Final Reflection

Group: Hope

Anton Ekman, Carl Andersson, Emil Pettersson, Joakim Lönnerstedt, Jonathan Lawrence, Thomas Solibi & Tor Clementz

Agile software project management (DAT257)

October 2020

1 Customer Value and Scope

1.1 The chosen scope of the application under development including the priority of features and for whom you are creating value

-[A]

Applikationen går ut på att alla användare har sitt eget konto som man kan addera, från en lista av mat och dryck, vad man ätit respektive druckit under dagen till en annan lista som ritar ut en graf där y-värdet är utsläppet (Kg CO_2e) samt x-värdet är vilken dag. Man skriver alltså in antalet gram av det man ätit så multipliceras det med koldioxidutsläpp per gram för just den råvaran, taget från en trogen källa från Chalmers. [1]
Applikationen är riktad till användare som vill få koll på sina utsläpp och få det visualiserat för att kunna arbeta med att förbättra sina utsläpp. Idén kom till oss när vi tittat igenom alla 17 FN-målen och funderat på vad som är mest relevant för oss, det vill säga vad vi kan arbeta på för att förbättra den nuvarande situationen och vilka mål vi kan jobba mot.
Från början hade vi även tänkt att inkludera alla sorters utsläpp, till exempel avgaser och annan allmän konsumtion, men insåg ganska tidigt att vi behövde minska vårt scope för att inte överrösta oss med idéer och inte få något gjort. Så då blev det matkonsumtion eftersom det är något alla människor behöver och det finns mycket man kan göra när det kommer till utsläpp inom detta.

-[B]

Målet med applikationen var att kunna visualisera sina utsläpp på maten man ätit över tid och kunna jämföra med vänner. Detta innebar då att varje användare ska kunna ha ett eget konto med personlig data och att man lade till andra användare för att sedan kunna jämföra sina grafer bredvid varandra. Denna funktion var främst tänkt som ett sätt att få de två användarna som jämför att börja fundera på vad man kan göra annorlunda, hur man kan minska sina utsläpp.
En annan viktig funktion som applikationen skulle ha var att kunna ta bort saker från listan till grafen. Det vill säga ifall man råkat lägga in något som inte borde vara där eller råkat välja fel mängd. Detta kände vi i efterhand att detta var en funktion vi borde haft högre prioritet på då vi insett att det är en efterlängtad funktion, bara genom att testa programmet efter buggar och fel.

[1] https://research.chalmers.se/publication/515057/file/515057_Fulltext.pdf

När vi först kom igång med vår idé och vårt scope var tanken att listan med maträtter och dryck skulle varit tagen ifrån en slags databas med råvaror och dryck som var automatiskt uppdaterad och kategoriserad. Då det säkerligen finns en sådan databas någonstans, insåg vi senare att även detta blir utanför vårt scope, då vi måste lägga ner alldeles för mycket tid på att fixa till databasen till vår applikation. Det var istället smartare för oss att göra en liten enkelt lista för att få den huvudsakliga funktionaliteten att fungera.

-[A -> B]

Då gruppens kommunikation var bra under hela projektets gång hade en bättre planering hjälp oss nå våra slutgiltliga mål. Hade vi haft en klarare bild från första början vad vårt scope var och vad våra prioriteringar skulle vara, hade vi nått fler slutgiltiga mål.
En stor anledning till att scopet var otydligt i början är för att ingen gjort ett liknande projekt innan, så det var mycket nytt som skulle sitta på plats med user stories och scrum och så vidare. Det resulterade i att vi började arbeta med applikationen innan vi hade en tydlig bild. Vi använde inte heller oss utav några tekniska dokument, vilket även det hade kunnat hjälpa oss få en bättre överblick.
Att arbeta mer vertikalt är också en stor bidragande faktor till att de slutgiltiga målen inte nåddes. Som nämnt tidigare var ju idén med att lägga till vänner där för att kunna jämföra två användares utsläpp, och hade vi istället bara använt oss utav en user hade vi kunnat implementera jämförelse-funktionen mycket tidigare och fått den att fungera. Istället implementerade vi först att lägga till vänner, vilket tog en del tid och ledde oss inte till något mål utan det blev mer en addition som inte gav något till applikationen. Hade vi då som sagt istället använt oss utav en användare så hade vi nått det målet vi hade, inte riktigt hela funktionaliteten kanske men en funktionalitet som vi redan från början hade eftertraktat.
Hade vi arbetat mer vertikalt hade vi fått fler funktionaliteter att visa upp, än bara idéer och tankar som inte blev av.

