Entwicklungsprozessse

Hauke Neemann

Abstract: Diese Arbeit untersucht unterschiedliche Vorgehensmodelle im Bereich der Softwareentwicklung. Dabei soll für ein Projekt mit einem festen Auslieferungstermin, 10 Projektmitgliedern mit geringer Erfahrung und flexiblen Anforderungen das sinnvollste Vorgehensmodell identifiziert werden. Da es eine große Anzahl an Vorgehensmodellen gibt, wird eine Auswahl anhand einer Klassifizierung erhoben. Aus dieser kleinen Auswahl werden konkrete Vertreter ausführlich beschrieben und miteinander verglichen, um das sinnvollste Vorgehensmodell zu ídentifizieren.

1 Einleitung

Projekte sind meist sehr komplex, denn sie überdauern einen langen Zeitraum und viele Menschen sind an ihrer Umsetzung beteiligt. Um die Prozesse innerhalb eines Projektes möglichst effektiv durchführen zu können, müssen diese strukturiert werden. Diese Struktur muss nicht neu erfunden werden, denn es gibt bereits eine große Anzahl an Vorgehensmodellen, so dass Projektleiter mittlerweile aus einer Vielzahl an Vorgehensmodellen wählen können.

Das Ziel dieser Ausarbeitung ist die Identifikation eines passenden Vorgehensmodells für die Projektgruppe StreamCars, wobei die Merkmale der Projektgruppe berücksichtigt werden. So gibt es einen festen zeitlichen Projektrahmen von einem Jahr und eine personelle Kapazität von 10 Personen. Alle Personen befinden sich noch in ihrer Ausbildung, weshalb das Team über keine große Erfahrung verfügt. Ein grober funktionaler Projektrahmen wurde zwar vorgegeben, aber die konkreten Anforderungen sind nicht klar bzw. flexibel.

Um eine Vorauswahl von Vorgehensmodellen zu treffen, werden in Abschnitt 2 zunächst die unterschiedlichen Familien von Vorgehensmodellen beschrieben. Durch diese Klassifikation lassen sich einige wenige konkrete Vorgehensmodelle identifizieren, die in Abschnitt 3 genauer untersucht werden. Schließlich werden diese miteinander verglichen, um das sinnvollste Vorgehensmodell auszuwählen.

2 Grundlagen

Vorgehensmodelle lassen sich mittels gemeinsamer Eigenschaften klassifizieren und in Familien einordnen. In der Literatur findet sich die Klassifizierung nach sequenziellen, prototypischen, wiederholenden und wiederverwendungsorientierten Vorgehensmodellen ([BvK08]). Um einen Überblick über diese Familien und die Eigenschaften ihrer Vertreter

zu geben, werden diese in diesem Abschnitt beschrieben.

2.1 Sequenzielle Vorgehensmodelle

Sequenzielle Vorgehensmodelle teilen den Entwicklungsprozess in unterschiedliche Phasen (wie beispielsweise Analyse, Entwurf, Implementierung, etc.), die sequenziell nacheinander abgearbeitet werden. Hier wird eine nachfolgende Phase erst nach dem Abschluss der vorherigen begonnen. Dabei sind Rückschritte in vorangegangene Phasen als Ausnahmen zu betrachten und damit nur in seltenen Fällen zu nutzen ([BvK08]).

Vorteile Ein Vorteil der sequenziellen Vorgehensmodelle ist, dass es wenige Probleme bei der Integration der Teilsysteme gibt, da zunächst eine vollständige Anforderungsanalyse durchgeführt wird und somit alle Anforderungen im Entwurf berücksichtigt werden ([BvK08]). Ein weiterer Vorteil im Vergleich zu anderen Familien von Vorgehensmodellen ist die geringe Anzahl an Versionen. Diese ergibt sich dadurch, dass es keine Rückschritte in vorangegangene Phasen gibt und somit jede Phase im günstigsten Fall eine Version produziert ([BvK08]). Dadurch ist ein weniger umfangreiches Versions- und Konfigurationsmanagement erforderlich, als beispielsweise bei iterativen Vorgehensmodellen.

