Individuell rapport, TNM094

Sandra Pettersson
Mikrokosmos

19 april 2016

Sammanfattning

I detta arbete studeras planeringsarbetet av ett projekt inom systemutveckling. Olika frågeställningar tas fram för att på ett bättre sätt kunna redovisa ett resultat. Detta visar att de agila metoder som nämns i rapporten kan användas för att utvecklingen av det specifika systemet ska förenklas och förbättras. Agil systemutveckling förbättrar standarden på arbetet i teamet men även kvalitén på slutprodukten som lämnas till kunden.

Innehåll

Kapitel 1

Inledning

Utvecklingsprocessen av ett system kan variera beroende på vilken metod som används vid framtagandet. Det är viktigt att rätt metod används för att både kunden och utvecklarna ska känna sig nöjda med den slutliga produkten.

1.1 Syfte

För att utvecklingen av ett system ska fungera krävs en välstrukturerad arbetsprocess och olika utvecklingsmetoder för projekthantering påverkar projekt på varierande sätt. De olika metoderna jämförs och analyseras, detta för att kunna bestämma vilken arbetsstruktur som lämpar sig bäst för det specifika projektet.

Kundens förväntningar på slutprodukten är en av de viktigaste parametrarna som måste tas hänsyn till. Eftersom kundens behov kan förändras under den period då projektet fortskrider är det viktigt att arbetssättet tar hänsyn till förändringar. Det kan handla om både små och stora förändringar som tar olika lång tid att åtgärda. För att slutprodukten ska motsvara kundens förväntningar är det viktigt att arbetsprocessen hela tiden tar hänsyn till kundens önskemål.

1.2 Frågeställningar

Teamets arbetsprocess undersöks för att sedan komma fram till vilken metod som anses som mest effektiv gällande tidsåtgång men som även ger bra kvalité på slutprodukten.

Olika metoder för projekthantering påverkar projekt på varierande sätt. För att kunna välja den metod som lämpar sig bäst för både teamet och projektet måste dessa studeras och jämföras.

- Vilken arbetsstruktur gör så att teamet fungerar så bra som möjligt och därmed kan utveckla en välfungerande applikation?
- Hur ska teamet arbeta för att applikationen ska bli intuitiv och motsvara kundens förväntningar?

Ett dåligt strukturerat arbetssätt kan skada projektet. Det är därför viktigt att gruppen trivs, både med varandra men också med arbetsmiljön där produkten tas fram. För att slutprodukten ska leva upp till de behov som ställs av kunden behöver teamet arbeta utifrån en struktur som passar både för teamet och projektets typ.

1.3 Avgränsningar

Denna rapport kommer inte att ta upp några etiska eller samhälleliga aspekter.

1.4 Typografiska konventioner

Kursiv text anger att en ny term tas upp i rapporten.

Kapitel 2

Bakgrund

2.1 Projektbeskrivning

Mikrokosmos är en del av en utställning på Visualiseringscenter C. Applikationen är till för en stor touch-skärm och används idag för att visualisera information inom ämnet biokemi och molekylärbiologi. Media, så som bilder, filmer och text, visas i form av flyttbara rutor som även kan roteras och skalas. Applikationen som finns är seg och ger dålig respons till användaren. Den är svår att förstå sig på när det kommer till funktionalitet och samband mellan olika data.

Projektet går ut på att skapa en ny applikation som ska ersätta den befintliga. För att underlätta för kunden att visa upp den information som är mest relevant vid ett visst tillfälle ska möjligheten att byta ut ämnet på informationen finnas i den nya applikationen. Den ska vara intuitiv och lätthanterad för både besökare och guider, då dessa är de personer som främst kommer i kontakt med applikationen. För att göra informationen mer lättförståelig för användaren ska samband mellan olika data i applikationen förändras och visas på ett tydligare sätt än i nuläget.

