

# Reflektionsrapport

The Billy Corgan Project för Terminal 2

DAT255/DIT543 Software engineering project VT 2017



Källa: <https://www.timeout.com/newyork/music/interview-billy-corgan>

Medlemmar:

*Jonatan Almén*

*Johan Berndtsson*

*Simon Ceder*

*Adam Liljenberg*

*Pontus Stjernström*

*Miki Swahn*

## 1. Introduktion till systemet

Verktyget är till för att en terminal ska kunna administrera sina hamnanlöp. Nedan förklaras arbetsflödet i applikationen.

Här kan användaren lägga till ett nytt anlöp genom att tillföra nedanstående information, i sin helhet eller endast delvis. Minimikravet är att ange laycan start och end samt en av cargo load eller unload.

### Portcalls

Cancel

Mandatory

Load Cargo:

Unload Cargo:

Laycan start date: 

åååå-mm-dd

Laycan start time: 

--:--

Laycan end date: 

åååå-mm-dd

Laycan start time: 

--:--

Optional

Nickname:

Vessel IMO:

Portcall ID:

Add PortCall

Det hamnar i en lista med terminalens anlöp:

### Portcalls

Add Portcall

Incoming Portcalls:

Cargo to unload: Oil, 500000 barrels  
Cargo to load: Slop  
Laycan: 2017-10-30T15:59UTC to 2017-11-02T23:59UTC  
Name:  
Vessel IMO:  
PortCDM Portcall ID: urn:mrn:stm:portcdm:port\_call:SEGOT:8c4408a8-fc7a-4461-be4b-f5ad1e53be08

Cargo to unload: Oil, 23000 oil  
Cargo to load: Slop  
Laycan: 2017-08-24T12:00UTC to 2017-08-28T13:53UTC  
Name:  
Vessel IMO: urn:mrn:stm:vessel:IMO:7431923  
PortCDM Portcall ID:

Billy Corgan Project for Terminal 2

1

När användaren klickar på ett anlöp dyker en detaljerad vy över det anlöpet upp. Här är det möjligt att lägga till mer information om anlöpet och det finns ett flöde med PortCallMessages från PortCDM som rör detta anlöp. Flödet förutsätter att man har lagt in ett vesselID och/eller ett portcallID.

När terminalarbetaren ser andra aktörers intentioner och uppfattning av läget i flödet, kan denna justera den interna informationen. När man ändrat en tid kan man också skicka in det som ett PortCallMessage till PortCDM och välja vilken typ av timestamp det är.

Denna webbapp testades under scenariot 24:e maj och klarade att dela med sig av samt lyssna efter rätt information till övriga parter.

## 2. Avgränsningar och antaganden

Som grund har vi ansträngt oss för att i agil anda inte fatta beslut om hur vi avgränsar oss förrän vi verkligen måste. Nedan syns de stora besluten vi fattat och när under processen de fattades.

**20/4** - Vi gör en webbapplikation. Den bör kunna användas på terminalarbetarnas datorer och kräver inga nya installationer från deras sida. De använder gamla webbläsare på terminalerna så applikationen ska gärna vara kompatibel med versioner från och med Internet Explorer 11.

**21/4** - Arkitektur för applikationen. En egen backend (REST-API) och en fristående hemsida. Detta för att hålla isär frontend och backend så att teamet kan jobba självständigt i mindre grupper och för att det ska vara lätt att vidareutveckla andra frontends ifall det behövs i framtiden. Vår backend ska abstrahera bort mycket av komplexiteten i PortCDM:s backend.

**28/4** - Använda barebone mot PortCDM, då vi tyckte att det hade en lägre tröskel än de Java-bibliotek som tillhandahölls av PortCDM. Utvecklarna på PortCDM var positiva till förslaget att använda barebone och våra egna försök att få det att fungera ihop med spring-boot tydde på detsamma.

**2/5** - Efter ett möte med Mathias på RISE Viktoria, kunde vi göra antagandet att den första informationen en terminal har för att initiera ett anlöp är ett laycan och en last. Därför tog vi beslutet att vi inte skulle kräva något mer än ett laycan och en last för att skapa ett nytt anlöp. Laycan består av ett start- och slutdatum som agerar som en tidsram för när fartyget kan ligga vid kaj. Last ska antingen lastas, lossas eller både och.

