
**Analyse der Anforderungen an das Personal und die Organisation
beim Einsatz von Extreme Programming anhand eines Praxisbeispiels**

Gutachter:
Prof. Dr. Gregor Sandhaus

Seminararbeit vorgelegt im Rahmen des Moduls
Software-Engineering, IT-Projekt- und Qualitätsmanagement
an der

FOM
Hochschule für Oekonomie & Management
Essen

Studiengang: Master of Arts – IT-Management

Semester 2

vorgelegt von:

Peter Heim

Albertstraße 31

45894 Gelsenkirchen

Matrikel Nr. 132153

Inhaltsverzeichnis

Inhaltsverzeichnis.....	i
Abkürzungsverzeichnis	ii
Abbildungsverzeichnis	iii
1 Einleitung	1
1.1 Ausgangssituation.....	1
1.2 Zielsetzung und Vorgehensweise	2
2 Theoretische Grundlagen	3
2.1 Extreme Programming – Die Prinzipien.....	4
2.2 Extreme Programming – Die Techniken	6
2.3 Managementstrategie.....	9
3 Praxiseinsatz.....	11
3.1 Projektablauf.....	12
3.2 spezifische Projektanalyse	14
4 Fazit.....	17
5 Ausblick	18
Literaturverzeichnis.....	XIX

Abkürzungsverzeichnis

Abkürzung	Bezeichnung
GUI	Graphical User Interface
ITIL V3	Information Technology Infrastructure Library in Version 3
SQL	Structured Query Language

Abbildungsverzeichnis

Abbildung 1: Gegenseitige Stützung der Techniken	8
Abbildung 2: Schrankschema	13

1 Einleitung

1.1 Ausgangssituation

Das Unternehmen Baumann Technologie GmbH stellt einerseits IT-Dienstleistung für eine mittelständische Kunden-Zielgruppe bereit und ist andererseits der interne IT-Dienstleister einer ca. 70 Unternehmen umfassenden Unternehmensgruppe. Von Leitungsebene ist gefordert, dass das gesamte Spektrum möglicher IT-Dienstleistungen durch das Know-How eigener Mitarbeiter, dazu zählt dann auch der Zugriff auf das Spezialwissen von Mitarbeitern jeweiliger Unternehmen aus der Unternehmensgruppe, erbracht werden soll. Dabei ist jedes Unternehmen auf bestimmte Schwerpunkte spezialisiert, beispielweise kümmert sich ein eigenes Unternehmen nur um die Erstellung und Wartung von SAP-Lösungen. Die Aufgaben der Baumann Technologie GmbH liegen hauptsächlich in der Bereitstellung von Infrastrukturlösungen und der Individualsoftwareentwicklung. Die Unternehmensorganisation teilt die interne Struktur somit in eine Infrastruktur- und eine Softwareabteilung.

Im Arbeitsalltag können nun auch Anforderungen aufkommen, die von einer der genannten internen Abteilungen an die andere heran getragen werden. So lautet die aktuelle Anforderung der Infrastrukturabteilung ein neues „Service Management System“ durch die Softwareabteilung entwickeln zu lassen. Durch dieses System sollen einzelne Module, wie eine Kundenverwaltung inklusive Geräteverwaltung, ein Ticketsystem und ein Termin-Auftragssystem, bereitgestellt werden, die im Zusammenspiel schlussendlich das Monitoring von Service Level Agreements oder die Termineinhaltung von Auftragsvereinbarungen gewährleisten sollen.

Diese interne Entwicklungsanforderung soll zum Anlass genommen werden um sich mit der, durch Fachvorträge und Presse immer wieder thematisierte, agile Softwareentwicklungsmethode auseinanderzusetzen und diese kennen zu lernen.

1.2 Zielsetzung und Vorgehensweise

Eine der bekanntesten und provokantesten Ausprägungen agiler Entwicklungsmethoden ist „extreme Programming“. Ziel dieser Arbeit ist es diese Methode theoretisch kennen zu lernen und am Beispiel des internen Entwicklungsprojektes Praxiserfahrungen zu sammeln. Danach ist eine eigene Einschätzung, in wie weit diese Methode im realen Geschäftsalltag hilfreich ist und welche Anforderungen an das Personal und die Organisation gestellt werden müssen, möglich.

Das Management und die Mitarbeiter der Baumann Technologie GmbH haben „extreme Programming“ bisher noch nicht praktisch eingesetzt und so soll gleichzeitig geprüft werden welche Mitarbeiter, aus dem Entwicklungsbereich und dem Management, in der Lage sind Projekte mit dieser grundlegenden Entwicklungsmethode umzusetzen.

Im zukünftigen Geschäftsalltag ist dann zusätzlich einschätzbar für welche Kundenprojekte, weitere interne oder sogar externe, diese Methode sinnvoll ist oder ob gegebenenfalls Kundenanfragen zur Unterstützung von Projekten, wo diese Methode angewandt wird, angenommen werden können.

