Seminar
Game Development
Development Methodology

Marcel Nienhold
1005766
11. Juni 2007

Aufgabensteller:
Prof. Dr. Uwe M. Borghoff
Betreuer:
Dipl.-Inform. Daniel Volk

Institut für Softwaretechnologie
Fakultät für Informatik
Universität der Bundeswehr München

Inhaltsverzeichnis

1 Einführung **3**

1.1 Einordnung . . . 3

1.2 Überblick . . . 3

2 Methodischer Wandel in der Spieleentwicklung **4**

3 Besonderheiten der Spieleentwicklung **7**

3.1 Methodische Besonderheiten (dynamische Struktur) . . . 7

3.1.1 Iteratives Vorgehen und Prototyping . . . 7

3.1.2 Interdisziplinäres Entwickeln . . . 8

3.1.3 Weitere Unterschiede . . . 8

3.1.4 Exkurs: Multiplayer- und Online-Spiele . . . 8

3.2 Teilbereiche im Entwicklungsteam (statische Struktur) . . . 9

3.2.1 Publisher und Producer . . . 9

3.2.2 Game Design . . . 10

3.2.3 Programmierung . . . 10

3.2.4 Grafik- und Sound-Design . . . 10

4 Phasen und Unterstützungsprozesse des Entwicklungsprozesses **12**

4.1 Konzeption . . . 12

4.1.1 Prototyping . . . 12

4.1.2 Risiko-Analyse . . . 13

4.1.3 Pitching . . . 13

4.2 Produktion . . . 13

4.2.1 Technischer Anteil (Tools, Engines und Middleware) . . . 14

4.2.2 Inhaltlicher Anteil (Game Content und Game Design) . . . 14

4.2.3 Methodische Diskussion . . . 14

4.3 Testen . . . 15

4.3.1 Qualitätssicherung und Fehlerarten . . . 15

4.3.2 Fertigungszustände . . . 16

4.4 Nachbereitung (Post Mortem) . . . 16

5 Aktuelle Trends und Entwicklungen **18**

5.1 Plattformunabhängige Entwicklung . . . 18

5.2 Agile Entwicklungsmethoden . . . 18

5.3 Weitere Trends . . . 19

1 Einführung

1.1 Einordnung

Die vorliegende Arbeit entstand am Institut für Softwaretechnologie der Fakultät für Informatik an der Universität der Bundeswehr München im Rahmen des Hauptseminars *Game Development* im Frühjahrs-Trimester 2007.

Der Grundgedanke und „rote Faden" dieses Seminars war es einerseits, einen Überblick über den aktuellen Stand der Spieleindustrie zu liefern und andererseits, aktuelle Standards in der Entwicklung von Computerspielen zu beleuchten.

Eingebettet ist die Arbeit hierbei in den ersten großen Teilbereich des Seminars, den *Game Development Process*, also in den Bereich des Entwicklungsprozesses von Spielen an sich. Ausgangspunkt für meine Arbeit ist die vorangeganene Ausarbeitung zum Thema *Game Business*. Aufbauend auf den darin beleuchteten Eigenheiten der Spieleindustrie im Vergleich zu anderen Wirtschaftszweigen, werde ich mich den Besonderheiten und Unterschieden in der Methodik des Entwicklungsprozesses von Computerspielen widmen.

1.2 Überblick

Als erstes werde ich in meiner Arbeit auf den Wandel in der Entwicklungs-Methodik von Computerspielen im Laufe der Zeit eingehen. Kapitel 2 soll dabei aufzeigen, wie in den Anfängen der Spieleentwicklung gearbeitet wurde und vor allem, welche großen Veränderungen sich im Vergleich dazu bis heute aufgetan haben. War es vor 20 Jahren noch möglich, in relativ kurzer Zeit und mit einem sehr geringen Budget ein Spiel zu entwickeln, das sich gut verkaufte und den aktuellen Standards entsprach, geht die Spieleindustrie seit einiger Zeit mehr und mehr zu einem sehr geplanten Vorgehen über. Dies bezieht sich sowohl auf den Entwicklungsprozess an sich als auch auf die zunehmende Verteilung von Aufgaben auf verschiedene Personen.

Kapitel 3 widmet sich dann ausführlich den Besonderheiten bei der Spielentwicklung (vornehmlich im Vergleich zu herkömmlicher Software). Es werden einerseits methodische Eigenheiten beleuchtet (Abschnitt 3.1) und andererseits diskutiert, wem während des Entwicklungsprozesses welche Aufgaben zukommen und welche Personen in dieser immer mehr interdisziplinär werdenden Entwicklung überhaupt beteiligt sind (Abschnitt 3.2).

Der Entwicklungsprozess selbst, eine mögliche Einteilung desselben in Phasen sowie seine Methodiken und Unterstützungsprozesse werden dann der zentrale Inhalt von Kapitel 4 sein.

Die Kapitel 3 und 4 als Einheit betrachtet bilden den Kern der gesamten Arbeit. Das große Thema der Entwicklungs-Methodik von Computerspielen soll in ihnen eingehend erläutert werden.

Das abschließende 5. Kapitel wagt dann einen kurzen Ausflug in aktuelle Trends der Spieleentwicklung.

2 Methodischer Wandel in der Spieleentwicklung

Um verstehen zu können, wie und warum der heutige Zustand der Spieleindustrie zustande gekommen ist, ist es wichtig zu wissen, wie früher Spiele entwickelt wurden und welche Faktoren eine *Veränderung in der Entwicklungs-Methodik* bewirkten und noch bewirken. Dieses Kapitel versucht aufzuzeigen, wie der Ursprung der Spielentwicklung aussah und wie heute Computerspiele entwickelt werden.

Bis Anfang der 60er Jahre des letzten Jahrhunderts hatte überhaupt niemand daran gedacht, die bis dahin noch sehr wenig verbreiteten Computer zum Spielen zu nutzen. 1962 änderte sich das Ganze als *Steve Russell* am MIT eines der ersten Video- und Mehrspieler-Computerspiele programmierte. Inspiriert vom Bau der PDP-1[1] im Jahre 1961 entwickelte Russell *Spacewar*. Obwohl die damalige Grafik für heutige Standards noch sehr rudimentär war, fand das Spiel eine große Fangemeinde und ist bis heute Vorbild zahlreicher Nachahmungen. Trotz des bereits angesprochenen Fehlens eines wirklichen Markts für Computerspiele, verbreitete sich Spacewar ziemlich schnell. Dies war damals auch nicht weiter verwunderlich, da der Quellcode für jeden zugänglich war, der ihn haben wollte. Spacewar erwies sich in den folgenden Jahren als *der* Auslöser für die Entwicklung zahlreicher weiterer Spiele. So entstanden beispielsweise *Lunar Lander* (die erste Flugsimulation) und *Adventure* (das erste Rollenspiel).

Diese frühen Produkte wurden meistens noch von *einzelnen Personen* im universitären Umfeld (Hacker[2]) in einer *Code-And-Fix*-Vorgehensweise produziert. Darunter ist ein im Prinzip völlig unstrukturiertes Vorgehen zu verstehen. Der Entwickler beginnt direkt mit der Implementierung des Spiels und verbessert diese solange, bis das Ergebnis seinen Vorstellungen entspricht. Bei dieser Methodik wird kaum Dokumentation erstellt und auch klassische Phasen wie Analyse und Entwurf fallen weg. Diese Art des Vorgehens ist heutzutage nicht nur in der Spieleindustrie veraltet, auch in der klassischen Software-Entwicklung ist sie abzulehnen, da eine Wartung oder Weiterentwicklung der Software meist nur durch den ursprünglichen Entwickler möglich ist.