Vorraussetzungen Vorraussetzungen für den Einsatz von sequenziellen Vorgehensmodellen sind verständliche Anforderungen und große Erfahrung im Umgang mit den verwendeten Entwicklungstechniken sowie mit der Abschätzung der Projektkosten ([BvK08]). Zusätzlich müssen die Anforderungen vollständig sein, damit das Anforderungsdokument bereits zu einem frühen Zeitpunkt des Projekts fertiggestellt werden kann. Verständliche Anforderungen sind beschreibbar, stabil und eindeutig verstehbar. Große Erfahrung der Entwickler über die verwendeten Entwicklungstechniken und große Erfahrung der Projektleiter über die Abschätzung der Projektkosten sind notwendig, da die Erfahrungen aus einem Projekt nicht für das selbe Projekt genutzt werden können ([BvK08]). Dies liegt daran, dass Erfahrungen aus einer Phase erst nach Beendigung dieser vorhanden sind und Rückschritte (in der Regel) nicht erlaubt sind.

Sequenzielle Vorgehensmodelle sollten nicht eingesetzt werden, wenn es einen festen Auslieferungstermin gibt ([BvK08]). Da jede Phase vollständig durchgeführt wird, kann es bei einer Termin- oder Aufwandsüberschreitung dazu kommen, dass es beim Auslieferungstermin kein ausführbares System gibt ([BvK08]). Außerdem sollten Vorgehensmodelle dieser Familie nicht eingesetzt werden, wenn sich die Anforderungen häufig ändern oder nicht stabil sind, da entweder die Analysephase sehr spät verlassen wird oder Rückschritte in die Analysephase notwendig sind. Wenn Änderungen an dem Anforderungsdokument gemacht werden, müssen alle weiteren Phasen nocheinmal durchlaufen werden, wodurch die Kosten für das Projekt erheblich steigen ([BvK08]).

Ein bekannter Vertreter dieser Familie ist das Wasserfallmodell ([Roy70]).

2.2 Prototypische Vorgehensmodelle

Prototypische Vorgehensmodelle sind ein Spezialfall der sequenziellen Vorgehensmodelle, denn sie teilen den Entwicklungsprozess ebenfalls in unterschiedliche Phasen, die nacheinander abgearbeitet werden. Allerdings lassen sie Rückschritte zu bestimmten Zeitpunkten zu, an denen Prototypen entwickelt werden sollen, d.h. funktional eingeschränkte Modelle oder Versionen des zu entwickelnden Systems. Rückschlüsse aus diesen Prototypen dienen der Anpassung der jeweils betroffenden Dokumente zu einem frühen Zeitpunkt im Entwicklungsprozess ([BvK08]).

Vorteile Die Identifikation unklarer oder schwieriger Anforderungen und ihre Klarstellung zu einem frühen Zeitpunkt ist ein wichtiger Vorteil. Auch unklare oder schwierige Entwurfsaspekte können auf diese Weise schnell identifiziert werden. Die Prototypen steigern somit sowohl die Qualität der funktionalen Anforderungen, als auch die Qualität der nicht-funktionalen und der Qualitätsanforderungen. Die Vorteile der unproblematischen Integration und der geringen Anzahl an Versionen der sequenziellen Vorgehensmodelle gelten für prototypische Vorgehensmodelle ebenfalls ([BvK08]).

Vorraussetzungen Prototypische Vorgehensmodelle sollten eingesetzt werden, wenn einige Anforderungen an das zu entwickelnde System unklar sind, da die Prototypen Unklarheiten beseitigen und somit das Risiko eines Projektfehlschlags verringern. Insbesondere wenn ein ausführbarer Prototyp entwickelt werden soll, müssen die technologischen Möglichkeiten zur Realisierung gegeben sein und im Idealfall ein ähnliches System vorhanden sein, das nur angepasst werden muss ([BvK08]). Bei den Vorraussetzungen gelten ebenfalls die Vorraussetzungen der sequenziellen Vorgehensmodelle, d.h. verständliche Anforderungen und große Erfahrung im Umgang mit den verwendeten Entwicklungstechniken sowie die Abschätzung der Projektkosten ([BvK08]).

Prototypische Vorgehensmodelle sollten nicht eingesetzt werden, wenn es einen festen Auslieferungstermin gibt oder wenn sich die Anforderungen häufig ändern (siehe Nachteile im Abschnitt 2.1 über die Vorraussetzungen der sequenziellen Vorgehensmodelle).

Ein typischer Vertreter dieser Familie ist das Spiralmodell von Boehm ([Boe88]).

