SCRUM GAME

udviklingen af en metode til kreation af spil

Et speciale udarbejdet af Søren Vejgaard, 2007 - Vejledt af Thessa Jensen - Bedømt af Michael Valeur

A a l b o r g U n i v e r s i t e t

Titel: SCRAME – udviklingen af en metode til kreation af spil.

Projektperiode: Fra d. 1. februar 2007 til d. 31. juli 2007

Semester: Humanistisk Informatik, 10. Semesters Multimedier

Deltagere:

_____

Søren Vejgaard

Vejleder: Thessa Jensen

Censor: Michael Valeur

Oplagstal: 5

Typeenheder: 205.000

Sideantal: 85 normalsider

INDHOLD

Dedikeret til mindet om mine to højtelskede bedstefædre…

INTRO

FORORD

Det speciale, der i læsende stund ligger foran dig, dokumenterer den proces, jeg har været igennem for at undersøge, hvilke processer og metoder der ligger bag udviklingen af spil. Ved at undersøge hvad spil er, hvad spiludviklingsprocessen minder om, hvilke problemer spilbranchen oplever, hvilke processer spillitteraturen foreslår, at man anvender, samt hvordan man kan planlægge et spilprojekt, har jeg opsat en metode til brug i udviklingen af spil. Denne metode har jeg valgt at kalde SCRAME. SCRAME er bygget på det samme *framework* som de agile metoder, og den er opsat i et forsøg på at tilbyde en metode, der opsætter rammer for både processen og planlægningen. Metodens formål er at sikre, at det vigtigste succeskriterium ved et spil er opfyldt – at det er underholdende.

Jeg vil gerne takke Thessa for god og trofast faglig og personlig vejledning gennem både mit 9. og 10. semester på overbygningen i Multimedier. Endvidere en tak til mit kæreste eje, Mette, for at lege fotograf og ufrivilligt tage del i de frustrationer, jeg er stødt ind i undervejs. Også en tak til min bror Christian for input til opgaven.

Den vedlagte CD-ROM indeholder kilder, der ikke er offentligt tilgængelige. Ligeledes er projektet vedlagt i digitalt format.

LÆSEVEJLEDNING

Hvis læseren af denne opgave blot vil have kendskab til SCRAME metoden, anbefales det at læse Fase 6. Hvis du ikke har kendskab til de agile metoder, vil det formodentligt være en fordel at læse Fase 5 først, da denne blandt andet indeholder en beskrivelse af disse.

Hvis du som læser er interesseret i at vide mere om de processer og metoder, der er tilgængelige i spillitteraturen, anbefales det at læse Fase 4, der også indeholder mine kommentarer til forfatternes idéer og tanker.

I opgaven anvender jeg kursiv til at markere ord eller sætninger, der er på engelsk eller har engelsk ordlyd. Hensigten med dette er at gøre opgaven mere læsevenlig.

Kildehenvisninger er struktureret som følger:

[Forfatters efternavn, Årstal for kilden: Sidetal (eller anden henvisning ved digital kilde)]

OPBYGNING AF OPGAVEN

Nedenstående er en beskrivelse af de afsnit specialet indeholder og deres relation. Hensigten med afsnittet er at synliggøre opbygningen af opgaven, og hvorledes jeg har løst den senere opsatte problemstilling.

Indledning

I dette afsnit vil jeg beskrive rammerne for opgaven. Målet er at give dig som læser en forståelse for opgavens formål og mål. Det vil sige en beskrivelse af, hvorfor det er interessant at kigge på spiludvikling som proces, og hvorfor jeg ønsker at bidrage til udviklingen af metoder til kreationen af spil.

Fase 1 – Definition af spil

Mit første trin er at undersøge, hvad spil minder om, for derved at undersøge hvilke processer der ligger bag skabelsen af disse. Før jeg kan gøre dette, altså sige noget om hvad spil minder om, bliver jeg nødt til at definere, hvad et spil egentligt er. Mit første trin er derfor at lave en beskrivelse af de elementer, et computerspil indeholder.

Fase 2 – Spil, film og IT-systemer

Når jeg har undersøgt, hvad et spil egentligt er, kan jeg begynde at se på, hvad spil minder om. Her ønsker jeg at kigge på skabelsen af film og IT-systemer, da det er områder med veldefinerede processer og traditioner. Målet er at undersøge, hvor det er muligt at hente inspiration til opsætningen af en proces til spiludvikling.

Fase 3 – Kroghs problematikker

For at kunne opsætte en metode til spiludvikling kræver det, at man har kendskab til de problemer, spilbranchen oplever. Jeg ønsker derfor at inddrage Tea Krogh, da hun i sit speciale forsøger at beskrive de problemer, spilbranchen oplever.

Fase 4 – Gennemgang af eksisterende spillitteratur

For at kunne bidrage til udviklingen af spiludviklingsmetodikker, kræver det, at man har kendskab til det stadie spilbranchen befinder sig på. Derfor vil jeg undersøge, hvilke metoder der allerede er beskrevet til udvikling af spil. Jeg gør dette ved at gennemgå den eksisterende spillitteratur med henblik på at undersøge de processer, der bliver foreslået og beskrevet.

Fase 5 – Planlægning i spiludvikling

I min gennemgang af spillitteraturen opdagede jeg, at der i mine øjne er en mangel på veldefinerede metoder til spiludvikling. De fleste forfattere fremsatte usammenhængende og selvmodsigende metoder, der var svære at anvende som udgangspunkt, så jeg blev nødt til at lede efter inspiration andre steder. Jeg kunne dog konkludere, at det vigtige ved processen er, at spil skal være underholdende og at dette medfører, at spil ofte ændrer sig undervejs i processen. Der er altså behov for en metode, der tager højde for dette. Det første problem var dog at undersøge, hvorledes det er muligt at planlægge et forløb, hvor man forventer, at der vil opstå ændringer undervejs. Derfor valgte jeg at belyse, hvorledes de agile systemudviklingsmetoder tager højde for dette, da disse er systemudviklingstraditionens løsning på udviklingen af IT-systemer, hvor kravene forventes at ændre sig undervejs.

Fase 6 – Opstilling af SCRAME metoden

Med udgangspunkt i de agile metoder følte jeg mig nu klar og godt rustet til at opstille en udviklingsmetode. Metoden, jeg har valgt at navngive SCRAME, beskriver en proces, hvor de elementer spillet skal indeholde bliver defineret undervejs i et forsøg på at gøre spillet så underholdende som muligt. Med SCRAME-metoden er målet for opgaven fuldendt - jeg har bidraget til udviklingen af metoder til spiludvikling og opsat en metode, jeg håber, vil blive et fundament for oprettelsen af en spiludviklingstradition, der fokuserer på underholdnings-værdien.

Diskussion

I dette afsnit har jeg inddraget de problemstillinger, der er opstået i forbindelse med SCRAME metoden. Heriblandt problemerne ved de agile metoder og den iterative tilgang, spillitteraturens problemer og specialets relation til min 9. semesters opgave.

ABSTRACT

The game industry is still young of age. In the seventies and eighties game development started as hobby projects in homes and garages. The development costs where low and a title did not need to sell more than a couple of hundred copies before profit where made. As the technology and the marked have changed game development is now a million dollar business. The team behind a single title can consist of many hundred people, and the sales number has to reach 300-500,000 units to break even. This rapid development has resulted in a number of challenges.

Game development has only been accepted as an academic field for a few years. This means that game development research is years behind the field itself, and this could be one of the reasons why the business is struggling to make profits. Compared to other industries like software development, the game business had not had the time to develop methods and techniques for the new and complex times. In this master thesis I have tried to do something about this.

In this assignment I have created a game development method. The first step in doing so was to define what a game really is. If you want to investigate which processes that are similar to game development, you have got to know what a game consist of. So this will be step one. Afterwards I want to compare game development to the creation of movies and software systems as these fields have well defined methods and techniques. My research resulted in the finding that games, movies and software systems do not have that much in common. Movies are like games created to entertain, but the process is so much different than the creation of a game. Software consists of code which is an essential part of games as well, but they are used as tools and not to entertain. The conclusion is though that software development properly can be used as a basis for a game development method.

The third step was to investigate what kind of problems the game development business is experiencing. In order to do so I used a research made by another student from Aalborg University. The research suggests four problems; internal problems with coordination and planning, adjustment of expectations, estimation and testing problems.

The next task in order to create a game development method where to investigate the research already made. In order to do so I read the following writers: Eric Bethke, Chris Crawford, Tracy Fullerton, Bob Bates, Richard Rouse and Dan Irish. The conclusion I made where, that the methods suggested where incoherent and contradictory and basically not a great basis for the development of a game development methodology. Hence I had to search elsewhere.

The one thing I did realize during the reading where, that the most important part in a game is that it is entertaining, and that this is a process where the requirements to the game changes all the time. The process most similar to this is the methods used in agile development. This is the software development traditions answer to a methodology that incorporates a process where the requirements are expected to change along the way. The next step where therefore to study the process behind these methods to see if they can find use in game development.

With this knowledge I was ready to create a method for game development which I have chosen to call SCRAME. SCRAME is built on an agile framework trying to incorporate a process where the game does not have to be defined in every aspect before development starts. It is done in order to make sure that the game does, what it is supposed to do – entertain. With the SCRAME method I have done what I wanted to do with this master thesis – create a basis for the development of a game development methodology.

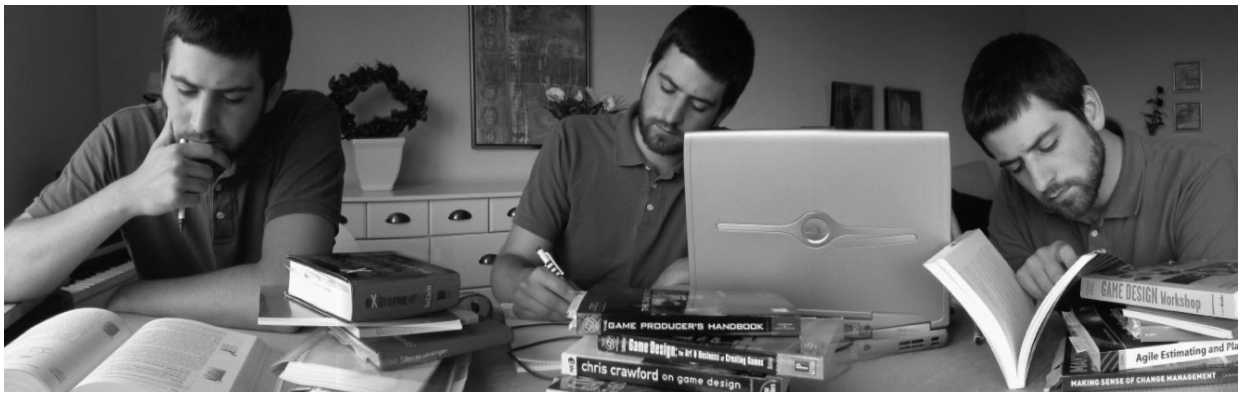
OM FORFATTEREN

Hver gang vi mennesker oplever noget, påvirker det den måde, vi betragter vores omverden på. Derfor, som jeg senere skriver i indledningen, gør jeg meget ud af at forsøge at forstå de forfattere, jeg læser gennem opgaven. Jeg mener derfor, at det er passende at give dig som læser et indblik i, hvem jeg er. Primært med fokus på min uddannelsesmæssige baggrund.

Jeg har en bachelor i Informatik ved Teknisk Naturvidenskabeligt fakultet. Uddannelsen blev oprettet i 1997 med det formål at tilbyde erhvervslivet nyuddannede, der både havde en forståelse for den tekniske og menneskelige side af systemudvikling. Studiet bestod derfor langt størstedelen af tiden af udviklingen af it-systemer med vægt på brugergrænsefladen. I længden blev det dog frustrerende at være lidt af det hele og ingenting af alting, så efter bachelorgraden sprang jeg ud som fuldblodshumanist ved Humanistisk Informatik – Multimedie overbygningen. Her har jeg brugt mine fire semestre på at skrive opgave om narrative spil, et praktikophold ved mobilspilfirmaet Progressive Media ApS, en opgave om ledelse af produkter i forandring, hvor jeg sideløbende var projektleder for *9K Games*, og nu mit speciale. *9K Games* var et projekt jeg startede med tre andre studerende med samme uddannelsesmæssige baggrund som jeg selv. Formålet var at få kendskab til spiludvikling på et konkret niveau – rent personligt var det for at afprøve rollen som projektleder for et lille hold.

Sideløbende med specialet har jeg arbejdet på deltid som Assisterende Producer ved Progressive Media ApS.

Efter denne introduktion vil jeg bevæge mig videre og præsentere opgaven.



INDLEDNING

Siden det første registrerede computerspil *Tennis for Two* tilbage i 1958 er der sket meget med teknologien og markedet. Igennem 70'erne og 80'erne blev spil primært udviklet af enkelte personer eller små teams som hobbyprojekter. Udviklingsomkostningerne var små og solgte én titel bare et par hundrede eksemplarer kunne det være en god forretning. Som teknologien og markedet har ændret sig, står spiludviklere i 2007 over for helt andre udfordringer. Spiludviklingsholdet kan bestå af flere hundrede mennesker og salgstallene for de største produktioner skal gerne op mellem 300.000-500.000 solgte enheder, før der er tale om et overskud [Bethke, 2003:16]. Ligeledes har betydningen af computerspil også ændret sig markant. Computerspil har indtaget sin rolle i oplevelsesøkonomien og fået mange flere funktioner end blot underholdning. Det er blevet et kulturfænomen, der blandt andet bruges som medie til marketing, til at videreføre budskaber og til undervisning.

Overstående faktorer har alle en fællesnævner – de øger kompleksiteten af udviklingen af spil.

Spilindustrien er stadig ung og i udvikling, hvilket er den primære årsag til at industrien kæmper med at indrette sig efter disse nye og komplekse forhold. Sammenlignet med andre industrier, der også bygger på udviklingen af software, har spilindustrien ikke haft tiden til at udvikle metoder og teknikker til at håndtere de nye tider [Rollings, 2004:229-231]. I dette speciale er det mit mål at gøre noget ved dette.

MOTIVATION

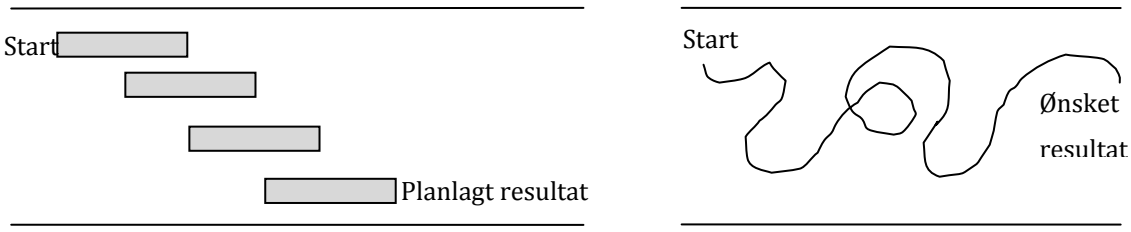
Jeg har igennem min bacheloruddannelse i Informatik under Teknisk Naturvidenskabelig Fakultet fået erfaring med systemudvikling, og de aspekter der er ved at lave systemer, der skal leve op til både kundens og brugernes krav. Blandt andet har jeg været med til at udvikle en applikation, der via en farveskala på et danmarkskort kunne vise benzinpriserne, en applikation der skulle fungere som løn- og registreringssystem for en café, samt en applikation til sprogtræning ved hjælp af talegenkendelse. En af de mest slående forskelle mellem systemudviklingen på Informatik og den erfaring jeg har fået med spil, er den enorme forskel, der er på modenheden af de to industrier. Der findes velafprøvede og veldokumenterede udviklingsmetoder til systemudvikling, mens jeg oplevede at spilindustrien var meget bagud på dette punkt. Der findes ingen fælles begreber eller teknikker, der omfatter hele processen i at

lave spil, og det må uden tvivl give nogle problemer i branchen. Specielt i denne tid hvor teknologien er i konstant forandring, og udviklingen af spil bliver, som en direkte konsekvens heraf, endnu mere kompleks. Spilbranchen skal modnes, og der skal tilbydes teknikker og metoder, der kan håndtere de nye tider. Der er skrevet enkelte bøger, der dækker dette område, men de er primært skrevet af folk, der har arbejdet i spilindustrien, og derfor baserer deres konklusioner på praktiske erfaringer. Der er ikke noget fælles og entydigt billede af udviklingsprocessen i spilproduktion, hvilket vanskeliggør kommunikationen og formodentlig er årsag til at meget arbejde går tabt. Ligeledes er spilindustrien blevet et sted, hvor det er svært at tjene penge. Dette øger blot behovet for værktøjer til at styre processen og sikre budgettet.

Det er først i de senere år, at spil er kommet i betragtning som et akademisk felt, så rent forskningsmæssigt halter spiludviklingen stadig bagefter. Det Danske Akademi for Digital, Interaktiv Underholdning er ikke mere end et par år gammelt, og det samme gør sig gældende for spillinjerne ved Aalborg Universitet. At spil nu bliver betragtet som et akademisk fagområde, giver mig muligheden for at undersøge et område, der formodentlig ikke er forsket meget i. De nye og komplekse tider og den voldsomme udvikling branchen har oplevet må betyde, at behovet for metoder og teknikker til at styre processen er steget tilsvarende. Målet med mit speciale er at bidrage til en udvikling af disse.

FREMGANGSMÅDE

I min opgave vil jeg følge en *agile* tankegang. Det vil sige, at jeg ikke arbejder mod et planlagt resultat, men at jeg erkender, at jeg gennem processen opnår nye erkendelser, jeg kan bygge videre på. I stedet for at gå igennem en række på forhånd kendte faser for at nå et bestemt resultat, ønsker jeg at lade min nye viden guide mig igennem processen. For at fastholde emnet vil jeg dog beskrive formålet med opgaven således, at dette står tydeligt igennem hele opgaven. Sat op mod hinanden vil de to processer fremstå som opsat på den følgende side:



Figur 0.1: Traditionel tankegang til venstre og agile tankegang til højre (Egen tilvirkning).

Den venstre model afbilleder et forløb, hvor man går igennem en række planlagte faser for at nå et planlagt resultat. Til højre afbilledes modellen over projektets forløb – en proces hvor man hele tiden anvender den nyerhvervede viden til at beslutte det næste trin i processen, for derved at nå frem til det ønskede resultat. De to modeller er identiske med tankegangene fra traditionel systemudvikling (som for eksempel vandfaldsmetoden) og de agile metoder (som eksempelvis Scrum).

HERMENEUTISK TILGANG

I opgaven ønsker jeg at anvende en hermeneutisk tankegang. Dette vil igennem projektet komme til udtryk ved, at der før og efter hvert emne i opgaven vil være en beskrivelse af min viden og formodning omkring emnet. Formålet med denne proces er at synliggøre min egen viden og fordomme om emnet, så denne i mindre grad vil spille ind på oplevelsen af emnet, og ligeledes gøre det muligt at synliggøre, hvilken ny indsigt jeg er kommet i besiddelse af, når et emne er behandlet.

Grundlaget for denne tilgang bygger på en hermeneutisk tankegang. Winograd og Flores skriver følgende om hermeneutikkens anvendelse:

> *"The opposing approach, ... , takes the act of interpretation as primary, understanding it as an interaction between the horizon provided by the text and the horizon that the interpreter brings to it"* [Winograd & Flores, 1987: 28].

En tolkning er altså en interaktion mellem afsenderens og fortolkerens forståelse. Det er vigtigt at betydningen af *text* i denne sammenhæng ikke bare dækker over skreven tekst, men også kan fortolkes som teorier, ideer, systemer, processer med mere. Winograd og Flores forsætter:

> *"Any individual, in understanding his or her world, is continually involved in activities of interpretation. That interpretation is based on prejudice (or pre-understanding),*

> *which includes assumptions implicit in the language that the person uses"* [Winograd & Flores, 1987: 28].

Fortolkning er altså baseret på forforståelse, der inkluderer de formodninger fortolkeren har. En fortolker har altså en forforståelse, der gør tolkninger subjektive, men hvordan kan man så tilstræbe en objektiv tolkning. Om dette skriver Winograd og Flores:

> *"We can become aware of some of our prejudices, and in that way emancipate ourselves from some of the limits they place on our thinking. But we commit a fallacy in believing we can ever be free of all prejudice"* [Winograd & Flores, 1987: 30].

Det er med dette udgangspunkt, og som en erkendelse af dette, at den beskrevne tilgang er valgt. I erkendelse af, at man aldrig kan blive fri for sine forforståelser, vil jeg i stedet forsøge at bevidstgøre mig selv om disse, i et forsøg på at bryde de grænser, de ellers kan lægge på mine tanker. Denne tilgang stemmer ligeledes overens med tankerne bag den hermeneutiske cirkel. Det vil sige forestillingen om, at betydningen af alt tekst er afhængig af konteksten, den bliver fortolket i og fortolkerens forforståelse. Som Winograd og Flores skriver:

> *"What we understand is based on what we already know, and what we already know comes from being able to understand"* [Winograd & Flores, 1987: 30].

Med denne tilgang håber jeg at synliggøre mine egne forforståelser og derved tilstræbe objektivitet gennem opgaven.

FORMÅL

For at have en rettesnor igennem specialet, vil jeg i det følgende definere projektets formål – det vil sige, at jeg vil beskrive hensigten med nærværende opgave. Mine observationer, der beror på såvel beskrivelser i litteraturen, som egne erfaringer i praktik og på studieture, peger på, at der ikke findes ét fælles og entydigt billede af udviklingsprocessen i spilproduktion. For at spilindustrien kan modnes, er det nødvendigt, at der sker en videreudvikling af de teknikker og metoder, der omhandler processen i spiludvikling. Formålet med nærværende speciale kan derfor formuleres som følger:

> *Jeg ønsker gennem mit speciale at bidrage til udviklingen af metoder og teknikker, der kan anvendes i kreationen af spil.*

Jeg vil altså i opgaven arbejde ud fra dette fastsatte formål. Målet står mere udefineret hen, men vil blive tydeligere gennem processen efterhånden som jeg opnår ny indsigt og viden. Fokus for

opgaven er således at tilbyde et værktøj til spiludviklingsprocessen, der gør det muligt at overholde såvel budget som *deadlines*. Jeg afgrænser mig således også fra alle de kreative processer i udviklingen, idet jeg ikke vil fokusere på, hvorledes man skaber et spilkoncept, skaffer finansiering til projektet, designer spillet og så videre. Mit fokus er at skabe nogle rammer, de kreative processer kan fungere inden for.

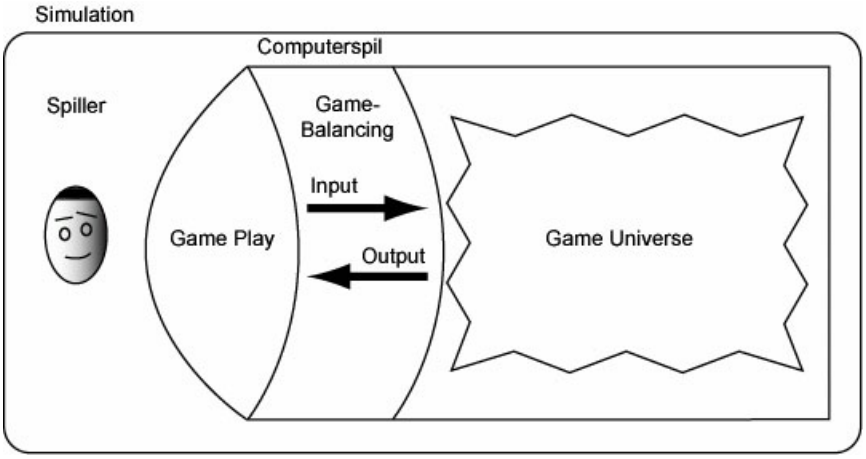
Hvordan kan jeg så bidrage til udviklingen af metoder og teknikker til spiludvikling? Det første trin må være at definere hvad et spil egentligt er, i et forsøg på at synliggøre de elementer et spil består af samt undgå, at der undervejs i processen skal opstå tvivl om dette. Nærværende speciales definition af et spil kan således siges at være en hypotese – dog ikke en jeg i opgaven vil forsøge at bekræfte eller afkræfte. Men en hypotese i den forstand, at det vil være rapportens syn på et spil og de elementer et sådant indeholder. Hvis denne senere kan afkræftes, eller det viser sig, at den ikke stemmer overens med virkeligheden, er det muligt, at projektets resultat ikke kan finde anvendelse.

FASE 1 – DEFINITION AF SPIL

Med dette afsnit ønsker jeg at belyse, hvad spil er og ikke er. Hvad kendetegner et spil, og hvad er et spil ikke? Formålet med dette afsnit er ikke at skabe en entydig definition af hvad spil er, da dette ligger udenfor projektets formål. Formålet er at danne en grundlæggende forståelse af spil, og de elementer et sådant indeholder, da dette må have en indflydelse på udviklingsprocessen og derved også definere nogle rammer for denne. Min formodning er, at spil er kendetegnet ved interaktionen mellem spilleren og spillet, og at det primære formål med spil er at underholde. Derved er det også disse to elementer, der påvirker den proces, hvorigennem spil skal skabes.

UNDERHOLDENDE INTERAKTIVE DIGITALE MEDIER

I et forsøg på at skabe et fællessprog i den danske spilindustri udarbejdede Claus Rosenstand og Per Kyed i 2003 et computerspilsmanifest. I denne opstillede de nedenstående model, til at definere et computerspil og de væsentligste elementer af et sådant:



Figur 1.0: Rosenstands og Kyeds visualisering af de elementer et spil indeholder [Rosenstand, 2004:5].

Et computerspil er en simulation. Det vil sige, at når et spil påvirkes af input så reagerer spillets output som en forventet følge heraf. Det vil sige, at vi har to parter: En spiller der leverer input, og et computerspil der leverer et forventet output. Spillerens interaktion med spillet foregår via

de regler, der bliver opsat i spillets *gameplay*. Et spils *gameplay* kan dermed siges at være de handlinger spilleren kan foretage sig i spillets *game universe*. *Game universe* er en model, der omfatter helheden af den verden spilleren kan bevæge sig i. *Game balancing* er forholdet mellem *gameplay* og *game universe* – et spil skal hverken være for nemt eller for svært. I brugssituationen set fra spillerens perspektiv har vi således et *gameplay*, der forudsætter *game balancing*, der så igen forudsætter *game universe* [Rosenstand, 2004:5].

Spil indeholder altså et regelsæt i form af *gameplayet*, der bliver synliggjort gennem *game balancing* og derved definerer de muligheder spilleren har i *game universe*. Men hvad er det så, der gør et spil underholdende? For at forstå dette inddrager jeg Greg Costykian og Celia Pearce, der begge har forsøgt at definere hvilke elementer, udover interaktionen, der gør et spil til et spil [Costikyan, 1994] & [Pearce, 1997]:

- Mål: Et spil skal indeholde mål eller gøre det muligt for spilleren at opsætte sine egne mål. Formålet med dette er at skabe en stræben efter opfyldelse af disse mål. Ofte opleves dette i spil ved at spilleren bliver givet en konkret mission; eksempelvis at redde verden fra en ond skurk. Ligeledes består et spil af delmål, som spilleren skal opfylde for at nå det primære mål.

- Modstand og forhindringer: Et spil er ikke underholdende, hvis ikke spillerens interaktion har en betydning for om spillets mål opnås eller ej. Hvis spilleren ikke skal gøre noget for at opfylde spillets mål, så vil dette ikke give spilleren en følelse af tilfredsstillelse, da der aldrig har været noget på spil [Costikyan, 1994: 14]. Et spil skal altså have en række delmål, der kan nås ved at klare en række forhindringer, der i sidste ende gør det muligt at opnå spillets mål.

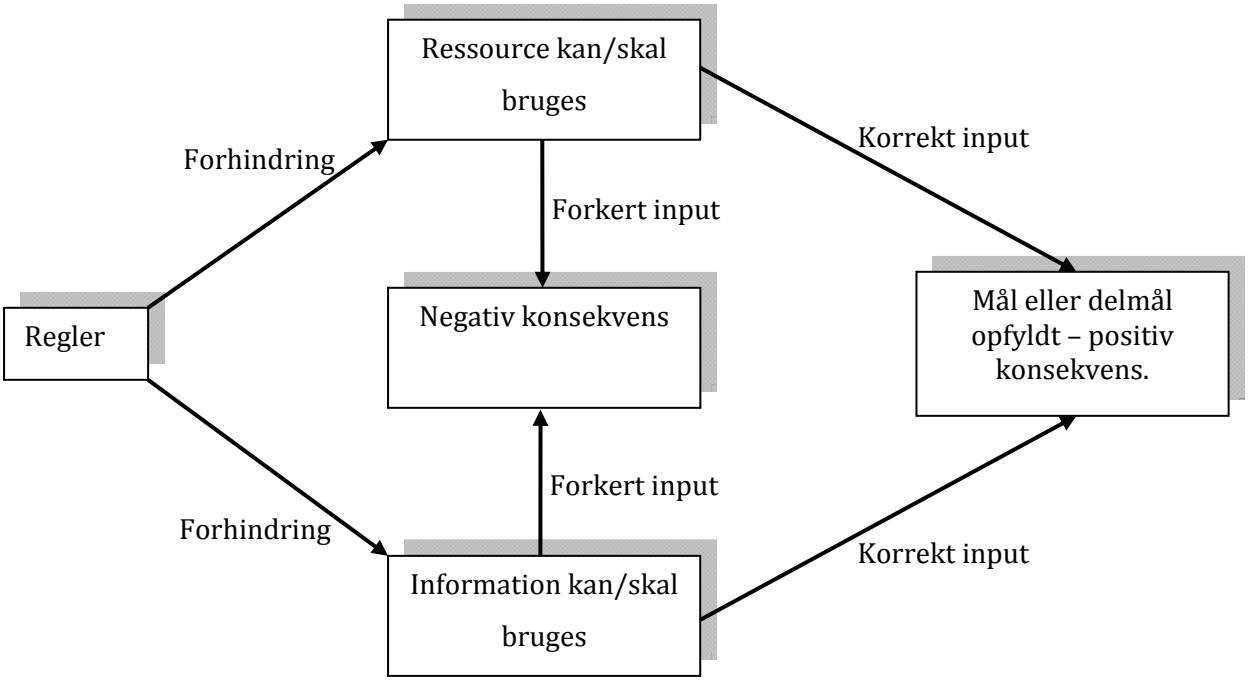
- Ressourcer: Spil indeholder ressourcer, der kan hjælpe spilleren med at passere forhindringerne. En ressource kan eksempelvis være en nøgle, der gør det muligt for spilleren at komme videre. Ligeledes kan ressourcer anvendes som belønning for at have klaret en forhindring og dermed nået et delmål. Eksempelvis ved at spilleren får adgang til et nyt våben eller lignende.

- Konsekvenser: Et spil skal indeholde konsekvenser, der bliver afspejlet i spillerens input. Hvis spilleren eksempelvis i et bilspil kører af banen og ind i et træ, skal der være en konsekvens af dette. Ligeledes, hvis spilleren giver det korrekte input og sætter en god tid, skal der være en positiv konsekvens af dette. Eksempelvis i form af at spilleren kan

komme til den næste bane eller får adgang til en ny og hurtigere bil (altså en ny ressource).

- Regler: I et brætspil vil reglerne i spillet blive udgjort af det regelsæt, der følger med i æsken. I computerspil bliver reglerne kodet ind i spillet og spilleren får så erfaring med disse via interaktionen. En regel er eksempelvis hvor hurtigt spilleren kan løbe, hvor højt han kan hoppe, hvordan han får point og så videre.
- Information: Et spil indeholder information, der enten kan være kendt af alle spillere, kun kendt af én spiller, kun kendt af spillet eller tilfældigt genereret. Manglende information, eksempelvis i form af et kodeord der skal anvendes til at passere en dør, kan udgøre en forhindring i spillet, og spilleren skal således finde den part, der har den ønskede information på det givne tidspunkt. På den måde kan distribution af information være med til at skabe en underholdende faktor i spillet.

Disse seks elementer har jeg opsat i nedenstående model, der viser sammenhængen og betydningen af de enkelte elementer:



Figur 1.1: Model der afbilleder elementerne i spil og deres sammenhæng [Egen tilvirkning]

Spillet udgøres altså af et regelsæt, der definerer de muligheder spilleren har i *game universe*. Spilleren møder forhindringer, der skal passeres enten ved hjælp af information eller en ressource spilleren besidder eller skal finde. Hvis der gives forkert input fejler spilleren – eksempelvis ved at spillerkarakteren dør - mens et korrekt input medfører, at et mål eller delmål

opfyldes og spilleren oplever en positiv konsekvens heraf – eksempelvis adgang til den næste bane.

DEFINITIONER AF GAME

Katie Salen og Eric Zimmerman opsætter i bogen *Rules of Play* en sammenligning af otte forfatteres definition af *game* [Salen, 2004]. De otte definitioner kan findes i Bilag 1. Salen og Zimmerman nedbryder først de enkelte definitioner for at belyse, hvilke elementer de indeholder. Dernæst opsætter de definitionerne i nedenstående tabel:

Elements of a game definition	**Parlett**	**Abt**	**Huizinga**	**Caillois**	**Suits**	**Crawford**	**Costikyan**	**Avedon |Sutton-Smith**
Proceeds according to rules that limit players	√	√	√	√	√	√		√
Conflict or contest	√					√		√
Goal-oriented/outcome-oriented	√	√			√		√	√
Activity, process, or event		√			√			√
Involves decision-making		√				√	√	
Not serious and Absorbing			√					
Never associated with material gain			√	√				
Artificial/Safe/Outside ordinary life			√	√		√		
Creates special social groups			√					
Voluntary				√	√			√
Uncertain				√				
Make-believe/Representational				√		√		
Inefficient					√			
System of parts/Resources and Tokens						√	√	
A form of art							√	

Tabel 1.2: Tabel over de elementer hver forfatters definition indeholder [Salen, 2004: Chapter 7 – *Comparing definitions*].

Som det tydeligt fremgår af tabellen, strækker de forskellige forfatteres definition af spil sig over vidt forskellige elementer. Interessante iagttagelser er blandt andet, at alle forfatternes definition, med undtagelse af Costikyans, indeholder elementet "*Proceeds according to rules that limit players*". Ligeledes indeholder fem af de otte definitioner "*Goal-oriented/outcome-oriented*". Dette stemmer meget godt overens med Costykians og Pearces beskrivelse af spil; regler muliggør eller begrænser spillerens interaktion og påvirker derved muligheden for at opnå et bestemt mål. En anden interessant iagttagelse er, at tre af forfatternes definitioner beskriver spil som værende en frivillig aktivitet. Hvis spil er en frivillig aktivitet, må det jo netop være en aktivitet vi indgår i for at blive underholdt. Dette betyder også, at det kræver et engagement fra spillerens side – han skal have lyst til det. Dette er nogle helt andre rammer, end hvad jeg kender fra systemudviklingen. Her er det funktionaliteten, der er i fokus, og anvendelsen er ofte arbejdsrelateret. Spil er altså – modsat eksempelvis IT-systemer – en frivillig aktivitet.

HVAD ER ET SPIL IKKE?

En definition af hvad spil er, må også nødvendigvis, om end ikke direkte, fortælle noget om hvad spil så ikke er. Her er computerspillets nærmeste sammenligningsgrundlag film og it-systemer. Førstnævnte fordi film, ligesom spil, i sidste ende begge er en del af underholdningsindustrien; deres formål er at underholde. It-systemer og computerspil har det tilfældes, at de begge er opbygget af kode, og derfor kan man være tilbøjelig til at tro, at der er et meget stort sammenligningsgrundlag.

Den primære årsag til at spil ikke er direkte sammenlignelige med film er interaktionen. Bob Bates skriver følgende om spil i relation til film:

> *"The basic interaction between a player and a game is simple: I do something. The game does something in response. This feedback is what distinguishes a game from every other form of entertainment. It's the interactivity that makes our games unique – without it, the player would just be watching a movie on a screen"* [Bates, 2001: 23].

I film forholder seeren sig passivt og har ingen indvirkning på, hvad der sker i det univers filmen udspiller sig i. Modsat er det i spil, hvor spilleren har mulighed for at påvirke *game universe* gennem de regler, der muliggør spillerens interaktion, og derved også bestemmer hvad spilleren kan og ikke kan. Inden for disse rammer kan spilleren altså frit bevæge sig rundt. Film og spil adskiller sig altså fra hinanden ved, at det kun er spil, der indeholder interaktion.

Computerspil og it-systemer har som nævnt det til fælles, at de begge er opbygget af kode. Men netop på det punkt hvor film og spil er ens, det underholdende element, er spil og it-systemer forskellige. It-systemer kan opfattes som et redskab, der skal hjælpe en person med at løse en opgave elektronisk, mens et computerspils opgave er at gøre en opgave eller forhindring så underholdende at løse som muligt. Processen i at udvikle computerspil og it-systemer må derfor netop adskille sig på dette punkt. Hvor processen i at udvikle computerspil formodes at være fokuseret på at gøre spillet underholdende, er den i systemudvikling baseret på at lave et værktøj, der kan løse en opgave som nemt og hurtigt som muligt.