Als erstes kommerziell verwertetes und für die Allgemeinheit wirklich zugängliches Computerspiel gilt heute das seit 1972 verbreitete *Pong*. Obgleich bereits zuvor Videospiele entwickelt wurden, gilt Pong heute allgemein als „Urvater“ aller Videospiele. Dies ist vor Allem dadurch zu erklären, dass frühere Produkte einerseits nicht die breite Öffentlichkeit erreichten wie dann Pong, sondern vornehmlich im Bereich der Universitäten genutzt wurden. Außerdem darf man aus heutiger Sicht durchaus anzweifeln, ob die den älteren Spielen zugrunde liegende Hardware es zulässt, sie im heutigen Sinne als Computerspiel zu bezeichnen[3]. Pong wurde fortan in Form von Spielautomaten verbreitet und trug somit wesentlich dazu bei, dass Computerspiele langsam einer breiteren Öffentlichkeit zugänglich wurden und ein Kommerzialisierungsprozess begann. Denn diese Automaten mussten seinerzeit natürlich für jedes weitere Spiel entsprechend mit Münzen „gefüttert“ werden.

Während bei den eingangs erwähnten Code-And-Fix-Projekten traditionell einzelne Personen oder maximal kleinere Teams als Entwickler auftraten, ging der *Trend* nun *zu größeren Teams*, die dann auch entsprechend strukturierter vorgingen. Man besinnte sich auf entspre-

[1] Der von DEC entwickelte erste Minicomputer der Welt. Der PDP-1 lies sich im Gegensatz zu den viel größeren IBM-Maschinen von einer Person hochfahren und steuern.

[2] Hacker war zu der Zeit keineswegs ein negativ belegter Begriff, sondern einfach nur die Selbstbezeichnung derjenigen, die sich aus eigenem Antrieb und Spaß daran mit Computern beschäftigten.

[3] Tennis for Two beispielsweise (der Vorgänger von Pong) lief auf einem Analogcomputer und einem kleinen Oszillographen als „Bildschirm“-Ausgabe.

chende Prinzipien des Software Engineering. So fand hier das klassische *Wasserfallmodell* Anwendung.

Ein entscheidender Anstoß zur Kommerzialisierung der Spieleindustrie war 1977 das Erscheinen der ersten Spielkonsole für den „Heimmarkt“, der *VCS* von Atari. Mit dieser Konsole bestand erstmals die Möglichkeit, im privaten Bereich und ohne zusätzliche Kosten für einzelne Spielrunden Computerspiele zu nutzen. Die entsprechenden Spiele wurden auf Steckmodulen ausgeliefert.

In der Folgezeit wurde ein regelrechter Boom für Heimcomputer-Spiele ausgelöst.

Im Laufe der 80er Jahre wurde dann auch die Hardware deutlich schneller, was einige neue Entwicklungen erst möglich machte. So wurden beispielsweise im grafischen Bereich die ersten 3D-Effekte und sogar Videosequenzen verwendet (erst gegen Ende dieses Jahrzehnts). Außerdem kam 1983 dann endlich auch das erste Online-Spiel auf den offenen Markt.

Zu Beginn der 90er Jahre war dann ein klarer Trend zur Entwicklung neuer Spielegenres zu erkennen. Als ein Beispiel dafür seien hier nur die Ego-Shooter erwähnt, die mit *Doom* erstmals auftraten und fortan einen Großteil des gesamten Spielemarkts ausmachten. Im Vergleich zu Pong war bei Doom schon viel mehr Wert auf grafisches Design gelegt worden. Doom war es auch, das einen Wandel im Vertrieb von Spielen bewirkte. Erstmals wurden Demoversionen von Spielen als Shareware herausgebracht und kostenlos oder als Zugabe zu Computer-Zeitschriften verbreitet. Dies ermöglichte den Nutzern, kleine Teile eines Spiels zu testen, ohne das Spiel komplett kaufen zu müssen.

Eine weitere wichtige Entwicklung im Verlauf der 90er war, dass erstmals mobile Spiele auf den Markt kamen. Der *Gameboy* (von *Nintendo* entwickelt) war das vorherrschende Produkt in diesem Segment.

Auch begannen Spieler erstmals, selbst Veränderungen an fertigen Spielen vorzunehmen. Diese *Mods* (Kurzform für Modifications) wurden von den Spieleherstellern gern gesehen. Sie lieferten zu ihren Spielen oft sogar entsprechende Werkzeuge zur Veränderung.

Heute werden Spiele von kompletten Spielestudios entworfen und die *Teams* werden von der Anzahl ihrer Mitarbeiter *immer riesiger*. Die Spieleindustrie ist mittlerweile zu einem eigenen großen Wirtschaftszweig aufgestiegen. Ein viel breiteres Publikum (auch bewirkt durch die schnelle Entwicklung auf dem Hardware-Markt) und damit verbundene größere Absatzmärkte machen das überhaupt möglich. Ein modernes Spieleprojekt zerfällt in mehrere strukturierte Phasen mit klar definierten Zielen, Ergebnissen und Meilensteinen[4] und die Produktion des Spiels an sich wird nunmehr auf viele Schultern verteilt. Stand vor vielen Jahren noch der Programmierer selbst im Zentrum der Spielentwicklung, besteht ein Spielestudio heute aus einem ganzen Team verschiedenster Mitarbeiter.[5]

Das zunehmende Anwachsen der Teamgröße, die damit einhergehende komplexere Kommunikation zwischen den Teammitgliedern und die entsprechenden eigenen Anforderungen im wirtschaftlichen Bereich[6] machten also einen Wandel in der konkreten Vorgehensweise beim Entwickeln von Computerspielen nötig. Heute wird nicht mehr das Wasserfallmodell verwendet. Vielmehr sind Stichworte wie das Spiralmodell[7] oder agile Entwicklung[8] kennzeichnend für modernes Game Development.

[4] Für Details hierzu siehe Kapitel 4.
[5] Die einzelnen Rollen, die dabei auftreten sind in Abschnitt 3.2 näher erläutert.
[6] Siehe die entsprechende Seminararbeit zum Thema *Game Business*.
[7] Siehe Unterabschnitt 4.2.3.
[8] Siehe Abschnitt 5.2.

Das erste große Kapitel ist hiermit abgeschlossen. Es wurde aufgezeigt, dass die Spielentwicklung im Laufe ihrer Geschichte einem starken Wandel unterworfen war. Die Ergebnisse dieses Wandels - vor allem bezüglich der Methodik der Spieleentwicklung - werden in den folgenden Kapiteln näher betrachtet. Für weitere Details bezüglich aktueller Trends wird auf Kapitel 5 verwiesen.

3 Besonderheiten der Spieleentwicklung

Dieses Kapitel soll detailliert auf Besonderheiten bei der Spieleentwicklung eingehen. Als wichtigster Abschnitt ist hierbei 3.1 anzusehen. Dort werden zentrale methodische Eigenheiten der Spielentwicklung beleuchtet, bevor dann in Abschnitt 3.2 die interdisziplinäre Entwicklung von Spielen durch die Darstellung der entsprechenden Teilbereiche eines Game Studios erörtert wird.