**4/5** - Vi bestämde oss för att spara vår data i en fil istället för en databas, för att vi ville göra så lite som möjligt för att få det att fungera just nu. Däremot skall det vara lätt att senare ändra till en databas, vilket vi tycker att vi lyckades med.

### 3. Tekniska aspekter

Vi valde att göra ett eget Web-API som kunde inkapsla den mesta logiken så att frontenden bara behöver innehålla logik för att presentera data på ett bra sätt. Fördelen med detta är att det skulle vara lätt att skapa nya frontends ifall behov uppstår, det vill säga vi ville ha en lös koppling mellan frontend, företagsregler och logik.

Vi testade programmet enskilt, medan vi jobbade med det. Alla tester gjordes manuellt och vi använde ofta Restlet pluginen till Chrome för att dubbelkolla vad vi fick tillbaka från vår backend. I slutet av de flesta sprintar hade vi också ett möte före review:n där vi mergade våra olika branches till samma branch, och kontrollerade gemensamt så att features fungerade tillsammans. Vi hade ett gemensamt integrations test tillsammans med de andra grupperna innan redovisning, vilket gick bra för oss. Vi var nöjda med hur lätt det var för oss att anpassa oss till integrationsservern, då vi hade använt PortCallMessage version 0.0.16 under hela vår utveckling, men integrationsservern använder version 0.6. Eftersom vi hade separerat all kod som påverkades av sånt i egengjorda Java bibliotek var det trivialt att ändra och vår kod påverkades inte alls.

### 4. Användandet av scrum


#### **Roles, teamwork and social contract (relates to D1B)**

Gruppen har från starten av arbetet varit tydliga med vilka kunskaper olika gruppmedlemmar har och roller har delats ut efter detta. Gruppen består både av studenter från Datavetenskap och Industriell ekonomi. Vissa har haft kunskaper som passat för att göra backenden medan andra har kodat i HTML eller varit experter på git. Då alla har haft olika kunskaper har gruppen arbetat på ett tvärfunktionellt sätt.


Rollerna i varje specifik user story har delats ut under varje sprint i Trello under Sprint planning. Vi gjorde estimat och acceptanskriterier samt tilldelade personer vilka antingen själva eller i grupp fick göra en lista med tasks. Sprint planning genomfördes varje onsdag före stormötet direkt efter sprint review och retrospective.


#### **Used practices (pair programming, stand-up meetings, etc.)**

Daily scrums i Slack: Varje dag kl 10 skrev en bot i Slack att det var dags för daily scrum. Där skapades en tråd där gruppmedlemmarna fick berätta vad de gjort, skulle göra och behövde hjälp med. Det fungerade mycket bra eftersom vi inte sågs varje dag i veckan och det tvingade oss att kommunicera vad vi höll på med så att gruppen fick koll. Nedan finns två exempelbilder på hur daily scrum i slack fungerade.



**Zapier** APP 10:04 AM ☆


Daily scrum reminder! Start your thread from me.  
 What have I done?  
 What am I doin'?  
 Whats my plan?


**3 replies** Last reply 13 days ago



**Zapier** APP 10:05 AM

Daily scrum reminder! Start your thread from me.  
 What have I done?  
 What am I doin'?  
 Whats my plan?


**6 replies** Last reply 12 days ago


**Zapier** APP 10:02 AM

Daily scrum reminder! Start your thread from me.  
 What have I done?  
 What am I doin'?  
 Whats my plan?


**9 replies** Last reply 11 days ago


**Zapier** APP May 18th at 10:05 AM  
 in #dailyscrum

Daily scrum reminder! Start your thread from me.  
 What have I done?  
 What am I doin'?  
 Whats my plan?

6 replies


**putzies** May 18

Igår gjorde jag ingenting efter mötet, kollade mest på effekterna i koden av att ändra till PortCallMessage v0.6. Idag tänkte jag sätta igång och ändra och se om det fungerar att skicka in messages till den servern (edited)