OPSAMLING

Udgangspunktet for et computerspil er altså de indkodede regler, der muliggør interaktionen og opsætter rammer for, hvad der er muligt i *game universe*. Interaktionen må derfor være et vigtigt udgangspunkt for processen i at udvikle spil, ligesom den faktor at spil er en frivillig aktivitet. Forhindringer i spillet skal passeres ved hjælp af det regelsæt spillet opstiller, og det er dette tilsammen, der skal gøre spillet underholdende.

Efter nu at have set på hvad et spil er og hvilke elementer det indeholder, skal jeg beslutte mig for, hvad mit næste trin er. Jeg har i afsnittet belyst, hvordan spil adskiller sig fra film og systemudvikling, men kun ved at belyse de elementer disse indeholder. Men hvilken indflydelse har det på udviklingsprocessen at IT-systemer ikke skal underholde og film ikke er interaktive? I det efterfølgende afsnit vil jeg derfor forsøge at belyse de procesmæssige forskelle mellem spil, film og systemudvikling.

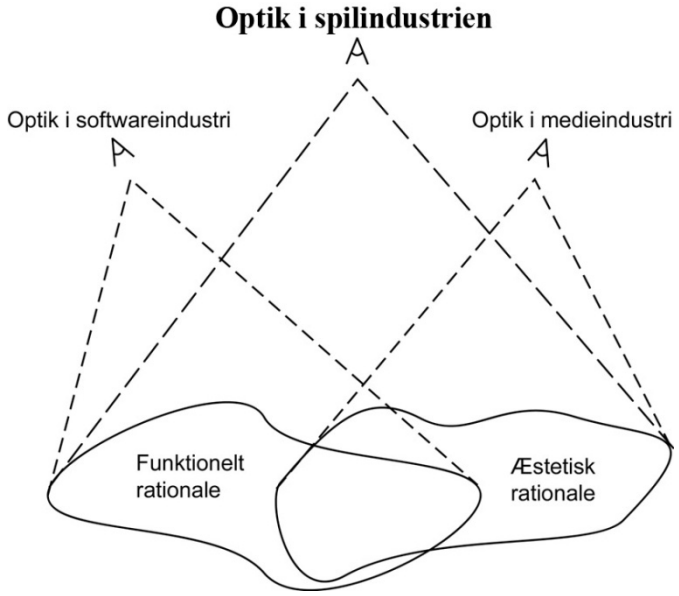
FASE 2 – SPIL, FILM OG IT-SYSTEMER

Spil og IT-systemer er begge baseret på kode, men formålet med de to produkter er som nævnt forskellige. Hvor det ene skal underholde, skal det andet fungere som et redskab til at løse en arbejdsopgave. Men hvilken indflydelse har dette på udviklingsprocessen? Jeg ønsker i dette afsnit at belyse, hvorledes spiludvikling adskiller sig fra udviklingen af traditionelle IT-systemer. Ved at belyse dette håber jeg at kunne beskrive, hvordan spiludvikling er anderledes end IT-udvikling og hvilken konsekvens dette har for processen. Ligeledes ønsker jeg at se på tilblivelsen af film, da dette har den dimension IT-systemer ikke har – de skal ligesom spil være underholdende. Ved at belyse processen i at skabe film, håber jeg også her at kunne definere, hvordan spil adskiller sig fra film, og hvorledes de minder om hinanden.

Formålet med afsnittet er altså at belyse processen bag film og software udvikling for at undersøge, om der er nogle paralleller til skabelsen af spil.

COMPUTERSPIL OG SYSTEMUDVIKLING

Jeg er af den opfattelse, at ethvert spil starter som en god idé – enten til et *gameplay*, en figur, en historie, et *setting* eller lignende. Procesmæssigt mener jeg dog, at udviklingen af spil i første omgang starter som systemudvikling, for uden programmering er der ikke noget spil. Det er igennem koden at reglerne og *gameplayet* for spillet bliver skabt. Men spil indeholder, i forhold til systemudvikling, flere aspekter, idet det ikke bare skal være funktionelt men også underholdende. Jeg formoder derfor at selve fundamentet i kreationen af spil må være koden, da det nødvendigvis er denne, der muliggør håndteringen af input og output. Men hvordan adskiller spiludvikling sig fra udviklingen af software? Under et oplæg ved *Dream Games* netværket i Aalborg [DreamGames, 2007], opstillede Claus Rosenstand modellen opsat på næste side:



Figur 2.1: Rosenstands illustration af optikken i spilindustrien [Rosenstand, 2007: slide 1].

I softwareindustrien skal et system oftest fungere som et 'værktøj'. Det vil sige, at det skal udføre og understøtte en række ønskede funktioner. Der er altså tale om, at man gennem processen arbejder ud fra et funktionelt rationale. I medieindustrien er der derimod tale om et (humanistisk) æstetisk rationale, idet der her bliver lagt vægt på sanseindtryk hos brugeren. Spilindustrien omfavner begge aspekter, idet det både skal udføre og understøtte en række ønskede funktioner (*gameplay*), samtidig med at det skal give et bestemt æstetisk udtryk (*game universe*). Spiludvikling indeholder altså i langt højere grad end systemudvikling overvejelser om det æstetiske udtryk.

Systemudviklingsmetoder som eksempelvis Objektorienteret Analyse og Design (OOA&D) tager udgangspunkt i, at konstruktionen af et IT-system kan ske ud fra en analyse af et brugsområde, der så bliver beskrevet via *use cases*, der så igen er med til at definere funktionerne i systemet [Mathiassen, 2001]. Dette er eksempelvis tilfældet, hvis en arbejdsgang, der nu foregår via papir, skal lagres elektronisk. Her kan systemudviklerne analysere, hvorledes de nuværende arbejdsgange er, og derudfra beskrive de funktioner, systemet skal have. Dette er ikke muligt i spil. Der er ikke noget område man kan gå ud og analysere, for at undersøge hvilke funktioner spillet skal have for at være 'fyldestgørende'. En undtagelse til dette er simulationsspil. Et eksempel på dette er det store antal af flysimulations-spil. Her eksisterer der et brugsområde, flyvning, der kan analyseres og nedbrydes med henblik på at definere de muligheder, spilleren skal have som pilot. Jo længere væk spillet befinder sig fra en realistisk simulation – desto vagere

bliver det brugsområde, der kan analyseres for at klarlægge funktionerne i spillet. Hvor IT-systemer kan bygge på en analyse af et genstandsfelt, er man altså i spiludvikling mere afhængig af en kreativ persons tanker i form af et designdokument, der i mere eller mindre grad er baseret på virkeligheden. Det er derfor meget sandsynligt at en metode som OOA&D ikke vil kunne anvendes i spiludvikling, idet den netop antager, at der eksisterer et brugsområde, der kan analyseres og nedbrydes.

Et andet problem, der delvist er en følge af ovenstående, er, hvorledes man så skal definere og beskrive funktionerne i spillet. *Gameplayet* kan brydes ned i opgaver, der minder meget om traditionelt systemudvikling. Eksempelvis: Spilleren kan trykke på *space,* hvorefter spillerkarakteren hopper. Langt sværere er det at beskrive kreationen af *game universe* brudt ned til en række funktioner, der kan tolkes i en traditionel systemudviklingstankegang. Et designdokument skal normalt også indeholde konceptskitser og en guide til det grafiske udtryk, der skal komme til udtryk i *game universe* [Bethke, 2003:224-225]. Et spils *gameplay* er regler opsat af koden, der kan nedbrydes til funktioner – altså en mulighed spilleren har i spillet. *Game universe* er den verden spillet foregår i og oftest meget afhængig af grafik, animationer, og lyde. Da IT-systemer som oftest ikke indeholder disse elementer, tager de heller ikke højde for udviklingen af disse – noget man bliver nødt til i spil. Dette passer meget godt overens med Rosenstands figur (Figur 2.1), hvor man i spil både har et funktionelt og æstetisk rationale. Systemudviklingen har primært det funktionelle rationale, hvilket også betyder at de to processer må adskille sig på dette punkt. Dette gør det yderligere interessant at se på, hvorledes film og spil adskiller sig fra hinanden, idet de også har denne dimension til fælles. Film har jo ligesom spil et univers, hvor fortællingen udfolder sig.

Spil adskiller sig altså rent procesmæssigt fra systemudvikling ved:

- I langt højere grad lægge vægt på det æstetiske rationale.
- Formålet er underholdning frem for at fungere som et redskab.
- Ikke at have et brugsområde der kan analyseres og nedbrydes (en undtagelse er simulatorer).

Efter at jeg nu har belyst hvordan spiludvikling adskiller sig fra systemudvikling, vil jeg kigge nærmere på, hvad der adskiller det at lave spil fra skabelsen af film.

COMPUTERSPIL OG FILM

Hvor systemudvikling skal skabe systemer, hvis formål er at fungere som et redskab, har spil og film det tilfælles, at det primære formål med begge medier er at underholde. Den umiddelbare forskel er dog, at hvor seeren kan forholde sig passiv i film, skal brugeren i spil give input til spillet, før dette vil opleves som underholdende. Formålet med dette afsnit er at belyse tilblivelsen af film, for derigennem at definere hvorledes denne proces adskiller sig fra spiludvikling. Min primære forestilling om dette emne er, at der ikke vil være meget tilfælles mellem spil og filmudvikling. Spil udarbejdes over lang tid og skal være underholdende på grund af interaktionen. Film indfanges i 'øjeblikket' og skal underholde ved hjælp af en historie, seeren kan indleve sig i – men uden at kunne påvirke den. Selve processen i tilblivelsen af film kender jeg ikke meget til, så det bliver interessant at lære noget om denne.

FILM

Processen i at lave film består af fire faser: *Script development*, præ-produktion, produktion og post-produktion [Long, 2000] & [Evans, 2006]. Hver fase har sine veldefinerede elementer, der skal være færdige, før den næste fase kan påbegyndes. De fire faser defineres som nedenstående:

- *Script development*: Et manuskript bliver udarbejdet. Manuskriptet indeholder information om hvad skuespillerne siger og gør, hvilke lokationer og *props* der skal anvendes, samt *sets*, grafik og *special effects* der skal udarbejdes til filmen [Long, 2000: 14-16]. Historien i manuskriptet er altafgørende for filmens succes, og den bliver oftest bygget op omkring treakts- eller berettermodellen, der beskriver de væsentligste punkter i historien [Evans, 2006: 100-104]. Reglerne for formateringen og opsætningen af et manuskript er meget veldefinerede, og der findes mange softwareapplikationer, der kan lette arbejdet.

- Præ-produktion: Ud fra manuskriptet påbegyndes arbejdet med at planlægge, visualisere og budgettere produktionen af filmen. I planlægningen er det første trin at beskrive hver enkelt scene, og de dele der skal til for at denne kan filmes. Det vil sige alt fra makeup og mad til køretøjer og lydeffekter. Dernæst skal scenerne indsættes i et *production board*, der angiver den rækkefølge, de enkelte scener skal filmes i, hvor der skal optages og hvilke skuespillere der skal anvendes [Long, 2000: 22-26]. På baggrund af planlægningen kan der nu udarbejdes et budget. Ud fra beskrivelsen af de nødvendige elementer i hver scene er det muligt at udregne omkostningerne for produktionen. I

præ-produktionen bliver filmen visualiseret i form af et *storyboard*, der via tegninger kan vise kameravinkler, *special effects*, placering af mennesker og køretøjer i scenen osv. [Long, 2000: 33-48].

- Produktion: Selve filmningen kan nu påbegyndes. Dagen starter med at instruktøren gennemgår og godkender dagens *production board*/*shooting schedule,* og sættet bliver gjort klart med lys, lyd, kameraer, *props* osv. Hvis der er dukket forhindringer op, såsom sygdom, skal der omrokeres på scenerne således, at der stadig kan filmens. Dagens optagelser kan hermed begynde og en scene er først færdig, når den er godkendt af instruktøren [Evans, 2006: 145-150].

- Post-produktion: I denne fase bliver filmen klippet sammen. Hvis der er *special effects* med i filmen, skal disse indsættes før selve editeringen af filmen kan begynde. Klipperen laver i første omgang et *rough cut*, hvor de bedste scener bliver udvalgt og sammensæt. Dette gøres ud fra en log, hvor de scener instruktøren har godkendt er markeret. Ligeledes anvendes et klaptræ til at identificere en scene og en optagelse. Herefter bliver filmen trimmet, så overgange virker naturlige og flydende. Til sidst bliver lyden tilføjet og filmen er færdig [Evans, 2006: 242-268].

Det både Evans og Long beskriver som det vigtigste i processen, er betydningen af præ-produktionen [Evans, 2006:2]. Det er denne fase, der i høj grad påvirker, hvorledes resten af filmproduktionen kommer til at udfolde sig, og hvor effektiv produktionen bliver. Filmen bliver brudt ned til hver enkelt scene, og hver scene bliver så igen brudt op i de elementer, der skal bruges for at muliggøre optagelsen. På den måde er det nemt at estimere og budgettere en produktion. Afvigelser fra planen kan dog være en enorm udgift. Hvis eksempelvis en skuespiller er syg, og dagens optagelser afhænger af denne skuespiller, går en hel arbejdsdag tabt. Netop fordi filmen bliver indfanget i 'øjeblikket' er det nemt at tage en scene om flere gange efter hinanden. Hvis man derimod først i post-produktionsfasen finder en fejl i en scene, kræver det enorme ressourcer at skulle bygge hele scenen igen, skaffe skuespillere, *props* og så videre. Den måde Long foreslår, at man tager højde for forstyrrende elementer såsom sygdom, er ved at lave et *production board*, der viser hvilke scener, der skal filmes på hvilken dato, samt de skuespillere der skal bruges. Hvis der så dukker sygdom op, skal scenerne omrokeres således, at der ikke går en dags arbejde tabt. Dette er altså med til at give et visuelt overblik og giver et konkret værktøj til at tage højde for de problemer, der højst sandsynligt vil opstå. Dette er interessant i forhold til min 9. semesters opgave, der omhandlede ledelsen af produkter i forandring, da man via et

production board tilbyder en konkret teknik til at tage højde for forandringer. Et tydeligt tegn på at teknikker og metoder til skabelsen af film er veldefinerede og gennemprøvede.

Det overraskende ved beskrivelsen af filmprocessen er den manglende kunstneriske frihed i forhold til indholdet. Jeg troede, at film var en meget kreativ proces, mens det egentligt ser ud som om, at det er en meget fastlåst proces. Når manuskriptet først er skrevet og præ-produktionen fastlagt, kan instruktøren kun ændre ved den enkelte scenes udtryk – alt andet er fastlagt. Kan det være dette, der gør det muligt at overholde et budget? Man planlægger rammerne så grundigt, at det er nemt at estimere og dermed budgettere. Uden at dette dermed har indflydelse på den kunstneriske frihed de kreative personer i produktionen har. Eksempelvis kan instruktøren stadig ændre udtrykket af en scene, lydmanden kan ændre stemningen via lyden, klipperen kan ændre hastigheden på klippene og så videre.

Et andet overraskende element ved filmproduktion er de meget veletablerede retningslinjer. Eksempelvis er der et bestemt og meget veldefineret format, manuskriptet skal skrives i. Ligeledes findes der et hav af værktøjer til at skrive manuskriptet i, planlægge produktionen, lave *storyboards* og så videre. Dette er en tydelig indikator på, at filmproduktion er en meget ældre og mere moden industri end eksempelvis spilindustrien.

En anden detalje, jeg har bemærket i gennemgangen af filmteorien, er, at mange af de samme termer man anvender i produktionen af film, også anvendes i spiludvikling. Eksempelvis i Michael Sellers artikel "The Stages of Game Development" [Sellers, 2003:Chapter 4.1], hvor faserne ligeledes er navngivet præ-produktion, produktion og post-produktion. Den første fase i filmproduktion *Script development* er i spillitteraturen oversat til *concept* eller bare *development* [Sellers, 2003] & [Rollings, 2004]. At man i spiludvikling anvender de samme termer som i filmproduktioner er dog ikke ensbetydende med, at processen er ens. Det kan også skyldes, at man i spilbranchen har manglet termer, og derfor har anvendt nogle fra allerede etablerede brancher. Dette er ikke nødvendigvis en fordel, da anvendelsen af de samme termer kan betyde at processen derved også opfattes som værende ens – hvad den ikke nødvendigvis er.

Ud fra processen til filmproduktion kan der altså opsættes følgende forskelle og ligheder mellem spil og film:

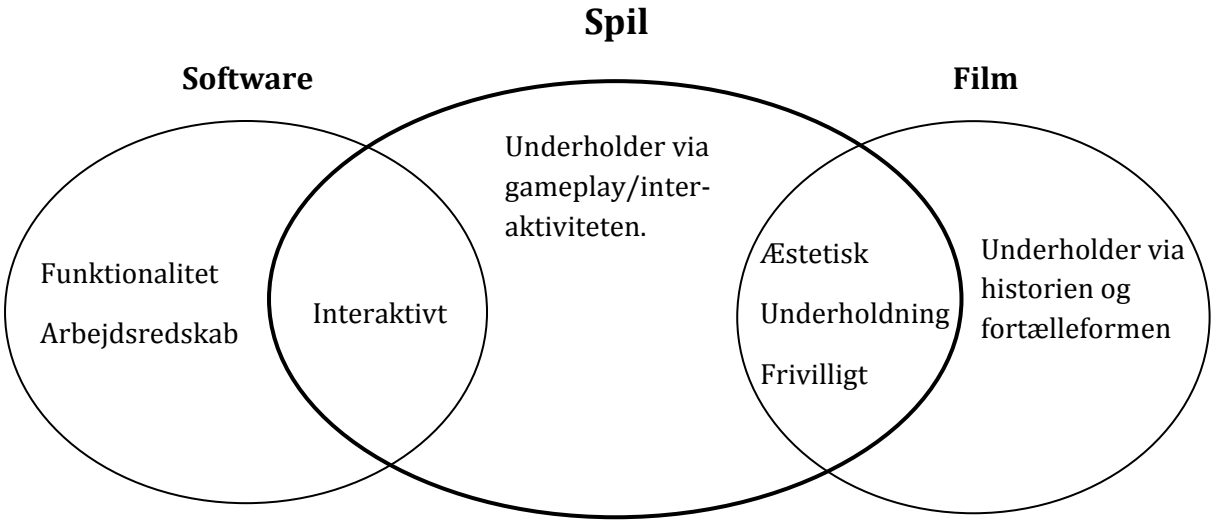
- Både film og spil lægger i høj grad vægt på det kreative.
- Film underholder gennem historien, mens spil underholder gennem *gameplayet*, der også kan indeholde en historie.

- Film er ikke interaktive.
- Både spil og film har et æstetisk udtryk, der adskiller sig fra de æstetiske overvejelser i systemudvikling (æstetikken skal skabe en bestemt stemning, underbygge en historie og så videre, mens den i systemudvikling skal anvendes til at skabe et forståeligt og lettilgængeligt værktøj).

Efter denne gennemgang af film vil jeg i nedenstående afsnit opsummere på forskellen mellem spil, film og IT-systemer.

OPSAMLING

Spil kan altså ikke sidestilles med hverken film eller IT-systemer. Det vigtigste element i spil (jævnfør afsnittet om spil side 18) er interaktiviteten og den anvendes med et helt andet formål end i systemudvikling. Spil er dog stadig i bund og grund kode, der opsætter reglerne i spillet, så det kan ikke udelukkes at systemudviklingen kan finde anvendelse. Det største problem er dog, at det funktionelle rationale skal skiftes ud med et fokus på at gøre spillet underholdende. Ligesom film indeholder spil et æstetisk udtryk, det er noget man går frivilligt ind til, og de skal begge underholde, men her hører ligheden op. Film er ikke interaktive, de er ikke konstrueret på baggrund af kode, og de indeholder ikke noget *gameplay*. Derfor er de to processer meget forskellige. I nedenstående tabel har jeg opsat de mest grundlæggende forskelle mellem software, spil og film:



Figur 2.1: Forskelle og ligheder i spil, film og software (Egen tilvirkning).

I Bilag 2 findes yderligere en tabel, hvor jeg belyser forskellene mellem de tre områder. Umiddelbart er det mit indtryk, at der er mest at hente i softwaretraditionen. Det største problem er dog stadig, at fokusset på det funktionelle skal skiftes ud med et fokus på det underholdende. Før jeg går videre med dette, vil jeg først belyse de problemer, spilbranchen oplever i udviklingsprocessen. Til at gøre dette inddrager jeg et relevant speciale.

FASE 3 – KROGHS PROBLEMATIKKER

En metode skal tilbyde en proces, der tager hånd om de problemer, der opstår under udviklingen. Derfor er det nødvendigt at have kendskab til disse problemer – ellers kan man ikke tage højde for dem. Jeg finder det derfor relevant at inddrage Tea Krogh, der i 2005 skrev speciale om udviklingsprocesserne i spilindustrien og de problemer, der relaterer sig til dette.

DE FIRE PROBLEMATIKKER

Krogh skrev i 2005 specialet med titlen "Udviklingsprocesser i Spilindustrien – kompleksitetsreduktion gennem kommunikation". I dette speciale forsøger hun at synliggøre de problemer, den danske spilindustri oplever i udviklingsprocessen. Gennem tre interviewrunder med seks anonymiserede spilvirksomheder, opstiller hun fire problematikker, der opsummerer de problemer, der af virksomhederne opfattes som de mest kritiske [Krogh, 2005]. Det er derfor sandsynligt, at det er disse problematikker, der ikke tages højde for i udviklingen af spil, og det er derfor interessant at inddrage dem, da der på den måde kan blive taget højde for dem i processen.

Krogh fremsætter også en lang række forslag til, hvorledes det enkelte problem kan løses. Dette vil ikke blive inddraget i dette afsnit, da jeg i første omgang ønsker at få indsigt i problemerne og først senere i processen undersøge, hvordan der så kan tages hånd om problematikkerne i processen.

De 6 virksomheder, der bliver inddraget i undersøgelsen, vil i det efterfølgende blot blive omtalt som 'virksomhederne'.

INTERNE PROBLEMER MED KOORDINERING, SYNKRONISERING OG PLANLÆGNING.

De adspurgte firmaer oplever det som problematisk at synkronisere udviklingsholdets individuelle forståelse og forestilling om det endelige produkt. "Hvis udviklingsholdet internt ikke forstår at kommunikere og videreformidle hvilke behov og forventningerne, der er til et

produkt, risikeres det, at produktet, der udvikles, ikke lever op til forventningerne og herunder eventuelt den indgåede kontrakt" [Krogh, 2005: 28]. Kroghs undersøgelse peger på, at virksomhederne ikke har nogen retningslinjer eller processer, der sikrer, at information deles med projektets medlemmer. Konsekvensen af dette er, at virksomhederne oplever, at deres medarbejdere ikke har føling med, hvad kunden gerne vil have, og dermed ikke hvad de skal fokusere på i deres arbejde.

Planlægningen af udviklingen opleves ligeledes problematisk. Specielt giver virksomhederne udtryk for at have problemer med at gå i gang med store projekter uden at have defineret klart og tydeligt, hvad produktet skal indebære. Virksomhederne oplever, at der bliver et stadigt større behov for at planlægge projektforløbet. Specielt med henblik på at synliggøre sammenhængen mellem delelementer, således man bliver i stand til at udvælge hvilke dele, der skal realiseres før et andet er muligt.

FORVENTNINGSSTYRINGSPROBLEMATIKKER

Virksomhederne oplever kommunikationsproblemer med eksterne partnere i henhold til produktet, licensbetingelser og distribution [Krogh, 2005: 35-40]. Misforståelserne skyldes ifølge virksomhederne primært at kunden ikke har viden om udviklingsprocessen, arbejdsopgaven, det pågældende medie samt dennes begrænsninger og muligheder. Dette kommer blandt andet til udtryk ved kunder, der kræver ændringer i spillet i sidste øjeblik, eller forslag til ændringer der påvirker spillets *gameplay* på en uhensigtsmæssig måde. Ligeledes oplever de problemer med kunder, der ikke mener de har fået det aftalte ved slutleveringen. På baggrund af dette understreger virksomhederne vigtigheden af løbende kontakt til de eksterne partnere samt at interessenternes krav bliver kommunikeret klart og tydeligt under kontraktforhandlingerne.

ESTIMERINGSPROBLEMATIKKER

Dette emne er nært beslægtet med forventningsstyringsproblematikkerne. Idet virksomhederne føler, at det er svært at opnå en fælles forståelse for produktet imellem virksomheden og kunden, medfører dette også, at estimeringen af projektet vanskeliggøres [Krogh, 2005: 41]. Konsekvensen af dette er, at budgettet nemt bliver overskredet, fordi der bliver lovet mere, end udviklerne kan nå indenfor den givne tidsramme. Ligeledes opleves det som problematisk at holde styr på, hvad der bliver lovet en kunde og til hvilken pris.

TESTNINGSPROBLEMATIKKER

Virksomhederne oplever problemer med at planlægge, udføre og koordinere testarbejdet [Krogh, 2005: 43]. Internt i virksomheden er der problemer med at holde styr på, hvad der skal rettes, og hvad det skal rettes til. Eksempelvis retter nogle udviklere, hvad de tror, der er en fejl, uden at det nødvendigvis behøver at være det. Eksternt kan det være kunden, der via deres tests kommer med rettelser til spillet, der påvirker spillets *gameplay* på en uhensigtsmæssig måde. Problemet med at få feedback på *gameplayet* af brugerne, opleves ligeledes problemfyldt, idet disse ofte kommer med forslag, der vil give dem en fordel i spillet, og ikke fordi det vil gøre spillet mere balanceret. Der opstår altså en række problemer med at få spillet testet på en sådan måde, at det er uden fejl og velbalanceret.

OPSAMLING

Kernen i Tea Kroghs speciale ser ud til at være, at der er for dårlig kommunikation om produktet og forventningerne til dette både internt og eksternt i de adspurgte virksomheder. Som en følge af dette bliver det svært at planlægge, at estimere arbejdsopgaver og at teste – da man ikke ved, hvordan det er meningen, at spillet skal opføre sig. Mere kommunikation er dog ikke nødvendigvis løsningen. Løsningen kunne være en proces, der sørger for at de rette personer har de informationer, der skal til for at de kan udføre deres arbejde. Altså er spørgsmålet om løsningen ikke i højere grad er en omlægning af arbejdsgange og en ny måde at tænke og arbejde på.

Om ikke andet synes jeg, at Kroghs problematikker peger i retningen af, at der er behov for flere rammer for processen. De problemer, virksomhederne oplever, kunne med stor sandsynlighed skyldes, at den proces der følges ikke er veldefineret og måske faktisk defineres ad hoc.

Men hvad findes der egentligt af litteratur, der kan hjælpe spilvirksomheder med denne type af problemer? Jeg har behov for at vide mere om de processer, der foreslås i den tilgængelige spillitteratur, før jeg kan arbejde videre på udviklingen af en metode.

FASE 4 - GENNEMGANG AF EKSISTERENDE SPILLITTERATUR.

For at bidrage til udviklingen af spilmetodikker, er det nødvendigt at kigge nærmere på de allerede etablerede forfattere. I dette afsnit ønsker jeg at belyse den eksisterende spillitteratur for at se nærmere på, hvordan denne opfatter spiludviklingsprocessen. Jeg forventer, da mange af forfatterne er praktikere, at deres konklusioner således baserer sig på deres oplevelser og ikke nødvendigvis er teoretisk funderet eller reflekteret. Men da spilforskningsfeltet stadig er ungt, er det et godt sted at starte med de folk, der har praktisk erfaring. Ligeledes, igen på grund af spilforskningens alder, tror jeg, at de forskellige forfattere vil have meget forskellige syn på processen. Der vil højst sandsynligt være nogle fællestræk, men de vil have vidt forskellige baggrunde og dermed også vidt forskellige tilgange til at skrive om udviklingen af spil. Jeg håber at kunne finde de procesmæssige forskelle og ligheder forfatterne imellem.

For at gøre dette, vil jeg analysere hvert litterært værk ud fra følgende model:

- Baggrund: En beskrivelse af forfatterens uddannelsesmæssige baggrund, hvilken rolle vedkommende har haft i spilindustrien samt anden relevant information vedrørende forfatteren. Formålet med afsnittet er at synliggøre forfatterens person, da dette unægtelig må have stor indflydelse på det syn, vedkommende betragter genstandsfeltet med. Ligeledes passer det ind i den hermeneutiske tankegang, hvor det vil give mig en mulighed for at få et indblik i de fordomme, personen ud fra sin baggrund kan have om selve processen, og dermed på en mere objektiv måde diskutere hvorvidt synspunkterne er relevante eller en direkte afspejling af forfatterens referenceramme.

- Procesmodel: Efter beskrivelsen af forfatteren ønsker jeg at se nærmere på, hvorledes denne betragter spiludvikling som proces. Det vil sige en beskrivelse af den proces forfatteren opstiller til udviklingen af spil. Det primære fokus vil, ud over selve processen, være en beskrivelse af, hvordan forfatteren gennem processen gør det muligt af overholde *deadlines* og skabe en realistisk plan for projektet.

- Syn på processen: Som det sidste ønsker jeg at belyse forfatterens syn på processen. I en beskrivelse af hvordan forfatteren ser udviklingsprocessen, er det sandsynligt, at der også vil fremstå et syn på, hvad processen i forfatterens øjne minder om, og hvad forfatteren finder vigtigt ved udviklingen af spil. Det vil eksempelvis være en beskrivelse af de kerneelementer, der indgår i forfatterens proces, og som derfor fortæller noget om forfatterens syn på spil og udviklingen af disse. Det er specielt i dette afsnit at kendskabet til forfatterens baggrund træder i karakter og muliggør en beskrivelse af, hvorfor forfatteren betragter processen på en bestemt måde. Formålet med dette afsnit er at få kendskab til den baggrund, hvorved den opsatte procesbeskrivelse er blevet til og for at muliggøre en diskussion af de forskellige syn på processen.

I behandlingen af hver enkelt forfatter, vil jeg ligeledes diskutere, hvordan deres idéer og tanker stemmer overens eller adskiller sig fra de øvrige.

De seks beskrevne forfattere er:

- Erik Bethke
- Chris Crawford
- Tracy Fullerton
- Bob Bates
- Richard Rouse
- Dan Irish

De seks forfattere er valgt ud fra, at deres bøger alle er udbredt spillitteratur og enkelte af bøgerne (Fullerton, Richard Rouse og Dan Irish) er brugt i undervisningen – enten ved projektledelses- eller spildesignkurserne. Ligeledes håber jeg, ved at inddrage seks forfattere, at få et bredt billede af, hvordan spiludvikling betragtes.

ERIK BETHKE: GAME DEVELOPMENT AND PRODUCTION

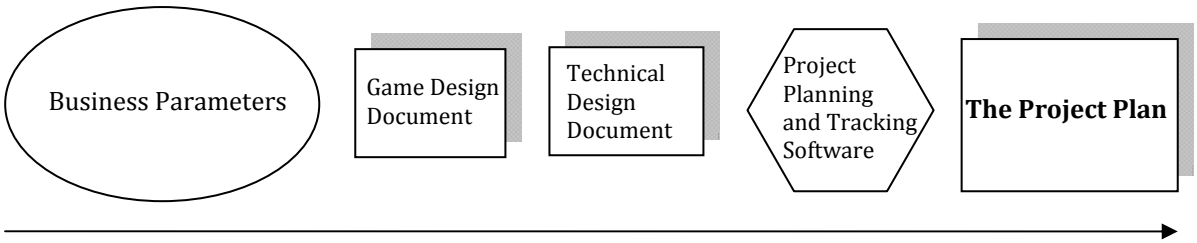
BAGGRUND

Erik Bethke er CEO ved det amerikanske spilfirma Taldren, der blandt andet udvikler Star Trek spil. Bethke fungerer som producer og *lead designer* på førnævnte spil. Han er uddannet i

Aerospace Engineering så hans baggrund må betegnes som værende af teknisk karakter. Bogen "*Game Development and Production*" er baseret på Bethkes erfaring fra spilproduktion samt omfattende interviews med andre producere fra spilindustrien [Bethke, 2003: Preface].

PROCESMODEL

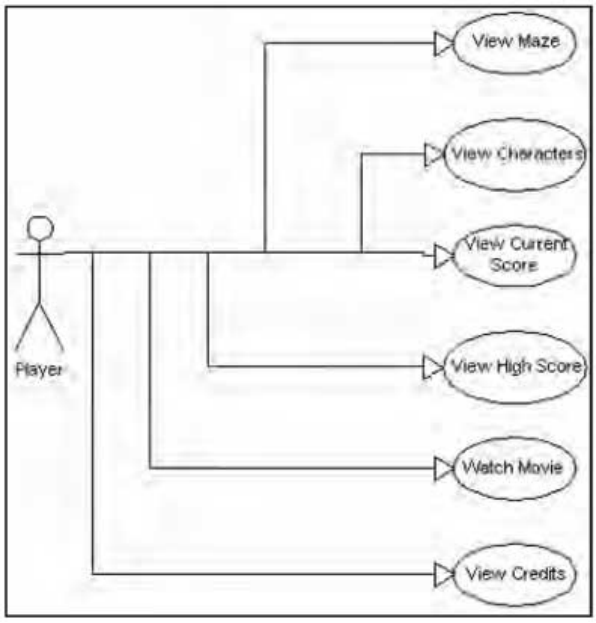
Bethke mener, at grundlaget for spiludviklingsprocessen er en velbeskrevet og gennemarbejdet projektplan [Bethke, 2003: 15]. De elementer, der skal til for at udarbejde projektplanen, er opstillet visuelt i nedenstående model, hvor pilen repræsenterer tiden:



Figur 4.1: The projekt plan pipeline [Bethke, 2003: 158]

For at projektplanen skal kunne udarbejdes, skal de forretningsmæssige parametre være fastlagt, og der skal forelægge både et *game*-designdokument og et teknisk designdokument. Metoden til at udarbejde projektplanen starter med, at man nedbryder de to designdokumenter. Til at nedbryde disse foreslår han en tilgang, der er meget identisk med systemudviklingsmetoden "Objekt Orienteret Analyse og Design". *Gameplayet* bliver beskrevet via *use cases*, der beskriver en handling spilleren kan foretage sig i spillet – eksempelvis '*open door*'. Når hele spillet er nedbrudt til funktioner, kan disse opsættes i UML diagrammer, der afbilleder en brugssituation, og de funktioner der tilknytter sig til denne. Et eksempel på en brugssituation kan være *Display System*, hvortil der så tilknyttes de relevante funktioner såsom *View Score*, *View Map*, *View Characters* og så videre [Bethke, 2003: 82-86]. Dette er afbilledet i figur 4.2 på modstående side:

Display System
Display maze
Display characters and their animation
Display score
Display high score
Display credits
View movie (Ms. Pac-Man)



Figur 4.2: Eksempel på brugssituationen *Display System*, og de funktioner der knytter sig til denne [Bethke, 2003: 82-86].

Når projektplanen skal skabes, kan man tage opgaverne direkte ud af diagrammerne og opsætte dem i et Gantt eller PERT diagram. Disse diagrammer opstiller arbejdsopgaverne og deres tilknytning til hinanden visuelt ved hjælp af en tidslinje. *Milestones* til projektplanen skal enten hentes ved ledelsen, der nedsætter rammerne for projektet, eller ved at gå tilbage fra afleveringsdatoen for derved at fastsætte, hvornår de enkelte dele af spillet nødvendigvis skal være færdige [Bethke, 2003: 158]. Hvordan grafikken bliver udarbejdet og planlagt er kun beskrevet meget overfladisk. Det er dog formuleret at *art* teamet skal udarbejde en række *sketches* i udviklingen, der kan hjælpe til at forstå spillets *look and feel* [Bethke, 2003: 224].

Den proces, Bethke foreslår, er altså meget lineær – ud fra et designdokument nedbrydes spillet til alle de enkelt dele, det skal bestå af, hvorefter disse placeres i en projektplan. Processen minder dermed meget om tankegangen bag de traditionelle systemudviklingsmodeller som eksempelvis vandfaldsmodellen, hvor man går igennem en række planlagte faser for at nå et planlagt og velbeskrevet mål.

SYN PÅ PROCESSEN

Bethkes syn på udviklingen er tydeligt – spil er systemudvikling. Dette er ikke noget, han holder hemmeligt, idet han på side fire benytter overskriften "*Game Development is Software Development*" [Bethke, 2003: 4]. I dette afsnit står der endvidere: "*Games are software with art, audio and gameplay*". De metoder han foreslår, er også direkte hentet ud fra softwaretraditionen

med både UML diagrammer, klasse diagrammer, *use cases* og så videre. Et problem, der dog er gemt godt af vejen i bogen, er nedenstående citat:

> *"Creating a finished design document is so difficult I have never been able to finish one of my own, nor have I seen anyone else finish his or her design documents" [Bethke, 2003: 101]. "In the real world you will have many competing time pressures. Instead of carrying everything out to full completion, you must use your judgment and determine what areas of the design require the most game design resources"* [Bethke, 2003: 224]

Problemet ved ovenstående er, at han, i sin model over processen, baserer projektplanen på designdokumentet, som han selv end ikke er i stand til at færdiggøre. Dette tager han ikke på noget tidspunkt højde for.