2.1.1 Teknisk beskrivning

För att ge teamet möjligheten att skriva så optimal kod som möjligt utvecklas applikationenmed programspråket C++. Då applikationen behöver touch interaktion för att fungera som planerat kommer tredjeparts-APIet SDL användas för att implementera detta. SDL är et utvecklingsverktyg som bland annat ger tillgång till touchskärm, tangentbord och ljud. Då projektet involverar mediafiler, dessa i formaten jpeg, png och mov, behövs även ett sätt att ladda ner dessa. SDL fungerar även för detta syfte. Informationen som ska läsas in, så som text och kategorier, finns samlad i en xml-fil. För att kunna läsa denna filen används rapidxml.

Målet är att applikationen inte ska lagga lika mycket som den befintliga versionen. För att detta ska vara möjligt ska en hög framerate användas, 25 fps är tillräckligt för att applikationen ska fungera utan problem.

För att strukturen på koden alltid ska hålla samma kvalité är det viktigt att alla i teamet följer samma standard när det kommer till namngivning av olika variabler i koden. Exempelvis ska varje variabelnamn endast bestå av gemener, om namnet inte består av två ord. Då ska den första bokstaven i det andra ordet skrivas med en versal. Varje konstant ska skrivas med enbart versaler och ska alltid bestå av bara ett ord.

2.2 Systemkrav

Förutom de krav som ställs på arbetsprocessen finns även krav från kunden. Kundens krav på slutprodukten är viktigt att ta hänsyn till då detta är ett av sätten att avgöra hur framgångsrikt projektet varit.

- Applikationen ska vara intuitiv och lätthanterad för användaren.
- Det ska vara möjligt för kunden att själv förändra innehållet i applikationen. Exempelvis ska ämnet på informationen lätt kunna bytas ut för att anpassas efter kundens önskemål.
- Det ska finnas *stories*, som ska fungera som färdiga berättelser som kan följas på skärmen. Dessa ska främst kunna användas av guider, men även av besökare för att förenkla upplevelsen.

Kapitel 3

Agil systemutveckling

Agila systemutvecklingsmetoder används vid programutveckling och kan också kallas för lättrörliga och iterativa metoder. De fyra grundpelarna inom agil systemutveckling är följande:

- Individer och interaktioner värderas högre än processer och verktyg.
- Fungerande mjukvara värderas högre än omfattande dokumentation.
- Kundsamarbeten värderas högre än kontraktsförhandlingar.
- Flexibilitet runt förändringar värderas högre än att följa en plan [3].

3.1 Extrem programmering (XP)

Extrem programmering, även kallad XP, är en agil metod som baseras på fem värderingar:

- Kommunikation
- Enkelhet
- Återkoppling
- Mod
- Respekt [1]

Då kontinuerlig kommunikation mellan utvecklarna och kunden värderas är det viktigt att ha en kund på plats. Detta gör det enklare att prioritera hantering av de viktigaste kraven först, vilket gör att systemet uppfyller de krav som ställs av kunden. De inblandade projektmedlemmarna måste vara ärliga och inse sina begräsningar gällande krav från kunden. Vet utvecklarna att ett visst krav inte kommer att kunna uppfyllas av olika anledningar är det viktigt att framföra detta till kunden redan när kravet ställs [1].

Genom att ofta strukturera om koden (refaktorering) uppnås enkelheten. Antalet dokument som varken innehåller kod eller bidrar till det slutliga systemet minskar i antal [1].

Det är viktigt att visa respekt för kunden, medarbetare och sig själv genom att exempelvis inte göra förändringar i medarbetares kod utan att bra tester har genomförts innan [1].

Utöver dessa fem värderingar har XP flera stöttande tillämpningar (se bilaga A). En stor del av de tillämningar som ingår är planeringsspelet. Detta spelas av både utvecklarna och kunden och går ut på att de tillsammans tar fram olika *stories* som beskriver kraven. Utvecklarna är ansvariga att ta fram hur kostsamma dessa stories kommer att vara och sedan är det kundens uppdrag att välja vilka stories som ska prioriteras framför andra [1].

Kod ska hållas enkel och valideras av tester som skrivs innan koden utvecklas. Koden ska hålla en viss standard genom hela projektet och denna standard ska följas av alla programmerare. Genom att även kontinuerligt ta bort överflödig kod ökar kvalitén på produkten som levereras. Parprogrammering ökar känslan av gemensamt ägarskap, vilket gör att alla i teamet har rätt att ändra i koden, oavsett vem som skrivit den. Det är viktigt att bygga systemet varje gång som en implementation har blivit klar [1].

