Reflektionsrapport

Software Engineering Project - DAT255

Samuel Eksmo
Mark Henriksson
Rebecca Leckborn
Anne Tarnow
Nils Thylen

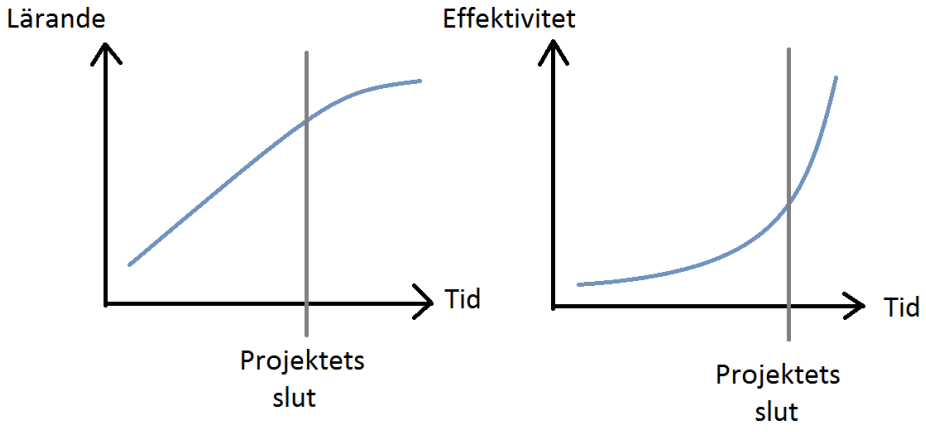
Juni 2017

Innehåll

1 Inledning

Våren 2017 fick studenter från Chalmers och Göteborgs Universitet möjlighet att utveckla tio stycken applikationer integrerade i ett nytt kommunikationssystem vid namn PortCDM. PortCDM är ett system vars syfte är att effektivisera hamnanlöp genom att förbättra och förenkla kommunikationen mellan olika aktörer i en hamn. Denna rapport beskriver processen bakom en av dessa applikationer inriktad mot hamnterminaler. Applikationen togs fram av en grupp på fem studenter. Rapporten innehåller reflektioner över gruppens arbetsprocess där Scrum-metodiken använts. Reflektionerna har baserats på R. Smiths citat (2001): "Assessment of what is in relation to what might or should be and includes feedback designed to reduce the gap".

Utvecklingsprocessen hos gruppmedlemmarna har i stora drag följt lärande- och effektivitetskurvorna som figur 1 visar. Effektiviteten i produktutvecklingen var till en början låg, samtidigt var inlärningskurvan desto brantare. Mycket var nytt för många i gruppen, både de tekniska delarna och att arbeta med Scrum. Under projektets gång har gruppmedlemmarna blivit allt mer erfarna inom dessa delar och längre fram i projektet utvecklades också ett bra kunskapsutbyte inom gruppen. Detta har lett till en kraftigt förbättrad effektivitet under projektets gång.

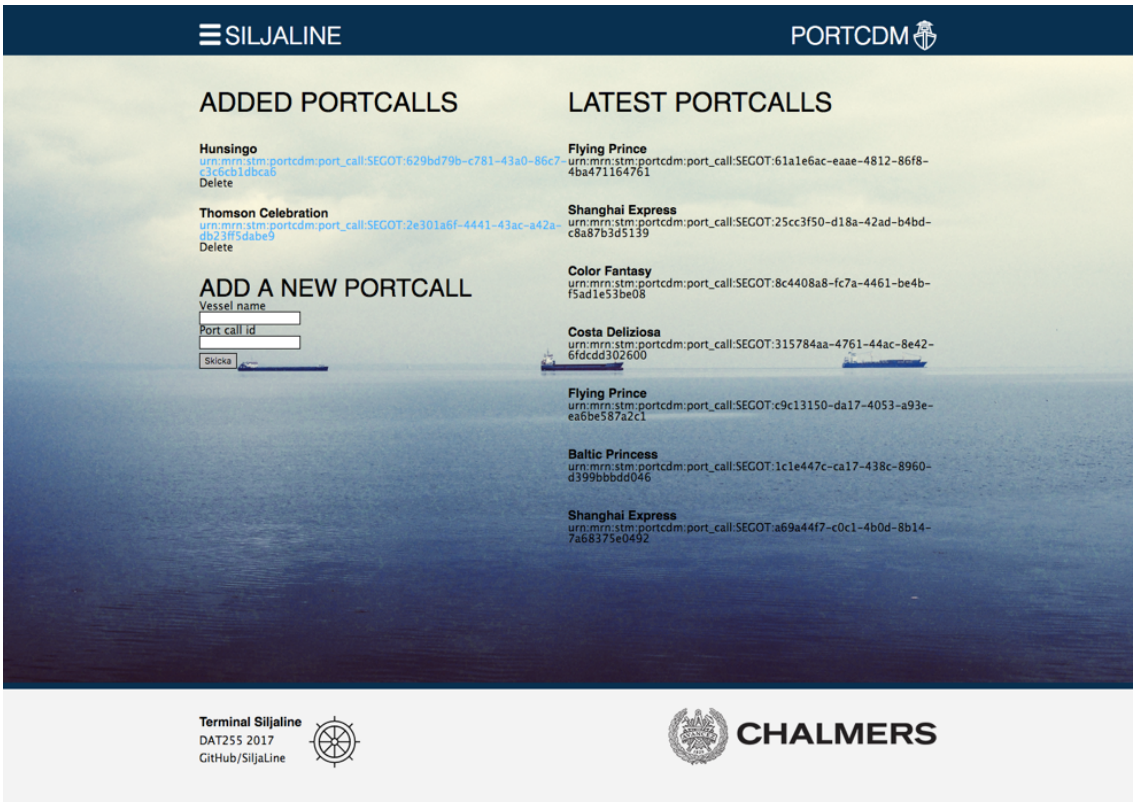


Figur 1: Till vänster visas en illustration för gruppens lärande under projektets gång. Till höger visas effektiviteten mätt i värdeadderande funktionalitet i relation till nedlagd arbetstid.

2 Prototyp

Produkten som tagits fram i projektet är en webbapplikation. Gruppen valde att bygga en webbapplikation då användarna, enligt representanter från PortCDM, skulle vara personal som arbetar med administrativa uppgifter på ett kontor. Därav antogs en webbapplikation vara bättre lämpad för användarna än exempelvis en mobilapplikation.

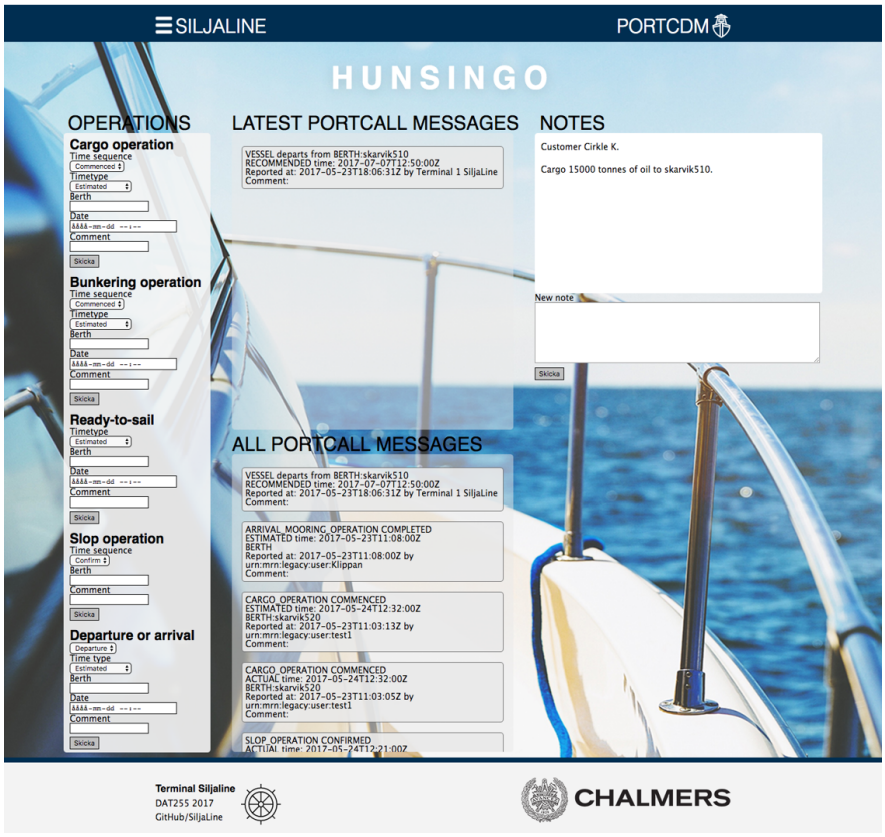
Webbapplikationen består av två delar. Den första delen är en startsida med övergripande information om hamnanlöp, hädanefter kallade anlöp eller portcalls. Från denna sida går det att öppna en ny sida med detaljerad information om ett specifikt anlöp, vilket är den andra delen av webbapplikationen. I figur 2 nedan visas hur förstasidan ser ut. Till höger i bilden under rubriken *Latest portcalls* visas de senaste anlöpen som skickats in till PortCDM med respektive fartygsnamn, detta för att terminalen ska kunna få en snabb överblick över vad som händer i hamnen. För att erhålla mer information om ett specifikt anlöp kan användaren kopiera id:t för anlöpet i listan och skriva in det i rutan under rubriken *Add a new portcall* tillsammans med fartygets namn. Användaren kan sedan trycka på länken för det inlagda portcallet i listan under *Added portcalls* och förs då till applikationens andra sida.