> *"Games are art forms realized in a piece of engineering brought forth by team effort. There is no way anyone has ever written down at the start of the project every detail about a game without changing his mind on the way to making a great game"* [Bethke, 2003: 224].

Igen skriver Bethke altså, at der ikke på noget tidspunkt er sket det, at nogen har nedskrevet hver eneste detalje om spillet fra starten uden at ændre noget undervejs. Problemet med dette er, at Bethke ikke forholder sig yderligere til, hvordan dette påvirker den proces, han foreslår og beskriver. Han foreslår dog, at man bryder designdokumentet ned i flere små dele, da det så er nemmere at uddelegere dele af designdokumentet til en gruppe af teamet. Eventuelt kan man anvende versionsstyringsværktøjer til at holde styr på designdokumentet [Bethke, 2003: 218]. Problemet ved dette er, at Bethke på denne måde foreslår en teknik til at gøre det nemt at lave ændringer i designdokumentet. Han beskriver ikke, hvorledes dette påvirker planen og hvordan man tager højde for, at et designdokument som regel er ufærdigt, når man starter produktionen af spillet. Endvidere nævner han jo selv, at han endnu ikke selv har prøvet at færdiggøre et designdokument, hvilket blot får hans procesmodel til at fremstå endnu mindre gennemtænkt.

CHRIS CRAWFORD: ON GAME DESIGN

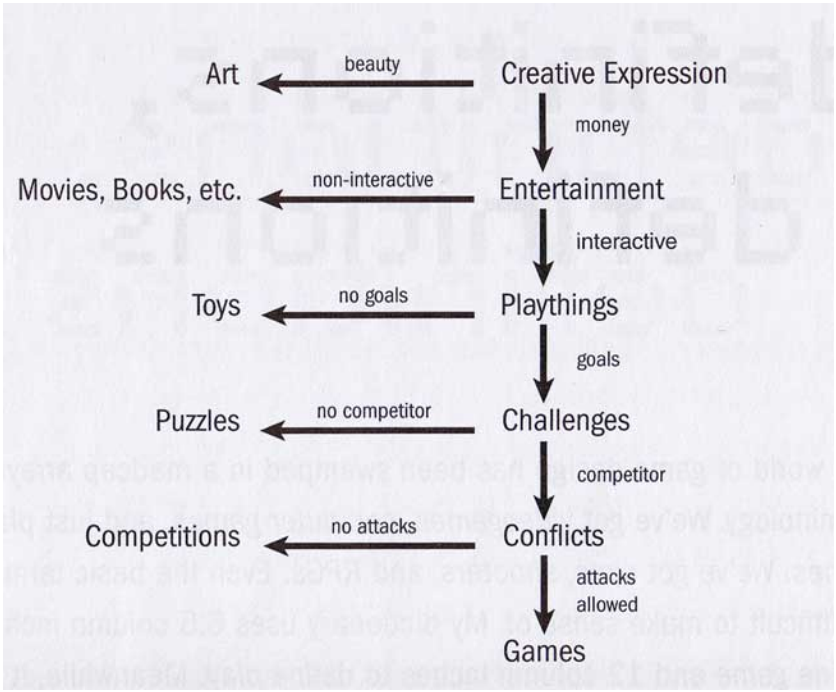
BAGGRUND

Chris Crawford har en uddannelse i fysik, men gjorde i 1979 sin hobby til sin levevej, da han blev ansat til at lede Atari's spilforskningsafdeling. Han har udgivet fjorten spil, der både er designet

og kodet af ham selv. I de senere år, har Crawford primært arbejdet med interaktive historier og udviklingen af værktøjer til produktion af disse spil.

PROCESMODEL

Da Crawfords bog mere handler om spildesign end processen bag udviklingen opstiller han ikke en model over processen. Han opsætter dog i starten af bogen, i et forsøg på at definere hvad spil er og ikke er, en model, der afbilleder dette forhold [Crawford, 2003: 6]:



Figur 4.3: Crawfords figur over de elementer et spil indeholder [Crawford, 2003: 6]

Det interessante ved modellen er, at udgangspunktet er *Creative Expression* lige meget om der er tale om film eller spil. Dette stemmer godt overens med resten af bogen, hvor det er de kreative processer, der er fokus på. Eksempelvis har Crawford et afsnit i bogen med overskriften: "Creativity: The Missing Ingredient" [Crawford, 2003: 93]. Her skriver han blandt andet om den manglede kreativitet i spilindustrien, og i afsnittet *How to Get Creative* beskriver han, hvordan man kan tænke kreativt. At Crawford i sin bog primært skriver om spildesign betyder dog ikke, at bogen er helt blottet for procesbeskrivelse. Crawfords udgangspunkt for spildesignet er interaktionen. Et tema kan anvendes til at guide spildesigneren gennem konceptualiserings-processen, men det er ikke en nødvendighed. Interaktionen skal så efterfølgende beskrives ved hjælp af et *storyboard,* der billede for billede viser, hvordan spillerens input påvirker spillet

[Crawford, 2003: 118-119]. Hvordan man planlægger produktionen og sikrer, at deadlines bliver overholdt, er ikke noget Crawford beskæftiger sig med i denne bog.

SYN PÅ PROCESSEN

Crawford fremsætter i sin bog nedenstående synspunkt:

> *"The real problem lies in the heart and soul of the industry. The games industry is a bastion of techie-geek triumphalism; it has no truck with those soft headed artsie-fartsie people. In their hearts, games people believe that any problem can be solved with sufficient attention to technical detail."* [Crawford, 2003: 164]

I mine øjne giver ovenstående en forklaring på hvorfor folk som eksempelvis Bethke, ser spiludvikling som ren systemudvikling. Det mest grundliggende i kreationen, det vil sige selve tilblivelsen af spil, er selve koden, og det har derfor kun været muligt at lave spil for folk, der allerede havde et kendskab til dette felt. Dette må have påvirket den måde spil er tænkt og udviklet på helt fra begyndelsen. Eksempelvis sætter dette ifølge Crawford stadig sine spor i de mange nyoprettede spiluddannelser:

> *"Academic institutions in the USA are still hobbled by a certain amount of "two cultures" antagonisms – the science and engineering side, and the arts and humanities side just don't understand each other. These difficulties hurt both sides, when it comes to game design."* [Crawford, 2003: 126]

Dette relaterer sig meget godt til Rosenstands model (jævnfør figur 2.1), der viser forskellen mellem det æstetiske udtryk og det funktionelle udtryk. Spørgsmålet, der så rejser sig i denne sammenhæng, er hvilken indflydelse dette har på processen? Hvordan påvirker det processen, at der er forskellige optikker, og at folk har forskellige uddannelsesmæssige baggrunde? Ifølge Crawford har det en negativ påvirkning på selve spildesignet, men dette forklarer ikke betydningen i forhold til processen.

Crawford beskriver i afsnittet *Hollywood Envy*, hvorledes spilfolk har betragtet Hollywood som deres nirvana. Spilfolk elsker ifølge Crawford at anvende filmterminologi i diskussion om spildesign, uden at dette tilføjer noget til selve designprocessen [Crawford, 2003: 182]. Han fortsætter diskussionen med følgende:

> *"The word 'cinematic' seems to be more common in game design discussions than 'interactivity', even though the latter is central to game design and the former is peripheral."* [Crawford, 2003: 82]

Gennem dette giver Crawford altså udtryk for, at interaktiviteten i hans øjne er en central del af spildesignet, og at anvendelsen af termer fra filmindustrien ikke nødvendigvis bidrager med noget i designprocessen. Det fremstår altså tydeligt at Crawfords syn på processen er, at spildesignet skal være det bærende element, og at det vigtigste område for spildesignet er selve interaktionen. Ligeledes efterlyser han, som også nævnt tidligere, kreativitet i en branche, der er præget af, at man blot tager et gammelt spil og tilføjer et par nye detaljer [Crawford, 2003: 93-98].

Crawford diskuterer også i sin bog vigtigheden af de forskellige elementer i spil, og der er ingen tvivl om, hvad Crawford finder vigtigst. Hans eksempler bygger blandt andet på de to film Shrek og Final Fantasy. Begge er animationsfilm, dog med to vidt forskellige prioriteter. I Shrek har man brugt den kreative energi på historien, og måden den bliver fortalt på i modsætning til Final Fantasy, hvor man har brugt energien på det grafiske udtryk, og filmen er kosmetisk den flotteste af de to. Men Shrek solgte ti gange flere billetter end Final Fantasy. Forklaringen på dette ligger ifølge Crawford i, at historien er det vigtigste for film og ikke nødvendigvis den grafiske repræsentation [Crawford, 2003: 109-110]. Formålet med dette eksempel er at fortælle, at grafikken og det grafiske udtryk ikke nødvendigvis er det vigtigste for et spil. Det skal stemme overens med spillets opsætning af konflikter. Eksempelvis skal ekstrem realistisk grafik ifølge Crawford kun anvendes i "*legal dramas, corporate politics, and the like*" [Crawford, 2003: 112]. Crawford skriver også om *Obsession with Cosmetics*:

> *"To further the gameplay by making the player's situation and options as clear as possible. This is the only good reason for pursuing cosmetics"* [Crawford, 2003: 107].

Ifølge Crawford er den eneste gyldige motivation til at udstyre et spil med god lyd og grafik, hvis det synliggør spillerens situation og muligheder. Men andre ord; hvis det understøtter spillets *gameplay*.

Ligeledes er det interessant at se, hvordan Crawford afgrænser spil i forhold til systemudvikling og film. Systemudvikling bliver slet ikke nævnt, mens det spil og film har til fælles er, at de begge er kunstneriske udtryk og et forsøg på at tjene penge (jævnfør Figur 4.3). Betyder dette, at processen i at udvikle spil ikke har noget til fælles med film- og systemudvikling?

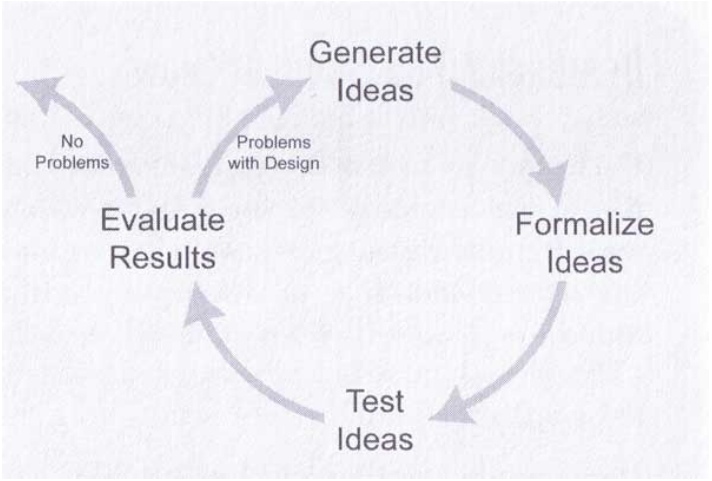
TRACY FULLERTON ET AL.: GAME DESIGN WORKSHOP

BAGGRUND

Tracy Fullerton er uddannet MFA (*Master of Fine Arts*) ved *University of Southern California* (USC), hvor hun primært har beskæftiget sig med film og Tv-produktion [Fullerton, 2007]. MFA uddannelsen fokuserer ofte på studier indenfor teater, filmproduktion, visuel kunst og kreativ skrivning [USC, 2007]. Hendes bachelor er taget i *Theater Arts* og *English Literature* [Fullerton, 2007]. Fullerton arbejder på nuværende tidspunkt som assisterende professor ved USC's afdeling for interaktive medier, hvor hun fungerer som *Co-Director* i Electronic Arts spil innovations afdeling. Tidligere har hun arbejdet ved *Spiderdance*, der laver spil til fjernsynsmediet, og hun har siden 2002 fungeret som *freelance game designer* og konsulent [Fullerton, 2007].

PROCESMODEL

Fullertons tilgang til spiludviklingsprocessen er baseret på et ønske om at realisere en spilbar udgave så hurtigt som muligt – hvad enten det er i form af et rollespil eller en papirprototype. Formålet med dette er at spille og forbedre en simplistisk model af spillet, før et udviklingshold bliver sammensat [Fullerton et al., 2004: 10]. Ligeledes skal prototyper gøre det muligt at inddrage målgruppen så hurtigt som muligt i udviklingen, så man kan få dennes feedback. Årsagen til denne tilgang skyldes ifølge Fullerton, at spiludviklerne på nuværende tidspunkt designer til *publisherne* og ikke til målgruppen/publikummet. Hendes løsning på dette er at fastlægge de vigtigste elementer af spillet tidligt i produktionen ved hjælp af *prototyping* og en iterativ proces [Fullerton et al., 2004: 11]. Hun afbilleder designprocessen i nedenstående figur:

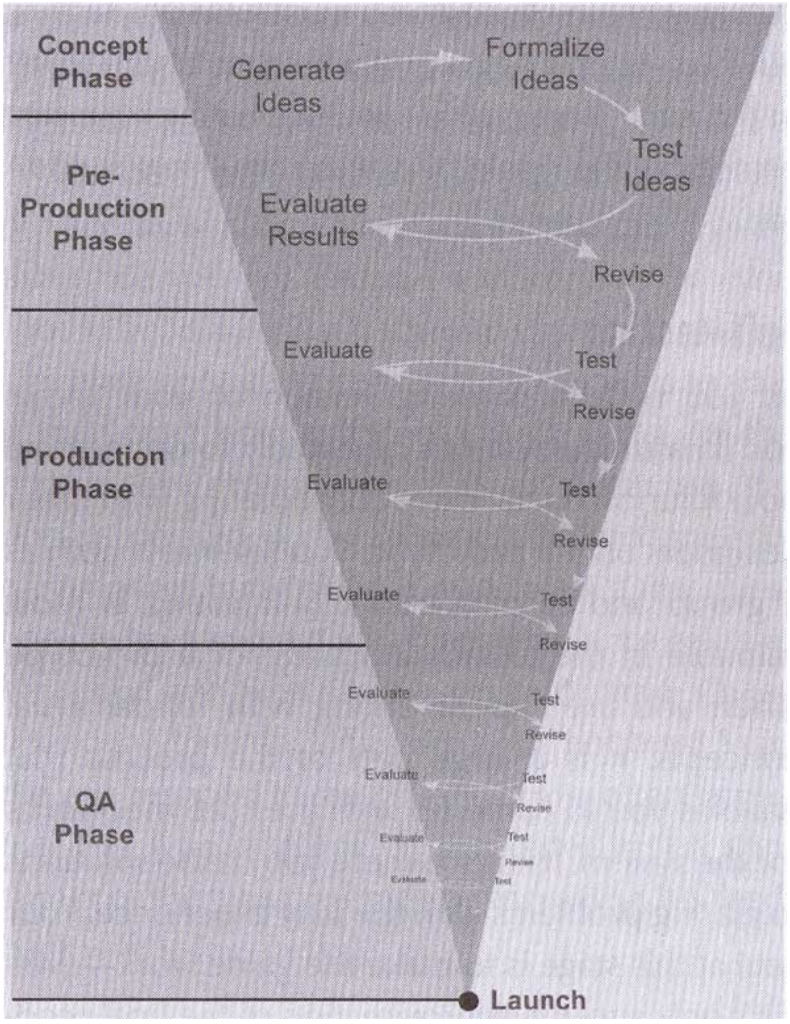


Figur 4.4: Ideen bag Fullertons procesmodel [Fullerton et al., 2004: 11].

Tankegangen er, at den iterative proces løber igennem nedenstående faser:

- Idé bliver genereret.
- Idéen bliver formaliseret. Det vil sige nedskrevet eller formaliseret i en prototype.
- Idéen bliver testet gennem spiltest eller der bliver opsamlet *feedback*.
- Resultaterne bliver evalueret, kategoriseret og prioriteret.
- Hvis resultaterne er negative og idéen dermed er fejlbehæftet, skal der startes forfra på forløbet, og en ny idé skal genereres.
- Hvis resultaterne peger på, at der er behov for forbedringer, så skal prototypen ændres og testes igen.
- Hvis resultaterne er positive og ideen virker med succes, er den iterative proces gennemført.

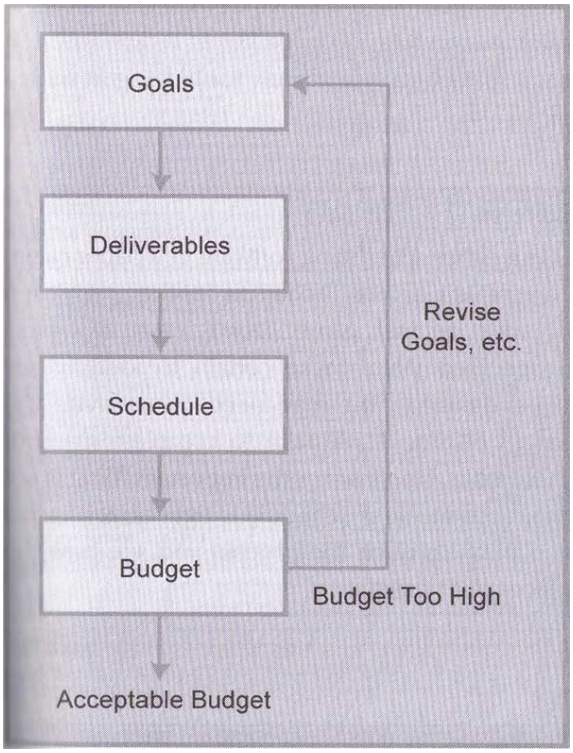
Sat ind i en samlet model over forløbet kan det repræsenteres således:



Figur 4.5: Fullertons iterative tilgang og dens relation til processen [Fullerton et al., 2004: 197].

Det mørkegrå 'V' formet stykke skal vise, at de kreative muligheder gennem processen bliver mere og mere begrænset. Fullertons tilgang er nærmest identisk med Boehms evolutionssoftware proces, der er bedre kendt som Spiral Modellen [Pressman, 2000: 35-40]. Denne model er kendetegnet ved, at man har en lineær proces, der bliver indført i *prototypings* iterative tilgang, og resultatet er dermed en proces, hvor systemet hele tiden forbedres gennem en række fastlagte faser, der gentages indtil projektet kan afsluttes.

Fullerton beskriver ligeledes processen i at udarbejde et budget. Denne tilgang er illustreret i nedenstående model:



Figur 4.6: Fullertons forslag til udarbejdelsen af et budget [Fullerton et al., 2004: 359].

Først fastsættes spillets mål – såvel *gameplay*-mæssige som tekniske. Ud fra disse mål formuleres en række *deliverables*, der indeholder en beskrivelse af de elementer spillet skal indeholde. Disse skal placeres ud fra det stadie i udviklingen, hvor de skal produceres. Eksempler er 3D modeller, lyd, animation af figurer, musik og så videre. Dernæst skal hver *deliverable* nedbrydes i de delopgaver denne består af, og disse skal så sættes ind i en plan. Delopgaverne skal tids- og ressourceestimeres, og placeres i eksempelvis et Gantt diagram [Fullerton et al., 2004: 358-360]. Ud fra dette kan der nu fastsættes et budget. Hvis budgettet er for højt, skal processen starte forfra, indtil man sidder med et tal, der matcher *publisherens* forventninger [Fullerton et al., 2004: 362].

Det interessante i denne sammenhæng er, at Fullerton ikke forklarer, hvordan denne opsætning af budgettet skal fungere sammen med den iterative designproces. Her opstår, i mine øjne, et modstridende forhold: Designprocessen er iterativ, men budgetplanlægningen forudsætter et fastlagt projekt med veldefinerede mål. Dette ligner præcist det samme problem, som Bethke sidder med: Hvordan planlægger man et projekt, der ændrer sig undervejs i processen? Både Bethke og Fullerton bemærker, at spildesign er en løbende proces. Bethke ved at foreslå en opsætning af designdokumentet, der gør det nemt at lave ændringer og holde det opdateret, mens Fullerton foreslår en iterativ designtilgang, så spillet hele tiden kan forbedres. Ingen af dem tilbyder dog en planlægning, der tager højde for disse ændringer, så deadlines og budget kan overholdes.

SYN PÅ PROCESSEN

Fullertons procesmodel giver, ifølge hende selv, én vigtig fordel:

> *"The primary advantage of the iterative design process we recommend, besides saving time and money, is that is puts the player at the center of the game design process. This seems logical, but all too often the player comes last, and the production process winds up dictating changes in the game"* [Fullerton et al., 2004: 20].

Fullertons syn på spil minder dermed meget om Crawfords – det er spildesignet, *gameplayet* og selve interaktiviteten, der er vigtig for processen. Dermed adskiller de to sig fra Bethke, der ser processen som systemudvikling. Fullertons model og syn adskiller sig dog fra de to andre, idet processen bliver bygget op omkring selve spilleren. Crawford prioriterer også designet højt, men nævner ikke elementer såsom spiltest, *feedback* fra målgruppen og så videre.

Fullertons tilgang til spiludvikling kan meget vel være præget af hendes baggrund. Hvor eksempelvis Bethke kommer med en ingeniør og teknisk baggrund, er Fullerton fra film og kunstverdenen. Dette kan forklare, hvorfor hun netop fokuserer så meget på selve spilleren – det er dennes oplevelse, der er vigtig i designprocessen af spil, ligesom det er tilfældet i eksempelvis teater og film.

Den kreative designproces Fullerton foreslår, bliver ikke fulgt af en planlægningsproces, der indretter sig efter den iterative struktur. Designdokumentet vil ændre sig, og derved vil såvel budget som tidsestimater ændre sig. Dette tager Fullerton bare ikke højde for. Så hvordan sikrer man, at et budget bliver overholdt i en produktion, hvor kravene uundgåeligt vil ændre sig undervejs?

BOB BATES: GAMES DESIGN - THE ART & BUSINESS OF CREATING GAMES

BAGGRUND

Bates uddannelsesmæssige baggrund består af filosofi og psykologi ved *Georgetown University* i Washington. I 1986 startede han firmaet *Challenge Inc.*, der udgav flere tekst-*adventure* spil. Sideløbende med dette var han i en årrække ansat som designer ved firmaet Infocom, og han nåede at udgive to spil inden firmaet blev lukket i 1989. Herefter var Bates ansat ved forskellige firmaer enten på grund af opkøb eller lukninger [Infocom, 2007]. På tidspunktet hvor *Games Design - The Art & Business of Creating Games* er skrevet, var Bates ansat som *studio head* ved Legend Entertainment [Bates, 2001: About the Author]. På nuværende tidspunkt arbejder han som selvstændig designer, forfatter og producer [IGDA, 2007].

PROCESMODEL

Bates foreslår to mulige tilgange til udviklingen af spil. Den første tilgang er vandfaldsmetoden:

> *"In a perfect world, software development would follow the classic waterfall model. Requirements would be completely defined. Then the system architecture would be designed, followed by detailed design, coding and debugging, and finally, testing"* [Bates, 2001: 228].

Bates skriver dog, at spiludvikling ofte er mere kaotisk end som så, og at vandfaldsmetoden derfor ofte kun kan finde anvendelse, når det er det næste spil i en serie, der skal produceres. Det vil sige, at det kun fungerer i projekter, hvor det er muligt at fastlægge store dele af spillet, og hvor forandringer undervejs ikke er sandsynlige.

Den anden løsning er iterativ *prototyping*:

> *"Rapid iterative prototyping is the best development model for most new games"* [Bates, 2001: 229]

Årsagen til dette ligger ifølge Bates i, at man således hurtigt kan fremstille en rå prototype, og ud fra denne definere hvad der fungerer, og hvad der ikke fungerer. Nøglen til denne model er ifølge Bates at redefinere spildesignet kontinuerligt, baseret på den prototype man er kommet frem til. Hver prototype skal så have sin egen planlagte cyklus med komplet udvikling, beskrevne krav, en

plan for aflevering af kravene og en tidsplan, der beskriver hvordan det skal nås [Bates, 2001: 229].

Bates beskriver også muligheden for at anvende forskellige modeller på forskellige dele af udviklingsprocessen. Eksempelvis kan vandfaldsmodellen anvendes til skabelsen af *assets*. Monstre, våben og baner kan alle designes, udspecificeres og bygges i en velordnet proces. Selve *gameplay* delen af udviklingen læner sig op af en mere iterativ proces, hvor eksempelvis den kunstige intelligent, monstre og den måde et våben skal fungere på kan designes, kodes, testes og redefineres over en række af prototyper [Bates, 2001: 230]. Specielt det sidste er interessant i henhold til de øvrige forfattere. Det er dog ikke noget Bates behandler yderligere udover en lille spalte, hvor han fremsætter forslaget om at bruge flere procesmodeller i udviklingen. Bethke foreslår, som beskrevet, en metode, der minder meget om vandfaldsmodellen, mens Fullertons tilgang nærmest er identisk med spiralmodellen. Hvad nu hvis man kombinerede de to modeller til de forskellige dele af udviklingen? Således at spildesignet bliver afprøvet, testet og redesignet gennem en iterativ proces. Når et element som for eksempel et våben så har den rette *gameplay*-mæssige funktion, kan grafik- og lydholdet begynde at designe den æstetiske udformning af våbnet. Dette løser dog ikke problemet med planlægningen, da forløbet på den måde vil tage udgangspunkt i et iterativt forløb. Men et sådan må vel også være muligt at estimere?

For at styre et softwareprojekt succesfuldt skal man ifølge Bates [Bates, 2001: 221]:

1. Starte med et komplet design.
2. Udvikle en fornuftig tidsplan samt en plan for den tekniske udførsel.
3. Forstå problemerne der kan opstå og forsøge at forhindre/undgå dem.
4. Vide hvordan man genopretter projektet, hvis det kommer i problemer.

Ligeledes skriver Bates, som en kommentar til de store spilprojekter, der ofte tager to til tre år at færdiggøre, hvorledes man skal håndtere dette:

> *"The trick to surviving this lonk strech is to break large tasks into small, manageable tasks that are rigorously tracked. You can't know whether you're behind on a project if you don't track the tasks"* [Bates, 2001: 214].

Hver udvikler skal derfor have sin egen liste over arbejdsopgaver og deres tidsestimat. Dette er ifølge Bates den mest succesfulde måde at håndtere kompleksiteten i de store projekter. Ligeledes giver det udviklerne mulighed for at sætte deres egne tidsestimater, hvilket ifølge

Bates øger deres tilknytning til tidsplanen, og de vil derfor være mere tilbøjelige til at overholde deadlines [Bates, 2001: 214].

SYN PÅ PROCESSEN

Ifølge Bates starter ethvert spil som én enkelt idé. Denne idé kan så omhandle en karakter, et *gameplay*, en ny teknologi eller lignende [Bates, 2001: 4]. Ligeledes er det ifølge Bates vigtigt, hvis man ønsker at lave et godt spil, at man har evnen til at sætte sig i spillerens sted og forudse dennes reaktioner ved hvert element i spillet [Bates, 2001: 22]. Det modstrider dog på visse punkter med Fullertons tilgang. Fullerton ønsker at inddrage spillerne så hurtigt og så meget som muligt i processen, mens Bates som beskrevet forudsætter at spildesigneren kan sætte sig i spillerens sted og forudsige deres reaktioner. Bates nævner dog også, at da spildesigneren ikke kan forudsige alt, kan testere anvendes.

Selvom Bates altså skriver, at et spil starter som en idé, og at det er spildesignerens opgave at sætte sig i spillerens sted, ændrer det ikke Bates opfattelse af selve udviklingsprocessen:

> *"Before anything else, building a game is a software development project"* [Bates, 2001: 220].

Det er med dette udgangspunkt han fremsætter de beskrevne metoder – vandfaldsmetoden og iterativ *prototyping*. Denne tilgang stemmer meget godt overens med nærværendes speciales definition af spil. Det er gennem reglerne at *gameplayet* kommer til udtryk, og reglerne er kodet ind i spillet. Da det er disse regler, der muliggør håndteringen af *in-* og *output* må koden betragtes som det mest fundamentale i at muliggøre spillets mest grundlæggende element – interaktiviteten.

Som noget unikt i forhold til de øvrige forfattere, skriver Bates om forholdet mellem spil og film:

> *"Because the end product is entertainment, many try to compare the process to producing a movie or writing a book. You can't. The techniques you apply to managing the process should be taken from the realm of software engineering"* [Bates, 2001: 220].

Her giver Bates klart udtryk for sin egen mening. Fordi både spil og films primære formål er at underholde, kan man altså ikke sammenligne processerne til skabelse af de to. Teknikker til at styre spilprocessen skal i stedet hentes fra softwaretraditionen. Bethke og Bates er altså enige på

dette punkt – spil er systemudvikling – men uenige om tilgangen. Hvor Bethkes tilgang tilnærmer sig vandfaldsmodellen, er Bates foretrukne tilgang baseret på et iterativt forløb. Det er højst sandsynligt også dette, der er medvirkende til, at Bethke kan opstille en komplet model for planlægningen af projektet, mens Bates slet ikke behandler dette. Endnu engang støder vi altså ind i problemet: Hvordan planlægger, tidsestimerer og budgetterer man et iterativt projekt?

RICHARD ROUSE: GAME DESIGN – THEORY & PRACTICE

BAGGRUND

Richard Rouse er *Design Director* ved firmaet *Surreal Software*, hvor han sidst har arbejde på *The Suffering* som projektleder, *lead designer* og forfatter [Rouse, 2005: VII]. Rouse startede som forfatter på Macintosh spilmagasiner *Inside Mac Games* og *Mac Games Digest,* mens han studerede på *University of Chicago* [Wiki, 2007: Richard Rouse III]. Da det ikke har været muligt at finde informationer om Rouses uddannelsesmæssige baggrund, har jeg kontaktet ham per mail. Denne kan findes i bilag 3, mens svaret kan findes i bilag 4. Rouse har en bachelor i matematik og computervidenskab, men han har også studeret litteratur og film. Han skriver dog følgende:

> *"Most of what I learned about game development and game design I learned through self-education and while developing projects"* [Bilag 4].

Rouses erfaring stammer altså primært fra erhvervserfaring.

PROCESMODEL

Selvom Rouse, ligesom Crawford, primært beskæftiger sig med spildesign, mener jeg stadig at bogen er relevant i forhold til dette projekt. Eksempelvis skriver Rouse om designdokumentet og brugen af dokumentation under udviklingen. Nødvendigheden af dokumentation er ifølge Rouse steget væsentligt, eftersom kompleksiteten og dermed udviklingsholdets størrelse er vokset markant [Rouse, 2005: 307]. Under et spilprojekt skal følgende dokumenter derfor skabes og vedligeholdes:

- Konceptdokument
- Analyse af konkurrencedygtighed

- Designdokument
- *Flowchart*
- *Story Bible*
- Manuskript
- *Art Bible*
- Spil Minuttet
- *Storyboard*
- Teknisk designdokument
- Tidsplaner og forretningsmæssige dokumenter

En mere fyldestgørende beskrivelse af ovenstående dokumenter kan findes i Bilag 5. Det interessante ved disse dokumenter er, at de alle på en eller anden måde er med til at justere, koordinere og synkronisere forventninger og vision for projektet. Eksempelvis skal *Art Bible* sikre, at alle har et ensartet billede af det æstetiske udtryk. Det samme gælder Spil Minuttet, der skal sikre en entydig forståelse af de mest grundlæggende spilmekanikker. Dette er interessant i henhold til Kroghs forventningsstyringsproblematikker og problemet med intern synkronisering og koordinering. Det, de ovenstående dokumenter gør, er at dele produktioner op i dokumenter, der beskriver en arbejdsgruppes retningslinjer for projektet. Med andre ord er deres funktion at kommunikere og fastholde denne kommunikation. Kunne dette tyde på at årsagen til at Krogs adspurgte virksomheder oplever problemer på dette punkt skyldes, at de ikke tager sig tiden til at dokumentere og beskrive spillets delsystemer? Hvis dette er tilfældet, hvordan kan man så i en proces tage højde for, at der ikke er tid til at dokumentere, og at der derfor skal kommunikeres på anden vis? Der er ingen tvivl om, at meget arbejde vil gå tabt, hvis man ikke kan fastholde projektets vision, og forskellige arbejdsgrupper som en følge heraf arbejder ud fra forskellige forestillinger.

Rouse skriver følgende om processen:

> *"In the games I work on, I prefer to keep the development process as organic as possible and I try not to plan anything out beyond what is necessary at that stage in development"* [Rouse, 2005: 283].

Årsagen til at Rouse foretrækker denne tilgang skyldes den uforudsigelige natur i spildesign. I stedet for at skrive en enorm mængde dokumentation foretrækker Rouse at gøre en del af spillet underholdende, før der bliver tilføjet yderligere detaljer og længde til spillet. Tidligt i

udviklingsprocessen skal man derfor koncentrere sig om spillets fokus, da det er let at omskrive, hvis spillet ændrer retning [Rouse, 2005: 285]. Det procesforløb, Rouse foreslår, minder derfor meget om Fullerton og Bates tankegang, hvor man starter processen med *prototyping* for at teste de mest basale funktioner i spillet. Rouse mener dog, at denne tilgang kan give ét problem for større etablerede virksomheder, der har et helt hold af udviklere. Ikke alle vil kunne arbejde på prototypen, så det mest optimale vil være at hive folk ind undervejs i processen efterhånden som designet bliver fastlagt. Rouse foreslår, at man anvender en *incremental* tilgang i udviklingen:

> *"The best way to build your game is incrementally. Instead of working a little bit on all the different components of the game, you should try to complete one system before moving on to the next. Work on the most basic and essential systems first, and then build the systems that depend on that system"* [Rouse, 2005: 286].

Fordelen ved denne tilgang er, at man kan implementere et system, teste det, se om det fungere som ønsket og derefter bevæge sig videre til implementeringen af det næste system. Når man har en fuldt fungerende prototype af spillet, er det næste trin af lave en del af spillet, der illustrerer, hvordan *gameplayet* kan fungere i det færdige spil. Eksempelvis i form af en bane, hvis der er tale om et førstepersonsskydespil. Dette skal være med til at opdage *gameplay*-mæssige elementer, der ikke fungerer som ønsket, dele man har glemt at implementere eller som fungerer på en ikke hensigtsmæssig måde [Rouse, 2005: 288].

Udarbejdelsen af egentligt indhold til spillet skal først starte, når man har en fungerende prototype af spillet:

> *"It makes no sense whatsoever to create these elements of the game until you have a firm grasp on what the gameplay will be, and have a working prototype that proves the gameplay to be fun"* [Rouse, 2005: 283].

Det forløb Rouse foreslår, ser altså ud som følger:

1. Design de mest grundlæggende funktioner i spillet.
2. Lav en prototype ved hele tiden at tilføje nye systemer.
3. Når prototypen er færdig skal *gameplayet* realiseres.
4. Når gameplayet er testet, kan man gå videre til at producere det egentlige indhold.

Rouses forløb minder meget om Fullertons tankegang, idet hun ligeledes ønsker at have en prototype af spillet så tidligt som muligt i forløbet for at undersøge spillets *gameplay*. Der er dog en forskel mellem de to. Hvor Fullertons tilgang er iterativ og baseret på *prototyping* foreslår

Rouses en *incrementiel* og evolutionær tilgang. Forskellen mellem de to tilgange er, at man i en *incremental* tilgang har fokus på at levere et funktionsdygtigt produkt efter hver *increment* [Pressman, 2000: 34]. Hvad der dog umiddelbart falder i øjnene ved Rouses tilgang er, at han beskriver, hvordan han i starten af forløbet forsøger at holde dokumentationen på et minimum [Rouse, 2005: 283]. Hvordan kan han så sikre, at alle på holdet fastholder den samme vision og arbejder mod det samme mål? Ligeledes skriver han:

> *"... and I try not to plan anything out beyond what is necessary at that stage in development"* [Rouse, 2005: 283].

Men hvordan kan man så budgettere projektet og overholde deadlines? Og hvordan skal man overhovedet kunne fastsætte deadlines?

Rouses tilgang giver også et andet problem. Ved hjælp af en prototype undersøger man, om det beskrevne *gameplay* rent faktisk fungerer – men hvad nu hvis det ikke virker? Hvad nu hvis der skal laves nye tunge programmeringsmæssige løsninger fordi *gameplayet* ikke fungerer, og derfor skal ændres til noget, der er mere ressource krævende? Rouse beskriver ikke, hvordan man tager højde for dette i den proces, han foreslår, og det samme gælder for de øvrige forfattere.

Rouse beskriver i otte trin processen i at designe og udvikle baner til spil:

1. *Preliminary*: Før man kan påbegynde processen i at lave og designe baner til spillet, bliver man nødt til at sikre sig, at alle de væsentligste *gameplay*-mæssige elementer er implementeret. Hvis spildesignet ændrer sig meget undervejs, kan det ifølge Rouse betyde, at meget af arbejdet med baner må kasseres, da de ikke længere er sjove at spille. Rouse's løsning på dette er at anvende en *preliminary* bane, mens *gameplayet* stadig er under udvikling, og først begynde den egentlige udvikling af baner, når man forventer at *gameplayet* ikke vil undergå store ændringer [Rouse, 2005: 468].
2. *Conceptual and Sketched Outline*: Før man kan påbegynde arbejdet på en bane, er det ifølge Rouse vigtigt at forstå, hvordan banen *gameplay*-mæssigt skal fungere, og hvordan den skal virke fra et narrativt perspektiv. Den bedste løsning på dette er derfor at nedskrive, hvorledes banen skal passe ind i spillet, og hvilke *gameplay*-elementer der skal anvendes. Derefter skal en tegning udarbejdes over banens layout. Formålet med at

anvende papir og blyant til dette er, at det er meget nemt at fange de første fejl og ændre dem. Ligeledes er det en fordel af få noget visuelt på plads, da en diskussion om mulige løsninger og forbedringer kan tage udgangspunkt i tegningen. [Rouse, 2005: 468]

3. *Base Architecture/Block out*: Det næste trin er ifølge Rouse at skabe grundarkitekturen til banen. Formålet er ikke at skabe en pæn bane, men at få en fornemmelse for hvordan spilleren kan navigere igennem banen, og at alle steder på banen er tilgængelige [Rouse, 2005: 469]. Målet er altså at skabe en bane, der uden det helt store arbejde kan give et indtryk af om layoutet fungerer.

