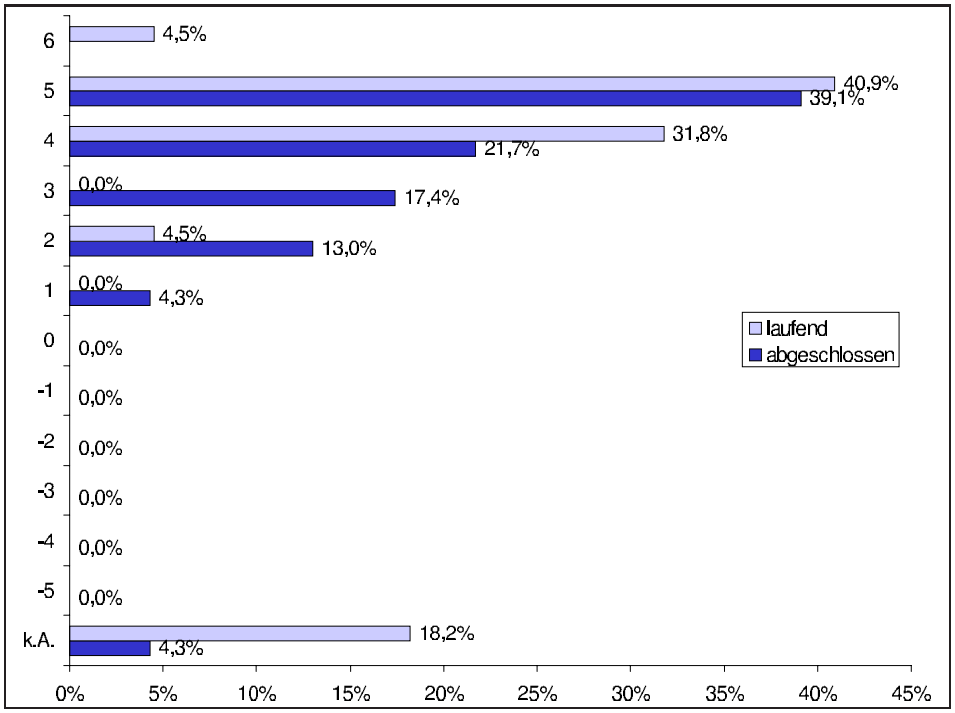


Abbildung 50: Schnelles Reagieren auf Änderungen (n=45)

Es fällt auf, dass die Bewertungen zur Erreichung der Ziele bei laufenden und abgeschlossenen Projekte grösstenteils übereinstimmen; die guten Ergebnisse, die während laufender Projekte erzielt werden, werden durch den Erfolg abgeschlossener Projekte bestätigt.

Insgesamt wird XP in den untersuchten Projekten also sehr erfolgreich eingesetzt. Bis auf die Metapher werden alle Elemente vorwiegend als hilfreich angesehen und stark genutzt. Die nicht ganz so stark genutzten - On-Site Customer und damit in Zusammenhang stehend Planning Game - werden eher aufgrund äusserer Zwänge nicht so stark genutzt als weil sie als nicht sinnvoll empfunden würden. Der Überblick über die Erreichung der Ziele stellt sich ebenfalls als sehr zufriedenstellend dar. Jedes der Ziele konnte in circa drei Viertel der Projekte voll erreicht werden, schlechter als sonst fühlte sich mit XP niemand bedient.

- *„XP definitely enabled us to work in an extremely efficient manner.“*

5.7 Die Schwierigkeiten mit den XP Elementen

Im vorherigen Abschnitt wurde bereits erkennbar, mit welchen XP Elementen Probleme bestehen. Hier soll nun erläutert werden, welche Probleme das sind und woher sie rühren. Gefragt wurde nach den drei im Projekt am wenigsten benutzten Elementen.

Abbildung 51 gibt nochmals einen Überblick über die am wenigsten benutzten Elemente. Die drei am wenigsten benutzten Elemente sind Metaphor, On-Site Customer und Planning Game. Metaphor und On-Site Customer liegen hierbei mit Abstand vorn. Sie waren in zwei Drittel der Projekte die am wenigsten benutzten Elemente. Es folgt das Planning Game mit 30% sowie Refactoring und Coding Standards mit jeweils 20%. Auch Simple Design und Pair Programming zählten in jeweils knapp 16% der Projekte zu den drei am wenigsten benutzten Elementen. Testing und 40-Hour Week zählten eher selten zu den am wenigsten benutzten Elementen, Short Releases, Common Code Ownership und Continuous Integration waren nur in ein oder zwei Projekten bei den selten genutzten dabei.

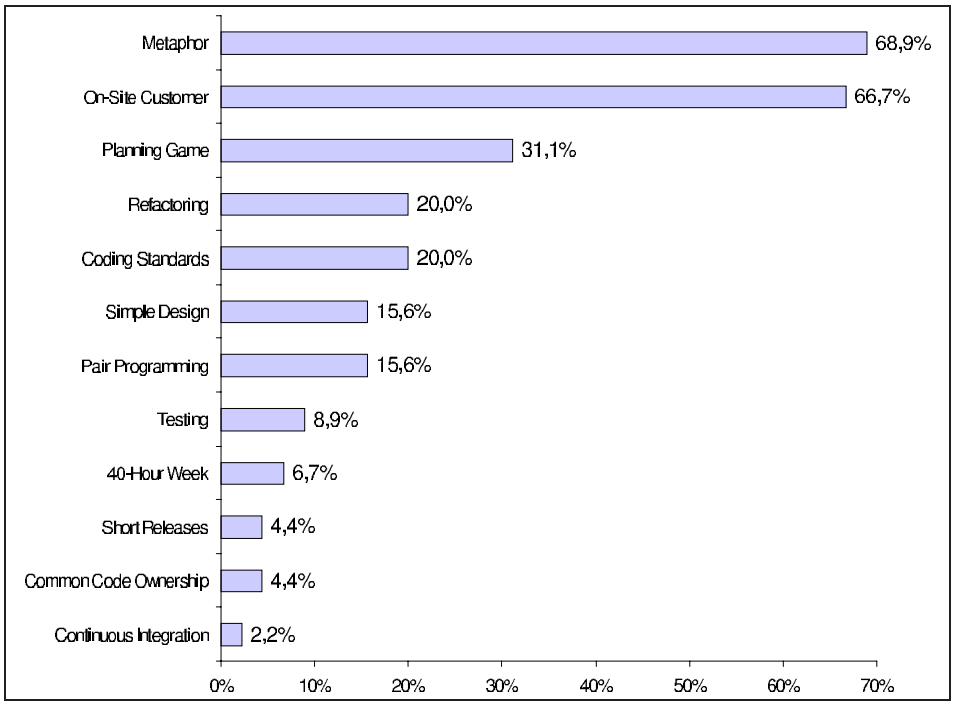


Abbildung 51: Am wenigsten benutzte Elemente (n=45)

Worin bestehen nun die Probleme mit den Elementen, die hier die Tabelle anführen? Häufig genannte Ursachen werden im folgenden diskutiert.

Die grössten Probleme gab es mit der Metapher. Hier herrscht Unklarheit darüber, wie dieses Element anzuwenden ist und wozu es überhaupt dienlich sein soll. Knapp 30% gaben an, keinen Grund für das Vorhandensein der Metapher zu sehen.

- *„No idea why this is there.“*

20 % versuchten dennoch, eine Metapher zu finden, es war ihnen jedoch nicht klar, wie dieses Element anzuwenden ist.

- *„Metapher: the concept is ill-explained in the XP canon, and needs more research into how it fits with the rest of the process.“*

Beim Element Metapher herrscht Handlungsbedarf; soll sie im XP Prozess erhalten bleiben, so muss sie verbessert werden.