3.1 Methodische Besonderheiten (dynamische Struktur)

Die Besonderheiten der Spieleindustrie machen in vielen Bereichen der Spielentwicklung ein besonderes Vorgehen nötig. In diesem Abschnitt soll darauf eingegangen werden, welche methodischen Vorgehensweisen dabei Verwendung finden und so in „normalem" Software Engineering weniger bzw. gar nicht zu finden sind. Exemplarisch dafür sollen zwei Punkte genauer beleuchtet werden:

3.1.1 Iteratives Vorgehen und Prototyping

Als erstes Beispiel für klassische Spielentwicklungs-Methodik soll das iterative Vorgehen beleuchtet werden. *Iterativ* ist wohl *das* Schlagwort beim Entwickeln von Computerspielen. Der in Kapitel 4 beschriebene Entwicklungsprozess findet immer wieder in iterativen Schleifen statt.

Im Gegensatz zu klassischen Software-Entwicklungsmodellen[9], treten im Spielentwicklungsprozess viel mehr Rückkopplungen auf. Es ist viel weniger ein lineares als ein zyklisch-iteratives Vorgehen zu erkennen. In [Ful04] werden diese Zyklen als ein ständiges Wiederkehren von Testen, Auswerten, Überarbeiten, Testen, Auswerten, Überarbeiten, ... beschrieben. Diese Iterationen beginnen natürlich nicht sofort zu Beginn des Entwicklungsprozesses, sondern setzen dann ein, wenn erste Ideen formuliert und formalisiert wurden. Dies geschieht üblicherweise schon in der Konzeptionsphase. Bis zum Ende der Qualitätssicherung erfolgt die Entwicklung dann fortwährend in den beschriebenen Schleifen.

Ein Grund für dieses sehr früh einsetzende iterative Vorgehen ist auch das hohe Maß an Prototyping bei der Spieleentwicklung. Potentielle Käufer müssen einerseits schon sehr früh mit einer Demoversion „versorgt" werden. Andererseits hat auch der Publisher ein Interesse daran, in regelmäßigen Abständen verbesserte Prototypen vorgelegt zu bekommen um das Fortschreiten der Entwicklung beurteilen zu können. Die iterative Struktur des Entwicklungsprozesses macht dies möglich.

Zusätzlich zu den bisher beschriebenen „kleinen" Iterations-Schleifen findet man im Game Development auch noch „größere" Iterationen, denn ein weiterer Aspekt des iterativen Entwicklens tritt in der Spieleindustrie bei Spielen auf, die regelmäßig fortgesetzt werden. Selbstverständlich entwickeln große Spielestudios nicht jede Version von Grund auf neu, sondern man baut in einem ähnlichen iterativen Zyklus auf älteren Versionen auf. Die Evaluation kann hierbei auch gut über ein entsprechendes Feedback der Spielergemeinde erfolgen. Ein Beispiel hierfür ist das alljährlich neu erscheinende *FIFA* von *Electronic Arts*.[10]

[9] Wie z.B. dem Wasserfall-Modell. Siehe dazu [Min07] oder [Bal00].

[10] Hier hat jahrelange Kritik der Spieler letztendlich dazu geführt einen Managermodus einzubinden und über das sogenannte *FIFA Fusion* eine Kooperation mit dem entsprechenden Manager-Produkt (*FIFA Manager*) zu ermöglichen.

3.1.2 Interdisziplinäres Entwickeln

In Abschnitt 3.2 werden einige klassische Rollen erläutert, die bei der Spieleentwicklung auftreten. Dabei wird auffallen, dass neben typischen „Software-Engineering-Rollen“, wie Programmierer und Producer auch andere Rollen auftreten, die bei herkömmlicher Software-Entwicklung nicht zu finden sind. Die Rede ist hier weniger vom Publisher als viel mehr von Teammitgliedern wie Grafik- und Sounddesignern oder Storyschreibern. Der spieletypische Aspekt der *content creation*[11] tritt bei anderer Software so nicht auf. Auch ist es dort nicht nötig, viel Arbeit in Grafikprogrammierung oder die Verwendung von Musik und Videos zu investieren. Dieser Bereich der sogenannten *Asset Creation*[12] ist eher spieletypisch.

Die grundverschiedenen Anforderungen der Spieleentwicklung machen also ein interdisziplinär aufgestelltes Team nötig. Auf der einen Seite steht dabei die Technik (der Programmierer) auf der anderen Seite der Inhalt (der Game-Designer, ...). Diese Zusammensetzung führt naturgemäß zu Kommunikationsproblemen, da Software-Techniker eine andere „Sprache“ sprechen als Designer. Ein gutes Spielestudio braucht daher einen kompetenten Producer, der diese Probleme lösen kann und zum Verständnis der einzelnen Teilteams untereinander beiträgt.

3.1.3 Weitere Unterschiede

Über die Unterschiede von Spieleentwicklung und traditionellem Software-Engineering könnte man noch viel mehr sagen. Die eben genauer beleuchteten Felder zeigen wohl schon ganz gut, dass man beides nicht gleichsetzen kann.

Weitere Besonderheiten der Spieleentwicklung sind beispielsweise: Eine flachere hierarchische Struktur, die Arbeit auf der Basis eines Vertrages (ähnlich der Filmindustrie) oder das im Durchschnitt wesentlich jüngere Alter der Teammitglieder. Weiterhin ist es im Bereich der Spieleentwicklung viel wichtiger, den aktuellen Stand der Technologie zu nutzen (beispielsweise aktuelle Grafik-Standards). Ein Arbeiten mit leichtgewichtigen Entwicklungsprozessen ist ein weiteres Charakteristikum der Spielindustrie. Schließlich treten auch im Bereich des Marketing deutliche Unterschiede auf. So ist die Spieleindustrie absatzorientierter ausgerichtet als jede andere Software-Sparte.

Man könnte hier noch viel mehr Unterschiede nennen und tiefer ins Detail gehen. Als grober Überblick soll dies jedoch genügen.[13]

Damit ist dieser Abschnitt abgeschlossen, in dem die wesentlichen Unterschiede von Computerspielen zu anderer Software beleuchtet wurden.

3.1.4 Exkurs: Multiplayer- und Online-Spiele

Nachdem nun eingehend auf die Unterschiede zu herkömmlicher Software-Entwicklung verwiesen wurde, soll jetzt exemplarisch auf die Besonderheiten eines speziellen Spieltyps eingegangen werden, die Multiplayer- und Online-Spiele.

[11] Darunter versteht man das Erfinden und Entwickeln von Inhalten wie der Spielhandlung oder Charakteren. Siehe dazu auch Unterabschnitt 4.2.2.

[12] Bezeichnet im Fachjargon die Speicherung und Verwaltung digitaler Inhalte (Grafiken, Videos, Musikdateien, ...).

[13] Für eine genauere Diskussion siehe z.B. [Nac05] Kapitel 2.2.

In der Spieleentwicklung ist seit Anfang der 90er Jahre des letzten Jahrhunderts ein klarer Trend zu Multiplayer-Spielen erkennbar. Begonnen hat dieser Trend im Genre der Rollenspiele. Durch die gleichzeitige Teilnahme vieler Spieler an Online-Spielen ist bei solchen ein höheres Maß an „menschlicher“ Interaktion gegeben als bei Single-Player-Spielen.