Figur 2: Startsidan med ett antal portcalls tillagda i *Added portcalls*

På den andra sidan finns en lista med senaste portcall messages, och en lista med samtliga portcall messages, se figur 3. Denna uppdelning har gjorts för att terminalen snabbt ska kunna se de mest aktuella händelserna. På denna sida kan användaren även skicka in, för terminalen, relevanta portcall

messages till PortCDM genom formulären till vänster under *Operations*. Det finns också möjlighet att lägga till anteckningar för anlöpet med information som endast är av intresse för terminalen och som inte hanteras av PortCDM. Exempel på anteckningar kan vara vilken typ av gods som ska lastas eller lossas och om säkerhetsvakt har bokats vid hantering av viss typ av gods. Användaren kan återgå till förstasidan genom att klicka på *Silja Line* i det vänstra hörnet.



Figur 3: Sida med detaljerad information om anlöpet med fartyg Hunsingo

Prototypen klarade av det slutgiltiga scenariot i projektet. Funktionerna samt användargränssnittet uppskattades dessutom av representanter från PortCDM och andra personer som närvarade vid slutpresentationen. Exempelvis ville Jouni Lindberg från Sjöfartsverkets innovationsavdelning titta närmare på applikationen och var imponerad av vad gruppen åstadkommit.

2.1 Planer på utökad funktionalitet

Det fanns planer på att implementera fler funktioner och ytterligare förbättra användarvänligheten av applikationen. Önskvärt vore dock en närmare kontakt med anställda på terminalen för att kunna ta fram de funktioner som de har störst behov av.

Ett förbättringsförslag skulle exempelvis kunna vara att användaren inte behöver kopiera portcall id från *Latest portcalls* och skriva in det i rutan *Add a new portcall*. Det skulle vara mer användarvänligt om det gick att klicka direkt på *Latest portcalls* för att lägga till det i listan *Added portcalls*. En annan möjlig förbättring vore om listan med messages kunde uppdatera sig själv efter ett visst

tidsintervall för att användaren inte ska behöva uppdatera manuellt.

En annan funktion som skulle kunna implementeras är att varna användaren när tiderna för ett portcall uppdateras, exempelvis om estimerad tid för ankomst till kaj ändras. I nuvarande version måste användaren manuellt gå igenom meddelandena för ett portcall för att upptäcka om någon tid har förändrats men med den tänkta funktionen skulle användandet av applikationen underlättas och risken att information förbises minskar. I den nuvarande applikationen finns det heller inget som förhindrar att två fartyg bokas in till samma kaj vid samma tidpunkt, detta är något som inte borde kunna ske utan att applikationen förser användaren med en varning.

En mer långsiktig funktion som gruppen hade planer på att utveckla var att åskådliggöra informationen i portcalls i form av Gantt-scheman för varje kaj som terminalen arbetar vid. Ett Gantt-schema är betydligt lättare att läsa av än en lång lista med portcall messages och det skulle bli tydligare för användaren att se vilka kajer som är bokade. Ett varningssystem och ett Gantt-schema var två funktioner som representanterna från PortCDM förmedlade som önskemål. Det skulle dock, som sagt, vara klokt att först diskutera dessa förslag med terminalaktören innan de utvecklas så de verkligen är i linje med deras behov.

3 Tillämpning av Scrum

I detta delkapitel beskrivs hur gruppen använt Scrum som metodik under utveckling av webbapplikationen. Gruppen har upplevt att det funnits både för- och nackdelar med Scrum-metodiken. Överlag har det varit uppskattat att metodiken fått gruppen att planera och diskutera mer, om allt från hur produkten ska utvecklas till hur gruppen arbetar. Denna typ av kommunikation kommer inte alltid naturligt vid arbete i grupper. Samtidigt har det emellanåt upplevts som att mycket tid har lagts på just planerande, när gruppen egentligen velat fokusera på att utveckla produkten.

3.1 Reflektion kring rollerna i scrum

Enligt Kinberg (2015, s.130-134) finns tre roller i Scrum-metodiken: Produktägare, Scrum master och Scrum team. I följande avsnitt diskuteras de tre rollerna samt vad rollerna har haft för funktion i detta projekt. Vidare reflekteras kring hur gruppen skulle kunna förbättra användning av rollerna. Under avsnittet om Scrum team reflekteras även över hur samarbetet i gruppen fungerat generellt.

3.1.1 Produktägare

Vad gäller produktägare har gruppen upplevt vissa svårigheter. Kontakten med de egentliga produktägarna, det vill säga terminalarbetarna, har varit bristfällig. Gruppen fick möjlighet att träffa produktägarna under den sista sprinten och fick då en något tydligare bild av hur de arbetar. Dock var terminalarbetarna inte så insatta i PortCDM och det var därför svårt för dem att komma med konkreta förslag på hur prototypen kunde förbättras. Mötet gav dock en ökad förståelse kring hur arbetet på en terminal går till.

Som substitut för bristen på kontakt med terminalen har representanter från PortCDM fungerat som produktägare. Representanterna har haft en ungefärlig uppfattning om hur arbetet på en terminal ser ut, men inte vetat exakt vad som skulle tillföra värde till den egentliga produktägaren. Gruppen har dock anpassat sig efter förutsättningarna genom att utnyttja de personer som funnits tillgängliga. Exempelvis har Mathias Karlsson från PortCDM givit gruppen bra input varje vecka om hur prototypen kunde förbättras. Vid brist på information om vad som skulle kunna skapa värde för terminalen har gruppen emellertid vid vissa tillfällen fått ta på sig en del av produktägarens ansvarsområden, utan någon direkt kommunikation med varken PortCDM eller terminal. Exempelvis har gruppen själva skapat produktbackloggen, vilket diskuteras vidare under rubriken *Sprint planning, produktbacklogg och sprintbacklogg.*

Legoövningen förmedlade på ett konkret sätt vikten av en genomgående kontakt med produktägaren. Om god kommunikation hålls kan exempelvis produktägaren motivera syftet med önskade funktioner, vilket ökar utvecklarnas förståelse för produktägarens behov. Detta effektiviserar implementering av nya funktioner eftersom utvecklarna förstår avsikten med dem. Gruppen applicerade principerna från legoövningen till exempel då representanter från PortCDM efterfrågade en indikator på när säkerhetsvakt behöver bokas till ett anlöp. Efter vidare dialog med representanterna utvecklades sektionen med anteckningar kopplat till specifika anlöp. Där kan användaren själv lägga till en påminnelse om exempelvis bokning av säkerhetsvakt. För att få bättre insikt i terminalens behov och krav hade kommunikation på kontinuerlig basis med terminalen under projektet varit nödvändig. Kontinuerlig kommunikation med terminalen hade också varit fördelaktig eftersom det

då hade funnits en tydlig produktägare. Samtidigt hade det varit nödvändigt att få terminalarbetarna att förstå syftet med PortCDM så att de skulle kunna formulera sina unika behov.

Visserligen kontaktades PortCDM via mail redan under första sprinten med en förfrågan om att få träffa terminalarbetare, men gruppen hade kunnat vara mer ihärdig i att försöka komma i kontakt med terminalen. Detta hade dock inneburit mindre tid till att lösa andra problem.

3.1.2 Scrum master

Enligt Scrum-metodiken har Scrum mastern ett antal viktiga uppgifter (Kinberg 2015, ss. 133). Dessa kan kortfattat beskrivas som att se till att gruppen följer Scrum-metodiken, att upprätthålla kommunikation med produktägare och se till att gruppen inte störs under sprinten.

Christian Frithiof från 8 dudes in a Garage talade under sin föreläsning om att det krävs en "Scrum-fascist", det vill säga någon som är kompromisslös med att de olika delarna i Scrum följs, för att Scrum-metodiken ska fungera på bästa sätt. I detta projekt har ingen Scrum-fascist funnits i gruppen, men Scrum mastern har varit bra på att se till att de flesta viktiga momenten i Scrum genomförts. Övriga gruppmedlemmar har även bidragit till att rekommendationerna från Scrum-metodiken följts.

Kommunikationen med PortCDM, projektets kompletterande produktägare, har inte enbart genomförts av Scrum mastern utan av hela gruppen. Dessutom genomfördes besöket hos terminalen under den sista sprinten utan att Scrum master närvarade. Att ansvar för kommunikation med produktägare inte enbart varit Scrum masterns ansvar har dock inte upplevts som något problem med tanke på avsaknaden av tydlig produktägare. Om en tydlig produktägare funnits hade det varit nödvändigt för Scrum mastern att lägga tid på att upprätthålla kommunikationen med denne i syfte att hela tiden ha en uppdaterad och genomarbetad produktbacklogg. Särskilt stor vikt har inte heller behövt läggas på uppgiften att hindra gruppen från störningsmoment. Anledningen till detta är att den problematik med att produktägaren eller andra chefer kommer med nya förslag eller krav på gruppen mitt under en sprint inte funnits.