2.3 Wiederholende Vorgehensmodelle

Die wiederholenden Vorgehensmodelle teilen den Entwicklungsprozess in unterschiedliche Inkremente, in denen jeweils wie bei den sequenziellen Vorgehensmodellen einzelne Aktivitäten (Analyse, Entwurf, Implementierung, etc.) durchgeführt werden. Allerdings wird im ersten Inkrement nur eine Teilmenge aller Anforderungen erhoben und diese entwickelt. In jedem weiteren Inkrement werden wieder die einzelnen Aktivitäten für die neu hinzugekommenen Anforderungen durchlaufen und somit stellt jedes Inkrement eine Erweiterung des vorherigen dar. Auch innerhalb eines Inkrements sind Rückschritte

vorgesehen ([BvK08]).

Vorteile Der wichtigste Vorteil dieses Vorgehens ist seine Flexibilität, denn auf geänderte oder neue Anforderungen kann dynamisch reagiert werden. Des Weiteren entsteht zu einem frühen Zeitpunkt (nach dem ersten Inkrement) ein ausführbares System, das dem Kunden als Ergebnis gezeigt werden kann. Auf diese Weise bekommt der Kunde frühzeitiges Feedback und die Anforderungen an das Produkt können validiert werden ([BvK08]).

Vorraussetzungen Wiederholende Vorgehensmodelle sollten eingesetzt werden, wenn einige Anforderungen an das zu entwickelnde System unklar oder instabil sind. Zum Einen entsteht zu einem frühen Zeitpunkt ein ausführbares System, zum Anderen festigen sich unter Umständen die unklaren Anforderungen nachdem das erste Inkrement entwickelt wurde ([BvK08]). Vorraussetzung für den Einsatz von wiederholenden Vorgehensmodellen ist, dass sich die Anforderungen sinnvoll in Teilmengen einteilen lassen, denn ansonsten ist die Entwicklung in Inkrementen nicht möglich. Außerdem ist der Einsatz sinnvoll, wenn die Erfahrungen mit den verwendeten Entwicklungstechniken oder die Erfahrung des Entwicklungsteams gering ist, da Erfahrungen aus einem Inkrement in allen Folgeinkrementen weiterverwendet werden können. Schließlich lässt sich ein fester Auslieferungstermin leicht einhalten, indem die Anforderungen priorisiert werden. Auf diese Weise wird sichergestellt, dass zumindest die wichtigsten Anforderungen bis zu dem Auslieferungstermin umgesetzt wurden ([BvK08]).

Vertreter dieser Familie sind beispielsweise alle agilen Vorgehensmodelle wie Extreme Programming ([Bec00]), Scrum ([Sch04]) und Crystal ([Coc05]) oder auch der Unified Process ([Kru00]).

2.4 Wiederverwendungsorientierte Vorgehensmodelle

Wiederverwendungsorientierte Vorgehensmodelle haben die Wiederverwendung von Softwarekomponenten sowohl in einem Projekt, als auch projektübergreifend zum Ziel. So fällt eine bestimmte Zeit der Entwicklung auf genau dieses Ziel, d.h. in einem Projekt werden bestimmte Kapazitäten für die Extraktion bestimmter Softwarekomponenten bereitgestellt.

Ein Vorteil dieser Vorgehensmodelle ist die Vermeidung von Fehlern, indem Erfahrungen und Muster aus alten Projekten genutzt werden. So können beispielsweise Prototypen schneller hergestellt werden. Bei diesen Vorgehensmodellen ist die Integration allerdings häufig schwieriger, da vorgefertigete Softwarekomponenten nicht exakt die Funktionalität erfüllen, die in einem konkreten Projekt benötigt werden ([BvK08]).

Der Einsatz von wiederverwendungsorientierten Vorgehensmodellen hat vor allen Dingen einen langfristigen Nutzen, wenn in einem Unternehmen ähnliche Softwareprojekte erstellt werden ([BvK08]).

Der Einsatz sollte vermieden werden, wenn vom Typ her unterschiedliche Software entwickelt wird.

Typische Vertreter dieser Familie sind Foda ([KCH+90]) oder Fast ([Har02]).

3 Ausgewählte Vorgehensmodelle

Da im Rahmen dieser Arbeit nicht alle existierenden Vorgehensmodelle detailliert beschrieben werden können, muss eine sinnvolle Auswahl an Vorgehensmodellen selektiert werden. Im folgenden Abschnitt werden zunächst die für die Projektgruppe „StreamCars" sinnvollen Vorgehensmodelle anhand der Familien von Vorgehensmodellen aus dem letzten Abschnitt identifiziert (3.1). Dazu werden die Eigenschaften des Projektes beschrieben und mit den dafür empfohlenen Familien von Vorgehensmodellen verglichen. Anschließend werden bekannte Vertreter der passenden Familien ausgewählt und miteinander verglichen.