Utvecklarnas välmående värderas högt inom Extrem programmering. Då en programmerares arbete innebär att man måste tänka hela tiden är det viktigt att de alltid är utvilade för att kunna arbeta bättre. En 40-timmars arbetsvecka är standard och övertid är inte tillåtet två veckor efter varandra [1].

3.2 Scrum

Detta är en metodik för systemutveckling där målet är att strukturera projekt där önskemål från kunden ständigt förändras. Scrum innefattar ett antal roller och ett antal beståndsdelar som består av obligatoriska möten och dokument [2].

De olika roller som är viktiga inom denna utvecklingsmetodik är *Produktägare*, *Scrum master* och *Utvecklingsteam* och kallas även för Scrumteam. Produktägaren är en del av teamet men måste inte nödvändigtvis vara en del av utvecklingsteamet. Denna ansvarar för att ta emot, hantera och prioritera önskemål från kunden gällande tillägg eller förändringar. Scrum mastern jobbar för att hålla ihop och coacha teamet genom projektet. Denna ska se till så att utvecklingsteamet stöter på så få hinder som möjligt och ser till så att alla aktörer inom projektet jobbar mot samma mål. Utvecklingsteamet är självorganiserande och ska bestå av 3-9 personer, där den sammanlagda kompetensen uppfyller de behov som projektet berör [2].

Produktbackloggen är ett dokument som fungerar som samlingsplats för alla önskemål som finns på produkten. Denna ägs och hanteras av produktägaren, som även ansvarar för att backloggen alltid är uppdaterad. Önskemålen prioriteras olika högt och rangordnas sedan därefter. Detta är något som ständigt kan förändras [2].

Projektets arbete delas in i *sprintar*. Varje sprint varar mellan 3 och 30 dagar, inleds med en planeringssession och avslutas med att de utlovade ändringarna granskas. Under planeringssessionen går produktägaren igenom alla önskemål inför hela Scrumteamet för att gruppen sedan ska kunna bryta ner och värdera vikten av alla delar. Produktägaren bestämmer sedan vilka förändringar som anses vara mest värda att lägga tid på under sprinten. Vid granskningen av sprintens resultat redovisas produktens funktionalitet för att undersöka vad som är klart och vad som inte är klart. Produktägaren presenterar sedan sina planer i backloggen för alla deltagare. Resultatet är sedan en uppdaterad backlogg där det tydligt framgår vad som ska göras nästkommande sprint. Sprinten avslutas med att hela Scrumteamet diskuterar vad som tas vidare från sprinten, vad som gått bra och vad som kan förbättras till nästa sprint [2].

En viktig del av varje sprint är en *Daglig Scrum*. Detta är ett planeringsmöte för utvecklingsteamet som maximalt får ta 15 minuter. Genom att inspektera och uppdatera de planer som finns för den aktuella sprinten maximerar teamet sina chanser att uppnå de satta målen. Vanliga frågor att ta upp på detta möte är:

- Vad har jag gjort sedan igår?

- Vad ska jag åstadkomma till imorgon?
- Vad hindrar mig?

För att undvika att mötet blir för långdraget genomförs alltid detta stående. Detta gör att endast det väsentliga tas upp och lite mer oviktiga konversationer kan tas vid ett senare tillfälle [2].

Kapitel 4

Projektplan

4.1 Tidsplan

Tidsplanen för projektet Mikrokosmos kan ses i bilaga B.

4.1.1 Sprintar

I projektet Mikrokosmos kommer utvecklingsteamet att arbeta i sprintar som varar i två veckor. Detta för att deltagarna ska hinna avsluta varje moment utan att det går onödigt lång tid mellan varje sprint. Sprintar är ett bra arbetssätt då det tydliggör vad som är relevant för teamet att jobba med vid just den tidpunkten. Likt utvecklingsmetoden Scrum (se 3.2), kommer det varje dag hållas ett möte där det gås igenom vad som har gjorts, vad som ska göras och vad som eventuellt hindrar utvecklarna.