Charakteristisch für Multiplayer-Spiele ist, dass diese nicht so schnelllebig sind wie klassische Single-Player-Spiele. Durch das ständige Nachliefern von Content durch Addons wird gewöhnlich viel länger Spielspaß geboten. Entsprechend aufwendiger ist auch die Modellierung und Wartung solcher Spiele.

Abschließend sei noch erwähnt, dass auch was die ökonomische Seite angeht, Online-Spiele eine Sonderrolle einnehmen. Einzelheiten dazu finden sich in der entsprechenden Arbeit zum Thema *Game Business* und sollen hier nicht weiter diskutiert werden.

3.2 Teilbereiche im Entwicklungsteam (statische Struktur)

Nachdem schon mehrfach angesprochen wurde, dass ein modernes Spielestudio aus vielen beteiligten Personen besteht, sollen nun die einzelnen, dabei auftretenden Rollen etwas genauer unter die Lupe genommen werden. Die Rollen des Publishers und des Producers werden dabei nur kurz erläutert, da eine eingehendere Betrachtung dieser in der Arbeit zum Thema *Game Business* zu finden ist. Detaillierter wird dann auf die *klassische Dreiteilung* eines Entwicklungsstudios in einen kreativen (Game-Design), einen technischen (Programmierung) und einen künstlerischen (Grafik- und Sound-Design) Part eingegangen. Schon allein an dieser Aufteilung ist zu erkennen, dass die Programmierung nur noch einen kleinen Teil der Spieleentwicklung ausmacht und eine moderne Spieleproduktion sehr stark interdisziplinär geworden ist.

3.2.1 Publisher und Producer

Publisher sind Unternehmen, die Computerspiele veröffentlichen. Ein Publisher ist durchaus mit einem Verlag zu vergleichen, der Bücher veröffentlicht. Nur dass er eben die Produktion von Datenträgern, das Marketing und den Vertrieb eines Computerspiels übernimmt. All diese Tätigkeiten werden in einem entsprechenden Vertragswerk mit den eigentlichen Entwicklern festgehalten. Anders ausgedrückt nimmt der Publisher also die *Rolle des Geldgebers* ein. Er finanziert ein Spiel vor und erhält im Gegenzug dazu die Rechte, das fertige Spiel zu veröffentlichen.

Die wesentliche Schnittstelle zwischen dem Publisher und dem eigentlichen Entwicklungsteam bildet der Producer, der oft vom Publisher selbst von außen eingesetzt wird. Er ist der Repräsentant des Teams nach außen sowohl gegenüber dem Publisher als auch der PR-Abteilung, dem Studio-Management und der Marketing-Abteilung. Seine Kernaufgabe besteht darin, sicherzustellen, dass das Spiel rechtzeitig und im Rahmen der finanziellen und inhaltlichen Vorgaben des Publishers fertiggestellt wird.

Kommen wir nun zu den eigentlichen Akteuren der Spieleentwicklung. Hier ist in vielen Spiele-Studios eine klassische Dreiteilung zu erkennen. Vorab sei noch angemerkt, dass üblicherweise keine dieser drei Rollen von nur einer Person ausgefüllt wird. Im weiteren Verlauf wird jeweils von „ihm“ die Rede sein, jedoch verbirgt sich dahinter oft ein ganzes Teil-Team.

3.2.2 Game Design

Betrachten wir zunächst den kreativen Teil, den Game Designer. Um zu der Analogie mit der Filmindustrie zurückzukehren, kann man sagen, dass der Game Designer am ehesten der Rolle des Regisseurs entspricht. Als kreativer Kopf des Teams ist er verantwortlich für das Layout, das Konzept und das Gameplay des Spiels, also für das eigentliche Game Design. Aus einer Idee ein schlüssiges und gutes Spielkonzept entwickeln zu können, macht einen Game Designer aus. Weiterhin sollte er die Fähigkeit besitzen, erzählen zu können, da die gesamte Story des Spiels in seiner Verantwortung liegt.[14] Mit der Entwicklung zentraler Ideen und Konzepte für ein Spiel steht das Game-Design-Team im Zentrum des gesamten Spiels.

Abschließend sei noch bemerkt, dass Game Designer häufig die Personen des Teams sind, denen in den Medien die größte Aufmerksamkeit und damit Bekanntheit zukommt.

3.2.3 Programmierung

Kommen wir nun zum technischen Teil eines Spiele-Studios, dem Programmierer. Die vom Designer gelieferten Spielideen müssen nun am Computer in die Realität umgesetzt werden. An dieser Stelle kommt also die klassische Implementierung ins Spiel, die natürlich zentrale Aufgabe des Programmierers ist. Hierbei kann man wiederum eine weitere Unterteilung der Rolle des Programmierers vornehmen. So besteht ein klassisches Programmierteam beispielsweise aus Grafik-, Netzwerk- und KI-Programmierern, denen jeweils sehr spezifische Aufgaben zukommen.

3.2.4 Grafik- und Sound-Design

Schlußendlich betrachten wir nun den künstlerischen Part. Ein modernes Computerspiel ohne ausgereifte Grafik- und Soundunterstützung ist nicht mehr denkbar. Daher rührt die Notwendigkeit, auch in diesem Bereich entsprechende Experten einzusetzen.

So ist der Sound-Designer für die Gestaltung der Klangkulisse sowie aller Geräusche von Charakteren, Waffen, Fahrzeugen etc. verantwortlich. Dazu werden einerseits reale Vorbilder kopiert, aber auch eigene Geräusche designt.

Der Grafik-Designer wiederum muss ein hohes Maß an räumlichem Vorstellungsvermögen mitbringen, da seine Aufgabe heute vor Allem im Design von 3D-Modellen besteht. Dazu sind natürlich auch gute Zeichenkünste von Nöten, die mit entsprechenden Werkzeugen am Computer umgesetzt werden. Die Darstellung und Widergabe natürlicher Strukturen und Bewegungen gehört dabei wohl zum schwierigsten Teil seiner Arbeit.[15]

Um den Kreis zur Filmindustrie zu schließen sei noch erwähnt, dass teilweise professionelle Schauspieler (aber auch Musiker) von Spiele-Studios engagiert werden um entsprechende Beiträge zu leisten.

Damit ist auch dieser letzte Abschnitt des Kapitels abgeschlossen und es wurde ein grober Einblick in die Struktur eines modernen Entwickler-Teams geliefert und gezeigt, dass zu moderner Spieleentwicklung viel mehr gehört als nur Programmieren. Die vorgenommene

[14] In den letzten Jahren ist auch zunehmend zu beobachten, dass professionelle Schriftsteller als Autoren fungieren um komplexe Hintergrundgeschichten und Charakterdarstellungen zu entwickeln.

[15] Dieses sogenannte *Motion Capturing* wird häufig nicht vom eigentlichen Studio erledigt, sondern an externe Firmen ausgelagert. Wohl auch weil dafür teilweise sehr spezielle Software notwendig ist.

Unterteilung hat keineswegs alle an einem Spiel-Projekt beteiligten Personen erfasst, sondern nur die wichtigsten Bereiche dargestellt. Außerdem muss gesagt werden, dass die Trennung zwischen den einzelnen Teilteams in der Realität häufig weniger strikt aussieht als hier beschrieben. So ist es beispielsweise gängige Praxis, dass das Kreativ-Team (Designer) auch technisches Personal (Programmierer) enthält, damit die Designer über technische Grenzen informiert sind und diese bei der Gestaltung respektieren können.