3.1 Identifikation sinnvoller Vorgehensmodelle

Das Projekt, das im Zuge der Projektgruppe „StreamCars" umgesetzt wird, hat einen zeitlichen Rahmen von einem Jahr und kann anschließend (zumindest in dieser Besetzung) nicht erweitert werden, d.h. es besteht ein fester Auslieferungstermin. Um dieses Projekt umzusetzen stehen 10 Personen zur Verfügung, die sich alle noch in ihrer Ausbildungsphase befinden. Sie verfügen also nicht über große Erfahrung. Die Anforderungen, die an das Projekt gestellt werden, sind zunächst nicht klar, bzw. variabel. So gibt es zwar einen funktionalen Projektrahmen, der die grobe Struktur des Projekts vorgibt, allerdings steht noch nicht fest, ob das Projekt mit einem erfolgreichen System abgeschlossen werden kann.

Da das Projekt einen festen Auslieferungstermin und flexible Anforderungen besitzt, sollten sequenzielle und rein prototypische Vorgehensmodelle nicht genutzt werden. Bei der Nutzung dieser könnte es unter Umständen dazu kommen, dass es am Auslieferungstermin kein ausführbares System gibt, da jede Phase vollständig abgeschlossen wird. Außerdem führen flexible Anforderungen dazu, dass entweder die Analysephase erst sehr spät verlassen wird oder Rücksprünge in diese nötig sind. Dadurch müsste jede Phase erneut durchlaufen werden. Bei wiederverwendungsorientierten Vorgehensmodellen ist der Mehraufwand für die Erstellung von wiederverwendbaren Komponenten für dieses Projekt nicht sinnvoll, da es ein Forschungsprojekt ist, dessen primäres Ziel die Überprüfung der Möglichkeit einer effektiven Kommunikation zwischen den beiden Frameworks Odysseus und Dominion ist. Somit sollte ein wiederholendes Vorgehensmodell verwendet werden, da es genau die Anforderungen des Projekts erfüllt. Durch die inkrementellen Iterationen steht mit großer Wahrscheinlichkeit am Auslieferungstermin ein ausführbares System zur Verfügung und auf sich ändernde Anforderungen kann auch eingegangen werden. Die Anforderungen des Projekts lassen sich außerdem mindestens in die Teilmengen „Odysseus" und „Dominion" einteilen (feingranularere Unterteilungen sind wahrscheinlich). Zusätzlich lernt ein unerfahrenes Team durch das inkrementelle Vorgehen, da auf Erfahrungen aus vorangegangenen Iterationen aufgebaut werden kann.

Den wiederholenden Vorgehensmodellen sind sehr viele Vorgehensmodelle zuzuordnen, weshalb auch hier wiederum eine Auswahl nötig ist. Um mit den flexiblen Anforderungen im Projekt umgehen zu können, sollten agile Methoden und Prozesse verwendet werden. Agile Softwareentwicklung setzt laut Beck, Beedle, Cockburn etc. auf Kundenzufriedenheit, Änderung von Anforderungen, Kommunikation, Motivation der Mitarbeiter, Einfachheit der Lösungen[1]. In Abbildung 1 ist eine Einteilung der agilen Vorgehensmodelle nach Flexibilität und Detailgrad zu erkennen.

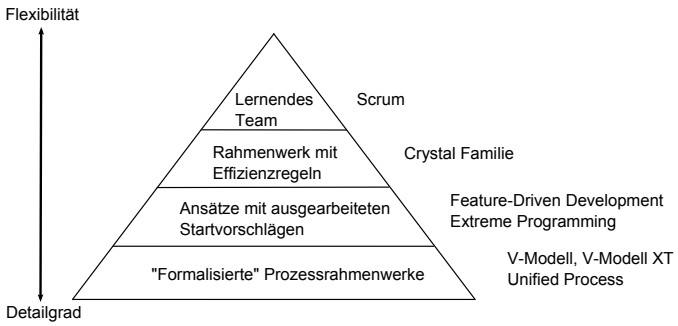

Abbildung 1: Agile Vorgehensmodelle ([BvK08])

Die drei bekannten agilen Vorgehensmodelle „Rational Unified Process", „Extreme Programming" und „Scrum" werden in den folgenden Abschnitten erläutert. Dazu wird jeweils die Vorgehensweise und die Rollen im Vorgehensmodell erläutert. Eine Rolle ist dabei eine abstrakte Person, die bestimmte Merkmale besitzt und bestimmte Aufgaben durchführen muss. Mehrere Personen können in einem Projekt die gleiche Rolle haben, wie beispielsweise die Rolle „Entwickler".