**skillzore** May 18 ☆

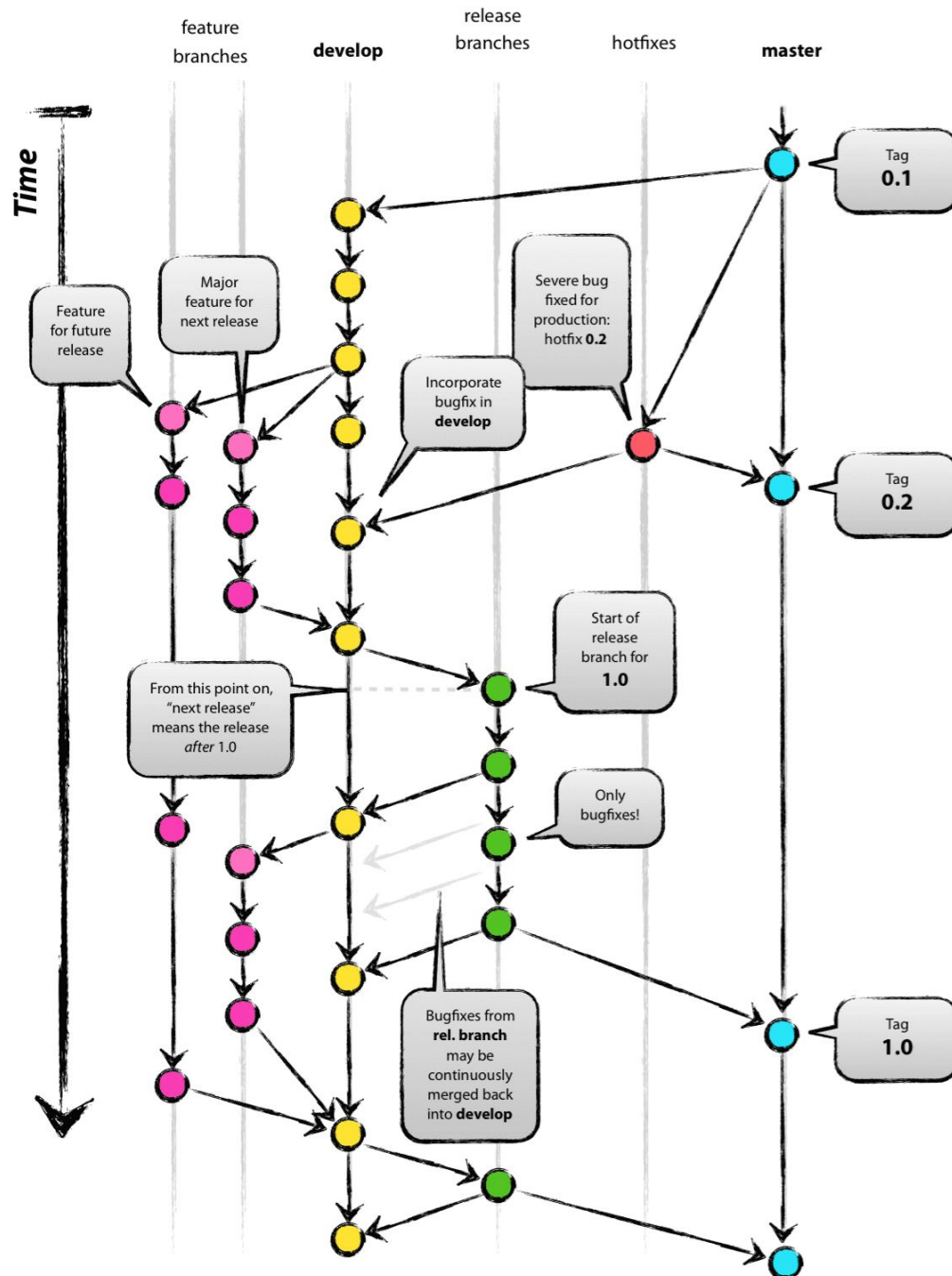
Har "lagat" de flesta buggar i min user story och bytt implementation av lagringen. Istället för att lagra själva (vilket leder till att vi missar alla meddelanden som kommer när vi inte är i detaljvyn), så låter vi portCDM "lagra" dem och hämtar alla meddelanden från en viss given tid istället. typ 1970-01-01 00:00:00 eller nått. Behöver förmodligen hjälp av @putzies att kunna hämta meddelanden från en viss tid, då vi behöver utöka MessageQueueService som han skrivit. (edited)


**pontusstjerna** May 18

Satt hela kvällen igår, har byggt ut hela datamodellen så att den innehåller allt. Gjorde om hela systemet för just tidsstämplar så det ska vara lätt att sättat ut fler.  
  