3.1.3 Scrum team och samarbete

Den tredje och sista rollen i Scrum-metodiken är Scrum-team, det vill säga gruppen som arbetar tillsammans för att utveckla produkten. I detta projekt har gruppen trivts mycket bra ihop och tagit ett gemensamt ansvar för produkten. Gruppen har varit bra på att utnyttja dess styrkor och efter ett tag uppnåddes en bra arbetsmetodik där gruppmedlemmarna lärde av varandra. Gruppmedlemmarna har också ställt upp för varandra om någon stötte på problem som uppdagades. Simon Hofverberg inspirerade gruppen genom sin föreläsning om hur Spotify utformar sina team och om hur agil mjukvaruutveckling används på företaget för att främja innovation. Det gruppmedlemmarna framförallt tog med sig från föreläsningen är att projektgrupper måste arbeta successivt för att finna de arbetssätt som är bäst anpassade för gruppen i fråga.

Överlag har arbetet i gruppen alltså fungerat bra. Dock har två övergripande problem påverkat arbetet. Det första var att kunskapsnivån i gruppen vid projektets start var låg i förhållande till vad som krävdes för att ta fram produkten. Det andra var att gruppmedlemmarna hade olika ambitionsnivå,

vilket medförde en ojämn arbetsbörda.

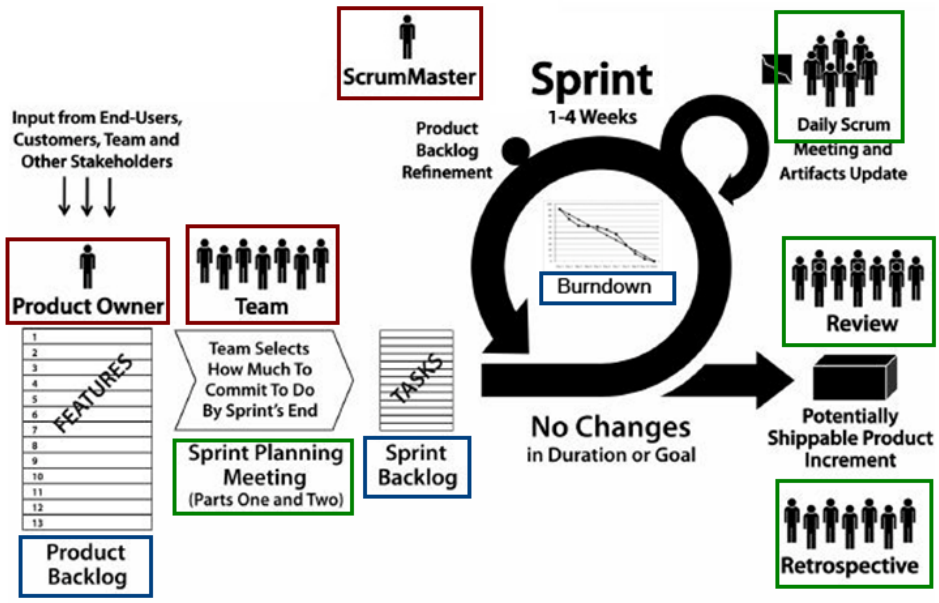
Vad gäller kunskapsnivån hade ingen i gruppen tidigare byggt en komplett applikation som hämtar data från en central punkt och publicerar denna i applikationen. Istället hade gruppmedlemmarna framförallt gjort mindre och väl avgränsade labbar i Java. Det var därför till en början svårt att förstå vilka lager som behövdes i applikationen och hur dessa skulle integreras. Dessutom var det svårt för gruppen att förstå hur kommunikation med den centrala datapunkten, det vill säga PortCDMs backend, skulle gå till. Till en början lärde sig halva gruppen hur kommunikationen med backenden skulle fungera medan den andra halvan lärde sig generellt hur en webbapplikation skapas. Tanken var att dessa två delar skulle integreras i ett senare skede. Arbetssättet ledde dock till att gruppmedlemmarna blev specialister inom olika områden. Gruppmedlemmarna behövde därför lära av varandra för att förstå hur systemen skulle integreras. Fler regelbundna träffar infördes därför där kunskaper utbyttes inom gruppen.

Det hade varit fördelaktigt att från början ha regelbundna träffar för kunskapsutbyte. Gruppens bus factor, som Christian Frithiof talade om, hade då stundtals inte varit så låg. Gruppen hade alltså inte i lika stor utsträckning varit beroende av ett fåtal personer för att kunna slutföra projektet. Däremot har uppdelning, likt den ovan beskrivna, i sig inte uppfattats som något negativt då det varit ett effektivt sätt för gruppmedlemmarna att lära sig nya verktyg. Detta beskrivs mer detaljerat under *Tillvägagångssätt vid användande av nya verktyg och teknik.*

Att det fanns olika ambitionsnivåer i gruppen framkom framförallt i slutet av projektet, då vissa gruppmedlemmar ville lägga ner mer tid på att få produkten så bra som möjligt, samtidigt som andra nöjde sig med att klara scenariot. Det gjorde att arbetsbördan blev ojämn i slutet då de som ville uppnå högre kvalitet satt fler timmar än övriga. Å andra sidan upplevde de som arbetade mer inga problem i att lägga fler timmar då de såg uppgiften som stimulerande. Christian Frithiof menade att det är positivt om en grupp är villig att lägga ner mycket tid på ett projekt då det tyder på en stark dedikation. Håkan Burden har dock flera gånger understrukit vikten av att inte arbeta fler timmar än beräknat, vilket gruppen trots allt haft i åtanke. Önskvärt hade varit att alla la lika mycket tid varje vecka, men framförallt att det hade funnits mer tydlighet gällande ambitionsnivå. För att undvika detta borde ambitionsnivå diskuterats mer utförligt i gruppkontraktet. För mer om detta se *Utvärdering av D1, D2, D3 och D4.*

3.2 Reflektion kring beståndsdelar i Scrum

Scrum-metodiken följer ett cykliskt mönster. I varje cykel, det vill säga varje sprint, genomförs ett antal möten med olika syften. Först ska sprinten planeras, under själva sprinten genomförs dagliga möten och i slutet av sprinten utvärderas både produkten och arbetsprocessen. Utöver dessa möten finns ett antal artefakter som ingår i Scrum, exempelvis produktbacklogg och sprinbacklogg. Figur 4 nedan illustrerar Scrum-metodikens cykliska mönster.



Figur 4: Scrum-metodiken och dess olika delar

Under denna rubrik återfinns reflektioner kring hur dessa beståndsdelar, både möten och artefakter, har tillämpats i projektet, hur det vore önskvärt att de tillämpats och hur gruppen borde agerat för att uppnå den önskvärda tillämpningen. Dessutom återges situationer då gruppen valt att avvika från Scrum-metodiken.

3.2.1 Sprint planning, produktbacklogg och sprintbacklogg

Varje sprint pågick från onsdag till onsdag. Vid början av sprintarna genomfördes ett sprint planning meeting för att strukturera arbetet inför nästa sprint. Under ett sprint planning meeting väljs de user stories med högst prioritet ut från produktbackloggen och förflyttas till sprintbackloggen. Produktbackloggen är en lista på funktioner som produktägaren önskar, och inför ett sprint planning meeting ska det finnas ett antal stories i produktbackloggen.

User stories kan dock även utformas för gruppen snarare än produktägaren. Detta skedde mycket i början av projektet, då många user stories utformades för att påskynda gruppens utveckling, exempelvis genom att ge gruppen i uppgift att genomföra tutorials. Detta var nödvändigt för att senare kunna ta fram funktioner med värde för produktägaren. Syftet med user stories utformade för gruppens utveckling är alltså att i förlängningen kunna skapa värde för produktägaren.

Dock ska produktbackloggen främst skapas i samarbete med produktägaren, men avsaknaden av en tydlig produktägare har gjort att detta inte varit möjligt. Produktbackloggen har därför uteslutande tagits fram av gruppmedlemmarna själva, även om produktägares önskemål tagits i beaktande. På grund av bristen av information gällande vad som skapar värde för terminalarbetarna har det dock inte varit helt självklart vad som ska ingå i den. Innehållet i produktbackloggen byggdes därför en hel del på egna antaganden. För att få en bättre grund för dessa antaganden, och få en tydligare bild av vad som skulle kunna ge värde, skapades en skiss med hjälp av verktyget Sketch. Skissen

var ett enkelt förslag på hur webbapplikationen skulle kunna se ut och fungera. Den var tänkt att användas som diskussionsunderlag vid möten med PortCDM eller terminalen samt inom gruppen.

Jan-Philipp Steghöfer föreläste om hur en produkt- och sprintbacklog bör utformas. Han menade att en backlogg bör bestå av vertikala skivor, det vill säga user stories bestående av en koppling mellan samtliga lager såsom användargränssnitt, databas och logik. Framtagandet av skissen stred mot det Steghöfer förespråkade då den inte var en vertikal skiva. Dessutom visade det sig senare i projektet att det inte fanns tillräcklig tid för att implementera all den tänkta funktionaliteten från skissen. Gruppen ansåg dock ändå att skissen var av nytta eftersom den som sagt fungerade som underlag för diskussion kring produkten.

Att endast skapa vertikala user stories var en utmaning även av andra anledningar. Exempelvis resulterade den initiala uppdelningen av arbetsuppgifter (som beskrevs under *Scrum team och samarbete*), då en del av gruppen arbetade med kommunikation med backend och andra med framtagning av en första webbapplikation, i horisontella user stories.