- *„Metapher: refactor this to something which helps to grow and evolve an architecture baseline.“*

Ganz anders verhält es sich mit dem On-Site Customer. Bei diesem Element besteht oft der Wusch, es stärker anzuwenden:

- *„On-Site Customer: This would be great, but we did not have a chance to experience it.“*

Hier gab es bei einem Siebtel der Projekte das Problem, dass sich der Kunde in einer anderen Stadt oder Land befand. Das grössere Problem war jedoch, dass er oder sie nicht überzeugt werden konnte, am Projekt teilzunehmen. In fast einem Drittel der Projekte war dies der Fall.

- *„Hard to convince the customer to be on-site always.“*
- *„[Customer] did not participate as much as would have been preferred.“*
- *„On-Site Customer, didn't use this because we couldn't get a customer to participate.“*
- *„The customer was very busy on other projects ...“*

Was dies für den Projekterfolg oder -misserfolg bedeutete, wird in Abschnitt 5.8 diskutiert.

Doch es gab auch drei Fälle, die angaben, das Element nicht stärker genutzt zu haben, da der Kunde nicht die ganze Zeit vor Ort gebraucht worden wäre:

- *„On-Site customer - We didn't need him on-site 100%."*

In knapp 10% der Fälle wurde das Element nicht benutzt, da es sich nicht um Auftragsarbeit handelte und deshalb kein direkter Kunde vorhanden war.

Resultierend aus den Problemen mit dem On-Site Customer ergaben sich Probleme mit dem Planning Game. In knapp 10% der Fälle konnte das Planning Game nicht zufriedenstellend durchgeführt werden, da der Kunde nicht verfügbar war, in einem Fall wurde angegeben, die Kommunikation zwischen dem Kunde und dem Team sei schlecht gewesen. Auch wurde mehrmals beklagt, der Kunde sei nicht in der Lage gewesen, seinen bzw. ihren Part im Planning Game zufriedenstellend wahrzunehmen:

- *„Customers not really competent (or to busy) to write stories."*

Aufschlussreich waren die Probleme beim Pair Programming. Nur in einem Fall wurde angegeben, dass das Management nicht willens war, zwei Programmierer an einen Rechner zu setzen - eine bemerkenswert niedrige Zahl, da die angebliche „Ressourcenverschwendung" beim Pair Programming ein sehr häufig gehörter Vorbehalt gegen XP ist. In über 10% der Fälle jedoch lag das Problem bei den Entwicklern selbst. Diese wollten nicht in Paaren programmieren, arbeiteten remote oder es wurde angegeben, Pair Programming wäre mit einigen Entwicklern schwer machbar.

- *„Pair programming - other developers were not interested."*
- *„Pair Programming, there were two members of the team which would not do this, never resolved this issue."*

Auch hier wird weiter unten noch diskutiert, was das für Projekterfolg oder -misserfolg zu bedeuten hatte.

Testing war ein eher selten genanntes Element unter den wenig genutzten; ein Befragter gibt an:

- *„The time pressure was so high that we didn't start developing with test cases. During the development we had not the time to add the number of test cases we missed at the beginning. This was a big mistake and I would never do that again in the next project."*

Die Bedeutung von Testing für den Projekterfolg wird ebenfalls später ausführlicher diskutiert werden.

Für die Elemente Refactoring, Short Releases, Simple Design und 40-Hour Week gaben nur wenige Befragte Gründe für die geringe Nutzung an; bei Refactoring wurde genannt, dass es für bestimmte Sprachen keine Refactoring-Tools gäbe, ein anderes Mal hiess es, es hätte zu viel alten Code gegeben um

ihn zu refaktorisieren. In einigen Fällen erschienen die Systeme zu komplex, um Simple Design anwenden zu können. Zu Short Releases wird angegeben:

- *„From a quality point of view we could deliver new framework versions daily but that would cause continuously major migration offers for the application projects using the framework.“*
- *„The issue here is that while the software could release quickly and frequently there are many other issues, besides technical, involved with a product release. I am not sure that short release cycles apply to a product due to confusion that it would cause in the market place.“*

Bei der 40-Stunden Woche wurde in zwei Fällen Zeitdruck als Grund genannt:

- *„40-Hour-Week, it is difficult when you are under pressure.“*

Aus dem oben beschriebenen wird ersichtlich, dass die Probleme mit den wenig benutzten Elementen von XP einiges mit einer traditionellen, starren "Kultur“ in der Softwareentwicklung und in der Arbeitswelt an sich zu tun haben. Dass man keinen oder keine Mitarbeiter als On-Site Customer zur Verfügung stellen will, zeugt von bedenklicher Kurzfristigkeit im Denken bei der Projektdurchführung. Pair Programming mag nicht jedermanns Sache sein, jedoch wird man weiter unten sehen, dass es in den Projekten, in denen es verwendet wurde, als einer der Faktoren für den Projekterfolg genannt wurde. Unternehmen, die teamfähige Entwickler haben, werden also einmal mehr denn je erfolgreicher sein.

5.8 Der Projekterfolg, die Erfolgsfaktoren, die Risiken

Die Zahlen in Abbildung 52 sprechen für sich: über 90% der abgeschlossenen Projekte wurden erfolgreich abgeschlossen. Nur in einem Fall wurde der Projektabschluss nur teilweise als erfolgreich bewertet (Wert 4 von 9); hier bemerkt der Bewerter folgendes:

- *„I certainly want to stress that insofar as the project is regarded here as a failure in important ways, nobody analyses the failures as failures of XP.“*

Als Gründe für das Scheitern gibt er an, dass der Zeitplan und die Erwartungen so hoch gesteckt waren, dass es nicht gelang, das Gewünschte zu liefern. Anzumerken ist auch, dass es sich bei diesem Projekt um das Projekt handelte, das von nur einer Person durchgeführt wurde.

Ein Teilnehmer mit einem erfolgreich verlaufenen Projekt berichtet folgendes:

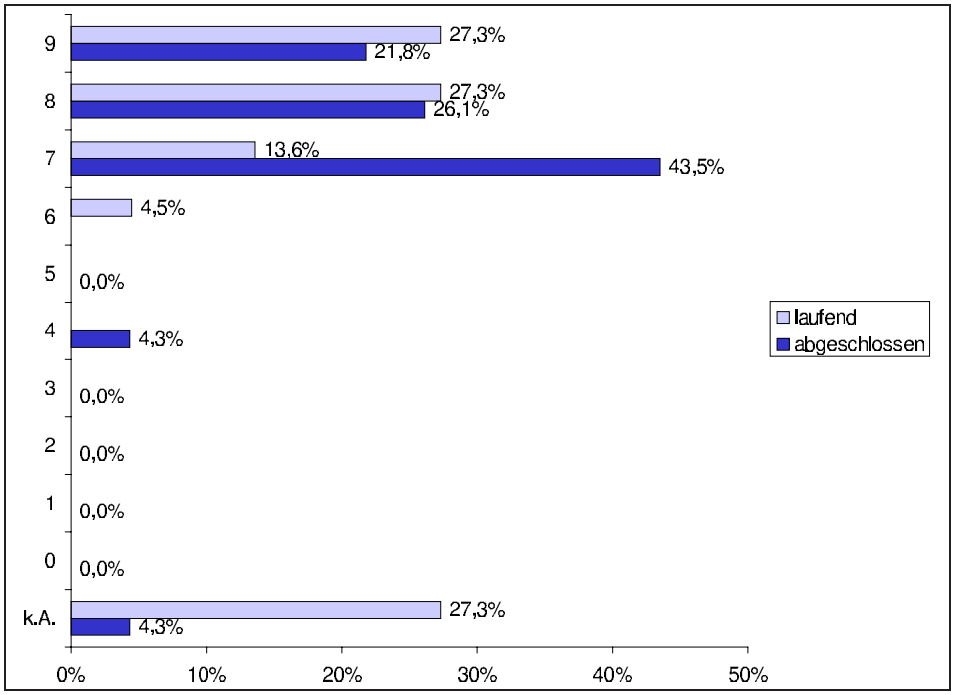


Abbildung 52: Erfolgreicher Projektabschluss (n=45)

