Post Mortem Report

# 1. Processer och rutiner under arbetets gång

Då mjukvaruutveckling av större system i grunden är relativt komplext i och med att det lätt kan bli ett flertal komponenter och tekniker som behöver samverka med varandra är det därför även viktigt med struktur och regler för utvecklingsprocessen. Genom att använda sig av olika processer och rutiner inom utvecklingsteamet kan man på så vis ge sken för projektmedlemmarna att processen är mindre komplex och därför även förenkla arbetet med mjukvaran.

Hur bör man då gå till väga för att på bästa sätt kunna hantera projektprocessen? Detta är givetvis beroende på projektets omfattning och komplexitet, och i just vårt fall har projektet varit tvunget att vara anpassningsbart. Detta tack vare att samtliga projektmedlemmar har varit nya inom hur man hanterar teknologierna som har använts i projektet, men även hur det är att, på ett effektivt sätt, jobba i ett projektteam. Då projektprocessen har haft kravet på sig att vara anpassningsbart är har det tack vare detta varit lämpligt att tillämpa en empirisk processtyrning (agil processutveckling). I förhållande till en fördefinierad processtyrning (s.k. ”vattenfall”) lämnar det agila arbetssättet mer utrymme för flexibilitet under hela projektets levnadscykel, och kräver inte heller allt för tung förundersökning innan man ger sig på skapandeprocessen. Detta har i vårt fall varit en bra metodik då det har varit begränsat med tid för att utföra projektet, det har därför varit vart viktigt för oss att ge oss på utvecklandefasen så fort som möjligt utifrån en provisorisk projektvision som sedan har justerats under hela projektprocessen.

Som förväntat av oss har vi under projektets gång använt oss av ett agilt arbetssätt med huvudinspiration hämtad från SCRUM-metodiken. Trots att ingen av de inblandade i projektet hade någon större kunskap sedan tidigare kring hur man jobbar agilt har det ändå fungerat bra, och det går definitivt att finna flertalet fördelar med att jobba agilt kontra att jobba med en klassisk vattenfallsmodell.

## 1.1. SCRUM med extreme programming (XP)

Även fast både SCRUM och XP är agila arbetsmetoder har vi funnit inspiration till en bra arbetsmetodik genom att kombinera olika element från de både metoderna. Då SCRUM är en förhållandevis fri arbetsform bidrar det trots detta till en vettig grundstruktur för dokumentation och arbetsstruktur under arbetet. Genom att använda sig av en *product backlog* där man via *user stories* beskriver vad man som användare i slutet kommer vilja ha ut av produkten är det på så vis enkelt för alla som är inblandade i projektet att se vad som finns kvar att göra. Man kan på så vis se hur arbetet ligger till och vad som behövs göras.

Genom att dela upp dessa *epics,* som man beskriver i sin product backlog, och sedan dela in dessa i flera olika deluppgifter kan man på så vis för varje vecka enkelt bestämma sig för vilka features man vill fokusera sig på inför veckan. Det är på så vis lätt att kunna skapa en struktur över vad man ska göra under arbetsperioden, och dela upp arbetet i flertalet *sprints* som man kallar det inom SCRUM. För oss har en sprint inte stäckt sig längre än under en vecka i och med att vi har haft som mål att ha en release per vecka, vilket även är ett kursspecifikt krav.

Något vi har anammat från XP är bland annat att hela utvecklingsteamet jobbar tillsammans med alla uppgifter som behövs ta hand om istället för att bara sköta det som man själv har blivit tilldelad för veckan. Detta har då även bidragit till att vi alla har suttit samlade tillsammans för att jobba med projektet. Även *pair programming* är en arbetsteknik som vi har hämtat från XP.

Att hålla kontinuerliga möten varje dag (*daily scrum*)är även en rutin som vi har haft under projektets gång. Detta har uppfyllts genom att varje projektmedlem i början av ett arbetspass har berättat för de andra om vad de har gjort sedan senaste arbetspasset och vad de ämnar göra tills nästa arbetstillfälle. Om det har funnits något hinder som har hindrat arbetet har även detta tagits upp samt hur man bör jobba för att lösa problemet.

## 1.1.1. Fördelar kontra nackdelar med SCRUM/XP

Det vi under denna projektperioden bland annat har reagerat över är hur pass effektivt det är med pair programming. Att låta två personer jobba med samma del av projektet har för oss varit väldigt tidseffektivt då två hjärnor oftast jobbar snabbare än vad en gör. Då problem har uppkommit har dessa oftast kunnat lösas mer smärtfritt än ifall man som ensam programmerare skulle ha attackerat problemet. Då vi även under våra dagliga scrum-möten har förklarat för varandra vilka slags problem som vi har stött på, har vi gjort som så att ett par utvecklare har blivit tilldelade uppgiften att lösa problemet efter mötet för att sedan kunna fortsätta med sprintmålen.

Även fast XP-metodikens tanke om att sitta och jobba mycket tillsammans kan vara effektivt då alla projektmedlemmar enkelt kan konversera idéer och enklare fatta större beslut tillsammans, har vi sett att det inte alltid har varit en bra arbetsmetodik. Då projektmedlemmarna i vår grupp har haft olika scheman och olika dagsrutiner har det tidvis varit problematiskt att hitta en tid som passar samtliga inblandade. Detta har vissa veckor lett till in

nackdel XP, sitta tillsammans: man måste komma in till skolan lite onödigt mycket ibland kanske

nackdel scrum: sprint om den tar slut innan veckan är över

daily scrum : . Genom att berätta för övriga gruppen om hur man ligger till i sitt eget arbete.

## 

## 1.2. Git Flow

För att alla i utvecklingsteamet enkelt ska kunna följa arbetsprocessen och jobba med olika delar av projektet samtidigt bestämdes det tidigt att vi skulle följa arbetsmodellen *Git Flow* för versionshanteringen på vår Git. Den allmänna uppfattningen om arbetsmetoden är mycket positiv då det har gjort projektet lätthanterligt och lättöverskådligt.



Genom enbart använda master-branchen för större programreleaser har det på så vis varit enkelt för oss att se hur projektet har utvecklats vecka för vecka.

Allt arbete har utgått från developbranchen, där gruppmedlemmarna sedan själva har grenat ut till olika feature-brancher. Detta är i stora drag vad vi har utnyttjat av Git Flow i just det här projektet, dvs develop-branchen, masterbranchen och feature-branches. I vårt fall har det ansetts redundant att blanda in release-brancher då sprinterna har legat så nära inpå varandra. Det har därför räckt för oss att sammanställa alla färdiga feature-brancher inför varje ny veckorelease och sedan merge:a in develop-branchen på master.

Det har dock blivit något spretigt för vår del då mycket funktionalitet inte har hunnit komma ut till develop-branchen innan en vecko-deadline,

## 1.2. Fördelar kontra nackdelar med processverktygen/arbetssätten

# 2. Tidsåtgång

# 3. Designval (?)