4. *Redefine Architecture Until it is Fun*: Efter at grundarkitekturen er skabt, er det næste trin at redefinere denne, indtil det er sjovt. Det er på dette trin, man skal undersøge om arkitekturen rent faktisk giver mening og skaber den ønskede fornemmelse. Først når grundarkitekturen er gennemtestet, kan man bevæge sig videre til det næste trin. [Rouse, 2005: 69]

5. *Base Gameplay*: Dette trin er ifølge Rouse meget afhængigt af hvilken type af spil, der er tale om. Hvis det for eksempel er et førstepersonsskydespil, er det på dette trin, der skal tilføjes fjender, døre, små gåder og så videre [Rouse, 2005: 470]. Med andre ord skal det i trin 1 fastlagte *gameplay* implementeres.

6. *Redefine Gameplay Until it is Fun*: Da *gameplayet* ifølge Rouse er det, der skaber (eller ødelægger) spillet, er det essentielt, at designeren gentager trin 5 indtil banen er sjov at spille. Det er i denne proces, at det æstetiske udtryk skal kombineres med spillets *gameplay*, og dette kan give problemer. Designeren kan have lavet en æstetisk korrekt bane, der så viser sig ikke at passe sammen med *gameplayet*. Det kan derfor være nødvendigt at gå helt tilbage til trin 3 og lave grundarkitekturen om. Først når *gameplayet* er implementeret, og banen er sjov at spille, kan man gå videre til næste trin [Rouse, 2005: 471].

7. *Redefine Aesthetics*: Når spillets *gameplay* fungerer som ønsket, kan arbejdet med at få banen til at se godt ud begynde. Det er på dette trin at lyseffekter, *textures*, dekorative objekter og så videre skal tilføjes [Rouse, 2005: 471]. Med andre ord bliver banen leveret fra *level designeren* til grafikkerne og 3d-modellørerne.

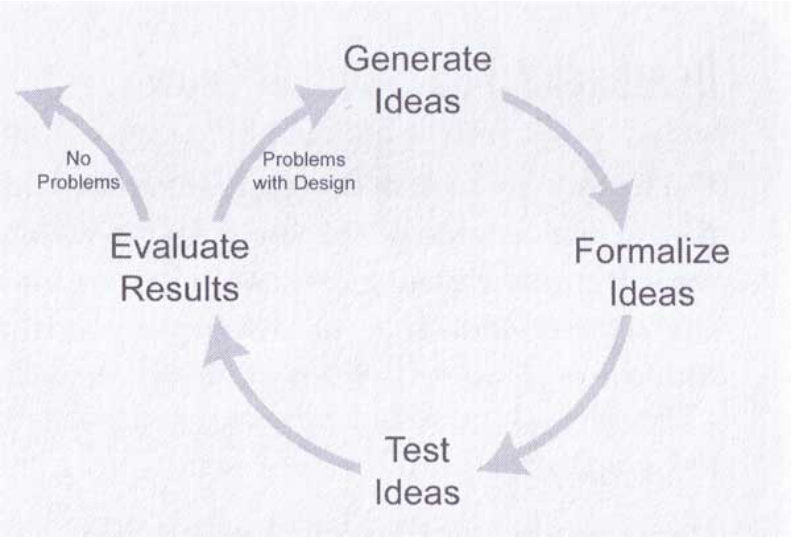
8. *Playtesting*: Når banens visuelle udtryk er på plads, er det på tide at få spiltestet banen. Dette kan gøres på flere måder. Eksempelvis kan man få folk udefra til at spille banen og derefter levere feedback. Herved kan man teste, om banen har den ønskede virkning. En

anden måde er at se en person over skulderen, for at undersøge om spilleren sidder fast og har svært ved at finde vej gennem banen. I værste fald vil man opdage, at banen slet ikke er sjov at spille, og at det derfor vil være nødvendigt at lave meget om. I bedste fald er det kun få detaljer, der skal ændres, før man sidder med en færdig bane [Rouse, 2005: 472].

Ifølge Rouse kan ovenstående tilgang dog varieres. Eksempelvis kan man i stedet for at tage en hel bane, starte med at lave layoutet til et enkelt rum. Ligeledes kan *playtesting* allerede finde sted efter trin 6 [Rouse, 2005: 472]. Selvom det grafiske udtryk mangler kan banens anvendelse af spillets *gameplay* sagtens testes.

Det er gennem Rouses otte trin tydeligt at se formålet med denne proces. Ved at udarbejde en bane på denne måde undgår man, at arbejdet med at designe og lave grafik til banen bliver spildt, da man alligevel ændrer det. Dette forudsætter dog, som Rouse også selv skriver, at spillets *gameplay* og historie er fastlagt fra begyndelsen. Ændrer et af disse elementer sig, er der en stor sandsynlighed for, at man må tilbage og lave banen om.

Rouses proces til udvikling af baner bygger på den samme tankegang som Fullertons procesmodel:



Figur 4.4: Idéen bag Fullertons procesmodel [Fullerton et al., 2004: 11].

Rouses tilgang bygger på præcist det samme fundament, hvor man starter på papirniveau og slutter med en færdig bane. Man bevæger sig først videre fra et trin til det næste, når man ikke længere kan forbedre det nuværende niveau. Et eksempel på dette er trin 2, hvor man starter med at tegne en papirskitse ud fra de i trin 1 fastlagte beskrivelser. Dernæst tager man en

diskussion, det vil sige tester idéen, og evaluerer så skitserne. Er den god nok til at vi kan bevæge os videre til det næste trin, eller er kritikken reel, så vi bliver nødt til at generere nye idéer? Dette er med til at understrege både Fullerton og Rouses formål med den beskrevne tilgang. De forsøger begge at minimere spildt arbejde ved at fastlægge som meget som muligt, mens det er let at lave ændringer. Til at starte med er banen tegnet på papir, og det er nemt at ændre designet. Herefter implementeres grundarkitekturen og denne skal så redefineres indtil den passer ind i spillets *gameplay*. Dernæst kan *gameplay* elementerne placeres og omrokeres på banen, indtil det giver den ønskede effekt. Til sidst kan grafikkerne og 3d-modellørerne arbejde på det visuelle af banen, og den kan testes. På den måde minimerer man risikoen for at eksempelvis en bygning senere skal laves om, fordi man ændrer på *gameplayet*.

Bemærk at fremgangsmåden forudsætter, at det er muligt at beskrive hvordan spillet kan passe ind rent historiemæssigt. Hvis det skal være muligt at køre efter Rouses model, skal spillets historie altså også være fastlagt inden designet af banen kan gå i gang.

SYN PÅ PROCESSEN

At det fremherskende element i Rouse forståelse af processen er *gameplayet* fremstår meget tydeligt, og han skriver da også i forbindelse med designet af baner til spil:

> *"In the end, gameplay must always trump all other considerations, whether aesthetics, story or technology. If it's not fun in the end, nothing else matters"* [Rouse, 2005: 471].

Dermed fremsætter han det syn, at spils primære formål er at underholde, og det derfor også er dette element, der skal være omdrejningspunktet for processen. Dette kommer til udtryk, ligesom ved Fullerton, i et ønske om at realisere *gameplayet* så hurtigt som muligt for at evaluere og forbedre det. Ligeledes har prototypen et andet formål. Den skal være med til at fastlægge så meget af spillet, at ændringer undervejs kun vil være små, og at risikoen for at smide store mængder eksempelvis grafisk arbejde ud minimeres.

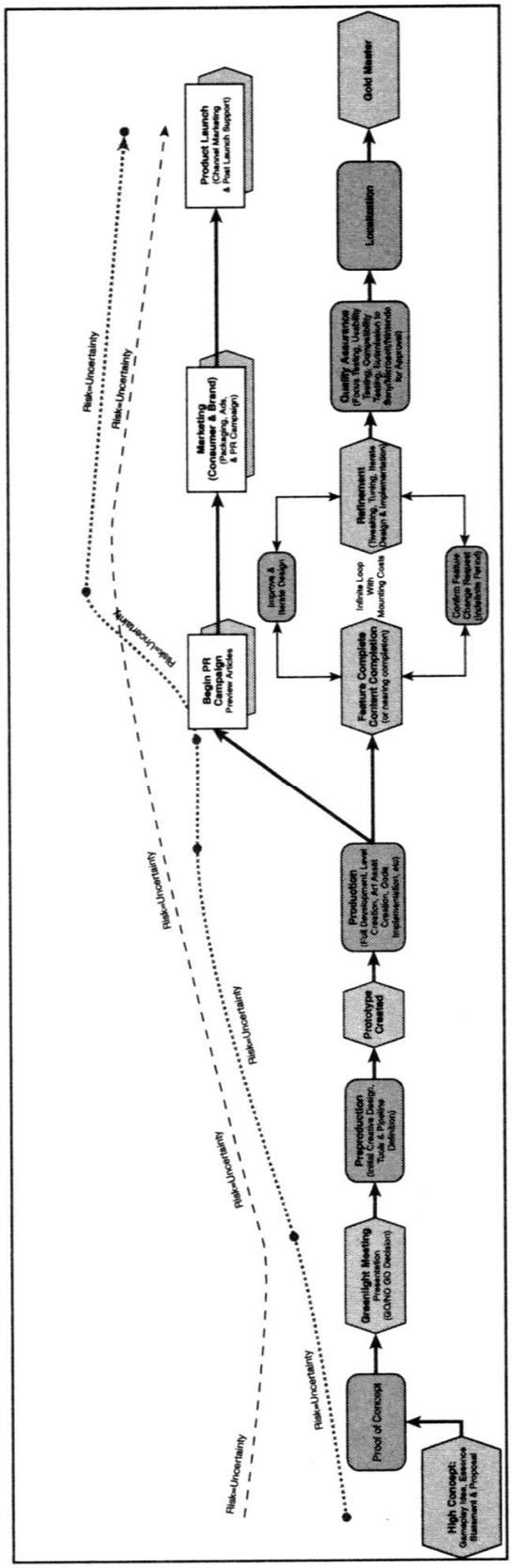
DAN IRISH: THE GAME PRODUCER'S HANDBOOK

BAGGRUND

Dan Irish har en lang erfaring som producer i spilindustrien, idet han har været aktiv siden 1993. Eksempelvis har han været producer på *Myst* ved *UbiSoft Entertainment* og de to *Homeworld* produktioner ved *Relic Entertainment* [Irish, 2005: vii]. Ifølge en beskrivelse af Irishs baggrund i et interview, har han ikke nogen uddannelsesmæssig baggrund, så al hans erfaring stammer fra erhvervslivet [Four Fat Chicks: 2007]. Det har ikke været muligt at undersøge dette nærmere, da det ikke har været muligt at finde nogle kontaktoplysninger. I så fald, er Irish den første og eneste af forfatterne, der ikke har en universitetsgrad.

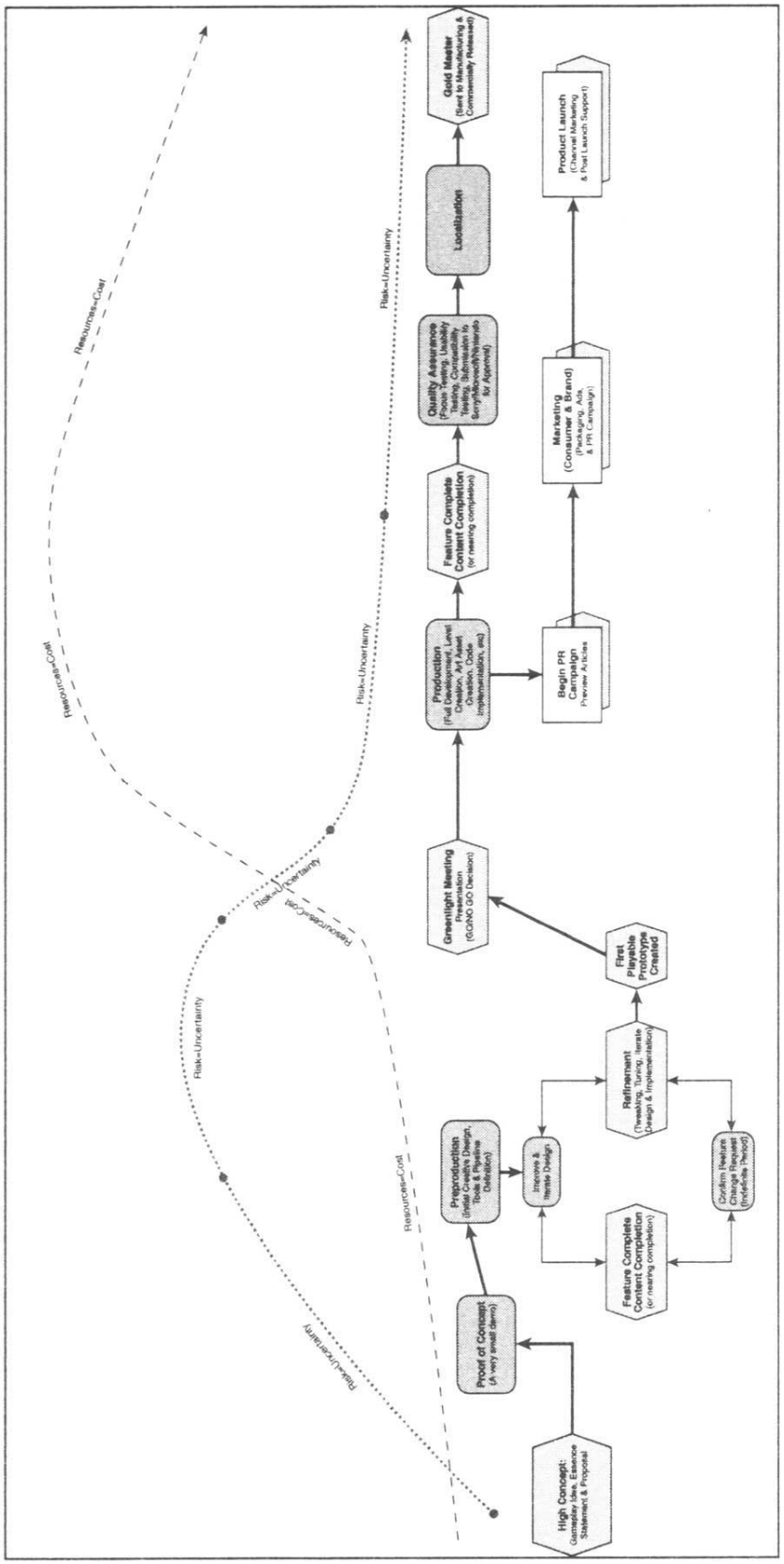
PROCESMODEL

Irish opstiller, som den eneste af de beskrevne forfattere, procesmodeller, der omfatter hele forløbet. Den første model han opstiller, illustrerer Irishs syn på, hvordan udviklingen af spil forløber på nuværende tidspunkt. Modellen er opsat på modstående side.



Figur 4.7: *The standard video game development process* [Irish, 2005: 208]. (Bemærk, at der er en fejl i modellen. Den stiplede linje repræsenterer *Resources=Cost* og ikke *Risk=Uncertainty)*.

Denne model illustrerer et forløb, hvor der først bliver skabt et koncept, og først når der er en komplet prototype kan spillet godkendes til fuld produktion. Problemet med denne tilgang er ifølge Irish, at som spillet bevæger sig igennem udviklingsprocessen stiger omkostninger i samme hastighed som usikkerheden. Irishs forslag til at undgå dette er en videreudvikling af standard modellen. Denne model kalder han *Front loading video game development process* og den er opsat på modstående side:



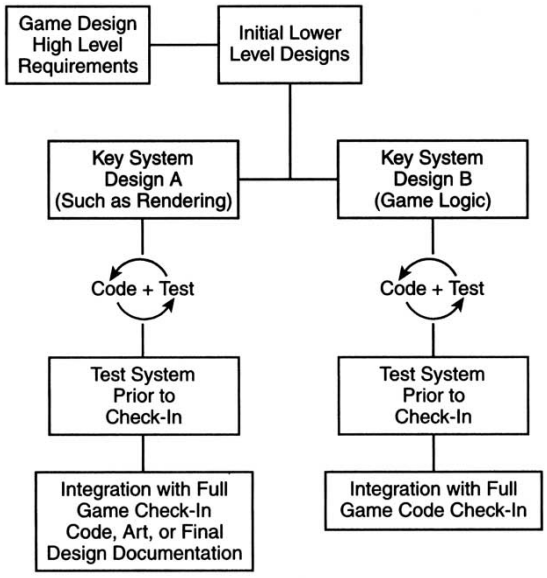
Firgur 4.8: *The front loading video game development process* [Irish, 2005: 208]

Denne model adskiller sig fra standardmodellen ved at have et iterativt forløb under præ-produktionen, og altså inden projektet har fået grønt lys til en fuld produktion. Her definerer Irish faserne [Irish, 2005: 208]:

- *Improve and iterate design.*
- *Refinement: Tweaking, tuning, iterate design and implementation.*
- *Confirm Feature change request.*
- *Feature complete, Content completion (or nearing completion).*

Irish beskriver ikke yderligere hvad disse punkter indeholder, men umiddelbart minder de meget om Fullertons punkter: *Generate ideas*, *Formalize ideas*, *Test ideas* og *Evaluate result*. Fordelen ved denne tilgang er ifølge Irish, at spiludvikleren og en eventuel *publisher* kan arbejde på at formindske risikoen i projektet gennem en iterativ tilgang, hvor man tester og redefinerer konceptet og spildesignet indtil en lovende prototype er fremstillet [Irish, 2005: 209]. På den måde er de kreative og tekniske risici reduceret i en sådan grad, at det bare er et spørgsmål om at producere spillet. Fordelene er ligeledes, at et lille hold hele tiden kan gennemgå prototypen for at undersøge, hvordan den kan forbedres, og at det ikke er nødvendigt med et helt udviklingshold før selve produktionen går i gang [Irish, 2005: 2008].

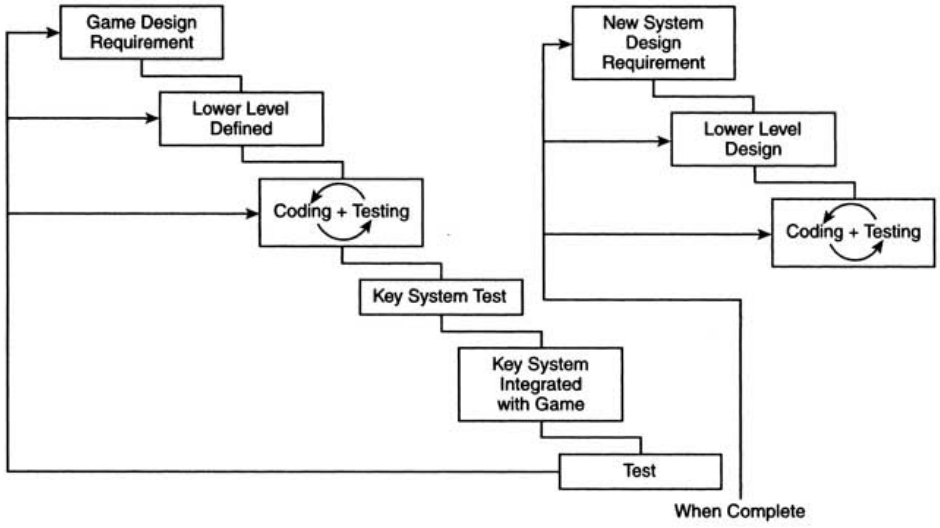
Irish opstiller fire procesmodeller for udviklingen af software. Da to af disse primært er en videreudvikling af de to andre, vil kun de to primære modeller blive behandlet i det følgende. Den første model, Irish kalder "*Increments to Completion*", ser ud som følger:



Figur 4.9: *Increments to Completion method* [Irish, 2005: 17]

Man starter med at designe systemet for derefter at dele denne op i systemer, der kan udvikles parallelt og uafhængigt af hinanden. Disse bliver designet, kodet og testet indtil de er færdige og klar til at blive integreret i spillet. Ifølge Irish er denne tilgang velegnet til eksempelvis udviklingen af en prototype [Irish, 2005: 17]. Dette virker problematisk, idet designprocessen ifølge modellen ikke er iterativ. Spillet bliver designet og derefter udviklet, uden at designet bliver revurderet. Dette er altså en kontrast til eksempelvis Fullertons tilgang, hvor designet bliver revurderet i hver eneste iteration. Ligeledes virker opdelingen i uafhængige delsystemer problematisk. Dette kræver jo netop, at designet ikke må ændres, da det kan betyde, at de to systemer ikke længere passer sammen, når de bliver integreret i spillet. Det er dog ikke utænkeligt, at tilgangen kan fungere til udviklingen af *enginen*, hvis denne kan opdeles i uafhængige systemer som eksempelvis *rendering* og spil logik.

Den anden model af Irish, jeg inddrager, er navngivet "*iterate-until-you-drop method*" og er afbilledet i nedenstående figur:



Figur 4.10: *Iterate-until-you-drop method* [Irish, 2005: 20].

Denne model er nærmest identisk med Fullertons iterative model (Figur 4.4 side 55). Irish åbner her for muligheden for enten at springe tilbage til at revurdere spildesignet, revurdere det detaljerede design, eller gå tilbage og ændre i koden. Dette afhænger af, hvad konklusionen er i testen på trin 6. Ifølge Irish er styrken ved denne model, at den lader *produceren* definere spillets design og efterfølgende tilbyde muligheden for at ændre det, imens spillet er under udvikling. Modellen lader spildesignerne revurderer, hvad Irish kalder for "*fun factor*", ved at give dem muligheden for at spille spillet og tilføje flere *gameplay*-elementer ved hver iteration [Irish, 2005: 19]. Ligeledes nævner Irish som en af de eneste forfattere ulemper ved en iterativ tilgang:

> *"The iterate-until-you-drop method becomes exactly that: a never-ending treadmill of software development that can always be improved"* [Irish, 2005:19]

Faren ved den iterative tilgang er altså ifølge Irish, at der ikke er defineret noget slutpunkt. Du kan hele tiden blive ved med at forbedre og tilføje elementer. Dette lægger et pres på de rammer, processen er underlagt, da det således gør det svært at definere og overholde *deadlines*. Irish skriver da også i sin bog om planlægningen af spiludviklingsprojekter, men han relaterer dem ikke til problemerne i den iterative tilgang.

Planlægning

Til udviklingen af et spil skal der ifølge Irish udfærdiges en produktionsplan. Om denne skriver Irish:

> *"Although a plan is often believed to remove uncertainty, in reality, the production plan is simply the best estimate of how the game is to be completed"* [Irish, 2005:5].

Med dette som udgangspunkt beskriver Irish to metoder til planlægning: *Top-Down* og *Bottom Up*. Med *top-down* tilgangen er det ifølge Irish en mindre gruppe, der danner et overblik over, hvordan processen kan se ud. *Top-down* tilgangen kan derfor kun anvendes til at fastsætte mål eller lave retningslinjer for projektet [Irish, 2005: 22]. Som han også skriver:

> *"At best, it is a guess; at worst, it is totally wrong"* [Irish, 2005: 22].

I *buttom up* tilgangen vil hele holdet ifølge Irish arbejde på at estimere og planlægge projektet. Fordelen ved dette er, at hele holdet er med til at skabe en plan for projektet, og dermed også får ejerskab af den [Irish, 2005: 23]. Denne tilgang kan dog kun anvendes, hvis designdokumentet er færdigudarbejdet, og Irish skriver da også, at denne form for planlægning først kan finde sted, efter præ-produktionen. Ifølge Irish er en plan for et spil dog aldrig færdig:

> *"Requirements, features, designs, available resources, and time all change during the course of a project, and there are now any solutions around having to constantly reevaluate and consider all of the variables and their impact"* [Irish, 2005: 75].

Planen skal derfor hele tiden evalueres og opdateres. Det interessante er, at Irish ikke på nogen måde integrerer dette i de processer, han beskriver. Han skriver, at det er en nødvendighed, og at det kan være nødvendigt at reevaluere planen op til 12 gange årligt [Irish, 2005: 75]. Det undrer mig derfor, at dette ikke er integreret i de procesmodeller, Irish opstiller.

Irish skriver også om brugen af *milestones* i planlægningen. Disse anvendes ofte til at synliggøre en dato, hvor spiludvikleren skal aflevere en delleverance til kunden (oftest en *publisher*). Hver *milestone* er en beskrivelse af, hvor projektet skal være på den fastsatte dato. Det er så op til *publisheren* at godkende delleverancen og afgøre, om den lever op til den fastsatte *milestone* [Irish, 2005: 93]. Ofte bliver der udbetalt en vis sum af den samlede ordre, ved hver *milestone* der godkendes. *Milestones* er interessante, idet de kan anvendes til at sætte rammerne for projektet. De er formodentlig fastsat ud fra en *top-down* tilgang, hvor der er blevet opsat en række mål for projektet, der så igen kan bruges til at definere og beskrive *milestones*.

Som det også ses af ovenstående, opstiller Irish ikke en entydig model for, hvordan man effektivt planlægger udviklingsprocessen. Han skriver da også:

> *"It seems that there's no unifying theory for scheduling in the game industry"* [Irish, 2005: 210].

Hvilket, set i forhold til de øvrige forfattere, indtil videre ikke kan modbevises. De enkelte af forfatterne, der har beskrevet selve planlægningen af spiludviklingen, har ikke opsat en entydig universal model for, hvorledes denne skal forløbe. Oftest er det, ligesom det også er tilfældet ved Irish, en ustruktureret gennemgang af forskellige teknikker og metoder, der kan anvendes i forbindelse med planlægningen. Det er efterhånden tydeligt, at planlægningsproblematikken i forbindelse med spiludvikling er større end først antaget, og en procesmodel til udviklingen af spil også skal beskrive hvorledes planlægningen er integreret.

SYN PÅ PROCESSEN

Irish giver tydeligt udtryk for, at et spil ikke er en succes, hvis det ikke er underholdende: Irish skriver eksempelvis følgende om *Front loading video game development process*:

> *"The goals remain the same as with the standard model: To create a great game, that's commercially successful"* [Irish, 2005, 208].

Formålet med den model han foreslår til udviklingen af spil, er altså at sikre et spil, der er underholdende, og en forretningsmæssig succes. Dermed er han også den første af forfatterne, der inddrager det økonomiske aspekt som argument for, at spillet skal være underholdende. Hvordan man præcist sikrer, at spillet bliver sjovt, beskriver han ikke nærmere. Dog foreslår han, som mange af de øvrige forfattere, en iterativ model til udviklingen af en prototype for at give spildesignerne mulighed for at reevaluere designet i processen. Det står efterhånden klart, at de

fleste forfattere mener, at den bedste model til at sikre et underholdende spil er et iterativt forløb.

Irish adskiller sig fra de øvrige forfattere ved at være den første, der inddrager de agile tankegange. Dette gør han i forbindelse med de opsatte metoder til selve systemudviklingsdelen af processen. Her inddrager han de fem faser, der udgør grundlaget for den agile proces; *Envision*, *Speculate*, *Explore*, *Adapt* og *Finalize* [Irish, 2005, 21]. Han nævner dog intet om, hvordan disse faser kan anvendes i en procesmodel eller i det hele taget, hvordan man kan anvende de agile tankegange. Det virker underligt at ingen af forfatterne indtil videre har inddraget de agile metoder i højere grad end som så. De agile metoder er jo netop systemudviklingens svar på en tilgang, der tager højde for, at systemet skal ændres undervejs i processen. Og det er jo netop dette problem forfatterne har opdaget, at der er ved spiludvikling: Et spil skal være underholdende ved hjælp af interaktiviteten, der skal realiseres for at kunne revurderes, hvorfor en iterativ tilgang altså er at foretrække. Agile udviklingsmetoder som SCRUM er jo netop en videreudvikling af tankegangene bag den iterative tilgang, så hvorfor har ingen af forfatterne i højere grad inddraget disse metoder?

Efter nu at have belyst de seks forfatteres tilgang til spiludvikling, vil jeg i den efterfølgende opsamling se nærmere på, hvilke konklusioner og erfaringer jeg har fået på baggrund af materialet. Ligeledes vil jeg forsøge at besvare de spørgsmål, jeg har rejst igennem afsnittet.

OPSAMLING

Gennem det netop beskrevne afsnit, synes jeg, at jeg har fået en af mine fordomme bekræftet – forfatternes konklusioner er baseret på deres oplevelser og ikke nødvendigvis teoretisk funderet eller reflekteret. Forfatterne fremsætter usammenhængende metoder, og ofte er deres forslag til anvendelse af teknikker til eksempelvis budgettering direkte modstridende med tankerne bag deres procesmodel. Dette ser vi eksempelvis ved Fullerton, der foreslår en iterativ tilgang til spiludvikling, mens teknikken hun foreslår til planlægning og budgettering kræver, at spillet kan opdeles i delleverancer, der så igen kan nedbrydes til delopgaverne, der skal tids- og ressourceestimeres og placeres i eksempelvis et Gantt diagram. Hun foreslår altså en iterativ tilgang til spiludvikling, mens hendes planlægningsproces følger en traditionel og lineær tankegang. Resultatet af dette er, at metoden bliver usammenhængende, da processerne nærmere modsiger end understøtter hinanden. Bates og Rouse gør det endnu værre end Fullerton, idet ingen af dem beskriver, hvorledes man kan estimere og planlægge i de iterative tilgange, der bliver foreslået. Irish foreslår ligeledes en iterativ tilgang og forsøger at give et bud på, hvordan en planlægningsproces kan se ud. Problemet er, at Irish allerede inden han går i gang med dette indser, at der ikke findes nogen veldefinerede metoder til planlægning af spil, og derfor bliver det opsatte forslag mere en beskrivelse af teknikker, der kan og bliver anvendt end en sammenhængende planlægningsproces.

Den eneste af forfatterne der formår at fremsætte en helstøbt og sammenhængende model er Bethke. Dermed ikke nødvendigvis sagt, at dette er en god model. Bethkes tekniske baggrund skinner tydeligt igennem, og han lægger heller ikke skjul på sine synspunkter: *"Games are software with art, audio and gameplay"* [Bethke, 2003: 4]. Dette resulterer i en fuldstændig lineær men dog sammenhængende model, der på ingen måde tager højde for, at det primære formål med spil er at underholde. Dermed er det svært at se anvendeligheden af Bethkes metode.

Det er mit indtryk, at alle forfatterne har baseret deres resultater på erhvervserfaring. Det virker ikke som om de har reflekteret over, hvad en spiludviklingsproces egentlig er. Endvidere virker det som om flere af forfatterne har nedskrevet deres metoder, efter hvad man gør i branchen, og derved ikke nødvendigvis sat spørgsmålstegn ved om denne tilgang er den bedste. Det er fortsat mit mål for resten af opgaven ikke at falde i samme fælde. Fordi man allerede nu har integreret nogle processer og teknikker i spilbranchen, betyder det ikke nødvendigvis, at dette er de mest optimale. Det kan blot være i mangel af bedre.

I nedenstående tabel har jeg lavet en oversigt over de forskellige forfattere, fordelt på den proces de foreslår, og det syn de har på processen. Formålet er at give dig som læser et overblik over forfatternes processer og synspunkter:

	Proces	**Syn**
Bethke	Tilgangen minder om vandfaldsmodellen. Planlægningen baseres på et komplet *game*- og teknisk designdokument.	*Games are software with art, audio and gameplay.*
Crawford	Interaktionen som udgangspunkt for spildesignet. Spiludvikling er en kreativ proces og udtryk.	Æstetiske indtryk er til for at forbedre *gameplayet*. Det vigtigste område for spildesignet er selve interaktionen.
Fullerton	Ønsker at fastlægge de vigtigste elementer i designet så hurtigt som muligt ved hjælp af *Prototyping*. Fullerton foreslår en iterativ proces gennem faserne design, formalisering, test og evaluering.	Spilleren som centrum for spildesignet og processen. Spildesign er en løbende proces. Fokusering på de mest grundlæggende elementer i spildesignet først.
Bates	Foreslår to tilgange; vandfaldsmodel og iterative *prototyping*, hvoraf han finder sidstnævnte mest anvendelig. Dette skyldes, at det ofte er svært at fastlægge store dele af spillet, og der er en stor sandsynlig for, at de allerede fastlagte elementer ændrer sig. Kombination af modellerne er dog også en mulig tilgang.	Spildesignet starter med en idé. Spiludviklingen starter med programmering. Spil er ikke film bare fordi formålet med dem begge er at underholde. Teknikker og forståelsen for processen skal hentes fra softwareudviklingstraditionen.
Rouse	*Prototyping* og *incrementiel* tilgang. Dokumentation til at sikre at alle arbejder mod det samme mål. Når *gameplayet* er implementeret skal man gå videre og udarbejde indhold, der	*Gameplayet* er vigtigere end alle andre overvejelser i forbindelse med spillet. Spil skal først og fremmest være underholdende.

	illustrerer *gameplayet*. Indhold udvikles, så det er nemt at lave ændringer undervejs. Forsøger at holde dokumentation på et minimum i starten af processen.	
Irish	Gennem en iterativ proces udvikles en prototype. Herefter overtager en produktionsfase, der følger en lineær model. *Milestones* som ramme for udviklingen. Der findes ingen entydig måde at planlægge spiludvikling. *Top-down* tilgang i starten af udviklingen. *Bottom up* efter præ-produktionen. En plan er aldrig færdig.	Spil skal være underholdende – ellers vil det ikke blive solgt. Det underholdende aspekt som fokus for processen.

Tabel 4.11: Opsamling på forfatternes proces og syn (Egen tilvirkning).

Med undtagelse af Bethke sætter samtlige forfattere enten interaktiviteten eller *gameplayet* som det vigtigste element i processen. Man kan derved sige, at de i bund og grund bygger processen op om det samme, idet interaktiviteten kommer til udtryk gennem hvad spilleren kan gøre i selve spillet - hvilket er det, vi også kalder *gameplayet*. Samtlige forfattere, igen med Bethke som en undtagelse, ser spiludvikling som en iterativ proces. Det besynderlige ved at Bethke foreslår en tilgang, der minder meget om den traditionelle vandfaldsmodel er, at han godt selv ved, at et spil ændrer sig undervejs. Da jeg begyndte at læse Bethkes bog, som den første af de seks bøger, kom jeg virkelig i tvivl – er planlægning løsningen? Kan man designe et komplet spil inden selve udviklingen går i gang? Efter af have læst de øvrige bøger er jeg ikke længere i tvivl – spiludviklingen er en iterativ proces. Fullerton og Bethke er dog, på trods af at deres modeller er så forskellige, enige om, at udviklingen ofte går i gang for tidligt [Fullerton et al., 2004: 10] [Bethke, 2003: 26]. Forskellen ligger dog i at Bethke ønsker at planlægge alt, inden man går i gang, mens Fullerton ønsker at anvende tiden til at lave en prototype, der kan visualisere spillet, så det kan testes af målgruppen. Bethkes tilgang er højst sandsynlig kun anvendelig i projekter, hvor der er meget få usikkerheder omkring slutproduktet. Eksempelvis hvis der skal produceres et spil, der lægger meget tæt op af en eksisterende titel. Jeg har fået det indtryk, at Bethke har skrevet bogen, i et forsøg på at redde samtlige spilvirksomheder fra at gå konkurs. Alt for mange

spil giver ifølge Bethke underskud, og gennem bogen fremgår det tydeligt, at Bethke kun ser én måde at sikre et overskud - planlægning. Han nævner dog selv, at spilprojekter ofte ændres undervejs i et forsøg på at lave et godt spil, og han har aldrig oplevet at designdokumentet var skrevet færdigt inden udviklingen gik i gang [Bethke, 2003: 224]. Dermed har han allerede selv skudt sin teori i stykker godt halvvejs inde i sin bog.

