

Creative Development of Games



Semester F6S Projekt

Skrevet af: Irma Nezirevic, Kim Beck
Mike Khampoukeo, Michael Nielsen

Afleveres d. 27.05.2005

CDoG – Creative Development of Games

Indhold

1. PROBLEMFORMULERING	4
2. INTRODUKTION.....	5
3. PROCES MODELLER TEORI.....	5
3.1 ITERATIVE PROCESSER	5
3.1.1 <i>Incremental Development</i>	6
3.1.2 <i>Spiral model</i>	6
3.1.3 <i>Prototyping model</i>	8
3.3 AGILE MODELLER	10
3.3.1 <i>SCRUM</i>	12
3.3.2 <i>Extreme Programming</i>	13
3.4 DELKONKLUSION.....	15
4. VIRKSOMHEDSBESKRIVELSE.....	17
4.1 INTERACTIVE TELEVISION ENTERTAINMENT	18
4.2 MEDIAMOBSTERS.....	23
4.3 DEL KONKLUSION	26
5. CDOG MODELLEN.....	28
5.1 FORUDSÆTNINGER FOR BRUG AF CDOG.....	28
5.2 INTRODUKTION.....	29
5.3 VÆRDIER FOR CDOG MODELLEN	32
5.4 RISK MANAGEMENT	35
5.5 CHANGE MANAGEMENT	36
5.6 PROJEKTSTYRING.....	38
5.6.1 <i>Principper for projektstyring</i>	42
5.6.2 <i>Gode råd</i>	44
5.7 MODULER.....	46
5.7.1 <i>Gode Råd</i>	50
5.8 INTEGRATION.....	52
5.8.1 <i>Gode råd</i>	60
5.9 PRINCIPPER FOR MODUL OG INTEGRATION FRAMEWORKS.....	61
5.10 CDOGS AFSLUTNINGS FASE.....	65
6. IMPLEMENTERING AF CDOG.....	66
6.1 TRINVIS IMPLEMENTERING	66
6.2 BIG-BANG.....	68
6.3 EVENTUELLE PROBLEMSTILLINGER VED INTEGRATION AF CDOG	70
7. METRIKKER.....	72
8. KONKLUSION	75
9. LITTERATURLISTE.....	79

CDoG – Creative Development of Games

Mike Sjørlev Khamphoukeo

Irma Nezirevic

Michael Kastoft Nielsen

Kim Beck

CDoG – Creative Development of Games

1. Problemformulering

Markedet for spil er i konstant vækst og udvikling. Det er meget konkurrencepræget og derfor er det vigtigt for et spiludviklings firma, at skabe et produkt der kan differentiere sig fra de andre produkter på hylden og samtidig gøre virksomheden konkurrencedygtig. Derfor er det vigtigt for virksomheden, at have en proces model for spiludviklingen der giver et klart billede af kravene og udviklingen af spillet. Problemet her er, at de eksisterende processer som bruges i udviklingsbranchen ikke passer godt til spil udvikling, da de er alt for struktureret og dermed indskrænkende for kreativiteten.

Selvom der er problemer med disse, så ønsker virksomhederne alligevel at implementere processer i deres arbejdsfremgangsmåde, da det giver dem et overblik over problemområdet. Som standard bruger de fleste ”code and fix,” som dog har mange ulemper. Derfor ønsker vi at:

- Identificere hvilke problemer spil virksomhederne har med de almindelige proces modeller. Det ønskes undersøgt, hvorfor de almindeligt brugte processer ikke virker så godt for spil miljøet og hvad der adskiller spiludviklingen fra almindeligt systemudvikling.
- Undersøge om nogle af de eksisterende systemudviklingsmodeller kan modificeres så det passer til spiludviklingsmiljøet.
- Udarbejde, ud fra resultaterne fra de to overstående punkter, en proces model som er tilpasset til udvikling af spil. Dette betyder, at der bl.a. skal tages stort hensyn til kreativiteten, så denne ikke bliver undertrykt under processen.
- Give forslag til hvordan den nyudviklede model kan implementeres i en virksomhed. Derudover vil vi kigge på mulige måder på at verificere om modellen virker.

CDoG – Creative Development of Games

2. Introduktion

Inden selve løsningen bliver foreslået, vil de næste par afsnit beskrive teorien bag nogle af de brugte procesmodeller i dag, som dele af vil blive brugt i løsningen. Derudover har gruppen undersøgt, hvilken fremgangsmåde spiludviklings virksomheder i Danmark bruger, når det kommer til spil udviklingen. Gruppen har besøgt virksomhederne ITE og Media Mobsters, samt snakket med en udvikler der har været med til at udvikle Hitman spillet. Resultater af disse samtaler vil blive brugt for at identificere problemområderne som vil blive brugt som baggrund for gruppens løsning.

3. Proces Modeller Teori¹

3.1 Iterative Processer

Før i tiden var køberne villige til at vente lang tid på, at software applikationer skulle blive færdige. Somme tider kunne der gå år mellem tiden, hvor man var i gang med at skrive ”behovs dokumentation” og hvor systemet endelig blev afleveret (også kaldet ”livs cyklus”). Men nu til dage er man ikke længere så tålmodige.

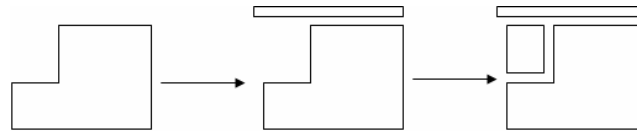
Software er blevet lige så vigtig eller endda vigtigere end hardwaren og køberne leder hele tiden efter nye kvaliteter og funktionalitet.

Målet med iterative processer er, at processen kører iterativt, hvor man afleverer dele af software til kunden så snart det er muligt, samtidig med at man fortsætter med videreudviklingen af denne. Modeller som er baseret på denne fremgangsmåde er incremental development, spiral modellen og prototyping modellen.

¹ Teorier til de forskellige modeller kan læses I bøgerne [SEP] og [SET]

CDoG – Creative Development of Games

3.1.1 Incremental Development

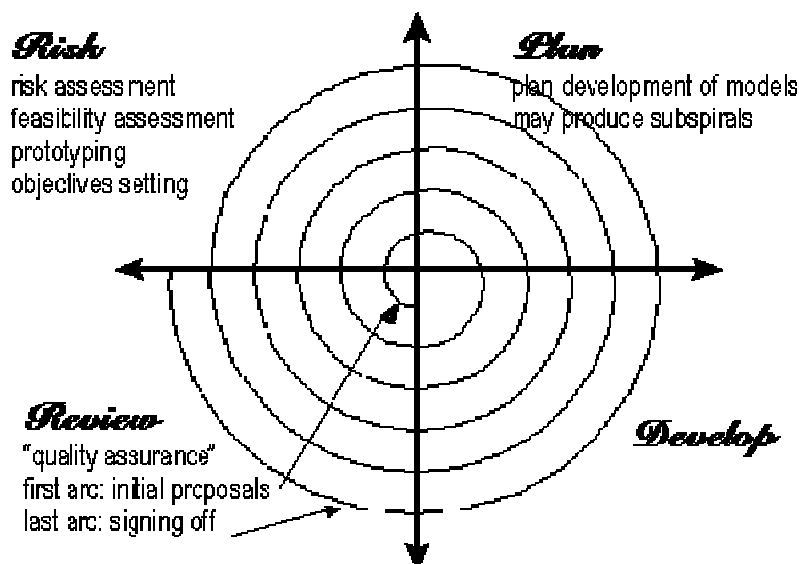


Figur 3.1 Eksempel på Incremental Development model

Denne model kombinerer elementer af waterfall modellen i en iterativ stil. Projektet bliver delt op i små dele, som går horisontalt med tiden.

Den første del af en incremental, er selve kerneproduktet. Denne har de fundamentale metoder og attributter. Som man kan se på [figur 3.1](#) bliver derefter flere og flere funktioner knyttet til kernen, så den udvikler sig til et færdigt produkt. Processen i modeller, som spiral og prototype modellen er iterativ. Men i modsætning til dem, er målet med denne her model, at aflevere et funktions dygtigt software med hver increment. Ulempen ved denne model er, at det kan være svært for kunden at forstå, at der stadigvæk er meget arbejde tilbage, efter den første increment er afleveret. Disse misforståelser med kunden kan undgås hvis man forinden redegør processen for kunden ved at gøre det klar for kunden at der kun er tale om et foreløbigt produkt.

3.1.2 Spiral model



Figur 3.2 Eksempel på spiral model

CDoG – Creative Development of Games

Boehm, som udviklede spiral modellen, så software processen i lyset af de risikoer, som er involveret i processen, og foreslog at en spiral model kunne kombinere proces aktiviteter med risk management for at minimere og kontrollere risikoen.

Spiral modellen, som på [figur 3.2](#), kører efter den iterative model. Processen begynder med behovene og en foreløbig plan for udviklingen (inkl. budget, afgrænsninger og alternativer for personale design og udviklingsmiljø). Før et "concept of operations" dokument er færdig skrevet, bliver processen inddelt i frameworks til evaluering af risiko og alternative prototyper. Disse frameworks giver en generelt beskrivelse af processen og systemet. Af "concept of operations" dokumentet bliver der specificeret et set af behov, som er undersøgt nøje for at sikre, at behovene er så komplette og ensartet som muligt.

Frameworks består af de følgende fem dele: kommunikation, planlægning, modellering, design og færdiggørelse.

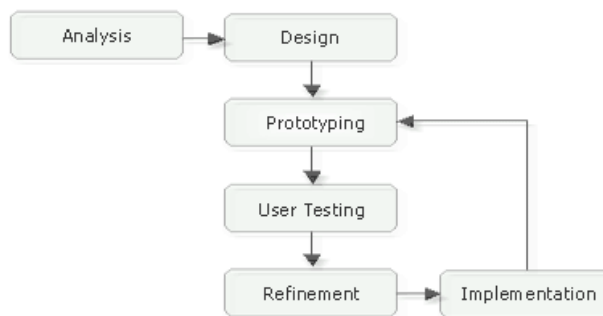
For at se det her fra en anden vinkel kan man sige at "concept of operation" er produktet af den første iterative proces, behovene er produktet af den anden, den tredje er design af systemet, mens den sidste er test fasen. Under hver iterativ proces vægtes de forskellige risk analyser anderledes i lyset af behovene og indskrænkelse. Desuden vurderes prototypernes fordele og svagheder, før en af de alternative former bliver valgt. I modsætning til andre proces modeller, som slutter når softwaren er afleveret, kan spiral modellen køre igennem hele livsprocessen af softwaret. Det vil sige, at man kan komme yderligere spiraler ind i modellen, for at videreudvikle softwaret.

Når en risiko bliver identificeret, må man beslutte sig for, hvordan man vil eliminere eller minimere den. F.eks. designerne er muligvis ikke sikre på, om brugerne vil foretrække den ene form for GUI over for den anden. For at minimere den risiko at vælge den forkerte GUI, kan designerne producere prototyper af hver GUI og køre nogle tests på dem for at se hvilken der er bedst, eller de kan vælge at inkludere begge typer GUI'er i produktet, så brugerne selv kan vælge hvilken de foretrækker.

CDoG – Creative Development of Games

Fordelen ved spiral modellen er at den mindsker risiko faktorene i større projekter som kan have indflydelse på udviklerne og den måde de arbejder på. Derudover kan man køre spiralen igen og igen indtil man er sikker på at produktet er færdigt. Men dette er også en ulempe da det kan være svært at vide hvornår man skal stoppe. Derfor er man nødt til at lave en god definition af hvornår målene er nået. End anden ulempe er at det kan være svært for projektlederen at styre denne model da der kan opstå tvivlssituationer som f.eks. om enkelte risiko situation er værd at tænke over eller om man har lavet nok prototypes. Spiral modellen er bedst brugt ved de projekter hvor usikkerheden omkring et produkt er stor.

3.1.3 Prototyping model



Figur 3.3 Eksempel på Prototyping model

Prototyping model, som man kan se på [figur 3.3](#), er en del af den evolutionære process. Ofte fremsætter kunden krav til software uden at give nogle specifikke detaljer, hvad angår data-output, data-input, databehandling og datalagring. Det kan også være, at kunder slet ikke er klar over hvad han vil have.

Da der er tale om en iterativ model, er Prototyping modellen især god, når det gælder at kunne tilpasse sig ændringer der sker i sidste minut. Selve udviklingen sker trinvis og der holdes reviews regelmæssigt.

Reviews er møder mellem udviklerne og kunden. På disse møder diskuteres projektets nuværende tilstand og hvad man ønsker af funktionalitet og hvordan det skal fungerer.

CDoG – Creative Development of Games

Det er almindeligt, at der på disse møder opstår eller findes nye krav. Der holdes også et møde mellem udviklerne og kunden før selve udviklingen og planlægningen af denne begynder, så de krav der er kendt pt. kan fastlægges. Selve prototyping processen starter med, at man fastlægger nogle krav.

Til de kommende reviews, udvikles en simple funktionel prototype af applikationen, som så kan vises frem for kunden. I denne prototype lægges der vægt på det synlige ”funktionalitet”. Det gælder f.eks. design af den grafiske grænseflade, formatering af input og output data.

Underliggende funktionalitet som database adgang og lign., er ikke vigtige på disse møder. Det er tvivlsomt at kunden besidde nok teknisk viden, til at kunne træffe beslutninger eller fremsætte krav her. Eller også er kunden ligeglad.

Under et review opsamles feedback i form af raffinerede og nye krav. På denne måde opnår man en trinvis software udvikling, hvor det sikres at der ikke afviges fra det kunden ønskede. Softwaren bygges som et hus, sten for sten.

Meget ofte kasseres den første prototype væk og bruges bare som en slags reminder om hvad virkede og hvad ikke virkede. Dog er det ofte muligt, at genbruge dele af prototypen i senere iterationer. Prototyping gør det muligt, at skabe en virkende applikation på kortere tid end f.eks. en traditionel lineær model som vandfaldsmodellen.

Da prototyping første gang blev introduceret midt i 1980erne, viste det sig, at det var en stor forbedring hvad angår kommunikationen mellem udviklerne og kunderne. Reviews minimerede misforståelser mellem parterne og dette førte til, at udviklingsomkostningerne også faldt.

Men som alle andre modeller, har prototyping også sine egne svagheder. Der er altid en risiko for, at nogle kunder kan blive fjendtligt stillede overfor udviklerne, når de præsenteres for en delvis virkende applikation. Personer som ikke har kendskab til

CDoG – Creative Development of Games

software udvikling kan have meget svært ved at forstå, at software udvikling og især quality assurance tager tid. Det kan være svært, at skulle forklare en kunde, at den prototype der er blevet lavet på kort tid, vil tage endnu længere tid før, at den kan betragtes som en færdig applikation. Fra en typisk kundes synspunkt er det jo kun funktionalitet der købes. Kunden har ikke kendskab til omfanget af quality assurance og testing. For de fleste brugere, er den grafiske brugergrænseflade jo programmet.

En anden ulempe kunne være at en algoritme også kan være implementeret via en let skrevet, men stærkt ineffektiv løsning. Ofte for at kunne få en prototype klar til et review møde, benyttes såkaldte beskudte løsninger, som senere skal gøres rene og pæne. Udviklerne skal også passe på, at de ikke bliver fanget i en uendelig løkke af små raffineringer eller at der bruges alt for meget tid på interface design. Hvis dette sker, går det ud over prototypings fordele mht. omkostnings reduktion.

Det skal også nævnes, at prototyping og iterativt design i sig selv er kendt for at være svært at manage.

3.3 Agile Modeller

Som nævnt tidligere er situationen i systemudviklingsverdenen i dag, at systemer alt for tit enten bliver leveret for sent, gået over budgettet eller ikke opfylder alle kravene fra kunderne. Der er udviklere som synes, at de proces modeller som mange følger, nemlig de iterative processer, er alt for overflødige og at de koncentrerer sig mest om værktøjer og modeller og ikke om menneskene der laver projekterne. Man mener, at iterative processer undertrykker kommunikationen med kunderne samtidig med, at de styrer udviklingsfremgangsmåden, uden at udviklerne selv har noget at sige til det.

Derfor blev der oprettet et agilt forbund, der udvikler agile modeller. De agile modeller fokuserer på de individuelle personer og kommunikationen mellem dem, i stedet for processer og værktøjer. Her er det menneskene der er det vigtigste værktøj for projektet. Med agile modeller er der blevet lavet en udviklingsfremgangsmåde der kun opfylder

CDoG – Creative Development of Games

dens formål og ikke mere. Dermed sørges der for at holde det hele så simpelt som muligt og let at forstå. De krav som blev specificeret af kunderne løses, dvs. at man ikke tænker frem.

Modellerne har fem værdier som skal følges og disse er:

- **Kommunikation**, her skal der sørges for, at der god og hyppig kommunikation mellem udviklerne i projektet, samt mellem udviklerne og kunderne. For det er kunderne der kender kravene til systemet. Dermed kan de hjælpe med at rette fejl tidligt i processen.
- **Enkelthed**, der skal laves ens udviklingsmodeller så simpelt som muligt. De nuværende krav skal modelleres og der skal ikke tænkes på morgendagens krav og opgaver. Simple modeller er lettere at forstå både for udviklere og kunderne samt med at de giver et tydelig billede af software opbygningen.
- **Feedback**, her snakkes der om feedback til ens modeller. Hvis der f.eks. laves et klassediagram kan der fås feedback fra andre udviklere og eller vise en prototype til kunden så de kan verificere den. Hermed kan de se om systemet opfylder kravene og om der er krav som skal tilføjes.
- **Mod**. Ved at arbejde tæt med andre mennesker skal man have mod til at stole på disse samtidig med at der stoles på ens egne egenskaber. Agil udvikling kan nogle gange være svær at bruge og der kan opstå modstand både fra ens kollegaer og selve projektet. Her skal der vises mod ved at holde sig til den agile fremgangsmåde og stole på, at ens eget og kollegaernes evner vil sørge for, at projektet i den sidste ende vil lykkes.
- **Ydmyghed**. En person skal være klar over at der er områder i udviklingen, som han ikke selv har styr på og dermed skal klares af andre folk der har erfaringer og ekspertise i disse.