Im Zuge der Zielerreichung beginnt diese Arbeit mit der Thematisierung der theoretischen Grundlagen des „extreme Programming“ und den entsprechenden Abweichungen des agilen Managements zu konventionellen Managementstrategien. Im Gliederungspunkt 3 wird die konkrete Projektumsetzung mit den gemachten Erfahrungen und Beobachtungen beschrieben. Auf Grund der durch die FOM vorgegebenen Begrenzung des Umfangs kann in diesem Abschnitt nur auf die wesentlichsten Abwandlungen der Vorgehensweise, im Vergleich zu zuletzt eingesetzten Vorgehensweisen, eingegangen werden und kein detaillierter Projektplan vorgelegt werden.

Anschließend folgen das Fazit zu der gemachten Zielsetzung dieser Arbeit und ein Ausblick wie mit den gewonnenen Erkenntnissen weiter verfahren werden kann.

2 Theoretische Grundlagen

Die Anforderung an Unternehmen jeder Zeit und zügig auf sich ändernde Umgebungsvariablen reagieren zu können ist in der heutigen Zeit zu einem entscheidenden Wettbewerbsfaktor geworden. So ist zu beobachten, dass flexible Unternehmen erfolgreicher wirtschaften als Unternehmen mit starren Strukturen und einer langen Anpassungszeit.

Die Forderung der Wirtschaft nach ebenso flexiblen Strukturen in der Softwareentwicklung ist ein daraus resultierendes Ergebnis. Somit gibt es eine relativ junge aufkommende Entwicklung von sogenannten „agilen Softwareentwicklungsmethoden“, die sich zu den klassischen Entwicklungsmodellen, wie dem Wasserfall- oder dem Spiralmodell, gesellen oder diese ergänzen.

Es gibt bereits diverse agile Methoden mit eigenen detaillierten Ausprägungen, allerdings mit gemeinsamen Grundprinzipien. Wann sich ein Vorgehensmodell tatsächlich agil nennen darf ist allerdings noch nicht grundlegend definiert. Die Schwierigkeiten bei der Definition liegen darin, dass die Prozesse innerhalb der einzelnen Ansätze von nur sehr wenig gemachten Vorgaben bis hin zu festen Regeln reichen.¹

Die meisten Gemeinsamkeiten lassen sich allerdings folgendermaßen charakterisieren:²

- Prozesse sollen so schlank wie möglich gehalten werden
- Die Dokumentation wird minimiert
- Ständige Iterationen, mit jeweils lauffähigen Versionen
- Pläne werden nur für die jeweilige Iteration gemacht

¹ Vgl. (Bunse & Knethen, 2008), S.114

² Vgl. (Balzert & Ebert, 2008), S.651

2.1 Extreme Programming – Die Prinzipien

Die „extreme Programming“ genannte Entwicklungsmethode ist wahrscheinlich die bekannteste der gesamten agilen Methoden und gleichzeitig die provokanteste. Kent Beck hat im Jahre 2000 das erste Buch zu dem Thema veröffentlicht und dokumentiert dort die detaillierten Techniken, die er zusammen mit einer Reihe seiner Kollegen und der allgemeinen Software-Engineering-Gemeinde in den vorhergehenden Jahren zuerst eingesetzt und somit erfunden hat.³

Die Namensfindung der „extremen Programmierung“ (deutsche Übersetzung für „extreme Programming“) beruht darauf, dass allgemein als vernünftig anerkannte Prinzipien in extremer Weise eingesetzt werden.

Beispiele für die extremen Ausprägungen sind:⁴

- Tests finden andauernd statt
- Code Reviews finden andauernd statt
- Iterationszeiten werden verkürzt (Minuten oder Stunden statt Wochen oder Monate)

Die gesamte Methode basiert auf vier Hauptwerten, aus denen sich 15 Prinzipien ableiten. Diese vier Werte sind:⁵

- *Einfachheit*: Es sollen möglichst einfache Lösungen gefunden werden, da diese schneller und preiswerter zu entwickeln und zu warten sind.
- *Kommunikation*: Durch persönliche Gespräche werden Informationen am effektivsten ausgetauscht.
- *Feedback*: Qualitätssicherung wird durch Rückmeldungen der Anwender über die Korrektheit der neusten Systemversionen gefördert. Dabei wird zwischen Komponententests, zur Prüfung der Anwendungsfehlerfreiheit, und Aufgabentests, zur Prüfung ob die Software zur Aufgabenerledigung beiträgt, unterschieden.

³ Vgl. (Wolf, Roock, & Lippert, 2005), S.1-2

⁴ Vgl. (Beck, 2000), Einleitung xv

⁵ Vgl. (Balzert & Ebert, 2008), S.654-655

- *Mut*: Um die erstgenannten Werte erfüllen zu können ist Mut erforderlich. Einfachheit erfordert den Mut eventuell etwas wegzulassen, Kommunikation erfordert die Auseinandersetzung mit anderen Meinungen und Feedback erfordert mit negativen Resultaten umgehen zu können.