Men hvilke muligheder er der så? Forfatterne er stort set enige. Svaret er en iterativ model, hvor man tager højde for, at projektet med stor sandsynlighed vil ændre sig undervejs i et forsøg på at gøre spillet underholdende. Problemet, der opstår i forbindelse med dette, er, at ingen af forfatterne foreslår den samme procesmodel. Alle procesmodellerne, med undtagelse af Bethkes, er baseret på en iterativ tilgang, men forfatterne opsætter processen forskelligt, prioriterer elementer forskelligt, og de anvender forskellige termer. Den primære årsag til at forfatterne foretrækker den iterative tilgang skyldes, at de alle ved, at spillet ændrer sig i et forsøg på at gøre det underholdende. Hvad der ligeledes er tydeligt er, at der er behov for et fælles sprog og en egentligt udviklingsmetode til spil. Primært for at skabe en fælles referenceramme, når man snakker om kreationen af spil. Dette er dog ikke det første problem. Ingen af forfatterne formår, som allerede beskrevet, at flette deres iterative procesmodel sammen med en planlægningsproces på en sådan måde, at de understøtter hinanden. Det næste trin må således være at belyse, hvordan man planlægger et projekt, der med stor sandsynlighed ændrer sig, og jeg tror allerede jeg ved, hvor jeg kan finde svaret…

Irish er den eneste af forfatterne, der nævner de agile metoder, og han bruger kun to sider på at beskrive de fem faser, der ligger bag de agile tankegange. Det virker besynderligt, at han er den eneste, der inddrager de agile metoder, da disse jo netop er systemudviklingstraditionens svar på en metode, der tager højde for at produktet ændres undervejs. Hvor ureflekteret og overfladisk Irish behandler emnet ses tydeligt, da han blandt andet skriver følgende om, hvorledes man skal bære sig ad med at aflevere delleverancer til tiden:

> "*You can do this using effective time-management, ressource-allocation, and risk-management strategies*" [Irish, 2005, 21].

Det er mig stadig en gåde, hvorfor ingen af de seks forfattere i højere grad har belyst de agile metoder. Min egen erfaring med det agile princip stammer blandt andet fra min bachelor uddannelsen i Informatik ved Teknisk-Naturvidenskabeligt Fakultet, hvor vi i samarbejde med FDM udviklede en applikation ved hjælp af *Extreme Programming*. Ligeledes skrev jeg i min

niende semesters opgave om, hvorledes man i udviklingen af et produkt tager højde for, at det ændrer sig, hvor jeg også kom ind på livet af ideerne og tankerne bag de agile metoder. Det er blandt andet dette, der er årsagen til, at jeg mener, at jeg kan finde hjælp til at planlægge et projekt i forandring i litteraturen om de agile metoder. Det næste trin er derfor at undersøge mulighederne i anvendelse af agile metode til spiludvikling, samt hvorledes planlægningen forløber i de agile metoder.

FASE 5 – PLANLÆGNING I SPILUDVIKLING

I dette afsnit vil jeg inddrage de agile metoder. Størstedelen af de i forrige afsnit behandlede forfattere betragter spiludvikling som en iterativ proces, primært fordi spil ændrer sig undervejs i udviklingen i et forsøg på at gøre slutproduktet underholdende. De agile metoder er netop systemudviklingstraditionens svar på en proces, der tager højde for, at kravene til produktet ændres undervejs. Derfor vil jeg, som det første i dette afsnit, beskrive de agile metoder, og hvordan jeg kan se en anvendelse af disse i spiludvikling. Til at beskrive de agile metoder vil jeg primært tage udgangspunkt i min niende semesters opgave "Fornemmelse for forandring – Om at lede udviklingen af et produkt i forandring" [Veigaard, 2006]. Efterfølgende vil jeg så se nærmere på, hvordan man planlægger et projekt, hvor kravene ændres undervejs.

DE AGILE METODER

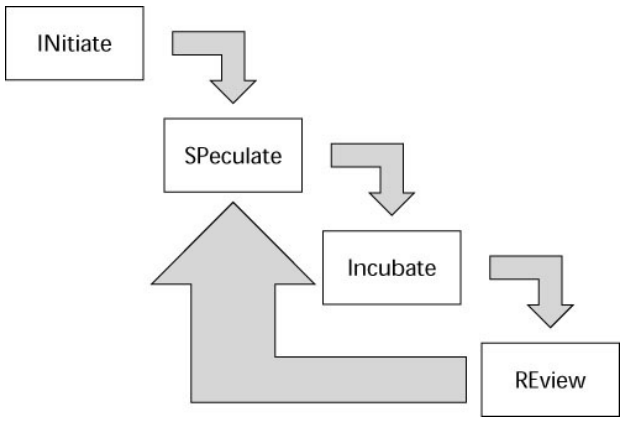
Som netop beskrevet vil jeg i dette afsnit beskrive de agile metoder. Jeg tror, baseret på mit kendskab til de agile metoder, at der er meget at hente. Jeg tror dog ikke, at jeg i de agile metoder finder en komplet løsning. *Agile* udvikling er stadig fokuseret på selve koden, hvor spil også indeholder grafik, lyd, animationer og så videre. Ligeledes fokuserer de agile metoder meget på afleveringen af funktioner, og i spil er en funktion ikke bare et stykke kode. I spilbranchen anvendes termen *game mechanic* ofte om beskrivelsen af noget, der kan ske i *game universe*, og før en sådan er færdig, skal der laves kode, grafik, og lyd, og den skal realiseres i eksempelvis et banedesign. På trods af disse formodede forskelle tror jeg stadig, at der er noget at hente i idéerne i den proces, de agile metoder følger.

De agile tankegange opstod i midten af 90'erne som en reaktion mod, hvad tilhængere kaldte "de tunge metoder". Det bedste eksempel på en metode, der er en direkte modsætning til de agile tankegange er vandfaldsmodellen. I 2001 skrev Kent Beck sammen med en lang række softwareudviklere det Agile Manifesto. Dette manifest er en dokumentation af, hvorledes de betragter udviklingen af software formuleret ved hjælp af en række principper. Et af disse punkter lyder som følger:

> *"Responding to change over following a plan"* [Beck et al., 2001].

Dette er altså en direkte kontrast til eksempelvis vandfaldsmodellen, hvor alt er planlagt inden selve udviklingen går i gang. Men hvordan ser udviklingsprocessen så ud for de agile metoder? For at belyse dette, inddrager jeg Extreme Project Management (XPM), som beskrevet af Doug Decarlo i *eXtreme Project Management* [DeCarlo, 2004], og SCRUM metoden, som beskrevet af Ken Schwaber i *Agile Project Management with Scrum* [Schwaber, 2004]. Jeg inddrager de to metoder, fordi de bevæger sig på forskellige niveauer indenfor det agile princip. XPM er baseret på selve ledelsen af den agile proces, og de krav de agile metoder sætter til organisationen. SCRUM er derimod en decideret udviklingsmetode, der beskriver en komplet proces. Formålet med at inddrage dem begge er at belyse processen fra netop de to niveauer, i en forhåbning om at dette vil give et godt billede af idéerne bag de agile metoder.

DeCarlo opstiller i sin bog, hvad han kalder *Flexible Project Model*. Dette er en model, der beskriver de grundlæggende faser i hver iteration i de agile metoder. Det er også disse faser, Irish bruger to sider i sin bog til at beskrive [Irish, 2005, 21]. Modellen ser ud som afbilledet i nedenstående figur:



Figur 5.1: Flexible Project Model, også kaldet INSPIRE, der viser de grundlæggende faser i den agile iteration [Wysocki, 2003, Overview]

Nedenstående er en beskrivelse af de fire faser *Flexible Project Model* indeholder:

- *INitiate*: I denne fase bliver det forretningsmæssige formål for projektet etableret, og der bliver sammensat et udviklingshold.
- *SPeculate*: Denne fase gentages hver gang en iteration starter. Formålet er at udforske idéer og løsninger til systemet, og at beslutte hvad der skal leveres hvornår. Dette gøres på et møde, hvor kunden og udviklingsholdet er til stede.

- *Incubate*: Dette er selve realiseringsfasen. Her bliver planen, der blev udarbejdet i SPeculate fasen, udført. Undervejs udforskes nye løsninger, og kunden kan altid foreslå ændringer, der så kan vurderes i samarbejde med udviklingsholdet.
- *Review*: I denne fase vurderes produktet både økonomisk og funktionsmæssigt. Det besluttes om projektet skal fortsætte og i hvilken retning. Med andre ord vurderes visionen for projektet.

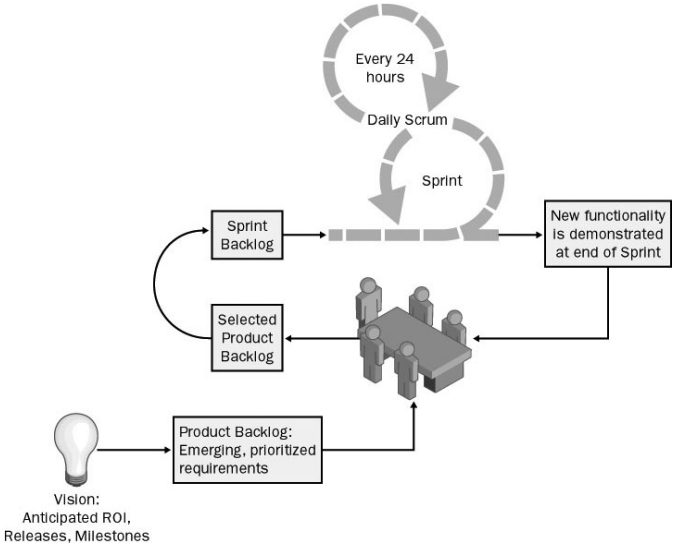
Det er altså disse rammer ethvert agilt forløb, ifølge DeCarlo, følger. Det er altså også disse trin, en metode som eksempelvis SCRUM vil være bygget på. Det efterfølgende er en beskrivelse af SCRUM processen.

SCRUM

SCRUM er opbygget omkring tre roller:

1. *ScumMaster*: Der underviser udviklingsholdet i, hvorledes man følger metoden. Det er personens opgave at sikre, at alle på holdet følger tankegangene bag SCRUM, og at metoden bliver tilpasset til den organisation, den skal fungere i.
2. *Product Owner*: Finansiere produktet og definere kravene til systemet. Det er *product owner*, der skal have visionen for projektet.
3. Udviklingholdet: Består af de personer, der skal lave systemet.

SCRUM processen er baseret på iterationer af cirka 30 dage. Selve processen ser ud som afbilledet i nedenstående figur:



Figur 5.2: Overblik over SCRUM processen [Schwaber, 2004, Scrum flow].

Det første trin er at fastsætte visionen for projektet. Det vil sige, det forventede *return on investment*, delleverancer og *milestones*. Hernæst skabes der en *Product Backlog*. Dette er en beskrivelse af de funktioner, systemet skal indeholde, en prioritering af disse samt en opdeling i delleverancer. Inden for det agile princip kaldes funktioner ofte *user stories*. Årsagen til dette er, at de er formuleret, så det er en historie om, hvad brugeren kan gøre i systemet. Et eksempel på en *user story* til et skriveprogram kunne lyde: "Som bruger ønsker jeg at kunne gøre teksten kursiv". *Backloggen* er altså en prioriteret og sorteret rækkefølge af *user stories*. Det er *product owners* ansvar at sikre, at *backloggen* hele tiden bliver opdateret. Bemærk at *backloggen* ikke er en komplet beskrivelse af, hvad systemet skal indeholde, men blot de funktioner *product owner* ønsker. Efter hver iteration kan *product owner* tilføje, fjerne eller udvide funktioner i *backloggen*.

Backloggen bliver herefter diskuteret på et møde, hvor udviklingsholdet får mulighed for at stille afklarende spørgsmål. Resultatet er en opdatering af *backloggen*, hvor det fremgår hvilke funktioner, der forventes implementeret under den næste *sprint*. *Sprint* er SCRUM-processens term for udviklingsfasen. Imens udviklingen er i gang, afholdes dagligt et *Daily Scrum Meeting*, hvor hvert enkelt medlem skal synliggøre, hvad de har lavet siden sidste møde, hvad de kan nå til det næste, og om de sidder med nogle problemer, eller forventer at problemer vil opstå. Når udviklingsfasen er gennemført, er resultatet et opdateret produkt, hvor de nye funktioner er blevet integreret. Herefter afholdes igen et møde, hvor *product owner* kan præsentere en opdateret *backlog*. Funktionerne, der skal implementeres i det næste sprint, udvælges, og løkken på det iterative forløb er således bundet. Det er tydeligt, at SCRUM metoden følger faserne i *Flexible Project Model*. Således er *Flexible Project Model* altså en afbildning af faserne, mens SCRUM er en konkret videreudvikling af disse opsat som en egentlig udviklingsmetode.

Ovenstående beskrivelse af de agile metoder skimmer kun overfladen. Formålet med afsnittet er primært, at du, som læser, skal forstå tankegangene bag de agile metoder, inden jeg kigger nærmere på selve planlægningen af et agilt forløb. For en mere udførlig beskrivelse af de agile metoder, deres formål og hvorfor de er et interessant alternativ til de traditionelle metoder, kan jeg anbefale mit 9. semesters projekt, der er tilgængeligt på den medfølgende CD-rom [Veigaard, 2006].

Efter nu at have belyst og beskrevet processen bag de agile metoder, vil jeg se nærmere på hvorledes man planlægger et agilt forløb.

MIKE COHN: AGILE ESTIMATING AND PLANNING

Den iterative proces, der ligger bag de agile metoder, virker velegnet til udviklingen af spil, da disse ligeledes er udsat for, at kravene ændres undervejs i processen. Men hvordan planlægger man et sådan forløb? Hvordan estimerer man en opgave, der med stor sandsynlighed vil ændre sig, før den er færdig? Det er dette, jeg i dette afsnit ønsker at belyse. Til at gøre dette inddrager jeg Mike Cohn og bogen *Agile Estimating and Planning* [Cohn, 2006]. Ved at belyse hvordan man estimerer og planlægger et agilt forløb, håber jeg også at blive klogere på, hvorledes dette kan lade sig gøre i spiludvikling.

EN *AGILE* TILGANG TIL PLANLÆGNING

Hvis man hele tiden skal kunne reagere på forandringer, kræver det en planlægningsproces, der muliggør dette. I bogen *Agile Estimating and Planning* forsøger Mike Cohn at tilbyde en proces, der netop muliggør en *agile* planlægning. Formålet med at kunne reagere på forandringer er ifølge Cohn, at man således vil kunne skabe et produkt, der har mere værdi for kunden [Cohn, 2006: 22]. Ligeledes skriver han:

> *"For all but the simplest projects, it is impossible for users to know every detail of every feature they want. It is inevitable that users will come up with new ideas, and almost as inevitable that they will decide that some features desired today will become lower priorities tomorrow"* [Cohn, 2006: 22].

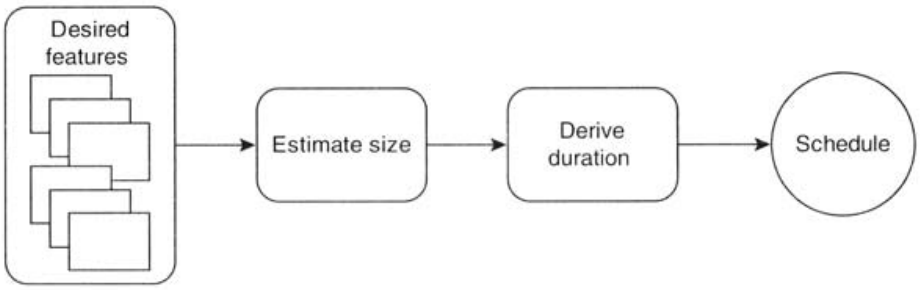
Netop denne beskrivelse passer godt på udviklingen af spil. Det er umuligt at kende alle detaljer til, hvorledes spillet skal være sjovt. Nye idéer vil uundgåeligt opstå, og elementer vil med stor sandsynlighed blive ændret under forløbet for at balancere spillet og gøre det sjovere. Dermed øger man også produktets værdi for kunden og spilleren.

Agile planlægning er ifølge Cohn en aktivitet frem for en egentligt plan [Cohn, 2006: 9]. De agile udviklingsmetoders formål er at tage højde for forandringer, så en agile planlægningsproces skal altså tage højde for, at ændringer nemt skal kunne indføres i planen. Gennem et projekt vil nye indsigter opstå, og i takt med at disse opnås, påvirker det projektets planer. Netop dette er ifølge Cohn årsagen til, at planlægningen bliver vigtigere end selve planen [Cohn, 2006: 9]. Den viden og indsigt vi får ved at planlægge holder længere end planen, der senere bliver splittet op og en revideret udgave sat i dens sted [Cohn, 2006: 10]. *Agile* planlægning er altså en proces, hvor man sætter fokus på at gøre det så nemt at foretage ændringer i planen som muligt. En agile plan er

altså dermed også en plan, der er nem at ændre. En yderligere beskrivelse af de krav Cohn opsætter for planlægningsprocessen kan findes i Bilag 6.

RAMMERNE FOR PROJEKTET

Når man starter et projekt, er det vigtig for kunden at vide, hvor lang tid projektet vil tage, og hvad det vil koste. Det første trin er derfor at fastlægge rammerne for projektet. Cohns fremgangsmåde til dette er opsat i nedenstående model:



Figur 5.3: Cohns visualisering af, hvordan man estimerer varigheden af et projekt [Cohn, 2006: 38]

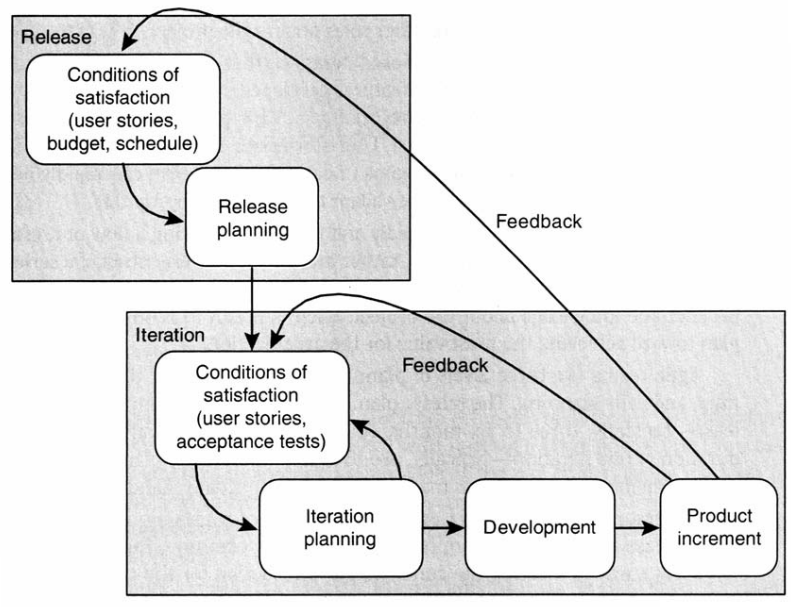
Idéen bag modellen er, at man først samler alle de funktioner, systemet skal indeholde. Dette kan enten være i form af *user stories* eller *themes*, der er en gruppering af relaterede *user stories*. Dernæst fastsættes hver funktions relative omfang ved hjælp af *story points*. Betydningen af 'relativ' ligger i, at omfanget ikke er bestemt ud fra en fastsat tid (som eksempelvis ti timer), men en relativ værdi i forhold til de andre arbejdsopgaver. Cohn foreslår, at der ved uddelingen af *story points* anvendes værdierne 1, 2, 3,5 og 8, hvor 8 er den største opgave og 1 er den mindste [Cohn, 2006: 53]. Det næste trin er således at tildele hver *theme* et *story point* alt efter deres relative størrelse. Til dette foreslår Cohn en teknik, han kalder *Planning Poker*. En beskrivelse af denne kan findes i Bilag 7. Når dette er gjort, skal varigheden af projektet estimeres. Til dette foreslår Cohn anvendelsen af *velocity*, der er en måling af udviklingsholdets hastighed. Det vil sige mængden af *story points,* udviklingsholdet kan nå hver iteration. *Velocity* kan fastsættes ved hjælp af et estimat, men dette forudsætter at udviklingsholdet har arbejdet sammen før. Hvis ikke dette er tilfældet, kan *velocity* først måles, når den egentlige udvikling er i gang. Dette gøres ved at se på, hvor mange *story points* udviklingsholdet når at implementere i løbet af en iteration [Cohn, 2006: 181-182]. Fremgangsmåden til at udregne projektets varighed er således at dividere *velocity* med det samlede antal *story points* for projektet, for derved at få antallet af iterationer, der skal til for at gennemføre projektet. Denne værdi skal så ganges med varigheden af en iteration, og man har således et estimat for projektets varighed. Eksempelvis:

1. Alle *themes* (altså samlinger af *user stories*), der skal til for at udføre projektet, estimeres til samlet 400 *story points*.
2. Varigheden af hver iteration fastsættes til en måned.
3. *Velocity*, altså mængde af *story points* der kan være færdig efter hver iteration, estimeres til 25.
4. Herefter ser regnestykket ud som følger: 400 *story points*/ *velocity* 25 = 16 iterationer af en måneds varighed. Altså kan projektet estimeres til at vare 1 år og 4 måneder.

Fordelen ved denne tilgang er, at det er muligt at se meget tidligt i forløbet, om projektet tager længere end forventet. Hvis *velocity* er estimeret til tyve *story points*, men udviklingsholdet i løbet af en iteration kun når ti *story points*, skal der udregnes et nyt estimat for varigheden af projektet. Dette gør det muligt meget tidligt i forløbet at vurdere, om tidsplanen for projektet er realistisk med de givne ressourcer. Når der er udregnet et estimat for varigheden af projektet, kan dette indføres i en tidsplan, og rammerne for projektet er således fastsat.

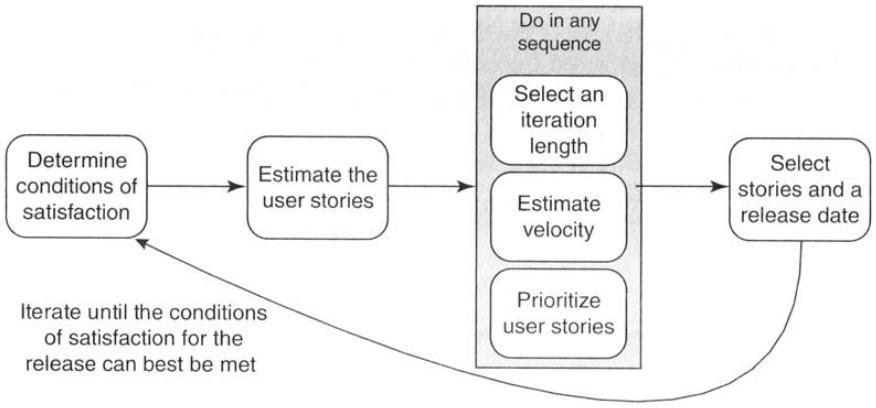
PLANLÆGNINGSPROCESSEN

Ifølge Cohn starter ethvert projekt med et formål. Det formål kunden opsætter for projektet, kan betragtes som værende succeskriteriet for projektet. Det vil sige, at hvis produktet opfylder dette formål, er kunden tilfreds [Cohn, 2006: 29]. Selve rammen for planlægningen, altså kundens tilfredshed, opsætter Cohn derfor i nedenstående model:



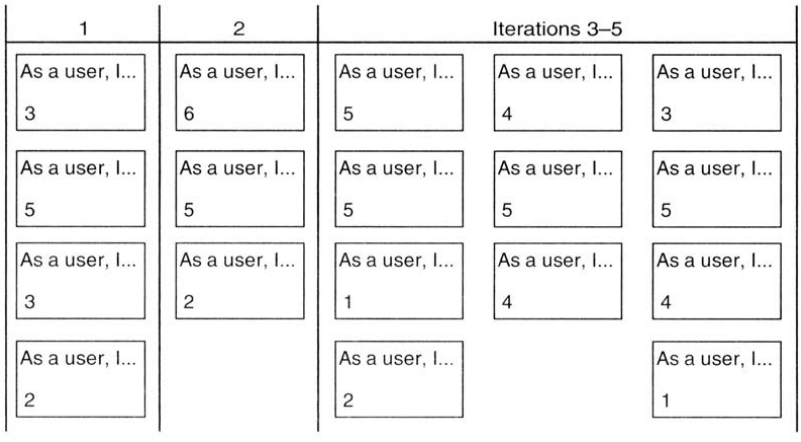
Figur 5.4: Cohns illustration af hvorledes succeskriterier driver processen [Cohn, 2006: 31].

Planlægningen starter med, at succeskriterierne for den næste delleverance (*release*) bliver forsøgt synliggjort i en diskussion mellem kunden og udviklingsholdet. Dernæst går man ind i en fase, hvor delleverancerne planlægges. Dette gøres ved, at man først sætter et estimat på hver *user storie*. Herefter vælges en iterationslængde, udviklingsholdets *velocity* bliver estimeret, og kunden foretager en prioritering af historierne. Dernæst tilknyttes hver *user storie* til en dato for en delleverance. Arbejdet med at planlægge delleverancer er opsat visuelt i nedenstående model:



Figur 5.5: Figuren illustrerer planlægningen af delleverancer [Cohn, 2006: 135].

Planlægningen af delleverancen er en iterativ proces, idet der arbejdes på at opfylde så mange af kundens succeskriterier som muligt. Resultatet er en plan, der beskriver hvilke *user stories*, der skal implementeres ved hver iteration. Et eksempel på, hvordan en plan over delleverancer kan se ud, er opsat i nedenstående figur:

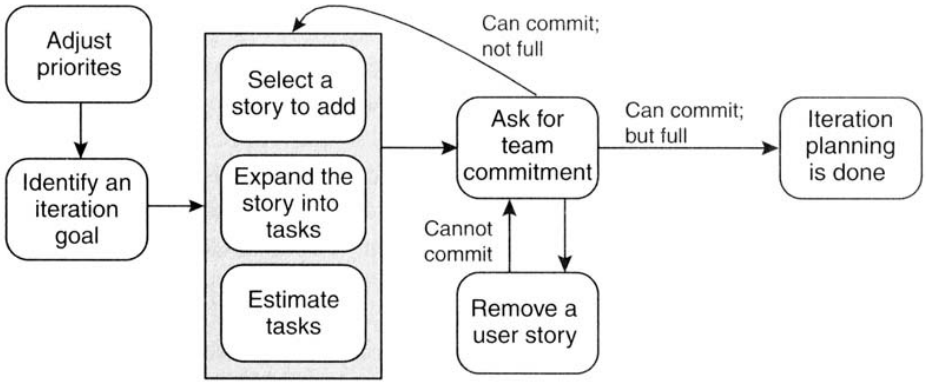


Figur 5.6: Plan over delleverancer [Cohn, 2006: 138].

Denne plan indeholder *user stories* og deres *story points*, opsat i kolonner alt efter hvilken iteration de knytter sig til. Fordelen ved planen er, at den kan laves rent fysisk, så kunden i

samarbejde med udviklingsholdet kan rykke rundt på de forskellige historier for at få det hele til at passe med den estimerede *velocity*. Ligeledes er det nemt at foretage ændringer, hvis *story points* bliver revideret eller kunden/brugerne opdager funktioner, der er vigtigere end de allerede fastsatte. Denne teknik minder meget om anvendelsen af et *production board* i filmproduktion. Begge teknikker foreslår en visuel fremstilling af 'arbejdsopgaver', da disse således kan give et overblik og nemt rykkes rundt, hvis det opstår et behov for at ændre planen.

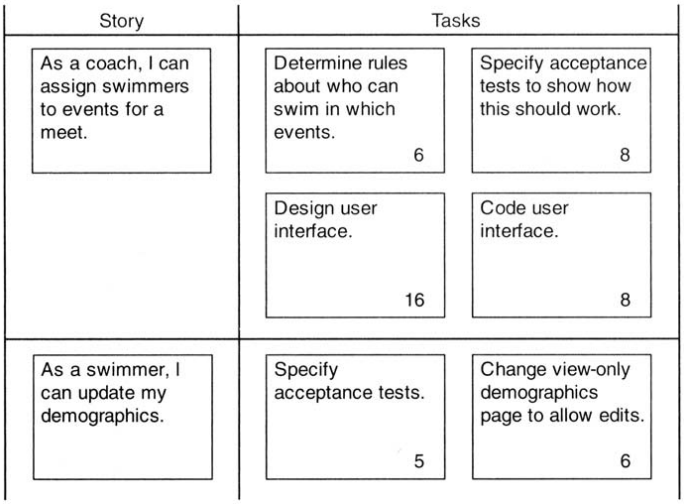
Når man har en plan for delleverancerne, går man videre til at fastlægge succeskriterierne for den næste iteration. Fasen hvor iterationen planlægges er opsat i nedenstående model:



Figur 5.7: Aktiviteterne i *commitment-driven* planlægning [Cohn, 2006: 159].

Ligesom ved planlægningen af delleverancerne er det første trin at definere kundens succeskriterier. Det vil sige, at prioriteringen af *user stories* bliver justeret og iterationens mål forsøges identificeret. Herefter starter en iterativ proces, hvor der udvælges *user stories*, de bliver nedbrudt til opgaver og til sidst estimeres opgaverne. Dernæst spørger man udviklingsholdet, om de vil forpligte sig til at aflevere de udvalgte *user stories* ved udgangen af iterationen. Hvis svaret er ja, tilføjes yderligere *user stories* indtil udviklingsholdet ikke længere mener, at de kan nå det i løbet af iterationen. Hvis udviklingsholdet ikke vil forpligte sig til de udvalgte historier, må der fjernes historier fra planen. Idéen med denne tilgang er, at udviklingsholdet ikke starter iterationen og dermed udviklingen før der ligger en plan, de er villige til at forpligte sig til at nå. Når udviklingsholdet har godkendt planen, har de dermed også forpligtet sig til at aflevere de udvalgte historier ved udgangen af den næste iteration.

Formålet med *iteration planning*-fasen er at udarbejde en iterationsplan, der indeholder en prioritering af historierne, en opdeling af historien i opgaver, samt en estimering af hver opgaves omfang. Et eksempel på en sådan er opsat på modstående side i figur 5.8:



Figur 5.8: Opsætning af iterationsplan [Cohn, 2006: 147]

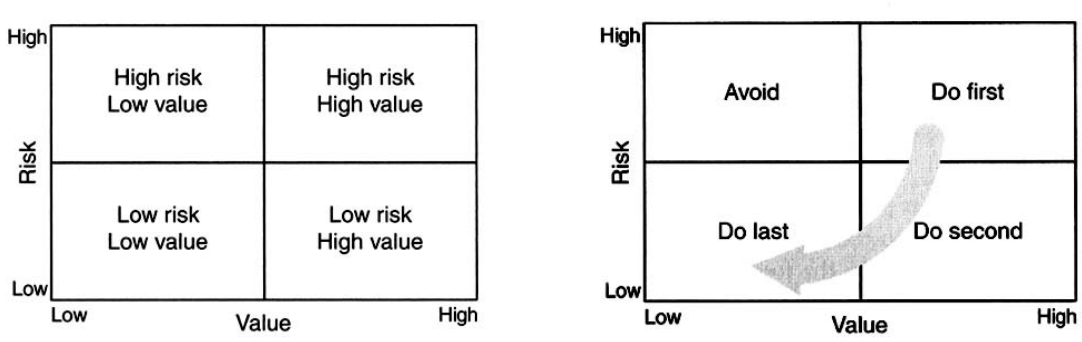
Når iterationen er planlagt, går selve udviklingen i gang. Hvor lang denne fase er, afhænger af hvilken iterationslængde kunden og udviklerne er blevet enige om. Efter udviklingsfasen forelægger der altså et funktionsdygtigt stykke software med de nye historier implementeret. Denne kan der så indhentes *feedback* på, ved de kommende brugere af systemet. Herefter starter planlægningsforløbet forfra, alt efter om det er en ny iteration eller en ny delleverance, der skal planlægges.

PRIORITERING EFTER *RISK* OG *VALUE*

Som hjælp til at prioritere *user stories* foreslår Cohn, at man anvender risikoanalyse kombineret med en analyse af historiernes værdi for brugerne. Cohn definerer en risiko som værende noget, der endnu ikke er sket, men som muligvis vil ske og derved sætte projektets succes over styr [Cohn, 2006: 84]. Han definerer tre typer af risici [Cohn, 2006: 84]:

- Tidsmæssige risici: Fare for at *deadlines* ikke kan overholdes.
- Omkostningsmæssige risici: Fare for at budgettet ikke kan overholdes.
- Funktionalitetsmæssige risici: Fare for at dele af systemet ikke vil fungere.

Spørgsmålet Cohn så opstiller er, hvorvidt man skal starte med historierne med en høj risici. Cohn svarer selv på spørgsmålet med et "både og" idet historierne ikke kun skal vurderes ud fra deres risici. Har en *user storie* en høj risici men en lav værdi for brugerne, skal man så vidt forsøge at undgå den [Cohn, 2006: 85]. Cohn opstiller dette perspektiv i en model, der er afbilledet i figur 5.9 på næste side:



Figur 5.9: Cohns illustration af hvorledes risici og værdi skal medregnes i prioriteringen af *user stories* [Cohn, 2006: 85].

Figuren til venstre afbilleder, hvad der kendetegner en historie i hver af de fire kategorier. Til højre er afbilledet Cohns syn på, hvorledes risici og værdi skal kombineres når *user stories* skal prioriteres. Det er dog vigtigt at bemærke, at risici og værdi ikke er konstante faktorer for en *user story*. De kan ændre sig i løbet af projektet, efterhånden som udviklerne opnår ny indsigt, og kunden og brugerne får mulighed for at afprøve systemet. En *user storie* kan derfor skifte fra en placering i eksempelvis *High risk – Low value* til et af de øvrige felter i løbet af projektet, hvorfor det hele tiden er vigtigt at opdatere prioriteringen af u*ser stories*.

USER STORIES

Cohn forsøger at bryde med den traditionelle idé om, at planlægning er en opdeling af projektet i aktiviteter [Cohn, 2006: 12]. Ulempen ved denne tilgang er ifølge Cohn, at kunden ikke får nogen værdi i færdiggørelsen af en aktivitet. Funktioner er derimod enheden, der er afgørende for, hvor stor værdi produktet har for kunden. Planlægningen skal derfor baseres på udarbejdelsen af funktioner, altså *user stories*, i stedet for aktiviteter.

En anden årsag til at traditionel planlægning ikke konsekvent medfører et produkt af høj værdi skyldes ifølge Cohn, at arbejdet i en traditionel plan ikke bliver prioriteret efter dens værdi for brugerne og kunden. Mange traditionelle planer er skabt i en formodning om, at alle de identificerede aktiviteter vil blive gennemført, og at prioriteringen og rækkefølge af arbejdsopgaverne derfor blot kan tilpasses udviklingsholdet [Cohn, 2006: 17].

FINANCIAL PRIORITIZATION

Det fremstår efterhånden tydeligt, at det er kunden og dennes tilfredshed, Cohn sætter som fokus for processen. Kunden skal ved udviklingen af it-systemet vurdere, hvilke *user stories* der har mest værdi. Med værdi menes der værdi for brugerne af systemer og/eller den økonomiske

værdi kunden har i projektet. Ifølge Cohn bliver udviklingen af it-systemer startet enten med et ønske om at generere et overskud eller for at reducere omkostninger [Cohn, 2006: 91]. Hvis man kan estimere, hvor stor økonomisk indflydelse en *user story* har, kan dette altså hjælpe, når man skal prioritere dem. Problemet ved dette i relation til spil er, at spil aldrig bliver udviklet med det formål at reducere omkostningerne, og at det kun genererer et overskud, hvis brugerne synes det er underholdende. Det må derfor netop være denne faktor, der kan hjælpe, når funktionerne i et spiludviklingsprojekt skal prioriteres. Hvis man overfører Cohns tankegang til spiludvikling, skal man altså have et estimat af, hvor meget en funktion tilføjer til spillets underholdningsværdi, så denne værdi kan hjælpe når funktionerne skal prioriteres. På den måde skaber man værdi for spilleren i produktet.

TIMEBOXING

Agile planlægning er bygget op omkring *timeboxing*. Hver iteration er tidsbestemt, og en delleverance skal afleveres på en fastsat dato [Cohn, 2006: 24]. Ligeledes er selve rammerne for det agile projekt nedsat ved hjælp af *timeboxing*. Cohn sammenligner det med at løbe et 10 kilometer løb. I traditionel udvikling vil du vide præcist hvor langt væk mållinien er, og dit mål er at nå den så hurtigt som muligt. I et agilt projekt ved man ikke med sikkerhed hvor mållinien er, men vi ved, at vi skal nå den til en bestemt dato [Cohn, 2006: 27]. Et agilt projekt er derfor snarere et tidsbestemt løb, frem for et løb med en fastsat distance. På den måde ved udviklingsholdet hvornår de skal aflevere produktet, men ikke hvad det er, de ender op med at aflevere.

Efter nu at have belyst hvordan det kan lade sig gøre at flette proces og planlægning sammen, vurderer jeg, at det er på tide, at jeg opsætter min egen metode.

FASE 6 – OPSTILLING AF SCRAME METODEN

I dette afsnit vil jeg opsætte en metode til udvikling af spil. Afsnittet kan opdeles i tre grupper: Krav til metoden – hvor jeg definerer hvilke krav min metode skal underbygge; SCRAME - hvor jeg beskriver processen og de mest fundamentale begreber; Og til sidst Planlægning – hvor jeg beskriver planlægningsprocessen i metoden.