3.2 Rational Unified Process (RUP)

Der RUP definiert einen generischen Projektrahmen, d.h. dieser Rahmen kann auf unterschiedliche Projekte angepasst werden.

Vorgehensweise Der RUP teilt den Entwicklungsprozess in mehrere Zyklen, wobei jeder Zyklus mit einem ausgelieferten Produkt endet. Jeder Zyklus besteht aus den vier Phasen Konzeptualisierung, Entwurf, Konstruktion und Übergang ([Kru00]).

In der Konzeptualisierungsphase werden die zu unterstützenden Geschäftsvorfälle und das Endprodukt des Zyklus beschrieben und der Produktumfang festgelegt, so dass mehrere Artefakte erstellt werden können. Neben einer Vision, in der die wesentlichen Projektanforderungen und Bedingungen stehen, müssen ein initiales Anwendungsfallmodell, ein Glossar, ein initialer Projektplan und ein initiales Domänenmodell erstellt werden. Das

[1] http://agilemanifesto.org/principles.html

komplette Dokument dient dazu allen Stakeholdern zunächst ein gemeinsames Verständnis über das Projekt zu geben ([Kru00]).

In der Entwurfsphase wird das Dokument aus der Konzeptualisierungsphase ausgearbeitet. Dabei werden die Aktivitäten und Ressourcen zur Softwareerstellung geplant, die Softwarearchitektur aufgestellt und die zu realisierenden Softwareeigenschaften festgelegt ([Kru00]).

In der Konstruktionsphase werden alle noch nicht eingebauten Komponenten und Produktfeatures des aktuellen Zyklus entwickelt und zu dem Produkt hinzugefügt. In dieser Phase entstehen aus den erstellten Dokumenten ausführbare Systeme, die anschließend auf ihre Funktionalität getestet werden, d.h. die Anforderungen werden validiert ([Kru00]).

Die Übergangsphase dient der Übergabe des Produktes an den Auftraggeber. Dabei werden zunächst Betatests durchgeführt um eventuelle Fehler aufzudecken, wobei das System parallel mit dem Altsystem läuft. Anschließend wird das Produkt an die interne Marketing-, Verteilungs- und Verkaufsabteilung weitergegeben oder direkt an den Auftraggeber geliefert ([Kru00]).

Rollen Kruchten führt in [Kru00] drei unterschiedliche Rollen beispielhaft auf, wobei er jedoch anmerkt, dass diese weiter aufgeteilt werden können (wie beispielsweise den Systemanalytiker in Anwendungsfallautor und Anwendungsfalldesigner):

Systemanalytiker Der Systemanalytiker führt die Anforderungserhebung durch und stellt Anwendungsfalldiagramme auf, indem er die Funktionalität des Systems analysiert.

Designer Der Designer entwickelt die Klassen und entscheidet wie sie in die bisherige Implementierung eingebunden werden.

Test Designer Der Testdesigner ist für das Entwickeln sinnvoller Tests und ihre Evaluation verantwortlich.

3.3 Scrum

Scrum bietet Entwicklern einen abstrakten Entwicklungsrahmen. Der Fokus liegt dabei in der Priorisierung der Aufgaben, der Verbesserung der Nützlichkeit der Ergebnisse und der Erzielung von Gewinnen ([BvK08]).

Vorgehensweise Der Entwicklungsprozess wird bei Scrum in drei unterschiedliche Phasen eingeteilt, die sich im gesamten Projekt wiederholen: Pre-Sprint, Sprint und Post-Sprint ([Sch04]).