CDoG – Creative Development of Games

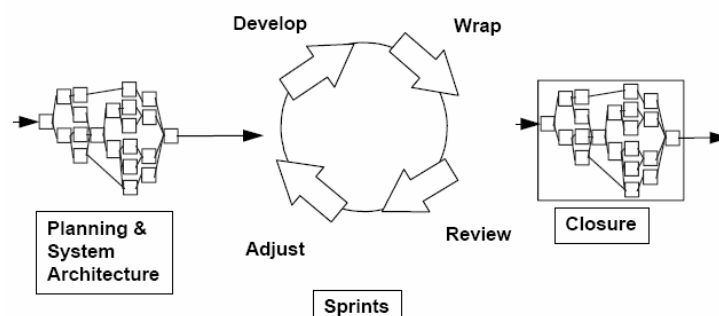
De første fem værdier genbruges fra Extreme Programming processen, som var den største inspiration for agile modeller.

Målet med den agile proces er, at opfylde alle kundens krav til systemet samtidig med, at udviklingen holdes så simpelt som mulig. Da ændringer er en naturlig del af systemudviklingsmiljøet skal ændringer hele tiden kunne foretages.

Måden hvorpå disse mål kan nås er ved at levere systemet i iterationer. Der kan laves en lille del af systemet med de højest prioriterede krav og leverer denne først, før der forsættes med resten af systemet. På denne måde har kunden en del af systemet kørende og kan dermed teste den og fange fejl i en tidlig fase. Samtidig bliver kvaliteten af systemet bedre da udviklerne dermed helt kan koncentrere sig om en bestemt del af systemet i stedet for hele systemet.

Der skal dog bemærkes at en agil model ikke er en komplet udviklingsproces. Dens største fokus er effektivt modellering og dokumentation. Den inkluderer bl.a. ikke test aktiviteter eller projektstyring. Derfor skal den bruges med andre udviklings processer som f.eks. XP eller SCRUM.

3.3.1 SCRUM



Figur 3.4 Eksempel på SCRUM model

Scrum modellen, se [figur 3.4](#), er en agil proces model, som henviser til at der skal arbejdes i små udviklingsgrupper. Hermed sørges der for, at der er optimal og god kommunikation mellem alle de involverende. Hver dag holdes review møder med ca.15

CDoG – Creative Development of Games

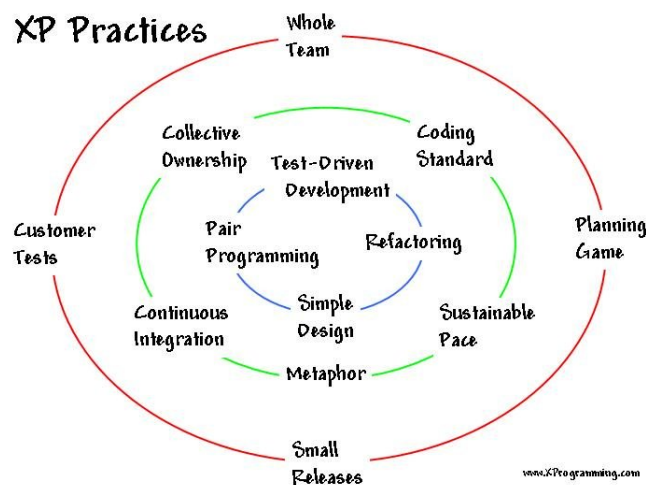
minutters varighed, hvor alle udviklerne fortæller, hvad de har lavet, hvilke problemer de er løbet i og hvad de ønsker at få lavet inden den næste møde.

Selve systemet bliver leveret i såkaldte demos, dvs. i små dele der indeholder de fastlagte krav fra kunderne. På den måde ser kunden, i en tidlig fase, hvilke funktioner der er blevet implementeret, og kunder kan dermed teste disse.

For at håndtere ændringer i krav, laves der i starten en backlog, som er et dokument der viser kundernes krav og dette kan hele tiden blive opdateret med nye krav og ændringer. Dog når dele af dette backlog bliver til en såkaldt sprint, må der ikke blive introduceret nye krav da selve udviklingen af denne del af systemet er startet.

Ulempen ved SCRUM modellen er at der er brug for meget projekt ledelse. Lederen skal hele tiden have styr på de mange små grupper der findes og dette kan være en kompleks opgave. Derfor er en projektleder nødt til at give meget mere ansvar til grupperne samtidig med at han skal sørge for at grupperne overholder planerne. Derudover er modellen rimelig ny så der kan være udviklere der yder modstand mod disse ændringer. For SCRUM giver meget mere ansvar til udviklerne hvilket nogle mennesker ikke kan lide.

3.3.2 Extreme Programming



Figur 3.5 Eksempel på XP model

CDoG – Creative Development of Games

XP, [figur 3.5](#), er en del af de agile udviklingsmetoder, som er blevet forholdsvis populær i de seneste par år i udviklingsverdenen, da det er en metode som kan være utrolig effektiv til mindre projekter. De er meget givende for de enkelte programmører da man kan lære utrolig meget af at benytte XP.

Der snakkes om at benytte XP i spiludviklingen, eftersom XP er god til at sikre at udviklingen forbliver kreativ. Samtidig fås der en del samarbejde imellem de to programmører da der altid er en makker at konsultere, hvis en er i tvivl om en ting og kan diskutere, hvad den optimale løsning til et givent problem er.

XP består af 4 faser nemlig planning, design, coding og testing. Disse fire faser er de ting som normalt bruges i udvikling planning. I XP er det dog lidt anderledes end normalt i softwareudvikling eftersom kunden udfylder nogle kort med en beskrivelse af forskellige ønsker af hvad der skal laves, med en prioritering over hvor vigtig det er for kunden. Hvert kort bliver så vurderet af en programmør og hvis opgaven tager over en vis tids mængde, skal kortet splittes op til det overholder tids mængden. Dette betyder, at når der laves et projekt via XP, fås der en masse små opgaver som bliver udført på relativ kort tid. Dette vil være en fordel når der udvikles spil, da der i et spil er en masse opgaver som skal løses og dermed er der risiko for at det vil virke uoverskueligt, hvilket så kunne blive bedre ved at en stor opgave deles op i mindre dele.

Design fasen i XP benytter KIS (Keep It Simple) modellen, hvilket gør, at der ikke gøres noget unødigt arbejde, hvilket generelt vil være en god ting da et simpelt design altid vil vinde over et stort kompleks design.

Coding fasen i XP er ret forskelligt fra de fleste andre modeller da den bruger par programming, hvor der sidder 2 personer og programmerer ved den samme computer, hvor den der ikke sidder ved tastaturet læser koden igennem og kommer med forslag, og andre måder at gøre det på. Dermed kommer der en dialog i gang og derved fås den optimale kode og eventuelle tvivlsspørgsmål bliver elimineret. Dog bliver der benyttet en

CDoG – Creative Development of Games

anden type af par programming som går ud på, at der er en som står for den del af koden og han har eventuel en makker, som han sætter til at udføre selve programmeringen efter anvisninger. Dette gør det nemmere at reviewe koden bagefter og er en effektiv måde til at lære nye folk op. Desuden bruges der i XP en del refaktorering, hvor koden opdateres, hvis nye ideer kommer, og ændrer koden så den hele tiden passer sammen med de andre dele af projektet.

Testfasen i XP går ud på, at de skemaer kunden selv har udfyldt med funktioner bruges til at lave en test modul der tester om funktionerne fungerer på den rigtige måde. Desuden testes der kontinuerligt efter en funktion er lavet, så der hele tiden kan følges med i om tingene virker tilfredsstillende.

XP er en udmærket procesmodel til mindre projekter, hvor der hele tiden er et overblik over hvad der skal laves og hvor der skal overskues, hvordan systemet hænger sammen. Den er mest anvendelig i mindre software grupper og med en forholdsvis lille tidshorisont.

En af ulemperne ved XP kan være at kunden altid skal være tilstede hvilket enkelte kunder ikke kan lide eller har tid til. Derudover er der brug for meget disciplin hvis man skal kunne følge alle principperne og værdierne i XP, hvilket kan være svært for enkelte udviklere. Derfor kan der være udviklere der skaber modstand mod XP og på den måde skaber en risiko for projektet. Desuden koncentrerer sig XP mest om kodningen og ikke analyse og design. Derfor kan det være nødvendigt at blande den med en anden model hvilket kan være forvirrende og svært at følge.

3.4 Delkonklusion

Agile moduller kan til en stor del bruges i spilbranchen. De sætter ikke udviklerne i nogle faste rammer, så der ikke kan være kreative tankegang i firmaerne. Men der bliver stadigvæk, især i Scrum, lagt vægt på dokumentation, så man har en form for overblik

CDoG – Creative Development of Games

over hele situationen. Hvad angår de iterative modeller, som har en del mere fast styring, kan man f.eks. ved spiral modellen udnytte at processen, i teorien, kan fortsætte i det uendelige. Derved kan man køre rundt indtil projektet er færdigt.

I vores model vil vi tage nogle af fordelene af de iterative, samt de agile modeller, for at skabe en model, som kan udnyttes til stort set alle i spilbranchen.

CDoG – Creative Development of Games

4. Virksomhedsbeskrivelse

Spiludviklingsvirksomheder har ikke haft de bedste resultater med at bruge egentlige processer som deres udviklings fremgangsmåde. De fleste udviklere, som arbejder i denne branche har ikke en relevant systemudviklings uddannelse og erfaringer og derfor har de ikke en grundlæggende kendskab til SE (Software Engineering) udviklingsværktøjer og modeller, som kan hjælpe dem med at lave et godt produkt på den mest produktive måde.

I dag bruges der mest metoden code and fix, hvor programmet bare kodes og håber på at det går godt. Da der naturligvis er ulemper ved denne metode har virksomhederne prøvet at køre med lineære proces modeller som f.eks. vandfaldmodellen, men dette har ikke givet de ønskede resultater, da modellen ikke passer til dette udviklingsmiljø.

Hvordan foregår den egentlige spiludvikling i en virksomhed? Inden selve processen starter skal man finde ud af hvilken slags spil man ønsker at udvikle. Dette gøres ved både at kigge på udviklernes interesse men også ved at overvåge markedet. For at kunne udgive spillet, skal der en udgiver til, som er interesseret i at investere i dette spil. Og udgiveren ønsker selvfølgelig, at tjene penge på dette og derfor er de mest interesseret i at investere penge i de slags spil som er mest populære på markedet.

Efter at have fundet ud af hvilken type spil som ønskes at laves, sørges der for at der laves en tidsplan. Der prøves at skabe et overblik over hele processen ved at dele den op i trin. Ud fra disse trin estimeres, hvor meget der skal bruges til hver og dermed fås den endelige tidsplan der indeholder hvornår projektet skal færdiggøres og arbejdsfordeling. Arbejdet fordeles ved at opdele udviklerne i grupper efter speciale. Men der er dog en stor cirkulation mellem grupperne, da der på den måde sørges for, at udviklerne har styr på de fleste områder af projektet og på den måde kan afløse andre, hvis dette bliver nødvendigt.

CDoG – Creative Development of Games

Det er dog meget svært for virksomhederne at overholde disse tidsplaner. Grunden til dette er, at der tit opstår situationer, hvor tidsplanen siger at projektet skal være færdigt, selvom der er enkelte features af spillet som ikke er færdige. Mens der i andre udviklingsmiljøer vælges at skære i features i stedet for at forøge tiden, er dette ikke tilfældet med spil udvikling. For features er det der sælger spillet så virksomhederne vil hellere bruge mere tid og penge på at forøge tidsfristen i stedet for at skære i produktets kvalitet.

Selve udviklingen af spillet sker ved at bruge lidt processer, men ikke meget, da der som nævnt før ikke kendes nogle som passer til dette. Der snakkes dog meget om agile metoder i spil udviklingsbranchen da der menes at metoder som f.eks. Extreme Programming og SCRUM måske kunne løse deres problemer.

Virksomhederne bruger spillerne til at hjælpe dem med kvalitetssikring af produktet. Dette gøres ved at invitere både erfarne og uerfarne spillere, til at spille spillet og dermed teste den.

Men som situationen ser ud nu så er de firmaer ikke så glad for processer da de er bange for at de hæmmer kreativiteten, og derfor prøves der at bruge så lidt proces som muligt. Derimod prøves der at ansætte folk med kreativ kunnen.

4.1 Interactive Television Entertainment

ITE er et spiludviklings virksomhed som bl.a. udgiver Hugo spil serien. De har 15års erfaring med dette, hvor de bl.a. har udviklet Hugo spil som TV2 brugte i et af deres programmer, hvor der folk kunne ringe ind og spille spillet over telefonen. Nu er markedet for disse slags tv spil ikke så populær som før og derfor har ITE videreudviklet deres produktion så nu udvikler de Hugo spillene til både PC, Playstation og Nintendo Gameboy. Afdelingen i København består af 40 medarbejdere. Projektlederen som gruppen snakkede med var tilknyttet Playstation 2 holdet der består af 16 personer. Af disse er 7 koder, 6 grafikere, en designer, en projektleder og en Art Director.

CDoG – Creative Development of Games

ITE er afhængig af julehandlen. Dette betyder at de hvert år skal leverere et nyt Hugo spil til julehandlen. Hermed har de ca. 1 år til selve udviklingen af spillet. Men da Sony, der ejer Playstation, har strenge kvalitetskrav betyder det at ITE skal aflevere spillet, før udgivelsen til Sony, hvor de har 5 uger til at teste den. Hvis det sker, at Sony finder en fejl, sendes spillet tilbage til ITE som retter fejlen og så må Sony igen bruge max. 5 uger til at teste spillet igen. Dette resulterer i at ITE egentlig kun har omkring 6 måneder til selve udviklingen af spillet, hvor resten af året bruges til bl.a. test hos Sony.

Som det kan ses, har ITE en slags fast tidsplan for hvert år, når det handler om Hugo spil til PS2. I starten brugte Playstation afdelingen den lineære vandfald model. Men denne passede ikke til deres behov da de dermed ikke kunne overholde tidsplanen og spillet ikke blev leveret til tiden. Modellens fremgangsmåde gjorde at medarbejderne blev alt for stressede lige fra starten af udviklingen og især i slutfasen. Der var alt for meget overarbejde og til sidst brugte hver medarbejder mellem 600 til 800 timer på bare overarbejde. Dette resulterede i at mange forlod virksomheden og dem der blev var alt for trætte til at starte den næste spil. Derfor besluttede projektlederen at gå over til Extreme Programming (XP) modellen.

Efter at det blev godkendt fra ledelsen om at bruge XP opstillede PS2 gruppen 12 regler, eller såkaldte XP dogmer som de skulle følge. Disse tager udgangspunkt i principperne fra XP.

1. Kunden skal altid være til stede.

Da ITE ingen kunder har så er det projektledere, Art Direktoren og designeren der spiller rollen som kunden. Udviklerne kan altid komme i kontakt med dem for at få svar på spørgsmål.

2. Korte iterationer, hyppige releases

For at kunden kan være i stand til at vurdere kvaliteten af spillet skal denne have leveret dele af spillet løbende. Dette gør ITE ved at de har korte iterationer der varer mellem 2-3 uger. Hver iteration bliver testet før den opfattes som færdig

CDoG – Creative Development of Games

og så fortsættes med den næste. Dette resulterer i at man tidligt i processen finder fejl og får dem rettet.

3. Planning game

ITEs opfattelse af softwareprojekter er at planlægningen skal være kortsigtet og detaljeret. For at opfylde dette holder projektgruppen møder hver 14. dage hvor alle medarbejdere er til stede. Her har de mulighed for at sige deres meninger og forslag til spillet. Dette gør folk også mere motiverede da de føler at de får lov til at involveres sig. På mødet gennemgår man det folk har lavet i de sidste 14 dage og samtidig laves der en plan over hvem der skal lave hvad i de næste 14 dage.

4. Metafor

For at fremme forståelsen mellem de tekniske og forretnings samarbejdspartnere skal disse kunne forstå hinanden indbyrdes. Derfor prøver ITE at beskrive projektet overordnet og med almindeligt sprog.

5. Enkelt design

ITE prøver med at holde koden og processen som enkelt som muligt. Derfor skal designet også holdes enkelt og beskrivende med så få klasser og metoder som muligt. På denne måde sørger de for at ændringer er let at lave uden at det går alt for meget udover det eksisterende design. Design dokumentet er simpelt. F.eks. fortæller den at en bane skal indeholde en kanon men ikke hvordan denne skal se ud og hvad den skal gøre.

6. Par programmering.

Par programmering er en stor del af XP men ITE kører ikke med dette. De forsøgte med par programmering i tre dage men programmørerne kunne ikke lide den fremgangsmåde så projektlederen valgte at afskaffe det.

CDoG – Creative Development of Games

7. Kode kommunisme, fælles kode ejerskab

Alle skal røre og se koden. Dette gør at medarbejderne har kendskab til alle dele af spillet og dermed kan afløse hinanden. Dette gør at når en medarbejder bliver syg så kan en anden træde ind i stedet for. Samtidig er arbejdsfordelings fordelt sådan at medarbejderne tit skifter hold så de kan lave noget andet.

8. Kontinuerlig integration

Modulerne bliver integreret dagligt så alle kodere integrerer deres kode sammen, hvilket sikrer at releases kan ske dagligt. På denne måde sørges der for at integrationsproblemerne minimeres og i tilfælde af konflikter skal der kun rulles en dag tilbage i versionen.