Alle vier Werte zeigen, dass das Endprodukt und die Projektbeteiligten im Mittelpunkt stehen und nicht Werkzeuge oder Prozesse. Diese müssen im Projektverlauf auch grundlegend beachtet werden um vom Einsatz von „extreme Programming“ sprechen zu dürfen.⁶

Die daraus resultierenden detaillierteren 15 Prinzipien sind die folgenden:⁷

- *Unmittelbares Feedback*: Es sollte so schnell und früh wie möglich erfolgen, denn dann ist der Lern- und Arbeitserfolg am größten.
- *Einfachheit anstreben*: Einfache Lösungen sind schneller zu entwickeln, besser zu verstehen und leichter zu ändern.
- *Inkrementelle Änderungen*: Notwendige Änderungen sollen in kleinen Schritten vorgenommen werden, um Risiken zu reduzieren.
- *Neues wollen*: Nur wer Veränderungen auch wirklich will, wird mit Fortschritten belohnt. Dafür müssen Risiken eingegangen werden.
- *Qualitätsarbeit*: Vorher festgelegte Qualitätsmaßstäbe (so hoch wie für das jeweilige Projekt erforderlich) kennen alle Projektbeteiligten und halten diese ein.
- *Lernen lehren*: Die Beteiligten sollen keine Anweisungen erhalten, sondern selbst lernen was sie benötigen und wie sie vorgehen um Ziele zu erreichen.
- *Geringe Anfangsinvestition*: So soll sich direkt auf die wichtigsten Aspekte konzentriert werden und bei eventuellem Abbruch das finanzielle Risiko eingegrenzt werden.
- *Auf Sieg spielen*: Projekte werden begonnen um gewonnen zu werden, aussichtslose Projekte werden nicht begonnen und Projekte, die nicht mehr gewonnen werden können, werden rechtzeitig abgebrochen.

⁶ Vgl. (Beck, 2000), Einleitung xvii

⁷ Vgl. (Balzert & Ebert, 2008), S. 655-656

-
- *Gezielte Experimente*: Durch gezielte Experimente soll geprüft werden ob Entwicklungsentscheidungen oder Prozessabläufe richtig oder falsch sind.
 - *Offene, ehrliche Kommunikation*: Das Verschweigen oder Verbergen von Problemen ist hinderlich für den Projekterfolg. Rückschläge werden akzeptiert und das Herstellen einer ehrlichen Kommunikation ist erforderlich.
 - *Team-Instinkte nutzen, nicht dagegen arbeiten*: Stellt sich heraus, dass im Verlauf der Programmierung das gesamte Team gegen eine vorherige Abstimmung ist, sollte dem Team-Instinkt gefolgt werden.
 - *Verantwortung übernehmen*: Das Team, oder ein einzelner, soll die Verantwortung übernehmen und nicht zugewiesen bekommen. Das steigert die Motivation erheblich.
 - *An örtliche Rahmenbedingungen anpassen*: Jedes Projekt ist individuell und somit auch die Durchführung. Das Ziel ist nicht die exakte Nachahmung von bereits durchgeführten Projekten, sondern die bestmögliche Projektdurchführung. Dafür ist auch eine Anpassung von Prinzipien gerechtfertigt.
 - *Mit leichtem Gepäck reisen*: Die Hilfsmittel sollten sorgfältig ausgewählt und flexibel einsetzbar sein.
 - *Ehrliches Messen*: Der Projektfortschritt muss immer ehrlich wieder gegeben und gemessen werden.

2.2 Extreme Programming – Die Techniken

Um die Entwickler zu unterstützen sich gemäß den Prinzipien zu verhalten wurden Entwicklungstechniken dokumentiert. Diese Techniken sind alle schon im Einzelnen bekannt, nur unterstützen sie sich beim Einsatz von „extreme Programming“ im Zusammenspiel gegenseitig. Die Schwäche eines Verfahrens wird durch die Stärken der anderen ausgeglichen.⁸

⁸ Vgl. (Beck, 2000), S.63

Die möglichen Techniken an denen sich die Projektbeteiligten orientieren können werden nachfolgend genannt und kurz beschrieben:⁹

- *Kunde vor Ort*: Kunden bzw. Anwender stehen den Entwicklern permanent zur Verfügung. Akzeptanztest übernehmen die Rolle der Anforderungsspezifikation.
- *Planungsspiel*: Im sogenannten Planungsspiel wird der Umfang des jeweils nächsten Versionsstandes direkt zwischen den Anwendern und den Entwicklern abgestimmt. Die Anwender geben die Anforderungen inklusive deren Prioritäten an und die Entwickler schätzen die Aufwände.
- *Standup-Meetings*: Tägliche kurze, maximal 15 minütige, Treffen um sich über den Projektfortschritt auszutauschen.
- *Kurze Releasezyklen*: Neue und geänderte Teilprodukte werden den Anwendern kurzfristig zur Verfügung gestellt. Die Anwender können daraufhin schnelle Bewertungen vornehmen.
- *Retrospektive*: Retrospektiven sollen Probleme und Blockaden im Softwareentwicklungsprozess ausmachen und beseitigen, indem die Projektbeteiligten nach längeren Abständen über den zurückliegenden Projektverlauf diskutieren.
- *Metapher*: Den Entwicklern werden wenige, aber klare Richtlinien für die Entwicklung vorgegeben, die eine konsistente Entwicklung des Systems unterstützt.
- *Gemeinsame Verantwortung*: Sämtlicher Quellcode darf jederzeit von jedem beteiligten Entwickler verändert werden.
- *Fortlaufende Integration*: Änderungen am System werden möglichst schnell integriert und den anderen Entwicklern zur Verfügung gestellt.
- *Programmierstandards*: Der Quellcode soll einheitlich gestaltet sein, dazu werden pragmatische Standards definiert.
- *Nachhaltiges Tempo*: Die Regelarbeitszeit muss eingehalten werden, da nur so ein hohes Maß an Kreativität und Engagement über einen längeren Zeitraum erhalten werden kann.
- *Testen*: Alles, was programmiert wurde, wird getestet.