1.2 The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

-[A]

När vi planerade vårt projekt var vi överens om att vi ville lära oss nya sätt att utveckla en applikation. Vi valde därför att använda JavaFX och Scenebuilder för att

utveckla vår applikation eftersom bara en av oss hade använt JavaFX innan. Vi ville också implementera en databas med hjälp av SQL för att lära oss hur man kan använda vår kunskap från databas kursen i ett verkligt scenario.

Vi valde att använda GitHub som vårt repository, dels för att vi alla hade lite erfarenhet med det, men också för att vi alla kände att vi ville lära oss mer om hur det funkar i ett stort projekt där flera kan arbeta på olika delar av projektet samtidigt.

Scrum är en teknik att utföra olika typer av projekt som inte någon utav oss arbetat med innan. Detta var första gången vi fick implementera det in i ett projekt och arbeta med det. Det vi som ett team främst lärt oss av Scrum är hur vi implementerar det in i arbetet för att hjälpa oss, hur alla veckornas Sprints underlättar för oss att dela upp arbetet och komma fram till vad som ska göras denna veckan samt alla dagliga Scrum-möten där vi diskuterar våra KPI:er och ser till så att alla är med på vad som ska göras under dagen.

-[B]

Ett utav målen vi hade från första början av projektets skapande var att använda oss utav JavaFx samt någon typ av databas. Eftersom ingen gruppmedlem tidigare arbetat så mycket med JavaFX kände vi att det skulle bli en rolig utmaning att bygga applikationen med hjälp av JavaFX.
Vi har alla tidigare läst en kurs om databaser, men aldrig riktigt implementerat det i ett annat egengjort projekt, så även det tyckte vi skulle vara en rolig utmaning att implementera.
GitHub är ett verktyg som bland annat är till för att kunna arbeta med projekt av denna kaliber, dock kan det vara lite otydligt för nya användare vilket var vårt fall. GitHub är bra för att dela upp projektet i olika branches och tillåta gruppen att arbeta med olika delar samtidigt, utan några konflikter i koden. Vi hade några problem med detta, då våra branches var otydliga i vissa lägen och istället för att åtgärda problemet med brancherna, laddade vi upp koden till huvud-branchen (“master”) och fixade problemet själva genom att skriva om koden, då egentligen Github hade kunnat lösa problemet

-[A -> B]

Genom att ha en bättre överblick av GitHub och inte testa sig fram är det lättare att få till arbetet med sitt repository att faktiskt underlätta arbetet och inte skapa onödiga komplikationer. Det finns många hjälpmedel och videor online som kan hjälpa till med att lära sig och förstå GitHub.

1.3 Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

-[A]

Från början av projektet när vi först skulle komma på våra user stories till vårt scope hade vi lite problem med vad just user stories är. Med tanke på att inte någon i projektgruppen hade arbetat med Scrum eller user stories överlag, behövde vi läsa på om just ämnet innan vi kunde börja. Detta resulterade i att vi fick feedback om våra stories från vår handledare, att de inte var riktigt ordentliga. Så vi fick lite tips och hjälp om hur vi skulle skriva om våra stories, vilket ledde till mer utförliga user stories med ordentliga acceptance criteria. Detta underlättade uppdelning av arbetet mellan gruppmedlemmarna, vilket vi tidigare haft lite problem med. Något vi inte gjorde ordentligt va att dela ut "poäng" för varje task, vilket hade underlättat i att få en inblick i hur mycket effort som krävdes för våra tasks och vad man borde fokuserat på.
User storiesen har som sagt främst hjälp oss med att dela upp arbetet, så att man vet vad som ska göras och vad som i nuläget utförs av någon annan medlem.