- *„The software is in production and the customer is satisfied. The project is not definitively closed, but suspended in order to start another more complex and important project with the same customer.“*

Auch bei den laufenden Projekten gaben die Teilnehmer häufig eine Wertung ab, da teilweise Releases bereits abgeschlossen waren. Auch hier gaben 70% an, das Projekt würde aller Voraussicht nach erfolgreich beendet; einer gab den Wert 6 an und knapp 30% machten noch keine Aussage. Im Falle der Bewertung mit 6 handelte es sich um ein Projekt, dasdas unter Verwendung des Wasserfall Modells mit der Waterfall-Methode begonnen worden war (für diesen Teil gab der Bewerter eine 0 für den Projekterfolg) und mit XP versucht wurde zu „retten“.

Erfolgsfaktoren

Was waren nun die Hauptgründe für diesen erfolgreichen Verlauf fast aller XP Projekte? Oft genannte Gründe sind in Abbildung 53 aufgeführt und werden nachfolgend beschrieben.

Am häufigsten genannt wurden Komponententests, Pair Programming sowie XP und seine Ziele an sich (*„Deliver quality software in time, fast re-*

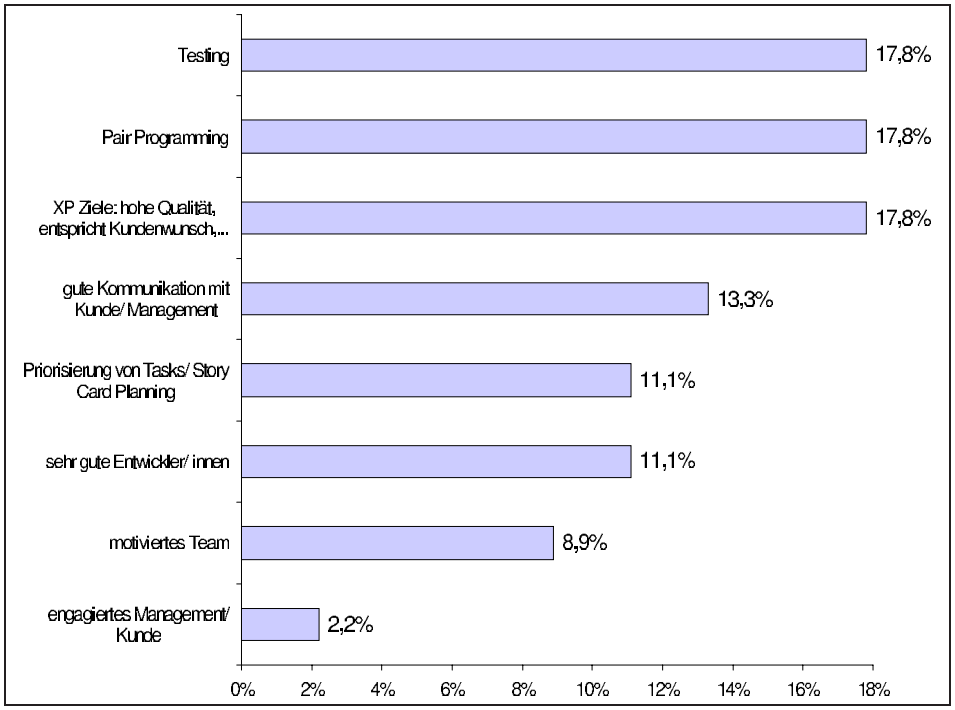


Abbildung 53: Gründe für den Projekterfolg (n=45)

act to changes, let developers have fun with their work“). In jeweils 18% der Projekte wurden diese Faktoren als entscheidende Gründe für den Projekterfolg gesehen. Auf die Frage “ *What where the major reasons for its success?*“ wurde unter anderem geantwortet:

- „*Tests and pair programming had prio 1 as success contributions.*“
- „*Test, test, test. Write test cases first. Have a good test driver available for ALL components.*“
- „*Quality Software delivered on time.*“
- „*Stability and defect rate is excellent.*“
- „*Total adoption of XP.*“

Während sich die Teilnehmer der Studie über den Einsatz von Testing und dessen Wichtigkeit weitgehend einig waren (lediglich einer klagte: „*Testing is hard.*“), ist der Umgang mit Pair Programming nicht in allen Fällen problemfrei. Da dieses Element jedoch ebenfalls häufig als wesentlich für den

Erfolg angegeben wurde, sollen die Probleme damit im folgenden noch einmal genauer untersucht werden.

- *„I was most sceptical about this [Pair Programming] before; I'm most in favor of it now.“*
- *„The Pair Programming was a major benefit to the project. Coding was completed much faster and there was immense knowledge transfer between the programmers.“*

Wie bereits oben anhand einiger Zitate gezeigt, sind Entwickler teilweise gegen Pair Programming eingestellt und wollen dieses Konzept nicht einsetzen. Anhand einiger Aussagen scheint sich jedoch zu zeigen, dass es sich hierbei auch um eine Gewöhnungssache handeln kann:

- *„Aller Anfang ist schwer. Aber so langsam ist es für die Leute "normal" mit zwei Leuten am Rechner zu sitzen. (das kostete am Anfang einige Überwindung)“*

Es erscheint jedoch auch nicht sinnvoll, Programmier, die partout alleine programmieren wollen, zum Pair Programming zu zwingen; diese sind alleine wahrscheinlich tatsächlich effizienter, da sie sich beim Pair Programming nicht wohlfühlen. Auch dem jeweiligen Programmierpartner wäre damit kein Gefallen getan:

- *„Pair Programming: never decided to use it at 100%, had two developers in team who refused to do it or were very difficult to work with.“*

Da dauerndes Pair Programming auch für Personen, die es gerne machen, anstrengend sein kann, könnte auch eine Abwechslung zwischen Pair und Single Programming in Betracht gezogen werden:

- *„I'm not sure why we have not done more pair programming. Actually, I'm not sure my perceived lack of pairing is actually a problem. Its quite possible that given our team and the nature of the work that the amount of pairing that we have been doing is quite acceptable.“*

In über 13% der Projekte wurde gute Kommunikation zwischen dem Entwicklerteam und den Kunden und/oder Management als ein wesentlicher Erfolgsfaktor bewertet. Gute Kommunikation ist in jedem Projekt wichtig, egal, welche Methodik benutzt wird, der Vorteil des XP Prozesses ist hier jedoch, dass stetige Kommunikation zwischen den Entwicklern sowie zwischen Kunden und Entwicklerteam bereits im Prozess selbst verankert ist.

Bei jeweils 11% spielten die Priorisierung von Stories und die guten Kenntnisse der Entwickler eine grosse Rolle.

- *„Major reason for success was focusing only on the most important stories and implementing them in the most needed order.“*

Die gute Teammotivation trug in 9% der Fälle wesentlich zum Erfolg bei. In einem Fall wurde ein engagiertes Management und ein engagierter Kunde als Grund für Projekterfolg genannt - ein nicht zu unterschätzender Faktor, wie bei der Diskussion der Risiken noch zu sehen sein wird.