 Nästa steg är att sätta ut ännu fler tidsstämplar i detaljvyn och skapa knappar för att skicka till PortCDM.  
  
 Kan behöva hjälp med att skapa alla dessa fält eller

Parprogrammering förekom men det mesta samarbetet var på olika maskiner med hjälp av commits sinsemellan. Detta var dock väldigt olika från sprint till sprint då varje delgrupp var ansvariga för att dela upp arbetet under sprinten, så det var flexibelt hur vi ville jobba. Oftast var två personer ansvariga för en user story då det skapade lagom stora team inom gruppen. Detta fungerade väldigt bra.

Vi beslutade oss för att använda versionshanteringsverktyget Git tillsammans med grätistjänsten Github. I och med detta val gjorde det att vi enkelt kunde arbeta distribuerat enligt följande bild nedan.



Källa: <http://nvie.com/posts/a-successful-git-branching-model/>

### **Tidsdistribution (person / roll / tasks etc.)**

Under arbetets gång uppskattades det för varje User Story en tidsåtgång. I nästan varje sprint överskattade vi våra estimates något och arbetet som utfördes var därför ofta mindre än väntat. Vi korregerade detta kontinuerligt under sprintarnas gång, tills det slutade med att vi underestimerade hur lång tid vissa user stories skulle ta. Då scrum bygger på att man skall klara av sina User stories under sprinten gjorde det att arbetsfördelningen skilde sig något mellan sprintarna men inte för mycket.

Den enda fasta rollen vi haft är scrummaster, som vi inte roterat då vi ville ha kontinuitet i scrum of scrums samt att det är ologiskt att byta ut en roll så snart en person blivit bra på den, dvs. kommit över inlärningskurvan. I övrigt har vi inte haft några givna roller, utan gruppmedlemmar har i stor utsträckning fått jobba med de User Stories de varit mest intresserade av. Detta fungerade bra då det aldrig blev några konflikter angående detta.

### **Effort, velocity och task breakdown**

Uppdelningen till tasks för varje user story har fungerat mycket bra under projektet. Våra tasks har konstant varit anpassade för de som arbetat med den delen av den vertikala User Storyn. Detta kan till stor del ha berott på att vi haft olika roller med olika kunskaper som krävt att våra tasks var anpassade för det vi varit bäst på. Om någon task visade sig vara mer ansträngande än någon annan fanns det ofta gruppmedlemmar vars tasks innebar mindre arbetsbörda än andras. Detta blev tydligt med daily scrums då man kunde be om/erbjuda hjälp dagligen.

Vår velocity har varit konstant över arbetets gång, men tidsuppskattningarna har fått anpassas för att stämma med velocityn. Nedlagd tid i kursen har varit mer eller mindre jämn. Internt sattes 15 timmar/sprint som velocity per person för att arbeta med produkten och det fungerade bra, speciellt då vi tidigt insåg att vi skulle bli klara med projektet i tid. Möten och gästföreläsningar beräknades nämligen ta 5h/vecka. Detta kommer ifrån att till att hela 20h/vecka antogs vara den tid var person lägger ner.



## 5. Reflektion över sprint retrospectives

Våra sprint retrospectives var korta och koncisa. Vi försökte gemensamt komma på saker att fortsätta med, att börja med och att sluta med. Allting vi tog upp var riktat mot hela gruppen och inte mot enskilda personer. Mötet organiserades genom att varje person i gruppen fick i turordning säga saker som borde tänkas på. Det var dock bättre närvaro av alla gruppmedlemmar i början och så småningom speglades resultaten från de sista sprint retrospectives av de som var med och deltog.

De stora och återkommande sakerna vi tog upp i retrospectives var att kommunicera mera, lägga ner tillräckligt med tid, och hjälpas åt. Kommunikationen rörde bland annat att fråga om hjälp av andra och att svara varandra mer frekvent, samt att skapa transparens i arbetet genom att i Trello (sprint backlog) dokumentera vad man arbetade på. Att lägga ner tillräckligt med tid handlade om att alla inte alltid gjorde 20 timmar i veckan och det fanns en stress över att bli klara och leverera ett tillräckligt bra system till den 24:e maj. Att hjälpas åt gjordes i stor utsträckning och var något som vi genomgående ville fortsätta med. Vi kunde följt upp förra sprintens retrospectives för att se om de hade uppfyllts vilket vi valde att inte göra. I efterhand kan det tänkas att vi lättare hade kunnat fånga upp återkommande problem såsom kommunikation och tidsplanering.