Vi var som tidigare sagt osäkra på hur user stories skulle se ut för att hjälpa oss med vårt arbete, så vi visste inte riktigt hur våra stories skulle se ut förens vi fick feedback från vår handledare och även examinatorn. Feedbacken vi fick var en översiktlig förklaring om vad user stories är och hur de ska hjälpa oss att underlätta arbetet. User stories behöver ha med en eller flera acceptance criterias som förklarar mer när user storiesen är klar, så att man har något att sträva efter och man ska få en bättre överblick om vad som ska göras. För att hjälpa till med att gå tillväga finns även tasks som behöver vara med.
Ett annat problem vi hade som vi märkte är att våra stories var för breda. De täckte alldeles för mycket för att kunna "trycka" ihop till några kriterier och det resulterade i att storiesen blir otydliga och man inte riktigt vet vad som behövs göras. Så mindre user stories är att föredra för att underlätta arbetet med storien och förståelsen för den.
Se bilaga 1, 2 och 3 för graferna.

-[A -> B]

För att förbättra sina stories krävs det att hela gruppen bryter ner de ofärdiga storiesen och frågar sig själva vad gruppen faktiskt vill ha som resultat när den storiesen är färdig. Genom att bryta ner den är det lättare att gå igenom stegen som

krävs för att nå målet, komma fram till den huvudsakliga uppgiften som storiesen ger. Om man först kommer fram till vad man vill få gjort, är det betydligt lättare att komma på sin/sina acceptance criteria/s som definierar storien. Har man bra och tydliga kriterier är det mycket lättare att komma fram till tasks, som hjälpmedel till att nå målet.
Även att ge poäng till alla tasks är ett viktigt tillägg som underlättar uppdelningen av arbetet för alla tasks och vilka som är viktiga att få färdigt först.

1.4 Your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

-[A]

För att se till så att vårt projekt håller sina krav har vi använt oss utav vår Definition of Done, där vi har några krav på vad det innefattar när en user story är klar. Vår DoD har 3 huvudsakliga punkter och en punkt innefattar en checklista. DoD lyder:

1. Koden har testats och fungerar som user storyn lyder.
2. Minst 2 personer har inspekterat koden genom att gå igenom en checklista med 5 punkter:
    2.1. Inga onödiga debugging rader
    2.2. Ingen gammal kod som ligger kvar som kommentarer
    2.3. Samma typ av kodstil
    2.4. Inga errors
    2.5. Alla varningar är inspekterade, så få som möjligt eller inga alls
3. Branchen är mergad till master-branchen på GitHub

Med den här DoD var det lättare för oss att "bocka av" user stories som var klara och kunde därefter fortsätta med projektets andra stories.
Vår Definition of Done ligger uppe på vår Trello-board, så att alla medlemmar kan enkelt och snabbt ta upp den när de ska kolla igenom en annan medlems kod eller merge.

-[B]

Trots våra godkännandekrav har det ibland blivit problem med koden, då man till exempel tagit bort eller skrivit om en del som samarbetar med andra klasser eller metoder den nuvarande user storyn inte täcker. Det har i vårt fall lösts sig hyfsat bra, då detta problem aldrig uppstått på en stor skala för oss, men det hade kunnat ge stora problem om projektet hade varit större. Detta innebär att en större

utvärderingsprocess hade underlättat arbetet då det hade varit lättare att se buggar och andra fel som uppkommit efter arbetet med koden.

-[A -> B]
För att underlätta testningen kunde den gruppmedlemmen som kodar den aktuella storyn skriver kommentarer i koden direkt, och inte efteråt, där han eller hon förklarar hur man bäst testar koden och vad koden gör. Detta hade underlättat för de andra gruppmedlemmarna, då de hade fått en större överblick och förståelse för vad just den kodbiten gör.
Vi hade även kunnat ha en slags automatiserad testning för vår databas. Istället för att självmant mata in data och läsa av det i applikationen, hade vi kunnat ha ett slags test som automatiskt matar in relevanta värden för oss att undersöka.

1.5 The three KPIs you use for monitoring your progress and how you use them to improve your process

-[A]
De tre kpi:er vi använde oss av i detta projekt var: Stressnivån för varje medlem, kommunikation inom gruppen samt hur vi låg till enligt vår projektplan. I vårt fall fyllde varje medlem i gruppen i dessa kpi:er enskilt och oberoende från andra i gruppen. Vi använde oss av ett nummersystem där vi för varje individuell kpi gav ett värde av 1-5 där ett låg nummer till exempelvis indikerar en låg stressnivå, sämre kommunikationen mellan gruppen samt att vi låg bakom den tänkta planen medans ett högre värde representerar en högre stressnivå, en bra kommunikation mellan gruppen samt att vi låg bra till enligt vår ideala plan. Vi har dock inte använt oss så mycket av dessa kpi:er och reflekterat över dem som vi optimalt hade kunnat. Det blev mer bara en syssla att fylla i varje vecka istället för att faktiskt vara ett hjälpmedel för oss att förbättra vårt projekt.

