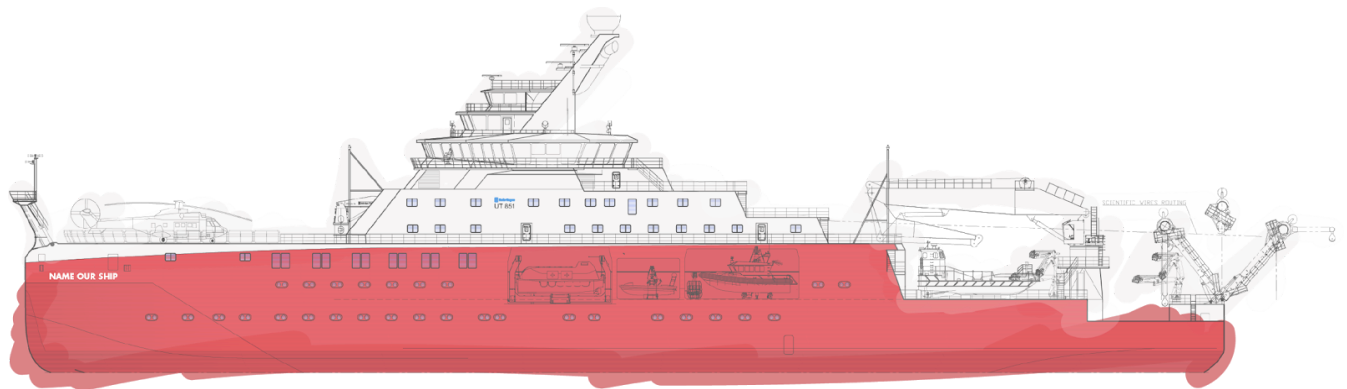


Boaty McBoatface

Ett agilt mjukvaruprojekt i DAT255 - Software Engineering



Anton Alexandersson	19950809-4712
Oskar Hagman	19910219-1336
Hannes Lagerroth	19930717-1919
Teodor Martinsson	19950915-5371
Aron Sai	19950313-1675



CHALMERS
UNIVERSITY OF TECHNOLOGY

Abstract

Denna rapport är en reflektion på åtta veckors arbete i kursen DAT255 - Software Engineering på Chalmers Tekniska Högskola i Göteborg, där vi använde agila metoder för mjukvaruutveckling till att ta fram en Java-applikation med målet att göra det enklare för hamnpersonal (mer specifikt fartygsagenter el. *shipping agents*) att visualisera och kommunicera när skepp antrar och lämnar en hamn. Gruppen testade på flera agila processer och verktyg, några som fungerade och vissa som inte gjorde det. Flertalet förändringar gjordes under veckorna för att kontinuerligt förbättra dessa. Resultatet blev en java-applikation som kommunicerade med PortCDM genom deras backend, och målade upp det visuellt med JavaFX.

[Inledning](#)

[Scrum](#)

[PortCDM](#)

[Vår applicering av Scrum](#)

[Roller och ansvarsområde](#)

[Tidsfördelning](#)

[Processer](#)

[Sprint Planning](#)

[The situation as it is - Hur vi gjorde Sprint Planning](#)

[What you would like it to be - Hur skulle vi gjort Sprint Planning?](#)

[Sprint Retrospectives](#)

[The situation as it is - Hur vi gjorde retros](#)

[What you would like it to be - Hur skulle vi gjort retros?](#)

[Sprint Reviews](#)

[The situation as it is - Hur vi gjorde reviews](#)

[What you would like it to be - Hur skulle vi gjort reviews?](#)

[Verktyg och teknologier](#)

[Våra verktyg för mjukvaruutveckling](#)

[Våra verktyg för projektledning](#)

[Vad hade vi gjort annorlunda om vi börjat om idag?](#)

[Reflektion av relationen mellan prototyp, process, och stakeholder-värde](#)

[Evaluation of D1 - D4](#)

[D1: Strategies for scrum implementation and social contract](#)

[D2: Three KPIs to monitor the implementation of Scrum](#)

[D3: An initial product backlog and business model canvas](#)

[D4: Half-time evaluation](#)

[Sammanfattning](#)

[Artefakter från prototypen](#)

[FindBugs report on final release](#)

[GitInspector](#)

[Documentation of sprint retrospectives](#)

Inledning

I de första stora mjukvaruprojekten anammades den projektledningsmetodik som fanns, designad för att bygga nya hus, bilar eller fabriker. Med noggrann planering, och strikta faser av design, implementering, verifiering, och underhåll. Att bygga en ny bil krävde en lång designfas, nya fabriker, hård säkerhetsverifiering, och när den väl var ute på vägarna gick den inte att ändra på. Märker man att något ändå var fel var det dyra återkallelser av upp till miljoner bilar.

I en värld av modern mjukvarutveckling är dock inte samma axiom sanna. Att ändra produkten är inte längre en dyr historia då man kan pusha kod till produktion på sekunder, och designfasen kan minimeras till att bygga en minimalt fungerande produkt på kanske endast några veckor, och sedermera iterera på den.

I denna kursen ämnade vi att utveckla vår förmåga att bedriva agil mjukvarutveckling genom att anamma metodik från *Scrum* för att utveckla en fungerande prototyp som ska vara värdeskapande för kunder i sjöfart och hamnkontroll, med användning av *PortCDM*-projektets backend.

Scrum

Scrum är en metodik för projektledning som är en vanlig implementation av agil mjukvaruutveckling. En nyckelingrediens i Scrum är att arbetet är uppdelat i *Sprints*¹ på vanligtvis 2-3 veckor.

Innan varje sprint så planeras arbetet i en *Sprint Backlog*, under sprinten så har man dagliga *Daily Scrum* där man har ett kort uppdatering kring hur teamet ligger till, samt vad de ska göra idag. Efter en sprint så sker en *Sprint Review*² där resultatet från sprinten presenteras och utvärderas, samt en *Sprint Retrospective*, där teamet reflekterar över sprinten och identifierar problem som kan åtgärdas inför nästa sprint.

I applicering av scrum så har teamet vanligtvis tre distinkta roller, en *produktägare* som bestämmer vad som ska prioriteras, och sätter teamets roadmap. En *Scrum master* vars uppgift det är att det operationella arbetet och processerna flyter på så smidigt som möjligt, och ett *utvecklingsteam* som givetvis består av mjukvarutvecklare, men även designers, testare, analytiker, eller andra nödvändiga roller.

PortCDM

PortCDM är ett projekt med ambition att digitalisera kommunikationen mellan samtliga aktörer involverade i ett hamnanlöp för stora fartyg. Grundkonceptet är att aktörer kommunicerar med varandra genom standardiserade *PortCallMessages* (PCM) för att antingen representera ett

¹ [Ken Schwaber, Jeff Sutherland. "The Scrum Guide"](#) (PDF). Scrum.org.

² <https://www.scrum.org/resources/what-is-a-sprint-review>

skepp i rörelse, så kallade Location States, eller resultat av en process, så kallade Service States.³

Vår applicering av Scrum

Roller, ansvarsområden, och processer

Roller och ansvarsområde

Inom gruppen var roller löst definierade direkt från början, något som dels var i linje med Scrum och dels ökade den s.k. bussfaktorn⁴, det antal personer som kan vara frånvarande i sjukdom, oförväntade händelser, eller bli uppsagda utan att teamet saknar grundläggande kompetens inom något område. Det fungerade bra med denna struktur. När projektet kom igång blev det tydligt att några andra grupper skapat tydligare roller och exempelvis separerat frontend/backend. Vi valde att ej gå den vägen, men allt eftersom blev olika gruppmedlemmar lite nischade inom varsitt område ändå. Överlag kunde ansvarsområden (lokalbokning, uppdatering av kalender, enhetstestning) ibland falla mellan stolarna. En lösning på detta hade nog inte varit en tydligare rollfördelning utan snarare tydligare fördelning av ansvarsområden och mer specifika definitions of done.

För att säkerställa kontinuitet vid Scrum of Scrums-mötena var samma person scrum master under hela projektets gång. Detta ansvar kunde ha roterats inom gruppen men det identifierades inget behov för det under tiden. Inte heller i efterhand syns någon tydlig förbättringspotential. Christian från 8 Dudes in a Garage påpekade att en "scrumfascist"⁵ behövs för att kunna lyckas med scrum, något vi i gruppen inte hade, men som vi samtidigt inte kände ett behov av. Gruppen hade en gemensam vision kring scrum och processen följdes tillräckligt väl för att undvika behov av striktare ledarskap. Ev. experiment kring detta hade varit spännande att utföra ifall denna vision skulle hamna ur fokus.

Majoriteten av det gruppkontrakt (Social Contract D1) som skrevs följdes, men själva dokumentet användes aldrig under projektets gång eftersom det inte var nödvändigt, då vi upplevde att gruppen drog åt samma håll ändå. Vad som saknades var eventuellt bättre återkoppling kring de mål som sattes upp i gruppkontraktet, och hur säkra teamet kände på att uppfylla dem. Mötesprocessen gick annorlunda till än som beskrivet. Det parallella kandidatarbetet gjorde mötesbokningen mer komplicerad och spontan. Detta var inget problem och synkningen blev bättre av den ökade flexibiliteten. Förseningsfika borde ha ätits oftare då en viss gruppmedlem, inga namn nämnda, hade problem med klockan under hela projektet. Genom att uppdatera gruppkontraktet löpande under projektets gång hade gruppen kunnat följa det på ett bättre sätt (sätt upp regler du följer, och följ reglerna du sätter

³ Niklas Mellgegård, PortCDM API Introduction.

<https://github.com/hburden/DAT255/blob/master/Slides/L6-ProjectSystem.pdf>

⁴ The Bus Factor - Why your 'best developer' is your biggest problem

<http://5whys.com/blog/the-bus-factor-why-your-best-developer-is-your-biggest-probl.html>

⁵ Christian Frithiof, Startup Life. <https://github.com/hburden/DAT255/blob/master/Slides/L9-IDGB.pdf>

upp), men i och med att inget problem uppstod finns risk för att detta bara hade varit ytterligare byråkrati att hantera.