9. Kodestandard

ITE prøver at opretholde en kode standard ved at de har faste regler for navngivning af variabler og hvordan man skriver sin kode, samt hele strukturen i koden. Dette gør at læsbarheden for koden øges. Samtidig skal der ikke laves så meget dokumentering til koden.

10. Automatiske test

Hver modul har tilføjet en automatisk test som tester netop dette moduls funktionalitet. Disse bliver skrevet inden selve programmeringen af modulet hvor udviklerne diskuterer hvilke funktioner modulet skal opfylde.

11. Refaktorering

Ved refaktorering opnår ITE en trinvis forbedring af kodebasen. Det er tilladt at eksperimentere med ny kode og lade det blive en del af den foreløbige version af spillet. Hvis koden bliver accepteret til at blive i versionen bliver koden revideret så den er robust og passer ind i kodestandarden. Samtidig skal den også kunne bestå den automatiske test som blev lavet til den i starten.

CDoG – Creative Development of Games

12. Max. 37-timers arbejdsuge

Det er projektlederens mål at sørge for at alle medarbejdere max. Arbejder 37 timer om ugen og ikke har for meget overarbejde. ITE vil meget gerne beholde de nuværende medarbejdere og derfor mener de at dette kan gøres ved at motiverer medarbejderne ved at lade dem være en del af processen og samtidig sørge for at de ikke stresser alt for meget.

Hver anden uge bliver en cd/dvd kopi af den foreløbige version brændt og sendt til quality assurance afdelingen som tester om spillet lever op til design dokumentet. Dvs. om den opfylder alle funktioner som blev specificeret. Derudover bliver spillet, før den udgives, testet af en børnehave som ITE har en aftale med. Her kigges på om børnene kan finde ud af banerne og om de egentlig gider spille spillet.

Skiftet af udviklingsmodellen har betydet at ITE har kunnet leverere til tiden. Samtidig blev medarbejderne meget mere motiverede og mindre stressede. De oplevede at XP gav medarbejderne mulighed for at bestemme mere over processen. De var ikke låst til enkelte faser ligesom man var med vandfaldsmodellen.

Ideerne for spil får ITE medarbejderne ved at lade sig inspirere af andre spil og samtidig kigge på markedet for børnespil. De prøver at lave spil som er rettet både til børn men også de lidt ældre børn. Da ITE er deres egen publisher finder de på en koncept og denne skal accepteres af ITEs ledelse. Inden selve udviklingsprocessen starter findes rammer i spillet, som giver en generel billede af handlingen. Derefter holder man en brainstorming møde hvor alle udviklere deltager og kommer med ideer til spillet.

Projektlederen laver ingen egentlig tidsplan ligesom f.eks. Gannt chart. I stedet for laver han en tidslinie der viser de trin som man skal gøre. Denne vises så til ledelsen der skal acceptere den. Derefter bliver hver trin yderligere beskrevet til de møder som holdes hver 14.dag. Projektlederens mål er at tilpasse XP modellen til hver medarbejder da der er store forskelle mellem dem. Efter hvert spil bliver udviklingsprocessen evalueret af medarbejderne.

CDoG – Creative Development of Games

Samtidig har projektlederen et godt kendskab til medarbejdernes personlige forhold. For at der er godt sammenhold på holdet holdes der tit hyggedage.

Som det ser ud så har ITE haft gode erfaringer med XP. Men de mener selv at XP i deres organisation er stadigvæk under udvikling da der skal bruges lidt mere erfaringer før den kan perfektioneres til deres behov.

Vurdering

Da virksomheden har været på markedet i mange år er deres udvikling fremgangsmåde meget stabil og optimal. Projektlederen har helt styr på de forskellige modeller der findes og han har udviklet en variation af XP der passer til hans medarbejdere. De har dog helt klart en fordel da udviklings processen er sådan set den samme for hvert spil. De har en fast udgivelse dato hvert år, så sidste års proces kan genbruges og samtidig forbedre den ved at rette de svagheder som de har oplevet. Og dette har de helt klart gjort da man får det billede af at de lærer fra deres fejl.

4.2 MediaMobsters

MediaMobsters, som har ændret navnet til WorldSimSoft, er et lille spillefirma, som startede omkring 2002 med at udvikle deres første spil ”Ganglang”. Nu er de i gang med ”Gangland 2” og noget mere seriøs software, som de udvikler til bl.a. politiet.

De laver en masse brainstorming til hvilket spil der skal udvikles. Derefter udarbejder de demo’s af deres ideer, som de så sender til forskellige udgivere i verdenen. Når de får en aftale med en af udgiverne går de fuld i gang med at udvikle det spil, som udgiveren er interesseret i. Desuden bliver løn og ”den midlertidige deadline aftalt”.

Et design dokument bliver udviklet af dem. Denne indeholder hvilke ting, som skal udvikles, hvordan de vil have alt til at fungere, og tidsplanen. Denne fylder i deres tilfælde omkring 300 sider.

CDoG – Creative Development of Games

Afdelingen i København består af ca. 20 personer hvor alle arbejder på spillet Gangland. Af disse 20 er 2-5 grafiker, 6 koder (3 engine + 3 til ganglang), 2 designere, producere og 4 dba (Database analytikere). Medarbejderne arbejder på faste pladser det meste af tiden. Der sker rokeringer, hvis nogle medarbejdere har travlt. Så kan de flytte folk som ikke har så meget at lave til det travle område. De ansatte har faste roller, bestemt efter deres evner.

Folk har frie hænder til at udvikle deres kreativitet til en grad. Ved deres første spil havde en person flere roller. F.eks. kunne en designer også være en producer/leder. Dette gav interessekonflikter. Dette har de dog nu lavet om på, så de ”kreative folk” ikke har den endelige beslutning mere.

Udviklingen kører i faste rammer, hvor det vurderes at de bruger ”code & fix” modellen hvor de programmerer og eftersom der dukker problemer op løser de dem. Desuden har de ikke nogen faste arbejdstider. De koder indtil de ikke kan mere. Derudover benytter de sig af en Objekt Orienteret fremgangsmåde og en variation af XP.

Virksomheden har desuden planlagt, at outsource noget af deres programmering især grafik. Dette gør de for, at de ikke behøver at lave den kedelige rutine arbejde. Desuden kan disse kodere ikke lave noget, mens spil-enginen er under udviklingen, så firmaet kunne spare penge her ved, at hyre nogle freelance arbejdere når tiden er inde til den del af spillet.

Da de er i gang med en efterfølger til deres første spil, har de planer om at spare penge og tid ved at genbruge noget af deres kode. De bruger f.eks. deres gamle motor (men med nogle forbedringer).

Deres tidsplan er noget ”løst”. De har møder ca. 1 gang om ugen for at diskutere om hvordan det skrider frem, og om forandringer i spillet og design dokumentet. Desuden har de møder med deres udgiver 1-2 gange om måned for at fremvise hvor langt de er og for at se om de overholder deres forventninger eller om der skal ske forandringer i spillet.

CDoG – Creative Development of Games

Hvad angår kvalitets sikring er dette udgivers ansvar. Så dette beskæftiger de sig ikke med.

Mens spillet bliver udviklet har de oprettet en hjemmeside, som fortæller om deres spil og hvad det indeholder. Desuden havde de til Gangland 1 et debatforum hvor brugere kunne udtrykke deres meninger om spillet og måske bidrage med ideer til udviklerne. Dette blev benyttet i mindre grad af firmaet. Efter spillet er udgivet og der kommer feedback om spillet, bliver nødvendige rettelser og features lavet og udgivet som patches.

Til Gangland 1 brugte virksomheden venner og familie til at teste spillet. Dette var ikke ligefrem positivt, da det blev mere sjov og ballade end seriøs testning af spillet. Desuden havde de en aftale med en amerikansk test firma. Dette var heller ikke ligefrem positivt da det var svært at kommunikere på grund af tidsforskellen. Desuden var dette firma ikke ligefrem effektivt.

Hvad angår Gangland 2 går de faktisk nogenlunde samme vej, med undtagelse af at test firmaet er stationeret i Europa denne her gang.

Lønnen får de fra deres publisher, så når de har fremskaffet sådan en, så behøver de kun at tænke på at færdiggøre spillet. Desuden har firmaet valgt at have en ekstra indtægtskilde, ved at samarbejde med bl.a. politiet om fremstilling af software til dem. Hvis spillet har succes får de desuden en bonusindtægt fra salget.

Firmaet er et delvis ungt firma, som lige har udviklet deres første spil, og er i gang med efterfølgeren. De har ikke haft noget erfaring med udvikling af spil inden de begyndte, og dette kunne også bl.a. ses på deres ”fleksibel deadline”, som kan ændre sig et par gange. De går meget kreativt til værks og laver de spil som de har lyst til. De fremstiller små demoer af spil, som de har en interesse i at lave, og sender dem derefter til forskellige udgivere hvorefter, når en bider på krogen, de går i gang med det samme.

CDoG – Creative Development of Games

De kører efter "code & fix" princippet, men følger et design dokument, som de udarbejder efter udgiver er fundet. Denne bliver selvfølgelig løbende revideret eftersom man får nye ideer eller skal begrænse sig.

Strukturmæssigt foregår det meget hierarkisk hvor de kreative folk (koder og grafikere) ikke har de sidste ord. Firmaets leder er den såkaldte producer, som har hovedansvaret for projektet. Der foregår møder hver uge for at revidere de ting, der er sket og hvad der skal ske senere. Desuden bliver "milestones" fremvist til publisherne 1-2 gange om måneden.

Vurdering

Da det er et relativt nyt firma, var der en masse "børne sygdomme" som de skulle igennem ved deres første spil. De benytter efter vores mening "code & fix". Dette er måske ikke den mest optimale metode løsning. Desuden er der ikke rigtig styr på processen. De kører derudaf med kun et design dokument som vejleder. Hvad angår deadline har de ikke nogen indflydelse, men de kunne lave en mere nøjagtig tidsplan når de har fået deadlineen at vide. Desuden kunne de estimere bedre hvor lang tid de forskellige processer vil tage ved måske at benytte critical path.

De kreative ansatte har lidt for frie hænder, hvad angår strukturering af processen.

Desuden kunne de godt bruge mere tid til at gennemgå brugernes forventninger og ideer til spillet (da det er dem, der i sidste ende skal "overtales" til at købe det"). Så alt i alt kan der laves en del om for at forbedre styringen og "flowet".

4.3 Del Konklusion

Ud fra virksomheds samtaler har gruppen fået en bedre forståelse for spiludviklingsprocessen. De to nævnte virksomheder har helt klart forskellige fremgangsmåder men dette er forståeligt da de arbejder på helt forskellige måder og har forskellige målgrupper.

ITE er et erfaren firma som selv udgiver deres egne spil. Dette giver dem lidt mere frie hænder og mulighed til selv at bestemme, hvilket vej spillene skal gå. Pga. deres erfaring

CDoG – Creative Development of Games

har de en proces der faktisk dækker deres behov. Men de er meget pressede hvert år da de ikke har så meget tid til at udvikle. Dette medfører at medarbejdernes kreativitet er begrænset da de ikke helt kan vise deres potentiale da der ikke er nok tid.

Sammenlignet med ITE så er Media Mobsters nybegyndere. De har kun udgivet et spil og er i gang med deres anden. Da de ikke er deres eget udgiver betyder det at de er afhængige af succes af et spil. De bruger ingen egentlig process model men man får en fornemmelse af at det kører lidt med code and fix. Hele processen kører ud fra en design dokument som i deres tilfælde består af ca.300 sider. Ud fra den lave de arbejdsfordeling og en slags tidsplan. Da virksomheden er ny er det selvfølgelig naturligt at de behøver mere erfaring med udvikling for de kan finde deres egen stil, herunder også proces model.

CDoG – Creative Development of Games

5. CDoG Modellen

Dette afsnit vil beskrive CDoG modellen. Vi vil beskrive hvordan modellen er opbygget og argumentere for hvordan den kan hjælpe spil udviklerne med at udarbejde tidsplaner og arbejdsfordeling for projektet, dele udviklingen op i moduler og til sidst integrere disse til et færdigt spil. Derudover vil der gives forslag til værdier og principper som udviklerne skal tage højde til under udviklingen.

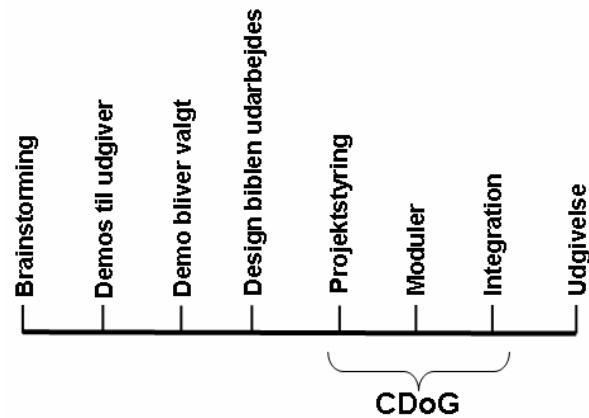
5.1 Forudsætninger for brug af CDoG

Inden CDoG modellen kan tages i brug, skal udviklerne færdiggøre en design bibel. En design bibel i spil udvikling er selve drejebogen for spillet. Indholdet af denne afhænger af, hvilken slags spil der laves, men som udgangspunkt indeholder den generelle specifikationer, både grafiske og tekniske samt brugsmæssige krav til spillet. Den beskriver handlingen i spillet, beskrivelse af dens karakterer og deres væremåde, beskrivelse af banerne og beskrivelse af gameplay. Kort sagt er det udviklernes bibel for den kommende udviklings periode.

Sammensætning og udarbejdelse af design biblen bliver ikke dikteret af vores model. Grunden for dette er at hvert spiludviklings virksomhed har deres egen fremgangsmåde for udarbejdelse af denne.

Dette betyder så at der forudsættes at design biblen er udarbejdet inden selve udviklingen påbegyndes, dvs. inden CDoG proces modellen anvendes (se [figur 5.1](#)) CDoG modellen gør det muligt at udviklerne, når de begynder med modul frameworket, kan gå tilbage til design biblen og lave ændringer. Dette er en naturlig fremgangsmåde, da udviklerne hele tiden får nye ideer og rettelser som er værd at tage med i spillet.

CDoG – Creative Development of Games



Figur 5.1 CDoGs placering i udviklingen

5.2 Introduktion

Vi lever i en globaliseret verden hvor alt går meget hurtigt. Dette gælder også for software udvikling. Konkurrencen på markedet er meget hård og virksomhederne er nødt til at udvikle systemer så hurtigt som muligt, samtidig med at de skal være af god kvalitet. For hvis ikke de kan levere til tiden, går kontrakten til andre og virksomheden mister penge. Ofte sker det at software virksomhederne giver kunden en alt for optimistisk leveringstid, som de ikke kan opfylde, for at sikre kontrakten. Leveringstiderne kan ofte kun overholdes hvis medarbejderne har meget overarbejde, hvilket så resulterer i at omkostningerne stiger.

Derfor bruger de fleste systemudviklings virksomheder proces modeller når de udvikler. Disse hjælper dem med, at få et bedre overblik over problemet, samtidig med at de giver forslag og værktøjer til, at lave tidsplaner og overholde disse samt, hvordan man kan håndtere ændringer. Når udviklerne bruger disse modeller bliver deres udviklingsproces meget fastlåst. Dvs. at der ikke er tid til at komme med nye ting og lave de helt store ændringer. Hvis man mangler tid bliver der, det meste af tiden, skæret ned i features i systemet. Man har en afleveringsdato som mål og den vil man helst overholde 100 %.

Men dette er ikke situationen i spil udviklings branchen. Selvfølgelig vil de også gerne overholde leveringsdatoen men, hvis det betyder, at spillets kvalitet og udviklernes

CDoG – Creative Development of Games

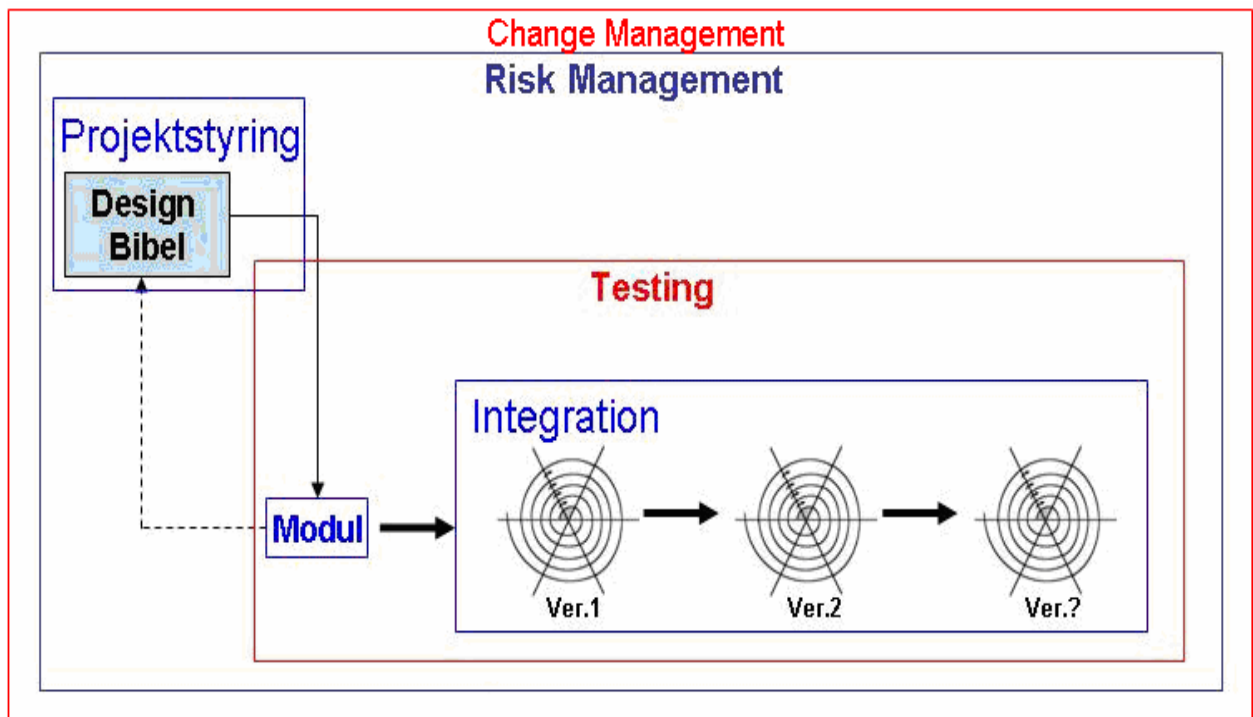
kreativitet bliver hæmmet, ønsker de ikke at følge de proces modeller, som de andre udviklere bruger. For i spil er det features der gælder og de spil med de bedste og flotteste baner bliver også de mest succesfulde. For dem er gameplay og indhold vigtigere end streng styring af udviklingsprocessen.