In der ersten Pre-Sprint-Phase werden die initialen Anforderungen erhoben und in das Product Backlog geschrieben. In diesem werden alle Anforderungen gesammelt und es kann von allen Entwicklern zu jedem Zeitpunkt erweitert werden ([Sch04]). Nachdem die

Anforderungen erhoben wurden, wird zunächst die Architektur entwickelt und ein Hauptarchitekt bestimmt. Der Hauptarchitekt entwickelt die Vision für den Entwicklungsprozess und sorgt im kompletten Entwicklungsprozess dafür, dass diese konsistent gehalten wird ([RJ00]). Die Architektur stellt nur einen Leitfaden dar, denn sie kann im Laufe des Projekts angepasst werden. Des Weiteren wird in der Pre-Sprint-Phase das Ziel der nächsten Iteration (des Sprints) festgelegt, indem Anforderungen aus dem Product Backlog in das Release Backlog übernommen und priorisiert werden. Alle Anforderungen, die im Release Backlog stehen, müssen im nächsten Sprint umgesetzt werden ([Sch04]). Da die Anforderungen im Release Backlog teilweise vom Kunden stammen und bestimmte Funktionen beschreiben, müssen diese in konkrete Aufgaben für die Entwickler übersetzt werden. Diese konkreten Aufgaben werden mit Kennzeichnung ihrer Priorität in das Sprint Backlog geschrieben. Eine Aufgabe aus dem Sprint Backlog sollte dabei einen Arbeitsumfang von 16 Stunden, d.h. zwei Arbeitstagen nicht überschreiten ([Sch04]).

Während der Sprint-Phase werden die Aufgaben aus dem Sprint Backlog umgesetzt. Der Experte (Scrum Master) kümmert sich dabei darum, dass die Aufgaben während eines Sprints nicht mehr von außen geändert werden, um eine Stabilisierung des Projekts zu gewährleisten. Des Weiteren finden tägliche Treffen des Entwicklerteams statt, um allen Beteiligten einen Überblick über den Fortschritt des Entwicklungsprozesses zu geben. Jeder Entwickler beantwortet dabei folgende Fragen [RJ00]:

1. Welche Aufgaben aus dem Sprint Backlog hast du seit dem letzten Treffen erledigt?
2. Welche Probleme gab es bei der Lösung der Aufgaben?
3. Welche Aufgaben aus dem Sprint Backlog willst du bis zum nächsten Treffen erledigen?

Die aufgezählten Probleme werden dabei nicht während dieses Treffens gelöst, sondern eventuell nach dem Treffen mit allen Beteiligten, die einen Lösungsvorschlag anzubringen haben.

Die Post-Sprint-Phase ist ein informatives Treffen nach dem Release, an dem der letzte Sprint analysiert wird. Dabei werden Probleme/Hindernisse identifiziert und in einem Impediment Backlog festgehalten, um aus diesen für weitere Sprints zu lernen. Außerdem werden neu identifizierte Anforderungen in das Product Backlog eingetragen. Nach der Post-Sprint-Phase, beginnt der Zyklus erneut mit der Pre-Sprint-Phase ([Sch04]).

Rollen Bunse und von Knethen identifizieren drei unterschiedliche Rollen ([BvK08]), wobei diese Unterscheidung sehr grob ist und ggfs. unterteilt werden kann.

Kunde Der Kunde liefert alle Anforderungen an das zu entwickelnde Projekt.

Team Das Team entwickelt das Projekt, indem es die unterschiedlichen Listen pflegt und die Aufgaben aus dem Sprint Backlog umsetzt.

Experte Der Experte übernimmt keine Entwicklungsaufgaben, sondern koordiniert die Entwicklung und Planung.

3.4 Extreme Programming (XP)

Extreme Programming setzt den Fokus auf die Implementierung und das Testen. Die Modellierung tritt dabei in den Hintergrund [BvK08]. Die Informationen in diesem Abschnitt wurden dem Buch des Erfinders von XP ([Bec00]) entnommen, soweit nicht anders angegeben.

Vorgehensweise Ein typisches XP-Projekt beginnt nicht gleich mit der ersten Planung, sondern mit einer Exploration. Diese Aktivität dient einer groben Gesamtschätzung des Projekts und einer Risikominimierung. Zu Beginn der Exploration wird von den Entwicklern und dem Kunden eine Risikoliste erstellt, die unter Anderem folgende Aspekte ([WRL05]) berücksichtigt:

- Was soll fachlich umgesetzt werden? Was gehört zum Projekt und wo liegen die Grenzen?
- Welche Technologie wird genutzt und welche Risiken birgt die Technologie? Werden neue Technologien eingesetzt, deren Risiken zunächst nicht abgeschätzt werden können?