Tidsfördelning




Tidsfördelningen inom gruppen blev högst jämn, till stor del tack vare att gruppen ofta befann sig i samma rum. På så sätt arbetade de som var närvarande under lika lång tid och parallellt. Detta fungerade riktigt bra, men när gruppen inte kunde träffas blev distributionen mer skev då tid spenderades på projektet under olika tider på dygnet och olika dagar i veckan. Ett sätt att strukturera upp detta hade varit att tydligare fördela tiden, men denna lösning kan bli överflödigt i de allra flesta fall, och addera mycket overhead. Under det här projektet uppstod inte problemet för att det skulle bli värt att investera tid i, och vanligtvis används Scrum när alla gruppmedlemmar har samma arbetstider vilket också raderar behovet. Gruppen kände också ett förtroende för andra gruppmedlemmar, och att alla sig tog sitt ansvar, vilket minimerade behovet av att mäta och evaluera tidsfördelning.

Processer

Under projektet användes några vanliga verktyg inom Scrum och Extreme Programming (XP), inspirerade av föreläsningar och kursboken. Här beskrivs dess processer, vilket värde de gav oss, vad vi lärde oss av dem, samt vad vi skulle gjort om vi fortsatt projektet.

En **daily scrum** infördes där man varje morgon uppdaterade gruppen om vad man gjort dagen innan och vad man planerade att göra under dagen, även om dessa aktiviteter inte nödvändigtvis hade med kursen att göra. Detta gjordes i Slack under en dedikerad #standup-kanal. Varje teammedlem uppdaterade med vad dem gjorde igår, och vad dem ska göra idag. På så sätt var alla synkade.

Detta fungerade bra när det användes som flitigast, men stressen under sista veckan gjorde att uppdateringarna blev sämre. När man satt i samma rum kändes det inte som att processen behövdes, men för medlemmar som ej var närvarande var det viktigt att hålla alla uppdaterade. Daily scrums är ett bra exempel på vad som kunde inkluderats i gruppkontraktet om det hade hållits uppdaterat under projektets gång.

-  **aron** 🏖️ 10:57 AM
Igår: var på handledning. hjälpte grupp Titanic med utvecklingsmiljö. felsökte massa kod 😊
Idag: refaktorerat. håller på med att få fram timestamps ur portcalls
-  **teomar** 🤔 10:58 AM
Igår: Krigade med maven för att kunna importera external libraries.
Idag: Har besegrat maven-monstret. Håller på att refaktorerar javaFX-kod för att förenkla tillägg.
-  **quaxi** 🖥️ 11:26 AM
Igår: Strulade lite VirtualBoxen, men hade inte tid att göra mycket produktivt
Idag: Ska fortsätta jobba i eftermiddag på messageparsing
-  **ant** 12:02 PM
Igår: Solade med Håkan Burden på handledningen. Felsökning
Idag: Försök till att skapa en första overview av ett port call (edited)
-  **oskar** 2:40 PM
Igår: jobbade med javafx, fixade scrollbarat fönster och refaktorerade kod i omgångar
Idag: jobbat kandidat, ingen kodning

Exempel på daglig asynkron daily scrum i Slack

Parprogrammering användes löpande vid behov. I och med att gruppen ofta satt i samma rum var det lätt att vända sig till grannen när ett problem uppstod och lösa det tillsammans, för att därefter återgå till separat kodande naturligt. Lösningen fungerade utmärkt och var en av de faktorerna som bidrog mest gruppens produktivitet. Det var dock inte inplanerat på något sätt utan bara något som föll naturligt, och ifall arbetet inte hade skett i samma lokal till lika stor utsträckning hade det behövts en mer uppstyrd process. Det hjälpte oss att lättare identifiera buggar, hitta bättre lösningar, och mer kvalitativ kod som fler personer förstod i teamet. Nackdelen var att det fungerade sämre med teammedlemmar på distans. Hangouts gick att använda, men du får inte samma kommunikation som om du sitter bredvid varandra.

Separata möten för separata ändamål var något som Kniberg identifierar i kurslitteraturen⁶. Gruppen drog tydliga gränser mellan olika typer av möten (Review, Retro och Sprint Planning). När det var möjligt genomfördes retron direkt efter sprint review, och nästa sprint planerades först när gruppen kunde ses igen nästa gång. Detta möjliggjorde för en paus mellan sprintar och fungerade bra. Sammantaget skapade detta tillvägagångssätt tydliga tillfällen för respektive viktig del inom Scrum utan att blanda ihop dessa. Något som kan förbättras är möteslängderna som skulle kunna specificeras tidigt för att undvika överplanering och att dessa tar för mycket tid från resten av sprinten.

Kontinuerlig integration och testning blev ett naturligt tillvägagångssätt för projektet. Det var viktigt för gruppen att alla hade den senaste versionen av kodbasen och därför pushades uppdateringar till repot så ofta som möjligt. Vi var noga med att enbart pusha fungerade kod, och alla pull requests var tvungna att vara recenserade av en annan teammedlem för att godkännas. För varje pull request så genomfördes manuell regressionstestning för att se om något hade gått sönder.

⁶ Kniberg, H. (2015) Scrum and XP from the Trenches - 2nd Edition

Den snabba förändringstakten medförde dock att dokumentation och testning inte alltid hängde med. Vi ville ha ett strikt krav att alla tester ska gå igenom för varje pull request, exempelvis med en `PULL_REQUEST_TEMPLATE.md` i git-repot som skapar en checklist för teststeg att göra, eller att testerna blev en del av varje korts Definition of Done.

Det blev dock snabbt märkbart att det här adderade mycket overhead när vi kontinuerligt gjorde drastiska förändringar och refaktoreringar i kodbasen. Både för att enhetstesterna behövdes skrivas om, men även för att våra use case ändrades drastiskt, så att till exempel en checklist för use cases att testa i en manuell regressionstestning hade fått uppdateras väldigt ofta.

Enhetstestningen hade som mål att täcka hela kodbasen, och i slutskedet av projektet täcker testerna alla metoder i modellklasserna i vår MVC-struktur. JavaFX-ramverket tillsammans med SceneBuilder som används gör det lite obekvämare att införa tester i resterande klasser och att lösa detta hade varit ett nästa steg om projektet hade fortlöpt. Under tiden har view- och controllerklasserna testats manuellt genom regressionstester.

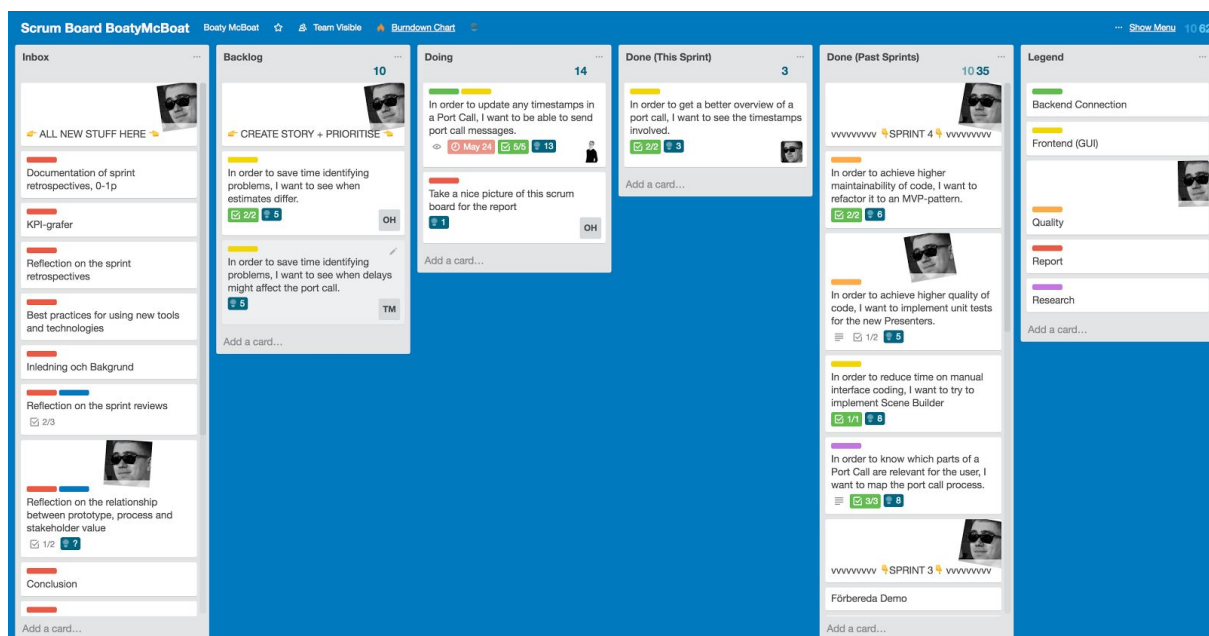
Ytterligare förbättringssteg för kontinuerlig integration hade varit automatiska enhetstester för varje pull request, samt testautomation som testar direkt mot GUI:t och/eller mot API:erna. Detta hade dock endast varit värt att implementera när vår kodbas hade stabiliserats mycket mer.

En **hållbar arbetstakt** var visionen under hela projektets gång. Gruppen jobbade inte ihjäl sig när det såg ut som att vissa user stories inte skulle hinnas med utan dessa flyttades över till sprinten därpå. I och med att den hållbara arbetstakten var tydligt definierad inom gruppen blev förväntningarna samma för alla medlemmar och stressen kunde hållas låg. Undantaget blev den sista sprinten där slutpresentationen gjorde att gruppens stressnivå stack iväg. Scrum gav intrycket av att vara arbetssättet där denna slutstress blir som minst, vilket var väldigt uppskattat.

Sprint Planning

The situation as it is - Hur vi gjorde Sprint Planning

Målet med varje Sprint planning var att identifiera, estimerar, och prioritera uppgifterna som teamet skulle göra under nästkommande sprint. Innan varje sprint så planerades arbetet med hjälp av hjälp av en **Scrum board** i verktyget Trello. Under de tre första sprinterna användes en ny board för varje sprint, men från fjärde sprinten och framåt valdes att gå över till en gemensam Scrum board som uppdaterades under varje planeringsmöte. Ingen särskilt stor förbättring märktes i produktiviteten men överskådligheten blev bättre och denna gemensamma Scrum board användes projektet ut.