⁹ Vgl. (Balzert & Ebert, 2008), S. 656-657

- *Einfaches Design*: Die Software soll möglichst einfach gestaltet werden, um diese leichter zu verstehen und somit einfacher testen zu können.
- *Refactoring*: Durch ständige Änderungen können anfällige Systemarchitekturen entstehen. Durch Umstrukturierungen der Architektur sollen eventuelle Defizite behoben werden.
- *Programmieren in Paaren*: Jeweils zwei Entwickler sitzen an einem Computersystem und wechseln sich bei der Entwicklung ab. Der Beobachter interveniert bei besseren Umsetzungsvorschlägen und weist auf Fehler hin.

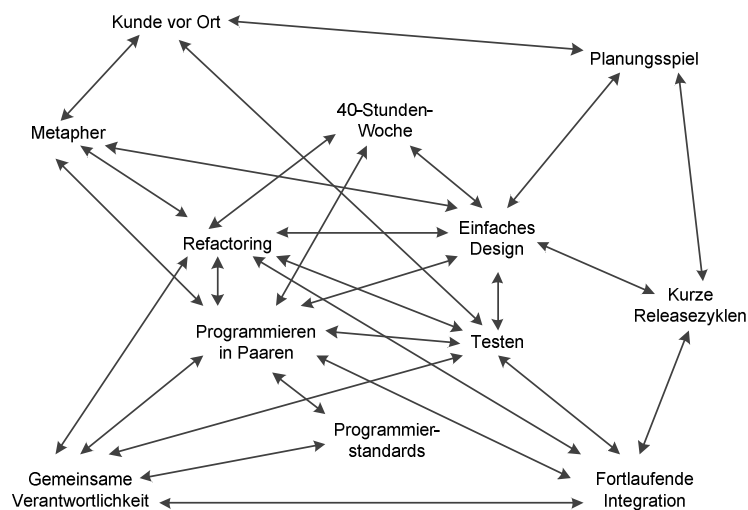


Abbildung 1: Gegenseitige Stützung der Techniken¹⁰

Abbildung 1 zeigt das Zusammenspiel und die gegenseitigen Abhängigkeiten der genannten Techniken. Je mehr Pfeilspitzen auf eine Technik verweisen, beziehungsweise von ihr abgehen, desto größer ist der Einfluss auf, beziehungsweise durch, eine Technik auf eine andere. Die Techniken mit vielen Pfeilspitzen scheinen eine entscheidende Rolle in der Methode „extreme Programmierung“ zu spielen, somit sollten diese in einem Praxistest, falls es das Projekt zulässt, bei der Betrachtung berücksichtigt werden.

¹⁰ Vgl. (Beck, 2000), S.70

2.3 Managementstrategie

Jedes Projekt benötigt eine leitende Instanz, die bei Entscheidungen die Verantwortung übernimmt und gegebenenfalls entscheidend eingreifen kann. Da es bei „extreme Programming“ immer um eine extreme Form der Umsetzung geht wäre es im Bereich des Managements einerseits das Extremum, dass ein Einzelner alle Entscheidungen trifft. Diese Strategie ist nicht erfolgsversprechend, denn eine Person kann nicht genug wissen um in allen Situationen die richtigen Entscheidungen fällen zu können. Allerdings funktioniert die umgekehrte Strategie, jeden Projektteilnehmer seine eigenen Meinungen durchsetzen zu lassen und zu vertrauen, dass sich das Projekt von alleine abschließt ohne jemals einen übergeordneten Überblick gehabt zu haben, auch nicht.

Das Management sollte daher in „extreme Programming“ Projekten einen Zwischenweg finden, der sich an den in Kapitel 2.1 genannten Prinzipien orientiert und somit meistens eher weiche Faktoren, wie soziale Kompetenz, erfordert. Mögliche Beispiele dafür sind:¹¹

- Der Projektmanager übernimmt die Verantwortung aufzuzeigen was zu tun ist, anstatt nur Aufgaben zu verteilen (Prinzip: „*Verantwortung übernehmen*“).
- Die Beziehung zwischen dem Manager und den Programmierern sollte vertrauens- und respektvoll sein, nur so kann die bestmögliche Arbeit geleistet werden (Prinzip: „*Qualitätsarbeit*“).
- Der Manager soll das Projekt dauerhaft lenkend begleiten, anstatt nur zu Beginn ein umfangreiches Verfahrenshandbuch auszugeben und dann erst zum Schluss wieder aufzutauchen (Prinzip: „*inkrementelle Änderungen*“).
- Der Manager erlangt eine führende Rolle in der Prüfung der Verträglichkeit von „extreme Programming“ mit der Firmenkultur oder der Persönlichkeitsstruktur der Teammitglieder. Entstehende Konflikte muss dieser dann frühzeitig erkennen und lösen (Prinzip: „*an örtliche Rahmenbedingungen anpassen*“).
- Die Projektfortschrittmeldungen müssen vom Manager realistisch eingeschätzt und geprüft werden (Prinzip: „*ehrliches Messen*“).