KRAV TIL METODEN

For at kunne opsætte en metode til spil, ønsker jeg først at beskrive de krav, en sådan skal opfylde. Kravene er opsat på baggrund af den indsigt, jeg har fået i spiludvikling gennem projektet, min uddannelse og den praktiske erfaring jeg har erhvervet. Jeg definerer på baggrund af dette følgende syv kriterier:

1. Metoden skal opbygges med en iterativ struktur.
2. Sikre underholdningsværdien.
3. Tilbyde en planlægningsproces, der understøtter den iterative struktur.
4. Tilbyde en proces til estimering, der understøtter den iterative struktur.
5. Synliggøre og reducere risici.
6. Kommunikerer hensigten med produktet såvel internt som eksternt.
7. Integrerer en testfase.

Kriterierne beskrives yderligere i det efterfølgende.

OPBYGGES MED EN ITERATIV STRUKTUR

Det primære formål med spil er at underholde. Altså skal man igennem processen sikre, at der er fokus på at tilføje så meget underholdningsværdi i slutproduktet som muligt. Den bedste måde at gøre dette er gennem en iterativ proces, hvor hver enkelt del af spillet hele tiden kan vurderes og testes. De agile metoder egner sig til dette, da de netop er opbygget med en iterativ struktur for at tage højde for, at projektet kan ændre sig undervejs. Det samme er gældende i spiludvikling, så

de agile metoder virker som et godt fundament til opsætningen af en metode til spiludvikling. En anden fordel ved at vælge en iterativ tilgang er, at man udnytter fordelene ved, at udviklingsholdet opnår ny indsigt undervejs i projektet - både om produktet, teknologien og deres egne evner. Dermed er man også med til at reducere risikoen for at udvikle et produkt, der ikke er underholdende, og man får fordelen af, at udviklerne opnår en ny viden. Den iterative struktur udnytter dette ved at alle idéer, der opstår undervejs i processen, kan integreres som en del af slutproduktet.

SIKRE UNDERHOLDNINGSVÆRDIEN

Processen skal, som allerede beskrevet, sikre at spillet bliver underholdende. Den iterative tilgang er den bedste måde at sikre et underholdende spil, da man således hele tiden kan revurdere spillets *gameplay*. Men før underholdningsværdien kan sikres, skal den først defineres. Underholdningsværdi betragter jeg som slutbrugerens opfattelse af den mængde underholdning, vedkommende har fået ved at købe spillet. Jeg mener ikke, at dette nødvendigvis kan måles kvantitativt, da det er en subjektiv fornemmelse alt afhængig af hvilken form for underholdning, der tiltaler køberen. Pointen er, at underholdningsværdien afgøres af spilleren, hvilket gør spilleren til en vigtig del af processen. Det samme er eksempelvis tilfældet med IT-systemer – værdien af systemet afhænger af de funktioner det tilbyder, og om de matcher de forventninger brugerne har.

Jeg mener ikke, at underholdningsværdien udelukkende er afhængig af spillets *gameplay* – altså de regler der opsættes af spillet. Ofte er æstetikken eller et narrativt element med til at gøre en spilmekanik underholdende, og det er et væsentligt element i at muliggøre en indlevelse i *game universe*. Det, det hele afhænger af, er derfor slutbrugerens oplevelse af underholdningsværdien i produktet. Et krav til metoden er derfor, at den inddrager brugerne i processen.

TILBYDE EN PLANLÆGNINGSPROCES DER UNDERSTØTTER DEN ITERATIVE STRUKTUR.

Hvis du arbejder iterativt, medfører det en væsentlig forøget kompleksitet i planlægningen. Hvis et delsystem ikke er beskrevet og defineret, hvordan kan du så planlægge og estimere arbejdsopgaverne, der påkræves for at færdiggøre systemet? En spiludviklingsmodel skal derfor ligeledes tilbyde en planlægningsproces, der er tilpasset til den iterative struktur. Spillitteraturen var ingen hjælp på dette punkt, så her har jeg indhentet inspiration fra de agile metoder. Planlægningen kan opdeles i tre niveauer – rammerne for projektet (planlægningen af

delleverancer), rammerne for hver iteration og rammerne for hver arbejdsdag. Denne tankegang er primært baseret på Cohns Planning Onion [Cohn, 2006:28]. Jeg vil i min opsætning af metoden forsøge at dække alle tre niveauer, ved at tilbyde en proces, der tager højde for, at planlægning foregår på disse tre niveauer. Det er ligeledes mit mål, at planlægningsprocessen skal fokusere på at få så meget værdi ind i spillet som muligt, ved at sætte underholdningsværdien i centrum.

TILBYDE EN PROCES TIL ESTIMERING DER UNDERSTØTTER DEN ITERATIVE STRUKTUR.

Ifølge Krogh oplever spilbranchen problemer med at estimere. Årsagen til dette ligger i den manglende fælles forståelse for produktet imellem kunden og spiludvikleren. Når spillet ikke er veldefineret og hele tiden kan ændres, er det svært at estimere, hvor lang tid projektet vil tage. Konsekvensen er at budgettet overskrides, fordi udviklerne har lovet mere, end de kan overkomme indenfor det givne budget. En metode til spiludvikling skal altså tilbyde en planlægningsproces til rammerne for projektet, der gør det muligt at estimere hvor lang tid projektet forventes at tage, og synliggøre dette på en sådan måde, at budgettet kan overholdes. Processen skal altså synliggøre overfor kunden, hvornår en ændring vil medføre en overskridelse af budgettet eller hvilke andre elementer, det vil være påkrævet at fjerne fra planen, for også at nå den ønskede ændring.

SYNLIGGØRE OG REDUCERE RISICI

Planlægningsprocessen, der skal integreres i metoden, skal give et indblik i de risici, der omgiver projektet. Dette skal være med til at sikre projektets succes. Den bedste måde at synliggøre risici er i estimeringen af arbejdsopgaver. Hvis personen, der skal estimere opgaven, finder det meget vanskeligt eller slet ikke mener, at det er muligt, før vedkommende har sat sig mere ind i eksempelvis teknologien, udgør denne opgave en væsentlig risiko for projektet, men den er herved blevet synliggjort og kan derfor håndteres. Cohn opsætter en prioritering efter værdi og risici, der vurderes anvendelig i denne sammenhæng (Figur 5.9 side 81). Arbejdsopgaver, der har en høj grad af usikkerhed og tilføjer en høj grad af værdi til produktet, skal prioriteres først. Metoden skal altså tage højde for risici ved at tilbyde en planlægningsproces, der synliggør disse, og en prioritering der tillader, at risikofyldte arbejdsopgaver har første prioritet. Ligeledes minimeres risikoen for ineffektivitet. Da der er tale om en iterativ proces, er det største problem, med henblik på effektiviteten, at arbejdsopgaver først bliver fastlagt undervejs. Ved at have en

prioriteret liste over arbejdsopgaver, kan den næste opgave altid påbegyndes, hvis man oplever problemer eller er færdig med den nuværende.

KOMMUNIKERE HENSIGTEN MED PRODUKTET SÅVEL INTERNT SOM EKSTERNT

Ifølge de problematikker Krogh opsætter (Jævnfør side 32), har spilbranchen problemer med at justere forventningerne til produktet både internt og eksternt. Det er altså væsentligt at processen gør det muligt at dele information internt i virksomheden, men også eksternt til eksempelvis kunden. Ved udviklingen af et IT-system vil det ofte kun være forventningerne til den funktionelle side af systemet, der er vigtigt at kommunikere, da det netop er systemets funktionelle værdi, der afgør systemets succes. Ved spiludvikling er der mange flere bolde i luften, idet der skal foretages forventningsjustering både til *gameplay*, æstetisk udformning (såvel grafisk som lydmæssigt) og selve interaktionen. Der er derfor vigtigt, at en metode til udvikling af spil kommunikerer forventningerne til disse elementer. Den bedste måde at kommunikere dette på er ved at visualisere de forskellige elementer, så disse kan afprøves, vurderes og justeres. Processen skal altså sætte fokus på at visualisere disse elementer af spillet så tidligt i processen som muligt.

INTEGRERE EN TESTFASE

Som Krogh beskriver (jævnfør side 34), oplever spilbranchen problemer med at integrere testarbejdet i processen. Metoden skal altså integrere en testfase, der beskriver hvornår der skal testes, og hvad der skal testes. Endvidere skal der holdes styr på testresultaterne, og testarbejdet skal hjælpe med at sikre spillets underholdningsværdi. Den iterative *incrementielle* tilgang virker egnet til dette, da man efter hver iteration vil stå med et system, der kan afvikles og dermed også testes. At visualisere systemet så tidligt i processen som muligt har altså ikke kun det formål at kommunikere hensigten med spillet, men også at gøre det muligt at teste underholdnings-værdien i spilmekanikker.

FORUDSÆTNINGER

I opsætningen af metoden har jeg været i tvivl om, hvorledes jeg skal anvende de allerede eksisterende begreber og termer, og overføre disse til spiludvikling. Det er dog tydeligt, efter at have læst en efterhånden ret omfattende mængde spillitteratur, at anvendelsen af de samme

termer kun medfører forvirring. Skabelsen af spil er hverken det samme som at lave film eller IT-systemer. Derfor skal der oprettes et nyt begrebsapparat. De teknikker og metoder der derfor ikke bærer nøjagtig den samme betydning, skal navngives så de passer ind i min metode til spiludvikling.

I opsætningen af metoden involverer jeg ikke de psykologiske aspekter. Det vil sige, hvordan man leder en gruppe, motiverer, ansvarliggør, støtter opbygningen af sociale relationer og så videre. Årsagen til dette skyldes den betydelige forøgelse i metodens omfang, hvis også disse overvejelser skal beskrives. Jeg inddrager dem dog i begrænset omfang, da de agile metoder bygger på et fundament, hvor man forsøger at ansvarliggøre den enkelte medarbejder. De tilbyder dog ingen konkrete teknikker til dette, og det virker derfor som en uoverskuelig opgave også at tilbyde psykosociale redskaber indenfor projektets omfang. Hvis man ønsker yderligere information om det psykologiske aspekt foreslår jeg Edgar Scheins *Organizational Psychology* for en introduktion til dette felt [Schein, 1994].

I beskrivelsen af metoden vil jeg ikke komme ind på, hvordan man får selve ordren på spilprojektet. Udgangspunktet for metoden er således, at der forelægger en kontakt og en vision for projektet. Jeg vil dog anbefale, at man, inden selve udviklingen går i gang, starter med en basal prototype – om end ikke andet så bare i skitseform, som rollespil eller lignende. Målet med metoden er at tilbyde en proces, hvorigennem man kan udvikle spil – hverken mere eller mindre.

Jeg har valgt at kalde metoden SCRAME på grund af rødderne til SCRUM metoden, og da den fonetiske udtale lægger sig op af det engelske *game.*

SCRAME

Før jeg kan beskrive selve SCRAME processen definerer jeg først rollerne i processen og nogle af de væsentligste begreber. SCRAME processen bygger på følgende roller:

- *ScrameMaster*: Det er *ScameMasterens* opgave at sikre, at processen bliver fulgt og undervise holdet i metodens processer og teknikker. Derved er det også vedkommendes opgave at have et indgående kendskab til metoden, og de teknikker der tilbydes. De vigtigste egenskaber *ScrameMaster* skal have er evnen til at lære fra sig, og et indgående kendskab til metoden og de teknikker denne tilbyder. En viden om de agile udviklingsmetoder er en fordel.
- *Gamer Deputy*: Fungerer i processen som repræsentant for spilleren. Det er denne person, der leder udførelsen af spiltest arbejdet, samler resultaterne og giver feedback videre.

Vedkommendes opgave er at vide eller undersøge, hvad spillerne finder underholdende, og hvordan spiloplevelsen kan forbedres. Den vigtigste egenskab *Gamer Deputy* skal have er evnen til at sætte sig i spillerens sted og forståelse for, hvordan balanceringen påvirker spillets *gameplay*.

- *Game Owner*: Dette er den eller de personer, der finansierer spillet, og som er økonomisk afhængig af projektets succes. *Game Owner* skal i processen definere målene, og være i stand til at prioritere de elementer spillet skal indeholde. Det er altid *Game Owner*, der har det sidste ord. De vigtigste egenskaber *Game Owner* skal have er finansielt overblik og en klar idé om, hvad formålet med spillet er.
- *Game Producer*: Skal sikre al ekstern kommunikation, forhandling af kontrakter, løse licens problemer og definere rammerne for projektet. Det vil sige, fastsætte det budget og de tidsrammer projektet er underlagt. Produceren overvåger spiludviklingen og sikrer, at alle dele er med. *Game produceren* skal sikre, at alle planer er opdateret og oplyse *Game Owner* om eventuelle påvirkninger af budgettet. *Game Producer* skal være i stand til at holde overblik over planlægningen samt kommunikere klart og tydeligt både internt og eksternt i forhold til gruppen.
- Spildesigner: Spildesigneren skal indenfor rammerne af visionen være kreativ og designe et *gameplay*, der stemmer overens med spillets formål og mål. Det er vedkommendes opgave at komme med idéer til, hvordan man kan skabe eller forbedre spiloplevelsen. Spildesigneren skal derfor være en kreativ person, der kan bevare et klart og tydeligt billede af, hvordan spillets *gameplay* er opsat og hænger sammen.

Ovenstående er ikke en udtømmende liste over rollerne i udviklingsprocessen men blot de roller, processen er bygget op omkring. Bemærk at *Game Owner* kan være repræsenteret ved en intern projektleder, der enten har tæt kontakt eller kendskab til kundens ønsker til spillet. Valget af denne rolle afhænger af projektets omfang, og i hvor høj grad kunden vil acceptere at være en del af processen. Ligeledes kan flere af rollerne kombineres; igen afhængigt af projektets størrelse. Det er spildesigneren, produceren, *Gamer Deputy* og *Game Owner*, der udgør beslutningsdelen i processen. Det er dog altid *Game Owner* eller en repræsentant for denne, der har det sidste ord.

GAMER STORY

Den bedste måde at forklare hvad en *gamer story* er, og hvordan processen er bygget op omkring dem, er ved hjælp af et eksempel. Forestil dig, at du går ind i et supermarked med tusind kroner i lommen. På hylderne ligger der varer, der alle har et prisskilt. Du skal nu købe det, du helst vil

have, og som du får mest ud af, men du må ikke overskride dit budget. I SCRAME processen er hver vare, der ligger i butikken, en *gamer story* og dit budget på tusind kroner er den tid, du har til rådighed. Det er *Game Owner*, der sammen med spildesigneren, produceren og *gamer deputy*, går rundt i butikken og udvælger de varer, de mener, der har mest værdi for spilleren. De skal dog sikre sig, at varerne i deres kurv ikke overskrider deres budget. *Gamer story* anvendes i SCRAME metoden som betegnelsen for en vare i ovenstående eksempel. En *gamer story* er altså en beskrivelse af en oplevelse spilleren kan få i spillet. Også her er den bedste fremgangsmåde eksempler, og jeg har derfor opsat et par stykker i nedenstående:

- Spilleren kan åbne en dør ved at trykke på en kontakt.
- Spilleren kan, hvis han har samlet en faldskærm op, springe sikkert ned fra store højder, hvis han når at folde faldskærmen ud.
- Spilleren kan justere sværhedsgraden af modstanderne, inden spillet starter.

Hver af disse *gamer stories* har et estimat, der altså dermed er historiens prisskilt. Dette estimat kaldes også *story points*, men mere om det senere.

Når der skal brainstormes over mulige *gamer stories* i spillet, kan de opdeles i tre grupper, for at øge overskueligheden:

- *Gamer stories* der relaterer sig til, hvad spilleren kan gøre før spillet begynder. (Eksempelvis: Som spiller ønsker jeg at kunne justere modstandernes styrke).
- *Gamer stories* der relaterer sig til, hvad spilleren kan gøre, mens han er i spillet. (Eksempelvis: Som spiller kan jeg se en visuel indikation af, hvor meget energi jeg har tilbage).
- *Gamer stories* der relaterer sig til, hvad spilleren kan gøre når en spilsession (eller en bane) er gennemført. (Eksempelvis i et bilspil: Som spiller ønsker jeg en visuel repræsentation af hvilke tider mine konkurrenter sluttede på).

Det er selvfølgelig muligt at foretage en yderligere opdeling, men det er meget afhængigt af hvilken type af spil der er tale om, og det er derfor ikke noget, jeg vil beskrive yderligere her.

Gamer Stories skal udskrives, så man under planlægningsfasen kan få en visuel og fysisk repræsentation af 'varen'. Lidt ligesom du nede i dit supermarked kan tage varen op, vende og dreje den og lægge den tilbage igen, hvis du alligevel ikke vil have den. *Gamer Story* skal opsættes som afbilledet på modstående side:

ID: *Gamer Storiens* identifikationsnummer. **Navn:** En kort beskrivelsen af historien.			
Beskrivelse	***Mechanics***	***Assets***	***Story Points***
En komplet beskrivelse af historien.	De programmerings-mæssige opgaver historien kræver.	De *assets* der hører til historien. Det vil sige lyd, grafik, animationer og så videre.	Et relativt estimat for hvor lang tid opgaven tager.

Tabel 6.1: Opsætningen af en gamer story (Egen tilvirkning).

Et eksempel vil altså se ud som følger:

ID: 1 **Navn:** Spilleren kan hoppe			
Beskrivelse	***Mechanics***	***Assets***	***Story Points***
Som spiller kan jeg, ved at trykke *space,* hoppe op på mindre genstande.	Tilføj hop-bevægelsen til spillerkarakterens bevægelser.	Hop animation. Lyd når han hopper. Lyd når han rammer jorden igen.	5

Tabel 6.2: Eksempel på en gamer story (Egen tilvirkning).

For at holde styr på *gamer stories* anvendes en *backlog*, der er en prioriteret liste over *gamer stories*. Hvis jeg skal relatere det til eksemplet i supermarkedet, vil det svare til, at du laver en liste over de varer, du gerne vil have, hvor den vare, du helst vil have, står øverst på listen. *Backlogs* er forklaret nærmere i det efterfølgende afsnit.

BACKLOG

I SCRAME-processen anvendes der to typer af *backlogs*; *Game Backlog* og *Sprint Backlog*. *Sprint Backlog* indeholder en liste over *gamer stories* og en opdeling af disse i opgaver. *Sprint Backloggen* er et arbejdsredskab til planlægning af udviklingsfasen, og vil derfor blive beskrevet i et senere afsnit ('Planlægning af *sprint*' side 97). *Game Backlog* er derimod SCRAME processens rygrad. Det er en prioriteret liste af *gamer stories*, der opdateres og vedligeholdes af *Game Owner*. Et simpelt eksempel på, hvordan en *Game Backlog* kan se ud, er opsat i tabellen på næste side:

Prioritet	ID	Navn	Estimat	Oprettet af
Meget høj				
	1	*Sound adjustment*	8	HW
	2	*Trees for jungle theme*	5	KN
Høj				
	-	*Player controls*		
	3	*Jump*	8	FO
	4	*Strafe*	3	KM & SB
	5	*Save high score*	5	SK
Lav				
	-	*Player weapons:*		
	6	*Shotgun*	2	TN

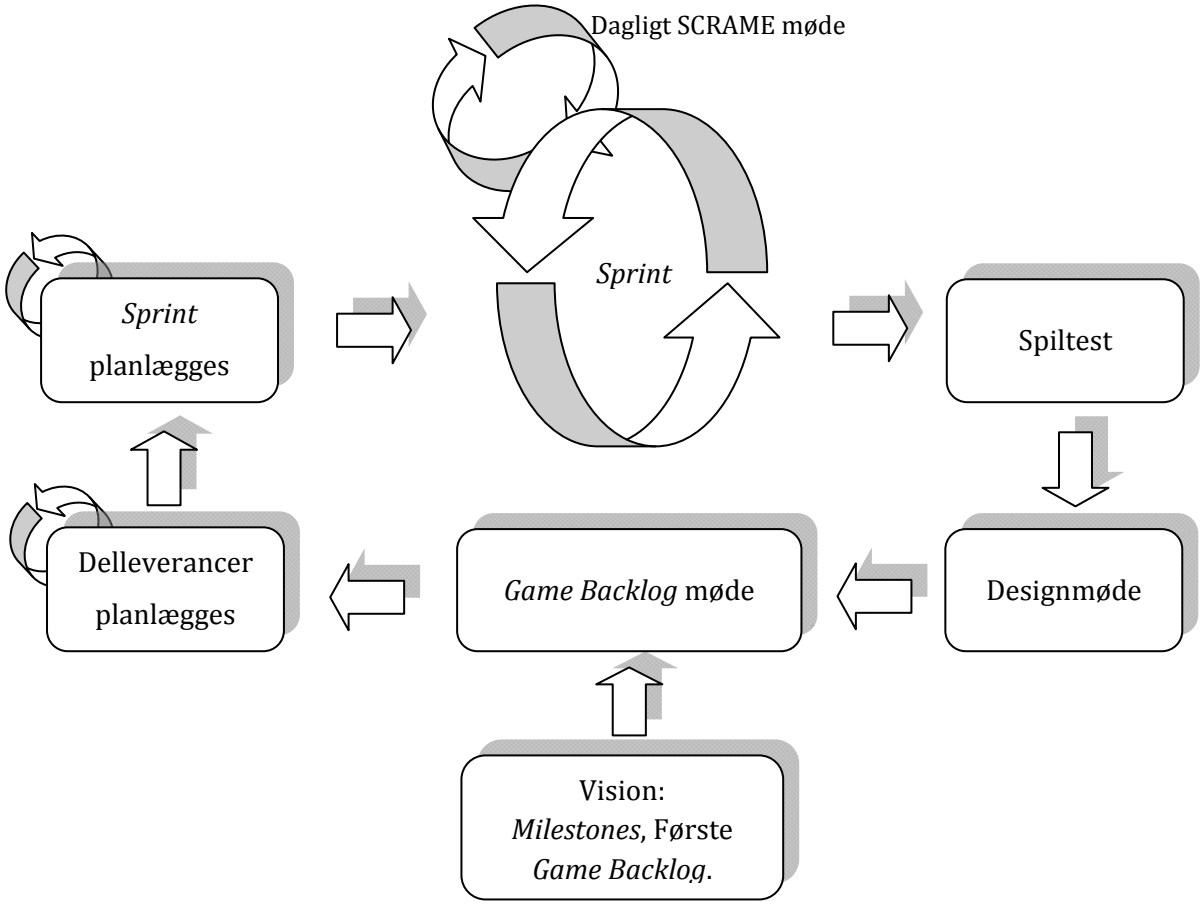
Tabel 6.3: Tabellen er et eksempel på hvordan en *Game Backlog* kan se ud (Egen tilvirkning).

Game Backlog er altså en prioriteret liste af *gamer stories*, der giver et overblik over, hvad spillet skal indeholde. Som det ses i eksemplet kan man gruppere *stories* i *themes*, der er en samling af *gamer stories*, der alle relaterer sig til et specifikt element i spillet. I eksemplet bliver *Player controls* og *Player weapons* anvendt som *themes*. *Game Backlog* er altså selve SCRAME-processens rygrad, da den beskriver hvad spillet skal indeholde og i hvilken rækkefølge, det skal udarbejdes.

Nu, hvor jeg har gennemgået de mest grundlæggende begreber bag SCRAME metoden, kan jeg præsentere selve processen. Hverken *gamer stories* eller *backloggen* er noget, du bare fastsætter og aldrig vender tilbage til. Det er dynamiske dokumenter, der hele tiden skal opdateres. Hvornår og hvordan disse skal opdateres og vedligeholdes beskrives i det efterfølgende afsnit.

PROCESSEN

Opbygningen af processen i SCRAME metoden bygger på en modificeret SCRUM proces. Det betyder også, at SCRAME er bygget op omkring de agile principper. SCRAME processen er opsat i figur 6.4 på modstående side:



Figur 6.4: Overblik over SCRAME processen (Egen tilvirkning).

Først fastlægges projektets vision, og der udarbejdes en *Game Backlog*, der beskriver de muligheder spilleren har i spillet. Herefter afholdes et møde hvor holdet, der skal skabe spillet, har mulighed for at stille spørgsmål til *backloggen*. Formålet er at give holdet mulighed for at stille uddybende spørgsmål, hvis de mangler at få synliggjort dele af spillet eller har forslag til forbedringer. Herefter går selve planlægningsforløbet i gang. Først opdeles *backloggen* i delleverancer, hvorefter man kan påbegynde arbejdet med at planlægge den næste *sprint*. *Sprint* er de agile metoders term for selve udviklingsfasen, der påbegyndes når planlægningen er ovre. En *sprint*-fase varer normalt et sted mellem 2-4 uger. Er man i starten af udviklingen, hvor man stadig er på prototype niveau, foreslår jeg, at man holder *sprint*-fasen på bare én uge. Hver arbejdsdag i løbet af *sprinten* afholdes et SCRAME møde – gerne som et stå-op-møde hver morgen. Her skal hvert medlem af holdet besvare følgende tre spørgsmål:

- Hvad har jeg lavet siden det sidste SCRAME møde?
- Har jeg nogen problemer, eller kan jeg se noget, der muligvis kan hindre mit arbejde?
- Hvad har jeg færdigt til det næste SCRAME møde?

Når *sprinten* er afviklet er resultatet en videreudvikling af det spil, man havde før *sprinten* gik i gang. Der fortages herefter en spiltest for at undersøge, hvilken effekt de nye muligheder har på spillerens oplevelse af spillet. Denne *feedback* opsamles og anvendes til det næste designmøde, hvor *Game Owner*, produceren, spildesigneren og *Gamer Deputy* sammensætter en opdateret *Game Backlog*. Herefter afholdes et *Game Backlog* møde, hvor den opdaterede *backlog* præsenteres, og holdet igen har mulighed for at stille spørgsmål og komme med forslag inden den næste iteration påbegyndes.

Udgangspunktet for selve processen er visionen for projektet. Hvordan denne udarbejdes, og hvad den skal indeholde, vil jeg behandle i det efterfølgende afsnit.

FASTSÆT PROJEKTETS VISION

Før selve udviklingen af spillet går i gang, bør man nedskrive visionen for spillet. Dette behøver blot være få sætninger, der beskriver spillet, men det er et væsentligt værktøj til at sikre, at alle arbejder mod ét fælles mål og har en fælles forståelse for, hvordan spillet skal underholde spilleren. Nedenstående er en opsætning af de spørgsmål man kan stille sig selv, i et forsøg på at definere projektets vision. Jan Krag Jacobsens bog "25 spørgsmål – en moderne retorik til planlægning af kommunikation" er anvendt som inspirationskilde [Jacobsen, 1997].

For at hjælpe med at definere visionen kan man besvare nedenstående spørgsmål:

1. Hvem skal spillet underholde?
2. Hvor befinder spilleren sig i brugssituationen og er der andre tilstede?
3. Hvilken spilgenre skal spillet følge?
4. Skal der anvendes en filmisk genre og i så fald hvilken?
5. Hvilken platform skal spillet designes til?
6. Hvilken kameravinkel skal anvendes?
7. Hvilke interaktionsmuligheder skal spilleren have?
    a. Hvilke *controls* skal spilleren anvende?
8. Skal der være en historie i spillet?
    a. Hvis ja, hvordan skal denne formidles til spilleren?
9. Hvordan skal læringskurven for spillet være?
    a. Hvordan skal spilleren introduceres for elementerne i spillet?
10. Hvilke typer af forhindringer skal spilleren forcere?
    a. Hvordan skal det være underholdende at forcere disse?
11. Hvilke andre spil findes der på markedet der minder om dette spil?

    a. Hvordan skal spillet adskille sig fra de eksisterende spil?
12. Hvilken æstetisk stil skal spillet følge?

Ved at besvare disse spørgsmål skabes et udgangspunkt for den første *backlog*, og man synliggør for teamet hvad hensigten er med spillet. Når man skal fastsætte den målgruppe, man ønsker at underholde, kan det i en større virksomhed være en fordel at inddrage salgs- og marketingsafdelingen, da en sådan må have et begreb om, hvem der kan tænkes at købe det beskrevne spil og således udgøre spillets målgruppe.

PLANLÆGNING

Før jeg kan beskrive planlægningen i SCRAME metoden vil jeg først gennemgå to begreber, der er vigtige for forståelsen af planlægningsprocessen – *Story points* og *Velocity*.

STORY POINTS

Til estimeringen af *gamer stories* anvendes relative værdier. Det vil sige, at der ikke fastsættes et bestemt antal timer eller dage til at færdiggøre en *gamer story*, men at man i stedet bruger værdier, der beskriver opgavernes relative omfang. Det vil sige *gamer stories* relative størrelse i forhold til hinanden. Et eksempel på et andet sted hvor der anvendes relative værdier opdager du, hvis du bestiller en cola på en restaurant. På en restaurant bestiller du ikke eksempelvis en 33 cl cola, men en cola, der enten er lille, almindelig eller stor. Disse værdier, altså lille, almindelig og stor, beskriver colaens relative størrelse. Den store cola er større end den almindelige, mens en lille cola er mindre end både den almindelige og den store. På den måde anvendes *story points* i SCRAME til at beskrive omfanget af en *gamer story*. Jeg foreslår anvendelsen af Fibonacci talrækken fra 1 til 89. Det vil sige at *story points* gives ud fra værdierne 1, 2, 3, 5, 8, 13, 21, 34, 55 og 89. Årsagen til at denne talrække findes anvendelig skyldes at afstanden fra værdi til værdi stiger. Da det oftest er de største opgaver, der er sværest at estimere, giver det mening, at der er færre værdier at vælge imellem. En *gamer story* med 1 *story point* er således den mindste opgave i projektet, mens en *gamer story* med 89 point er den største. Fordelen ved at anvende relative størrelser i estimeringen er muligheden for at beregne og anvende *velocity* i planlægningen.

Velocity

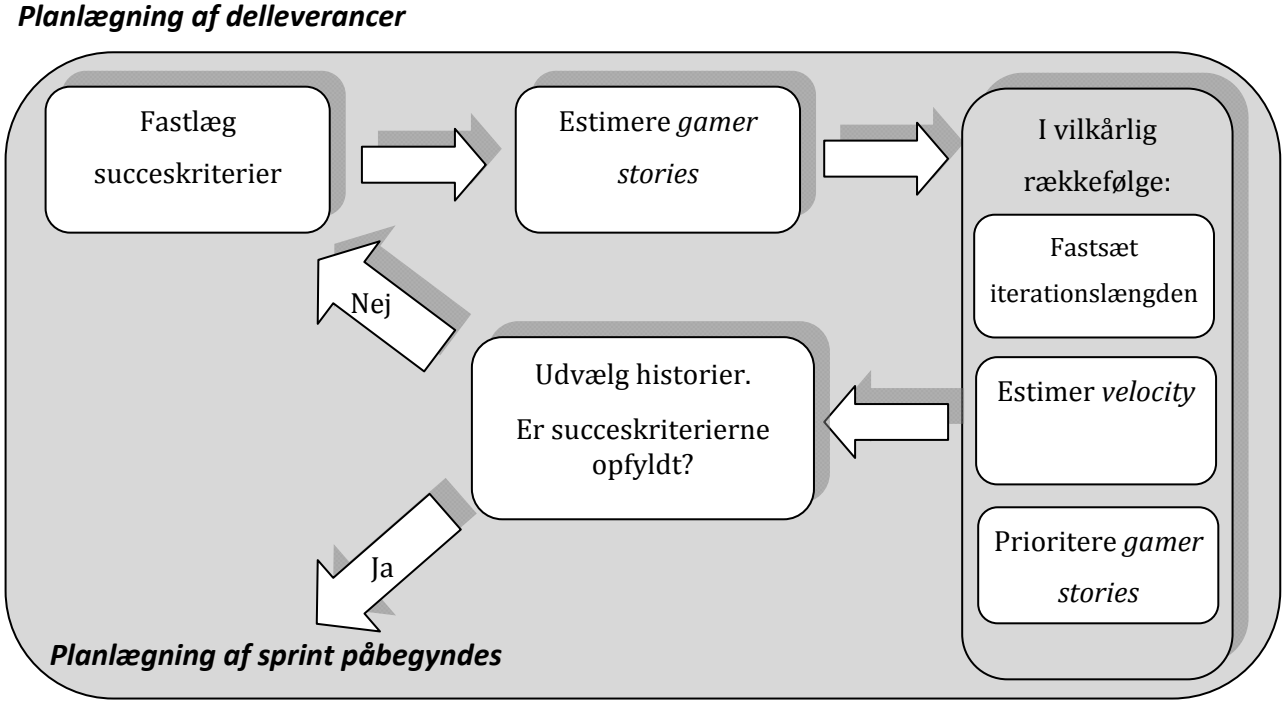
Velocity anvendes til at beskrive holdets samlede arbejdskraft. Det vil sige et begreb, der anvendes til at beskrive det samlede antal *story points*, holdet kan færdiggøre i løbet af et *sprint*.

Velocity kan, hvis holdet har arbejdet sammen før, fastsættes ved hjælp af et estimat. Er det første gang holdet arbejder sammen, kan *velocity* først måles, når den første *sprint* er gennemført. Værdien udregnes ved at ligge alle *story points,* der er færdiggjort under *sprinten*, sammen. *Velocity* kan herefter anvendes til at estimere, hvor meget arbejde holdet kan nå i løbet af et sprint, og til at vurdere hvor mange *sprints* der skal til, før spillet er færdigt. *Velocity* skal løbende igennem processen opdateres. Hvis *velocity* for to *sprints* er henholdsvis 20 og 30, må det formodes, at *velocity* for den efterfølgende *sprint* er omkring 25.

Nu hvor *story points* og *velocity* er forklaret, kan vi bevæge os videre til planlægningsfasen. Planlægningen er, som det ses af Figur 6.4, delt i to faser; planlægning af delleverancer og planlægning af sprint. I det efterfølgende vil de to planlægningsfaser og deres relation til resten af processen blive beskrevet.

PLANLÆGNING AF DELLEVERANCER

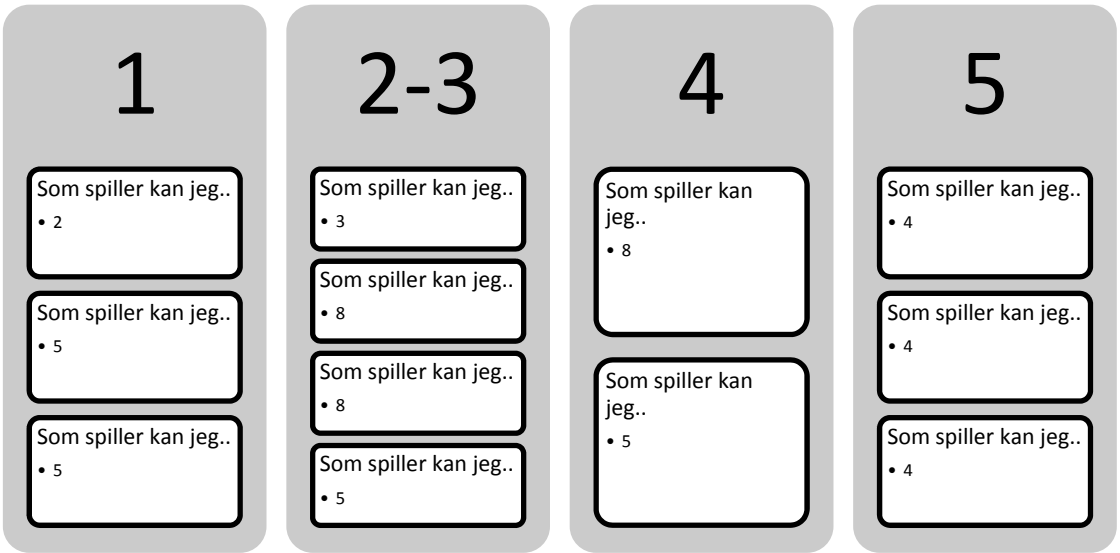
Det primære formål med planlægningen af delleverancer er at opsætte en plan, der beskriver hvilke *gamer stories,* der forventes færdig ved de enkelte delleverancer. Processen til udarbejdelsen af denne ser ud som afbilledet i nedenstående figur:



Figur 6.5: Planlægning af delleverancer (Egen tilvirkning).

Først fastlægges *Game Owners* succeskriterier for delleverancerne. Dette vil formodentlig være identisk med visionen, idet det er denne, der beskriver de vigtigste elementer, og det oftest er disse, der skal realiseres som det første. Når succeskriterierne for planen over delleverancer er fastsat, skal *gamer stories* estimeres. En teknik til dette er beskrevet i Bilag 7 – *Planning Poker*. Når hver *gamer story* har fået tilknyttet et *story point*, skal iterationslængden fastsættes, *velocity* skal estimeres og *gamer stories* skal prioriteres. Iterationslængden kan variere fra en til fire uger – en iterationslængde kortere end dette vil ikke give holdet den tilstrækkelige arbejdsro, mens en længere periode vil betyde at planerne opdateres for sjældent. *Velocity* – altså det antal *story points* holdet kan nå i løbet af *sprinten* – skal estimeres, så der kan udvælges *gamer stories* til hver iteration. Ligeledes skal *gamer stories* prioriteres, så man synliggør *Game Owners* prioriteter. Herefter udvælges *gamer stories* til hver iteration. Hvis succeskriterierne er opfyldt, kan man gå videre med planlægningen af *sprinten*. Er de ikke opfyldt, starter processen forfra og det kan være nødvendigt at revurdere forventningerne til planen indtil *Game Owner* er tilfreds.