Ytterligare en utmaning för gruppen var att inte förändra user stories i sprintbackloggen under en pågående sprint. Enligt Scrum ska fokus ligga på att arbeta med user stories i sprintbackloggen tills att sprinten är slut, utan att sprintbackloggen förändras. Denna princip har i vissa fall frångåtts i projektet då nya user stories uppkommit och förverkligats under sprintens gång. Anledningen till detta kan delvis förklaras av att gruppen själv skapat den ursprungliga produktbackloggen. Hade detta skett med produktägaren hade förändring av sprintbackloggen troligtvis inte förekommit. Ett konkret exempel på när förändring i sprintbackloggen skedde var efter mötet med terminalarbetarna. Som tidigare nämnt hölls mötet under den sista sprinten. Med kunskap om hur arbetet på terminalen går till som grund valde gruppen att implementera möjlighet att göra anteckningar på sidan för ett specifikt portcall, vilket inte stod som en user story i varken produktbackloggen eller sprintbackloggen. Hade gruppen haft god kommunikation med terminalen från början hade denna user story existerat vid sprintens start. I detta fall avvek gruppen från Scrum-metodiken, men följde en riktlinje från det agila manifestet – ”responding to change over following a plan”. Överlag hade gruppen svårt att få user stories att följa INVEST-kriterierna och vissa fall även att sätta acceptanskriterier. Detta gällde framförallt under de senare sprintarna då störst fokus låg på att färdigställa produkten. För att bibehålla en hög kvalitet hos samtliga user stories hade ytterligare tid därför behövt läggas på utformande av dem.

För att underlätta prioritering och utformning av produktbackloggens innehåll hade en tydlig produktägarroll underlättat. Som stöd för gruppen vid framtagande av produktbacklogg hade en utförlig internetbaserad informationssökning kring terminalarbete kunnat genomföras. Det hade då möjligtvis varit enklare att avgöra vilken funktionalitet som skapar värde för terminalen. Något annat som underlättat utformning av produktbacklogg hade varit om gruppens förkunskaper varit större. Gruppen hade i sådant fall snabbare fått en bild av vilka lager som krävdes för att skapa webbapplikationen och hade då från start kunna skapa vertikala user stories. En åtgärd som gruppen kunde vidtagit hade kunnat vara att rådfråga andra grupper om hjälp för hur de utformade sina vertikala user stories och tasks. Exempelvis framgick det under halvtidsutvärderingen att den andra terminalgruppen kommit underfund med detta snabbt och kunde därför komma med tips.

3.2.2 Tidsplanering, task-uppdelning och estimering

Tidsplaneringen för varje sprint utgår från en bestämd arbetstakt, så kallad velocity. För att kunna beräkna velocity uppskattades innan projektets start hur många arbetstimmar per person och sprint som fanns tillgängligt i gruppen. Under sprint planning meeting, alltså vid varje ny sprint, lades så många user stories i sprintbackloggen att summan av tidsestimaten (som gruppen också uppskattade) hos dem var lika med velocityn. Därefter delades user stories ut till olika gruppmedlemmar. Ofta fick flera gruppmedlemmar dela på en user story. I dessa fall delades antingen tasks upp mellan personerna eller genomfördes i form av pair programming.

Som diskuterat under *Scrum team och samarbete* blev det i slutet av kursen så att vissa gruppmedlemmar lade ner fler arbetstimmar än andra. Avvikelse från den ovan beskrivna tidsplaneringen skedde alltså i syfte att få en så bra prototyp som möjligt till slutredovisningen.

Som nämnt ovan estimerades hur lång tid varje user story skulle ta, för att kunna uppskatta hur många user stories som skulle kunna genomföras under en sprint. En user story med en effort av åtta beräknades kräva åtta arbetstimmar. Dessa estimat var mycket svåra att ta fram och ofta blev uppskattningarna felaktiga. Det var något som gruppen hade erfarit redan innan projektets start vid legoövnigen. Framförallt tog vissa stories med en låg effort betydligt längre tid att genomföra än väntat.

Samtidigt visade det sig i vissa fall att gruppen lagt mycket tid i onödan på att genomföra en uppgift. Exempelvis lades en stor mängd arbete i början på att utnyttja en lokal backend genom att använda VirtualBox. Det visade sig dock efter mycket nedlagd tid att PortCDM färdiga developer-backend fyllde samma syfte. Gruppen spenderade också mycket tid på att testa olika sätt att utnyttja PortCDMs APIer. Både Connector Lib i Java och API-klienter genererade av Swagger testades innan gruppen valde att använda barebone-metoden. Ibland var det även en utmaning att dela upp user stories i tasks, framförallt i början av projektet. Det berodde på att gruppen inte fått en förståelse för alla lager i prototypen än, vilket gjorde det svårt att konkretisera vad som vad som var nödvändigt att utföras för att klara av user storien.

Både svårigheterna att estimera och dela upp user stories i konkreta tasks berodde alltså till största del på gruppens brist på förkunskaper. Enligt Adzic (2012) ska processen kunna förenklas genom att stories bryts ner i enklast möjliga delar via the hamburger method men även detta visade sig svårt på grund av den inledande kunskapsbristen. I takt med att kunskaperna ökade under projektets gång blev dessa aktiviteter dock enklare att genomföra för gruppen. Även i detta fall hade ytterligare kommunikation med andra grupper möjligtvis underlättat. Den andra terminalgruppen hade kunnat vara till hjälp då de arbetade med samma aktör och hade dessutom mer erfarenhet av att bygga webbapplikationer.

3.2.3 Daily scrum

Daily scrums är korta, dagliga möten som hålls löpande under sprinten. Under daily scrum ska alla i gruppen besvara följande tre frågor:

1. Vad har gruppmedlemmen gjort sedan förra mötet?

2. Vad ska gruppmedlemmen göra under dagen?

3. Ser gruppmedlemmen något hinder för det som ska göras under dagen?

Då projektet genomfördes på halvtid hölls inte dessa möten dagligen, utan i genomsnitt två gånger i veckan.

I början hade gruppen problem med att hålla daily scrums korta. Detta berodde på att många frågor dök upp när någon i gruppen berättade vad de hade gjort. Eftersom övriga i gruppen ville förstå problemen direkt ställdes en mängd följdfrågor som krävde extra tid. Därav skedde ibland en avvikelse från mallen för hur daily scrums bör utföras. Under projektets gång började dock gruppen arbeta för att hålla daily scrums korta, genom bland annat stand-up-meetings. Att stå upp under mötena blev en påminnelse om att mötet skulle hållas kort. Dessutom lyckades mötestiden förkortas då en större medvetenhet om hur daily scrum är tänkt att genomföras rådde inom gruppen. Några veckor in i projektet accepterade också gruppen att alla inte behövde förstå allt som de andra gjort. Istället kunde de som var intresserade av att förstå hur någon i gruppen löst ett problem be om att få en förklaring efter daily scrum. I slutet av projektet lyckades gruppen nå den önskvärda situationen där daily scrum användes som tänkt i Scrum-metodiken. Mötena fungerade då som ett bra tillfälle för gruppmedlemmarna att förstå hur det gick för gruppen och för att be om hjälp av övriga i gruppen, istället för ett tillfälle där gruppmedlemmarna lärde varandra vad de lärt sig sen förra mötet. Kunskapsutbyte är viktigt, men det ska ske vid separata tillfällen och inte under Daily Scrum.

3.2.4 Sprint review

Då gruppen inte hade någon kontakt med terminalarbetare förrän i sista sprinten utfördes sprint reviews vid handledningstillfällena med representanter från PortCDM. Avsikten med en sprint review är bland annat att produktägaren ska få en chans att se det som åstadkommits under senaste sprinten. Under några sprint reviews, framförallt de tidiga, skedde dock ingen demonstration då ingen större utveckling av prototypen skett sedan föregående sprint.

Anledningen till att inga större framsteg uppnåddes, med avseende på prototypen, var att tiden under dessa sprintar främst gick till att försöka få förbättrad förståelse kring exempelvis utnyttjande av PortCDMs APIer och grunderna i de nya verktygen bakom webbapplikationen. På grund av att prototypen inte utvecklats i någon större grad under de tidiga sprintarna kändes det mer relevant att under sprint reviews ställa frågor till representanterna, snarare än att visa upp prototypen.

Under sprint reviews är det viktigt att utvärdera om målet som sattes upp under sprint planning uppnåtts. Under flera sprint reviews var de representanter från PortCDM som närvarade inte insatta i gruppens projekt i särskilt hög utsträckning. Det blev därför svårt för dem att avgöra om uppsatta mål uppnåtts. Att representanterna inte var så insatta i projektet gjorde också att feedbacken från dem i vissa fall inte kändes relevant. Exempelvis kom flera förslag på funktionalitet som inte var möjlig att implementera med tanke på gruppens förutsättningar.

Representanter från PortCDM kom dock även med värdefulla synpunkter under sprint reviews som påverkat utvecklingen av produkten. Exempelvis visar senare versioner av webbapplikationen både

fartygets namn och portcall id istället för endast portcall id. Denna ändring genomfördes efter en sprint review där representanterna menade att det skulle vara svårt för terminalarbetarna att skilja mellan anlöp enbart baserat på portcall id.