¹¹ Vgl. (Beck, 2000), S.71-72

Alles in allem ist trotz der selbständigen Organisationsstruktur in „extreme Programming“ Projekten ein Management, das gleichzeitig eine leitende Instanz verkörpert, nötig um ein Projekt erfolgreich abschließen zu können. Das Management muss seinen Führungsstil und die wahrgenommenen Aufgaben nur gegebenenfalls anpassen und sich nach den entsprechenden Prinzipien richten.

3 Praxiseinsatz

Zur Erfahrungssammlung und Analyse der theoretischen Grundlagen anhand eines Praxisbeispiels werden die, oder einige, Ansätze des „extreme Programming“ in der internen Entwicklungsanforderung der Baumann Technologie GmbH eingesetzt. Da es sich um ein größeres Softwareprojekt mit mehreren Einzelmodulen handelt, die allerdings alle auf eine zentrale Datenbank zugreifen und ihre Daten dort ablegen, wird diese Entwicklungsmethode an dem ersten Grundmodul „Terminverwaltung“ inklusive einer Workflowkomponente und der Erstellung der dahinterliegenden Datenbasis getestet. Das Endprodukt „Service-Management-System“ soll nach der Fertigstellung das gesamte Infrastrukturmanagement, nach ITIL V3, übernehmen, wie zum Beispiel Terminplanung, Ticketsystem, Änderungsanforderungen, Mitarbeiter-, Kunden- und Geräteverwaltung, Dienstleistungserfassung und –Abrechnung und Projektplanung und –Steuerung.

Das Einzelmodul „Terminverwaltung“ soll in der Lage sein persönliche Termine, z.B. Besprechungen oder Urlaub, und projektbezogene Termine, z.B. Auslieferungen und Konfigurationen, zu verwalten um beispielweise dem später zu erstellenden Ticketsystem melden zu können welcher Mitarbeiter für eine Fehlerbehebung oder Systemanpassung verfügbar ist. Dazu werden die Mitarbeiter des Unternehmens innerhalb des Systems erfasst und mit Eigenschaften, wie Kompetenzen, Vorgesetzten, Produkt- und Kundenverantwortlichkeiten versehen. Diese Daten werden dann auch für die Prüfung von Urlaubsanträgen und einer internen Revision, bei der auf Doppelbesetzung aller kritischen Fachbereiche geprüft wird, genutzt.

Als Entwicklungstechnologie wird eine in vorherigen Projekten genutzte und bekannte Umgebung genutzt, die auf „Java“ und einem „Apache“ Webserver mit darunterliegender „MySQL“ Datenbank basiert. Dies geht auch einher mit den Überlegungen, dass die eingesetzte Entwicklungssprache objektorientiert sein soll um die ständig ändernden Anforderungen umsetzen zu können, ohne das technische Grundgerüst zu gefährden und gegebenenfalls Instabilität zu erreichen. Per Webbrowser sollen die Daten dann abgerufen werden können.

3.1 Projektablauf

Bei einem ersten Projektmeeting, an dem das Projektteam (Bestehend aus vier Programmierern, einem Projektleiter und 2 Personen aus der Fachabteilung Serviceannahme und Verwaltung) teilnahmen, wurden die groben Umrisse der Gesamtsoftware und des zum jetzigen Zeitpunkt zu entwickelnden Moduls erörtert. Anhand der Anforderungen wurde eine ungefähre Umsetzungsdauer von 3 Wochen geschätzt.

Weiterhin wurden die Grundlagen des „extreme Programming“ erläutert und der Einsatz dieser Methode als Modellversuch angekündigt. Die ersten Reaktionen waren Verweigerung, seitens der Entwickler, und Befürwortung, seitens der Fachabteilung beziehungsweise dem Kunden. Die Entwickler wollten sich nicht erklären lassen wie sie Ihre Arbeit zu machen haben, wogegen die Fachabteilung begeistert war direkt in die Entwicklungen einbezogen zu sein und schon im Entwicklungsverlauf Meinungen und Wünsche äußern zu können. Nach der Erläuterung der Hintergründe, dass es ein Modellversuch und Praxis-Verifikationstest werden soll, ließen sich die Programmierer auf den Test ein.

Gemeinsam wurde ein Prozess und Basisverfahren definiert, die einige Prinzipien und Techniken des „extreme Programming“ beinhalten ohne immer daran denken zu müssen die spezifischen Eigenschaften einhalten zu müssen. Dabei wurde auch darauf geachtet nicht so viele Techniken wie möglich einzusetzen, sondern für das individuelle Projekt sinnvolle Techniken anzuwenden. Denn wie es auch die Prinzipien des „extreme Programming“ vorschreiben steht der Projekterfolg an erster Stelle und nicht der gezwungene Einsatz bestimmter Techniken (Managementprinzip: *„an örtliche Rahmenbedingungen anpassen“*).