4 Phasen und Unterstützungsprozesse des Entwicklungsprozesses

Nachdem nun beleuchtet wurde, welche Aufgaben in einem Entwicklungsteam zu besetzen sind, widmen wir uns dem Prozess des Entwickelns selbst. Es sei vorab darauf hingewiesen, dass die folgende Einteilung desselben keineswegs Anspruch auf Allgemeingültigkeit erhebt. Es existieren viele andere Einteilungen[16], die der Folgenden aber sehr ähneln. Im Folgenden wird eine zeitliche Unterscheidung des Entwicklungsprozesses in eine Konzeptions- und eine Produktionsphase vorgenommen. Diese beiden Phasen werden in den Abschnitten 4.1 und 4.2 im Detail erklärt. In den darauf folgenden Abschnitten 4.3 und 4.4 werden dann noch zwei eher querschnittlich während der gesamten Produktion stattfindende Prozesse betrachtet. Einerseits das Testen, was unerlässlich für jede Software-Produktion ist und die Nachbereitung.

4.1 Konzeption

Am Anfang eines jeden Projekts steht die Konzeptionsphase. Hier müssen zunächst einige Rahmenbedingungen für die weitere Entwicklung klar definiert werden. Es ist wichtig bereits jetzt zu wissen, wie das Spiel später aussehen soll (um sehr früh einen ersten Prototyp liefern zu können), wie lange die Entwicklung dauern wird (damit frühzeitig ein Release-Datum angekündigt werden kann), wieviele Personen dazu benötigt werden und vor allem, was das Ganze in etwa kosten wird (damit der Publisher von Anfang an über die Kosten im Bilde ist). Als grobe Abschätzung kann man sagen, dass diese Phase etwa 10 bis 25 Prozent der gesamten Produktionszeit in Anspruch nimmt.

Das vorherrschende Ziel dieser ersten Phase ist es, das sogenannte *Game Design Document* zu erstellen. Dieses Dokument bildet den Rahmen für die weitere Arbeit am Projekt. Es enthält die komplette Story für das Spiel, detaillierte Beschreibungen der entsprechenden Gameplay-Komponenten sowie konkrete inhaltliche Beschreibungen des Spiels wie beispielsweise vorkommende Charaktere oder Level.

4.1.1 Prototyping

Ein wesentliches Charakteristikum der Spieleentwicklung ist ein hohes Maß an Prototyping. Im Klartext heißt das, dass bereits in dieser frühen Phase ein erster Prototyp des Spiels vorgelegt werden sollte. Diese Notwendigkeit ist im Wesentlichen durch zwei Fakten zu erklären:

Einerseits ist der Publisher natürlich daran interessiert, möglichst schnell zu sehen, was aus seinem Geld wird und wird es nicht tolerieren, lange auf erste Ergebnisse zu warten.

Andererseits ist es natürlich notwendig der Öffentlichkeit so früh wie möglich erste Spielausschnitte zeigen zu können oder im Idealfall schon eine Demo-Version mit ausgewählten Features zur Verfügung zu stellen um die Kaufbereitschaft zu fördern. Bedingt durch die immense Schnelllebigkeit des Spielemarktes will die potentielle Kundschaft eben mit Nachschub „gefüttert“ werden. Wie bei jeder Software spielt hier natürlich auch der Aspekt der Früherkennung von Schwächen und Fehlern im Konzept eine Rolle.

[16] So sieht [Nac05] beispielsweise eine Einteilung in Concept, Pre-Production, Prototyping und Full Production vor.

Wenn man von Prototyping spricht sollte man jedoch nicht nur an spielbare Demos denken. Auch in der modernen Spieleentwicklung wird teilweise noch viel Wert auf Papier-Prototypen gelegt, auf deren Basis dann digitale Prototypen entstehen.[17]

4.1.2 Risiko-Analyse

Ein weiterer Arbeitsschritt, der bereits in dieser frühen Phase stattfinden muss (und in späteren Phasen immer wieder ergänzt wird), ist die Risiko-Analyse. Die Entwickler müssen sich sehr früh darüber im Klaren sein, welche (ungeplanten) Ereignisse das Projekt gefährden können. Hierbei differenziert man einerseits zwischen der Wahrscheinlichkeit des Auftretens solcher Ereignisse und andererseits bezüglich der Höhe des Einflusses, die diese auf den Fortlauf des Projekts haben würden. Auf dieser Grundlage wird jedes Risiko klassifiziert und entsprechend dieser Klassifikation ein Risiko-Management-Plan erstellt, der die kritischstes Faktoren behandelt.

4.1.3 Pitching

Eine typische Vorgehensweise in der Entwicklung von Spielen ist auch das sogenannte *Pitching*[18]. Pitching im Sinne des Game Development besteht im Wesentlichen aus einem oder mehreren Zusammentreffen des Publishers mit den Entwicklern in einer sehr frühen Phase der Spieleentwicklung. Der konkrete Zeitpunkt kann dabei je nach Spiel leicht variieren.

Zweck dieses Treffens ist es, dem Publisher ein erstes Konzept für das Spiel vorzustellen. Die Entwickler müssen dabei in der Lage sein, ein klares Bild des Spiels zu zeichnen, damit der Publisher sich genau vorstellen kann, auf welchen Personenkreis das Spiel ausgerichtet sein wird und ob das Konzept erfolgversprechend ist. Schließlich wird der Publisher vor allem aufgrund des Pitching-Treffens entscheiden, ob er das Projekt unterstützt oder nicht. Im positiven Fall wird dann ein Vertrag zwischen Publisher und Entwicklern abgeschlossen.

Da ein gefragter Publisher jährlich hunderte Ideen durch solche Treffen präsentiert bekommt, ist es sehr wichtig, dass die Entwickler genau wissen, welche Informationen und Materialien für ein Pitching notwendig sind. Dazu zählen beispielsweise eine *kurze* und kompakte Beschreibung der Story und ein spielbarer Prototyp. Dieser kann sehr klein sein, sollte aber im Bezug auf Grafik und Gameplay bereits dem Endprodukt entsprechen. Im Gegensatz zu „herkömmlicher“ Software-Entwicklung haben diese Treffen bei der Spieleentwicklung immer mehr an Bedeutung gewonnen. Dies liegt vor allem an der zunehmenden Story-Lastigkeit moderner Computerspiele.

4.2 Produktion

Nachdem in der Konzeptionsphase nun die Grundlagen gelegt wurden, kann in der Produktionsphase mit der eigentlichen Erstellung des Spiels (also der Implementierung) begonnen werden. Hierbei ist anzumerken, dass bei den meisten Produktionen eine strikte Trennung der beiden Phasen gar nicht möglich ist. Vielmehr findet hier ein fließender Übergang statt. Wie in der Spieleentwicklung allgemein wird auch hier immer wieder iterativ gearbeitet und auch an Ergebnissen der Konzeptionsphase noch gefeilt. So tritt es z.B. sehr häufig auf, dass erst während der Produktion die Notwendigkeit zusätzlicher Features erkannt wird, die dann

[17] Eine detailliertere Beschreibung verschiedener Prototypen-Arten liefert [Nac05] Kapitel 2.3.
[18] Ursprünglich vom engl. „to pitch“, was etwa soviel wie (sein Lager) aufschlagen bedeutet.

natürlich auch in der Konzeption noch berücksichtigt werden müssen. Mindestens genauso oft kommt es übrigens auch vor, dass geplante Features noch gestrichen werden.