Risiken

Doch was kann nun den Projekterfolg in einem XP Projekt gefährden? Abbildung 54 zeigt häufig genannte Risiken.

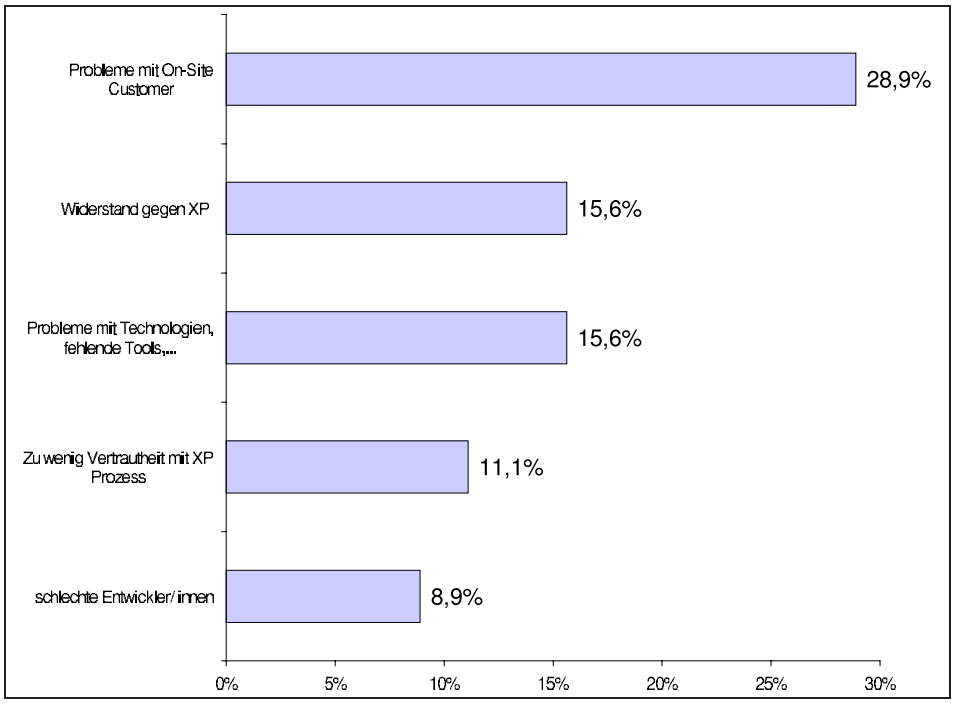


Abbildung 54: Grösste Risiken für den Projekterfolg (n=45)

Das weitaus häufigste Risiko waren Probleme mit dem On-Site Customer. Dies wurde bei 29% der Projekte angegeben. Auch dieses Problem soll hier etwas genauer untersucht werden.

Teilweise wurde das Fehlen eines On-Site Customer als hohes Risiko angesehen, teilweise als Risiko, dem mit Hilfe eines gut geführten Planning Games gut beizukommen war.

- *„no customer on site. not too dangerous since there were clear requirements and regular meetings. “*

- *„Lack of an on-site customer - very dangerous, causes a lack of focus in the project.“*

Die Gründe für das Fehlen des On-Site Customers liegen oft in traditionellen Einstellungen und der Unternehmensphilosophie begründet:

- *„On-site customer - the current culture of how software development “works“ makes it extremely hard to apply this in practice, i.e. to involve a non-technical stakeholder as a peer within the team. Instead, the relationship between engineers and users are implicitly viewed as „adversarial“.*

- *„On-Site Customer, difficult from a logistic point of view; not very well compatible with company's culture.“*

- *„In general customer involvement in the project was lacking. This was primarily due to political issues with our customer.“*

Strategien, den fehlenden Kunden zu ersetzen, wurden ebenfalls als unterschiedlich efffizient angesehen:

- *„Customer proxy. This simply involves appointing someone to make immediate decisions on behlaf of the customer. Of course, they have to be onsite.“*

- *„Customer substitues are not sufficient. They don't know enough about the everyday work of the 5-7 customer groups addressed by the project.“*

Auch wurde gefordert:

- *„On-Site Customer: Some projects just don't allow it. There should be other elements to make up for that.“*

War der Kunde da, konnte es trotzdem noch zu Problemen kommen:

- *„The problem with this is that quite often the customer doesn't know how to drive. As such is unable to set priorities. It would very useful for the customer to understand how to drive before doing the planning game.“*

Das Element On-Site Customer kämpft noch mit vielen Problemen. Dennoch sollte versucht werden, diesen beizukommen, da der bei XP verfolgte Ansatz, den Kunden stark in die Entwicklung einzubeziehen, dieser entscheidende Impulse geben kann.

- *„I believe that XP is a highly effective process to use when the client wants to be engaged a drive the solution. “*

Auf Platz zwei der Risiken rangiert Widerstand gegen XP. Dieser Widerstand kam sowohl vom Management als auch von Kunden, von anderen am Projekt beteiligten Departments und teilweise auch von den Entwicklern selbst. Knapp 16% der Projekte hatten damit zu kämpfen.

- *„Project managment had no trust in team and XP - very dangerous.“*
- *„Human obstacle : getting other people than the development team to play at XP.“*
- *„We tried having our QA person do automated functional tests using a GUI testing tool. That person did not follow thru.“*

Teilweise wurden XP Projekte sogar „heimlich“ durchgeführt. Da dies in XP-Projekten häufiger vorzukommen scheint, wurde dafür mittlerweile der Begriff „Guerilla-XP“ eingeführt.

- *„The only obstacle was time and the customer. The customer wasn't informed...“*
- *„Main obstacles? hiding xp from the upper management .“*
- *„On <project name >, we did XP “under the hood“, i.e. the customer never knew that we were using XP (<company name >has a standard methodology !) “*

Ebenfalls knapp 16% der Projekte kämpften mit den in Softwareprojekten üblichen Schwierigkeiten mit (neuen) Technologien, schlechten Tools, etc. 9% hatten Probleme mit nicht genügend qualifizierten Entwicklern; hier wurde in einigen Fällen angegeben, Pair Programming sei hilfreich gewesen, um die Fähigkeiten dieser Entwickler zu verbessern.

11% sahen eine noch zu geringe Vertrautheit bei den Projektbeteiligten mit dem XP Prozess als Risiko.

- *„Major risk? The other 7 developers! Unfortunately they are very traditional developers, and have been slow on accepting XP practices.“*

- *„Getting customer used to XP and away from traditional development practices. Learning curve for customers has been very dangerous and has required a significant amount of communication and XP education on our part.“*
- *„Noch nicht selbstverständliches XP-Verhalten (nicht so gefährlich für die Iteration, aber für den Prozeß in Zukunft).“*

Und schliesslich wurde noch diese wohl nicht ganz ernstgemeinte “Gefahr“ für den Projekterfolg angegeben:

- *„The development went so fast that the customers couldn't keep up. This forced the developers to take time off in July to lie on the beach and wait ;-) “*

Der gerade aufgeführten Erfolgsfaktoren und Riskien sind sehr aufschlussreich für den zukünftigen Umgang mit XP Projekten. Werden als Erfolgsfaktoren wichtige XP Elemente wie Testing und deren erfolgreiche Implementierung im Projekt angegeben, so herrscht bei den Risiken eine noch ungenügende Vertrautheit mit dem doch recht neuen und ungewöhlichen XP Prozess vor, die zu Ablehnung und Skepsis führen kann, was wiederum dem Projekterfolg hinderlich ist. Die Projektbeteiligten sind noch nicht ausreichend trainiert und erfahren in der Anwendung des XP Prozesses, die Kunden ungenügend vorbereitet.