4.1.2 Avstämning med kund

För att varje sprint ska gå framåt är det viktigt att hela tiden vara uppdatera kundens önskemål på förändringar. Detta görs lättast genom att ha en kundrepresentant inom teamet.

4.1.3 Milstolpar

För att lätt få en överblick över vad som ska göras under projektet sätts milstolpar upp av teamet. Dessa milstolpar ska ha genomförts vid olika tilllfällen under projektets gång. Milstolpar kan inte jämföras med stories utan är snarare ett mått på hur långt teamet ska ha kommit vid en viss tidpunkt. Mellan varje milstolpe kan olika många sprints och stories ha genomförts.

Dessa är de milstolpar som satts upp för Mikrokosmos:

- Ha ett pappersutkast på utseende till applikationen.
- Visa flera bilder, i form av vykort, som användaren kan interagera med.
- Visa samband mellan olika bilder och information på ett tydligt sätt.
- Skapa berättelser som användaren kan följa.
- Koden ska lätt gå att förändra om kunden vill ändra ämne på informationen som visas på applikationen.

4.2 Uppbyggnad av projektet

4.2.1 Utvecklingsplattform

Teamet kommer inte begränsas till en viss IDE eller kompilator då C++ och SDL fungerar på många plattformar. För att applikationen ska kunna köras från både Windows- och Appledatorer är det viktigt att *preprocessor directives* används.

4.2.2 Kravhantering och spårning

Programmeringen kommer ske både enskilt och i grupper om två, så kallad parprogrammering. När en teammedlem programmerar ensam är det viktigt att koden granskas av en eller flera teammedlemmar innan den läggs samman med den gemensamma koden. Detta för att undvika små fel som sedan kan leda till mer allvarliga och svårlösta problem. Då programmering sker i par kommer koden att granskas på ett annat sätt eftersom det aldrig är en person som skriver allt. Trots detta ska koden granskas ordentligt.

För att kunna hantera de krav som kunden och utvecklarna ställer på systemet krävs en uppdaterad backlogg [4]. Backloggen består av ett antal stories som inkluderar teamet och kundens krav finns. Det är viktigt att teamet hela tiden är beredda på att kundens krav kan förändras samt har en plan för hur detta kan lösas. Produktägaren ser till att backloggen hela tiden är uppdaterad efter kundens krav på förändringar.

4.2.3 Versionshanteringssystem och rutiner

Då flera personer kommer arbeta med samma kod kommer GitHub att användas som gemensam plattform. Detta för att lätt kunna dela och uppdatera sin kod. Då alla har tillgång till detta är det därför också viktigt att den koden som läggs upp är en stabil version.

4.2.4 Testningsprinciper och rutiner

För att få en överblick över vilka delar av systemet som uppfyller kundens krav är det viktigt att tester sker kontinuerligt genom hela projektet. Detta görs i samband med att varje sprint avslutas men även efter att en stor del av koden har bytts ut. Detta görs för att undvika att mycket tid går åt till att jobba med kod som inte fungerar.

Tester bör även göras när det kommer till användargränssnittet på applikationen. Det ska vara lättförståeligt för användaren. Då användaren är både guider och besökare kommer målgruppen att vara inom ett stort åldersspann och kompetensnivån kommer att variera kraftigt. Dessa tester kommer bestå av dels pappersversioner av olika delar av applikationen men även färdiga versioner som kan användas på en mindre touchskärm. Papperstester, gjorda i exempelvis Adobe Photoshop, är ett bra sätt att testa applikaionens funktionalitet då själva applikationen i sig inte behöver vara färdig för att genomföras.

4.2.5 Dokumentationsprinciper och rutiner

Det finns flera typer av dokument som kommer att användas under projektets gång. Dessa är statiska dokument och levande dokument. De statiska dokumenten är färdigskrivna och kommer inte att

förändras, exempel på ett sådant dokument är mötesprotokoll. De levande dokumenten är dokument som hela tiden uppdateras för att ge en mer korrekt bild av projektets status. Ett tydligt exempel på ett levande dokument är backloggen [3].