Sprint reviews hade varit mer givande om de utförts med personer som var väl insatta i gruppens projekt. Det hade varit optimalt om dessa personer var de riktiga produktägarna, det vill säga terminalarbetare. För att uppnå detta hade bättre kontakt med terminalen behövts, vilket diskuterades under *Produktägare*. Det andra alternativet hade varit att upprätthålla bättre kommunikation med representanter från PortCDM. Exempelvis hade gruppen, tillsammans med PortCDM, kunnat bestämma en representant med huvudsakligt ansvar för att följa gruppen i projekten. Mathias Karlsson var den representant som kom med mest feedback till gruppen och kunde därför ha haft uttalat huvudansvar för gruppen.

3.2.5 Sprint retrospective

I slutet av varje sprint genomfördes en retrospective där gruppen såg tillbaka på hur samarbetet fungerat. Fokus låg alltså på hur gruppens arbetssätt fungerade, snarare än den produkten som gruppen byggde. Följande tre huvudfrågor diskuterades och anteckningar från dessa möten finns i bilaga 1 - *Dokumentation av sprint retrospectives*.

1. Vad gick bra under sprinten?
2. Vad gick fel under sprinten?
3. Vad kan vi som grupp göra annorlunda för att förbättra?

Dessutom besvarades enkäten bakom KPI:n gruppupplevelse inför sprint retrospectives, vilken finns beskriven under *Utvärdering av D1, D2, D3 och D4*.

Gruppen uppskattade den tid för utvärdering som en retrospective medförde eftersom olika typer av problem kopplade till samarbetet inom gruppen då uppdagades. Arbetssättet kunde således förbättrats under projektets gång. Exempelvis framgick det tidigt att några i gruppen föredrog pair programming framför att arbeta själva. Fler inslag av detta infördes därför under resten av arbetet. Det var också under en retrospective som det visade sig att ett ökat kunskapsutbyte inom gruppen var nödvändigt, vilket beskrevs under *Scrum team och samarbete*. Något annat som upptäcktes under en retrospective var att dubbelt arbete utförts på grund av att vissa tasks inte följts fullt ut och att de överlappat varandra. Detta löstes genom en noggrannare uppdelning och utdelning av tasks.

För att förbättra utvärdering av en sprint hade det varit önskvärt att tydligare skilja på retrospectiven, sprint planning och sprint review. I projektet skedde alla dessa tre aktiviteter under onsdageftermiddagar då det fanns möjlighet att träffa representanter från PortCDM, vilket skapade ineffektivitet. Exempelvis avbröts retrospecitves ibland eftersom det inte var möjligt att styra exakt när sprint reviews skulle genomföras. För att undvika stress och att möten avbröts valde gruppen att de sista veckorna avsluta sprinten innan lunch och påbörja nästa efter lunch. För att få en ännu tydligare uppdelning av momenten hade en sprint kunnat avslutas på fredagar, och en ny påbörjas på måndagar. Detta betraktades dock inte som ett alternativ då sprintarna lämpade sig att börja och sluta på onsdagar. Ytterligare förbättringsmöjlighet av retrospective hade kunnat vara att ge

varje gruppmedlem mer tid till att på egen hand reflektera över den gångna sprinten innan gruppdiskussionen. Detta upplägg fungerade väl för gruppen med avseende på gruppupplevelseenkäten och hade därför kunnat applicerats i större utsträckning.

3.3 Övriga metoder

Utöver de metoder som diskuterats tidigare har gruppen även testat andra metoder som är vanliga inom agil mjukvaruutveckling. Vissa metoder fungerade riktigt bra för gruppen. Ett exempel är stand up meetings, vilket diskuterades under rubriken *Daily Scrum*. En annan metod, som också berörts, vilken fungerade bra för delar av gruppen var pair programming. Metoden var framförallt användbar när nya programspråk användes. Andra gruppmedlemmar trivdes dock bättre med att arbeta självständigt.

Enligt Cohn (2007) är Scrum of Scrums ett viktigt inslag som kan underlätta samarbetet mellan olika Scrum-teams genom fasta möten och tydliga frågeställningar. Scrum of Scrums var ett inslag i projektet som gruppen ansåg var givande. Dessutom var kommunikationen med andra grupper genom Slack till stor hjälp. Samarbete med andra grupper var framförallt ett stöd under slutet av projektet när grupperna hade liknande problem, exempelvis gällande version av portcall message. I början arbetade grupperna dock med många skilda verktyg vilket medförde att det var svårt att helt utnyttja värdet i Scrum of Scrum eller kommunikation via Slack.

Att lista Definition of Done var också till nytta. Det var givande för gruppen att ha diskuterat vilka krav som kunde ställas för att betrakta en user story som avklarad. Det föll sig därefter naturligt vad som var nödvändigt att genomföras vid avslutande av en task eftersom en konsensus rådde i frågan. Vid applicering av nya metoder arbetade gruppen för att tillämpa lärdomar om kontinuerlig förbättring från övningen i Kata. Övningen gav insikt i att det blir enklare att mäta förändring i prestation om gruppen medvetet genomför en eller ett fåtal förändringar i arbetsmetod. Gruppen strävade därför efter att inte genomföra ett för stort antal förändringar i arbetsmetod under samma sprint.

4 Utnyttjade verktyg

Nedan redogörs för de tekniska verktyg som varit nödvändiga att använda i projektet. Då förkunskapen inom de flesta verktyg och teknologier som använts varit låg har det i många fall varit nödvändigt för gruppmedlemmarna att lära sig saker som innan projektet var främmande. Därför ges en beskrivning av tillvägagångssättet för hur gruppen lärt sig använda de nya verktygen.

4.1 Verktyg för utveckling av webbapplikation

För att utveckla webbapplikationen användes Express, som är ett ramverk för att utveckla webbapplikationer i Node.js. Innan Express valdes undersöktes olika sätt att ta fram en prototyp som skulle kunna skapa kundvärde. De alternativa förslagen diskuterades men det var svårt att avgöra vilket sätt som var bäst utifrån gruppens förutsättningar. Därför bokades ett möte in med två IT-konsulter från företaget Hyperlab. Utefter produktägarens önskemål och gruppens förutsättningar rekommenderade IT-konsulterna att använda Express. Trots att gruppen hade störst erfarenhet av att arbeta i Java ansåg de att det skulle vara mer fördelaktigt att arbeta i JavaScript. Anledningen till detta är att det då skulle gå att dra nytta av att Express är ett smidigt verktyg för att utveckla webbapplikationer. Vidare poängterade IT-konsulterna att gruppen med stor sannolikhet skulle ha nytta av att lära sig JavaScript och Express för en framtida karriär.

Vid framtagande av webbapplikationen var det nödvändigt att använda HTML. Istället för ren HTML användes dock Pug för att kunna integrera JavaScript i HTML-koden och på så sätt skapa en koppling mellan backend och frontend. För att skapa ett mer användarvänligt GUI användes också CSS. Som databas utnyttjades MongoDB. Trots att gruppmedlemmarna enbart hade erfarenhet av att arbeta med SQL-baserade databaser föll sig valet av MongoDB naturligt då det är standard att använda ihop med Express. Dessutom visade sig MongoDB vara väldigt lättanvänt och intuitivt.

I efterhand tyckte samtliga gruppmedlemmar att valet av de olika verktygen för utveckling av webbapplikationen var bra eftersom det resulterade i en lyckad prototyp. Givetvis är det svårt att avgöra huruvida andra verktyg, som gruppmedlemmarna hade större vana vid innan projektets start, hade lett till en ännu bättre prototyp. Detta är dock tveksamt med tanke på rekommendationerna från IT-konsulterna. Att använda de valda verktygen har även varit uppskattat då det varit väldigt givande för gruppen att lära sig nya saker.

4.2 Ytterligare utnyttjade verktyg

Förutom de verktyg som använts för att ta fram webbapplikationen har ett flertal andra verktyg varit nödvändiga i projektet. För att hantera olika versioner av applikationen har Git använts. Som scrum board technology användes Trello och för kommunikation, både inom gruppen och med andra grupper användes Slack. Sketch nyttjades som skissverktyg och slutligen användes Google Drive för dokumenthantering, exempelvis för planering och retrospectives. Git var en obligatorisk del i projektet, resterande program valdes utifrån gruppens tidigare kunskaper. Valet av verktygen baserades även på att gruppen uppfattat att dessa är vanligt förekommande inom agil mjukvaruutveckling.

4.3 Tillvägagångssätt vid användande av nya verktyg och teknik

Det generella tillvägagångssättet vid upplärning av ett nytt verktyg har varit att endast en mindre del av gruppen lärt sig det nya verktyget till att börja med. När dessa känt sig någorlunda bekväma med det nya verktyget har de sett till att även övriga gruppmedlemmar lärt sig det. Nedan beskrivs mer specifika tillvägagångssätt för arbete med olika verktyg och teknologier där förkunskaperna varit låga.

Express
Ingen av gruppmedlemmarna hade tidigare erfarenhet av att arbeta med Express. Dock hade Mark tidigare testat ett annat ramverk för att utveckla webbapplikationer (under cirka 20 timmar) samt arbetat lite i JavaScript. Samuel hade erfarenhet av att ta fram statiska webbsidor. Mark och Samuel började därför lära sig Express mot slutet av den första sprinten genom att leta upp och genomföra tutorials på internet. Detta resulterade i att en ”Hello World”-applikation i Express kunde tas fram innan sprinten var slut. Övriga gruppmedlemmar lärde sig sedan Express genom att genomföra de tutorials som Mark och Samuel hittat. Gruppen upplevde dock fortfarande ett gap mellan medlemmarna med avseende på kunskaper inom Express. För att minska detta kunskapsgap träffades alla i gruppen under en kväll för att dela med sig av sina kunskaper. Detta visade sig vara en effektiv lösning på problemet.