Screenshot av vår Scrum Board i Trello

För boarden användes fem kolumner, en Inbox-kolumn där alla nya tasks alltid kunde läggas in, en backlog för prioriterade tasks denna sprint. En Doing-kolumn som representerar vad medlemmarna jobbar på just nu. En Done (This Sprint)-kolumn som representerar vad teamet har gjort hittills denna sprinten, och en Done (Past Sprints)-kolumn som representerar vad teamet har gjort under tidigare sprints. En legend-kolumn finns även för att göra det tydligare vad färgkategoriseringen innebär.

För varje planering togs tasks från inboxen och lades i backloggen där den skrevs om enligt "In order to <receive benefit>, I want <goal/desire>", enligt en variation av Chris Matts definition av en user story.⁷ Vi valde den eftersom att vi kände att rollen inte gav oss så mycket värde, då den nästan alltid för oss var "Ship Agent 2", och Chris Matts format gjorde det mer tydligt vad det faktiska värdet var av att implementera denna storien. Gruppen var

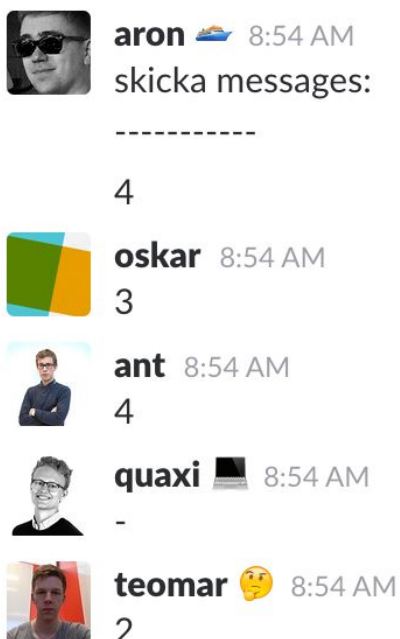
⁷ http://antonymarcano.com/blog/2011/03/fi_stories/

duktig på att skapa vertikala och kundfokuserade tasks efter den mycket givande elefantövningen. Det gick överlag bra att arbeta med ett vertikalt tankesätt.

För att lättare kunna estimerar user stories och avgöra om de är avklarade användes **Definitions of done (DoD)** redan från tidigt skede i projektet, men vi blev striktare med upplägget senare i projektet. Allt eftersom en större vana vid systemet och projektet uppnåddes blev dessa definitioner lättare att identifiera och fick en bättre struktur. Konkret implementerades det så att alla kort i Trello hade en checklista som hette DoD med alla steg som behövdes genomföras innan kortet skulle kunna röras till *“Done (This Sprint)”*-kolumnen. Det blev ett bra sätt att se hur långt man kommit på respektive task, men även i slutet av projektet var det ibland svårt att definiera konkreta DoDs. Ett par veckor till med systemet hade antagligen medfört en ytterligare förbättring på denna front.

Ytterligare relaterat till estimering användes **planning poker** för att uppskatta storleken på user stories. Då alla medlemmarna inte alltid var fysiskt närvarande så sköttes detta i en dedikerad slack-kanal, medan medlemmarna kommunicerade över Google Hangouts. En teammedlem räknade ner från 3, och sedan skrev alla sina estimat samtidigt. Detta för att inte påverka varandras beslut. Var det stora skillnader fick de med lägst och högst resultat förklara varför de valt just så, och sedan vid behov om-estimerades storyn.

Under de två första veckorna var uppskattningarna helt off jämfört med verkligheten på grund av de underliggande tekniska svårigheterna, men mot slutet fick vi ett mycket mer förutsägbart arbetsflöde efter att vi lärde oss verktygen och ramverken, samt kom över mycket setup-problem. Detta hängde även ihop med att stories definierades på ett tydligare sätt med hjälp av bl.a bättre definitions of done. Planning poker användes enbart för att estimerar user stories som tänktes ingå i sprinten.



aron 🚤 8:54 AM
skicka messages:

4

oskar 8:54 AM
3

ant 8:54 AM
4

quaxi 💻 8:54 AM
-

teomar 🤔 8:54 AM
2

Exempel på Planning Poker genomförd i en Slack-kanal

What you would like it to be - Hur skulle vi gjort Sprint Planning?

Grunden som lades av ovanstående process känns stabil men är inte utan förbättringspotential. Nedan följer några av de områden som identifierats som ytterligare kan förbättras.

Involvera produktägare och övriga stakeholders mer i estimeringen

Trots förbättringar i actual velocity vs estimerad velocity under projektets gång kan estimering bli ett mycket mer kraftfullt verktyg. En strukturell förändring som naturligt hade förbättrat processen är att involvera produktägaren i sprintplaneringen. I detta projekt var inte denna förändring aktuell (på grund av den begränsade tiden gruppen hade med produktägaren) men rekommenderas starkt av ex. Henrik Kniberg⁸. I så fall hade en preliminär estimering för hela/delar av backloggen gjorts innan planeringsmötet, där stories därefter väljs ut för att ingå in sprinten. På så sätt får estimeringen ett tydligt syfte: bistå med att sätta upp rimliga sprintmål. Även sättet som planning poker används kan förbättras: en vanlig fråga under estimeringsprocessen var "hur mycket är en 5a?". Referensstories som agerar exempel på olika stora stories kan hjälpa till här.

Experimentera med andra sätt att sätta velocity

Gruppen valde ovan nämnda actual velocity och estimerad velocity som en KPI för projektet, något som kan läsas mer om senare i denna rapport. Projektets korta horisont och fluktuationen i mängden tid som andra åtaganden (kandidatarbetet) kunde ta varje vecka gjorde det dock svårt att använda velocity som ett planeringsverktyg. Ett sätt att göra detta på hade varit att under varje planeringsmöte bestämma velocity för innevarande vecka, relativt föregående veckas velocity, efter att produktbackloggen estimerats. Därefter hade sprintbackloggen kunnat fyllas med gott samvete. Ett liknande experiment genomfördes inför den tredje sprinten där velocity sänktes tidigt i planeringsprocessen och laget hann med allt som planerats inför veckan. Därför finns bevis på att ytterligare försök kan ge effekt.

Vara striktare med att bryta ner tasks i tunnare bitar

Gruppen hade som ovan nämnt lätt för att skapa vertikala och kundfokuserade stories. Dessa blev följdaktligen dock rätt stora, och ytterligare smalare tasks hade underlättat för parallellisering av utvecklingen, testning och code review. Att enbart nämna "gör skivorna tunnare" som förbättringspunkt visade sig inte ha särskilt stor effekt. Förslagsvis kunde det ytterligare experimenterats med riktlinjer för tasks, så som maximal tidsåtgång för dessa, rader kod eller liknande. På så sätt får gruppen en tydlig referenspunkt när det kommer till tasks.

⁸ Kniberg, H. (2015) Scrum and XP from the Trenches - 2nd Edition

Sprint Retrospectives

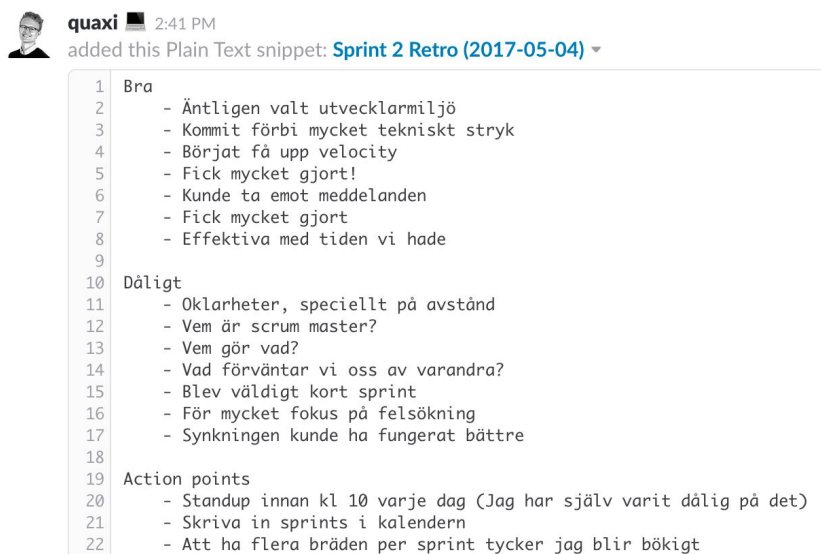
The situation as it is - Hur vi gjorde retros

Sprint retrospectives kördes enligt formen:

1. Vad har gått bra under förra sprinten?
2. Vad har gått dåligt under förra sprinten?
3. Från detta - vad vill vi ändra och förbättra till nästa sprint?

Målet med frågorna var att på ett effektivt sätt kunna identifiera saker att förbättra tills nästa möte, i mån av att kontinuerligt förbättra inte bara vår produkt, utan också våra processer. Både för att behålla processer som fungerar bra, eller att förändra eller att sluta bedriva processer som fungerar dåligt.

Varje teammedlem fick nämna vad de kände efter sprinten på punkt 1 och punkt 2, och efter att alla närvarande medlemmar fått säga sitt, så gjorde vi likadant på punkt 3. Resultatet dokumenterades under en Slack-kanal som vi döpte till #Retro. Dokumentationen var inte avsedd att vara djupare än snabba anteckningar som vi kunde referera till, för att minimera tiden det tar att göra en retro. I början dokumenterades varjes persons input, men då vi ville att detta ska reflektera teamet och inte individer så buntade vi ihop dokumentationen för hela teamet istället. Generellt fungerade Retros bra för att hitta idéer kring hur vi bäst strukturerar upp vårt arbete, och att identifiera problem. De var samtidigt snabba och effektiva att genomföra.



Exempel på anteckningar från en Retro

Våra retros hjälpte oss till exempel att identifiera värdet av att jobba i samma rum, värdet av att alla hade samma utvecklingsmiljö (IntelliJ), och värdet av god kommunikation.

What you would like it to be - Hur skulle vi gjort retros?

I mångt och mycke fyllde Retros sitt syfte, i att det hjälpte oss att identifiera problem och göra kontinuerliga förbättringar på det. Det tillhandahåller också en möjlighet för teamet att diskutera processer och ways-of-working som normala arbetsförhållanden sällan ger utrymme för.

Även om grundkonceptet för retros är ganska likt i de olika tappningar som finns, att kontinuerligt förbättras som team, så finns det olika versioner av frågorna som ställs. En vanlig variant är att istället fråga vad teamet ska...