Arbejdsprocessen er også lidt anderledes i spil udvikling. Mens man, i de store virksomheder som udvikler f.eks. økonomistyringssystemer, ofte har en streng og konservativ organisation af arbejdsgang med faste møde tider og faste arbejdsroller, så er spil branchen præget af et uorganiseret net cafe miljø, hvor udviklerne møder op når de vil og ”spiller spil hele dagen”.

Dette betyder, at spil branchen har anderledes behov til proces modeller end den mere almindelige software branche. De skal have en model der forbedrer styringen af udviklingen, uden at det går ud over kvaliteten af spillet og udviklernes kreativitet. Disse krav vil vi nu opstille en model for. Modellen, se [figur 5.2](#), har vi valgt at kalde CDoG modellen, som står for ”Creative Development of Games”. Som man vil se i de næste par afsnit er modellen meget inspireret af bl.a. XP og Spiral modellerne².

² [SEP], [SET], [ESE]

CDoG – Creative Development of Games



Figur 5.2 CDoG Modellens opbygning

I denne model er det forsøgt, at opstille en iterativ proces, hvor der hele tiden er åbent for sidste øjeblikke ideer som ofte opstår i spilbranchen. Dette var nødvendigt for ikke at hæmme den kreative del. De traditionelle processer er ofte kendetegnet af, at man har bestemte faser, hvor der efter et bestemt punkt er nået, ikke længere er mulighed for nye input. Modellen prøver at sikre, at der hele tiden er en ”spilbar” udgave der evt. kan fremvises for en stakeholder.

Modellen har vi delt i tre frameworks aktiviteter:

- **Projektstyring**, hvor projektlederen i samarbejde med de andre udviklere, laver en tidsplan over processen samt arbejdsfordeling.
- **Modul**, hvor selve udviklingen af spillet går i gang, hvor hver gruppe får moduler som de skal udvikle. Modulerne findes ved at fordele dele af designbiblen til grupperne som derefter deler disse op i moduler.
- **Integration** hvor modulerne bliver samlet og testet.

CDoG – Creative Development of Games

Udover disse frameworks så er der tre paraply aktiviteter:

- **Testning** er en aktivitet for modul og integration frameworks.
- **Risk Management**, som er en paraply aktivitet for hele modellen.
- **Change Management**, som også er en paraply aktivitet for hele modellen.

Desuden indeholder modellen værdier og principper som vi mener, vil være fordelagtige for spil udviklere. Værdierne gælder for hele proces modellen, mens hver enkelt framework aktivitet har dens egne principper.

5.3 Værdier for CDoG modellen



Figur 5.3 Værdier i CDoG

For at gøre et godt stykke arbejde i spilbranchen, skal de forskellige medarbejdere have nogle generelle værdier som de følger og lever efter. Disse værdier skal skabe en arbejdskultur som opfylder både de menneskelige og forretningsmæssige mål. Værdierne er inspireret af XP modellen minder udviklerne om at de skal følge normal sund fornuft når de udvikler. CDoG indeholder følgende værdier:

CDoG – Creative Development of Games

1. Kommunikation

Der skal søges for at der er en god kommunikation mellem alle parter på projektet.. Derudover skal kommunikationen være uformel og åben uden nogen form for bureaukrati.

Eksempel er, at en fra engine afdelingen skal snakke med en fra design afdelingen. I stedet for at snakke med lederen først omkring emnet, går personen direkte til designeren. Hvis emnet er for stort til at de to parter kan løse det selv, skal det gå videre til det næste møde, hvor alle så kan komme med noget konstruktivt til det.

2. Enkelhed

Der skal sørges for at gøre tingene så simple så muligt. Dog uden at det bliver for oversimpelt.

3. Feedback

Alt fra ideer, design, til kodeliner. Modtagerne af feedback, skal kunne bruge dem til noget, så de kan forbedre det de har fået feedback på.

4. Samarbejde og Mod

Man skal kunne tage initiativet til de ting man er i gang med. Derudover skal man have mod til at sige sin mening til bl.a. de ting der er oppe til debat så man også har indflydelse på udviklingen. Alle skal have lov til at blive hørt, da det nogle gange kan lige være ens mening der afgør om udviklingen bliver en succes eller fiasko. Dette betyder at udviklerne skal respektere og være åbne overfor hinandens forslag, uden at nedgøre dem. Desuden skal man være en teamplayer da udviklingen af et spil kræver en masse folk, som kan arbejde godt sammen. Så derfor dur det ikke, at man kører sit eget løb og tror at man kan lave det hele selv, men i stedet accepterer at man arbejder godt sammen i et team.

CDoG – Creative Development of Games

5. Kreativitet

I spilbranchen er det alfa og omega, at udviklerne af spil er kreative. De skal kunne skabe noget, som folk gider købe og spille. De skal være nytænkende og skabe noget, som ikke allerede findes eller som er bedre end de eksisterende spil. Input behøver ikke kun at komme fra udviklerne, men også fra brugerne (kunderne). Udviklerne bør holde øje med, hvad fagpressen og kunderne siger om deres tidligere publikationer.

6. Kultur

Der skal være en åben stemning, hvor man kan sige sin mening og komme med ideer. Desuden skal der ikke herske et alt for styrende miljø, hvor man kan risikere at kreativiteten lider. Desuden skal folk kunne lide at være på stedet og have det sjovt, mens man udvikler. Det er vigtigt, at der hersker et miljø, hvor man motiverer hinanden til at yde sit bedste og hvor man kan støtte og hjælpe hinanden.

7. Fleksibilitet

Det er vigtigt med fleksibilitet i denne her branche. Udviklingen ændrer sig hele tiden, så folk skal kunne følge med når der sker ændringer i spillet eller når der sker nyskabelser i grafik-verdenen. Desuden skal folk være meget fleksible med, hvad de arbejder på, da det er muligt at de i den ene uge arbejder på en del af projektet, hvor de ugen efter arbejder på en hel anden del af projektet. Så de skal være klar til at omstille sig til at arbejde i forskellige grupper.

8. Lederskab

Det er vigtigt her at der ikke er den sædvanlige ledertype, som ånder en i nakken hele tiden. Lederen skal kunne motivere sine folk til at gøre deres bedste. Han skal være åben for alting, alt fra ideer til deres meninger om ting. Desuden skal lederen være meget fleksibel. Men selv om lederen skal være ”en af dem” skal han også tage beslutninger og have en vis autoritet.

CDoG – Creative Development of Games

9. Åbenhed

Man skal søge for, at der er en åbenhed generelt. F.eks. kan alle mennesker have dage hvor de bare ikke har energi eller lyst til at arbejde. I stedet for at komme med alle mulige undskyldninger, skal man bare sige det ligeud. Og næste dag så komme fuld af energi tilbage til arbejdet og fortsætte med at gøre sit. Det man skal sørge for er at medarbejderne ikke har for mange af disse dage.

5.4 Risk Management

Risk Management³ er en paraply aktivitet⁴, hvilket betyder at den kører under hele modellen. Risk management benyttes til at identificere risikoer, som kan dukke op under udviklingen af et spil. Det er projektlederen der sammen med hans udviklere skal diskutere og identificere risikoerne (se [tabel 5.1](#)) for udviklingen og derefter lave en top ti liste over risikoer.

Risks	Sandsynlighed	Prioritet
Server nedbrud	20 %	2
Sygdom	50%	4
Ændring i game engine	15%	1
Uerfarne udviklere	30%	2

Prioritet skala:
1 – Katastrofalt
2 – Kritisk
3 – Marginal
4 – Ubetydelig

Tabel 5.1 Eksempel på risiko analyse

Efter at risikoerne er blevet identificeret skal man:

- Udregne sandsynligheden for at den opstår
- Se på hvor meget ”skade” risikoen kan medføre og hvilke områder den rammer
- Lave en plan for hvordan man skal håndtere risikoen

³ [SEP]

⁴ Paraply aktivitet er et begreb der nævnes i bogen [SEP] kapitel 2.

CDoG – Creative Development of Games

Ved at projektlederen analyserer risikoerne, mindsker det chancen for at risikoen dukker op og dermed kan man håndtere den hvis den dukker op. Dermed kan man styre projektet, så man ikke ramler ind i uheld. Hvis risikoen har en meget høj sandsynlighed for at indtræffe (70 % og opefter), skal den vurderes som værende et vilkår for projektet. Dette betyder, at man regner med, at risikoen sker under udviklingen og dermed håndterer den med det samme, hvis det er muligt.

Problemet med risikoanalyse er, at det kan være svært at beregne, hvor sandsynligt det er at en risiko opstår, og derfor kan det være svært at ramme præcist. Så projektlederen må hellere regne en lidt højere sandsynlighed end for lav, så han er forberedt.

Et eksempel på, hvordan risk management kan bruges med fordel er, at det er meget vigtigt at overholde tidsplanen, som ens udgiver har bestemt. Fordi nogle gange kan dette bestemme om udgiveren bliver ved med at støtte projektet. Hvis udviklingen af projektet trækker ud i for lang tid kunne det tænkes, at de vurderer at de hellere vil støtte et andet spil og dermed kunne firmaet miste penge. Og medmindre de finder en ny udgiver til dette spil, kan de blive nød til at stoppe udviklingen af spillet. For dette kan man lave en handlingsplan, hvori man beskriver, hvad man skal gøre, hvis man kommer bagud i tidsplanen eller hvis udgiveren vælger at opsige kontrakten.

5.5 Change Management

Den bedste måde at håndtere ændringer i software på er, at benytte et versions styrings system. Det anbefales kraftigt, at udviklerne anvender et versions styrings system som f.eks. CVS⁵ eller MS VSS⁶. Meningen med disse systemer er, at når der ændres i koden, skal systemet sørge for, at ”flette” det ind i de korrekte source filer. Har flere udviklere arbejdet på samme source fil, vil systemet flette deres ændring ind i den korrekte

⁵ Concurrent Versions System <http://www.gnu.org/software/cvs/>

⁶ Microsoft Visual Source Safe

CDoG – Creative Development of Games

rækkefølge. I tilfældet af konflikter, som ændringer i samme linie, informere systemet brugerne. Det er også muligt, at skabe nye udgaver af tidligere versioner.

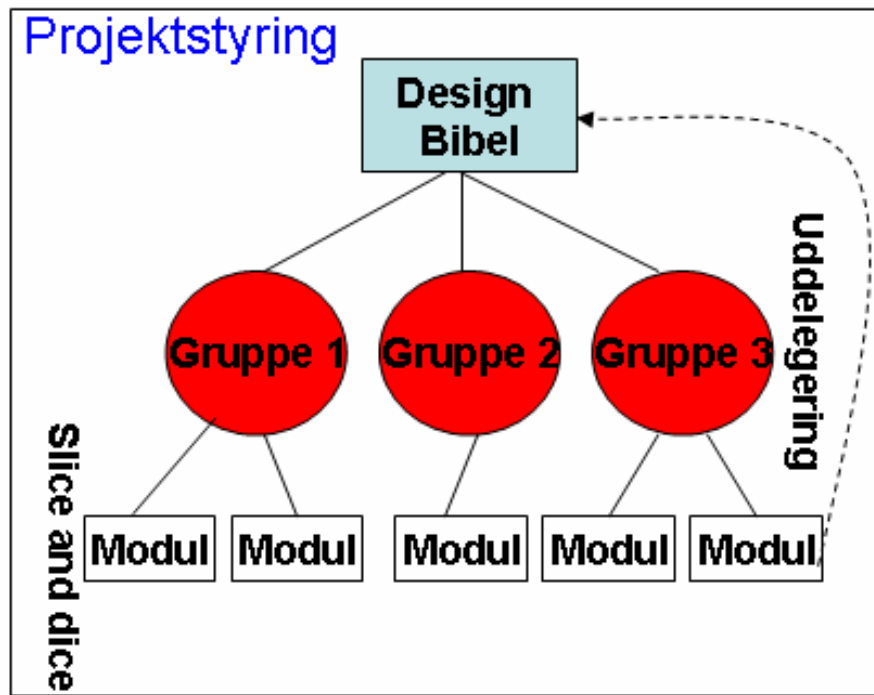
På denne måde, kan man altid gå tilbage til en tidligere version af applikationen, idet systemet gemmer, komprimerer og håndterer alle de forskellige versioner som er ”opstået” hidtil. Det gør det også muligt, at flere udviklere kan arbejde på samme kode samtidigt, uden at ødelægge hinandens arbejde, som ville være tilfældet, hvis man blot arbejdede i samme bibliotek.

Nogle systemer kan også vælge, at kun tillade én udvikler, at arbejder på den enkelte source fil ad gangen.

Der anvendes en dedikeret server til at håndterer change management, hvor udviklerne så er klienter på. Denne server kan enten være lokal eller global.

CDoG – Creative Development of Games

5.6 Projektstyring



Figur 5.4 Projektstyring framework

Forudsætningen for at kunne bruge CDoG modellen er at man har en Design Bibel udarbejdet før selve udviklingen går i gang. Design biblen skal være udviklet så den passer overens til de forventninger der er til spillet. Det vil gøre, at udviklerne har en god generel viden om, hvordan spillet skal hænge sammen. Hvis design biblen ikke er udførlig nok, vil det smitte af på spillet. Derfor er man nød til at have lavet forarbejdet godt før man går i gang med selve udviklingsfasen.

Dette betyder, at når selve projektet starter med udviklingen, er det meste af spillet specificeret. Dermed kan de gå i gang med at dele biblen op i de emner som passer til grupperne (Uddelegering). Som man kan se på [figur 5.4](#) skal disse opgaver deles op i moduler internt i gruppen (Slice & Dice), så de kan løses på forholdsvis kort tid. Hvert modul, som man også kan se i det næste afsnit, kan betragtes som en lille vandfaldsmodel. Opdeling i moduler gør, at gruppen vil have nemt ved at overskue opgaven og sørge for, at gruppen hele tiden har en forholdsvis realistisk deadline, hvor der ikke bruges mere tid end højst nødvendigt. F.eks. hvis gruppen havde en deadline på

CDoG – Creative Development of Games

et større modul på f. eks. et halv år, ville dette resultere i, at gruppen følte at de havde god tid og derfor slappede af med opgaven. Men så kort før aflevering får de travlt, fordi de er bagud og dermed bliver de måske forsinket.

Et problem med at dele Design biblen op i mange små moduler kunne være, at man får for meget tidsplanlægning, da man skal lave en plan for hvert modul. For meget projektstyring vil virke hæmmende, da man kommer til at bruge alt for meget tid på det, hvilket vil være en uheldig situation da det dermed vil hæmme den frie udviklingsform. F.eks. hvis man delte design biblen op i moduler, hvor udviklingen af denne tager kun en uge. Så vil man ende med at have for mange review møder om ugen, hvilket ikke ville være hensigtsmæssigt. Derimod hvis man delte modulerne op i forholdsvis længere tidsperioder, ville det betyde at grupperne hele tiden kan se hvilke mål der skal nås for at opnå den deadline.

Arbejdsfordeling

Udviklerne deles op efter hvilke kvalifikationer de har, hvilket betyder f.eks. at de udviklere som kender til at lave en grafikengine arbejder på de dele, som har med den at gøre. Dog er der mulighed for at flytte ressourcerne. F.eks. når grafikenginen er færdigudviklet kan disse udviklere gå over og arbejde på selve spillet. Dette vil betyde at man udnytter ressourcerne mere, så medarbejderne ikke bare sidder og slapper af fordi deres del af spillet er udviklet. Ved hjælp af en critical path, kan projektlederen flytte medarbejderne efter behov, hvilket betyder at opgaven kan blive fuldført hurtigere og dermed øger man sandsynligheden for at overholde tidsplanen. Men dette afhænger selvfølgelig af udviklernes egenskaber.

Når man laver grupper skal man passe på at man ikke får for mange grupper, da dette vil gøre kommunikationen mere vanskelig. Men på den anden side skal man også sørge for, at ens grupper er forholdsvis små, eftersom for store grupper vil betyde at det bliver sværere for udviklerne at blive enige.

CDoG – Creative Development of Games

Tidsplan

Det er de enkelte grupper som selv laver tidsplanen for hvert enkelt modul. Dermed kan firmaet være sikker på at de kan overholde planen, eftersom de jo selv ved, hvor lang tid det vil tage dem at udvikle det modul. Dette vil betyde, at udviklerne har et stort overblik over om de følger tidsplanen eller om den skal ændres. Desuden vil de enkelte grupper have et større ansvars følelse overfor om de følger tidsplanen, hvilket øger deres motivation for at følge den.

Efter alle grupperne har taget stilling til, hvor langt tid det vil tage at udvikle deres moduler, er det op til projektlederen at samle alt information i en fælles tidsplan der også inkludere integration af disse moduler. Dette betyder, at projektlederen skal lave tidsplanen for hele udviklingsprocessen ud fra estimeringer fra grupperne.