För att alla i teamet ska ha tillgång till dessa dokument hela tiden är det viktigt att de delas. Detta kan göras antingen på GitHub tillsammans med projektkoden eller i en mapp på Google Drive eller Dropbox.

4.2.6 Modelleringsstandard och rutiner

För att modellera hur programmet ska vara uppbyggt är UML:s klassdiagram ett bra alternativ att använda sig av. För att dessa hela tiden ska vara uppdaterade vidareutvecklas under varje sprint.

4.3 Organisation

4.3.1 Ansvarsfördelning

Precis som i scrum kommer teamet bestå av en Scrum master, en produktägare och ett utvecklingsteam (se 3.2). Utöver dessa roller ska det finnas en person som ansvarar för att regelbundna tester genomförs och dokumenteras. Det är också bra att det alltid är någon som har huvudansvaret för att varje sprint genomförs på korrekt sätt.

4.3.2 Mötesprinciper

Varje dag kommer börja med ett scrummöte för att få en inblick i hur det går för alla och om det finns några hinder som behöver uppmärksammas extra mycket. Det är viktigt att de dagliga mötena hålls korta för att inte ta för mycket tid från arbetet med projektet. Utöver dessa hålls ett planeringsmöte i början, samt ett utvärderingsmöte i slutet av varje sprint.

Kapitel 5

Analys och diskussion

5.1 Diskussion

För att arbetsprocessen ska gå så smidigt som möjligt är det viktigt att ha en struktur som ska följas. Scrum är en bra systemutvecklingsmetodik som gör det lätt för teamet att hålla sig till sina arbetsuppgifter.

För att alla i teamet ska veta om sina arbetsuppgifer är det bäst om alla i gruppen har ett bestämt ansvarsområde. Detta för att det är lättare att veta att saker blir gjorda som de ska om någon har ansvar för just den delen i projektet. Det behöver inte nödvändigtvis vara personen som har ansvaret som ska göra detta men personen i fråga har ansvar för att det blir gjort. Detta fungerar nog bäst om är lagom stora och noga strukturerade ansvarområden.

Sprintar är ett bra sätt att strukturera upp arbetet på då detta gör att alla har en arbetsuppgift i en bestämd tid. Problem kan dock uppstå om det är någon i teamet som inte hinner slutföra de uppgifter som tilldelats personen. En positiv sida med sprintar är att det är lätt att tillgodo se de krav och förändringar som kunden kan tänkas ha. Detta för att teamet stämmer av hur de ligger till vid varje utvärderingsmöte. Finns det då något som ska förändras är det smidigt att kunna göra det då så att alla vet vilket mål de arbetar mot.

För att det ska vara lätt, både för en själv och för andra, att gå tillbaka och kolla i dokumentationen av projektet. Det finns många olika sätt som detta kan göras på, men det viktigaste är att alla i teamet går efter samma mall. Dokumentationen av koden är lika viktig som dokumentationen av möten och liknande.

5.2 Resultat

Syftet var att hitta en lösning som fungerar för just detta projekt. De mest väsentliga delarna av agila utvecklingsmetoder har tagits upp och dessa är de som påverkar hur arbetet med projektet kommer att ske. Även om Scrum (se 3.2) är den metod som kommer implementeras mest i projektet finns det även inslag av Extrem Programmering (se 3.1) och tillfällen då dessa två kombineras för att skapa bättre arbetsförhållanden för utvecklarna.

Kapitel 6

Slutsatser

För att kunna dra några slutsatser är det smidigast att titta tillbaka på de frågeställningar som ställdes i början av rapporten.

> *Vilken arbetsstruktur gör så att teamet fungerar så bra som möjligt och därmed kan utveckla en välfungerande applikation?*

I sin helhet är Scrum den utvecklingsmetod som ger utvecklingsteamet bäst förutsättningar att klara av de olika sprintarna i tid. De olika rollerna i teamet gör att alla kan fokusera på den del som är mest relevant för tillfället och detta kan öka kvalitén på slutprodukten. Dock ger det bättre resultat att kombinera flera olika metoder för att anpassa arbetssättet för teamet som arbetar med projektet. Exempelvis ger Extrem Programmering en mer strukturerad start medan Scrum sedan gör att projektet går vidare på ett bra sätt.