1. Börja göra
2. Sluta göra
3. Fortsätta göra

Detta sättet ger en mer direkt relation till konkreta punkter kring vad som behövs förbättras, men vi kände att det inte passade oss på grund av flera anledningar.

Dels är det svårt att från vår situation göra evalueringar på processer. Till exempel fanns det i vårt team ofta tekniskt strul kring utvecklingsmiljöer, backend-anslutningar och dylikt. Att säga "sluta göra" eller "börja göra" något från den punkten är svårt som oerfaren utvecklare. Att däremot att kontinuerligt identifiera tekniskt strul som en pain point i processen genom att benämna det som något "dåligt" kan belysa att det kan behövas investera tid och resurser att fixa tech debt.

Ett problem med den nuvarande processen är att den också inte tog hela gruppens mål i beaktning. I gruppens sociala kontrakt benämns att hela gruppen vill ha en femma, lära sig om Scrum, Git, Kata, och projektledning, samt att ha kul. Retrospectives fyllde sitt mål med att utvärdera teamets processer för att leverera värde till kund, men missade dessa målen.

Detta skulle kunna exempelvis genomföras likt hur Spotify varannan vecka håller en "OKR Demo"⁹, där varje team får berätta om det nuvarande läget kring mål de satte upp för kvartalet, samt hur högt förtroende de har för att kunna uppfylla målet inom utsatt tid. Likväl skulle detta kunnas appliceras för vår grupp, men med målen som presenterades i det sociala kontraktet.

⁹ Gästföreläsning Spotify, Simon Hofverberg, 2017-04-24

Sprint Reviews

The situation as it is - Hur vi gjorde reviews

Sprint reviews gjordes tillsammans med PortCDM, vår primära stakeholder, och var av mycket stort värde för oss. Dels när det gällde att kartlägga processen som ett hamnanlöp innebär, och dels för att det hjälpte oss att skapa en bild av vad vår klient (agent 2) efterfrågade. Det som vi ganska snabbt insåg var att införandet av PortCDM skulle eliminera en ganska stor del av agentens arbetsuppgifter som i dagsläget handlar om samordningsarbete genom att sköta koordinering och kommunikation mellan olika aktörer vid ett hamnanlöp. Därmed kom vi sedan fram till att vår produkt skulle kunna visualisera ett hamnanlöp och olika problem som uppstår om estimerade tider från olika aktörer skiljer sig åt. På så skulle vår produkt kunna underlätta agentens arbete och därmed leverera värde till kunden.



Screenshot från "Overview"-vyn i vår applikation

För att kunna ge en bild av hur väl vi kommunicerade vår vision med produktägaren samt för att de skulle kunna sätta ett betyg på vårt inkrement av lösningen för en viss sprint införde vi ett produktägar-KPI. Detta KPI var ett enkelt betyg på 0-10 poäng som produktägaren fick sätta i slutet av varje handledningstillfälle. Produktägar-KPI:t innebar en visualisering för oss som grupp som visade hur väl vi kommunicerade vår vision och hur komplett vårt inkrement ansågs vara. Detta gjorde att vi fick en kombinerad "morot och piska" som gjorde oss motiverade att hela tiden försöka förbättra vår produkt och sättet vi levererade värde på till produktägaren. KPI:t gjorde även att vi efter ett par sprintar kunde följa vår utveckling över produktägarens uppfattning om vår produkt, vilket innebar en slags validering för hur väl vår produkt mottogs av de som produkten i slutändan skulle levereras till.

Som tabellen indikerar fick vi inget betyg av produktägaren efter första sprinten, detta på grund av att vi då saknade ett inkrement med värdefull funktionalitet att visa upp för produktägaren. När vår produkt sedan utvecklades över sprintarna ökade vårt betyg, vilket vi anser visar på att vi lyckades uppfatta produktägaren främsta behov och därefter leverera värde till dem. Den sista sprint reviewen, som vi tolkar som den faktiska gemensamma presentationen för PortCDM, fick vi inget betyg av produktägaren då detta tillfälle mer var dedikerat till den faktiska presentationen. Dock pratade vi med många olika representanter för PortCDM och olika aktörer från hamnen, som alla uttryckte positiva åsikter kring vår produkt. Mer specifikt nämnde de den vy som är tänkt att ge fartygsagenten en snabb

överblick över statusen för ett anlop, där eventuella varningar kan ges om problem finns med till exempel estimat som inte stämmer överens. Med dessa positiva reaktioner i åtanke känner vi att vårt produktägar-KPI i minsta fall bör hamna på samma nivå som föregående vecka. Vi tog ett medvetet val i början av projektet att inte använda burndown charts. Vissa teammedlemmar har använt det tidigare, och hade erfarenheten av att det skapar mer overhead än vad det skapar värde. Henrik Kniberg, författaren till delar av kurslitteraturen, nämner även att det inte har fungerat väl när han har testat att implementera det på Spotify.¹⁰ Jimmy Janlén från samma konsultbyrå nämner även att han hittat ytterst få fall där det fungerat väl.¹¹ Det fanns heller inget krav från kurshandledare att vi skulle använda Burndown charts.

What you would like it to be - Hur skulle vi gjort reviews?

Bättre struktur och tydligare förväntningar för mötena med produktägarna

I och med KPI:t fick vi en uppfattning kring hur väl vår produkt mottogs och hur vi kommunicerade vår vision med den. Dock så finns det en del saker som hade kunnat förbättras gällande just sprint reviews. Tillfällena då vi träffade produktägarna var ganska ostrukturerade och vi kände inte att vi fick så nära koppling till produktägarna som vi borde ha haft. Det saknades även en uppfattning kring hur vi skulle kunna ha förbättrat våra sprint reviews. Dessa problem hade kunnat lösas om vi körde med en striktare mall och agenda för sprint reviews som följdes vid varje tillfälle. Att följa en sådan struktur hade gjort att vi hade kunnat få ur mer koncentrerad information ur produktägarna, dels om vår produkt men även om just vår prestation på handledningstillfällena så. På detta sätt hade vi lättare kunnat reflektera över hur varje handledningstillfälle gick och hur vi skulle kunna förbättra det till nästa gång.

Skippa burndown charts, men var tydliga och transparenta mot stakeholders

Ett alternativ till Burndown charts som nämns i Jimmys blogg¹² är att använda "Confidence Smileys". För att öka transparens mot våra stakeholders använder vi då glada, nervösa, eller ledsna smileys för att indikera hur säkra teamet är att kunna genomföra en story under denna sprint, men då teamet möttes endast med stakeholders efter varje sprint kände vi att vi likväl kunde ha lika hög transparens genom att hålla en bra dialog under mötena. Jimmy nämner även att en stor fördel med detta är att det skapar diskussion och frågor kring stories, men med dagliga standups och korta sprint-perioder kände vi att det löste sig naturligt.

Fortsätta utveckla mot en gemensam vision

Vår vision med produkten vi utvecklade var som sagt främst att den skulle vara ett visuellt verktyg för att underlätta agentens dagliga arbete, med möjlighet att kunna skicka portcalls. Vi anser att vi uppnådde en del av den visionen då vårt befintliga produktinkrement klarar av att varna om flera aktörer har givit estimat som skiljer sig åt för en viss location eller service. Vi

¹⁰ Spotify Engineering Culture, Part 1. <https://vimeo.com/85490944>

¹¹ The Sprint Burndown is dead, long live Confidence Smileys, Jimmy Janlén. <https://blog.crisp.se/2015/04/01/jimmyjanlen/the-sprint-burndown-is-dead-long-live-confidence-smileys>

¹² The Sprint Burndown is dead, long live Confidence Smileys, Jimmy Janlén. <https://blog.crisp.se/2015/04/01/jimmyjanlen/the-sprint-burndown-is-dead-long-live-confidence-smileys>

har kompletterat denna funktion med en vy där mer exakt information kring alla location-states och service-states kan hämtas och avläsas, så att agenten kan ta mer informerade beslut kring de situationer där vår produkt varnar för avvikelser. Genom diskussioner med produktägarna under sprint reviews kom vi fram till att vi i framtiden hade velat utveckla produkten så att varningar i översiktsvyn skulle inkludera förseningar som kan fortplantas i kedjan av händelser i ett anlöp samt problem som kan uppstå vid krockar med tider för andra anlöp. En annan sak som vi skulle vilja göra med produkten är att den detaljerade vyn tas bort och att den detaljerade informationen ska kunna komma åt från översiktsvyn genom ett klick på den händelse man vill ha mer information om. Till sist skulle vi vilja utveckla översiktsvyn så att den blir dynamisk. Detta då varje anlöp är unikt, vilket gör det onödigt att visa händelser som inte är aktuella för ett specifikt anlöp.

Verktyg och teknologier

Best practices för att anamma nya verktyg i ett mjukvaruprojekt

Våra verktyg för mjukvaruutveckling

Användningsområde	Verktyg	Potentiella alternativ
Språk	Java	<i>JavaScript, Kotlin, Swift...</i>
IDE	IntelliJ	<i>Eclipse, NetBeans</i>
Versionshantering	Git + Github.com	<i>Perforce</i>
Enhetstestning	JUnit	<i>TestNG</i>
Byggsystem	Maven	<i>N/A</i>

Vi bestämde oss för att lösa uppgiften genom att bygga en **Java**-applikation då alla i teamet kände till språket sedan innan. *JavaFX*-biblioteket användes för att måla upp GUI:t. *Scene Builder* användes då detta är ett verktyg som medför en visualisering av hur GUI:t kommer att se ut. I *Scene Builder* finns även en drag-and-drop-funktion som möjliggör byggandet av ett GUI på ett mer överskådligt sätt. Dock krånglade ofta denna drag-and-drop-funktion, vilket gjorde att en ganska stor del av fxml-koden fick skrivas "manuellt".

Vilken Java-IDE som användes var dock delat i teamet. Vissa föredrog *Eclipse*, mellan andra föredrog *IntelliJ*. Vi märkte dock tidigt i projektet att ha flera IDE:er i samma team introducerade mycket fler problem än vad personliga preferenser var värt det. Mycket av vår tid spenderas på teknisk support, vilket dubblerades med dubbelt antal plattformar. Samtidigt gjorde vi bedömningen att *Eclipse* fungerade mycket sämre för vårt use case, och hela teamet anammade **IntelliJ** istället.