Doch trotz dieser momentan noch bestehenden Probleme mit XP konnte bei den teilnehmenden Projekten eine Erfolgsquote von über 90% erreicht werden. XP könnte noch effizienter eingesetzt werden, wenn alle Projektbeteiligten genügend vorbereitet wären und Skepsis und Misstrauen gegenüber XP verringert werden könnte.

Man versuche sich auch einmal vorzustellen, wie erfolgreich herkömmliche Softwareentwicklungsmethoden noch wären, wenn das Management oder die Kunden kein Vertrauen in die Methode hätttten oder gar nicht von deren Anwendung wissen dürften ...

5.9 Die Bewertung des XP Einsatzes

Auch bei der Frage nach dem weiteren Einsatz von XP sprechen die Zahlen eine klare Sprache (Abbildung 55): ausnahmslos alle wollen XP wieder verwenden und vertreten es aktiv. In drei Fällen wurde angegeben, es in angepasster Form wiederverwenden zu wollen, angepasst an den Typ bzw. die Grösse des Projekts bzw. in Form von Extreme Modeling (XM). Hier muss jedoch nochmals betont werden, dass die Studie, wie oben ausgeführt, eventuell von XP nicht überzeugte Personen nicht erreichen konnte.

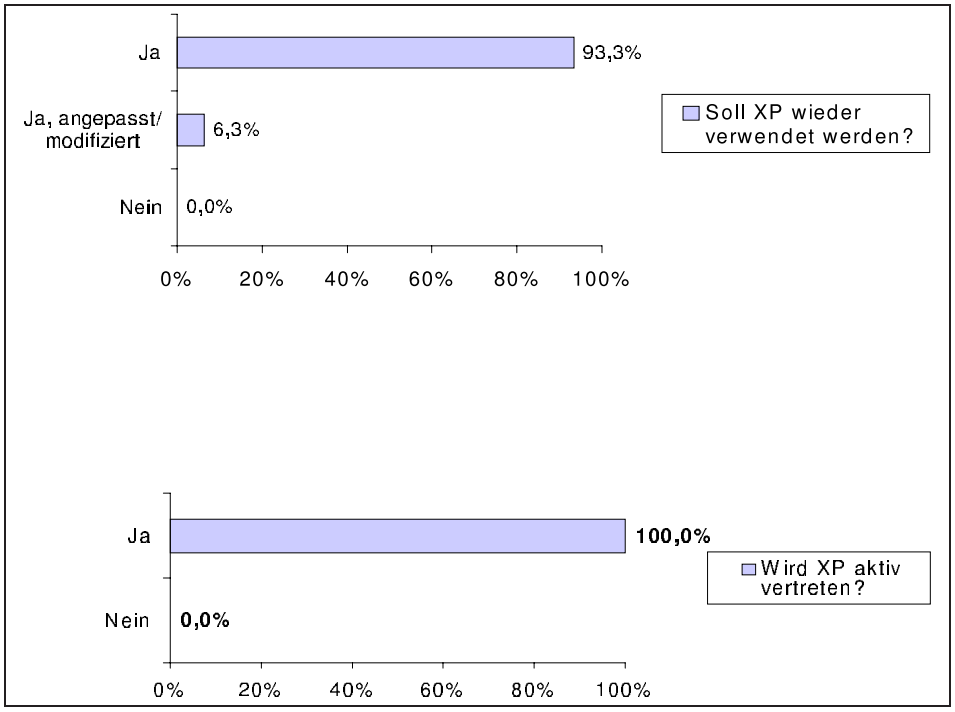


Abbildung 55: Bewertung des XP Einsatzes (n=45)

- *„We are advocating it [XP] to our present and potential customers.“*
- *„I'll be starting at a new company in a month. They're still looking for methodology really so I'll try hard to introduce it.“*
- *„Yes, we are trying to use XP as much as possible on all of our projects.“*

Auch hier zeigen Antworten (auf die Frage *„Will you use XP again?“*) wie *„If I have a chance.“* oder *„If possible yes.“*, dass der Einsatz von XP noch nicht selbstverständlich ist und es gerade erst beginnt, sich als Softwareentwicklungsmethode zu etablieren.

5.10 Das Interesse an Extreme Modeling

Hier soll noch auf das Interesse an Extreme Modeling (XM) eingegangen werden, einer stärker formalen Weiterentwicklung von XP, die es mit der Unified Modeling Language (UML) kombiniert.

Über 85% Prozent der Befragten kannten sich aus mit UML, doch vermisst wurde sie nicht wirklich. Knapp 50% gaben an, sie nicht vermisst zu

haben, nur knapp 5% vermissten sie. Allerdings gaben 35% an, sie nicht vermisst zu haben, da sie sie für kurze Designskizzen etc. benutzt hätten.

- *„I do not feel that UML is prohibited by XP. Just use it with care.“*

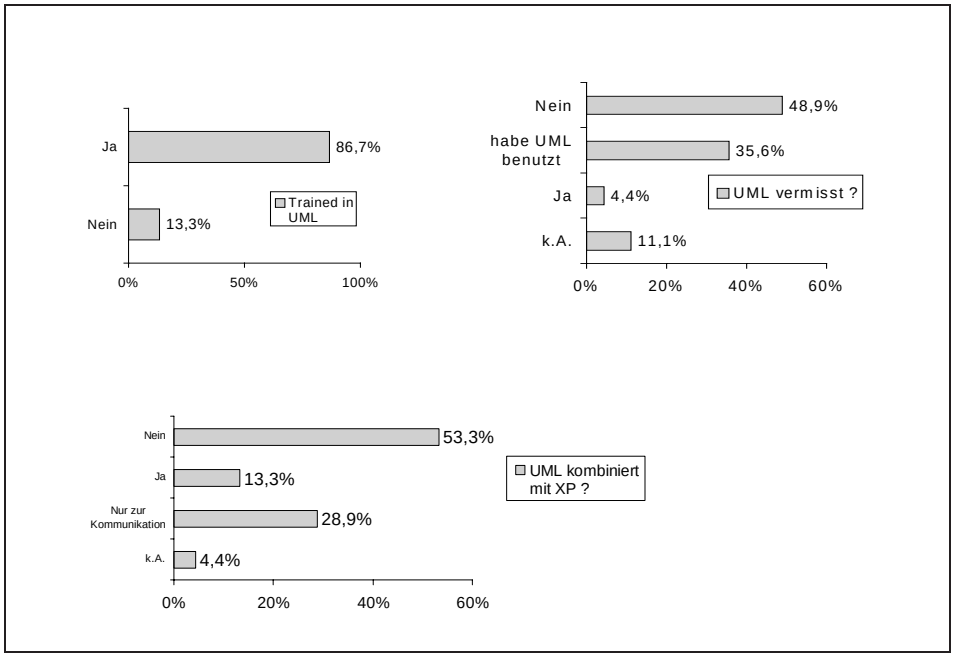


Abbildung 56: Fragen zur Kombination mit UML (n=45)

Bei der Frage, ob UML mit XP kombiniert werden solle, gab es auch klare Aussagen. Nur 13% wünschten dies. Knapp 55% hatten kein Interesse daran. 29% wollten die UML auch in XP Projekten benutzen, gaben aber klar zum Ausdruck, sie solle nur zur Kommunikation benutzt werden, zum schnellen Kommunizieren von Design-Ideen, es solle aber kein Code daraus generiert werden.