Vi anbefaler, at man laver tidsplanen tilgængelig for alle grupperne, eventuel via en intern hjemmeside, som gør at alle udviklerne hele tiden nemt kan overskue, hvad de andre grupper arbejder på. Eller f.eks. kan projektlederen vælge at hænge, hele tidsplanen inkl. arbejdsfordeling på en tavle. På den kan udviklerne følge med, hvad de forskellige grupper/udviklere laver, hvor langt de er, hvor langt tid der er tilbage til en modul. Samtidig kan udviklerne også selv notere på denne tavle, når de er færdige med deres del, så tidsplanen hele tiden er opdateret. Dette gælder også hvis der sker ændringer i tidsplanen, så skal man sørge for at opdatere det på tavlen.

Fordelen ved dette er, at udviklerne kan bl.a. se om der er en opgave som direkte afhænger af deres arbejde, som kan gøre at udviklerne vil oppe sig, for at være sikker på at deres opgave bliver færdig til tiden, så de andre ikke skal vente på dem.

Det er muligt at ændre i Design biblen, hvis man får nye ideer i løbet af processen. Dette indebærer, at tidsplanen nødvendigvis må ændres ellers vil det være umuligt at overholde den sidste deadline. F.eks. får en udvikler en ide under udviklingen, som de ikke har overvejet før. Derefter bliver det besluttet at den nye ide skal med, hvilket betyder at

CDoG – Creative Development of Games

tidsestimeringen for alle de efterfølgende delprojekter skal ændres så der er tid til den nye ide. Dette problem kan løses ved at man har adgang til, at ændre i Design biblen, og dermed kan lægge ændringen ind i tidsplanen og udskyde spillet med det samme. En anden mulighed ville være, at man lavede plads i tidsplanen, som gjorde at eventuelle mindre ændringer kunne blive sat ind alligevel, uden at man nødvendigvis ændrer i tidsplanen fordi man allerede har ”planlagt” ændringen på forhånd.

Problemet med så meget styring kan være, at udviklerne bliver holdt i en for kort snor, så de ikke kan være frie og kreative til at løse den opgave de skal løse så spillet bliver godt. Dermed risikerer man at spillet bliver samlebandsarbejde, hvilket gør at den ikke kan leve op nutidens krav til spillene.

Lederskab

Projektlederen i spiludvikling skal have en grundlæggende viden om branchen samt, hvordan den kreative proces virker. Uden denne viden vil personen koncentrerer sig om, at lave stramme tidsplaner og presse udviklerne til at arbejde så hurtigt som muligt, hvilket ikke vil være en optimal løsning, fordi det vil gøre udviklerne trætte og frustrerede. Desuden vil lederen have helt andre mål end udviklerne og de vil dermed have svært ved at kommunikere med hinanden. Det bedste valg vil være en projektleder, som har en god kendskab til udviklingsmiljøet i branchen og samtidig også kender til projektstyring. Personen skal kunne finde en balance mellem styring og kreativitet så processen ikke løbet af sporet samtidig med at udviklerne realiserer deres ideer.

Projektlederen skal have et godt forhold med udviklerne. Personen skal være en af gutterne. Personen skal kunne kende deres behov og egenskaber, så personen kan prøve at opfylde udviklernes arbejdskrav og dermed på den måde motivere dem. For hvis en udvikler ved, at projektlederen gør sit bedste til at opfylde udviklerens forventninger til jobbet, vil udvikleren være mere produktiv. Der er mange udviklere i branchen, som ikke har stort kendskab til Software Engineering. Dette medfører at de ikke har tiltro til det. Det er derfor lederens opgave, at fortælle udviklerne om disse fremgangsmåder og oplyse

CDoG – Creative Development of Games

dem om at det er værktøjer, som kan hjælpe dem med udviklingen (og deres kreativitet) og ikke er deres fjende.

Efter at projektlederen har lavet en tidsplan, som inkl. arbejdsfordelingen, er det op til udviklerne at begynde med udviklingen af modulerne.

5.6.1 Principper for projektstyring



Figur 5.5 Principper for projektstyring framework

Det er vigtigt at have nogle retningslinier for, hvordan tingene skal gøres i virksomheden. Derved har medarbejderne let ved, at håndtere grundlæggende ting og bliver derved også lettere integreret i miljøet, når de ved hvordan de kan forholde sig. Derfor indeholder CDoG modellen principper for hvert framework og disse principper bliver i CDoG opdelt i to områder: Principper i projektstyring og principper for moduler og integration frameworks. Principperne for projektstyring bliver beskrevet nu.

CDoG – Creative Development of Games

1. Kigge på folks evner

Det er vigtigt at have folk, som har de evner man har brug for til at udvikle et spil, f.eks. have nogle der er dygtige til at kode en game engine. Desuden skal der kigges på folks evner til at arbejde sammen. F.eks. skal en leder være yderst opmærksom på folk, som bare ikke kan arbejde sammen. Til dette problem skal der hurtigt findes en løsning (ændrer arbejdsfordelingen eller i det yderste tilfælde afskedige en af dem), inden det spreder sig.

2. Være realistisk

Man skal være realistiske i sine mål. Det er vigtigt at der laves en nogenlunde realistisk tidsplan, så man ikke kommer i tidsnød til sidst. Det samme gælder, hvis man laver en for ”stor” tidsplan, så folk bare driver den af, da de har for meget tid at arbejde med.

Desuden kan lederen sammenligne, hvis de har lavet tidligere projekter af samme slags, tidligere tidsplaner for at se om den her er forholdsvis realistisk.

3. Tidsplan skal hele tiden være synlig

Når tidsplanen er synlig for alle og enhver kan alle se, hvor langt projektet er og hvor meget der mangler endnu. Desuden kan de forskellige grupper se om de overholder deres tidsplan eller om de er kommet en smule bagud. En tidsplan kan f.eks. være en simpel tidslinie eller et Gantt kort.

Derved hvis man skal kommunikere med en fra en anden gruppe, kan man se hvem man skal komme i kontakt med.

4. Åbenhed ved møder:

Det er vigtigt at folk er åbne for nye ideer når de er til møder. Derudover skal man acceptere, at andre lige kan have en god ide, eller at ens egen ikke lige dur. Lederen skal her være specielt god til at lytte til ideerne og argumentere hvorfor en ide ikke dur, på en måde, som ikke demotivere folk med at komme med nye forslag. Man skal desuden huske, at have en referent til møderne, så alle kan læse

CDoG – Creative Development of Games

hvad mødet drejede sig om og hvad konklusionen af det. Derudover er det vigtigt at man ved møderne kun diskuterer problemstillinger og ikke løsninger til disse.

5.6.2 Gode råd

Efter hvert af de 3 frameworks vil vi give gode råd, som virksomheden kan overveje at følge. Grunden til at disse ikke er med i modellen er, at disse foreslag er meget generelle og dermed er det ikke sikkert at de vil virke ved alle virksomheder.

- Når en milestone er opnået, vil det være en god ide at lederen fortæller alle udviklerne om det, da de får et større indblik i hvor langt de er nået og det kan dermed også hjælpe på moralen.
- Man kan arrangere gruppe aktiviteter uden for projektet, såsom bowling eller fællesspisning. Derved kan man slappe en smule af, og desuden få mere sammenhold.
- Det kan være en ide fra begyndelsen at se på mulige opgaver som man kan outsource. F.eks. kan opgaver, som ikke er kernekompetencer af spilfirmaet og er trivelt blive outsourcet til andre steder, måske endda udlandet, for at spare omkostninger og bruge ens egen ressourcer til kerne af spillet. En anden ting som kan outsources er opgaver, såsom lydeffekter, musik og mulige videosekvenser som optræder i spillet. Da udviklerne normalt ikke har kendskab til disse ting, vil det være en god ide at outsource dem til folk som har erfaring med dette.
- Et godt arbejdsmiljø vil medvirke til, at medarbejderne kan føle sig tilpas. Dette har den effekt at ens effektivitet bliver øget.
- Der kan oprettes en hjemmeside på nettet, som forklarer spillet. Fortæller om formålet, features og viser screen shots for at brede kendskabet af spillet. Desuden kan man oprette et forum på siden. Der kan udestående komme med kommentare, hvad de synes om spillet og hvad de syntes kunne være ”sejt” at have med i det. På den måde kan udviklerne få inspiration fra andre, som kan hjælpe med at øge spillets underholdningsværdi.

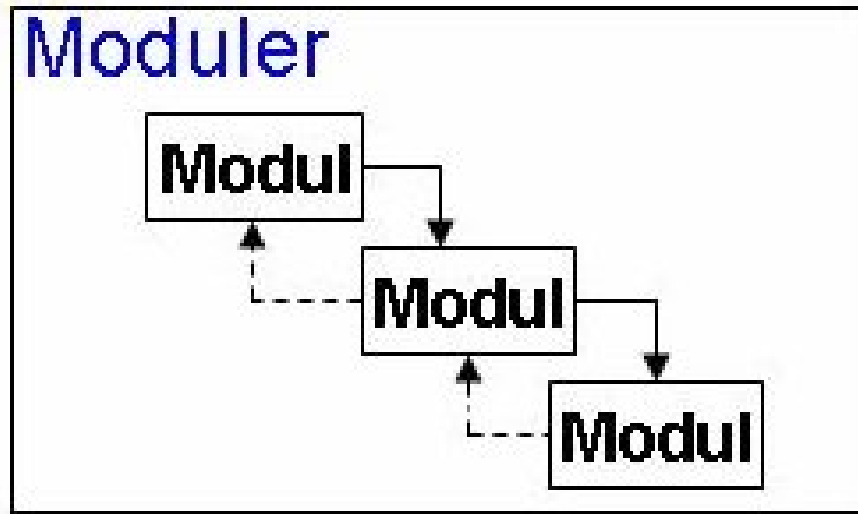
CDoG – Creative Development of Games

- Når udviklerne har gjort et godt stykke arbejde, kan det være en god ide fra lederens side at rose dem offentligt. Dette kan øge moralen blandt udviklerne, da de føler at de har gjort et godt stykke arbejde som lederen er tilfreds med.

CDoG – Creative Development of Games

5.7 Moduler

Når en gruppe modtager deres del af Design biblen skal de dele denne ”klump” i mindre moduler (Slice and Dice), hvor objekterne hænger naturligt sammen. På den måde arbejder man med mindre dele ad gangen.



Figur 5.6 Modul framework

Efter grupperne hver især med lederen har planlagt forløbet de næste måneder / år og diskuteret hvor mange moduler der skal være til stede og hvad de skal indeholde, går udviklingen rigtig i gang. Dette betyder at når udviklingen går i gang har alle en klar ide om hvad de skal arbejde på, i hvilken rækkefølge spillet skal udvikles i.

Ideen med de adskillige moduler er, at give medarbejderne og lederen en større klarhed og overblik over hele spillet. Medarbejderne kan lettere få et overblik over, hvad de skal lave og hvor lang tid det vil tage, hvis man i fællesskab har snakket om det og inddelt det de skal lave i enkelte moduler. Derved har de desuden en ide om, hvor lang tid et modul vil tage. F. eks. hvis man giver en gruppe hele forløbet og forventer at modtage en realistisk tidsplan, bliver det meget svært. Grunden til dette er, at der mangler overblik.

Når planlægningen er færdig, kan lederen bruge tidsplanen for alle moduler til at lave et Gantt chart eller tidslinie over hele processen, for dermed at danne sig et overblik over

CDoG – Creative Development of Games

forløbet. Samtidig kan lederen muligvis opfange flaskehalse tidligt, ved brug af critical path. Derved kan han, tidligt i forløbet mindske eller endda fjerne denne flaskehals, inden den opstår.

Der kan dog ske, at grupperne i deres iver laver alt for mange små moduler. Dette kan have den modsatte effekt, at man derved taber overblikket i stedet for. Dette må undgås ved, at lederen griber ind når det går for vidt med moduler.

Derudover skal man være 100 % klar over, hvad hvert modul skal indeholde, så man kan lave en så nøjagtig tidsplan som muligt. F.eks. hvis udviklerne ikke laver klare rammer for, hvad hvert modul skal indeholde, kan det have konsekvenser som, at tidsplanlægningen ikke rigtig hænger sammen, eftersom der er mere usikkerhed om, hvor lang tid et modul vil tage. Desuden kan et ”overfladisk” modul medføre en del misforståelser blandt gruppemedlemmerne, så de f.eks. laver det samme eller helt misser nogle objekter, da man troede at de andre skulle lave dem. Dette vil så medføre en klar forsinkelse i modulet. Dette kan forhindres ved, at lederen har det store overblik og skal kunne se, hvis der er unøjagtigheder og overlappning i gruppernes udvikling af moduler. Som nævnt i projektledelse afsnittet, kan en opslagstavle med alle medarbejdernes opgaver gavne, hvis udviklerne er i tvivl kan de altid checke tavlen for at se, hvad de skal lave.

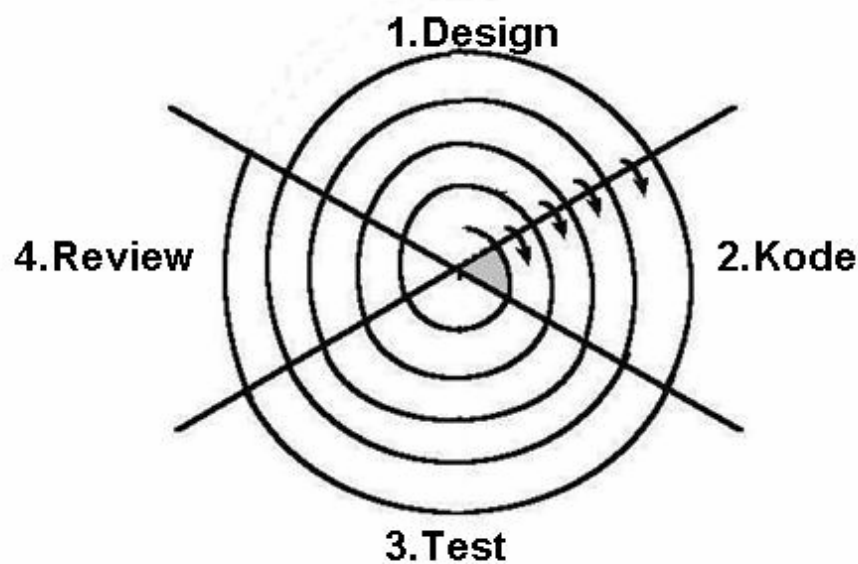
Som man kan se på [figur 5.6](#), er der pile både fra og til modulerne. Dette resulterer i, at man kan gå tilbage til et færdigt modul, hvis der skal ændres i modulet.

F.eks. hvis man i modul 1 laver en brugergrænseflade og i modul 2 laver game play til modul 1. Der kan ske det, at udviklerne kommer på nye ideer og nye features, som de godt vil have med i modul 2, men som ikke kan lade sig gøre for modul 1. Så har de muligheden for at gå tilbage til det forrige modul og foretage de nødvendige ændringer, så de nye features kan integreres og modulerne passer sammen. Ulempen ved denne fremgangsmåde er, at der kan opstå forsinkelser i tidsplanen, hvis ændringerne er alt for omfattende.

CDoG – Creative Development of Games

Modul delen er inspireret af vandfalds modellen. Dog med visse modifikationer, eftersom det normalt i vandfaldsmodellen ikke er muligt at gå tilbage for at rette i det forrige modul. Dette medfører at et modul skal være udviklet færdigt, før man kan gå i gang med det næste. Men her skal man dog være forsigtig og sørge for at man overholder tidsplanen

for hvert modul, så der ikke opstår en sneboldeffekt, så hele tidsplanen bliver forskudt. Derfor er man nød til at ændre i tidsplanen, så snart man går tilbage til et modul. Derudover kan der senere under integrationen af modulerne opstå flaskehalse, da det specifikke modul derved kan forsinke de efterfølgende modulers udvikling. F.eks. hvis der lige skulle laves nogle ændringer i engine delen, som var mere omfattende end man troede, vil det medføre en forsinkelse i gruppens forløb. Derved når de andre grupper er klar ved integrationen, bliver de nød til at vente indtil engine gruppen er klar med deres ændringer.



Figur 5.7 Moduls opbygning som er baseret på en spiral model

Hvert moduls forløb foregår som en spiral model, som man kan se i [figur 5.7](#). Dvs. at gruppen kører igennem spiralens faser flere gange, indtil modulet er færdig. Denne spiral består af følgende faser:

CDoG – Creative Development of Games

- **Design fasen**, hvor udviklerne planlægger forløbet over de næste 14 dage. Dvs. at opgaverne bliver fordelt imellem gruppemedlemmerne samt indholdet af disse opgaver bliver diskuteret og fastlagt, så alle ved hvad de skal gøre i de næste 14 dage.
- **Kodning fasen**, hvor selve koden bliver til. Her skal hver udvikler løse deres opgave. Der er selvfølgelig regelmæssig uformel kommunikation mellem gruppemedlemmerne og de forskellige grupper. Hvis en udvikler får problemer, er det muligt at diskutere med medlemmer af en anden gruppe, såvel som hans egen gruppe med det samme, i stedet for at vente til den næste review møde. Under kodnings fasen skal gruppen tage stilling til om par programmerings fremgangsmåden skal bruges. Fordele ved at bruge denne form for programmering er at fejl i koden bliver reduceret samtidig med, at læsbarheden for koden bliver øget. Samtidig er der flere end en udvikler der har styr på den samme kode, hvilket gør at hvis den ene bliver syg eller forlader virksomheden, kan den anden altid tage over uden at skulle opdatere sin viden. Men selvom man skulle tro at det vil være dyrt i personale at bruge denne metode, er det ikke nødvendigvis sådan. Eftersom når man sidder 2 personer og arbejder ved samme skærm vil effektiviteten være højere end, hvis man kun sad en person, eftersom hvis den ene er i tvivl har den anden måske en ide, eller hvis de begge er i tvivl kan de diskutere indtil de finder den optimale løsning. Desuden kan den der ikke sidder ved tastaturet opfange tastefejl så det sparer tid. Hvor hvis man kun sad en person ville det være mere besværligt at finde den optimale løsning. Dog er det et krav til medarbejderne, at de skal kunne arbejde sammen.