Als Basisverfahren wurden die Arbeitsplätze der Programmierer in die Räumlichkeiten der Fachabteilung verlegt, so dass diese in direkter Sprechweite sitzen und je 2 Programmierer erhielten einen gemeinsamen Entwicklungsarbeitsplatz. Ein Programmierer tippt dabei aktiv den Quellcode, wogegen der andere diesen auf Syntax oder Denkfehler prüft und gegebenenfalls direkt korrigierend eingreift. Des Weiteren schreibt dieser parallel Testfälle für die umgesetzten Anforderungen (Techniken: *„Kunde vor Ort“* und *„Programmieren in Paaren“*). Der Projektleiter ließ, in einem

Gespräch zwischen den Entwicklern, die Zusammensetzung und die Teilung der Verantwortlichkeiten in Systemkomponenten und GUI eigenständig abstimmen (Managementprinzip: „*Verantwortung übernehmen*“). Darüber hinaus wurde ein tägliches Meeting um 9:30 Uhr in der Etagenküche vereinbart, dieses sollte für die avisierte Zeit von 15 Minuten nur für Projektthemen genutzt werden (Technik: „*Standup-Meeting*“). Weiterhin sollte die tägliche Arbeitszeit von 8 Stunden, trotz der neuen Methode, eingehalten werden (Managementprinzip: „*Nachhaltiges Tempo*“).

Nach dem Einführungsgespräch wurde von den Programmierern die Grundumgebung erstellt und auf dieser Grundlage ein grober Prototyp entwickelt. Für den Programmierprozess wurde dann eine große freie Schrankwand aus dem Projektbüro als Instrument genutzt und mit Klebeband, gemäß Abbildung 2, in die unterschiedlichen Bereiche eingeteilt.

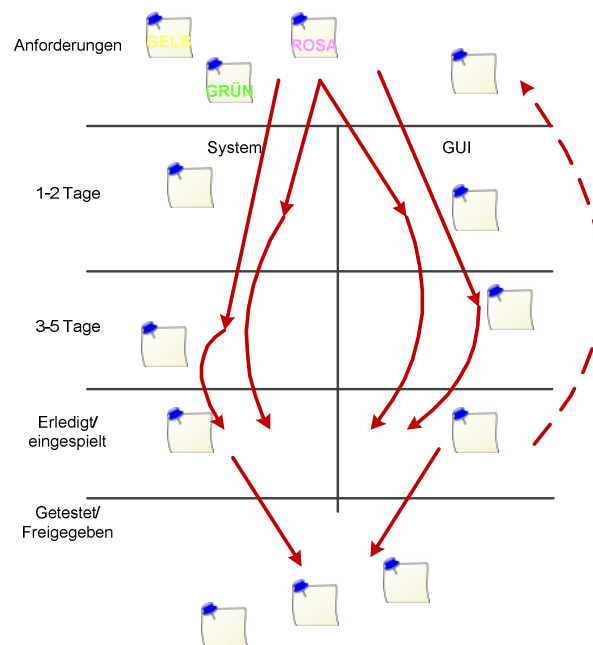


Abbildung 2: Schrankschema

Im oberen Anforderungsbereich wurden, nach der ersten Projektabstimmung, die Anforderungen des Fachbereiches auf Klebeblätter geschrieben, durchnummeriert und dort platziert. Jede Person aus dem Fachbereich hatte dabei seinen Anfangsbuchstaben als führende Angabe mit angegeben. Die Anforderungen ließen sich zudem noch, durch die Auswahl von unterschiedlichen Klebeblätternfarben, unterschiedlich priorisieren. Die Priorisierungsstufen waren: gelb = normal, rosa = dringend, grün = „nice to have“. Die

Programmierer können so jederzeit überblicken welche neuen Anforderungen mit welcher Priorität vorliegen, den jeweiligen Aufwand schätzen und im eigenen Verantwortungsbereich dem entsprechenden Aufwand zuordnen. In diesem Projekt wurden die beiden Umsetzungszeitspannen 1-2 Tage und 3-5 Tage definiert. Bei Rückfragen zu einzelnen Anforderungen sollte der direkte Kontakt zur Fachabteilung, der Anfangsbuchstabe des Erfassers war mit auf dem Klebezettel erfasst, gesucht werden und mit diesem eine Klärung erreicht werden (entspricht der Technik: „*Planungsspiel*“). Abgearbeitete Anforderungen sollten direkt eingespielt und danach in den „erledigt“ Bereich geklebt werden (Technik: „*fortlaufende Integration*“). Die bei der Entwicklung erstellten Testfälle wurden mit führender Anforderungsnummer auf einem zentralen Dokumentenmanagementserver veröffentlicht. Der Fachbereich konnte dann die umgesetzten und eingespielten Anforderungswünsche testen und bei einer Freigabe vom „erledigt“- in den „getestet“-Bereich überführen. Wurde diese Anforderung nicht zur Zufriedenheit umgesetzt oder enthielt Fehler wurde der Klebezettel wieder zurück in den Anforderungsbereich zurückgeführt und durch den Programmierprozess getragen bis diese und alle anderen Klebezettel im letzten Abschnitt angekommen und als getestet freigegeben wurden (Technik: „*kontinuierliches Testen*“). Zum Abschluss fand die Dokumentation des erstellten Moduls statt.