- *„I would never make it part of the xp process. If some project likes to do it: ok. But do not make a formal prerequisite of xp.“*

Aber es gab auch gegenteilige Meinungen zu der Frage, ob UML mit XP kombiniert weden solle:

- *„Yes, UML and use cases. XP is too lightweight in regards customer requirements.“*

Fazit: über 80 % der Befragten wollen UML im XP Prozess zwar benutzen dürfen, aber nicht direkt integrieren.

5.11 Diskussion der Ergebnisse

In einer am Anfang dieses Kapitels bereits erwähnten IBM-Umfrage zum Thema *„What are your thoughts on Extreme Programming?“* ([IBM01]) glauben 49%, dass XP nicht funktionieren kann. Hierbei sagen 25% *„It's a good idea, but it could never work“*, 16% glauben *„It's a bad idea – it could never work“* und 8% sagen gar: *„I've tried it and hated it“*. Die restlichen 51% sagen *„I've tried it and loved it“*.

Die Ergebnisse der vorliegenden Studie zeigen, dass XP erfolgreich funktionieren kann. Trotzdem sollen diese Ergebnisse hier noch einmal kritisch bewertet und in Relation gesetzt werden.

Wird in den Fragebögen, die an der Studie teilnahmen, zwar bis auf einen Fall nur über erfolgreiche Projekte berichtet, so findet sich doch in einem Bogen ein Hinweis auf ein weniger erfolgreiches XP Projekt:

- *„They had an unmentored „XP“ project earlier. It went horribly wrong because they were badly organised and none of them had done XP before. XP is therefore not a good thing to talk to them about. The project I'm involved in does not therefore say it's XP; but it uses some of the XP practices ...“*

Pelrine betont in [PUN01] die Wichtigkeit des Coaches bei einem XP Projekt:

- *„eXtreme Programming must be learned, and the successful application of the methodology is largely dependant upon the ability of the developer to keep all elements end practices in balance throughout the whole lifecycle of the software. We maintain that the use of a coach is necessary to successfully learn XP, and to guarantee the balance in the development process.“*

Die Verantwortlichkeit dieses Coaches sei *„...to accompany the whole XP-Process, [...], and to ensure the balanced introduction of the XP methodology and practices.“*

In einem anderen Bogen, in dem der Projektverlauf von den Entwicklern positiv bewertet wurde, findet sich ein Hinweis, dass die Sicht auf Projekte durchaus verschieden sein kann:

- *„We believe the project has been a success. People on other projects have not been interested in what has been going on and have only judged the results based on how long the project has lasted (longer than expected due to having to build OO framework code and troubles with 3rd party software) and have judged XP based on that alone.“*

Desweiteren wurde am Anfang dieses Kapitel bereits erwähnt, dass nach Auswertungsschluss noch zwei Bögen eingingen, die nicht mehr mit ausgezählt wurden. In einem dieser Bögen wird ein aus Sicht des Bewerters eher negativ verlaufendes XP Projekt beschrieben. Als Hindernisse werden schlechte Kommunikation, zu hohe Teamgrösse (35 - 40 Leute) und schlechtes Design der anfänglich nicht mit XP entwickelten Software angegeben. Den Spass bei der Arbeit bewertet er mit -3, die Qualität der Software mit -4 und das schnelle Reagieren auf Änderungen mit -5 (auf einer Skala von -5 bis +5). Er gibt an, dass aufgrund fehlender Tests Angst vor dem Refaktorisieren bestehe, dass es aber sehr schwierig sei, Tests zu schreiben, da der Persistence Layer und viele Frameworks das sehr erschwerten. Auch Common Code Ownership würde nicht praktiziert, da die technischen Leiter den Code selbst releasen wollten. Dies führe ebenso zu grossen Problemen mit der Continuous Integration. Er berichtet:

- *„An important part to XP is not explained very well in the books. It has to do with the open communication and willingness of the developers to want to make the system better. They have to want to change, want to accept the XP way. People on this project, although they say that they want XP, don't really want to change. They don't communication. They want too much control over everything. And they think that they've done well in the past, so why use XP?“*

Da er vorher in einem anderen Unternehmen an einem sehr erfolgreichen XP Projekt teilgenommen hat, sieht er die Unterschiede zu diesem Projekt besonders deutlich:

- *„It was, without a doubt, the best experience I've had as a professional. Sure, we had our doubts going in, but we were willing to try. I can say now, that all software should be built like that.[...] Everything was fantastic. But, it starts with the developers. Each and every one HAS to be on board. If they're fighting it, as we are here, then it can be an absolute disaster.“*

Obige Erfahrungen zeigen, dass XP Projekte durchaus scheitern können. XP kann nicht bedenkenlos immer und überall eingesetzt werden; es kann auch kein Allheilmittel gegen alle Probleme bei der Softwareentwicklung sein.

- *„XP is not a silver bullet. It cannot solve all of the problems that can arise in a project.“*

Gefahr für das Scheitern eines XP Einsatzes kann wie oben beschrieben bestehen, wenn zu viel alter, ungetester Code im Spiel ist, wenn die Teamgrösse zu hoch ist und wenn die Teammitglieder nicht richtig vorbereitet sind oder XP schlicht nicht wollen.

- *„A team of 35-40 people cannot agree on anything.“*
- *„A final reason for success: The team has been willing to think about the process and adapt it to suit our problems.“*
- *„Not all people are suited for or enjoy XP-style development. These people have left the company.“*
- *„It is extremely important to hire staff by making them spend half a day Pair programming with you.“*

Auch kann XP nicht bedenkenlos für jeden Typ von Projekt eingesetzt werden:

- *„The company is in the process of finding a corporate standard process. Because most of the projects are large and involve business design as well as software design, the plain XP approach isn't suitable.“*

Aber:

- *„XP has been criticized for not being scalable/applicable to larger projects. So what? It doesn't have to. I'd rather have a best solution for solvinf a known subset of problems than a pseudo-solution that works equally badly on all problems.“*

Also: XP da, wo es für die Art des Projekts geeignet ist, mit entsprechend guter Vorbereitung und Teammitgliedern, die gewillt sind, den XP Prozess umzusetzen. Dann kann es , wie in den Ergebnissen der Studie gesehen, sehr erfolgreich sein.

- *„XP as a simple, well-defined, disciplined process is productive and enjoyable. A huge improvement overall compared to “code & fix“.*

5.12 Schlussbetrachtung

Zu Beginn dieser Arbeit wurde erwähnt, dass viele Unternehmen Bedenken gegenüber einem Einsatz von XP hegen. Darauf soll nun, nachdem die Leser einen tieferen Einblick in die Thematik erlangt hat, noch einmal eingegegangen werden. Die vorhergehenden Ausführungen haben gezeigt, dass XP erfolgreich eingesetzt werden kann. Eine Studie für das Bundesministerium für Bildung und Forschung ([SWR+00]) macht in diesem Zusammenhang einige interessante Aussagen zur Softwareentwicklung in Deutschland, die zeigen können, dass XP in mehreren seiner Aspekte Anforderungen trifft, die deutsche Unternehmen an ein Vorgehensmodell stellen.

Wie diese Studie aussagt, benutzt nur die Hälfte der befragten Unternehmen in Deutschland bei der Softwareentwicklung ein Vorgehensmodell, meist ein unternehmenseigenes, das sich aber oft an Lehrbuchmodelle wie Wasserfall oder iterative Modelle anlehnt. Junge Firmen der New Economy (Start-Ups) benutzen meist keines der etablierten Modelle, sondern bezeichnen ihre Softwareentwicklung als eher „chaotisch“. Grosse Unternehmen haben meist je nach Projektart und Umfang mehrere Vorgehensmodelle etabliert, insbesondere auch Rapid-Development-Modelle. Als hinderlich bei der Sofwareentwicklung geben viele Firmen Probleme beim Requirements Engineering durch unzureichende Analyse und Management von Anforderungen, Zeitprobleme bei der Abwicklung von Projekten sowie Probleme bei der Qualitätssicherung bzw. beim Testen an.