Efter hver udvikler er færdig med at programmere deres del af modulet, skal koden integreres til den endelige modul. Dette kan ske løbende eller man kan vente helt til sidst med at samle det hele. Dette afhænger af udviklerne og deres behov.

CDoG – Creative Development of Games

- **Test fasen** i modulfasen er meget simpel og består af unit tests og automatiske test. Unit tests bliver udført på hvert modul for at tjekke, om de enkelte moduler virker, dvs. om de opfylder kravene som er stillet til dem. Disse krav som skal overholdes kommer alle fra Design bibelen, hvor alle spillets krav og funktioner er defineret. De automatiske test laves kun på de moduler, hvor brugerne har indflydelse på kodens handling. F.eks. kan en automatisk test være, at teste om en pistol bliver affyret, hver gang man klikker på venstre museknap. Testen simulerer så denne handling og checker om den samme handling bliver udført hele tiden. De mere avancerede test kommer først når integrationen er startet.
- **Review fasen** hvor gruppen, efter selve modulen er færdiggjort, tager et skridt tilbage, hvor de ser på om, det hele hænger sammen og om koden opfylder alle krav som er stillet. Hvis den gør det, kan de gå videre til det næste modul, da den her så er færdig. Men hvis der stadig er nogle problemer tilstede, skal man tage en runde mere i spiralen for at fjerne de sidste bugs.

Ved at bruge spiral modellen her sørger man for at det er tilladt for udviklerne at forbedre modulen igen og igen indtil de mener at den er færdig. Men som nævt i beskrivelsen af spiral modellen så skal man have et klar overblik over hvornår modulet kan betragtes som færdigt, da man ellers kan køre spiralen uendeligt.

5.7.1 Gode Råd

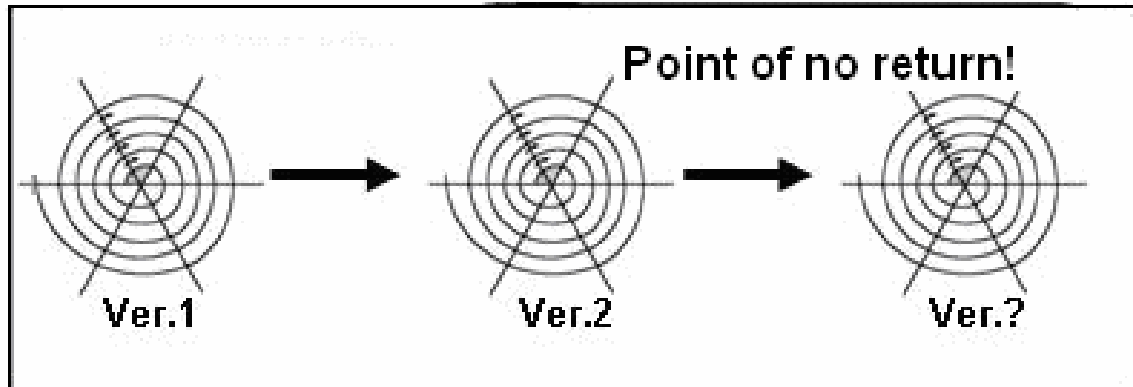
- Der oprettes en database med det formål at udviklerne, hver gang de opdager en ny bug, beskriver denne og placere denne oplysning i databasen. Oplysningerne om denne bug kan omfatte: ID, beskrivelse af den, hvem der har fundet den, hvornår den blev fundet m.m. Desuden skal der være en status for buggen, så man kan se, om den er i gang med at blive ordnet eller er færdig. Denne database kan så bruges til, at hjælpe udviklerne med at løse lignede bugs, da de har en beskrivelse af de tidligere. Denne database kan blive brugt i både modul samt integrationsprocessen.

CDoG – Creative Development of Games

- Mens udviklerne er igang med modul / integration, kan der laves demos over den version, som bliver udviklet nu. Det kan f.eks. være demonstration med screen shots over spillet. Eller man kan lave en kort demo over de færdige moduler så udgiverne kan se hvordan brugegrænsefladen og gameplayet ser ud. Dette er inspireret af prototyping modellen der har den fordel at udgiverne kan se hvor langt i forløbet spillet er og komme med konstruktiv input, hvis nødvendigt.

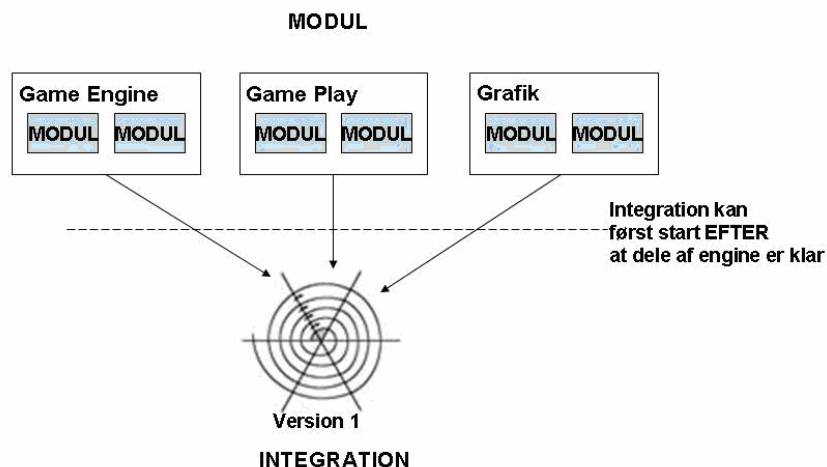
CDoG – Creative Development of Games

5.8 Integration



Figur 5.8 Integration framework

Integrations frameworket begynder så snart der er moduler, der er klar til at blive integreret. Det er projektlederens opgave at finde ud af, hvornår dette skal ske og finde udviklere til at udføre integrationen. Man skal dog bemærke at det med hensyn til spil, ikke er muligt, at integrere moduler før dele af game engine er udviklet og klar til integration (se [figur 5.9](#))



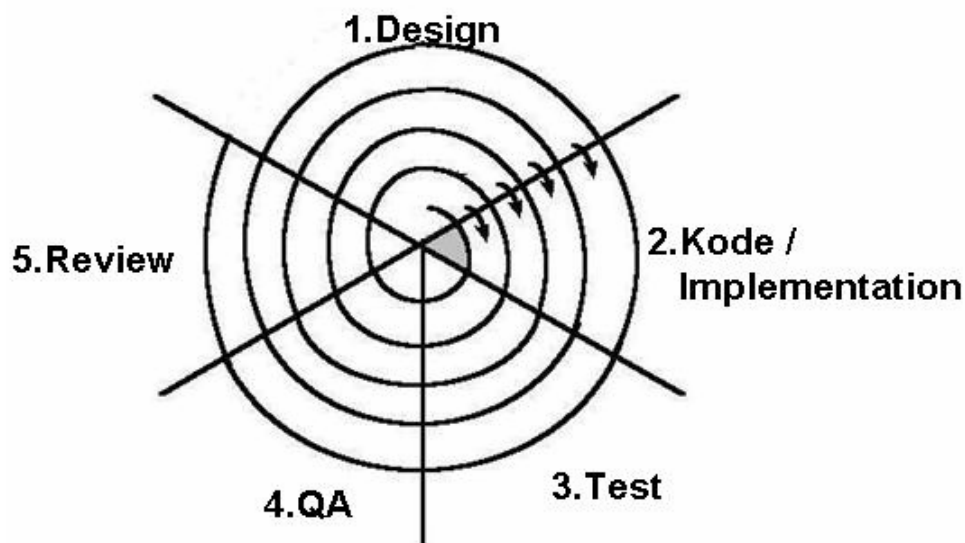
Figur 5.9 Modulerne bliver integreret EFTER game engine er klar

Når integration af moduler starter, [figur 5.8](#), opstiller projektlederen et par milestones til hvornår de forskellige integrations dele skal være færdige. Milestones er Milepæle.

CDoG – Creative Development of Games

Lederen definere disse for at kunne fortælle udvikleren hvad der skal laves, for at nå denne milestone og derved fortsætte processen. F.eks. kan lederen erklære, at selve færdiggørelsen af Game Engine er en milestone. Efter modulfasen bliver selve spiralen ændret en smule, så den passer ind i integrationsfasen, se [figur 5.10](#).

Spiralen begynder, når udviklerne har identificeret et par moduler, som skal sættes sammen.



Figur 5.10 Integrationfasens spiral

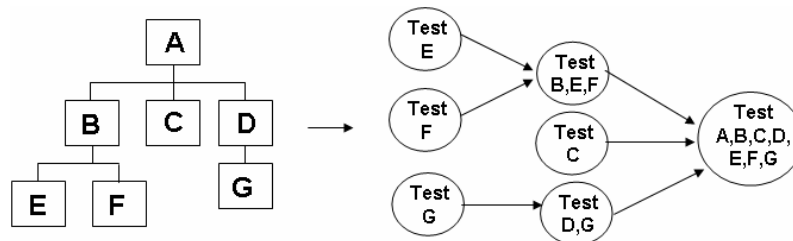
Denne integration kører så med følgende faser:

- **Design**
Ligesom i modulfasen, planlægger udviklerne forløbet over de næste 14 dage. Dvs. at opgaverne bliver fordelt imellem gruppemedlemmerne samt indholdet af disse opgaver bliver diskuteret og fastlagt, så alle ved hvad de skal gøre i de næste 14 dage. På denne måde skal man finde ud af, hvordan modulerne skal kommunikere med hinanden, om der dermed er brug for ekstra klasser og lignende. Da der sker integration af forskellige gruppers moduler, kræver denne del at grupperne kommunikerer mere med hinanden og i fællesskab integrerer disse moduler til det kommende produkt.

CDoG – Creative Development of Games

- **Kode / implementation**
Her bliver moduler implementeret sammen og desuden bliver ekstra kode skrevet efter behov.
- **Test**⁷
Når moduler skal til at integreres til selve spillet, så takket være unit testning i modul fasen, er der sørget for at modulerne virker. Når integrationen starter er der 3 forskellige fremgangsmåder integration og testfasen kan foregå:

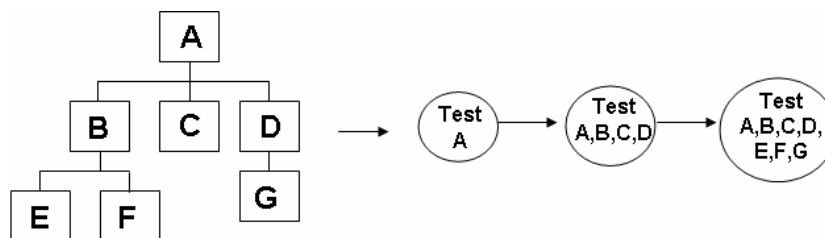
- **Bottom-up Integration**



Figur 5.11 Bottom-Up Integration

Her bliver modulerne på det laveste niveau først testet. Derefter bliver de integreret med modulerne et niveau op osv. Driver komponenter skal laves som kalder modulerne og med testdata kalder modulet for at se om den reagerer rigtigt, dette betyder at man hurtigere kan isolere fejl da man ved at de nederste komponenter fungerede.

- **Top-Down Integration**



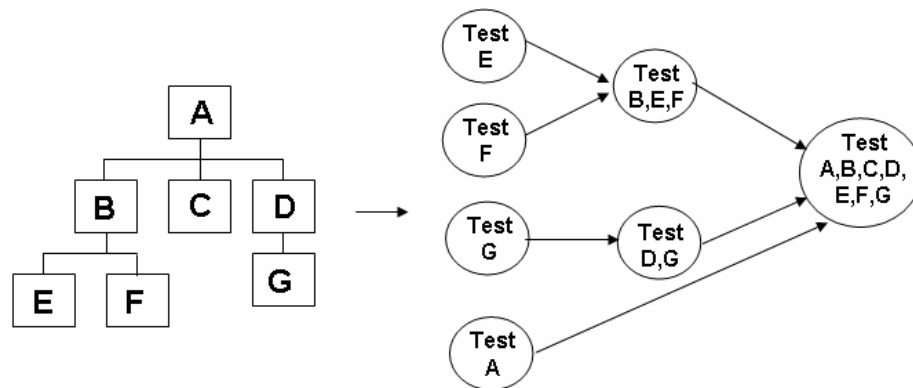
Figur 5.12 Top-Down Integration

⁷ [SEP] og [SET]

CDoG – Creative Development of Games

Her er det brugergrænsefladen i spillet, som testes først, hvorefter modulerne nedenunder bliver integreret og testet i sammenhæng med den. I top-down integration gør man det omvendte af bottom-up, hvor man først til sidst tester brugergrænsefladen, gør man det først her. Her gør man det ved, at man checker at brugergrænsefladen kalder det rigtige og dermed er sikker på, at den virker rigtigt. I top-down integration bruger man depth first integration eller breadth first integration. Dette forudsætter, at man har inddelt modulerne i træ struktur. Selve integrationen gøres ved, at man starter med toppen og integrere til den nederste og backtracker til der findes en der ikke er integreret, hvorefter den bliver integreret. Ved breadth first integration "søges" der i det vandrette plan.

- **Sandwich Integration**



Figur 5.13 Sandwich Integration

Dette er en blanding af Bottom-Up og Top-Down Integration. Udviklerne bruger to forskellige fremgangsmåder. En del af gruppen arbejder med modulerne på det nederste niveau, hvor de bruger Bottom-Up Integration. Imens arbejder den anden del af gruppen med det øverste niveau, hvor de bruger Top-Down Integration. Disse to grupper arbejder sig ind mod

CDoG – Creative Development of Games

midten. Når de møder hinanden, bliver koden fra de øverste og de nederste lag integreret med hinanden.

Vores mening er, at det er bedst at bruge sandwich integrationen, fordi det blander begge fremgangsmåder. Hvis man bruger Top-Down eller Bottom-Up skal der udvikles mange ekstra test klasser. F.eks. hvis man arbejder med Top-Down, skal der laves ekstra test klasser, som simulerer en af de nederste klasser, da disse ikke er testet endnu. Ved Bottom-Up, skal der laves driver for at simulere det næste niveau i hierarkiet. Dette gøres igen, for at sikre sig at man kan teste modulerne, hvis det næste lags moduler ikke er klar.

Grunden til, at vi synes at Sandwich modellen skal bruges er, at man på samme tid integrerer og tester begge sider af integrations moduler. Dvs. at man integrerer og tester modulerne i det nederste niveau, eftersom man har dem klar når man har sørget for at teste dem i modul frameworket med unit testning. Samtidig er man i gang med at teste og integrere den øverste del, hvis hoved modul er brugergrænsefladen, dvs. selve grafikken for spillet. Disse to møder så hinanden i midten og bliver dermed sammensat og til sidst testet for at se om det hele fungerer som det skal.

- **Automatiske test**

Efter at integration testing er færdigt og man har integreret modulerne skal man køre automatiske test på disse. I integration frameworket bruger man automatiske tests til at teste om den klynge, som består af flere moduler, virker som planlagt. Dette betyder at man ikke kører de samme automatiske tests som man havde til hvert modul i modul frameworket men udvikler nye automatiske tests som passer til den klynge. Dog kan der nogle gange være behov at køre enkelte af de automatiske teste fra modul frameworket bare for at være sikker på at modulet stadigvæk virker. Dette

CDoG – Creative Development of Games

er dog op til virksomheden da det kan være en økonomisk omkostning ved at køre disse tests igen.

- **Kvalitetssikring (QA)**

Med kvalitetssikring⁸ tester man om funktionerne i spillet opfylder kravene i Design biblen. Dette betyder, at man kigger på om produktet lever op til kravene. Denne fase kan udføres af selve virksomheden eller af spillets udgiver. Normalt plejer det at være udgiveren der står for QA fordi de dermed er med i selve udviklingen og kan komme med forslag til spillet.

QA udføres ved at kvalitetssikringsholdet har en liste over alle features som den integrerede modul (version) skal indeholde. Derefter tester de så om selve versionen opfylder disse ved enten at spille spillet eller at kigge koden igennem. QA kan også lave deres egne automatiske tests for at teste om spillet crasher.

- **Review**

Her mødes QA folkene med udviklerne for at diskutere de resultater, de har fundet frem til. Hvis der er ting der skal laves om, køres spiral en gang mere, ellers hvis versionen lever op til milestone beskrivelsen, er den færdiglavet. Udviklerne diskuterer derefter, hvordan processen for versionen har foregået og om der er ting der skal laves om inden man går i gang med den næste version.

Efter spiralen er kørt igennem flere gange og projektlederen er tilfreds og mener at produktet opfylder den opstillede milestone, har man den første version af spillet. Derefter kører den beskrevne proces igen bare for en ny version, version 2. Version 2 er en overbygning af version 1. Den indeholder f.eks. flere features og baner m.m. Fordelen ved dette er, at man efter den første milestone har man en færdiglavet version af spillet, som man så kan vise til udgiverne. Hvis virksomheden f.eks. ender med at komme i tidsnød kan man overveje at udgive den seneste version og dermed spare penge ved

⁸ [SEP]

CDoG – Creative Development of Games

ikke at bruge ekstra tid på de manglede features. Men som nævnt i afsnittet om Incremental Development modellen kan det ske at udgiveren ønsker at udgive spillet efter udgiveren har set en af de første versioner af spillet selvom udviklerne ikke mener at spillet er klar til udgivelse. Dette kan føre til konflikter mellem virksomheden og udgiverne og også mellem projektlederen og hans udviklere, hvilken kan gøre samarbejdet mellem dem besværligt. Derfor skal man, sammen med udgiverne, blive enige om hvordan modellen virker så alle kan forstå at de tidlige versioner ikke nødvendigvis er egnet til udgivelse selvom det ser sådan ud.