## 6. Dokumentation av sprint retrospectives

Efter sprint 1: 26/4 (*Sprint 1 inleddes 19/4*)

### **Fortsätta**

Alla tar ansvar och försöker lösa problemen!

- Engagemang
- Fokusera på GitHub-processen
- Hjälpa varandra
- Träffas (framförallt sitta och jobba nära varandra)
- Vara nyfikna
- Posta ALLA möten på slack

### **Börja**

- Skapa mer specifika tasks och att någon tar på sig en specifik uppgift
- Använda trello mer, assigna sig själv till en user story och skapa tasks varefter det kommer fram.
- Använda slack för daily scrums! (DVS INTE SLARVA)
- Börja skapa trådar på slack (inte nya posts)
- Buggar kan läggas in som tasks, om de inte fixas så läggs de in som issues på github
- Kommunicera mer angående möten. Pinga på slack.

### **Sluta**

- Tänka för mycket på vad som ska göras i framtiden, fokusera på nuvarande sprint!

### **Sammanfattning**

- Hjälpa varandra, Träffas, Mer specifika tasks och personligt ansvar, Använda trello mer.

Efter sprint 2: 3/5

### **Miki:**

Positivt sprint:

- Fortsätt med Daily Scrum!
- Fortsätta hjälpa varandra, ( kommer skära genom alla lager )

Negativt:

- Börja följ daily scrum-protokollet:  
Vad har jag gjort igår?  
Vad ska jag göra idag?  
Vad behöver jag hjälp med?
- Börja skapa trådar i slack och sluta med random grejer där det inte hör hemma

### **Pontus:**

Positivt:

- Fortsätta vara engagerade.

Negativt:

- Börja lägga den tiden som är utsatt.

### **Jonatan:**

Negativt:

- Börja jobba mer hållbart.

**Simon:**

Negativt:

- Börja jobba mer holistiskt.
- Börja planera workshops.
- Börja kommunicera dina intentioner.

**Sammanfattning:**

Kommunicera mera och lägg varken ner för mkt eller för lite tid och gör daily scrum mer rätt.

Efter sprint 3: 10/5

**Fortsätta**

- Dela upp i sub-teams, som sen kan dela upp arbetet med user stories ytterligare.
- Fortsätta diskutera och använda slack så pass mycket som vi har gjort
- Bra atmosfär
- Göra bra och vertikala user stories (lyckades vi med i sprint 3, men inte 1 och 2)

**Börja**

- Lägga den tid som är utsatt (behöver vi fler user stories per sprint?)
- Försöka börja jobba med user stories så tidigt i sprinten som möjligt, för att upptäcka om det är svårare än man trodde. (svårt för oss med avseende på andra kurser)
- Se till att ha kontakt med alla i gruppen kontinuerligt, om inte in person, så via slack.

**Sluta**

- Vara klar med planeringen på fredag
- Vara sena (det är vårt sociala kontrakt)

**Sammanfattning**

- Mindre team, lägga ner tid, håll kontakt.

Efter sprint 4: 17/5

**Börja**

- Jobba mer. Lägg ner tid. Upp med tempot.

**Sluta**

- Jobba individuellt om vi är beroende av varandra.

**Fortsätta**

- Kommunicera frekvent.

**Sammanfattning**

- Kämpa nu sista biten för att bli klara

Efter sprint 5...

...hölls inget retrospective eftersom projektet tog slut och gruppen upplöstes. Med andra ord; behovet för gruppens utveckling var lågt 24/5.

## 7. Reflektion över sprint reviews

Sprint reviews gjorde vi i muntlig form där vi var och en gick igenom vad vi hade implementerat under sprintens gång. Sprintbackloggen fanns till hands för att verifiera att alla user stories implementerats. Det var värdefulla möten för gruppen där man fick lära sig om vår nuvarande MVP och dess funktioner. Eftersom vi delade upp de user stories som fanns mellan oss var inte alla delaktiga i all implementation, men fick en catch up varje vecka om vad som gjorts. Oftast gjorde vi det rent praktiskt genom att testa mjukvaran på respektive brancher på våra egna maskiner.