För revisionshantering kändes det självklart att använda **Git** då det förutom i vissa fall är en de facto branschstandard. Dessutom är det det enda versionshanteringssystem som medlemmarna hade tidigare erfarenhet av. Redan innan vi skrivit en enda kodrad bestämde vi oss för att anamma en Git-modell där varje medlem har sin egen fork av Git-repot, och enda sättet för kod att hamna i produktion är att skapa en Pull Request som en annan teammedlem måste granska och godkänna. Det fungerade väldigt bra, och jämfört med andra mjukvaruutvecklingsprojekt som deltagarna var med i så hade vi väldigt få stora merge conflicts eller andra liknande problem.

För enhetstestning användes **JUnit**, då en medlem hade erfarenhet av det innan, plus att det är det vanligaste testramverket för Java¹³. För byggsystem så var **Maven** helt enkelt det som hade bäst stöd för PortCDM, så det var det självklara valet.

¹³ <http://blog.takipi.com/junit-vs-testng-which-testing-framework-should-you-choose/>

Våra verktyg för projektledning

Användningsområde	Verktyg	Potentiella alternativ
Visualisera arbete	Trello	<i>Favro, Jira</i>
Kommunikation	Slack + Google Hangouts	<i>Facebook, HipChat</i>
Schemaläggning	Google Calendar	<i>iCal, Outlook</i>

Vi använde oss av **Trello** för att hantera user stories, då de flesta i teamet hade erfarenhet av det innan, och det uppfyllde våra behov. Alternativ hade varit *Jira*, som ofta används i större produktionsmiljöer, eller *Favro*, som främst har fått fäste i spelutvecklingsbranschen. Vi kände inget behov av Jiras utökade funktionalitet, och Favro erbjöd inte något ytterligare i relation till kostnaden av att lära sig ett nytt verktyg.

Initialt användes mycket *Facebook Messenger* för kommunikation i gruppen, men även om det erbjöd en enkel integration med ett verktyg som alla redan använder, så blev det snabbt bökigt att ha all kommunikation i en och samma kanal, utan bra stöd för tredjeparts-integrationer likt Google Docs eller Trello. **Slack** kändes som det naturliga valet då förutom tidigare erfarenhet så användes det redan användes för diskussion med lärare och handledare. Facebook Messenger användes dock fortfarande i teamet ifall man snabbt behövde få tag på gruppmedlemmar.

I början hade inte teamet någon form av kalender eller schemaläggning alls. Då teamet hade en medlem på distans i Stockholm blev det ibland rörigt när möten var, var man skulle infinna sig, och när Sprints började och slutade. Detta ledde till att teamet började använda **Google Calendar** för att tydligare synka kring möten, deadlines, och Sprints.

Vad hade vi gjort annorlunda om vi börjat om idag?

Fortsätt använda väl använda verktyg och verktyg som teamet har tidigare erfarenheter av
Generellt har vi angripit vilka verktyg vi använder från ett perspektiv av:

1. Kan vi använda verktyget för att uppfylla de grundläggande behoven vi har?
2. Har teamet någon tidigare erfarenhet av verktyg som löser detta problemet?
3. Av de identifierade alternativen, är något branschstandard?
4. Av de identifierade alternativen, vilket är vanligast?

Även om prioritet ett självklart är att verktygen uppfyller våra behov, så är det tydligt med att det endast menas de absolut grundläggande behoven, inte specifika features som vi tror skulle vara ytterligare hjälpsamma eller annars bra att ha. Det ska snarare bara ses som en hygienfaktor. Till exempel är skillnaden mot att använda Trello istället för Favro i benämning

vilka features de har inte så stor, och Favro kanske till och med är bättre för att använda story points eller bättre på annan feature, men teamet har ingen erfarenhet av det innan.

Vi väljer att använda vanliga verktyg eftersom att av erfarenhet vet vi sedan innan att mycket av vår tid kommer gå åt för problemlösning, och att söka på sidor likt Stackoverflow för lösningar och anfallsvinklar. Då tiden är begränsad fram till vi ska leverera slutprojekt är denna support-resurs otroligt värdefull, och något vi prioriterar högt i de verktyg vi använder. Detta är ännu mer viktigt för högst tekniska verktyg, likt vilken IDE, språk, och bibliotek man använder, då tiden spenderat på felsökning där är betydligt högre än mer enkla och användarfokuserade verktyg.

Den här approachen fungerade väldigt bra för oss, och med ett par undantag likt Eclipse valt exakt samma verktyg en annan gång. Hade vi fortsatt hade vi nog skrivit om stora delar av front-enden i JavaScript. Teamet hade mer erfarenhet i Java, men att skriva ett GUI i JavaScript ligger mer i linje i branschstandard, samt gör det trivialt att en webbapplikation av programmet, vilket skulle göra att våra stakeholders slapp ladda ner en applikation och manuellt uppdatera den.

Håll teamet på samma plattformar

Vad vi underskattade var hur mycket tekniskt strul som skulle uppstå, och att vi inte alltid tog rätt steg för att minimera det. Hade vi gjort det igen hade vi nog varit mer strikta på att alla använder samma plattformar i så hög mån som möjligt (Om någon kör OS X och en annan teammedlem kör Windows är det svårt att sälja in att någon ska byta.)

Håll teamet i synk

Även om vi hade tidigt etablerat Slack som ett verktyg var det inte förrän senare i projekt när vi började anamma våra Scrum-processer kring Slack som det blev ett användbart verktyg för att hålla teamet i synk. Initialt hade vi mycket problem med att synka Göteborg och Stockholm, och det underlättades markant av Slack, Google Hangouts, och att Google Calendar. Om vi hade gjort projektet igen hade vi nog tagit problemen med att vara ett team i två städer mer seriöst, och anpassat våra verktyg för det.

Reflektion av relationen mellan *prototyp*, *process*, och *stakeholder-värde*

Vi har under projektets gång försökt extrahera user stories från de samtal vi har haft med representanter från PortCDM och hamnen, det vill säga produktägaren, under handledningstillfällena. Handledningstillfällena, samt de user stories vi har fått med oss från dessa, har varit centrala i planeringsarbetet. Vi ser det som viktigt att alla designbeslut kopplade till prototypen är förankrade i våra user stories, för att vi ska kunna leverera så mycket kundvärde som möjligt med prototypen. Det är såklart meningslöst att skapa avancerad funktionalitet om den inte tillför något värde till produktägaren och är något som en eventuell kund skulle vara beredd att betala för.

Produktägaren presenterade ett antal user stories som ansågs grundläggande för en komplett applikation, det vill säga det produktägaren såg som en MVP för praktisk användning i hamnen. Vi insåg tidigt i projektet att alla dessa user stories inte skulle uppnås under projektets gång med den velocity vi hade och prioriterade därför bland user stories internt. Utifrån de möten vi hade med produktägaren drog vi slutsatsen att de viktigaste delarna av funktionaliteten för kundvärdet var att kunna generera ny data och att på ett enkelt sätt kunna filtrera och få en överblick över existerande data inom ett portcall.

Något som vi ser som bristfälligt i vår process är att vi saknade formella acceptanstester i kontakt med produktägaren, det vill säga ett återkommande förfarande där produktägaren avgör om funktionalitet som vi har utvecklats uppfyller deras krav, eller om den behöver förfinas ytterligare. Under möten med produktägaren ställde vi frågor kring hur nöjda de var med vad vi har åstadkommit och vi hade ett KPI kopplat till kundnöjdhet. Detta ger en viss uppfattning om ifall det vi levererat motsvarar förväntningar, men vi kan inte med säkerhet veta att någon funktionalitet vi levererat helt uppfyller produktägarens förväntningar eller önskemål. Detta är något som vi skulle vilja åtgärda vid en fortsättning av projektet eller start av ett nytt. Eftersom vi aldrig kunde vara säkra på om vår funktionalitet var tillräckligt bra hände det att vi gick tillbaka och filade på funktionalitet som levererats i tidigare sprintar när vi såg en förbättringspotential. Om vi hade haft acceptanstester hade vi kunnat lägga funktionalitet som klarat acceptanstest(erna) åt sidan och fokusera på nya user stories eller att komplettera de user stories där den levererade funktionaliteten inte klarat acceptanstesterna. Vi tror att en dialog med produktägaren kring acceptanstester i ett tidigt skede hade hjälpt oss genom att bättre fokusera våra insatser där de behövs.

Som ett resultat av att vi saknade formella acceptanstester mötte vi problem gällande Definition of Done (*DoD*), för våra user stories. Då vi saknade exakt kunskap om vad produktägaren förväntar sig innan de anser en viss user story färdig, mötte vi svårigheter med att internt bedöma om vi tyckte att en user story var färdig eller inte. Då vi inte efterfrågat specifika krav från produktägaren lyckades vi inte formulera *DoD* så att kriterierna var binära. För de flesta user stories i våra backlogs blev *DoD* något abstrakt och kunde besvaras med "delvis" eller "kanske". I praktiken så kopplades *DoD* till huruvida vi kände oss nöjda, snarare än om vi uppfyllit kundens krav, vilket motsätter sig syftet med att ha *DoD* överhuvudtaget. Vi tror återigen att rotproblemet är att vi inte hade en dialog med produktägaren kring

acceptanstester i ett tidigt skede för att garantera att vi levererar funktionalitet som uppfyller produktägarens krav utan att innehålla onödig gold plating.

Som tidigare nämnt har vi för att kunna leverera värde knutit prototypens utformning tätt till stakeholder value. Vi har sett processens primära syfte som att optimera hur vi fångar in stakeholder value i user stories och realiserar värdet i prototypen. Av denna anledning har vi låtit processen vara så flexibel som möjligt och vara öppna för att förändra i processen om vi tror att det innebär att vi kan leverera mer värde i prototypen. Denna inställning återspeglas till exempel i att vi under första veckan testade tre olika utvecklingsmiljöer innan vi fann en som vi tyckte passade våra behov. Under projektets gång solidifierades processen genom att vi fann metoder och verktyg som vi kände passade oss, vilket möjliggjorde för oss att uppnå en högre velocity senare i projektet.