3.2 spezifische Projektanalyse

Das Projekt konnte, unter Einsatz der „extreme Programming“ Techniken und Prinzipien abgeschlossen werden. Durch den Einsatz der Schrankwand als Prozessinstrument wurde den Beteiligten die Methode auf einfache Weise vermittelt und eine Umsetzung ohne Rückfragen, wie in bestimmten Situationen in dieser Methode weiter zu verfahren ist, erreicht. Weiterhin konnten so die Techniken im Zusammenspiel, also unter gegenseitiger Stärkung, eingesetzt werden. Alle Projektbeteiligten bemerkten, dass diese Methode bestimmte andere Anforderungen an das Personal und die Organisation aufweist als diese gewohnt waren.

Folgende erweiternde Anforderungen, an die herkömmlichen Methoden oder theoretischen Anforderungen des „extreme Programming“, ergaben sich an das Personal der Fachabteilung:

- Die dort eingesetzten Personen müssen wissen und formulieren können wie die Anforderungen sind.
- Durch das zeit- und aufmerksamkeitsintensive ständige Formulieren von Anforderungen und das Testen der Umsetzung ist eine parallele Partizipation am normalen Geschäftsbetrieb nicht möglich.

Allerdings hat die direkte Beteiligung am Entwicklungsprozess die Fachabteilung besser einschätzen lassen wie viel Aufwand hinter einer, eventuell einfach formulierten, Anforderung steht. Geänderte Anforderungen an die Entwicklerarbeit waren:

- Das Arbeiten am Quellcode in Zweiertteams erfordert ein Umdenken und ist nicht für jeden Entwickler eine machbare Aufgabe, da man aus den gewohnten Denkstrukturen ausweichen muss. Das Verhältnis zwischen den Teammitgliedern muss auch sehr gut sein, da es eine sehr enge Zusammenarbeit bedeutet.
- Der regelmäßige Wechsel der Arbeitsgeräte (Maus und Tastatur) funktionierte zunächst nur nach Anmahnung des Projektleiters. Nach einigen Mahnungen wurde zeitweise tatsächlich eine Stoppuhr auf 15-30 Minuten Zeitintervalle gestellt. Die unregelmäßige Weitergabe resultierte aus den unterschiedlichen Qualifikationen. Der erfahrenere Programmierer wollte zunächst mehr im Vordergrund stehen um später weniger Probleme hervorzurufen, Diskussionen im Entwicklerteam zu vermeiden und einfach schneller zu sein.
- Der direkte Dialog mit der Fachabteilung muss von Entwicklern, die Computer gewöhnt sind, geübt werden. Da dort auch andere Meinungen zugelassen werden müssen und in Diskussionen auch einmal Niederlagen eingesteckt werden müssen ohne später verärgert zu sein.
- Beim Planungsspiel musste eine bidirektionale Kommunikation unterbrochen werden, da bei Fragen an die Entwickler diese andauernd aus ihren aktuellen Denkansätzen gerissen wurden. So wurden die Anforderungen zunächst bestmöglich formuliert und dann gegebenenfalls von den Programmierern bei der Fachabteilung angefragt. Rückmeldungen durch die Fachabteilung geschahen nur durch eine Interaktion mit den Zetteln auf der Schrankwand.

-
- Um bei der hohen Anzahl von Tests wenige Wiederholungsrunden zu erreichen muss von Beginn an einwandfreie Arbeit geleistet werden, dies wurde durch den zweiten Programmierer meistens erreicht.

Die abgewandelten Anforderungen waren, laut den Aussagen der Programmierer, zwar zeitweise mit mehr Aufwand, aber im Endeffekt wahrscheinlich mit weniger Nacharbeit verbunden. Überblickend werden diese aber als anstrengender wahrgenommen, als sich alleine mit einem Thema auseinanderzusetzen und kurze Pausen einlegen zu können ohne die Arbeit mehrerer Projektbeteiligter zu beeinflussen.

Das Management ist weniger straff, gibt nur einen groben Rahmen vor und setzt zusätzlich hauptsächlich auf sozialer Kompetenz auf. Im Folgenden werden die geänderten Anforderungen an die Organisation dargestellt:

- Das Stand-Up Meeting konnte nach den ersten Treffen bereits abgeschafft werden, da Gespräche und Rückfragen meisten direkt geführt wurden und es dort keine Zusatzinformationen mehr auszutauschen gab.
- Auf Einhaltung der Arbeitszeit musste weniger Wert gelegt werden, da die Zusatzanstrengungen einen rechtzeitigen Feierabend von alleine hervorriefen.