Dessa möten hölls i på onsdag förmiddag 4 av 5 sprintar. Endast en gång höll vi retrospectivet ihop med produktägaren på onsdagens stormöte. Detta visade sig vara dåligt då tiden inte räckte till att göra sprint planning samma dag vilket gick ut över sprintens produktivitet. Vi höll oss därför till att göra retrospectives utan produktägaren. Under stormötena då vi träffade produktägaren brukade vi oftast bara förklara vad vår MVP gjorde och vad vi planerade göra nästkommande sprint, samt ställde frågor för att få vidare information. Vi kunde vart bättre på att varje vecka visa upp vår MVP, då hade vi kanske fått mer konkret feedback på layouten vilken inte kunde förklaras i ord.

## 8. Tillvägagångssätt för användandet av nya verktyg och teknologier

Första sprinten försökte vi lära oss hur ett REST-API fungerar och hur PortCDMs API fungerar genom att använda en plugin till webbläsaren Chrome vid namn Restlet. Utifrån det kunde vi förstå hur man skickar PortCallMessages, visar data samt prenumererar på t.ex. ett portcallId.

Vi bestämde oss sedan för att skapa en webbapplikation med ett eget REST-API som vi sedan anropar från en hemsida för att visa upp i våran frontend. För att uppfylla detta använde vi oss av verktyget och javabiblioteket Spring-boot som är ett java ekosystem som är gjort för att skapa webbapplikation och webb-API:er. Argumentet för detta REST-API är att kunna utvidga vår applikation med andra frontendar såsom appar, desktop klienter m.m. Detta verktyg var både komplicerat och nytt för de flesta av oss men som tur är väldokumenterat.

När webbservern var på plats delade vi generellt upp arbetet efter det i backend och frontend där backend var att koppla vår java backend med PortCDMs API och frontend som var att visa data från vårt eget REST-API någorlunda användarvänligt på en HTML-sida genom javascript.

Efter detta, när all boiler-plate kod var på plats, använde vi så gott som denna uppdelning tills projektet var klart. På slutet testade vi mycket tillsammans och försökte lösa de största problemen utan att egentligen skapa flera nya brancher och dela upp arbetet.

Under projektets gång har vi följt ungefär samma mönster hela tiden:

1. Testa det som redan finns och lära sig hur koden är uppbyggt och hur allt fungerar.
2. Skapa en git branch för varje ny feature i programmet
3. Skriva kod utifrån nuvarande kunskap, internet samt existerande kod
4. Testa funktionaliteten och repetera eventuellt steg 3
5. Gör pull request till dev-branchen i repositoryt
6. Någon annan i gruppen granskar PR och uppdaterar sedan utvecklingsbranchen med den nya featuren

I punkt 1 ingår en hel del googlande, läsande av dokumentation samt testande.

## 9. Reflektion över förhållandet mellan prototyp, process och värde för intressenter

Under arbetets gång har relationerna mellan prototypen, processen och projektägarens värde förändras. Till att börja med hade gruppen ett väldigt stort fokus på prototypen (hur den skulle se ut, vad den skulle byggas i och hur man skulle imponera med en bra prototyp). När gruppen fått mer information och en större förståelse för hur Scrum fungerar insåg vi att man med hjälp av en scrum-process kunde utveckla prototypen på ett effektivt sätt med gruppuppdelade och vertikala User Stories. Processen var i de första sprintarna mycket effektiv för att uppnå de gruppen ville för att nå sin prototyp men produktägarens värde var under denna tid av mindre värde för gruppen.

Processen såg i stort sett likadan ut under arbetets gång då vi tidigt märkte att den fungerade, men när produktägarens värde blev tydligare genom möten förändrades prototypen. Prototypen blev mer avskalad och funktionell och mindre anpassad för att fungera som en lösning med extrafunktioner. En tydligare förståelse för värdet gjorde att vi skrev om vår product backlog något.

Prototypen förändras alltså på grund av en ökad förståelse för produktägarens värde. Den kan bli felaktig på grund av att projektgrupper siktar fel och missar målet. I vårt fall var det agila arbetssättet avgörande för att vi skulle kunna ändra vårt sikte mot det riktiga värdet. Att använda sig av en process som ständigt tillåter förändring gör att prototypen kan ha ett maximalt värde då man kontinuerligt kan uppdatera målen när prototypen alltid skall ha ett strikt förhållande till användarens värde. Vi kommer efter projektets slut vid liknande uppgifter att fortsätta öka vår förståelse för stakeholder value så kontinuerligt som möjligt med processer som anpassas för detta.