4.2.1 Technischer Anteil (Tools, Engines und Middleware)

Ein wesentlicher Anteil während der Produktion ist die technische Umsetzung der Spielanforderungen. Hierfür existiert eine Vielzahl entsprechender *Game Development Tools*. Moderne Spiele werden häufig mit Hilfe ganzer *Game Development Suites* entwickelt. Hierbei handelt es sich um komplette Pakete, die sämtliche zur Spielproduktion notwendigen Tools enthalten. Mit Hilfe solcher Werkzeuge ist es möglich, die Spielproduktion vom abstrakten Design-Entwurf auf eine weniger abstrakte Ebene der Umsetzung herunterzubrechen.

Bei der technischen Umsetzung von Spielen kommt man heutzutage nicht mehr an *Engines* und *Middleware*[19] vorbei[20]. Eine Game Engine bildet das Grundgerüst der meisten modernen Computerspiele. Sie besteht aus einer Programmbibliothek, die den Entwicklern häufig verwendete Werkzeuge zur Verfügung stellt. In der Praxis besteht eine Engine häufig aus mehreren „Teil-Engines“ (Grafik-Engine, Sound-Engine, Physik-Engine, ...). Es sei noch erwähnt, dass mittlerweile auch einige Open-Source-Engines verfügbar sind. Die *Quake-Engine* von id Software ist ein Beispiel dafür.

4.2.2 Inhaltlicher Anteil (Game Content und Game Design)

Selbstverständlich ist die inhaltliche Umsetzung der Ergebnisse der Konzeptionsphase ein Kernbestandteil der Produktion. Voraussetzung für die Entwicklung des *Game Contents* ist die abgeschlossene Entwicklung der erwähnten Tools. Daher findet das tatsächliche *Game Design* im Regelfall erst gegen Ende der Produktionsphase statt. Die Erstellung von Levels, Charakteren etc. sind typische Bereiche dieses Produktionsschrittes.

Im Bezug auf die Teilbereiche des Entwicklungsteams bleibt festzuhalten, dass während der Tool-Entwicklung naturgemäß die Programmierer stärker eingebunden sind als die Designer. Bei der content creation dreht sich dieses Verhältnis dann um.

4.2.3 Methodische Diskussion

Die Eigenheiten der Spieleentwicklung machen deutlich, dass bei der Produktion nicht auf klassische Methoden der Software-Entwicklung zurückgegriffen werden sollte. Auch die unterschiedlich starke Einbindung der Teilbereiche während der verschiedenen Produktionsphasen macht es nötig, andere Vorgehensweisen als beispielsweise das traditionelle Wasserfall-Modell zu verwenden, denn dieses ist auch aufgrund des iterativen Charakters des Game Development kaum nützlich.

Im Folgenden wird exemplarisch ein alternatives Vorgehensmodell genauer beleuchtet, das *Spiralmodell* nach *Boehm*: Dieses Modell fasst den Entwicklungsprozess eines Software-Produkts als iterativen Prozess auf, wobei jeder Zyklus in den einzelnen Quadranten folgende Aktivitäten enthält:

[19] Unter Middleware versteht man „anwendungsunabhängige Technologien, die Dienstleistungen zur Vermittlung zwischen Anwendungen anbieten, so dass die Komplexität der zugrundeliegenden Applikationen und Infrastruktur verborgen wird.“ (Nach [Ruh01])

[20] Für detailliertere Informationen siehe die entsprechende Seminararbeit.

1. Festlegung von Zielen
2. Auswerten von Alternativen
3. Realisierung und Überprüfung des Zwischenprodukts
4. Planung des nächsten Zyklus

Bei der Betrachtung dieser einzelnen „Teilphasen“ wird deutlich, dass das Spiralmodell im Prinzip eine Weiterentwicklung des Wasserfall-Modells ist, in der dessen Phasen eben mehrfach (spiralförmig) durchlaufen werden. Dieses Modell weist also einen inkrementellen und iterativen Charakter auf. Ein so entwickeltes Software-Projekt nähert sich den Zielen an, auch und gerade wenn sich diese während der Produktion verändern. Mit dieser Vorgehensweise steht den Spielentwicklern also eine deutlich flexiblere Methode zur Verfügung, die den iterativen Ansprüchen (siehe 3.1.1) der Spieleentwicklung gerecht wird.

Es gibt selbstverständlich noch weitere gut geeignete Methoden für die Spieleentwicklung. Beispiele dafür sind agile Entwicklungsmethoden und domainspezifisches Entwickeln (Siehe dazu Kapitel 5). Als abschließender Kommentar dieses Unterabschnitts sei noch auf die zunehmende *Verwendung leichtgewichtiger Prozesse* in der Spieleentwicklung hingewiesen. Die Methode des *Extreme Programming* ist ein Beispiel dafür.

Zusammenfassend lässt sich sagen, dass die Produktionsphase sehr auf den Inhalt fixiert ist, d.h. es wird im Wesentlichen programmiert und so Spielinhalte realisiert. Entsprechend der Unterschiede zu anderer Software treten in dieser Phase auch spezielle Produktionstechniken auf. Dazu zählen z.B. *Critical Stage Analysis* (CSA)[21], ständige Meetings oder wöchentliche Status-Reports.[22].

Nachdem nun die zeitlich nacheinander ablaufenden Phasen der Spieleentwicklung betrachtet wurden, wenden wir uns noch zwei grundsätzlich querschnittlich während des gesamten Projekts ablaufenden Prozessen zu:

4.3 Testen

Wie bei der Entwicklung jedes Software-Produkts ist auch in der Spieleentwicklung das Testen ein wichtiger und kritischer Prozess. Das gesamte Spiel muss auf Fehler überprüft werden. Durch unzählige Testläufe soll sichergestellt werden, dass alles korrekt läuft und keine (oder realistischer gesagt, wenige) Bugs enthalten sind, die das Spiel im schlimmsten Fall sogar zum Absturz bringen würden.

4.3.1 Qualitätssicherung und Fehlerarten

Bereits während der Produktionsphase beginnen die sogenannten QA-Tester[23] mit ihrer Arbeit. Sobald die ersten spielbaren Teile fertig sind beginnen sie mit der Fehlersuche und sind im Laufe des Entwicklungsprozesses oft die letzten, die noch Veränderungen am Spiel vornehmen. Ihre wesentliche Aufgabe ist es, sämtliche Funktionalitäten des Spiels gegenüber einem Testplan zu prüfen. Dadurch sind sie tatsächlich den Großteil ihrer Arbeitszeit damit beschäftigt, zu spielen.

[21] Darunter versteht man die Auswertung der Entwicklungsfortschritte des Spiels an kritischen Punkten (Meilensteinen) während der Entwicklung. Für Details siehe [Ham03].
[22] Für Details siehe [Cha06].
[23] QA = Quality Assurance (dt. Qualitätssicherung)

Bei der Unterscheidung von Fehlerarten hat jedes QA-Team eine eigene Klassifikation. Typische Fehler sind beispielsweise Textfehler (logische Fehler im Sourcecode), ästhetische Fehler (meist grafische Fehler ohne Einfluss auf das Gameplay) oder Programm Crashes (Fehler im Code, die einen Komplettabsturz des Spiels bewirken können). Diese Auflistung ist unvollständig. Entsprechend der Klassifikation eines QA-Teams können noch zahlreiche andere Fehlerarten hinzukommen.