Die Risikoliste wird zu Beginn nicht vollständig sein, sondern inkrementell erweitert. Wenn alle anfänglichen Risiken gesammelt wurden, werden die Risiken priorisiert und anschließend bewertet. Dies geschieht, indem die Projektbeteiligten das Risiko untersuchen und herausfinden, ob es wirklich ein Risiko darstellt oder ob es gelöst/umgangen werden kann. Im Laufe der Explorationsaktivität werden neue Risiken identifiziert, aber auch alte von der Liste entfernt, falls sie kein Risiko mehr darstellen ([Bec00]).

Die nächste Aktivität ist die Planung, die nur sehr wenig Zeit in Anspruch nimmt, wenn die Exploration sauber durchgeführt wurde. In dieser Aktivität einigen sich der Kunde und die Entwickler auf ein Datum für das erste Release, an dem ein System mit dem kleinsten Funktionsumfang fertiggestellt werden kann. Eine Technik für diese Planung stellt XP zur Verfügung: das Planungsspiel ([Bec00]). Dabei schreibt der Kunde die Anforderungen auf sogenannte StoryCards. Diese Anforderungen werden in XP User Stories genannt. Diese UserStorys auf den StoryCards kann man sich wie Karteikarten vorstellen auf denen eine vom Kunden formulierte Anforderungen steht. Anschließend schätzen die Entwickler die Zeit, die sie für die Umsetzung jeder Anforderung benötigen und die Kunden priorisieren diese abschließend. Auf diese Weise kristallisieren sich die wichtigsten Anforderungen für das erste Realease heraus. Das erste Release sollte zwischen 2 und 6 Monaten dauern.

Die nächste Aktivität teilt das erste Release in mehrere Iterationen, die zwischen einer und vier Wochen dauern. Dadurch entsteht ein Zeitplan für das erste Release. Da die erste Iteration die Architektur des Systems bestimmt, sollten die User Stories für die erste Iteration so ausgewählt werden, dass ein vertikaler Prototyp des Systems entsteht. Falls Abweichungen zum aufgestellten Zeitplan entstehen, muss darauf reagiert werden, indem beispielsweise User Stories entfernt oder deren Ausmaß geändert wird ([Bec00]).

In jeder Iteration werden für die jeweiligen User Stories Testfälle entwickelt und die Funktionalität implementiert. Anschließend wird das System mit allen Testfällen getestet und bei erfolgreichem Testen die Architektur entsprechend angepasst. Am Ende eines Releases sollten die Iterationen kürzer werden und die Entwickler sollten sich jeden Tag treffen, damit jeder weiß woran der andere gerade arbeitet ([Bec00]).

Nachdem das Realease fertig ist und ausgeliefert wurde, beginnt die Releaseplanung für das darauffolgende Release, das die gleichen Aktivitäten beinhaltet wie das erste. Zusätzlich zu den neuen Funktionalitäten müssen die alten Funktionalitäten gewartet werden, wodurch zusätzliche Arbeit auf die Entwickler zukommt was vom Projektleiter berücksichtigt und eingeplant werden muss ([Bec00]).

Rollen Beck gibt folgende Rollen in seinem Buch [Bec00] an. Diese Unterteilung ist detaillierter als die Unterteilungen bei RUP und Scrum, aber dennoch sind weitere Unterteilungen möglich.

Kunde Der Kunde liefert die Anforderungen an das Projekt und schreibt User Stories.

Programmierer Neben der Entwicklung und Implementierung von Klassen muss der Programmierer zusätzlich mit anderen Projektbeteiligten kommunizieren und Tests schreiben.

Tester Ein Tester ist ein Programmierer oder Kunde, der die Tests regelmäßig laufen lässt.

Projektleiter Der Projektleiter ist verantwortlich für den Entwicklungsprozess. Dabei sollte er nicht zu viele Regeln aufstellen und das Team möglichst frei entwickeln lassen.

XP-Trainer Der XP-Trainer arbeitet möglichst passiv und zeigt Probleme in der Entwicklung des Projektes auf.

Verfolger Ein Verfolger beobachtet und analysiert die Schätzungen im Projekt, d.h. er gibt feedback wie „Unsere Zeit- und Kostenschätzungen waren in der letzten Iteration 50% zu hoch.".

Technologieberater Der Technologieberater unterstützt das Team bei speziellen Technologien, mit denen das Team noch nicht gearbeitet hat.