## 10. Relatera arbetsprocessen till litteratur och gästföreläsningar

Jämfört med böcker och föreläsningar har våra sprintar varit väldigt korta, vilket självklart gjorde att allt blev väldigt hoptryckt. På sprint planning fokuserade vi från början mycket på vad vi skulle göra nästa sprint, vilket gjorde att vi aldrig byggde upp en stor product backlog som vi kunde plocka user stories från, utan våra user stories växte snarare fram efter diskussioner med PortCDM på handledningsmöten. Vi hade några stora epics från början, från vilka en hel del av våra user stories utvecklades från.

## 11. Utvärdering av D1 - D4

### **D1: Lego scrum exercise and a social contract for the team**

Legoövningen var mycket bra för att få en idé om scrum och gav värdefulla insikter om förhandlig och att leverera värde. Lärdomarna vi kom fram till har vi aldrig återkommit till men det är nog både lika och olika från våra lärdomar av projektet.

Det sociala kontraktet gav kanske inte så mycket att ha första veckan innan vi ens börjat samarbeta, eftersom vi inte visste vad för problem som skulle dyka upp. Två veckor in visste vi till exempel att folk naturligt höll tider väl och att vi inte behövt diskutera igenom om man är sen efter 1, 5 eller 15 minuter. Vi märkte istället att det fanns andra problem i hur slack användes för kommunikation och vad vi förväntade oss på den fronten.

### **D2: Choose three KPIs**

Se sektion 11, KPI charts för en diskussion angående våra KPIer.

### **D3A: An initial product backlog & D3B: An initial business model for your system**

Produktbackloggen var jättesvår och konstig att göra när man inte hade koll på vad PO ville ha. Det blev många vilda gissningar. Sen visste vi inte hur vi fick lov att ändra på produktbackloggen under sprintens gång eller om den var låst. Vi hade ju hittat på mycket och ändrade därför självständigt, eftersom produktägaren inte var så delaktig i produktbackloggen ändå.

Lärorikt och intressant att göra en affärsmodell. Däremot var det väldigt förvirrande om vi skulle applicera konceptet från BMC på att vi läser en kurs, att vi levererar ett koncept till RISE Viktoria eller att vi skulle lansera en produkt. Det hade vi behövt bestämma oss för/ få bestämt först.

#### **D4: Half-time evaluation**

Intressant att se vilka problem man hade gemensamt med andra gruppen och se skillnader i hur man arbetat. Vissa saker som den ena gruppen misslyckats med vad beträffar scrum hade den andra gruppen lyckats med. Det påverkar bilden av om scrum är en bra metod för utveckling eller ej.

## **12. KPI-diagram, 0-1p**

Under projektets gång har vårt behov av KPI:er förändrats. Planen var att använda en *Burndown chart*, en *Happiness People Measurement* och en *Defect Ratio*. Användandet av en burndown chart visade sig likt att mäta antalet defekter vara tidskrävande på möten och gav inte gruppen någon information som ändrade vårt arbetssätt.

I fallet med burndown-diagrammet var det tillräckligt tydligt för gruppen hur långt projektet hade kommit och hur mycket som var kvar. De user stories vi upplevde vi var tillräckligt få för att kunna överblicka utan en Burndown Chart. Antalet defekter var inte heller viktigt att mäta då vi upplevde att de fel som upptäcktes kontinuerligt åtgärdades under sprintens gång och inte "levde kvar".

De KPI:er som vi istället valde att fokusera på var de Sociala KPI:erna. Vi märkte att då projektet pågick under en kortare tid och med människor som inte arbetat tillsammans tidigare var viktigare att mäta gruppens uppfattningar av projektet och varandra istället för själva produkten. Dessa värden gjorde att vi kontinuerligt kunde utvärdera vår arbetsmiljö utan att ta allt för mycket tid från projektet.

Informationen samlades in via ett google form minst en gång i veckan och sammanställdes grafiskt automatiskt. Genomgående trender är svåra att urskilja men under projektets gång har värdena inte sjunkit allt för lågt enligt gruppen.