Vi anser att denna approach fungerade bra för oss, då den funktionalitet som vi levererade uppfyllde de user stories vi prioriterat och även stakeholder value. Något som hade kunnat förbättras är effektiviteten, framförallt under projektets första fas. Under de första sprintarna levererades relativt lite kundvärde, vilket reflekteras i KPI:erna kundnöjdhet och velocity. Med projektets korta tidsram i åtanke tror vi att det hade varit möjligt att uppnå mer genom att hålla processen mer statisk i projektets första del. I projektet spenderade vi mycket tid på att bygga upp vår effektivitet och hade sedan relativt lite tid kvar att skörda frukterna av effektiviteten.

Evaluation of D1 - D4

D1: Strategies for scrum implementation and social contract

Strategier

I den första leverabeln togs ett antal strategier för Scrum-implementation fram med grund i de erfarenheter vi fick från Lego-övningen. De olika tillvägagångssätten som diskuterades kan sammanfattas i tre punkter: vikten av kommunikation inom och mellan grupper, förståelse för rotproblemet och att konsekvent använda scrum-arbetsättet.

Att ha en bra kommunikation i teamet men även gentemot andra grupper har verkligen känts som en nyckelfaktor. Inom gruppen har det övergripande fungerat bra vilket möjliggjort ansvarsfördelning. På sätt har en tydlig arbetsfördelning kunnat göras och onödigt extraarbete undvikts. Även mellan team har det fungerat hyfsat men mer synkande kring lösningar av generell problematik hade kunnat kommuniceras bättre. Vissa problem löstes vid Scrum of scrum-möten medan andra löstes i slutna kommunikationskanaler för att sedan behöva förklaras för ytterligare en part. För att bättra på detta samarbete grupper emellan hade en möjlighet kunnat vara att skapa en gemensam Wiki-sida med lösningar. En faktor till varför det förmodligen inte blev av var att ingen såg någon vinning i att spendera den dyrbara tiden på något som inte direkt tillförde värde till just deras grupp.

Vidare upplevdes det visserligen som viktigt att förstå grundproblemet likt under Lego-övningen. Det krävdes dock långt ifrån lika många varför-frågor för att nå rotproblematiken, kanske på grund av en initierad produktägare. Detta tillsammans med den goda möjligheten att få parterna inkluderade i projektet i samband med handledningstillfällen och mailkontakt har underlättat. På så sätt har ingen felaktig eller överflödig funktionalitet byggts, vilket ofta var fallet i övningen med de danska byggklossarna. Något som antagligen hade riskerat att inträffa om det inte funnits så goda möjligheter att bolla tankar och ideer.

Att arbeta agilt och att fullt ut implementera de rekommendationer som finns kring Scrum har inte varit fullt möjligt under detta projekt. Henrik Kniberg föreslår exempelvis i sin bok att en viloperiod på ett antal dagar bör finnas mellan två sprintar¹⁴. Något som gruppen även erfarade under projektets gång när den enda viloperioden, helgen, hamnade mitt i en sprint. På så sätt kunde man aldrig riktigt känna att man avslutat en sprint och sprintplanering tvingades därav ofta att sammanfalla med retrospektive. För övrigt fungerade det bra och den ständiga reflektionen som föreslogs i den första leverabeln upplevdes som hjälpsam. På så sätt möjliggjordes ständiga förändringar. Att undvika onödiga planering nämndes också som en viktig del vilket följdes ganska hårt under projektet. Planeringssessionerna hölls kanske inte lika korta som under lego-övningen men ambitionen var att snabbt få ihop nästa veckas user stories för att kunna påbörja arbetet. När vissa diskussioner gled iväg var det enkelt att inse att värdetillförseln minskade i samma takt.

Socialt kontrakt:

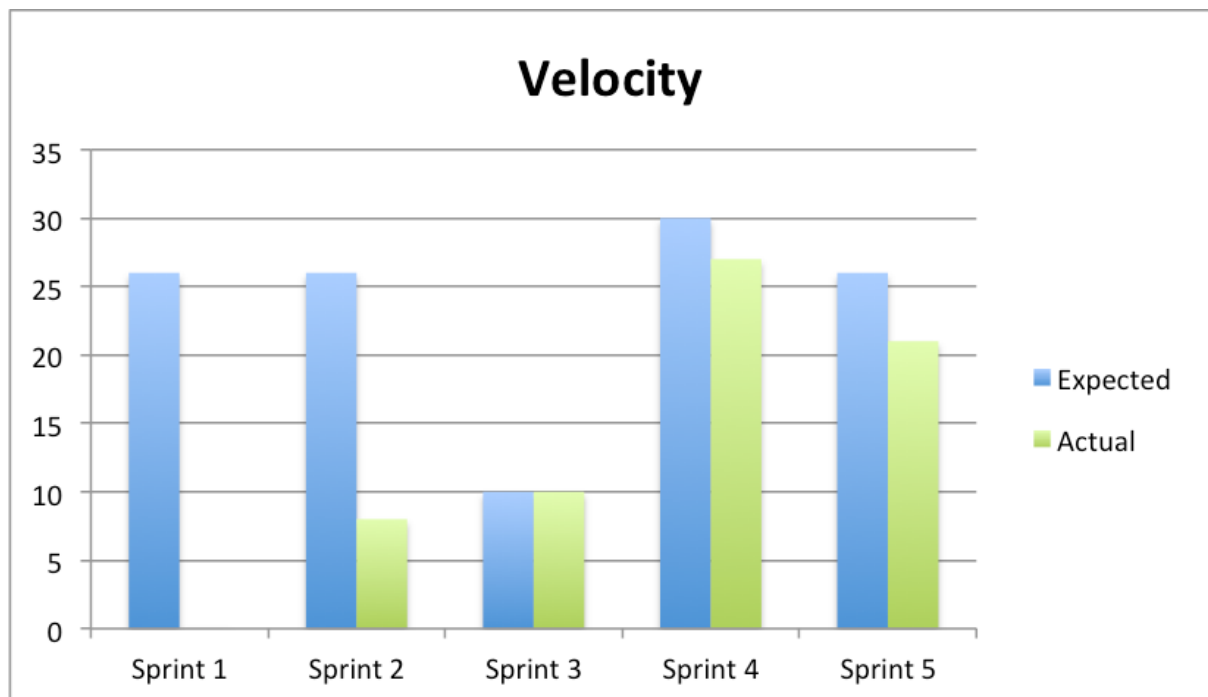
Det sociala kontraktet behandlas närmare i stycket som handlar om vår applicering av Scrum.

¹⁴ Kniberg, H. (2015) Scrum and XP from the Trenches - 2nd Edition

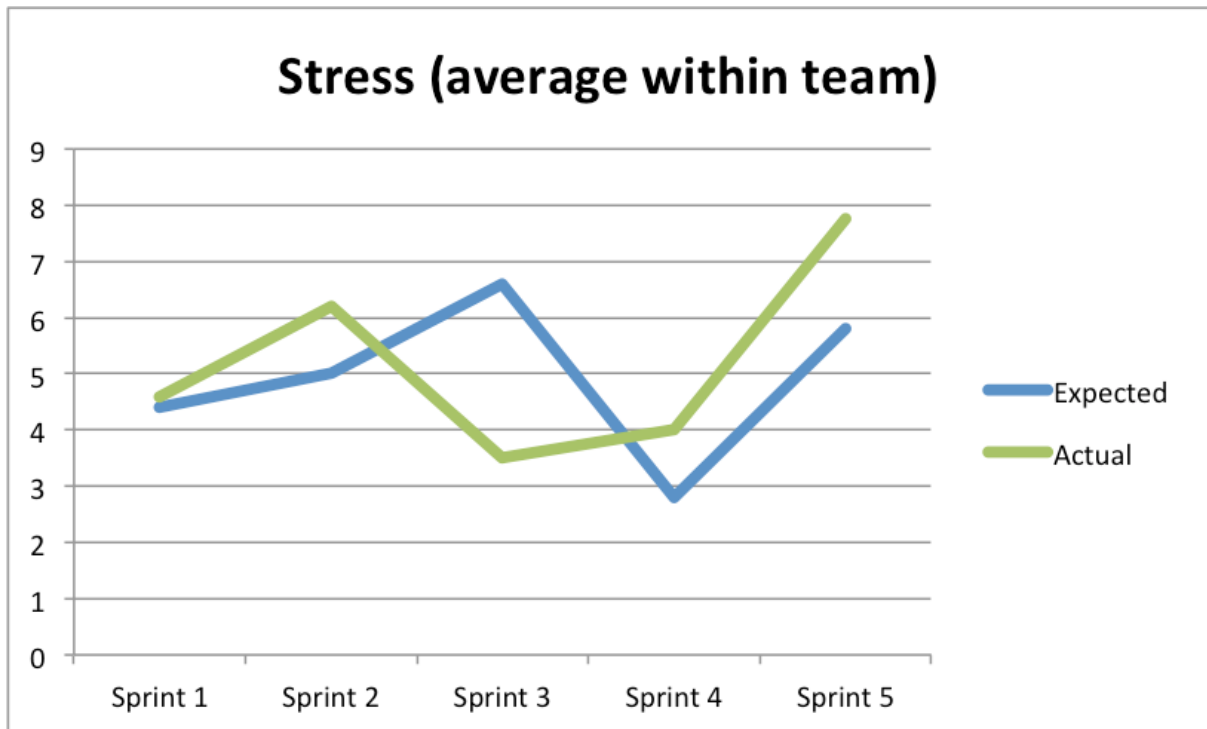
D2: Three KPIs to monitor the implementation of Scrum

Vi tog fram tre KPI:er med syftet att fånga de viktigaste aspekterna i Scrum-processen (se Bilaga för KPI-diagram). Tanken var alltså att KPI:erna tillsammans skulle ge en bra, övergripande bild över hur arbetet under projektets gång gick. De tre KPI:erna var Velocity, Upplevd stress inom gruppen samt Kommunikerbarhet mot produktägare. Det tredje KPI:t kom att utvecklas till att inkludera produktägarnas uppfattning kring vår produkts completeness när det var dags att implementera det, vilket gör att det KPI:t blev något bredare än vad syftet var från början. Med dessa KPI:er ansåg vi täcka in produktägarens behov, gruppens välmående och hur arbetet med produkten framskred. Stress inom gruppen sattes två gånger. Inför en sprint gav gruppen en siffra på hur stressigt vi trodde att det skulle bli under den kommande sprinten, och efter sprinten satte vi en siffra på hur stressigt vi faktiskt uppfattade arbetet under sprinten.