3.5 Vergleich von RUP, XP und Scrum

Alle drei Vorgehensmodelle sind sich von der groben Vorgehensweise sehr ähnlich, doch wie bereits in Abbildung 1 zu erkennen ist, gibt es Unterschiede im Detailgrad und der Flexibilität. Der RUP gibt genaue Vorgaben zum kompletten Prozess und den jeweiligen zu erstellenden Dokumenten, wobei alles sauber und mit vielen Details dokumentiert wird. Bei der Nutzung von XP gibt es zwar Startvorschläge, aber insgesamt ist das Projektteam freier in der Umsetzung. Die Modellierung und die Dokumentation sind nicht Fokus des

Vorgehensmodells. Scrum gibt einen sehr groben Projektrahmen vor, so dass das Projektteam frei in der Umsetzung ist.

Für ein Projekt das sinnvollste Vorgehensmodell zu identifizieren ist - vor allem vor dem Hintergrund, dass der Autor dieser Arbeit wenig Erfahrung in der Projektleitung besitzt - ein schwieriges Unterfangen. Zwar lässt sich eine grobe Auswahl über die Familien von Vorgehensmodellen finden, aber eine feine Auswahl zu treffen ist schwierig.

Ausgehend von den formalen Kriterien, d.h. ohne auf Erfahrung zurückgreifen zu können, ist Scrum das geeigneteste Vorgehensmodell. Durch seinen iterativen Charakter wird sichergestellt, dass an dem Auslieferungstermin ein ausführbares System läuft (u.U. mit geringem funktionalem Umfang). Dies kann auch (nahezu) bei unklaren, flexiblen Anforderungen sichergestellt werden. Außerdem ist es sinnvoll für Teams mit maximal 10 Personen ([RJ00]), die wenig Erfahrung mit den eingesetzten Technologien besitzen. Durch den inkrementellen iterativen Charakter ist es dem Projektteam möglich auf Erfahrungen aus vorangegangenen Iterationen zurückzugreifen.

Auch XP und RUP könnten sinnvoll in der Projektgruppe eingesetzt werden, allerdings sind die vorgegebenen Strukturen strenger und ermöglichen damit ein nicht so flexibles Vorgehen.

4 Zusammenfassung

Die Projektgruppe „StreamCars" sollte auf jeden Fall auf einen Vertreter der Familie der wiederholenden Vorgehensmodelle zurückgreifen. Auf diese Weise entsteht mit hoher Wahrscheinlichkeit ein ausführbares System zum Auslieferungstermin (siehe Abschnitt 3). Auf die flexiblen Anforderungen kann eine Projektleitung am Besten reagieren, wenn agile Vorgehensmodelle eingesetzt werden (siehe Abschnitt 3.1). Der Vergleich der drei in dieser Arbeit vorgestellten agilen Vorgehensmodellen Rational Unified Process, Extreme Programming und Scrum ergibt, dass Scrum das sinnvollste Vorgehensmodell ist. Scrum unterstützt vor allem unerfahrene Entwicklungsteams durch einen flexiblen Projektrahmen. Dennoch ist dieser Projektrahmen genügend strukturiert um der ebenfalls unerfahrenen Projektleitung die benötigte Unterstützung zu geben. Des Weiteren ist die personelle Projektgröße mit 10 Personen optimal für Scrum ausgelegt.

Literatur

[Bec00] K. Beck. *extreme Programming explained: embrace change*. Addison-Wesley, 2000.

[Boe88] B. W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21:61–72, 1988.

[BvK08] C. Bunse und A. v. Knethen. *Vorgehensmodelle kompakt*, Jgg. 2. Spektrum Akademischer Verlag, 2008.

[Coc05] A. Cockburn. *Crystal Clear*. Addison-Wesley, 2005.

[Har02] M. Harsu. FAST Product-line Architecture Process. Bericht, Institute of Software Systems, Tampere University of Technology, 2002.

[KCH+90] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak und A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Bericht, Carnegie-Mellon University Software Engineering Institute, 1990.

[Kru00] P. Kruchten. *The Rational Unified Process An Introduction*, Jgg. 2. Addison-Wesley, 2000.

[RJ00] L. Rising und N. S. Janoff. The Scrum Software Development Process for Small Teams. *IEEE Software*, 17:26–32, 2000.

[Roy70] W. Royce. Managing the development of large software systems: concepts and techniques. In *Proc. IEEE WESTCON*, 1970.

[Sch04] K. Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[WRL05] H. Wolf, S. Roock und M. Lippert. *eXtreme Programming: Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis*, Jgg. 2. dpunkt.verlag, 2005.