HTML & CSS
Eftersom Samuel hade mest erfarenhet av att tidigare arbeta med HTML och CSS fick han huvudansvar för detta. Även Mark hade en del erfarenhet av HTML och CSS och kunde därför också ta visst ansvar. Även övriga gruppmedlemmar kunde bidra genom att de lärde sig under projektets gång. Inom HTML och CSS ställdes inte lika höga krav på att samtliga gruppmedlemmar skulle ha samma kunskapsnivå som inom Express. Istället tilläts att fokus lades på andra problemområden, vilket ansågs vara nödvändigt för att kunna leverera en färdig prototyp. Anledning till detta var framförallt att HTML och CSS integrerades i allt större utsträckning ju längre projektet löpte. Om ytterligare tid hade lagts på projektet hade mer arbete skett för att utjämna kunskapsnivån inom HTML och CSS.

MongoDB
Anne fick huvudansvar för att lära sig MongodDB och integrera databasen i webbapplikationen. Då hon hade lyckats med denna integration fanns ett antal metoder för att både sätta in och hämta ut information i databasen. Då övriga gruppmedlemmar behövde utnyttja databasen i sina tasks var det enkelt att utgå från dessa metoder. Att arbeta på detta sätt var tidseffektivt samtidigt som det resulterade i den avsedda effekten.

Backend
Något av det svåraste i projektet var att arbeta med kommunikation mot PortCDMs backend. Detta var besvärligt framförallt i början av projektet då det krävde kunskaper om hur APIer utnyttjas och HTTP-requests skickas. Samtidigt behövde gruppen vara införstådd i begrepp som portcalls och portcall messages. Redan i den första sprinten påbörjades därför arbetet med backenden genom att Rebecca, Anne och Nils jobbade för att koppla upp sig mot den. För att göra detta var det till stor hjälp att använda Restlet. Första steget i att komma i kontakt med backenden bestod därför i att kunna ta emot meddelanden via Restlet. Därefter integrerades funktionaliteten att ta emot med-

delanden i webbapplikationen. Att initialt utnyttja Restlet ansågs vara ett bra arbetssätt eftersom antalet problem som var tvungna till att lösas på samma gång minskade. Arbetssättet utnyttjades därför även då meddelanden skulle skickas till backenden.

Git
Då gruppmedlemmarna redan innan projektets start var medvetna om att versionshantering skulle ske via Git började samtliga lära sig systemet innan projektstart. Detta innebar dock att det inte fanns några tydliga acceptanskriterier för uppgiften. Det var därför ingen som fullständigt behärskade Git förrän efter ett antal sprintar. Detta var problematiskt framförallt då det i början ofta dröjde innan ny kod var tillgänglig för alla gruppmedlemmar. Därav borde det generella tillvägagångssättet för upplärning ha applicerats även på Git, det vill säga att ett mindre antal först ha borde lärt sig Git och sedan delat med sig av sina kunskaper.

Trello
Trello var ett nytt verktyg för samtliga gruppmedlemmar förutom Mark. Det blev därför naturligt att Mark lärde övriga gruppmedlemmar hur verktyget fungerar. Eftersom Trello är väldigt intuitivt var det inga problem för samtliga i gruppen att få samma färdigheter inom verktyget. Anne tog sedan över huvudansvaret för Trello. Detta ansvar var dock inte uttalat utan föll sig naturligt eftersom Anne var Scrum master. Då ansvaret inte var uttalat blev det dock lätt att glömma att flytta kort mellan listor mitt i en sprint eftersom fokus då låg på annat. Det hade troligtvis varit lättare att förflytta kort kontinuerligt om Anne hade haft officiellt ansvar för Trello. Hon skulle då kunna påminna övriga i gruppen om att flytta kort till dess att det blev naturligt.

Slack
Även inom Slack var det enbart Mark som hade förkunskaper och ansvarade därför för att övriga i gruppen lärde sig verktyget. För användande av Slack inom gruppen utnyttjades bara en kanal samt direktmeddelanden. Det hade möjligtvis varit fördelaktigt att utnyttja fler kanaler för att separera olika typer av kommunikation. Detta ansågs dock inte vara nödvändigt med tanke på det låga antalet medlemmar i gruppen.

Sketch
Samuel hade tidigare erfarenhet av Sketch och kunde på ett smidigt sätt ta fram en överenskommen skiss samtidigt som resterande medlemmar kunde fokusera på andra tasks. Om produktägaren hade efterfrågat fler skisser hade gruppen lagt mer tid på att utjämna kunskapsnivån inom programmet. Eftersom en skiss är en horisontell skiva hade gruppen dock argumenterat mot framtagandet av fler skisser.

5 Testning och defekter

Under utvecklingen av applikationen har enhetstester, systemtester, integrationstest och acceptanstester genomförts i syfte att säkerhetsställa att ny funktionalitet fungerar och beter sig på ett korrekt sätt samt är godkända av produktägaren. Nya funktioner har till en början testats var för sig, det vill säga genom enhetstester (unit testing), innan de integrerades med tidigare framtagna funktioner. Denna testning utfördes genom att testfunktioner skapades och resultatet eller responskoder skrevs ut i terminalen/cmd. Restlet har även varit till stor hjälp vid stegvis testning av nya funktioner. Om ett meddelande exempelvis skickades in från webbapplikationen var det möjligt att med hjälp av Restlet se om meddelandet kom fram, innan det fanns funktionalitet för att kunna se meddelanden från webbapplikationen. När funktionerna enskilt betedde sig enligt vad som var förväntat genomfördes systemtest för att undersöka om de fungerade korrekt tillsammans.

Integrationstester har även genomförts då webbapplikationen initialt var uppkopplad till en lokal backend och senare i projektet till en gemensam backend. Alla funktioner, till exempel skicka meddelande, skapa en kö och lista anlöp, testades mot den nya backenden. Ett exempel på något som uppmärksammades vid dessa tester var att versionerna för strukturen hos ett portcall message skiljde sig mellan den lokala och gemensamma backenden. Detta ledde till att vissa justeringar var nödvändiga att genomföras. Då applikationen integrerats med den gemensamma backenden upptäcktes också problem kopplade till portcall messages med tidstypen cancelled, vilket beskrivs mer i avsnittet *Defekter* nedan. Angående acceptanstester undersöktes användarupplevelsen med representanter från PortCDM. Ny funktionalitet demonstrerades för representanterna som med synpunkter på vad de var tyckte bra och vad som kunde göras bättre.

Ofta programmerades en ny funktion utan att tester för funktionen var specificerade i förväg. Möjligtvis hade det varit bättre om det fanns tydliga mål som ny funktionalitet skulle klara av innan utvecklingen av den påbörjades, det vill säga en mer testdriven utveckling. Genom att först uttrycka vilka tester som funktionen ska klara av, stegvis programmera funktionen och därefter kontinuerligt testa ny kod med alla tänkta tester minskar risken för defekter. Efter hand kan också fler testfall skapas. För att uppnå en mer testdriven utveckling är det viktigt att testerna är genomtänkta och täcker hela den nya funktionaliteten. Testerna skulle kunna utgå från acceptanskriterier som tagits fram för en user story, dock ställer detta högre krav på att sätta relevanta acceptanskriterier än vad gruppen gjort.

Ytterligare tillvägagångssätt för att förbättra testning kunde ha varit att inte låta samma person som utvecklat en ny funktion genomföra tester på den. Genom att endast tala om vad funktionen är tänkt att göra för personen som ska testa den kan mer objektiva tester genomföras, vilket möjliggör för att fler problem fångas upp. Det vore också önskvärt att acceptanstesterna genomförts med någon anställd från terminalen istället för PortCDM, men som tidigare nämnt var detta inte möjligt.

5.1 JSLint

För att undersöka om koden till gruppens prototyp innehöll några buggar användes JSLint, som är ett program vars syfte är att finna problem i JavaScript-filer. Under testet var inställningen för JSLint att filerna körs med Node.js. När filerna testades utnyttjades även en inställning som tillåter

att strängar markeras med ‘ istället för “ (allow single quote strings). Detta eftersom att det i JavaScript inte spelar det någon roll vilket som används.

Därutöver har en inställning om att JSLint ska tillåta whitespace mess använts, det vill säga att JSLint inte undersöker koden för bland annat onödiga mellanslag och radbrytningar. Motiveringen till detta är att radbrytningar och mellanslag inte påverkar funktionaliteten utan endast läsning av koden. Dessutom har vissa av de radbrytningar och mellanslag som JSLint inte godkänt ansetts underlätta läsandet av koden. Inställningen att JSLint ska godkänna “for statement” användes också.

Problemen som JSLint fann var av trivial karaktär och åtgärdades fort. Problemen berörde booleanska jämförelser, oanvända variabler, missade semikolon, och ologisk placering av funktioner. JSLint borde kontinuerligt ha använts i projektet under utvecklandet av applikationen för att tidigt upptäcka problem och undvika att samma misstag upprepas. Testning med JSLint applicerades enbart på JavaScript-filer, inte på några av filerna som hanterar det grafiska gränssnittet i applikationen. Dessa filer hanterar ingen funktionalitet utan enbart utseende. Därav ansågs testning av dessa inte vara nödvändig.