Die erstellte Übersichtsschrankwand konnte immer einen guten Stand aussagen und dem Projektleiter Gefahren für einen Projektstillstand aufzeigen. So konnte er gezielte Rückfragen stellen und gegebenenfalls zwischen den Projektteilnehmern vermitteln, oder Anforderungen verwerfen lassen um den Projektfortschritt nicht zu gefährden. Dies erfordert aber eine regelmäßige Partizipation innerhalb des Projektes, im vorliegenden Projekt war dies, je nach aktuellem quantitativem Anforderungsumfang, mehrmals täglich.

4 Fazit

Durch den Abschluss des internen Entwicklungsprojektes konnte verifiziert werden, dass die theoretischen Vorgaben von „extreme Programming“ auch in der Praxis zum Erfolg führen können.

Mit den vier Hauptwerten lassen sich die Merkmale dieser agilen Methode treffend beschreiben, denn durch die *Einfachheit* kann Software schneller und kostengünstiger entwickelt werden, durch die starke *Kommunikation* wird eine sehr enge Zusammenarbeit der Projektbeteiligten gefordert, das ständige *Feedback* und der Austausch innerhalb des Projektes legt nahe, dass diese Methode auf kleine Projektgruppen ausgelegt ist. So konnte auch im Praxisversuch festgestellt werden, dass bei noch größeren Gruppen ein ständiges Feedback den Projektfortschritt eher bremsen kann. Schlussendlich benötigt jeder Projektteilnehmer *Mut* seine Arbeitsgewohnheiten zu ändern. Die Entwickler müssen die ständige Zusammenarbeit akzeptieren, die Kunden müssen sich ihrer Anforderungen immer klar sein und das Management muss weniger fachlich führen, sondern eher auf sozialer Ebene den Projekterfolg anstreben.

Die beim Praxisversuch eingesetzten Testpersonen wären alle in der Lage ein Projekt, unter Einsatz der „extreme Programming“ Methode, umzusetzen. Mögliche Kundenprojekte können somit, an Hand von den gemachten Erfahrungen, angegangen und der Sinn des Einsatzes von „extreme Programming“ bei der Umsetzung bewertet werden. Dabei muss beachtet werden, dass die Übertragbarkeit des internen Entwicklungsprojektes gegebenenfalls in realen Kundenumgebungen, die eventuell nicht über entsprechendes technisches Know-How verfügen, nicht komplett gewährleistet sein muss.

Der finanzielle Aspekt konnte in dem vorliegenden Beispiel nicht analysiert werden, allerdings liegt die Vermutung nahe, dass bei vorher nur grob abschätzbaren Anforderungen eine Abrechnung nach Stundenaufwand, statt nach Festpreis, sinnvoller ist.

5 Ausblick

Derzeit ist ein Trend in Richtung der agilen Softwareentwicklungsmethoden zu erkennen, deshalb wird dieses Thema auch zukünftig interessant bleiben.

Die Baumann Technologie GmbH hat sich entschlossen die Methode „extreme Programming“ weiter zu verfolgen und im nächsten Kundenprojekt, sofern die Eignung für das Projekt zutreffen sein könnte, auszutesten. Dabei soll, falls möglich, ein EDV gestütztes Instrument zur Prozessabbildung genutzt werden und nicht mehr auf eine Schrankwand zurückgegriffen werden. So könnte auch nachverfolgt werden wer etwas wann als Anforderung eingestellt hat und wie der Vorgang dann bis zum Ende des Tests fortgeführt wurde. Mit dieser Analyse könnten Schwachstellen im Prozess ausgemacht werden und Optimierungsansätze geliefert werden.

Einige der Ansätze, die Kent Beck in seiner ersten Veröffentlichung aufgezeigt hat, sind heutzutage allgemein akzeptiert und gar nicht mehr als extrem anzusehen, so hat er bereits eine nächste Variante „extreme Programming 2“ veröffentlicht. Diese ergänzt die erste Veröffentlichung um den weiteren Hauptwert Respekt und Menschlichkeit, weiterhin gibt es weitere „extreme“ Techniken, wie zum Beispiel die Erstellung von Tests vor der Programmierung, nur der Quellcode und die Tests werden als Dokumentation genutzt und die Abrechnung der Software wird nicht per Release sondern pro tatsächlicher Benutzung durchgeführt¹².

Gerade der letztgenannte Punkt ist in einer heutigen technischen Entwicklung, wo Dienste „aus der Wolke“ (Cloud Services, Software as a service, etc.) nach tatsächlich benötigtem Bedarf bezogen werden können sehr interessant. Eine Analyse der neuen Veröffentlichung ist innerhalb der Baumann Technologie GmbH eventuell schon bei der internen Programmierung des nächsten Moduls des „Service-Management-Systems“ geplant.

¹² Vgl. (Balzert & Ebert, 2008), S. 662-666

Literaturverzeichnis

Balzert, H., & Ebert, C. (2008). *Softwaremanagement*. Heidelberg: Spektrum Akademischer Verlag.

Beck, K. (2000). *Extreme programming*. München: Addison-Wesley Verlag.

Bunse, C., & Knethen, A. v. (2008). *Vorgehensmodelle kompakt*. Heidelberg: Spektrum Akademischer Verlag.

Wolf, H., Roock, S., & Lippert, M. (2005). *eXtreme Programming*. Heidelberg: dpunkt.Verlag.