Når fasen er gennemført er resultatet en plan over delleverancer. Et eksempel på hvordan en sådan kan se ud, er opsat i nedenstående figur:



Figur 6.6: Plan over delleverancer (Egen tilvirkning).

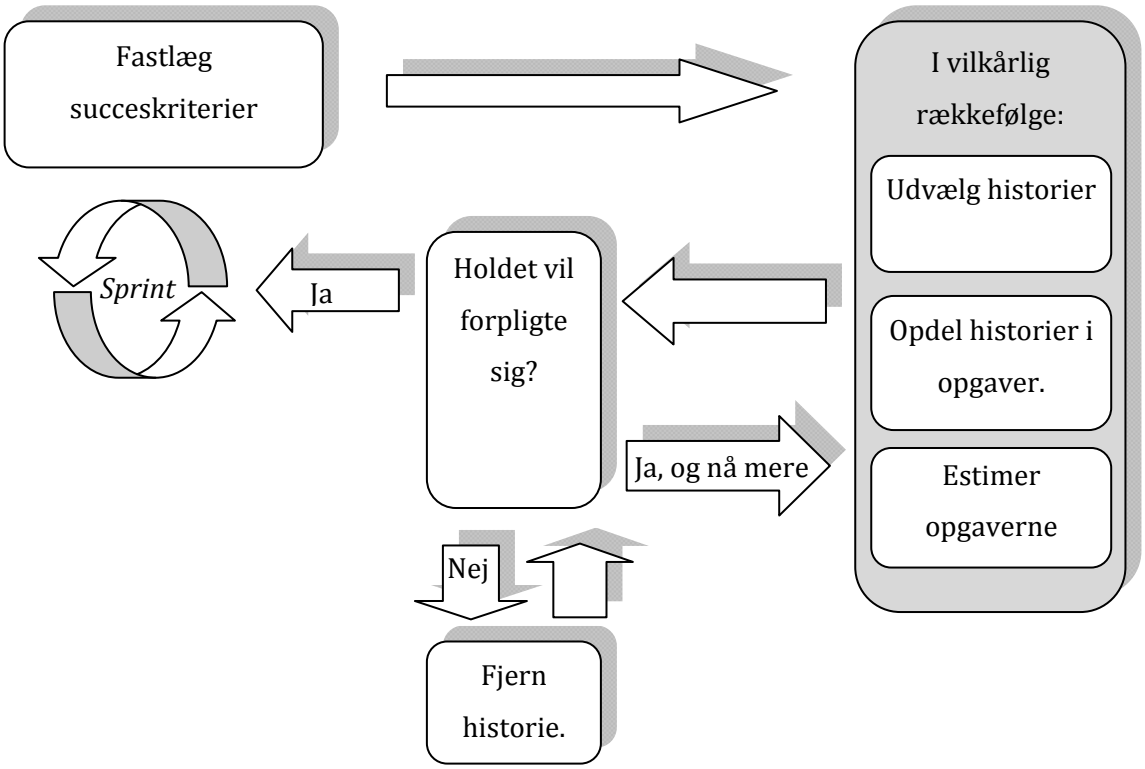
Planen er opsat så *gamer stories* er sat ind i den iteration, hvor de skal udarbejdes. Planen beskriver dermed også, hvad der forventes færdigt efter hver iteration. I ovenstående eksempel er *velocity* estimeret til 13 *story points*. Det vil sige, at planen er forsøgt udarbejdet på en sådan måde, at det samlede antal *story points* for en iteration skal være så tæt på 13 som muligt. En undtagelse er dog iteration 2-3 hvor man, på grund af omfanget af de udvalgte *gamer stories,* har valgt at lade dem overlappe.

Når ovenstående plan er udarbejdet, kan man påbegynde arbejdet med at planlægge det forestående *sprint*.

PLANLÆGNING AF SPRINT

Det primære formål med planlægningen af *sprinten* er at opsætte en plan over *gamer stories* og de opgaver, der relaterer sig til disse. Endvidere er det en vigtig del af *sprint*-planlægningsfasen, at holdet forpligter sig til at opfylde planen. På den måde forsøger man at ansvarliggøre holdet og derved det enkelte projektmedlem. Nedenstående figur afbilleder *sprint*-planlægningen:

Planlægning af sprint



Figur 6.7: Planlægning af *sprint* (Egen tilvirkning).

Som det første fastlægges *Game Owners* succeskriterier for *sprinten*. Det væsentligste er at definere, hvilke nye muligheder *Game Owner* forventer at spilleren har i spillet, når *sprinten* er gennemført. Planen over delleverancer kan anvendes som udgangspunkt for dette. Når succeskriterierne er fastlagt, skal der udvælges *gamer stories* til *sprinten*, *gamer stories* skal opdeles i delopgaver, og der skal foretages en estimering af hver opgave. Når dette er gjort, skal planen godkendes af holdet. Hvis de mener, at de indenfor iterationslængden godt kan nå mere, går man et trin tilbage og revurderer planen. Mener de ikke, at de kan forpligte sig til at nå at

implementere de udvalgte *gamer stories,* fjernes opgaver indtil holdet vil godkende planen. Når planen er godkendt går selve *sprinten* i gang. Et eksempel, på hvordan en plan over *sprinten* kan se ud, er opsat i nedenstående figur:

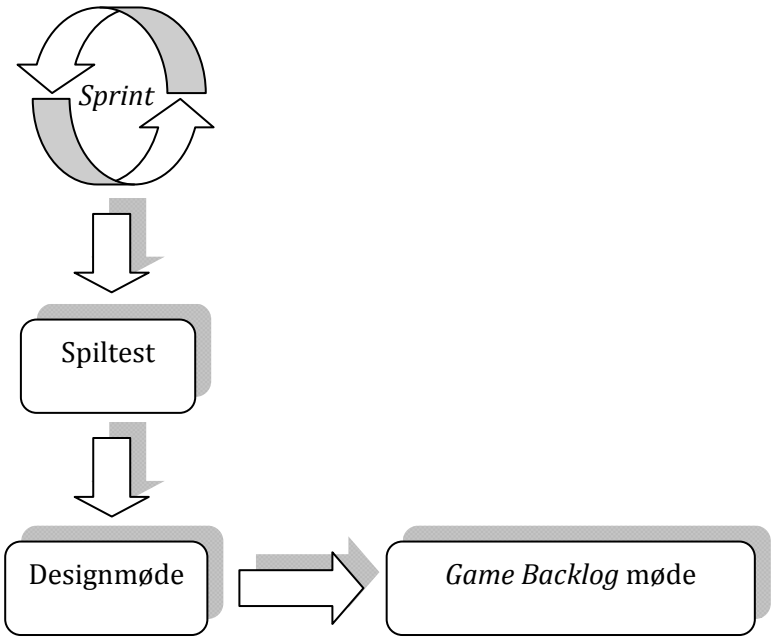
Gamer Story	**Delopgaver**	
Som spiller kan jeg få karakteren til at hoppe. 8	Fastlæg højden. 2	Animere hop-bevægelsen. 15
	Tilføj hop til karakteren. 11	Test hop-bevægelsen. 7
Som spiller kan jeg åbne dører ved at trykke på en knap. 5	Animation af knap og dør. 8	Test af knap der åbner dør. 1
	Implementering af dør/knap. 6	

Figur 6.8: Eksempel på opbygningen af en *Sprint Backlog* (Egen tilvirkning).

Bemærk af talværdierne i *gamer story* kolonnen er *story points*, mens talværdierne på delopgaverne er ideal timer/dage. En yderligere beskrivelse af forskellen mellem de to beskrives i det efterfølgende afsnit – planlægning i og efter et *sprint*.

PLANLÆGNING I OG EFTER ET *SPRINT*

En afbildning af processen, fra udviklingsholdet har godkendt *sprint*-planen, til den næste iteration påbegyndes, er opsat i figur 6.9 på næste side:

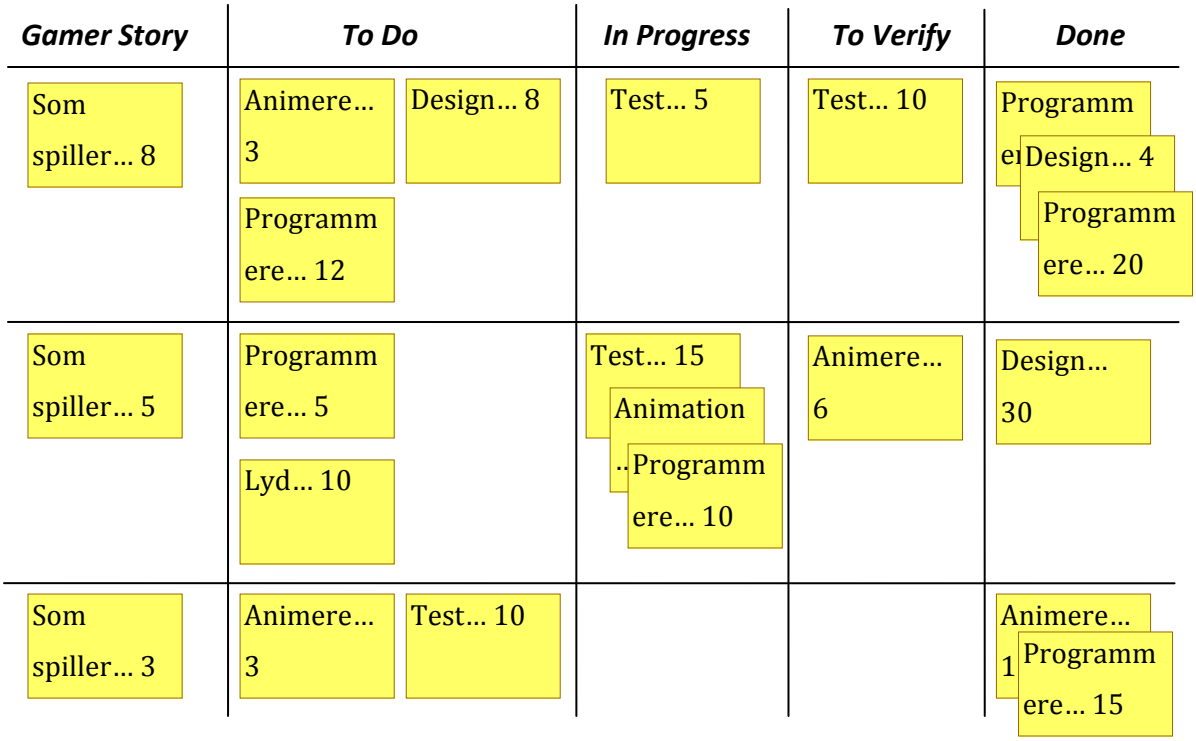


Figur 6.9: Processen fra udviklingsfasen til næste iteration påbegyndes (Egen tilvirkning).

Under *sprinten* anvendes et *task board* til at holde styr på *gamer stories* og delopgaver. Dette består eksempelvis af en opslagstavle eller væg, hvor hver opgave præsenteres ved hjælp af en seddel eller *post-it notes*. Tavlen skal inddeles i følgende segmenter:

- *Gamer Story*: Der er en prioriteret liste over de *stories*, der skal implementeres i indeværende *sprint*.
- *To Do*: Der indeholder en opdeling af historien i delopgaver, der skal udføres for at implementere historien.
- *In Process*: Der indeholder de opgaver, der på nuværende tidspunkt er under udarbejdelse.
- *To Verify*: Indeholder de arbejdsopgaver, der er implementeret men mangler at blive verificeret. Det vil sige, at der til hver arbejdsopgave skal oprettes en tilhørende verificeringsopgave. Når en arbejdsopgave er udført, skal den flyttes herover for at blive verificeret af enten *Game Owner*, en spiltester eller produceren.
- *Done*: Indeholder de *post-it notes* med opgaver, der er udført.

Et eksempel på hvordan et *task board* kan se ud, er opsat i figur 6.10 på modstående side:



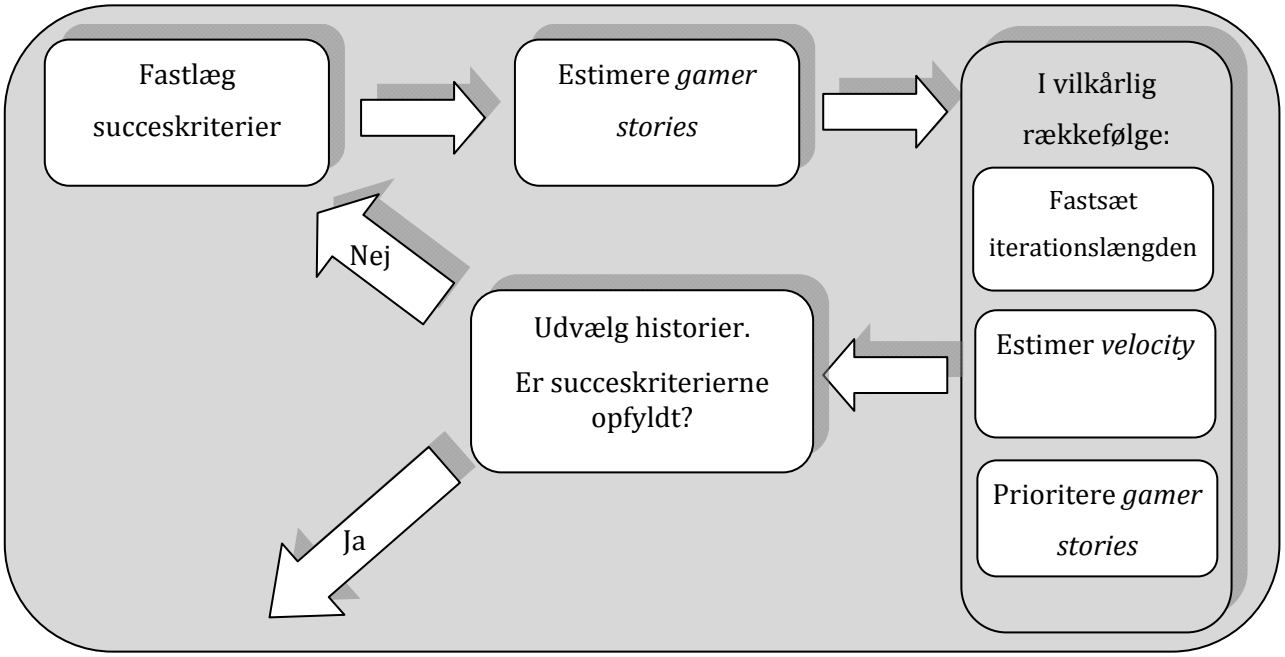
Figur 6.10: Eksempel på et *task board* (Egen tilvirkning).

Bemærk at tallene i *gamer story* kolonnen angiver *story points.* Tallene på delopgaverne i de øvrige kolonner er et estimat for hvor lang tid opgaven tager i ideelle timer. En ideel time er en times ren arbejdstid. Det vil sige, at antallet af ideelle timer per dag ikke nødvendigvis er det samme som antallet af arbejdstimer per dag. Hvis det er et større projekt, kan der anvendes ideelle dage i stedet for timer. Ligeledes, hvis det er en større produktion, anbefales det, at der anvendes to tavler; en til programmeringsopgaverne og en til udviklingen af *assets*. Når en *gamer story* således er implementeret og verificeret på programmørernes *task board*, flyttes det over til punktet *Gamer Story* på *asset*-udviklernes *task board*.

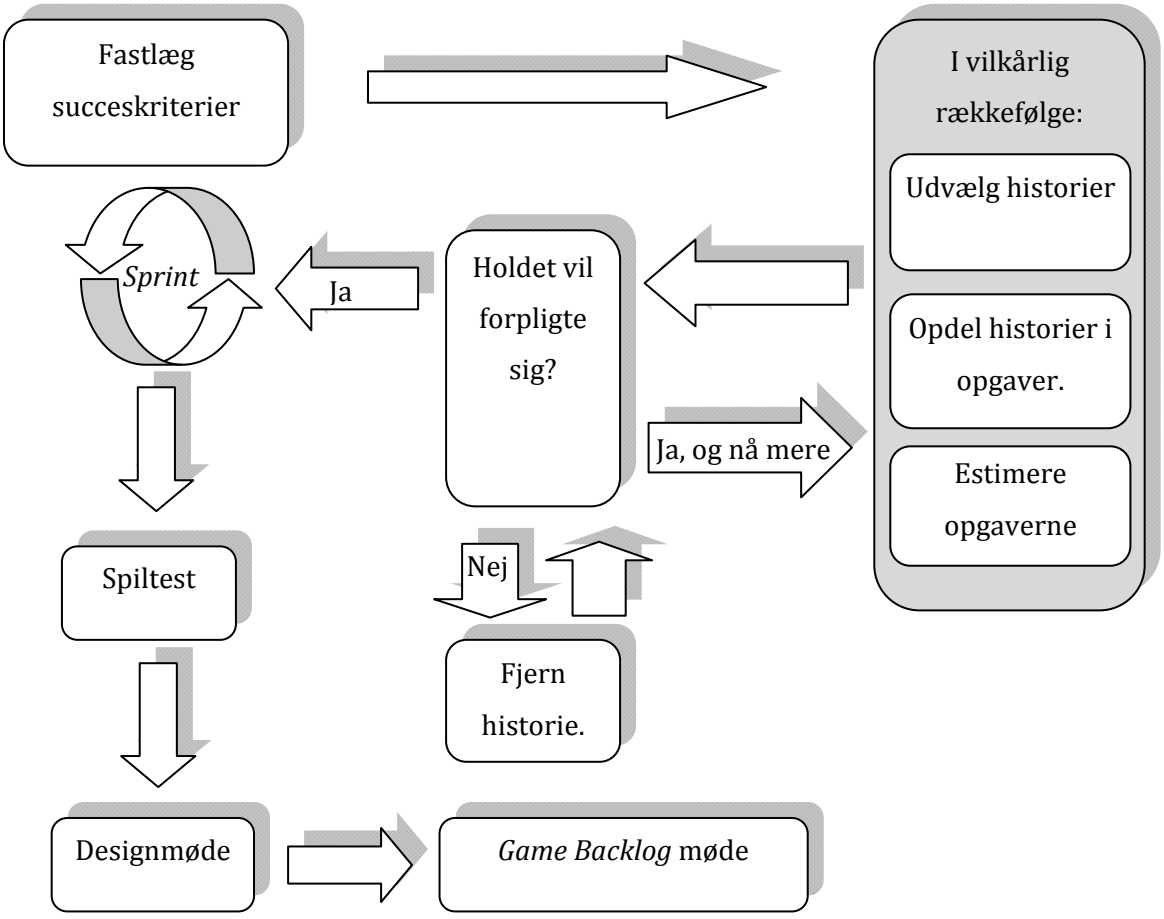
Når *sprinten* er gennemført, står man således med en ny udgave af spillet. Denne skal nu testes for at undersøge, om det lever op til spillernes forventninger. *Gamer Deputy* leder arbejdet med spiltest og opsamling af resultater, og det er hans opgave at fungere som repræsentant for spilleren. På det efterfølgende designmøde med produceren, spildesigneren og *Game Owner* er det *Gamer Deputy,* der har ansvaret for at præsentere resultaterne fra testen. Dette input anvendes til udarbejdelsen af en opdateret *Game Backlog*. Designmødet er slut, når der forelægger en ny *Game Backlog*, der indeholder enten nye, reviderede, omprioriterede eller fjernede *Gamer Stories*. Herefter præsenteres holdet for den nye *backlog*, og de har mulighed for

at stille spørgsmål og komme med forslag. Formålet er at sikre at en eventuel viden i projektgruppen synliggøres, for på den måde at drage nytte af at holdet hele tiden tilegner sig ny viden. Når holdet ikke har flere spørgsmål eller kommentarer til *backloggen* starter hele processen forfra, hvor man starter med at revurdere planen for delleverancer. Et samlet billede over SCRAME processen er opsat i figur 6.11 på modstående side:

Planlægning af delleverancer



Planlægning af sprint



Figur 6.11: Samlet planlægningsproces i SCRAME (Egen tilvirkning).

MILESTONES

Brugen af *milestones*, altså fastsættelsen af datoer hvor en del af spillet skal være færdigt, kan fungere sammen med SCRAME metoden. Det kræver dog, at de er formuleret på en sådan måde, at de beskriver, hvad der skal være færdig, og ikke beskriver selve udformningen. Således kan man i *milestones* anvende beskrivelser som: *Character controls implemented*. Formuleringer som *Character can jump, run, strafe and grap a ledge* er ikke hensigtsmæssige, idet der ikke kun beskrives hvad der skal være færdigt men også formen af det. *Milestones* er faktisk allerede en del af SCRAME metoden, idet hver delleverance kan betragtes som en sådan. Eneste forskel mellem de to er, at samtlige *milestones* normalt fastsættes inden projektet går i gang. Delleverancerne i SCRAME er mere dynamiske og måske ikke endelig fastsat før efter de første iterationer.

Milestones kan være brugbare i forbindelse med at fastsætte målet for en *sprint*, idet man således har nogle succeskriterier at arbejde ud fra. Det er vigtigt, hvis *milestone*-planlægningen skal være brugbar, at planen for *milestones* hele tiden opdateres efterhånden som projektet tager form.

LEVEL DESIGN

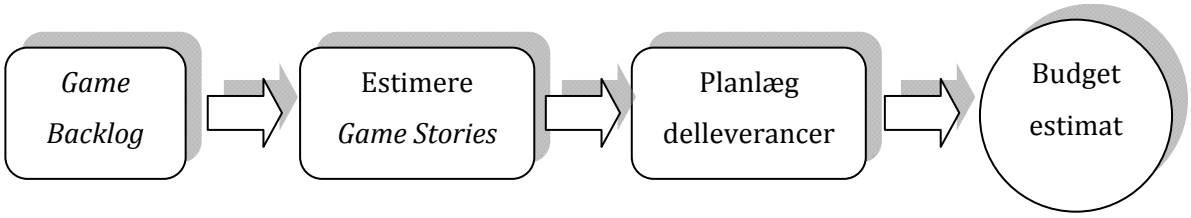
Når en *gamer story* er implementeret, skal denne afprøves. Hvis historien er afhængig af den verden, man som spiller befinder sig i, skal der laves en demobane, hvor det er muligt at afprøve den nye *feature*. Eksempelvis historien hvor spilleren kan åbne en dør ved at trykke på en knap. I denne situation er det nu op til banedesigneren at sætte disse to elementer sammen, så de forbedrer spiloplevelsen. Til en *gamer story* som denne, skal der altså laves en tilhørende delopgave, hvor historien skal indsættes og afprøves i spilverdenen.

Den egentlige proces med at udvikle baner til det endelige spil skal optimalt set først foretages, når så meget af *gameplayet* er fastlagt som muligt. Jeg foreslår, at man anvender den proces, Rouse foreslår til udviklingen af baner. Det er en iterativ tilgang, hvor man starter på et skitseniveau og bevæger sig fremad. Min beskrivelse af Rouses proces kan findes på side 53.

BUDGET

Hvordan udregner man en pris for et produkt, man ikke kender indholdet eller omfanget af? Dette er reelt umuligt og derfor er dette også et af de største problemer ved en *agile* tilgang til spiludvikling. Den eneste løsning til dette er et estimat, ud fra den *backlog Game Owner* og

udviklingsholdet som udgangspunkt er blevet enige om. Det vil sige et estimat baseret på følgende faser:



Figur 6.12: Processen i tilnærmelsen af et budgetestimat (Egen tilvirkning).

The Project Management Institute (PMI) foreslår anvendelse af et progressivt budgetestimatet undervejs i processen fordelt på i tre niveauer [Cohn, 2006:4]:

- *Initial order of magnitude estimate*: Et budgetestimat der kan svinge fra -25% til +75%.
- *Budgetary estimate*: Der kan svinge fra -10% til +25%.
- *Definitive estimate*: Der kan svinge fra -5% til +10%.

Det er derfor vigtigt at *Game Owner* bliver informeret om, at det indledende budget kan variere med -25% til +75% afhængig af de til- og fravalg, der foretages undervejs i processen. Det er herefter *Game Producerens* rolle løbende at opdatere budgetestimatet og holde *Game Owner* informeret om ændringernes økonomiske konsekvens. Beslutningen om hvorvidt en ændring skal foretages eller ej er hele tiden *Game Owners*, men det er producerens rolle at informere ham om prisen ved ændringen.

Umiddelbart er budgetteringen det største problem ved den nuværende *publisher-developer* model, der i øjeblikket er den mest anvendte fremgangsmåde i spilbranchen. Dette er derfor en problemstilling jeg ønsker at belyse yderligere i diskussionsafsnittet på de efterfølgende sider.

DISKUSSION

I denne diskussion vil jeg behandle nogle af de problemer, jeg er stødt ind i, men som jeg ikke nødvendigvis har taget stilling til. Det er blandt andet spørgsmål vedrørende de agile metoder og den iterative tilgang, spillitteraturens problemer og specialets relation til min 9. semesters opgave.

PROBLEMER VED DE AGILE METODER

Et af problemerne ved de agile metoder er, at de er bygget op omkring en proces, hvor kunden i høj grad er involveret. Det første problem opstår altså, hvis kunden ikke ønsker at afsætte den nødvendige tid til at være en del af processen eller ikke har mulighed for det på grund af en geografisk afstand. Ligeledes opstår der problemer, hvis kunden ikke har overvejet, hvilke forventninger der skal opstilles for projektet, idet grundlaget for de agile metoder netop er de krav, denne opstiller. Da SCRAME bygger på de agile metoder, er der fare for at samme problemer kan opstå her. Løsningen er dog relativ simpel - at lade en intern person tage rollen som *Game Owner*. Denne interne person skal altid have mulighed for at stille spørgsmål til *Game Owner*, hvis der opstår spørgsmål, det kun er kunden, der kan svare på eller godkende. Er dette heller ikke en mulighed skal andre metoder nok overvejes.

Den største fordel, men samtidig også den største ulempe, ved de agile metoder er, at de tager højde for at forandringer opstår undervejs i processen. Fordelen burde fremgå tydeligt af opgaven – muligheden for at sikre et underholdende spil. Ulempen ved dette opstår, når projektet skal budgetteres, for hvordan sætter man prisen på et produkt, man ikke ved hvad skal indeholde? Den bedste fremgangsmåde er at tilnærme sig et estimat, og så løbende gøre *Game Owner* opmærksom på, hvordan en *Gamer Story* vil påvirke budgettet. Dette er dog ikke det største problem. Selv med et budgetestimat vil det være svært at synliggøre, hvad det er *Game Owner* får for sine penge. Den foreslåede proces i SCRAME metoden kræver, at *Game Owner* kan acceptere og forstå, at ikke alle elementer er fastlagt fra starten, og at dette øger chancen for, at slutproduktets underholdningsværdi er væsentlig højere, end hvad den ellers ville være. Dette kræver tillid. *Game Owner* skal have tillid til udviklingsvirksomheden og SCRAME. Det er derfor højst sandsynligt, at SCRAME kun kan introduceres for *Game Owner*, hvis de to parter har

kendskab til hinanden fra tidligere produktioner. Dette betyder dog ikke, at SCRAME metoden ikke kan fungere, idet man altid kan vælge en intern repræsentant for *Game Owner*.

Kun to af forfatterne nævner faren ved en iterativ tilgang. Cohn eksemplificerer problemet på følgende vis:

> *"A hiker who wishes to reach the summit of a mountain may head for the highest peak he sees. However, once he reaches that summit he learns that it was a false peak. A higher summit had been obscured by the one he has reached. The hiker sets off toward the higher summit, only to find that it, too, is a false peak, and an even higher summit is now visible"* [Cohn, 2006:133-134].

Cohns eksempel skal vise problematikken i, at man i en iterativ tilgang ikke ved, hvornår man er færdig, idet spillet hele tiden kan gøres bedre. Der er således en fare for at projektet bliver en uendelig løkke, der først slutter, når pengekassen er tom. De agile metoders – og derved også SCRAME's – løsning på dette er ved at inddrage den finansielle part; *Game Owner*. Det er *Game Owners* egen interesse at sikre, at udviklingen af en *Gamer Story* stopper, når der ikke længere er en økonomisk bonus ved at arbejde videre. Derfor er det også vigtigt, at de succeskriterier, der går forud for planlægningsprocessen, er veldefineret, idet det er gennem disse, *Game Owner* beskriver, hvad han ønsker at se færdigt ved hver delleverance. Ved at inddrage *Game Owner* i processen burde risikoen for at projektet fortsætter i en uendelig løkke være reduceret drastisk.

PLACERINGEN AF TESTFASEN I SCRAME

I SCRAME processen (jævnfør figur 6.4 side 92) er der placeret en testfase efter hvert *sprint*. Jeg er i opgaven ikke gået i dybden med at beskrive, hvordan testen rent konkret skal udføres, da test af spil er et helt projekt i sig selv. Det vigtige er dog at understrege, at testen har det formål at undersøge, hvad spillerne oplever som underholdende og udforske deres forslag og idéer. Problemet ved placeringen af denne fase er, at den, i nuværende opstilling, vil være flaskehalsen i processen. Når et *sprint* er afsluttet, kan planlægningen af det næste ikke gå i gang før testen er udført. Dette vil betyde, at udviklingsholdet reelt vil være passive og uden arbejde i denne periode. Det er altså nødvendigt, at man tager højde for dette problem i processen. Den umiddelbare løsning er at forskyde testfasen, så den løber parallelt med *sprinten*, og man derfor står med både et videreudviklet produkt og et testresultat efter hvert *sprint*. På den måde undgår man, at processen står stille, fordi man skal bruge testresultaterne i den videre planlægning. Hvis

SCRAME metoden på et senere tidspunkt skal udvides, vil det således være oplagt at tilbyde yderligere metoder og teknikker til brug i spiltest-fasen.

ORGANISATION, GRUPPE OG INDIVID

I min 9. semesters opgave hvor jeg skrev om "Ledelsen af udviklingen af produkter i forandring", var én af mine konklusioner, at det at tage højde for forandring foregår på flere niveauer. Groft kan man inddele det i organisation, gruppe og individ [Veigaard, 2006:30]. Med organisation menes der alt det, der ligger uden om selve gruppen. Hvis udviklingsholdet eksempelvis er en del af en større virksomhed, udgør det omkringliggende altså organisationen. Med gruppe menes der de personer, der har en reel tilknytning til projektet. Det er eksempelvis på dette niveau SCRAME metoden opererer. Individet er de enkelte personer, gruppen består af. Ved at anvende en agile metode sætter man krav til alle tre niveauer. Organisationen skal være indstillet på de særlige forhold, det agile forløb kræver, såsom den manglende beskrivelse af hvad målet med projektet er. Ligeledes skal gruppens arbejde organiseres således, at ændringer undervejs i processen er muligt uden at det betyder, at deadlines overskrides. Det enkelte individs arbejde skal tilrettelægges på en sådan måde, at ændringer er nemme at lave og man derved undgår at arbejde går tabt. I denne opgave har jeg kun beskæftiget mig med gruppeniveauet. Hvis man overvejer at anvende SCRAME metoden, skal man derfor være opmærksom på, at der i metoden ikke er taget hånd om organisations- og individniveauet. Hvis man ønsker yderligere information om, hvorledes man styrer organisationen i et agilt projekt, anbefaler jeg Doug DeCarlos *Extreme Project Management* [DeCarlo, 2004]. Til individniveauet kan jeg anbefale *Extreme programming* som retningslinjer for den programmeringsmæssige del af spilprojektet. Her foreslår jeg Kent Becks *Extreme Programming Explained* [Beck, 2004].

ÅRSAGEN TIL BRANCHENS PROBLEMER

Som flere af forfatterne påpeger, er der i øjeblikket problemer med at overholde deadlines og budgetter i spilbranchen [Bethke, 2003:15-16] [Laramee, 2003: Chapter 2.1]. Men hvad skyldes dette egentligt? At jeg i denne opgave påpeger, at der mangler metoder og teknikker til udviklingen af spil, behøver dette ikke nødvendigvis være årsagen til spilbranchens problemer. Der kan være andre faktorer – såsom den rivende teknologiske udvikling der i øjeblikket finder sted. Computerne bliver stærkere og stærkere, og brugerne forventer, at det visuelle udtryk er bedre end det sidste spil, de spillede. Hvis en producent i dag laver et spil til eksempelvis 50

millioner for at følge dagens standard grafiske standard, er det ikke muligt om et eller to år. For om et eller to år vil brugernes forventninger til det æstetiske udtryk igen være skubbet, og udgifterne vil derfor ligeledes stige væsentligt. Den teknologiske udvikling kan altså, ligesom så mange andre faktorer, også være en mulig årsag til spilbranchens problemer.

Et eksempel på at det ikke nødvendigvis behøver at være sådan, finder vi i Nintendo's Wii. Grafikken på denne konsol er langt under dagens standard; det der også bliver kaldt *next gen*. Alligevel er det den spillekonsol, der bliver solgt flest af i øjeblikket [Nexgenwars, 2007]. Dette kunne være en indikation på, at der findes en anden løsning end hele tiden at forbedre det grafiske udtryk. Et spil behøver ikke være flot for at være sjovt. Og som det efterhånden burde fremgå tydeligt af denne opgave, er det vigtigste altså stadigt at spillet er underholdende. SCRAME metoden er netop et forsøg på at sætte det underholdende som fokus for processen, så det kan bestemt ikke udelukkes, at metoden kan hjælpe de virksomheder, der kæmper mod den teknologiske udvikling. Det kunne i øvrigt være interessant at vide hvilke proces Nintendo anvender, når de skal producere en ny spilkonsol.

AFRUNDING

Jeg må ærligt indrømme, at jeg ikke havde forventet det store, da jeg skulle gennemgå den eksisterende spillitteratur med henblik på at finde beskrivelser af processen. Trods dette er jeg blevet skuffet over indholdet i litteraturen. Modellerne virker usammenhængende, og de er fyldt med fejl og selvmodsigelser. Hvis den litteratur, jeg har inddraget, er et repræsentativt udsnit, er der virkelig behov for yderligere forskning på dette område. Ikke én af forfatterne har formået at opstille en model, der har fået mig til at tænke: Det er da klart - Sådan skal man lave spil! De har masse af gode idéer, og de fleste af dem har også indset fordelene ved den iterative proces, men her ophører de positive kommentarer. Der er slet ingen tvivl om, at en metode eller en beskrivelse af en udviklingsproces er et af de vigtigste værktøjer, når man skal lave et komplekst produkt, så det overrasker mig, at der ikke er skrevet bedre materiale om dette end som så. Det er mit håb, at der findes nogen derude, der vil arbejde videre med dette. Enten ved at afprøve min metode, belyse den fra en anden vinkel eller opsætte en ny metode. Der er i mine øjne ingen anden forskning, der i øjeblikket vil være i stand til at bidrage ligeså meget til spilbranchens udvikling som denne.

REFLEKSION

Dette afsnit indeholder to dele: "SCRAME's anvendelse i *9K Games*" og "Min forforståelse". Førstnævnte er en refleksion over, hvorledes SCRAME metoden kunne have været en hjælp i *9K Games* projektet. Her vil jeg kigge på de ting, jeg mener, der gik godt, og de steder, hvor SCRAME kunne have hjulpet os. I afsnittet "Min forforståelsen" vil jeg se nærmere på, hvordan min egen baggrund kan have påvirket mig undervejs i mit speciale. Årsagen til at jeg ønsker at belyse dette, skyldes den hermeneutiske vinkel jeg har lagt for projektet. Endvidere har jeg gennem opgaven kunnet se, hvordan forfatterne er påvirket af deres baggrund. Det er derfor interessant at belyse, hvordan jeg er blevet påvirket af min egen baggrund og forforståelse. Før jeg reflekterer over SCRAME og dens anvendelse i *9K Games*, vil jeg kort beskrive, hvad 9K Games er.

9K Games er betegnelsen for et projekt startet af fire studerende fra Aalborg Universitet i sommeren 2006. Formålet med projektet var at få kendskab til udviklingen af spil på et mere konkret plan ved siden af 9. semesters opgaven. Vi har alle samme uddannelsesmæssige baggrund – det vil sige en bachelor i Information og kandidat overbygningen i Multimedier. Projektet var på frivillig basis (ulønnet) og ikke et forsøg på at tjene penge. På grund af behovet for lokaler, licenser og computere søgte vi økonomisk støtte. Det fandt vi ved Aalborgs Erhvervsråd, der tilbød indkøb af computere, licenser og et lokale i erhvervshuset Dream House [Dream House, 2007]. Projektet startede i september 2006 og blev afsluttet i februar 2007.

SCRAME'S ANVENDELSE I *9K GAMES*

I *9K Games* projektet tog jeg rollen som projektleder. Dette betød, at jeg havde frie hænder til at afprøve teknikker og metoder. Da vi startede projektet, havde jeg ikke læst om SCRUM metoden så de fleste af de teknikker, jeg ville afprøve, var baseret på de erfaringer, jeg ellers havde. Dette betød for en stor del af tiden, at den proces vi fulgte blev planlagt ad hoc. Resultatet var en delvis usammenhængende men samtidig også meget lærerig proces. Mine idéer var baseret på følgende:

- Designdokumentet er en byrde. Det skal opdateres og vedligeholdes konstant, da spillet hele tiden bliver ændret i et forsøg på at gøre det sjovt. I stedet skal der fastsættes et formål med projektet, der kan være en rettesnor gennem projektet.