5.2 Defekter

En defekt är då mjukvara beter sig oväntat eller felaktigt i relation till dess krav. Applikationen uppvisade några sådana defekter. Två identifierade defekter är kopplade till funktionen *Add a new portcall* på applikationens startsida. Den första defekten är att användaren kan fylla i ett ogiltigt portcall id i rutan och klicka på skicka. Om användaren sedan försöker följa länken kraschar applikationen. Den andra är att om användaren lägger till ett anlöp, utan att fylla i ett portcall id, så går det inte att ta bort anlöpet från *Added portcalls*. Bägge dessa defekter hade varit möjliga att åtgärda genom att exempelvis lägga in krav på vad användaren får ange för information i formuläret vid *Add a new portcall* eller på vad för information som får läggas in den bakomliggande databasen, men detta prioriterades inte då andra funktioner var viktigare att implementera för att klara scenariot.

Ytterligare en defekt upptäcktes i projektets slutskede. När ett portcall message skickas in med tidstypen (timetype) cancelled försvinner portcallet från PortCDM. Om en användare försöker gå in på sidan för det specifika portcallet så kraschar applikationen. Detta eftersom det aktuella portcallet inte finns kvar i PortCDMs backend. Denna defekt upptäcktes sent eftersom det inte hade förekommit ett meddelande med tidstypen cancelled tidigare, vid arbete mot lokal backend. Inte heller detta åtgärdades då fokus låg på att klara scenariot i detta skede.

6 Utvärdering av D1, D2, D3 och D4

Nedan görs en återkoppling till och utvärdering av de inlämningar som gjorts under kursens gång och hur de använts i projektet.

D1 – Strategier och gruppkontrakt
Strategierna utgick från det som gruppen lärt sig under legoövningen. Följande tre strategier formulerades:

1. Dela upp arbetsuppgifter tidigt och tydligt.
2. Upprätthåll god kommunikation och samarbete mellan grupper.
3. Samarbeta och kommunicera med kunden.

Överlag lyckades gruppen väl med att följa den andra strategin men upplevde problem med den första och tredje strategin. Gruppen hade problem med att följa den första strategin då det visade sig vara svårt att bryta ned user stories, framförallt på grund av gruppmedlemmarnas initiala kunskapsnivå. Detta var dock något som gruppen blev bättre på med tiden. Att följa den tredje strategin var framför allt svårt på grund av att terminalen inte visade sig vara delaktig i projektet i samma utsträckning som gruppen först trodde den skulle vara.

Gruppkontraktet var nyttigt eftersom gruppen fick reflektera över hur samarbetet skulle se ut. Kontraktet ingav också en trygghet i att alla visste vad som förväntades. Dock borde resonemang kring ambitionsnivå förts i högre utsträckning i kontraktet. Exempelvis hade betyg kunnat diskuteras och formuleringar såsom att arbetet ska bli "så bra som möjligt" hade kunnat förtydligats. Lyckligtvis har samarbetet fungerat bra och hela gruppen är nöjda med prestationen.

D2 – KPIer
De KPIer som valdes var en burn down chart, gruppenkät, och kundenkät. Burn down chart används för att mäta arbetstakten. Gruppenkäten hade till syfte att mäta trivseln i gruppen och kundenkäten var menad att mäta produktägarens nöjdhet med prototypen. Både grupp- och kundenkäten bestod av ett antal påståenden som graderades på en skala ett till fem och möjlighet att lämna textsvar för att förtydliga.

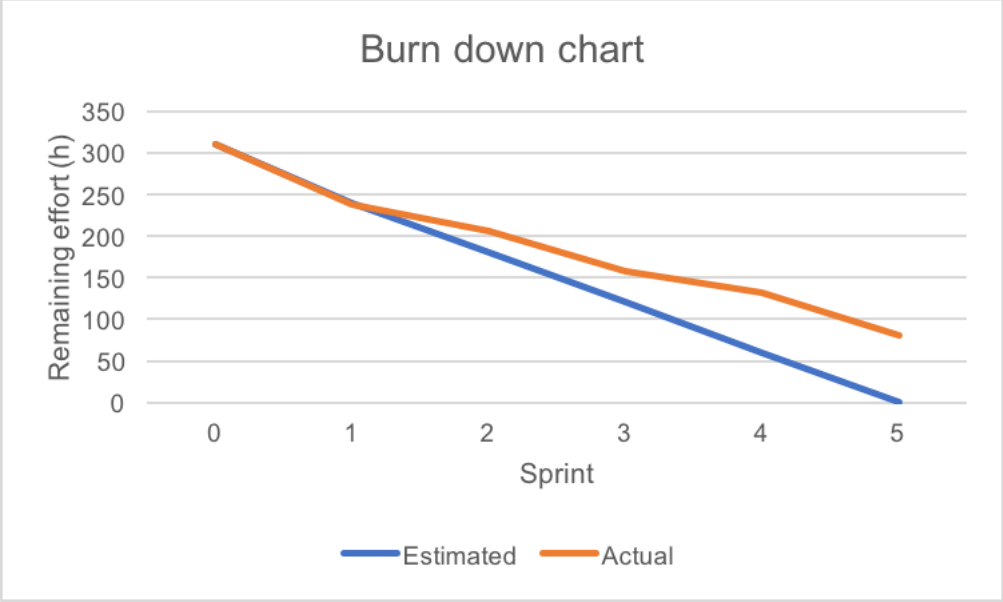
Burn down charten användes för att kunna kartlägga var gruppen befann sig i relation till projektmålet. Som tidigare nämnt var det svårt att kontinuerligt uppskatta olika tasks effort-nivå men burndown-charten gav en indikation på hur gruppen presterade från vecka till vecka. Burn down chart för projektet illustreras i figur 5 nedan. I figuren framgår det att gruppen inte hann färdigställa så många user stories som avsetts. Därav avviker grafen *Actual*, som visar hur många story points som avklarades, från grafen *Estimated*, som visar hur många story points som var planerat att avklaras under respektive sprint.

En anledning till avvikelsen är att gruppen i vissa fall hade väldigt stora user stories, vilket var fallet under den andra sprinten. En stor user story som nästan är uppfylld vid slutet av en sprint innebär en lika stor avvikelse som om den inte ens skulle vara påbörjad. Den stora user storyn i den andra

sprinten gällde att kunna skicka meddelanden till PortCDM. För att uppfylla acceptanskriteriena skulle det vara möjligt att skicka meddelanden både från Restlet och från webbapplikationen. Gruppen lyckades skicka meddelanden från Restlet men inte från applikationen. En bättre nedbrytning av user storyn skulle medfört att avvikelsen mellan kurvorna i figur 5 hade varit mindre.

Anledningen till att estimeringen för sprint fyra inte stämde berodde främst på kandidatarbetet. Tiden som gruppmedlemmarna behövde spendera på sina respektive kandidatarbeten underskattades vilket resulterade i att inte alla user stories uppfylldes. Under de övriga sprintarna har estimeringarna varit goda vilket syns i figuren genom att de båda graferna i stort sett är parallella.

Grafen *Actual* når dock inte ner till x-axeln även om det i slutet fanns en produkt som klarade av scenariot. Det beror troligtvis på att gruppen i slutet av projektet inte lade stor vikt vid att följa sprintbackloggen. Exempelvis byttes vissa user stories ut mitt under en sprint samtidigt som oavklarade user stories fanns kvar i sprintbackloggen. Det blev en spurt där fokus låg på att klara scenariot snarare än följa Scrum-metodiken. Detta gör att burn down chart blir något missvisande.

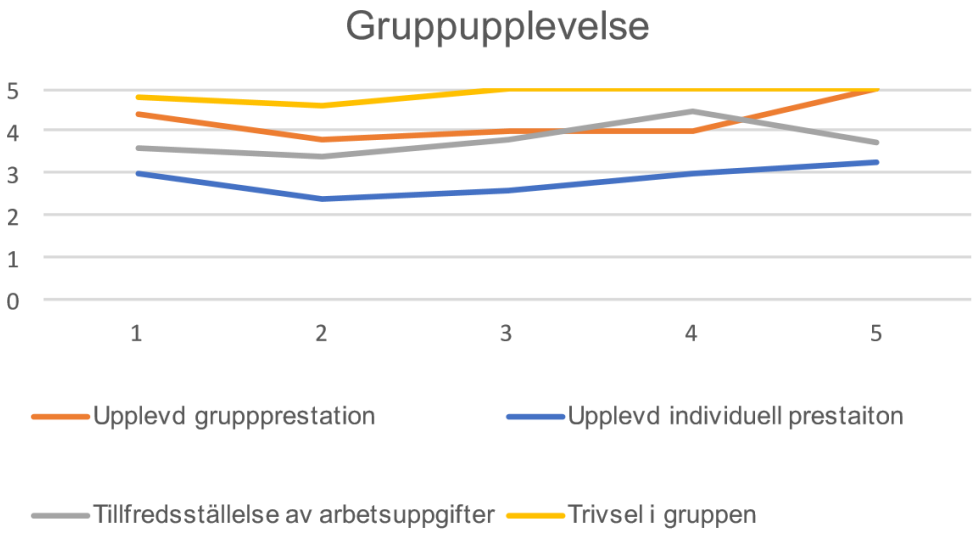


Figur 5: Graf över verklig arbetstakt och estimerad arbetstakt

Gruppenkäten var en uppskattad del av processen som gav större insikt i hur gruppmedlemmarna uppfattade det pågående samarbetet. Som tidigare nämnt bestod enkäten av både möjlighet att gradera påståenden på en skala och lämna kommentarer. Kommentarerna blev en grund för att motivera graderingen och gav ett mer detaljerat underlag för att förbättra problem som dök upp. Fullständigt resultat av enkäten går att återfinna i gruppens repository på GitHub.