En anden ulempe ved denne fremgangsmåde er, at det kan være svært at estimere hvor stor og hvor lang en milestone skal være. Derfor skal projektlederen, sammen med udviklerne, sørge for at milestones er veldefineret, så udviklerne har et klart billede af hvornår man har opnået målet. For hvis ikke dette sker, kan man komme i den situation, at en spiral kører uendeligt uden man ved, hvornår man skal stoppe med udviklingen af den ene version.

Derudover skal projektlederen udpege en ”point of no return”, hvor udviklerne ikke må komme med nye ideer til spillet. Dette er inspireret af SCRUM modellen der har såkaldte sprints hvilket er design dokumenter hvor der ikke må tilføjes nye krav eller features i. Dette punkt kan være i en af de sidste versioner af spillet, som ligger tæt på udgivelsesdatoen. Når man når til ”point of no return” bliver Design biblen og de relevante dokumenter fastlåst. Dermed risikerer man ikke, at man kommer i tidsnød til sidst da man skal bruge ekstra tid til at få de nye features til at virke.

Alpha testning

Når projektlederen og udviklerne mener at de har en spilbar version af spillet, skal man begynde med alpha testning⁹. Testen her kan udføres af en test firma eller venner og bekendte. Men at lade venner og bekendte kan være en ulempe, da testningen kan blive alt for uformelt og man dermed ikke får resultater som man kan bruge til at rette fejl i

⁹ [GDP]

CDoG – Creative Development of Games

spillet. Det feedback man får fra testen kan resultere i, at man laver ændringer i spillet. Derfor er man nødt til at planlægge alpha testningen et godt stykke tid før udgivelsen, så man har tid nok til at lave ændringer.

Beta testning

Efter alpha testning påbegyndes den næste fase af testningen, som hedder beta testning¹⁰. Her har man et større team af frivillige testere der spiller spillet og dermed hjælper med at finde mulige design fejl og andre bugs i spillet. Testere kommer desuden med feedback til selve indholdet af spillet og dets gameplay. Hvis man arbejder med et multiplayer spil, kan det kræves, at lave en open beta testning også, hvor man ligger en spilbar demo ud på nettet, som kan downloades og spilles på deres server. Dermed laver man en såkaldt stress test, som tjekker om kommunikationen mellem server og spillerne foregår uden problemer. Denne form for test kan indebære over tusinder af brugere samtidig.

Som man kan se er der mange tests, som skal udføres i processen. Derfor er det bedst, at man lige fra starten laver en test plan for hele forløbet som skal følges. Dermed har man hele tiden overblik over, hvad der skal testes og hvornår og dermed mindsker man risikoen for, at test af spillet komme til at forsinke udgivelsen.

Når beta testen er udført og bestået af spillet, bliver spillet selvfølgelig udgivet. CDoG modellen kan også bruges efter spillet er udgivet. Dette kan gøres ved, at køre spiralen hver gang man har patches eller nye features, som man ønsker at udgive til spillet. Det betyder, at spillet køres igennem hele spiralen, indtil de ønskede ændringer er implementeret og en nyt version er klar.

¹⁰ [GDP]

CDoG – Creative Development of Games

5.8.1 Gode råd

- Udviklerne skal selv spille spillet min. en eller to gange om måneden. Dette kan hjælpe udviklerne med inspiration, når de spiller og samtidig finde bugs, som de kan ordne senere.
- Desuden vil det være godt, hvis lederen også en gang imellem spiller spillet, for at få indblik i, hvordan processen forløber og komme med ideer.
- Efter et spil er udgivet og det næste er i gang, kan man se om det nye spil ikke kan gøre brug af den gamle game engine. Ved at beholde eller modificere den en smule kan udviklerne spare en masse tid og kræfter med det nye spil.

Når man næsten er færdigt med at udvikle et spil, kan man dele gruppen op. Den ene fortsætter med at udvikle spillet, mens den anden (mindre gruppe) kan sætte sig sammen og brainstorme ideer til det næste spil. Derved når spillet er udviklet, har man forhåbentligt et godt koncept til det nye spil og muligvis også begyndelsen af en Design bible. Hvis man kommer bagud ved udviklingen af spillet kan man benytte ressourcerne fra den anden gruppe og kompencere, hvis det er nødvendigt.

5.9 Principper for modul og integration frameworks



Figur 5.14 Principper for Modul og Integration frameworks

1. Kunden skal altid være til stede

Kunden, som for det meste er ens udgiver, hvis man ikke selv udgiver spillet, skal helst altid være til stede. Det vil sige, at lederen af projektet skal mødes med dem (måske et par gange om måneden), for at vise dem fremskridtet og for at få input fra dem, hvis de har andre ideer eller ændringer. Det er så lederens opgave, at finde ud af om disse så er realistiske eller om de skal skrottes. Derudover skal udgiverne holdes (up to date) med, om tidsplanen overholdes og hvis ikke, hvad der så skal ske.

2. Korte iterationer, hyppige releases:

Korte iterationer og hyppige releases er en forudsætning for, at sætte kunden i stand til, at vurdere og ændre prioriteter undervejs. Ved løbende at frigive et velfungerende (del)produkt, kan udgiverne træffe beslutninger om eventuelle ændringer og justeringer baseret på fakta, frem for forventninger fra en kravspecifikation. At have et produkt, som fungerer med features, giver en mulighed for, at teste funktionaliteten med brugerne eller udgiverne på et tidligt

CDoG – Creative Development of Games

tidspunkt. Dermed øges muligheden for, at foretage ændringer som ofte viser sig, at være en nødvendighed tidligt i forløbet, i stedet for dette bliver gjort til sidst lige før det skal afleveres. Desuden giver det her en unik mulighed for at gøre udgiverne tilfredse.

Ved at holde iterationerne korte (2-3 uger), kan udviklerne og udgiverne give hinanden gensidig inspiration løbende og derved gøre udgiverne tilfredse. I CDoG modellen er det i integration frameworket at disse iterationer begyndes at lave.

3. Planning game

Hver uge holdes der et møde. Den ene uge er det internt møder mellem grupperne mens den næste uge er en review møde hvor alle grupperne mødes for at snakke om udviklingen i projektet. Dette gør at udviklerne har en egentlig plan for hver uge og ved hvad de skal i gang med.

Som nævnt så er gruppemøderne til at planlægge gruppens tidsplan for de næste 14 dage. Desuden snakker man om mulige problemstillinger som er opstået under udviklingen.

Reviewmøderne hvor alle grupperne mødes er der for at alle udviklerne kan få en overblik over status for udviklingen samtidig med at man kan komme med nye ideer til spillet og få diskuteret andre problemstillinger.

4. Metafor

Det er vigtigt, at når man beskriver spillet til andre mennesker, om det er ens udgivere eller brugere, så skal det gøres på en let forståelig måde, så alle forstår hvad det drejer sig om. Desuden skal der ikke være for meget jargon i beskrivelserne, så selv personer uden erfaring i spilbranchen kan forstå hvad der bliver sagt. Metaforen beskriver det væsentlige i projektet på et visionært plan. Dette kan f.eks. være brugergrænsefladen, ydelsen eller funktionaliteten.

CDoG – Creative Development of Games

5. Enkelt design

Det er vigtigt, at holde det hele så enkelt så muligt, så man har et overblik over tingene og det, som stadig skal nås. Desuden skal designet holdes enkelt, beskrivende og give mulighed for automatiske test. Når man taler om gode design skal man se på:

- Sikre at alle automatiske og manuelle test kan afvikles
- Design ikke indeholder duplikeret logik
- Den udtrykker udviklernes intention med modulet
- Har så få klasser og metoder som muligt
- Har det bedste udgangspunkt for ændringer, hvis det bliver nødvendigt

6. Kontinuerlig integration

I traditionelle projekter sker integrationen af moduler i slutningen af processen. Med XP metoden sker det flere gange om dagen. Alle udviklere integrerer deres kode med alle andres med et par timers mellemrum, hvilket sikrer at releases kan ske dagligt. Ud over at integrationsproblemerne minimeres, sikres det at tilbageløbning kan ske hurtigt og kontrolleret i tilfælde af problemer. Ud fra udviklerens synspunkt er det vedvarende fordel, at arbejde med den sidste nye kode fra versionskontrollsystemet. Det betyder at det ikke er nødvendigt at flette kode, som er opdateret parallelt. Desuden betyder den hyppige check ind og ud at alle automatiske test kører tilfredsstillende. Desuden kan de forskellige udviklere se, hvor langt projektet er.

7. Kodestandard

Alle skal holde ens form for standard i koden, så alle kan læse og forstå det. Man skal have nogle minimønstre, som medarbejderne skal følge. Dette gør koden mere overskueligt for andre og en selv. Desuden er koden fælles eje og skal til enhver tid kunne refaktoreres af enhver på udviklingsholdet. En anden fordel er, at dokumentationskravet minimeres, idet en udviklers indblik i kodestandarden vil sætte vedkommende i stand til umiddelbart at læse, forstå, ændre og bruge koden.

CDoG – Creative Development of Games

8. Refaktorering

Planen med refaktorering er, at kodebasen hele tiden revideres for at optimere den så godt så muligt.

Når man har lavet noget kode arbejder man med den og ser om den kan optimeres. Det bevirker, at man har mere erfaring med koden og at det her ikke sker på grund af noget krav specifikationer. Når man har optimeret koden og den er blevet ”ny” i forhold til den grundlæggende kode, bliver denne så revideret og hvis man beslutter sig for at beholde den, bliver den gjort robust og den bliver integreret i kodestandarten.

Desuden skal den køres igennem de testforsøg, som den oprindelige kode har været igennem, da kravene for funktionaliteten ikke har ændret sig.

9. ”37” times arbejdsuge

Når man arbejder med XP metoden, er en af konsekvenserne, at den enkelte udvikler er på mærkerne hele tiden. Dette medfører, at der ikke kommer kvalitets resultater ud af det arbejde, når man kommer noget ud over 37 timer i ugen. Derfor skal projektlederen sammen med medarbejderen sørge for at der ikke er alt for meget overarbejde, da dette medfører stress som kan hæmme kreativiteten. Derfor er det bedst at holde arbejdsugen på 37 timer. En anden styrke ved denne metode er at man nemmere kan beholde sine medarbejdere, eftersom hvis der er konstant overarbejde er der høj risiko for at medarbejderne søger andet arbejde fordi de simpelthen ikke kan holde til at have en arbejdsuge på 75 timer og mere.

CDoG – Creative Development of Games

5.10 CDoGs afslutnings fase

Efter at spillet er udgivet skal virksomheden lave en ”postmortem” over processen. De skal diskutere, de positive og negative ting som foregik under processen. Ud fra dette skal de identificere problemområder, som der skal gøres noget ved og samtidig de gode ting, som skal gentages igen til den næste spil.

Implementationen af CDoG modellen kan godt tage tid i virksomheden. Projektlederen skal sørge for, at tilpasse modellen til hans udviklere og til virksomheden. CDoG giver projektlederen de nødvendige værktøjer og råd til, hvordan spil udviklingen kan ledes effektivt og uden de store problemer. Efter Design biblen er skrevet færdigt og veldefineret, laver udviklerne sammen med projektlederen, det tilsvarende projektplan til udviklingen. Derefter deles udviklerne i grupperne, hvor hver gruppe får deres del af biblen som de skal udvikle. Efter de udviklede modulerne er testet og kan integreres, bryder en gruppe ud af modul udviklingen og begynder med integration af disse. Integrationen frameworket sørger for, at der (efter den første version er færdigt) er et spilbar version af spillet.

CDoG er en proces model der tager hensyn til, at der skal masser af kreativitet til, når man udvikler spil. Derfor skal udviklerne have lidt frie tøjler, så de kan udfolde sig, samtidig med at projektlederen bevarer overblikket under hele udviklings forløbet. Dermed får man mulighed for, at lave et spil der kan klare sig på markedet samtidig med at den bliver udgivet til tiden!

CDoG – Creative Development of Games

6. Implementering af CDoG

Der er flere forskellige måder, at gribe denne problemstilling an på og de afhænger af, hvordan firmaet fungerer. Desuden skal man kigge på hvilken type medarbejdere, der arbejder i virksomheden, da nogle mennesker vil have nemmere ved at acceptere nye ændringer end andre. Nogle medarbejdere kan finde på at gøre modstand imod at bruge en model, da de ikke er vant til at arbejde på denne måde, hvilket betyder at de skal omstille sig. Desuden vil der være folk, som vil mene at den nye model vil hæmme kreativiteten. Dette betyder, at medarbejdere vil føle at deres arbejdsvilkår vil ændre sig og eftersom mange medarbejdere arbejder i spilbranchen fordi de syntes det er sjovt vil de måske have den frygt at det bliver helt ændret efter der bliver tilføjet strammere styring.

Når en ny model skal implementeres i en virksomhed vil det være en god ide, at tage medarbejderne med på råd. Dermed får man en dialog i gang om hvordan det skal være og se om medarbejderne har forslag til, hvordan implementationens processen kan gøres bedst. Dette vil gøre, at medarbejderne føler at de er medbestemmende og dermed vil det være lettere at få dem indstillet til at bruge den nye model.

6.1 Trinvis implementering

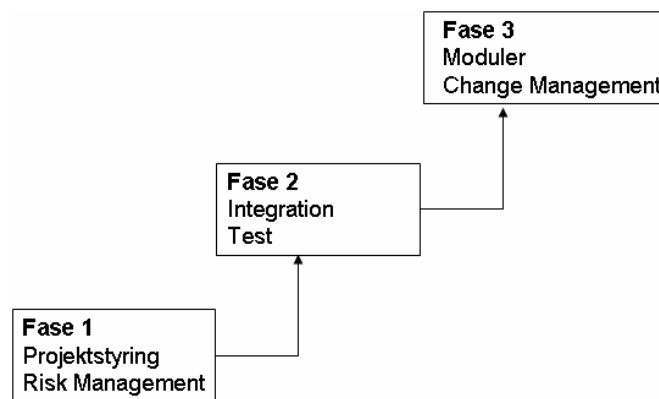
Selve fremgangsmåden virker ved at man deler modellen op i mindre dele, så man implementerer en del af gangen. Dette betyder, at medarbejderne ikke får et stort kulturшок og dermed får nemmere ved at overskue, hvordan tingene sker og hurtigere kan vænne sig til, at arbejde på den nye måde.

Grunden til, at man vil implementere i små trin er hvis ens medarbejdere ikke har erfaringer med udviklingsmodeller og dermed ikke er klar over, hvilken styrke der ligger i at arbejde med det her værktøj. Ledelsen skal derfor vise, at modellen faktisk vil gøre det lettere for medarbejderne og gøre det nemmere at overholde tidsplanerne. Derfor er man nødt til at informere om, hvordan modeller virker, samt hvorfor de kan bruges.

CDoG – Creative Development of Games

Denne fremgangsmåde vil passe bedst til virksomheder som f.eks. MediaMobsters. MediaMobsters har en løs arbejdsstruktur i deres organisation. Da virksomheden på nuværende tidspunkt ikke kører med en egentlig model vil det være en fordel at starte langsomt. Dvs. implementering af CDoG modellen trinvis.

Implementeringen af CDoG i MediaMobsters kan deles i tre faser:



Figur 6.1.1 Trinvis implementering af CDoG

- **Fase 1**

Virksomheden skal starte med projektstyring frameworket. Man skal vænne sig til at lave tidsplaner. Derfor skal man dele udviklerne i grupper og desuden skal design biblen også deles og uddelegeres til de forskellige grupper. Derefter skal grupperne enkeltvis finde ud af hvor langt tid det vil tage dem at udvikle deres del og meddele dette til projektlederen så han kan lave en tidsplan. Udviklingen i fase 1 foregår på samme måde som de har gjort før. Udviklerne skal desuden holde review og gruppe møder som beskrevet i CDoG modellen. I denne fase skal man også begynde med risk management delen af CDoG modellen.

- **Fase 2**

Efter man har fået styr på projektstyring kan man implementere Integration frameworket da man allerede har dele der skal integreres. Her skal man så følge den iterative fremgangsmåde som CDoG foreslår ved at lave versioner af spillet

CDoG – Creative Development of Games

lige fra starten af integrationen. Derudover skal paraply aktiviteten Test for integration frameworket implementeres hvor man kører de beskrevne tests.

- **Fase 3**

Man skal vente med at implementere fase 3 til det næste projekt begynder, da man efter implementering af fase 2 er færdig med udviklingen. Når man begynder med det nye spil skal man først køre fase 1 som allerede er implementeret. Derefter kører man fase 3 med moduler og så til sidst fase 2 med integration. For det er i denne rækkefølge at CDoG modellen er opbygget. Dermed fuldfører man implementeringen af CDoG. Dvs. man starter med at lave tidsplanen ud fra modul fordeling til grupperne. Derefter udvikles modulerne efter CDoG beskrivelse og integreres. Under hele forløbet kører man med alle tre paraply aktiviteter.

Ulempen ved trinvis implementering af udviklingsmodellen er, at der kan opstå konflikter mellem modulerne fra den gamle model og den nye da det jo langt fra er sikkert, at modellerne kan fungere sammen. F.eks. hvis virksomheden går fra en linear model til en iterativ model, kan det være svært at få dem til at fungere sammen. Desuden vil det tage længere tid at implementere hele modellen, eftersom man gør det langsomt, hvilket betyder at den endelige omkostning bliver langt højere. Desuden er det sandsynligt at det nuværende udviklingsprojekt bliver meget forsinket, hvis der kommer konflikter imellem den gamle og nye model.