- Arbejdsopgaver skal defineres ud fra succeskriterier. Det vil sige, at der til hver arbejdsopgave følger en beskrivelse af, hvilke nye muligheder spilleren får.
- Vi arbejder iterativt. Der afholdes et designmøde hver uge, hvor man fastlægger hvilken retning spillet nu skal tage. Der afholdes et dagligt møde, hvor hvert enkelt medlem skal give et status for deres arbejdsopgaver.

Med disse tre retningslinjer ville vi forsøge at lave et spil. De beskrevne retningslinjer er ikke blot taget fra min hukommelse. I februar, det vil sige ved udgangen af projektet, holdt jeg et oplæg ved *Dream Games* om vores projekt og min rolle i det. De *slides* jeg anvendte i denne præsentation er vedlagt på den medfølgende CD [Veigaard, 2007]. Vores formål med projektet blev defineret som nedenstående:

- At skabe et underholdende spil for 2-4 personer.
- Spillet skal foregå i en lukket verden og skal kunne spilles over internettet.
- Spillet skal være skalerbart.

Det var vores mål at lave et spil, der opfyldte dette formål. Godt halvvejs i processen blev vi dog nødt til at ændre designet af spillet markant. Årsagen til dette lå til dels i den *game engine* vi havde valgt, og vores manglende programmeringsmæssige kompetencer. Formålet forblev dog det samme gennem hele processen.

SCRAME OG *9K GAMES*

Et af de største problemer vi havde i forbindelse med projektet, var estimeringen af arbejdsopgaver. En sådan var nærmest ikke eksisterende. Årsagen til dette skyldes primært, at vi stødte næsen mod en mur, når vi skulle estimere. Ingen af os havde de tekniske kompetencer til at sige noget om, hvor svær en opgave var. Dette var der som sådan ikke noget problem i, da vi ikke havde en kunde, vi skulle stå til regnskab overfor, men det havde dog alligevel en uheldig virkning. Når der ikke angives et estimat, er der heller ikke noget tidspunkt, opgaven skal være færdig på. Dette kan medføre et tidsspild, da der bliver brugt tid på at finpudse detaljer, spilleren alligevel aldrig vil opdage. De relative estimeringsværdier, i form af *story points,* kunne have hjulpet os meget med dette. Det ville være et fint værktøj til at måle vores fremgang og give en indikation af, hvor meget arbejde det var realistisk at nå i løbet af én uge.

Formålet med spillet var som sagt, at det frem for alt skulle være underholdende gennem et *multiplayer* modul. Forud for det ugentlige designmøde blev der altid afholdt en spilsession fra tredive til tres minutters varighed. Formålet var at teste spillets *gameplay* og opsamle feedback,

så alle havde det samme udgangspunkt forud for mødet. På mødet blev der så diskuteret mulige elementer, der kunne tilføjes, elementer i spillet der kunne forbedres, og elementer i spillet der ikke fungerede hensigtsmæssigt, og som derfor skulle fjernes. Vi fulgte altså nærmest SCRAME metodens proces fra udviklingsfasen til planlægningsfaserne som afbilledet i figur 6.9 side 99. Vi havde kun udefrakommende personer til at spilteste få gange i løbet af projektet. Dette var uden tvivl en fejl, da de eksempelvis fandt spillet meget sværere, end vi selv gjorde. Alt i alt er min konklusion dog, at spiltesten, med et efterfølgende designmøde, var effektivt. Vi havde i løbet af produktionen ikke noget problem med, at folk havde forskellige idéer om, hvor projektet var på vej hen. Alle deltog i spiltesten og det efterfølgende designmøde, så alle vidste, hvad der var aftalt, der skulle være færdigt til næste møde, og derfor var der ikke nogle overraskelser for holdet. Ligeledes er jeg slet ikke i tvivl om, at det at vi selv spillede spillet så meget, havde en stor betydning for, at spillet rent faktisk blev underholdende. Alle de elementer der ødelagde spiloplevelsen, kunne vi fjerne undervejs i processen, efterhånden som holdet fik mere indsigt i *game enginen,* og de muligheder vi havde.

En ting vi dog ikke gjorde til designmøderne var at prioritere arbejdsopgaverne efter deres værdi for spillerne. Da deltagelsen i projektet var frivilligt, var det også mit mål, at de enkelte gruppemedlemmer fik nogle arbejdsopgaver, de havde det godt med. Derfor blev arbejdsopgaverne ofte udvalgt efter hvad folk helst ville, frem for hvad der skabte mest værdi i spillet. Vi kunne dog alle sammen efter en spiltest se de områder af spillet, der haltede mest efter, så oftest var det også de opgaver, folk havde lyst til at gå i gang med. Ligeledes betød det ikke, at en person bare kunne lave det, vedkommende havde lyst til. Ud fra en liste over arbejdsopgaver, kunne hvert medlem vælge den opgave, vedkommende havde mest lyst til. Så helt frit var valget altså ikke.

En anden ting, der kunne have bidraget, er SCRAME metodens *task board*. Vi brugte tavlerne i lokalerne til at holde styr på arbejdsopgaverne, men vi fik dem aldrig delt op i delopgaver. Ligeledes blev hver enkelt arbejdsopgave ikke testet specifikt – kun gennem den ugentlige spiltest. Da vi ikke var mere end fire mand på projektet blev dette aldrig et problem. Men der er ingen tvivl om, at et *task board* ville have gjort det mere overskueligt, og det ville være uundværligt, hvis der havde været flere personer tilknyttet projektet.

Hvorvidt SCRAME's planlægningsfaser kunne anvendes, har jeg svært ved at se. Projektet var jo som nævnt frivilligt og *non-profit*, så vores mål var jo bare at nå så langt som muligt og lære så meget som muligt undervejs. Eksempelvis er det i SCRAME's planlæsningsfase en vigtig detalje, at holdet forpligter sig til at nå den beskrevne plan. Jeg tror ikke, at dette ville have medført en

forøgelse af effektivitet i *9K Games* projektet, da der ikke var en tredje part tilstede, vi kunne forpligte os overfor. Vi var vores egen kunde. Der er slet ikke tvivl om, at det er et andet forhold, når der pludselig er en betalende klient inde i billedet. I *9K Games* projektet måtte vi dog undvære dette aspekt, så metodens anvendelighed på dette punkt kan jeg ikke kommentere.

At vi fastlagde projektets formål fra starten var helt sikkert en vigtig detalje for projektet. Det var dette, der blev brugt som rettesnor ved vores designmøder. Hvis der opstod en diskussion om hvorvidt den ene eller anden funktion skulle implementeres, var definitionen af formålet med til at gøre beslutningen nemmere. Vi ville lave et spil, der skulle underholde 2-4 personer. Bidrog funktionen ikke væsentligt til dette, kom den i anden række. På den måde var formålsdefinitionen med til at gøre det nemmere at prioritere arbejdsopgaver.

Alt i alt mener jeg, at SCRAME metoden kunne have bidraget med meget. Rammerne for projektet var løse, og når jeg stødte ind et problem, skulle jeg rode rundt i kassen med de ledelsesmæssige værktøjer, jeg havde kendskab til, og se om jeg kunne finde noget frem, der passede til anledningen. SCRAME metoden tilbyder en komplet ramme for udviklingen og en beskrivelse af udviklingsprocessen. Med SCRAME metoden kunne *9K Games* projektet måske have resulteret i et bedre spil, men med sikkerhed resulteret i et bedre ledet udviklingsprojekt.

MIN FORFORSTÅELSE

Dette afsnit er skrevet som noget af det sidste i opgaven. Årsagen til at jeg skriver dette, er at du, som læser, skal være klar over, at jeg først indså nedenstående, efter jeg havde skrevet opgaven.

Min uddannelsesmæssige baggrund er fokuseret på udviklingen af IT-systemer, hvor man inddrager brugerne som en vigtig del af processen, da det er gennem disse, man kan måle, om systemet opfylder de krav, der sættes til det. Hvor tilfældigt er det så, at jeg i min metode til spiludvikling har taget udgangspunkt i systemudviklingstraditionen og beskrevet en proces, hvor man i høj grad inddrager brugerne for at definere, hvad der er underholdende?

Men er jeg så blevet påvirket at min forforståelse? Svaret må være et klart og tydeligt ja. Jeg har haft en idé om hvad spiludvikling er, og allerede i min 9. semesters opgave kiggede jeg på de agile metoder. Men hvad betyder dette så? I mine øjne betyder det ikke nødvendigvis, at den metode jeg opstiller er ubrugelig – blot at den er opstillet på baggrund af min opfattelse af verden omkring os. Det betyder derimod, at det ikke kan udelukkes, at der findes metoder og teknikker til spiludviklingsprocessen andre steder. Det er muligt, at man kan anvende metoder og teknikker fra andre felter, jeg ikke har kendskab til. Spiludvikling er virkelig et flerfagligt projekt

– programmører, designere, animatorer, grafikere, lyddesignere, forfattere og så videre. Det er ikke utænkeligt, at et af de andre felter, end dem jeg har belyst, kan bidrage med teknikker og metoder til spiludvikling. Jeg vil dog lade det være op til andre at belyse min metode fra disse aspekter.

Det jeg med andre ord forsøger at sige er, at jeg er blevet påvirket af min forforståelse, og at min metode ikke er et endegyldigt svar på, hvordan man skaber computerspil. Den er ikke nødvendigvis det eneste rigtige svar.

KONKLUSION

Mit mål er nået. Jeg ønskede at bidrage til udviklingen af metoder og teknikker, der kan anvendes i kreationen af spil, og med nærværendes speciales opstilling af SCRAME metoden er mit mål nået. På baggrund af de agile metoder er det lykkedes mig at opsætte en metode, der inddrager den iterative proces foreslået af den eksisterende spillitteratur, og samtidig formår at definere en planlægningsproces, der understøtter den iterative struktur. Ligeledes sætter metoden fokus på den vigtigste egenskab ved slutproduktet - at det kan underholde. Nu er mit håb, at dette speciale vil fungere som fundament for en videre forskning på området. Der er behov for det…

KILDELISTE

B

[Bates, 2001] Bates, Bob, 2001: *Games Design – The Art & Business of Creating Games*, Prima Publishing, ISBN: 0-7615-3165-3

[Beck et al., 2001] Kent Beck: *Manifesto for Agile Software Development*, http://agilemanifesto.org/

[Beck, 2004] Beck, Kent og Andres, Cynthia: *Extreme Programming Explained – Embrace Change, second edition*. Addison Wesley Professional, 2004, ISBN: 0-321-27865-8

[Bethke, 2003] Bethke, Erik, 2003: *Game development and production*, Wordware publishing, ISBN: 1-55622-951-8

C

[Cohn, 2006] Cohn, Mike, 2006: *Agile Estimating and Planning*, Prentice Hall, ISBN: 0131479415

[Costikyan, 1994] Costikyan, Greg, 1994: *I Have No Words & I Must Design*, http://www.costik.com/nowords.html

[Crawford, 2003] Crawford, Chris, 2003: *On Game Design*, New Riders Publishing, ISBN: 0-13-146099-4

D

[DeCarlo, 2004] DeCarlo, Doug: *eXtreme Project Management – Using leadership, Principles, and tools to deliver value in the fase of volatility*. Jossey-Bass, 2004, ISBN: 0-7879-7409-9

[Dream House, 2007] Erhvervshuset Dream House, http://www.dreamhouse.dk/

[DreamGames, 2007] Aalborg Erhversråd, Dream Games netværket: *Om Dream Games*, http://www.dreamgames.dk/

E

[Evans, 2006] Evans, Russell, 2006: *Practical DV Filmmaking – second edition*, Focal Press, ISBN 13: 978-0-240-80738-6

F

[Four Fat Chicks: 2007] Four Fat Chicks, *Interview with Dan Irish by Orb*, http://fourfatchicks.com/Rants/Interviews/Dan_Irish/Dan_Irish.shtml

[Fullerton et al., 2004] Fullerton, Tracy & Swain, Chrisopher & Hoffman, Steven, 2004: *Game Design Workshop - Designing, Prototyping, and Playtesting Games*, CMPBooks, ISBN: 1-57820-222-1

[Fullerton, 2007] Tracy Fullerton, *Game Design*: http://www.tracyfullerton.com

I

[IGDA, 2007] Board of Directors, *Bob Bates - Chair Emeritus*, http://www.igda.org/board/

[Infocom, 2007] Authors, Bob Bates, http://www.infocom-if.org/authors/bates.html

[Irish, 2005] Irish, Dan, 2005: *The Game Producer's Handbook*, Thomson Course Technology, ISBN: 1-59200-617-5

J

[Jacobsen, 1997] Jacobsen, Jan Krag, 1997: *25 spørgsmål – en moderne retorik til planlægning af kommunikation*, Roskilde Universitetsforlag, ISBN: 87-7867-030-6

K

[Krogh, 2005] Krogh, Tea: *Udviklingsprocesser i Spilindustrien – kompleksitetsreduktion gennem kommunikation*, Vedlagt på CD.

L

[Laramee, 2003] Laramee, Francois Dominic, 2003: *Secret of the Game Business*, Charles River Media, ISBN: 1-58450-282-7

[Long, 2000] Long, Ben & Scheck, Sonja, 2000: *Digital Filmmaking Handbook*, Charles River Media, ISBN: 1-58450-017-4

M

[Mathiassen, 2001] Mathiassen, Lars, 2001: *Objektorienteret Analyse og Design*, Marko, ISBN

87-7751-153-0

[McConnell, 1996] McConnell, Steve, 1996: *Rapid Development – Taming Wild Software Schedules*, Microsoft Press, ISBN: 1-55615-900-5

[Munk-Madsen, 1996] Andreas Munk-Madsen, 1996: *Strategisk Projektledelse*, Marko, ISBN: 87-7751-115-8

N

[Nexgenwars, 2007] Nexgen Wars, http://nexgenwars.com/, *Console sales*.

P

[Pearce, 1997] Pearce, Celia, 1997: *The Interactive Book*, Alpha Books, ISBN: 1578700280

[Pressman, 2000] Pressman, Roger S., 2000: *Software Engineering – A Practitioner's Approach (European Adaption)*, McGraw-Hill Publishing, ISBN: 0 07 709677 0

R

[Rollings, 2004] Rollings, Andrew & Morris, Dave, 2004: *Game Architecture and Design – a new edition*, New Riders Publishing, ISBN: 0-7357-1363-4

[Rosenstand, 2004] Rosenstand, Claus & Kyed, Per, 2004: *Computerspil – manifest*. Spilforskning.dk, ISBN: 87-990066-1-8

[Rosenstand, 2007] Rosenstand, Claus*: Et oplæg om projektledelse i spilindustrien*, Vedlagt på CD.

[Rouse, 2005] Rouse, Richard, 2005: *Game Design – Theory & Pratice (Second Edition)*, Wordware Publishing, ISBN: 1-55622-912-7

S

[Salen, 2004] Salen, Katie og Zimmerman, Eric, 2004: *Rules of Play – Game Design Fundamentals*, MIT Press, ISBN: 0-262-24045-9

[Sellers, 2003] Sellers, Michael, 2003: *The stages of game development*, Artikel i *Secrets of the Game Business Industri* af Laramee, Francois Dominic, 2003

[Schein, 1994] Schein, Edgar H., 1994: *Organizational Psychology*, Prentice Hall, ISBN: 0-13-641340-4

[Schwaber, 2004] Schwaber, Ken, 2004: *Agile Project Management with Scrum*. Microsoft Press, 2004, ISBN 0-7356-1993-X

U

[USC, 2007] University of Southern California, Master of Fine Arts Program: http://roski.usc.edu/mfa/

V

[Veigaard, 2006] Veigaard, Søren: *Fornemmelse for forandring – Om at lede udviklingen af et produkt i forandring*. Vedlagt på CD

[Veigaard, 2007] Veigaard, Søren: *9K Games og Brick Walker.* Et oplæg ved Dream Games d. 8. februar 2007. Vedlagt på CD

W

[Wiki, 2007: Richard Rouse III] http://en.wikipedia.org/wiki/Richard_Rouse_III

[Winograd & Flores, 1987] Winograd, Terry & Flores, Fernando, 1987: *Understanding Computers and Cognition – a New Foundation for Design*, Addison-Wesley, ISBN: 0-201-11297-3

[Wysocki, 2003] Wysocki, Robert og McGary, Rudd: *Effective Project Management - Traditional, Adaptive, Extreme, Third Edition*. Wiley Publishing Inc, 2003, ISBN: 0-471-43221-0

BILAG

BILAG 1 – DEFINITIONER AF GAME

David Parlett:

> *"A formal game has a twofold structure based on ends and means:*
>
> *Ends. It is a contest to achieve an objective. Only one of the contenders, be they individuals or teams, can achieve it, since achieving it ends the game. To achieve that object is to win. Hence a formal game, by definition, has a winner; and winning is the "end" of the game in both senses of the word, as termination and as object.*
>
> *Means. It has an agreed set of equipment and of procedural "rules" by which the equipment is manipulated to produce a winning situation".*

Clark C. Abt:

> *"Reduced to its formal essence, a game is an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context. A more conventional definition would say that a game is a context with rules among adversaries trying to win objectives".*

Johann Huizinga:

> *"[Play is] a free activity standing quite consciously outside "ordinary" life as being "not serious", but at the same time absorbing the player intensely and utterly. It is an activity connected with no material interest, and no profit can be gained by it. It proceeds within its own proper boundaries of time and space according to fixed rules and in an orderly manner. It promotes the formation of social groupings, which tend to surround themselves with secrecy and to stress their difference from the common world by disguise or other means".*

Roger Caillois:

> *Free: in which playing is not obligatory; if it were, it would at once lose its attractive and joyous quality as diversion;*
>
> *Separate: circumscribed within limits of space and time, defined and fixed in advance;*
>
> *Uncertain: the course of which cannot be determined, nor the result attained beforehand, and some latitude for innovations being left to the player's initiative;*
>
> *Unproductive: creating neither goods, nor wealth, nor new elements of any kind; and, except for the exchange of property among the players, ending in a situation identical to that prevailing at the beginning of the game;*
>
> *Governed by rules: under conventions that suspend ordinary laws, and for the moment establish new legislation, which alone counts;*
>
> *Make-believe: accompanied by a special awareness of a second reality or of a free unreality, as against real life.*

Bernard Suits:

> *To play a game is to engage in activity directed towards bringing about a specific state of affairs, using only means permitted by rules, where the rules prohibit more efficient in favour of less efficient means, and where such rules are accepted just because they make possible such activity.*

Chris Crawford:

> *Representation: A game is a closed formal system that subjectively represents a subset of reality. By "closed" I mean that the game is complete and self-sufficient as a structure.The model world created by the game is internally complete; no reference need be made to agents outside of the game. By formal I mean only that the game has explicit rules. A game's a collection of parts which interact with each other, often in complex ways. It is a system. A game creates a subjective and deliberately simplified representation of emotional reality.*

Interaction: The most fascinating thing about reality is not that it is, or even that it changes, but how it changes, the intricate webwork of cause and effect by which all things are tied together. The only way to properly represent this webwork is to allow the audience to explore its nooks and crannies, to let them generate causes and observe effects. Games provide this interactive element, and it is a crucial factor in their appeal.

Conflict: A third element appearing in all games is conflict. Conflict arises naturally from the interaction in a game. The player is actively pursuing some goal. Obstacles prevent him from easily achieving this goal. Conflict is an intrinsic element of all games.It can be direct or indirect, violent or nonviolent, but it is always present in every game.

Safety: Conflict implies danger; danger means risk of harm; harm is undesirable. Therefore, a game is an artifice for providing the psychological experiences of conflict and danger while excluding their physical realizations. In short, a game is a safe way to experience reality. More accurately, the results of a game are always less harsh than the situations the game models.

Greg Costikyan:

A game is a form of art in which participants, termed players, make decisions in order to manage resources through game tokens in the pursuit of a goal.

Elliot Avedon

Games are an exercise of voluntary control systems, in which there is a contest between powers, confined by rules in order to produce a disequilibrial outcome.

BILAG 2 – FORSKEL PÅ FILM, SPIL OG SOFTWARE

Spil	Film	Systemudvikling
Interaktionen skal være underholdende. Historien og fortælleformen kan være med til at gøre spillet underholdende. Der behøver ikke være en historie i et spil.	Historien og fortælleformen skal være underholdende. Historien er altafgørende for filmen.	Skal ikke være underholdende. Udvikling af IT-redskaber. Funktionaliteten er altafgørende.
Test: Kan spillet køre fejlfrit, og er det underholdende.	Test: Kan målgruppen forstå sammenhængen i filmen og er filmen underholdende.	Test: Brugervenlighed, Fejltest, Funktionstest.
En scene/bane er dyr og tidskrævende at lave om, hvis produceren kræver store ændringer. Mangler i scenen kan oftest indsættes på et senere tidspunkt.	En scene er billig at tage om, indtil produceren er tilfreds. Mangler i scenen kan medføre store forsinkelser.	Kan opbygges i uafhængige moduler, der gør ændringer billige. Store ændringer sent i produktionen kan medføre store omkostninger.
Følger processen: Development, pre-production, production, post-production.	Følger processen: Development, pre-production, production, post-production.	Ingen entydig proces er defineret. Der findes mange veldefinerede og lettilgængelige processer.
Kan være skalerbart.	Kan ikke være skalerbart. (med mindre der er tale om en serie)	Kan være skalerbart.
Spil bygger på et designdokument – der er ingen klare linjer for, hvordan det skal opsættes, og hvad det skal indeholde.	Film bygger på et manuskript – der er klare linier for, hvad et sådan skal indeholde, og hvordan det skal opsættes.	Bygger oftest på en kravspecifikation. Kan bygges på kundens ønsker og behov.
Det er billigt at rette små fejl sent i produktionen.	Det er dyrt at rette små fejl sent i produktionen.	Det er billigt at rette små fejl sent i produktionen.
Terminologi ikke entydig. Mængden af værktøjer er begrænset.	Veldefineret terminologi og en stor mængde værktøjer til at styre processen er tilgængelige.	Veldefineret terminologi og en stor mængde værktøjer til at styre processen er tilgængelige.

BILAG 3 - MAIL TIL RICHARD ROUSE

Sendt til: gdtp@paranoidproductions.com, paranoid@paranoidproductions.com

Topic: Regarding your educational background

My name is Søren Veigaard and I'm currently studying Game Design at Aalborg University - Denmark. I'm writing my master thesis on game development and production and I'm using your book (Game Design - Theory & Practice) alongside many others (to see a list of these please read further down). My problem is that I'm interested in knowing the writers background as I think this reveals a lot about why the writer see game development the way that he/she do, but I haven't been able to find any information on your educational background. Could you please supply with these information's?

Hope to hear from you..

Søren Veigaard

List of work cited:

[Laramee, 2003] Laramee, Francois Dominic, 2003: Secret of the Game Business, Charles River Media, ISBN: 1-58450-282-7

[Rollings, 2004] Rollings, Andrew & Morris, Dave, 2004: Game Architecture and Design – a new edition, New Riders Publishing, ISBN: 0-7357-1363-4

[Rouse, 2005] Rouse, Richard, 2005: Game Design – Theory & Pratice (Second Edition), Wordware Publishing, ISBN: 1-55622-912-7

[Pearce, 1997] Pearce, Celia, 1997: The Interactive Book, Alpha Books, ISBN: 1578700280

[Fullerton et al., 2004] Fullerton, Tracy & Swain, Chrisopher & Hoffman, Steven, 2004: Game Design Workshop - Designing, Prototyping, and Playtesting Games, CMPBooks, ISBN: 1-57820-222-1

[Costikyan, 1994] Costikyan, Greg, 1994: I Have No Words & I Must Design, http://www.costik.com/nowords.html

[Crawford, 2003] Crawford, Chris, 2003: On Game Design, New Riders Publishing, ISBN: 0-13-146099-4

[Bates, 2001] Bates, Bob, 2001: Games Design – The Art & Business of Creating Games, Prima Publishing, ISBN: 0-7615-3165-3

[Bethke, 2003] Bethke, Erik, 2003: Game development and production, Wordware publishing, ISBN: 1-55622-951-8

BILAG 4 – MAIL FRA RICHARD ROUSE

Glad to hear you are finding my book useful.

My educational background was a standard four-year degree from the University of Chicago. The degree was bachelors of Science in applied mathematics/computer science, but I also studied literature and film. In the US this is what they call a "liberal arts" education, meaning it was not overly focused on one subject. Most of what I learned about game development and game design I learned through self-education and while developing projects.

- Richard

Good luck with your paper!

BILAG 5 – ROUSES DOKUMENTER

- Konceptdokument: Dette dokuments primære funktion er at kommunikere til hvem end det er, der betaler for projektet. Det skal anvendes til at skaffe og sikre, at projektet bliver finansieret. Konceptdokumenteret indeholder derfor et overordnet beskrivelse af spillet sammen med konceptgrafik til at illustrere *gameplayet* og det æstetiske udtryk [Rouse, 2005: 308].

- Analyse af konkurrencedygtighed: Dette dokument skal ligeledes sikre, at projektet kan skaffe finansiering. Det indeholder en beskrivelse af de andre spil på markedet, der ligger tæt op af det foreslåede – det vil sige en beskrivelse af konkurrenternes *gameplay*, salgstal, anmeldelser med mere. Dokumentet skal give et indblik i spillets konkurrencedygtighed, og det kan måske vise, at projektet allerede er realiseret af andre [Rouse, 2005: 309].

- Designdokument: Designdokumentets formål er at beskrive spillets *gameplay* ned til mindste detalje. For store projekter er dette dokument en essentiel reference til at definere, hvordan forskellige dele af spillet skal fungere. Selve kernen i dokumentet er udspecificeringen af spilmekanikker. Det vil sige en beskrivelse af, hvad spilleren kan gøre i spiluniverset, og hvordan dette skal medvirke til en samlet *gameplay*-mæssig oplevelse. Designdokumentet bliver ifølge Rouse ligeledes ofte anvendt, når der skal laves en tidsplan for projektet [Rouse, 2005: 309].

- *Flowcharts*: *Flowcharts* har to formål. De kan enten visualisere, hvorledes spilleren kan navigere gennem spillets menuer, eller det kan illustrere spillerens progression i eksempelvis en bane. Formålet er at visualisere, hvad spillerens handlinger vil medføre, og kommunikere dette ud til det øvrige team [Rouse, 2005: 311].

- *Story Bible*: *Story Bible* skal indeholde spillets historie. I spil hvor historien ikke er essentiel, kan dette dokument være flettet ind i designdokumentet, men hvis historien fylder en stor del af spillet, kan dette gøre designdokumentet uoverskueligt. *Story Bible* kan, foruden selve historien, indeholde information om karaktererne, en beskrivelse af stederne hvor historien udspiller sig og så videre. Formålet med dokumentet er at give grafikkerne en mulighed for eksempelvis at kende mere til en karakters baggrund og derved skabe en 3d-model, der passer bedre ind i historien og det æstetiske udtryk. Ligeledes er *Story Bible* et vigtigt dokument, når der skal laves et manuskript [Rouse, 2005: 311].

- Manuskript: Hvis spillet indeholder meget historie, er det sandsynligt, at der vil forekomme dialog i spillet. Manuskriptet skal bruges til at beskrive dialogen, og den situation hvor den udfolder sig i. Manuskriptets form er meget afhængig af hvordan historien er integreret i spillet. Hvis det er i form af *cut-scenes*, kan man bruge den samme form, der anvendes i film-manuskripter. Hvis historien bliver fortalt via *in-game* dialoger, er det nok bare at beskrive dialogen, da spilleren således har kontrollen med kameraet [Rouse, 2005: 313].

- *Art Bible*: Dette dokument indeholder konceptskitser og beskrivelser af spillets æstetiske udtryk. Det er dokumentet, hvor spillets udseende og fornemmelse kommer til udtryk og bliver beskrevet i detaljer. Ligeledes skal *Art Bible* også indeholde tekniske retningslinjer for det grafiske arbejde. Eksempelvis hvilket format *assets* skal have for at fungere i spillets *engine*, hvor mange polygoner der er til rådighed, og hvor mange *frames* forskellige animationer må anvende [Rouse, 2005: 315].

- Spil Minuttet: Dette dokument beskriver på få sider, hvordan kerneelementerne i *gameplayet* vil fungere i spillet. Det vil sige en beskrivelse af, hvad spilleren giver af *input* og hvordan spillet så reagerer på dette. Hvis der er flere mulige måder, hvorved et scenarie kan udspilles, kan man anvende flere dokumenter til at beskrive de forskellige situationer. Dokumentets formål er at sikre, at hele udviklingsholdet forstår, hvorledes spillets mest grundlæggende spilmekanikker skal fungere [Rouse, 2005: 316].

- *Storyboards*: Dette dokument finder sin anvendelse i spilproduktioner, hvor der bliver gjort brug af *cut-scenes*. Her kan de allerede etablerede retningslinjer fra filmproduktion anvendes, når scener skal beskrives, inden de bliver udarbejdet. Dette giver mulighed for at de øvrige projektdeltagere kan give feedback og korrigere *cut-scenes* inden arbejdet med at sætte dem op, filme og rendere dem begynder [Rouse, 2005: 317].

- Teknisk Designdokument: Hvor designdokumenter beskriver, hvordan spillet skal fungere, indeholder det tekniske designdokument en beskrivelse af, hvorledes disse funktioner skal implementeres. I dette dokument bliver de strukturmæssige retningslinjer opsat for koden, klasserne der vil blive anvendt, renderings-arkitekturen, hvordan den kunstige intelligens vil fungere og så videre. Dokumentet skal gøre det muligt for programmørerne at se, hvorledes et bestemt system skal implementeres for at fungere i henhold til spillets *engine* og design [Rouse, 2005: 317].

- Tidsplaner og forretningsmæssige dokumenter: Hvis man vil gøre sig nogen forhåbninger om i det mindste at få projektet til at løbe rundt, er det ifølge Rouse nødvendigt med en gennemtænkt forretningsplan. Dette betyder, at der skal udarbejdes marketingsforslag, budgetter, tidsplaner og alle andre nødvendige dokumenter, der kan hjælpe pr-afdelingen, salgsfolkene og marketingsafdelingen. Ligeledes er det ifølge Rouse en vigtig måde at sikre, at investorerne forstår, hvad der gør spillet godt, og derved er mindre tilbøjelige til at forlange ændringer eller helt stoppe finansieringen af projektet [Rouse, 2005: 318].

BILAG 6 - COHNS KRAV TIL PLANLÆGNINGSPROCESSEN

Estimering og planlægning er ifølge Cohn ikke bare at fastsætte en passende *deadline* og tidsplan. Planlægning er i ligeså høj grad en søgen efter at maksimere produktets værdi [Cohn, 2006, 5]. En god planlægningsproces skal derfor gøre det muligt at undersøge, hvad systemet skal indeholde for at tilføje mest værdi. Ifølge Cohn kræver det, at processen tilbyder følgende fem elementer:

- Reducering af risici: Planlægning øger chancen for, at projektet bliver en succes ved at tilbyde en indsigt i de risici, der knytter sig til projektet. Det er ofte, når opgaver skal estimeres, at potentielle risici opdages, da en diskussion om opgaven kan give en fornyet indsigt i, hvad opgaven kræver, og de risici der er forbundet med dette. Når først risikoen er synliggjort, kan den håndteres ved eksempelvis at afsætte tid til at lære mere om opgaven [Cohn, 2006: 5].

- Reducere usikkerheder: Igennem et projekt vil teamet opnå nye færdigheder og en ny viden – om produktet, teknologien, dem selv og så videre. Denne nye viden skal inkorporeres i planlægningsprocessen, så man undgår den mest kritiske risici: at udvikle det forkerte produkt. Den bedste måde til at undgå dette er ifølge Cohn ved hjælp af en iterativ planlægningsproces, der hjælper teamet med at udvikle deres forståelse af produktet og dens vision [Cohn, 2006: 6]. Dette passer meget godt ind i udviklingen af spil, hvor teamet skal lære, hvad der gør spillet underholdende og så bygge videre på dette. Ligeledes mener Cohn, at den iterative tilgang til planlægning er med til at øge forståelse for selve produktet. Det er netop et af de punkter, Krogh nævner som et af de største problemer ved spiludvikling; Dét at skabe en fælles forståelse for produktet.

- Understøtter bedre beslutninger: Planlægning er en konstant afvejning af funktionalitet, anstrengelser, omkostninger og tid, og for at tage den rigtige beslutning er det nødvendigt at have estimater af både omkostningerne, og den værdi der bliver tilføjet. Den agile planlægning gør det nemmere at tage beslutninger ved at synliggøre hvilke elementer, der tilføjer størst værdi til produktet, og hvad det enkelte element koster [Cohn, 2006: 6-7].

- Skaber tillid: En god planlægningsproces skal ifølge Cohn skabe tillid mellem de forskellige parter. Den agile tilgang gør dette ved jævnligt at levere de lovede funktioner i

systemet. Det skaber tillid, at kunden kan udvælge en række funktioner, og at de så er implementeret til aftalt tid og med den ønskede funktionalitet. Årsagen til at dette er muligt, er de estimater, hvorfra kunden vælger hvilke dele, der skal implementeres til den næste leverance [Cohn, 2006: 7].

- Transportere informationer: En plan transporterer forventninger og beskriver et muligt udfald af, hvad der sker i løbet af et projekt. En plan skal ifølge Cohn ikke garantere et nøjagtigt antal funktioner til en bestemt dato og til en bestemt pris. I stedet skal den kommunikere og etablere en grundlæggende forventning til projektet. En plan skal derfor indeholde et estimat for hvor lang tid projektet vil tage, hvad der vil være færdig i løbet af de første et til to måneder af projektet, de grundlæggende formodninger projektet er baseret på, og en beskrivelse af, hvordan fremgang måles [Cohn, 2006: 8].

BILAG 7 – PLANNING POKER

Følgende er en beskrivelse af Mike Cohns *Planning Poker* – en teknik til at estimere omfanget af en *user story* eller et *theme* [Cohn, 2006: 54-60]:

Ifølge Cohn er der tre teknikker, der er de mest anvendte i forbindelse med estimering:

Expert opinion: Man estimerer ved at spørge en person, der er ekspert på området. På grund af anvendelsen af *user stories* er denne metode dog ikke særlig velegnet til estimering. En *user story* indeholder ofte mere end funktionalitet (eksempelvis lyd, grafik, animationer mm.), og det kan således være, at den skal udarbejdes af mere end én person. Det gør det svært for en person at estimere *user storiens* omfang.

Analogy: Man estimerer ved at se på opgavens omfang i forhold til de øvrige opgaver. Hvis opgaven bliver estimeret til eksempelvis fem *story points,* skal opgaven gerne virke større end den historie, der er estimeret til tre men mindre end den, der er estimeret til otte.

Disaggregation: Man estimerer en opgave ved at dele den op i mindre dele, der så er nemmere at estimere. En anden fordel ved *disaggregation* er, at man får nedbrudt opgaven, hvilket også betyder, at man får synliggjort de dele, opgaven består af, hvilket er en stor fordel, når man arbejder med *user stories*.

Planning Poker er ifølge Cohn en måde at estimere på, hvor man bygger videre på alle tre ovenstående teknikker, samtidig med at man forsøger at gøre estimeringen mere underholdende. Idéen bygger på, at man samles i en gruppe bestående af personer, der alle har en tilknytning til de *user stories*, der skal estimeres. Hvert medlem af gruppen får et sæt kort med den estimeringsværdier, der anvendes. Dette kunne eksempelvis være 0, 1, 2, 3, 5, 8, 13, 20, 40 og 100 (eller Fibonacci talrækken). Herefter går man igennem hver enkelt *user storie*, ved at en moderator (som regel produktejeren) læser beskrivelsen. Teamet får mulighed for at stille spørgsmål til den enkelte *user story* for at synliggøre elementer, der påvirker estimatet. Når alle spørgsmål er besvaret, udvælger hver deltager det kort, der bedst repræsenterer vedkommendes syn på estimatet. Når alle deltagere har udvalgt et kort, vendes de simultant om. Hvis der er store udslag i estimatet, gives der nu mulighed for, at den enkelte deltager kan begrunde estimatet. Det er muligt, at vedkommende besidder en viden, de andre ikke gør, og på den måde kan se en stor risiko eller en nem løsning af opgaven. Gruppen får mulighed for at diskutere estimatet

yderligere et par minutter, hvorefter de igen skal udvælge et kort. Er der også anden gang stor uenighed om estimatet foretages endnu en diskussion, da der stadig må være dele, der ikke er blevet synliggjort tilstrækkeligt.

Det kan være nødvendig at lære gruppen teknikken før selve udviklingen går i gang. Dette har også det formål, at alle har en idé om, hvad værdierne på kortene repræsenterer, inden de bliver opdelt i mindre estimeringsgrupper.

Cohn definerer to steder i processen, hvor *planning poker* kan være anvendeligt. Det ene er før hver iteration og/eller inden selve udviklingen går i gang. Det andet er undervejs i processen, hvor der midt i en iteration kan identificeres nye *user stories*, der skal estimeres, før man kan placere dem i planen.

Bemærk at formålet med denne teknik ikke er at definere et estimat, der kan holde i al evighed. Målet er at definere en teknik, hvor et gyldigt estimat kan opsættes indenfor fornuftige tidsmæssige og økonomiske grænser.