Viele der in dieser Studie befragten Unternehmen erhoffen sich, der stetig wachsenden Forderung nach kürzeren Entwicklungszyklen und doch hoher Qualität durch verbesserte Methodenunterstützung gerecht zu werden.

Laut der Studie geht ein deutlicher Trend hin zu kürzeren Releases mit einer kleineren Menge neuer Funktionalitäten, die dem Kunden in kürzeren Abständen verfügbar gemacht werden. Die meisten Unternehmen sehen time-to-market zunehmend wettbewerbsbestimmend und als eine der wichtigsten Herausforderungen der Zukunft, der nur durch organisatorische und methodische Innovationen begegnet werden könne. Bei der Qualitätssicherung liegt momentan die Betonung in aller Regel auf den späten Phasen der Softwareentwicklung, zunehmend sind Qualitätssicherungsmassnahmen jedoch auch direkt in die Softwareentwicklung integriert. Es wurde auch eine bessere Integration von Werkzeugen in das etablierte Vorgehensmodell gefordert. Insbesondere wird auch das Fehlen von guten Werkzeugen zur Testautomatisierung beklagt. Die Einbindung des Kunden wird als wichtiger Beitrag zur Verbesserung der Produktqualität und Kundenzufriedenheit bezeichnet. Hierbei beschränkt ein Teil der Unternehmen die Kundeneinbeziehung auf Review von Anforderungsdokumenten und den Abnahmetest,

ein anderer Teil prüft darüber hinaus alle Zwischenprodukte mittels Kundenreviews, und einige wenige Unternehmen haben während der gesamten Projektlaufzeit integrierte Kunden-/Entwicklerteams.

Interessant hier ist, dass mehreres, das die befragten Unternehmen wünschen und das sie derzeit in ihren unternehmenseigenen Vorgehensmodellen teilweise bereits praktizieren, eine gewisse Affinität zu XP und seinen Konzepten aufweist. XP mit seiner starken Einbeziehung der Kunden, seiner auf kurze time-to-market ausgerichteten Entwicklungsweise und seiner Einbindung von Werkzeugen wie JUnit oder Refactoring Tools trifft Anforderungen und Wünsche der Unternehmen an ein Vorgehensmodell, und mehrere seiner Elemente wie beispielsweise Short Releases werden in den unternehmenseigenen Modellen bereits praktiziert.

XP wird für extrem gehalten, aber bei genauerer Betrachtung können interessierte Unternehmen feststellen, dass es lediglich einige ungewöhnlichere Konzepte wie Pair Programming mit häufig von Unternehmen gewünschten oder bereits eingesetzten Techniken kombiniert.

6 Ausblick und Bewertung

Extreme Programming ist eine leichgewichtige aber rigorose Softwareentwicklungs-Methode, mit einem Wertesystem sowie einer Sammlung von Grundprinzipien und Entwicklungspraktiken. Neuartig an XP ist die veränderte Gewichtung einzelner Konzepte und insbesondere der explizite Verzicht auf bestimmte wesentliche Tätigkeiten klassischer Projektorganisation. So führt z. B. der Verzicht auf Dokumentation zu einer veränderten Projektkultur mit intensiver Kommunikation der Entwickler untereinander und enger Einbeziehung des Kunden.

Trotz des extremen Namens ist XP nicht so radikal wie es den Anschein hat. Es ist z. B. möglich und in der Praxis sinnvoll, einzelne Konzepte der XP-Vorgehensweise in eine vorhandene Projektkultur einzubauen. Insbesondere die stärkere Betonung der automatisierten Tests und des Refactorings bieten sich an. XP führt also keineswegs zurück zu den Anfängen der Software Engineering-Kultur, sondern sucht auf Basis der vorgestellten "best practices" manchen Ballast heutiger Vorgehensmodelle zu vermeiden.

Nach der Meinung der Autoren gehören XP und seine Techniken ins Portfolio eines Softwareentwicklers, um sich derer bei Bedarf zu bedienen. XP kann gleichberechtigt neben anderen Software-Engineering-Techniken dem Entwickler als Handwerkszeug zur Verfügung stehen. Weil XP außerdem für die Lehre gewisse Vorteile besitzt, läßt sich XP in einer geeigneten Vorlesung oder besser noch einem Praktikum den Studenten relativ einfach nahe bringen.

XP stellt in seiner heutigen Form nur einen Zwischenzustand dar und wird sich weiter entwickeln. Dazu gehören bessere Werkzeugunterstützung für die Softwarekonstruktion und –analyse, genauso wie die Entwicklung eines besseren Verständnisses über die Vor- und Nachteile von XP auf Basis wissenschaftlich gesicherter Untersuchungen.

Der vorliegende Bericht stellt dazu einen Baustein dar. Er hat gezeigt, wie der Extreme Programming Prozess von Softwareentwicklungsteams in der Praxis eingesetzt wird, und in einer ihren Inhalten signifikanten, quantitativen Studie belegt, dass XP als Methode sinnvoll, erfolgreich und zielführend in der Softwareentwicklung eingesetzt werden kann. Sie hat auch gezeigt, welche Schwierigkeiten verstärkt auftreten, wenn XP in der Praxis angewendet wird. Die im Rahmen der Arbeit durchgeführte Studie zu XP Projekten hat einige signifikante Aussagen und Daten zum Thema XP gebracht. Manche davon waren durchaus überraschend. Der Einsatz von XP wird immer noch durch Vorurteile behindert und XP von vielen nicht als eine seriöse Methode angesehen, sondern oft vorschnell als "Hacking" verurteilt. Da sich die Vorgehensweise von XP in einigen Merkmalen sehr deutlich von konventio-

nellen Methoden unterscheidet, sind diese Vorurteile zunächst verständlich. Ziel dieser Studie war die Erhebung eines ersten Satzes quantitativer Zahlen zum Einsatz von XP, die es erlauben, diese Vorurteile besser zu beurteilen. Denn bisher konnten die nur durch vage Intuition begründeten Vorurteile und ihre ebenso auf Basis von Intuition basierenden Gegenargumente weder bestätigt noch widerlegt werden.

Durch die Aufbereitung der Ergebnisse der Studie konnte dem Leser ein tieferer Einblick in industrielle XP Projekte in Unternehmen aller Grössen und Branchen weltweit gegeben werden. Eine grosse Menge an interessanten Blickwinkeln, Meinungen und Aussagen trug dazu bei.

Durch die immer größer werdende Verbreitung von XP wir in der nächsten Zeit eine größere Basis an Erkenntnissen über Vorzüge und Nachteile der XP entstehen. Es ist wesentlich, dieses Wissen von Entwicklern, Kunden und Projektleitern in quantitatives Datenmaterial zu übersetzen und zugänglich zu machen. Deshalb werden weitere Umfragen notwendig sein. Die in diesem Bericht beschriebene Umfrage kann als Grundlage auch für Verbesserungen zukünftiger Umfragen gesehen werden. Insbesondere ist mehr Gewicht auf die inoffiziellen Projekte („Guerilla-XP“) zu legen, die Augrund firmenpolitische Umstände nicht öffentlich gemacht werden können.

Danksagung

Diese Arbeit wurde unterstützt von der Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst durch den Bayerischen Habilitationsförderpreis und dem Bundesministerium für Bildung und Forschung durch das ViSEK-Projekt. Unser Dank gilt der Firma ESG, der Mummert + Partner Unternehmensberatung und allen Teilnehmern der Umfrage.