6.2 Big-bang

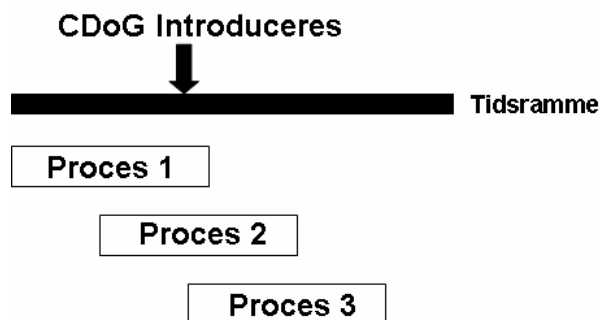
Denne integrationsform er ret simpel. Man finder ud af, hvordan medarbejderne skal deles op, så de passer ind i grupper til den nye model. Derefter holder lederen et møde med medarbejderne, hvor man informerer om den nye model, hvordan den fungerer og hvordan de skal arbejde fremover. Dette er langt den hurtigste måde, at implementere modellen på. Problemet er bare, at medarbejderne kan føle, at modellen bliver presset ned over dem. Derfor kan denne form bruges effektivt, hvis man har medarbejdere som er gode til at omlægge sig og som har erfaringer eller kendskab med proces modeller.

CDoG – Creative Development of Games

F.eks. et firma som ITE, som allerede bruger XP i deres udvikling. Dermed er medarbejderne klar over, hvordan det er at arbejde under en speciel form for projektstyring. Desuden arbejder ITE allerede i forudbestemte grupper, så udviklerne kender i forvejen hinanden så godt, at det vil være lettere for dem og deres gruppe medlemmer at omlægge sig til den nye arbejdsform som modellen CDoG foreslår.

Fordelen for virksomheden er, at det vil være billigere at omskifte modellen på en gang, da det sker forholdsvis hurtig. Men der kan opstå kulturchok for medarbejderne, da det sker fra det ene projekt til det andet, at der kommer en helt ny arbejdsform. Man må dog i det første projekt indstille sig på, at der kommer en højere fejlprocent i spillet, fordi folk ikke er vant til at arbejde i den nye model og ikke kender den.

F.eks. har ITE hele tiden flere spil i udvikling samtidig. Når CDoG bliver introduceret, skal man så beslutte i hvilken proces denne skal implementeres. Her kan man så bruge en hybrid metode, som består af big-bang og trinvis implementation. Som man kan se på eksemplet i [figur 6.2.1](#) er der 3 processer, som startede på forskellige tidspunkter.



Figur 6.2.1 Parallel spil udvikling

Når CDoG bliver introduceret mener vi at det vil være bedst at proces 1 forsætter med den nuværende model. Imens skal proces 2 bruge trinvis implentering, da den er så langt inde i udviklingen, at det er for sent at kunne implemetere hele modellen.

CDoG – Creative Development of Games

Derimod skal proces 3, som lige er startet bruge big-bang. Dermed når man skal i gang med en 4. proces, er der 2 udviklingsgrupper som har fået erfaring med CDoG.

Hvis en virksomhed har flere end et spil i gang, skal projektlederen vurdere hvor det er bedst at benytte big-bang.

6.3 Eventuelle problemstillinger ved integration af CDoG

Informationsmøder/briefing

Når CDoG modellen skal implementeres, er det lederens opgave at informere og forberede medarbejderne på arbejdsformen i den nye model. Dette kan gøres ved, at holde informations kurser der beskriver modellen og dens fremgangsmåde.

Der skal lægges vægt på, at alle medarbejdere har fået svar på alle deres spørgsmål og kender deres plads og opgaver i den nye organisation. Disse kurser kan bruges til, at give medarbejderne en introduktion samt en generel beskrivelse af, hvordan systemudviklingsmetoder virker. Derefter skal man beskrive CDoG modellen så medarbejderne kan se, hvordan modellen fungerer.

Det antages, at omkostningerne til disse informations møder vil tjene sig ind igen i form af, at det undgås, at arbejdet står stille i tilfælde af tvivl, konflikt og forvirring om modellen. Hvis medarbejderne ikke har været ordentligt oplyst og ikke helt er klar over deres funktion, kan disse tre ting let ske. Desuden vil der altid være omkostninger i forbindelse med at omlægge en virksomhedsstruktur, men det antages at disse vil blive tjent ind i det lange løb ved at den nye model er mere effektiv.

De nævnte kurser kan indføres og holdes sideløbende med den normale arbejdsgang, hvorved der så sikres en glidende tilvænning til den nye arbejdsform, som kan påbegyndes ved starten af næste projekt. Dette vil betyde, at medarbejderen vil have en viden om, hvad der skal ske og hvorfor ændringerne sker. Dermed vil de være mere åbne

CDoG – Creative Development of Games

overfor ændringerne, hvilket betyder mindre modstand og øget samarbejde med medarbejderne.

CDoG – Creative Development of Games

7. Metrikker

Metrikker¹¹ bruges i den normale industri til at måle kvaliteten og kvantiteten af et produkt. Hvor man har et fysisk produkt, kan man tælle, hvor meget der kommer ud i den anden ende. Problemet med software er, at det er svært at bedømme kvaliteten af et produkt, da kvalitets fornemmelsen i et stykke software nærmere findes i, om det opfylder kravet fra kundens synspunkt og om programmet er effektivt i forhold til andre programmer som udfører den samme funktion.

Når det kommer til software, er der mange måder at måle størrelser på. De har hver især deres fordele og ulemper. Ofte benyttes f.eks. LoC – Lines of Code som måling for størrelsen på software. Fordelen ved dette er, at det er let at tælle antal kodelinier, men det betyder også, at en simpel effektiv algoritme vil blive ”straffet”, hvis den fylder mindre. Det kan også være, at et andet programmerings sprog gør det samme på færre linier. Man kan heller ikke direkte ud fra antal kode linier aflæse en given algoritmes effektivitet.

En anden måde er, at bruge Funktion Point (FP) metrikker, hvor man udregner størrelsen af softwaren ud fra antallet af funktioner, som er defineret i analyse fasen. F.eks. kan man bruge design biblen til at udregne hvor mange funktioner spillet kommer til at indeholde. Funktioner i et spil kunne f.eks. være spillerens interaktioner med spillet eller spillets interaktion med en server. Ud fra disse points kan man så udregne, hvor mange lines of code det vil tage, at implementere funktionen.

For at FP og LOC kan bruges er det nødvendigt for virksomheden at have historiske data fra forrige projekter. F.eks. kan virksomheden, ud fra andre projekter, udregne hvor meget en funktion point er værdi i lines of code, omkostninger, fejlrate og mandetimer.

Disse resultater kan man bruge til, at ”udregne” kvaliteten af en proces. Dette betyder, at en spilvirksomhed, efter at de har implementeret CDoG modellen, kan verificere om modellen er mere effektiv end den forrige. Dette gøres ved at man udregner den samlede

¹¹ [SEP]

CDoG – Creative Development of Games

funktion point værdi for hver projekt og derefter sammenligner resultaterne. På den måde kan man se om en model er bedre end den anden da man bl.a. kan sammenligne omkostninger for projektet, fejlratere og mandetimer. Desuden kan projektlederen bruge metrikkerne til at vurdere omkostningen og arbejdstimerne for spillet.

Hvilken metode man vælger at køre med er op til virksomheden. Men vi mener at det for spiludvikling vil være en fordel at bruge funktion points. Fordelen ved denne er at den er uafhængig af programmeringssprog. Grunden til dette er, at FP kigger på den overordnede funktion. Den kigger ikke på LoC, hvis værdi kan være forskelligt afhængig af hvilket sprog man bruger. FP kigger på antal af funktioner og hvornår disse er færdig implementeret.

Som et eksempel kan vi sige, at virksomheden ITE har implementeret CDoG i stedet for den XP proces model som de kører med. De har i forvejen FP værdier fra de forrige projekter, hvor man kan se hvor stor værdi en funktion point har osv. Ud fra design biblen for det nye spil, kan de finde ud af hvor mange funktioner spillet har. For hvert funktion udregner man dens værdi. På [figur 7.1](#)¹² kan man se et eksempel på hvordan en funktion point for en funktion udregnes.



Figur 7.1 Eksempel på Function Point vægtninger

¹² [SEP] kap.16

CDoG – Creative Development of Games

Efter værdien for total er fundet, skal denne ganges i en formel og ud fra resultatet får man den egentlige funktion point som man så bruger til sammenligne med.

CDoG – Creative Development of Games

8. Konklusion

Da arbejdsmetoderne i spiludviklingsbranchen er anderledes end de fleste systemudviklings metoder er der stor behov for proces modeller som passer til deres arbejdsmetoder. Indtil nu har de fleste udviklede proces modeller fokuseret på de mere faste arbejdsmetoder, hvor man har et fast tidsplan og den ønsker man at opnå lige meget hvad. Udviklernes kreativitet og kunnen bliver på denne måde hæmmet. Og det er processen og værktøjerne der bliver fokuseret mest på og ikke på ens menneskelige ressourcer. Dette er ikke acceptabelt i spiludviklingsmiljøet, da succes af et spil afhænger af en udviklers fantasi og kreativitet, som sørger for at spillet bliver interessant og sjovt at spille.

Men da udvikling af et spil er en lang og uoverskuelig proces, er der behov for modeller der kan hjælpe med at styre processen, samtidig med at de ikke hæmmer udviklernes kreativitet. Ud fra gruppens samtaler med to af Danmarks spiludviklings virksomheder, MediaMobsters og ITE, fandt vi ud af, hvilke forventninger disse to har til en proces model. Der var stor forskel mellem de to virksomheder, hvor den ene allerede arbejdede med proces modellen XP, mens den anden egentligt fulgte Code og Fix modellen, som kan være en stor ulempe for et projekt af den størrelse. Vi fandt ud af at mange af udviklerne der arbejder i branchen ingen egentlig kendskab har til software engineering modeller, da de ikke har en relevant uddannelse. Derfor er det som det første nødvendigt at informerer udviklerne om disse værktøjer og fortælle dem fordelene ved at bruge en proces model. Ud fra samtalerne og gruppens kendskab til de eksisterende proces modeller, udviklede vi en model der er inspireret af bl.a. XP, Scrum, Spiral og prototyping modellen. Vi valgte at kalde modellen for Creative Development of Games (CDoG).

Forudsætning for brug af denne model er at udviklerne allerede har udarbejdet en design bibel for spillet. Denne design bibel skal indeholde HELE beskrivelsen af spillet. Gruppen har valgt at CDoG modellen ikke skal diktere sammensætning og udarbejdelse af design

CDoG – Creative Development of Games

biblen, da hvert spiludviklings virksomhed har deres egen fremgangsmåde for udarbejdelse af denne.

CDoG modellen består af tre frameworks: projektsyring, moduler og integration. I projektstyrings frameworket udarbejder projektlederen en tidsplan og arbejdsfordeling for spil udviklingen. Dette bliver gjort ved at design biblen bliver delt op, Uddeligering, som vi kalder det. Disse dele bliver tildelt til forskellige grupper som så inbyrdes har til opgave at dele deres del af spillet i moduler (Slice and Dice) og derudover estimere hvor langt det vil tage at udvikle disse moduler. Når dette er gjort er det projektlederens opgave, ud fra gruppernes estimeringer, at lave en tidsplan.

Med modul frameworket begynder den egentlig udvikling af spillet. Udviklingen følger spiral modellen der bliver gentaget indtil modulet er færdig udviklet. Spiralen består af design fasen hvor udvikleren udarbejder en plan for udviklingen i de næste 14 dage samt specificerer, hvad modulet skal indeholde før det kan blive kaldt færdigudviklet. Derefter følger kodningsfasen og testfasen. Spiralen slutter med reviewfasen, hvor man analyserer forløbet og selve koden for at finde ud af om modulet er færdigt. Udviklingen af de forskellige moduler foregår som en slags vandfaldsmodel, hvor man dog altid kan gå tilbage til en færdigudviklet modul og lave ændringer efter behov.

I Integration frameworket bliver modulet integreret med hinanden. Dette kan dog først ske så snart en del af game engine er udviklet og er klar til integration, da game engine er kernen af spillet. Integrationen følger også en spiral model der udover de nævnte faser i moduler også indeholder kvalitetsikring(QA) fasen. Her tester en uafhængig gruppe om spillet overholder de krav, som blev stillet i design biblen. Integration frameworket består af milestones, hvor man kører spiralen, indtil milestonen er opfyldt. Når dette er sket har man faktisk en version 1 af spillet der er køredygtigt og kan, hvis behovet er der, udgives. Dette kan være en mulighed, hvis man f.eks. er meget bagud i tidsplanen og ikke ønsker at forlænge den for at nå at implementere alle features i spillet. Ellers forsætter

CDoG – Creative Development of Games

integrationen med at integrere yderligere features i version 1, som så bliver til version 2. Man forsætter med at udvikle versioner indtil projektlederen bestemmer, at spillet er færdigudviklet og klar til udgivelse.

Udover framework er der også tre paraply aktiviteter, hvoraf to gælder for hele modellen. Disse to er Change Management og Risiko Management. Med Change Management holder man styr på de forskellige moduler og versioner af spillet så udviklere ikke går ind og ændrer i noget som ikke er tilladt. Dette gør man ved hjælp af versionstyrings software, der kan konfigureres til at håndtere adgang og fletning af koden.

Med Risk Management prøver man at mindske risikoen for fejl ved, at på forhånd at tage stilling til mulige risikoer og håndteringen af disse. Udover disse to så er der en tredje paraply aktivitet, Testing, som gælder for Modul og Integrations frameworks. Disse testst består af bl.a. automatiske og manuelle tests der sørger for at et modul og en version af spillet lever op til kravene og er mere eller mindre fejlfrie.

Ligesom i XP har vi valgt at opstille værdier og principper for CDoG modellen. Hensigten med disse er at øge effektiviteten og samarbejdet mellem udviklerne ved at give dem retningslinier for udviklingen. Dermed kan de have en fælles udviklens fremgangsmåde og en kultur som passer til deres behov. Et af de vigtigere principper for CDoG modellen er review og gruppe møderne. Med gruppe møderne mødes gruppemedlemmerne indbyrdes hver anden uge for at diskutere hvor langt de er og mulige probemstillinger som er kommet frem i de foregående 14 dage. Med review møderne mødes alle grupperne for at informere hinanden hvor langt de er og desuden tage stilling til nye feature forslag som kan have opstået siden sidst.

CDoG modellen giver projektlederen et større overblik over udviklingen samtidig med at udviklerne beholder frie tøjler, hvad angår kreativiteten. Desuden har vi forsøgt at forøge samarbejdet mellem udviklerne ved, at give forslag til hvordan de i fælleskab kan arbejde

CDoG – Creative Development of Games

for at opnå målet. Dette kan bl.a. være ved at alle har ”rørt” ved koden og ved at alle har styr på hvad de andre laver og hvor langt de er. På denne måde får udviklerne mere at sige mht. spillet og processen hvilket i sig selv er en god motivation for at gøre sit bedste.

Men vi kan ikke med sikkerhed sige, at CDoG modellen opfylder virksomhedernes krav da vi på ingen måde kan teste denne påstand. Vi kan kun formode, hvordan tingene i spil udvikling hænger sammen og ud fra det give en model forslag som vi mener, er optimal. For at teste vores påstande er det nødvendigt at en virksomhed faktisk implementerer vores model og derefter sammenligner dens resultater med de andre foregående projekter. Dette giver vi også forslag til i rapporten. Mht. implemetationen af CDoG mener vi at der er to muligheder: big bang hvor hele modellen implementeres på relativt kort tid og trinvis implementation, hvor man implementerer dele af modellen over en længere periode. Hvilken fremgangsmåde en virksomhed skal vælge konkluderer vi ikke da vi mener at det afhænger af organisationens opbygning og deres normer. Så det er op til projektlederen at finde ud af hvilken passer bedst til hans udviklere. Efter at virksomheden har udviklet deres første spil, ved at følge CDoG, kan de sammenligne slut resultaterne med de andre projekter. Her kommer metrikker ind i billedet. Virksomheden kan bruge funktion points fremgangsmåden. Ved hjælp af denne udregner man FP værdierne for alle funtkioner i spillet. Ud fra disse værdier kan man sammenligne hvor langt tid det tog at udvikle spillet, hvor mange udviklere det tog og omkostningerne for hele forløbet. På denne måde kan virksomheden sammenligne disse resultater for at verificere om CDoG modellen passer bedre til dem end de andre udviklingsmodeller.

Alt i alt mener vi at CDoG er en udviklingsmodel der passer godt til spiludviklingsprocessen. Men for at den også kan løse virksomhedernes behov er man først nødt til at køre med den over et par projekter og dermed forme den til virksomhedens behov og normer.

CDoG – Creative Development of Games

9. Litteraturliste

[SEP] Software Engineering: A Practitioner's Approach

Forfatter: Roger S Pressman, Roger Pressman

Udgiver: McGRAW-HILL

ISBN: 0071238409

[SET] Software Engineering: Theory and Practice (2nd Edition)

Forfatter: Shari Lawrence Pfleeger

Udgiver: Prentice-Hall, Inc

ISBN: 0130931292

[ESE] Extreme Software Engineering: A Hands-On Approach

Forfatter: Daniel H. Steinberg, Daniel W. Palmer

Udgiver: Prentice-Hall, Inc

ISBN: 0130473812

[AM] Agile Modeling: Effective Practices for Extreme Programming and the Unified Process

Forfatter: Scott W. Ambler, Ron Jeffries

Udgiver: Wiley

ISBN: 0471202827

[GDP] Game Development and Production (Wordware Game Developer's Library)

Forfatter: Erik Bethke

Udgiver: Wordware Publishing, Inc

ISBN: 1556229518