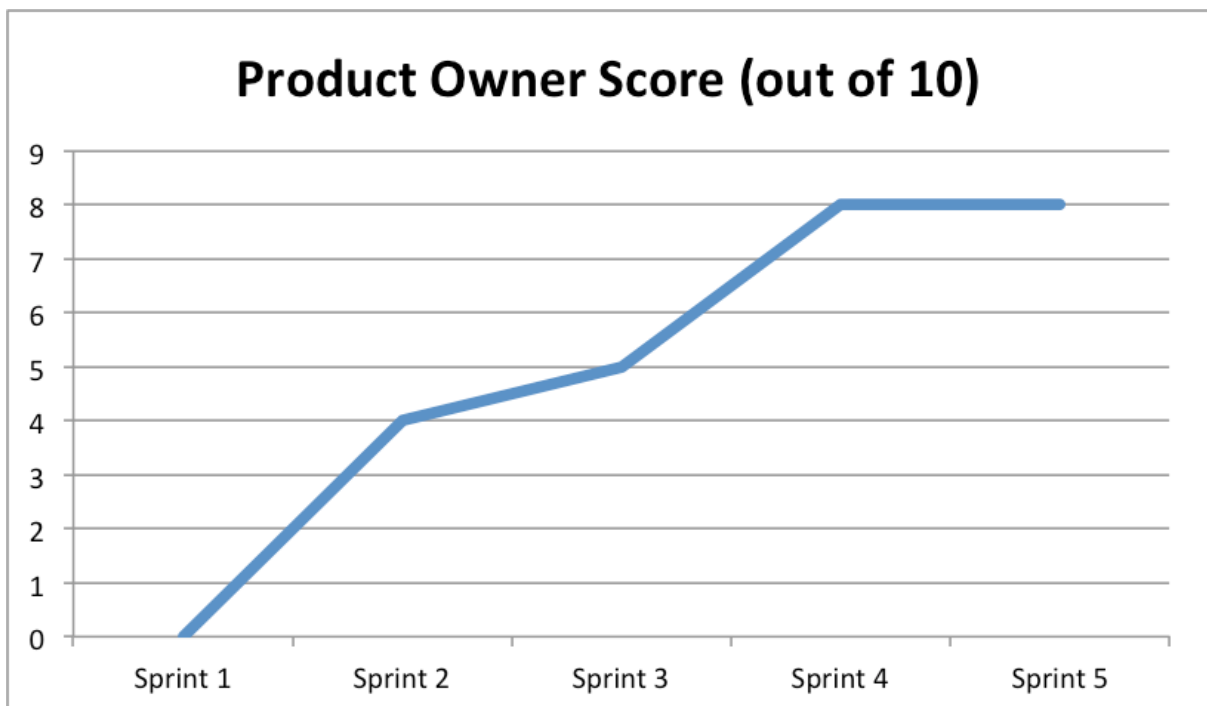
Det vi hade kunnat göra bättre är hur vi tolkade KPI:erna, och speciellt hur stress inom gruppen eventuellt hänger ihop med vår uppskattade och faktiska velocity. En av de positiva aspekterna med att jobba enligt Scrum är att stressfaktorer minskar. I vårt fall använde vi dock oss inte av de föregående sprintarnas stress i kombination med velocity för att se hur vi skulle kunna förbättra arbetet framåt och hur vi planerade sprintar för att minska stressen inom gruppen. Det är dock svårt att ur graferna avläsa någon särskild korrelation mellan stress och velocity. Hade vi arbetat med detta projektet under en lång tid framöver är sannolikheten stor att man skulle ha haft nytta av att titta tillbaka på huruvida stressen på något sätt kunde kopplas till hur vi planerade våra sprintar, i form av velocity och andra faktorer. På det sättet hade förbättringar i sprintplanering och arbetet förmodligen kunnat uppnås.



Förväntad mot faktiskt velocity för teamet, per sprint



Förväntad mot faktiskt stressnivå för teamet, per sprint



"Product Owner Score" för teamet, per sprint

D3: An initial product backlog and business model canvas

Product backlog

När det var dags att skapa den första product backlogen upplevdes det som mycket svårt. Komplexiteten berodde delvis på att det var kort om tid att sätta sig in i det relativt omfattande caset men även att det saknades tid att interagera med produktägaren. Det var dock väldigt lärorikt att redan från start tvingas tänka i banor av kundvärde och tunna skivor, en erfarenhet som var nyttig att ha vidare i projektet.

Business model canvas

Att ta fram en business model canvas var inte främmande för någon i gruppen. Å andra sidan kändes det som att appliceringen av en sådan inte var klockren i denna situation. Hela processen byggde ändå på att skapa värde för kunden. Hade varit viktigare att implementera om projektet hade helt fria tyglar.

Ett antal risker påtalades i inlämningen. Primärt nämns risker kopplade till kunskapsbrist och kommunikationskanaler som ej är färdigutvecklade eller för smala. För att täcka kunskapsgapet har det fungerat relativt bra att inhämta information på egen hand även om det tagit mycket tid. Något som i slutändan gjort att värdeerbjudandet fått lida lite. Risken kopplad till kommunikationsmöjligheter med kundsegmentet har inte varit lika stor som förväntat. Det har funnits tillräckliga möjligheter att interagera. En risk som identifierades redan vid denna inlämningen var den problematik som uppstod kring utveckling av backend.

D4: Half-time evaluation

I halvtidsutvärderingen analyserades vad som dithills fungerat bra respektive sämre. Den del som genomgående hade fungerat bra var samarbetet inom gruppen. En bidragande faktor till detta var de många timmar av kodande som spenderades i samma rum. Det var på så sätt enkelt att kommunicera, överbygga kunskapsgap samt parprogrammera. Denna arbetsmetod togs därför med till de kvarvarande sprintarna. Viktigt att nämna är dock att en gruppmedlem befunnit sig på annan ort under stora delar av projektet och problematik med detta nämns också halvtidsutvärderingen. Primärt handlade det om att vissa saker som för övriga gruppmedlemmar, vilka närvarat vid fler fysiska möten, verkat självklara inte varit tillräckligt tydliga samt att möten inte annonserats tillräckligt uppenbart. Den åtgärdsplan som presenterades i form av dagliga stand-ups via Slackkanalen samt bättre visualisering av möten i den gemensamma kalendern förenklade vid de avslutande sprintarna. Givetvis hade det varit önskvärt att samtliga deltagare kunnat närvara fysiskt vid möten men med hjälp av videosamtal och ovan nämnda åtgärder utgjorde den geografiska spridningen inga större problem för projektets utfall.

Den del som orsakade den största problematiken under den inledande delen av projektet var att få upp en utvecklingsmiljö. Samtliga deltagare var införstådda med att det inte skulle vara

någon smekmånad att få denna att fungera men hade inte heller tänkt att det skulle bli så problematiskt som det blev. När den grundläggande setupen var på plats flöt det hela dock på bättre och en viktig erfarenhet angående en sådan process omfattning erhöles. De många timmar som lades på att söka upp information om APlar och mjukvaruverktyg bidrog visserligen till att bygga upp kunskap men mer tid hade kunnat avvaras till värdeskapande för kunden om bättre dokumentation erbjudits från PortCDMs håll. Vidare hade en även bättre kommunikation grupper emellan kunnat föras. Problem som var generella lyftes ofta i den gemensamma Slack-kanalen och även det faktum att de var lösta men kanske inte alltid på vilket sätt.

Trots att mycket problematik med det tillhandahållna APllet uppstod nämns i halvtidsutvärderingen att utvecklingen av frontend fungerat bra. Även under de kvarvarande sprintarna byggdes dessa delar till viss del separat. Det var en fördel eftersom att en grafisk bild kunde ges till produktägaren även om all bakomliggande logik inte var fullständig. Därav möjliggjordes kontinuerlig respons och riktningsändringar kunde utföras vid behov. Att istället göra både back- och frontend samtidigt hade givetvis varit ett alternativ men då den förstnämnda aspekten medförde en del problem hade det förmodligen saktat ner utvecklingen. Det hade dessutom gjort att den tidigare nämnda responsen inte varit möjlig, därför fortsatte utvecklingen på samma sätt.

Ytterligare en detalj som diskuteras i halvtidsrapporten är det faktum att interaktion med produktägarna har fungerat bra. Även under slutfasen av projektet flöt det på bra och det kändes enkelt att tala om lösningar i termer av user-stories. Det faktum att närvarande produktägare var insatta i tekniska och inte enbart praktiska aspekter underlättade. Hade aktören inte haft denna insyn hade det förmodligen inte varit möjligt att hålla diskussioner på samma detaljnivå.

Sammanfattning

Vad har vi lärt oss och skulle gjort annorlunda om vi hade gjort det igen, eller fortsatt med projektet?

Fortsätt experimentera hur vi estimerar, och involvera stakeholders mer

Vi hade reviews med PortCDM varje vecka, men vi involverade dem inte i planeringen. Vi hade gärna fortsatt vilja experimentera med att involvera dem i planeringen för att skapa en bättre förståelse mellan vad vi kan leverera mot deras krav, och skapa en högre transparens. T ex konkretisera vad velocity-poängen innebär och att hålla korten kundfokuserade. Det var inte tydligt att vilket värde velocity-beräkningar och planera med scrum poker hade, men tillsammans med PortCDM hade det kunnat vara mer till hjälp.

Välstädd scrum-bräde med tydliga Definitions-of-Done och ansvarsområden

Initialt använde vi nya scrumbräden för varje sprint, user stories var löst definierade och det var oklart vem som var ansvarig för vad. Situationen blev mycket tydligare när vi endast hade ett välkategoriserat bräde, när user stories var skrivna med en definierad syntax, när vi blev striktare med våra Definitions-of-Done, och även om det kan anses vara "icke-scrum", så var det även hjälpsamt att ha en ytterst ansvarig person på varje story. Detta är något vi skulle ha varit striktare med från början, och något vi skulle fortsatt med framöver.