> *Hur ska teamet arbeta för att applikationen ska bli intuitiv och motsvara kundens förväntningar?*

Genom att arbeta efter Scrums riktlinjer genom hela projektet kommer slutprodukten motsvara kundens förväntningar. Det är förmodligen inte exakt de förväntningar som ställdes vid arbetets start men då en kundrepresentant varit närvarande under hela processen har utvecklingsteamet jobbat efter de förändringar som kunden gjort.

Litteraturförteckning

[1] Kent Beck och Cynthia Andres, *Extreme Programming Explained, Second Edition*, Pearson 2007

[2] Kenneth S. Rubin, *Essential Scrum, Third Edition*, Pearson 2013

[3] Shari Lawrence Pfleeger och Joanne M. Atlee, *Software Engineering, Fourth Edition, International Edition*, Pearson 2010

[4] Ulf Eriksson, *Kravhantering för IT-system, Upplaga 2*, Studentlitteratur 2008

Bilaga A

Tillämpningar i Extrem programmering

Utöver de fem värderingar i Extrem programmering finns även tolv så kallade tillämpningar.

- Planeringsspelet
- Små leveranser
- Metafor
- Enkel design
- Testning
- Omstrukturering av kod
- Parprogrammering
- Gemensamt ägarskap
- Kontinuerlig integration
- 40-timmars arbetsvecka
- Kund på plats
- Kodstandard

Bilaga B

Tidsplan för projektet Mikrokosmos

		V	Viktiga datum	Arbeta med				
Måndag	1/2	5				Efterforskning		
Onsdag	3/2	5				ang lämpliga		
Fredag	5/2	5				språk och hur		
Måndag	8/2	6				dessa fungerar.	Projektplan	
Onsdag	10/2	6						Individuell
Fredag	12/2	6						rapport
Måndag	15/2	7						
Onsdag	17/2	7	Dedline individuell rapport (18/2)					
Fredag	19/2	7						
Måndag	22/2	8			Sprint 1			
Onsdag	24/2	8						
Fredag	26/2	8	Deadline för projektplan					
Måndag	29/2	9						
Onsdag	2/3	9						
Fredag	4/3	9						
Måndag	7/3	10			Sprint 2			
Onsdag	9/3	10	Avstämningsmöten					
Fredag	11/3	10						
Måndag	14/3	11		Studieuppehåll				
Onsdag	16/3	11		(Tenta-P, Omtena-P,				
Fredag	18/3	11		Påsk)				
Måndag	21/3	12						
Onsdag	23/3	12						
Fredag	25/3	12						
Måndag	28/3	13						
Onsdag	30/3	13						
Fredag	1/4	13						
Måndag	4/4	14	Spårseminarium (tisdag, 5/4)					
Onsdag	6/4	14		MTD				
Fredag	8/4	14						
Måndag	11/4	15			Sprint 3			
Onsdag	13/4	15						
Fredag	15/4	15						
Måndag	18/4	16						
Onsdag	20/4	16						
Fredag	22/4	16						
Måndag	25/4	17	Avstämningsmöten (25/4 och 26/4)		Sprint 4			
Onsdag	27/4	17						
Fredag	29/4	17						
Måndag	2/5	18						
Onsdag	4/5	18						
Fredag	6/5	18						
Måndag	9/5	19			Sprint 5			
Onsdag	11/5	19	Deadline för utkast rapport (torsdag, 12/5)					
Fredag	13/5	19						
Måndag	16/5	20						
Onsdag	18/5	20						
Fredag	20/5	20						
Måndag	23/5	21						
Onsdag	25/5	21	Deadline för individuell opposition					
Fredag	27/5	21						
Måndag	30/6	22						
Onsdag	1/6	22	Deadline projektrapport (torsdag, 2/6)					
Fredag	3/6	22	Slutseminarium					