Literatur

[BBB+85] F.L. Bauer, R. Berghammer, M. Broy, W. Dosch, F. Geiselbrechtinger, R. Gnatz, E. Hangel, W. Hesse, B. Krieg-Brückner, A. Laut, T. Matzner, B. Möller, F. Nickl, H. Partsch, P. Pepper, K. Samelson, M. Wirsing, and H. Wössner. *The Munich Project CIP, Vol 1: The Wide Spectrum Language CIP-L.* LNCS 183. Springer-Verlag, 1985.

[Bec99] K. Beck. *Extreme Programming Explained.* Addison Wesley, 1999.

[BF01] K. Beck and M. Fowler. *Planning Extreme Programming.* Addison Wesley, 2001.

[Cun01] W. Cunningham. Extreme Programming Roadmap, 2001. http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap.

[Eck00] J. Eckstein. XP-Extreme Programming. *basicpro*, (35):6–11, 2000.

[EH00] A. Elting and W. Huber. Immer im Plan? Programmieren zwischen Chaos und Planwirtschaft. (2), 2000.

[EH01] A. Elting and W. Huber. Vorgehensmodelle contra Extreme Programming. 2 teilig. *sw development. Magazin für Software-Entwicklung*, (1&2):25–30, 2001.

[Fow99] M. Fowler. *Improving the Design of Existing Code.* Addison Wesley, 1999.

[GE01] M. Grund and H. Ehrlich. Smalltalk und XP im „Corporate Development“: ein Erfahrungsbericht, 2001. http://www.sigs.de/html/andrena.html.

[IBM01] IBM. IBM Java Poll, 2001. http://www-106.ibm.com/developerworks/library/java-pollresults/xp.html.

[JAH00] R. Jeffries, A. Anderson, and C. Hendrickson. *Extreme Programming Installed.* Addison Wesley, 2000.

[Jef01] R. Jeffries, 2001. http://www.xprogramming.com, www.xprogramming.com.

[JR01] C. Jacobi and B. Rumpe. Hierarchical XP – Improving XP for large scale projects. In G. Succi and M. Marchesi, editors, *Extreme Programming Examined*. Addison-Wesley, 2001.

[Jun01] Junit. Junit Test-Frameworks für mehrere Sprachen, 2001. http://www.junit.org, www.junit.org.

[Kru00] P. Kruchten. *The Rational Unified Process, 2nd Edition*. Addison Wesley, 2000.

[Nos98] J.T. Noseki. The Case for Collaborative Programming. In *Communications of the ACM, VOL 41 NO.3*, pages 105–108, March 1998.

[PUN01] J. Pelrine, R. Uhtes, and C. Noack. Why is it hard to learn eXtreme Programming? Philosophical and psychological aspects of learning a new methodology, 2001. http://www.netobjectdays.org/pdf/00/slides/pelrine.pdf.

[Rum01] B. Rumpe. Extreme Programming - Back to Basics? *Modellierung 2001, Workshop der Gesellschaft für Informatik e.V.(GI)*, (1):121–131, 28.-30.3.2001 2001.

[Ser01] XP Conference Series. XP 2001 Papers, 2001. http://www.xp2001.org/.

[SWR+00] P. Stahl, R. Wucher, D. Rombach, S. Hartkopf, K. Kohler, M. Friedewald, S. Kimpeler, P. Zoche, M. Broy, and I. Krüger. Analyse und Evaluation der Softwareentwicklung in Deutschland, Eine Studie für das Bundesministerium für Bildung und Forschung. 2000. http://www.dlr.de/IT/IV/Studien/evasoft_abschlussbericht.pdf.

[Ver00] G. Versteegen. *Das V-Modell in der Praxis*. dpunkt.verlag, 2000.

[WC01] L. Williams and A. Cockburn. The Costs and Benefits of Pair Programming, 2001. http://collaboration.csc.ncsu.edu/laurie/Papers/-XPSardinia.PDF.

[Web01a] Wiki Web. Projektberichte im Wiki Web, 2001. http://c2.com/cgi/wiki? ExtremeProgrammingProjects.

[Web01b] Wiki Web. XP im Wiki Web, 2001. http://xp.c2.com/ExtremeProgrammingRoadmap.html.

[Weg99] H. Wegener. Extreme Ansichten. Für und Wider des Extreme Programming. (12), 1999.

[Wel01] D. Wells. Extreme Programming, 2001. http://www.extremeprogramming.org/, http://www.extremeprogramming.org/.

[WKCJ00] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries. Strengthening the Case for Pair Programming. *IEEE Software*, 17(3), 2000.

A Anhang A: Fragebogen

Little Questionnaire on XP Projects

1. On the Company:

1.1 Name of project (not disclosed) (but important, to identify if two or more persons report about the same project):

1.2 Name and role of person who filled in this questionnaire (not disclosed):

1.3 Name of company (not disclosed):

1.4 City and country where company is located (for statistics):

1.5 Some information about the company (how big, founded when, what line of business is it in, how many other XP projects where carried out before?, ...) (for classification within statistics):

2. On the XP-Project:

2.1 Duration of project (from when till when):

2.2 Teamsize:

2.3 Total manpower e.g. in person-months?

2.4 How good was the general software engineering training/knowledge of the team members nitially?

2.5 How many team members had made experiences in XP previously?

2.6 How many development companies/independent consultants where involved?

2.7 Why did you decide to develop this project with XP?

2.8 Programming languages used:

2.9 Technologies used:

2.10 What kind of software was developed?

2.11 Has it been a development from scratch (new system), legacy maintenance or adding new functionality on an existing system?

2.12 What was the project structure (how many people where there for each role)?

Programmers (writing production code and code for component tests):
Customers:
Testers (helps customer developing functional tests):
Coach:
Further roles (consultant, big boss, tracker...)?:

2.13 How many customers with different stakes (requirements, forms of usage for the system) where involved?

2.14 Did the project terminate successfully? (9=very successfull, 0=not at all successful)

2.15 What where the major reasons for its success / failure? Can you priorize them?

2.16 If it was a success what where the main obstacles? How dangerous have they been?

2.17 XP was invented to make software development more successful. Some of its main goals are listed below. In your XP-project, could these goals be reached? If not, explain the obstacles? (5=fully achieved, 0=as always, -5=much worse)

Deliver software in time:
Let developers have fun with their work:
Develop software with a high quality (less bugs):
Let changes don't cause big costs, because one can react fast to changes:

2.18 Which XP-Elements did you use in the Project? (9=fully used, 0=not at all) Please say for every element how strong you used it (9-0) and if you consider it helpful(h), improvable(i), or even making-difficult(m) for success of development.

Planning Game:
Short Release Cycles:
Metapher:
Simple Design:
Testing:
Refactoring:
Pair Programming:

Common Code Ownership:
Continuous Integration:
40-Hour-Week:
On-Site Customer:
Coding Standards:

2.19 Please give reasons for the three least used concepts, why you didn't use them? Did you explicitly decide not to, or had there been other obstacles?

2.20 Do you have improvement suggestions for any of the XP elements (perhaps in your project you already used this elements in the way you improved it for yourself)?

2.21 Have you used additional concepts, tools or modelling languages that go beyond the pure XP approach? How did they integrate to XP?

2.22 Some comments about the project and the project progress:

2.23 Further comments:

3. Future plans and personal background

3.1 Will you use XP again?

3.2 Are you actively advocating XP in the future?

3.3 Are you trained in UML or a similar modelling language?

3.4 If you know UML, did you miss it?

3.5 Would you like to use UML combined with an XP approach, e.g. for generation of code or tests?