Visualisera och kommunicera mycket, speciellt i ett geografiskt splittrat team

Initialt hade vi problem med att synka saker med teammedlemmar som inte var fysiskt närvarande, då mycket av diskussion och beslut togs i samma rum. Vi introducerade dagliga standups över slack, bättre visualisering över möten i Google Calendar, samt blev striktare på att visualisera i Trello-brädet kring vem som jobbade med vad. Detta hjälpte till att håll teamet i synk, och något vi absolut skulle fortsatt med.

Tänk arkitektur tidigt, hantera tech debt tidigt

Vi började koda utan någon större tanke på design patterns eller struktur, något som snabbt skapade stor tech debt och refaktoreringar som tog tid, och ett team som var i osynk kring arkitekturen. Istället borde vi tidigt ha bestämt övergripande design patterns och bygga utifrån dem. Vi stötte även på ytterligare problem när den övergripande design pattern vi valt (MVP) var svår att kombinera med det grafiska ramverket vi valt. Om vi fortsatt hade vi därav skapat övergripande modeller kring programmets struktur, och guidelines kring hur vi sammanfogar vårt designmönster, ramverk, och teknologi. Detta för att minimera hur mycket tech debt vi skapar, och för att hålla teamet i synk kring arkitektur.

Bättre metoder för testning vid kontinuerlig integration

Fullständig testdriven mjukvaruutveckling (TDD) hade eventuellt saktat ner oss för mycket, att skriva Test Automation (TA) skulle nog vara överkill, men vi hade absolut kunnat tjäna på att implementera en checklista för manuell regressionstestning, samt inkorporera enhetstestning som en del av processen istället för som en eftertanke. Fortsatt utveckling efter det hade varit automatiska byggtest vid varje Pull Request, och en TA-svit efter att vi skapat mer stabil kod.

Välj verktyg teamet/branschen har erfarenhet av, och få alla att använda det.

Vi valde alltid verktyg som teamet antingen hade tidigare erfarenhet av, eller som var branschstandard. Speciellt när det är ett kort projekt så vill man inte ha för mycket overhead med att lära sig nya verktyg, och man vill kunna ha så mycket supportresurser som möjligt. Vi lärde oss också att tiden för felsökning och synk minimeras drastiskt om alla till så stor grad som möjligt kör samma plattformar. Detta är något vi hade fortsatt med, men hade vi haft längre tid och mer resurser hade vi kanske utvärderat ytterligare verktyg för att se hur dem passar våra use cases. Hade vi fortsatt projektet hade vi också gått ifrån JavaFX som GUI-lösning, eventuellt till förmån för JavaScript.

Integrera målen från det sociala kontraktet i retrospectives/review

Retrospectives fungerade bra för att identifiera problem att förbättra inför nästa sprint, men vi saknade uppföljning och utvärdering kring de målen vi definierade i vårt sociala kontrakt. Detta kunde likt retro ha gjorts efter varje sprint där teamet pratar om hur säkra de känner sig över att uppfylla målen. Fortsatt skulle detta även kunna integreras i tre-månaders roadmaps.

Artefakter från prototypen

Screenshot av vår applikation

FindBugs report on final release

FindBugs-IDEA: Analysis Finished
Found 0 bugs in 18 classes [more...](#)
using FindBugs-IDEA 1.0.1 with Findbugs version 3.0.1

FindBugs-IDEA: found 0 bugs in 18 classes
using FindBugs-IDEA 1.0.1 with Findbugs version 3.0.1

portcdm[portcdm]

Compile Output Path (IDEA)

C:/Users/arono/workspace/portcdm/src/main/java
C:/Users/arono/workspace/portcdm/src/main/resources
C:/Users/arono/workspace/portcdm/src/test/java

Sources Dir List (3)

C:/Users/arono/workspace/portcdm/src/main/java
C:/Users/arono/workspace/portcdm/src/main/resources
C:/Users/arono/workspace/portcdm/src/test/java

Analyzed Classes List (18)

C:\Users\arono\workspace\portcdm\target\classes\model\ApiConfig.class
C:\Users\arono\workspace\portcdm\target\classes\model\LocationManager.class
C:\Users\arono\workspace\portcdm\target\classes\model\MessageSender.class
C:\Users\arono\workspace\portcdm\target\classes\model\PortCallManager.class
C:\Users\arono\workspace\portcdm\target\classes\model\StatementReader.class
C:\Users\arono\workspace\portcdm\target\classes\model>Status.class
C:\Users\arono\workspace\portcdm\target\classes\model\TimeStampManager.class
C:\Users\arono\workspace\portcdm\target\classes\presenters\DetailedViewPresenter.class
C:\Users\arono\workspace\portcdm\target\classes\presenters\Main.class
C:\Users\arono\workspace\portcdm\target\classes\presenters\MainViewPresenter.class
C:\Users\arono\workspace\portcdm\target\classes\views\View.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\ApiConfigTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\LocationManagerTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\MessageSenderTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\PortCallManagerTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\StatementReaderTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model>StatusTest.class
C:\Users\arono\workspace\portcdm\target\test-classes\model\TimeStampManagerTest.class

Aux Classpath Entries (2)

C:/Users/arono/workspace/portcdm/target/classes
C:/Users/arono/workspace/portcdm/target/test-classes

GitInspector

Analysen kördes med filändelserna .java och .fxml

```
Statistical information for the repository 'dat255project' was gathered on 2017/06/02.
The following historical commit information, by author, was found:

[1mAuthor          Commits    Insertions    Deletions    % of changes[0;0m
Aron Sai           63         3115         2581         54.11
Hannes Lagerroth   5           219         194          3.92
Quaxi              13         521         274          7.55
anton1904          3          296         4           2.85
oskar3000          14        1418         212         15.48
teomar            23        1088         605         16.08

Below are the number of rows from each author that have survived and are still intact in the current revision:

[1mAuthor          Rows      Stability    Age      % in comments[0;0m
Aron Sai         1662       53.4         1.5       14.56
Hannes Lagerroth 4           1.8         2.4        0.00
Quaxi            97        18.6         0.0        1.03
anton1904        65        22.0         2.1        3.08
oskar3000        739       52.1         1.9        0.00
teomar          377       34.7         0.8       35.54

The following history timeline has been gathered from the repository:

[1mAuthor          2017w17    2017w18    2017w19    2017w20    2017w21    2017w22    [0;0m
Aron Sai          -++-+++++  -+++++    ++  -++-+++++  ++++++    -++-++
Hannes Lagerroth  -+
Quaxi              -++
anton1904          +
oskar3000          +
teomar            ++++++    +
[1mModified Rows:  [0;0m          370    1495      88      4459      1755      2360
```

Hannes Lagerroth och Quaxi är samma person, men på olika konton.

Appendix

Documentation of sprint retrospectives

Sprint 1

Vad har gått bra under förra sprinten?

- Fått en del grundläggande saker "up and running"
- Samarbetet har fungerat bra
- Bra moral i teamet
- Även om vi inte har producerat mycket, så har vi jobbat mycket och försökt mycket

Vad har gått dåligt under förra sprinten?

- Fått en del grundläggande saker "up and running"
- Samarbetet har fungerat bra
- Bra moral i teamet
- Även om vi inte har producerat mycket, så har vi jobbat mycket och försökt mycket

Från detta - vad vill vi ändra och förbättra till nästa sprint?

- Testa dagliga standups asynkront i Slack
- Vara bättre med att parallellisera arbetet
- Fortsätt jobba mycket när vi sitter i samma rum, det fungerar bra

Sprint 2

Vad har gått bra under förra sprinten?

- Att hela teamet konvergerade till en utveckelmiljö (IntelliJ) har fungerat bra och tagit ner på mycket strul
- Kommit förbi mycket tekniskt strul
- Hög velocity jämfört med förra veckan
- Fick mycket gjort!
- Vi var väldigt effektiva med den tiden som vi hade

Vad har gått dåligt under förra sprinten?

- Oklarheter i teamet, speciellt på avstånd.
 - Vem är Scrum Master?
 - Vem gör vad?
 - Vad förväntar vi oss av våra teammedlemmar?
- Denna sprint blev väldigt kort
- Mycket fokus på felsökning, inte så mycket på att leverera värde till kund
- Synkning mellan teammedlemmar kunde ha fungerat bättre

Från detta - vad vill vi ändra och förbättra till nästa sprint?

- Var bättre på att använda kalendern för att hålla hela teamet i synk
- Dagliga standups måste ske innan 10:00 varje dag
- Använd en scrum board istället för flera för att göra det enklare att se vad som gjorts innan.

Sprint 3

Vad har gått bra under förra sprinten?

- Ur teknikträsket!!!
- Fått mycket funktionalitet implementerad och levererad värde till slutkund
- Bättre struktur kring våra ways of working
- Fungerat bra med att parallellt utveckla funktionalitet som fungerar tillsammans.

Vad har gått dåligt under förra sprinten?

- Synkningsproblem pga kandidatarbeten.
- Inte uppnått att utvecklingen sker i "tunna tårtbitar" än.
- Tasks är relativt horisontella fortfarande.

Från detta - vad vill vi ändra och förbättra till nästa sprint?

- Inget speciellt

Sprint 4

Vad har gått bra under förra sprinten?

- Mycket gjort!
- Fått gjort mer eller mindre allt vi satt på sprinten, bra planering!
- Tydliga definitions-of-done
- Mest produktiva veckan hittills
- Bra DoD ger oss klaritet kring vad som är klart och inte klart
- Alla behöver inte koda för att vara produktiva och fånga kundvärde, och det har fungerat bra
- Holistisk vy över projektet

Vad har gått dåligt under förra sprinten?

- Rörigt med parallellisering. Oklart med hur mycket som var färdig på andra stories
- Vi borde ha prioriterat bättre

Från detta - vad vill vi ändra och förbättra till nästa sprint?

- Fortsätt fokusera på att vara holistiska
- Knyta ihop säcken

Sprint 5

Vad har gått bra under förra sprinten?

- Levererade vad vi skulle
- Lyckades nästan estimeras

Vad har gått dåligt under förra sprinten?

- Skitstressigt
- Alla kunde av olika anledningar inte vara delaktiga

Från detta - vad vill vi ändra och förbättra till nästa sprint?

- Ingen nästa sprint!