4.3.2 Fertigungszustände

Einhergehend mit der Testphase ist auch das Durchlaufen mehrerer Fertigungszustände des Spiels. Als erstes kommt die sogenannte *Alpha-Version*. Zu diesem Zeitpunkt sind die Kernfunktionalitäten und -eigenschaften des Spiels implementiert und (soweit vorhanden) auch getestet.

Dieser Zustand geht dann in eine *Beta-Version* über, bei der der Code alle gewünschten Eigenschaften und Features des Spiels enthält. Zu diesem Zeitpunkt wird häufig auch ein *feature freeze* erklärt, also ein Zustand, bei dem keine neuen Funktionalitäten mehr implementiert werden dürfen und auch keine Änderungen mehr vorgenommen werden dürfen (es sei denn es handelt sich um gravierende Bugs). Die Beta-Version wird häufig einem eingeschränkten Personenkreis (Spieletestern) zugänglich gemacht, um weitere Fehler entdecken und beheben zu können.

Wenn dies abgeschlossen ist, erreicht das Spiel den Status *Gold-Master*, den Zustand, in dem es veröffentlicht wird. Aufgrund des von der Spieleindustrie ausgehenden Termindrucks kommt es jedoch nicht selten vor, dass die Veröffentlichung bereits zu einem früheren Zeitpunkt stattfinden muss. Die dann noch auftretenden Fehler müssen vom Spieler selbst behoben werden, indem der Hersteller Patches zur Verfügung stellt. Dies stellt erneut einen Unterschied zu klassischer Software-Entwicklung dar, da diese Patches keine Updates im herkömmlichen Sinne sind, sondern das Spiel erst komplettieren.

4.4 Nachbereitung (Post Mortem)

Wir haben nun einen Zeitpunkt erreicht, an dem die Produktion des Spiels eigentlich komplett abgeschlossen ist, aber eben nur eigentlich. Um ein Projekt endgültig abzuschließen, sind noch einige wichtige Schritte notwendig.

Während und vor allem auch am Ende eines Projekts ist es wichtig, aus den Erfahrungen und Fehlern der Entwicklung für zukünftige Projekte zu lernen. Dazu ist es üblich, ein sogenanntes *Post Mortem*[24] anzufertigen. Dieses Dokument stellt eine Nachbetrachtung der Entwicklung dar. Jeder, der am Projekt teilgenommen hat, bekommt so die Chance, positive und negative Kritikpunkte einfließen zu lassen sowie Verbesserungsvorschläge zu machen, um künftige Entwicklungen zu optimieren. In der Praxis werden häufig schon sehr früh im Entwicklungsprozess Punkte notiert, um sie später im Post Mortem zu sammeln.

Ein weiterer wichtiger Bestandteil der Nachbereitung ist die Archivierung des Projekts (auch um es später wiederverwenden zu können). Die Archivierung erfolgt, indem ein sogenanntes *Closing Kit* angelegt wird. Es beinhaltet alle Design-Dokumente, den gesamten Quellcode, sämtliche Multimedia-Dateien und alle anderen Dinge, die verwendet wurden, um das Spiel zu entwickeln.

[24] Eigentliche Bedeutung: Post Mortem (lat.) = „Nach dem Tod“

Mit dem Abschluss dieses Kapitels haben wir nun einen Einblick in den groben Entwicklungsablauf eines modernen Spiels erhalten. Im Zusammenhang mit dem vorangegangenen Kapitel sind nun die zentralen Inhalte der Methodik der Spieleentwicklung abgeschlossen.

5 Aktuelle Trends und Entwicklungen

Abschließend sollen nun noch einige ausgewählte aktuelle Trends behandelt werden. Wohin geht die moderne Spieleentwicklung? Welche Tendenzen sind erkennbar? Diese Fragen sollen im Zentrum dieses Kapitels stehen und anhand von zwei Entwicklungen exemplarisch beantwortet werden. In Abschnitt 5.1 soll dabei auf die Herausforderungen und Chancen plattformunabhängiger Entwicklung eingegangen werden. Abschnitt 5.2 versucht danach einen kurzen Einblick in die Prinzipien agiler Software-Entwicklung zu geben.

5.1 Plattformunabhängige Entwicklung

Viele moderne Spiele werden heutzutage sowohl für den PC als auch für die entsprechenden aktuellen Produkte der Konsolenhersteller produziert um den entsprechenden Herstellern ein Vielfaches an Gewinn zu bescheren. Natürlich sollte damit im Idealfall nicht ein Vielfaches an Aufwand einhergehen.

Computerspiele plattformunabhängig zu entwickeln stellt Spielestudios jedoch vor ganz eigene Herausforderungen. In der Theorie sieht das Ganze etwa folgendermaßen aus: Man produziert ein Spiel, veröffentlicht es für drei verschiedene Plattformen und streicht den dreifachen Gewinn bei einfachem Aufwand ein. Praktisch sieht das ganze natürlich nicht ganz so einfach aus.[25] Schon die Hardware macht einen Strich durch diese Rechnung, denn jede Plattform hat andere Möglichkeiten und Leistungsprofile (man denke dabei beispielsweise nur an die verschiedensten Eingabegeräte (Tastatur, Maus, Gamepads, ...) oder die variierenden Möglichkeiten des Speicherns) und das macht plattformübergreifende Entwicklung schwierig. Trotzdem will man natürlich versuchen, den Anteil wiederverwendbaren Codes zu maximieren. In der Praxis sieht das meist so aus, dass auf einer entsprechend niedrigen Hardware-Abstraktions-Schicht für alle Plattformen uniform entwickelt wird und nur „darüber“ auf die Eigenheiten eingegangen wird, die jede Plattform mit sich bringt. Häufig führt dies dazu, dass man sich bezüglich der Leistungsfähigkeit auf den kleinsten gemeinsamen Nenner einigt. Dies erleichtert zwar die Arbeit und reduziert Kosten, führt aber oft dazu, dass die Möglichkeiten bestimmter Plattformen nicht ausgenutzt werden. Genau deshalb ist diese Entwicklungs-Methodik in der Spieler-Szene nicht gern gesehen und es ertönt häufig Kritik, dass auf diese Weise vermehrt mittelmäßige Spiele entstehen.

Ein besserer Ansatz ist es, die jeweiligen Fähigkeiten der einzelnen Plattform geschickt zu nutzen. Dies erfordert zwar mit sehr hoher Wahrscheinlichkeit mehr Aufwand, aber das entstehende Spiel ist im Regelfall deutlich besser.

5.2 Agile Entwicklungsmethoden

Agile Entwicklungsmethoden stellen Möglichkeiten bereit, auf die immer unvorhersehbarer werdende Umgebung, für die ein Software-Produkt entsteht, zu reagieren. Dies geschieht nicht dadurch, dass man mit einem Riesenaufwand von Anfang an versucht, jedes mögliche Szenario zu planen. Vielmehr legt agile Entwicklung Wert darauf, es zu ermöglichen, durch kurzfristige Zeitplanung während des gesamten Projekts, permanent Änderungen vornehmen zu können. Die agile „Bewegung“ in der Software-Industrie hat ihren Ursprung im *Manife-*

[25] Es lässt sich etwa sagen, dass die Entwicklung eines Spiels für eine zweite Plattform die Kosten veranderthalbfacht.

sto for Agile Software Development[26], das 2001 durch eine Gruppe von Software-Experten veröffentlicht wurde. Auf der Basis dieses Dokuments interpretiert, geht agile Software-Entwicklung nach [Gli02] von den folgenden Prinzipien aus:[27]

- Je kleiner die Teilaufgaben eines Software-Projekts und umso kürzer die Rückkopplungszyklen sind, desto besser.
- Der Kunde gestaltet das Produkt während seiner Entstehung.
- Je freier und konzentrierter die Entwickler arbeiten können, desto besser.
- Die Qualität wird an der Quelle sichergestellt.
- Keine Arbeit auf Vorrat.