I figur 6 nedan har resultatet av graderingarna i gruppenkäten för varje sprint sammanställts. Ett intressant genomgående tema var att individer rankade sin egen prestation lägre än gruppens prestation. Gruppmedlemmarna kände alltså att just deras bidrag inte var tillräckligt men att gruppens prestation i helhet var i linje med uppsatta mål. I figuren syns även att grupprestation och individuell prestation sjönk något efter första sprinten. Detta berodde främst på att gruppmedlemmarna

under den andra sprinten insåg att kunskaperna som krävdes för att genomföra projektet skiljde sig avsevärt från deras förkunskaper. I takt med att gruppmedlemmarna samlade på sig nya kunskaper och löste många problem steg dock prestationen, vilket också återspeglas i figuren. Det var även genom denna KPI, och på grund av retrospective, som gruppen kom fram till att det var mer effektivt att ha fler regelbundna träffar.



Figur 6: Graf över gruppupplevelsen

Kundnöjdhetsenkäten visade sig svår att implementera då gruppen inte visade prototypen under vissa reviews, som diskuterat under *Sprint review*. Istället fördes en kontinuerlig dialog med representanterna från PortCDM.

D3A – Initial product backlog
Den första produktbackloggen byggde på introduktionsföreläsningen med PortCDM samt Pattons (2008) resonemang om att backloggen bör innehålla en ryggrad av user stories som garanterar en MVP (minimum viable product). Den inledande backloggen utformades alltså med en tydlig struktur, övergripande teman, stora epics samt mindre user stories som kunde avklaras under första sprinten. Gruppen var dock inte var särskilt insatta i terminalens arbete i detta skede vilket resulterade i en del oanvändbara user stories. Vid detta tidiga skede i kursen fanns heller inte några minimum requirements från PortCDM att ta i beaktande vilket senare påverkade projektets riktning. Att initialt tänka i form av user stories var dock en bra övning som kastade in gruppen i Scrum-metodiken på ett effektivt sätt.

D3B – Business model canvas
Att utforma en business model canvas i en tidigt fas visade sig vara svårt eftersom gruppen saknade erfarenhet och kunskap om vad för sorts produkt som kunde vara aktuell i sammanhanget. Gruppen hade tidigare erfarenheter om hur modellen teoretiskt skulle utformas om det finns en färdig produkt eller tjänst. Nu skulle dock en affärsplan tas fram för en hypotetisk produkt utifrån syftet med PortCDM och terminalens roll i hamnen. Detta resulterade i en business model canvas som

inte blev särskilt användbar senare i projektet. Möjligtvis hade affärsmodellen kunna ge något typ av stöd till projektet om gruppen sett tillbaka på den och gjort justeringar, likt principerna i Lean Startup.

D4 – Halvtidsutvärdering
Halvtidsutvärderingen var givande för gruppen eftersom fokus då lades på att reflektera över vad som gått bra och utmaningar som inte ännu var avklarade. Det var intressant att höra den andra terminalengruppens erfarenheter där vissa stämde väl överens med gruppens egna samtidigt som andra skilde sig drastiskt. Till exempel hade den andra gruppen problem med att tasks underskattades vilket var tvärtemot gruppens egna upplevelser. Mötet med den andra gruppen gav också nya perspektiv i hur Scrum kunde användas, exempelvis kortare daily scrums. Att det fanns tid till att göra förändringar efter halvtidsutvärderingen innan deadline var positivt. Vid denna tidpunkt hade också minimum requirements för de olika grupperna publicerats vilket medförde en viss förändring i fokus och mål.

7 Summering

En viktig lärdom som gruppen erhållit genom projektet är att det går att skapa en fungerande prototyp med nya tekniker och verktyg på kort tid genom engagemang för projektet och för varandra. Genom enkel och snabb kommunikation och frekventa möten har gruppen lyckats hålla ett bra kunskapsutbyte. Att sätta sig in i Scrum-metodiken har också krävt viss ansträngning men har varit till hjälp då gruppen avsatt tid åt planering och reflektion. Genom planeringen fick gruppen en gemensam målbild. Reflektion gav möjlighet att identifiera förbättringsmöjligheter och implementera åtgärder under kursens gång, något som gruppen tar med sig inför framtida projekt.

Avslutningsvis kan det tilläggas att kursen har varit utmanande och stundtals kaotisk men överlag mycket givande. Gruppmedlemmarna har även uttryckt att detta projekt är det närmaste de kommit arbetslivet under sina tre år på Chalmers.

Referenser

[1] Adzic, G. (2012). *Splitting user stories – the hamburger method.*
Hämtad från: https://gojko.net/2012/01/23/splitting-user-stories-the-hamburger-method/

[2] Cohn, M. (2007). *Advice on Conducting the Scrum of Scrums Meeting.*
Hämtad från: https://www.scrumalliance.org/community/articles/2007/may/advice-on-conducting-the-scrum-of-scrums-meeting

[3] Kinberg, H. (2015). *SCRUM AND XP FROM THE TRENCHES: How we do Scrum.*
Hämtad från: https://www.infoq.com/minibooks/scrum-xp-from-the-trenches-2

[4] Patton, J (2008). *The New User Story Backlog is a Map.*
Hämtad från: http://jpattonassociates.com/the-new-backlog/

Bilaga 1 - Dokumentation av sprint retrospectives

Retrospective 1, 26/4

What went well during the sprint cycle?
Vi delade upp arbetet bra, det var skönt att jobba i mindre grupper, men inte vara helt själv. Vi uppnådde det vi hade planerat under sprinten och uppskattade alltså ganska bra. Vi gjorde en bra task breakdown.

What went wrong during the sprint cycle?
Vi jobbade lite fler timmar än vad vi skulle. Det var svårt att veta vilket "spår" man skulle köra på vad gäller hur vi skulle kommunicera med backenden, vilket gjorde att man la ner många timmar på att t.ex. försöka förstå hamnens API och sedan kommer det ändå inte användas. Dock vet vi inte hur man skulle gjort annars.

What could we do differently to improve?
Inte jobba för många timmar. Faila fast om vi väljer fel spår (om möjligt). Se till att arbetsfördelningen blir jämn, denna sprint drog vissa ett tyngre lass.

Retrospective 2, 3/5

What went well during the sprint cycle?
Vi klarade de flesta av tasksen. Vi fick fram en bra skiss. Alla har fått lära sig mer om JS.

What went wrong during the sprint cycle?
Vi har inte lärt oss så mycket mer om backenden (några av oss har!). Vi har inte setts tillräckligt mycket i gruppen och missat daily scrums, det har gjort att vi har försökt uppfinna hjulet på olika håll. Alla har inte lagt tillräckligt mycket tid.

Vi delade inte upp userstoriesarna i tasks, men de var ganska tydliga. Blir så mycket tid på planerande annars. Våra userstories var inte smala delar av kakan, en del var fokus på backend och en del frontend, men det är svårt att få med alla lager innan man förstår grunderna i ett lager. Var trötta när vi skrev dem så ibland var de lite otydliga.

What could we do differently to improve?
Nästa sprint ska vi sitta mer tillsammans och göra en koll så kunskapsnivån jämnar ut sig, så att alla kan bidra lika mycket. Vi ska försöka komma igång tidigt under sprinten.

Retrospective 3, 10/5

What went well during the sprint cycle?
Bra catch-up session. God pizza. Bättra på att samarbeta och träffas. Bra uppdelning av tasks i förhållande till vad man är bra på, den som kan pug kan få fokusera på det. Vi var bättre på att

dela upp user stories i tydliga hanterbara tasks.

What went wrong during the sprint cycle?
Daily scrum tar för lång tid. Om vi hade haft tid hade det varit nice att sitta mer ihop, men kandidatarbetet har fått ta fokus.

What could we do differently to improve?
Det blir lättare att ses mer nu när vi inte har kandidaten. Försöka separera daily scrum från möten där vi lär oss av varandra.

Retrospective 4, 17/5

What went well during the sprint cycle?
Satt mycket tillsammans, man börjar förstå saker och kunskapsnivån är jämnare. VI har lyckats ha en snabb daily scrum vilket var bra.

What went wrong during the sprint cycle?
TIdsbrist pga kandidat. Svårt när vi inte sitter ihop och helt andra saker än det man vill sitta med krånglar, typ server eller ny xml-version. Problem med Git.

What could we do differently to improve?
Sitta mer tillsammans. Bli bättre på Git.

Retrospective 5, 24/5

What went well during the sprint cycle?
Vi kom väldigt långt. Vi klarade demon. Vi känner oss riktigt bekväma med att använda systemet. Vi fick en slutprodukt vi är nöjda med.

What went wrong during the sprint cycle?
Vi jobbade obekväma arbetstider, på söndagen och på kvällen. Vi gick ifrån våra userstories eftersom vi insåg vissa saker under sprinten och vi ville ha en bra slutprodukt, och för att vi fick träffa terminalen mitt i sprinten.

What could we do differently to improve?
Försöka att inte gå ifrån scrum-metodiken.