Das vornehmlichste Ziel ist demnach, den Kunden durch schnelle und kontinuierlich verbesserte Bereitstellung der Software zufrieden zu stellen. Durch entsprechend kurze iterative Entwicklungsschritte und ausführliche Dokumentation wird erreicht, dass jederzeit Änderungen der Kundenwünsche in die Spezifikation der Software einfließen können.

Diese Flexibilität stellt den wesentlichen Vorteil agiler Entwicklung dar und macht sie zu einer (wenn nicht *der*) zukunftsfähigen Entwicklungsmethodik auch und gerade in der Spieleindustrie. Konkrete Beispiele für agile Methoden sind *Extreme Programming* und *Scrum*. Für eine genaue Beschreibung dieser und weiterer agiler Methoden wird auf [Abr02] verwiesen.

5.3 Weitere Trends

Die beiden erläuterten Bewegungen sind bei weitem nicht die einzigen, die in der Spieleentwicklung zu beobachten sind. Ein genaueres Eingehen auf noch mehr Tendenzen würde hier jedoch zu weit führen. Weitere aktuelle Trends sollen daher nur noch genannt werden:

So ist ein klarer Zukunftstrend auch in der *domainspezifischen Entwicklung* und den damit verbundenen *software factories* zu erkennen.[28] Auch der Siegeszug von *Java* macht vor der Spielindustrie nicht halt. So werden beispielsweise viele Handy-Spiele heute auf Java-Basis entwickelt. Gleiches gilt für entsprechende *Flash*-Anwendungen. Die erwähnten *Handygames* an sich stellen in Zeiten immer leistungsstärker werdender Mobiltelefone wohl auch einen klaren Trend dar.

Im Bereich der Entwicklungsmethodik lässt sich vermehrt beobachten, dass immer mehr Wert auf Wiederverwendung gelegt wird. Dazu kommen *Engines, Frameworks* und *Development Suites* stärker zum Einsatz als je zuvor.[29]

Auf der wirtschaftlichen Seite kann man klar feststellen, dass Konsolenspiele gegenüber PC-Spielen immer mehr Marktanteile gewinnen. Dabei ist dieser Trend in den USA noch stärker zu beobachten als in Europa.

Auf weitere Tendenzen soll nun nicht mehr eingegangen werden. Die genannten Trends sollen genügen um einen kurzen Einblick in aktuelle Entwicklungen zu liefern. Mit diesem kurzen Blick auf aktuelle Standards und zukunftsfähige Trends in der Spieleentwicklung ist auch dieses letzte Kapitel der Arbeit beendet.

[26] Siehe [www1].
[27] Eine detailliertere Beschreibung dieser Prinzipien findet sich in [Lud03].
[28] Vertiefend dazu wird [FuS05] und [Fur06] empfohlen.
[29] Nähere Informationen zu diesen Themen liefern die entsprechenden Seminararbeiten.

Literaturverzeichnis

[Abr02] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, Juhani Warsta, Agile software development methods - Review and analysis, University of Oulu, 2002

[Bal00] Helmut Balzert, Lehrbuch der Software-Technik, Spektrum Akademischer Verlag, Berlin, Heidelberg, 2000

[Bri04] Mathias Bricke, Die Entwicklung von Computerspielen, 2004

[Cha06] Heather Chandler, The Game Production Handbook, Thomson Learning, Massachusetts, 2006

[CNS04] David Callele, Eric Neufeld, Kevin Schneider, Requirements Engineering and the Creative Process in the Video Game Industry, University of Saskatchewan, Saskatchewan, 2004

[Coc02] Alistair Cockburn, Agile Software Development, Addison Wesley, 2002

[FuS05] Andre Wilson Brotto Furtado, Andre L. M. Santos, Using Domain-Specific Modeling towards Computer Games Development Industrialization, Universidade Federal De Pernambuco, Recife, 2005

[Fur06] Andre Wilson Brotto Furtado, SharpLudus: Improving Game Development Experience Through Software Factories And Domain-Specific Languages, Universidade Federal De Pernambuco, Recife, 2006

[Gli02] Martin Glinz, Vorlesungsskript - Software Engineering I, Institut für Informatik der Universität Zürich, Zürich, 2002

[Gul05] Danilo Gulamhussene, Buccaneer Inc. Entwicklung von Computerspielen, Studienarbeit an der Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2005

[Ham03] Wolfgang Hamann, Goodbye Postmortems - Hello Critical Stage Analysis (CSA), 2003

[Iri05] Daniel Irish, The Game Producer's Handbook, Thomson Learning, 2005

[Ke06a] Clinton Keith, Agile Methodology in Game Development: Year 3, Game Developers Conference 2006

[Ke06b] Clinton Keith, Agile Methods in Software Development (a real world story), San Diego Software Industry Council 2006

[LiW01] Balazs Lichtl, Gabriel Wurzer, Software Engineering in Games, Technical University Vienna, Wien, 2001

[Lop06] Paulo Lopes, Next-Gen Game Design: (Valve's Cabal Process), University of Coimbra, Coimbra, 2006

[Lud03] Matthias Luder, Open Source Software Development, Seminararbeit am Institut für Informatik, Universität Zürich, Zürich, 2003

[MaN04] Maic Masuch, Lennart Nacke, Power And Peril Of Teaching Game Programming, University of Magdeburg, Magdeburg, 2004

[Min07] Mark Minas, Vorlesungsfolien - Software Engineering I, Vorlesung an der Universität der Bundeswehr München, Neubiberg, 2007

[MMN04] Simon McCallum, Jayson Mackie, Lennart Nacke, Creating a Computer Game Design Course, Dunedin, 2004

[Nac04] Lennart Nacke, Co-Development, Delivery and Structural Analysis of a Computer Game Course, Otto-von-Guericke-University Magdeburg, Magdeburg, 2004

[Nac05] Lennart Nacke, Facilitating the Education of Game Development, Diplomarbeit am Institute for Simulation and Graphics der Otto-von-Guericke Universität Magedeburg, Magdeburg, 2005

[Rab05] Steve Rabin, Introduction to Game Development, Thomson Learning, 2005

[Ruh01] W. Ruh u. a., Enterprise Application Integration, Wiley, 2001

[Web06] Niels Weber, Spiele-Software und Open Source, Diplomarbeit am Lehrstuhl Informatik und Gesellschaft der Technischen Universität Berlin, Berlin, 2006

[Wol07] Robert Wolf, Kurzeinführung in weitere agile Softwareprozesse, Freie Universität Berlin, Berlin, 2007

Online-Referenzen

[www1] www.agilemanifesto.org

[www2] www.gamasutra.com

[www3] www.geemag.de

[www4] www.gelegenheitsspieler.de

[www5] www.introgamedev.com

[www6] www.ultimategameprogramming.com

[www7] http://de.wikipedia.org/wiki

[www8] http://www.heise.de/tp/r4/artikel/18/18284/1.html