-[B]

Det vi hade velat att kpi:erna skulle ha gjort var att vara ett större hjälpmedel till oss som vi kunde använt oss för att förbättra vår process under projektets gång.
En till KPI som mäts i början och slutet av veckan som reflekterar över om man nått de mål man satt upp hade varit bra. Det hade kunnat hjälpa gruppen att diskutera om ifall vår arbetsbelastning är tillräcklig eller om vi måste arbeta på lite mer för att nå våra mål.

-[A -> B]

Vi tror att vi kunde ha kunnat använt kpi:erna mer optimalt om vi möjligtvis hade valt andra kpi:er. Vad vår grupp har insett nu efter vi har slutfört projektet var att de kpi:er vi valde var inte så användbara för oss framförallt eftersom vi hade en bra kommunikation mellan oss alla med hjälp av discord och andra sociala medier. Vårt öppna kommunikations arbetssätt var tillräckligt för att vi skulle kunna arbeta framåt. Vi spekulerar att i andra möjligtvis större projekt där kommunikationen är mindre öppen skulle kpi:erna varit mer användbara.

2 Social Contract and Effort

2.1 Your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project

-[A]
I början av projektet skrev vi ett social contract som sedan följde med oss genom hela projektet. Den uppdaterades aldrig och var generiskt skrivet eftersom vi inte pratat ihop oss tillräckligt. Det gjorde att kontraktet blev för generiskt och att det inte var lätt att följa eftersom det var väldigt luddigt. Det i slutändan innebar att social contract i princip inte gjorde något eller i alla fall inte gjorde vad det var tänkt att göra eftersom vi i gruppen följde reglerna per automatik utan att behöva göra något. Vi har annars fungerat väldigt bra som grupp och haft daglig kommunikation, även utanför scrum-möten vilket underlättat jämfört med om gruppen inte hade kommit överens, då hade ett social contract verkligen gjort skillnad.

-[B]
Det vi tar med oss till framtida projekt är att uppdatera kontraktet under projektets gång. Dels ändras förutsättningarna och dels ändras planer och mål under projektets gång. Det är därför social contract också behöver uppdateras, för att reflektera det som förändras. Att uppdatera det gör även att man läser igenom det igen, och påminner sig själv om vad som gäller.

I framtiden är det också bra att ha med en punkt om hur man ska hantera situationen om andra kurser tar väldigt mycket tid. Det var en situation vi hamnade i och det hade varit bra om vi tillsammans kommit överens om hur vi hanterar det.

-[A -> B]
Till att börja med måste gruppen tillsammans gå igenom saker som vad de har för ambitionsnivå, hur mycket energi och tid de vill lägga på projektet, konfliktlösning och annat. Sedan när man pratat om det så kan gruppen tillsammans skriva ett social contract.

Vad gäller uppdatering av social contract så kan man exempelvis gå igenom det på ett bestämt möte varannan vecka eller vid något annat jämna mellanrum. Alternativt kan man tilldela en roll inom gruppen till någon som blir ansvarig för social contract och som ser till att gruppen tillsammans uppdaterar dokumentet med jämna mellanrum. En anna fördel med att någon är ansvarig för social contract är att denna person kan ha extra koll på att det efterföljs. Något som man då bör ha med i social contract är när och hur dokumentet ska uppdateras för att vara relevant.

2.2 The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

-[A]

Vi i gruppen kom överens i början av projektets gång att vi inte kände att det var behövligt att specificera en tid, utan att det var viktigare för oss att få gjort det vi bestämt varje sprint. Vi alla känner varandra såpass väl att vi är medvetna om att ingen inte kommer göra något, utan att alla kommer att respektera varandra och ta den tid som behöver för projektet.

-[B]

Vårt sätt att planera den tid vi vill lägga på projektet funkar endast i sådana fall där man känner varandra. Det är ingen hållbar lösning ifall man arbetar med personer man inte är bekanta med, så som sagt så är detta ingen ideal lösning.
Man borde ha satt upp ett schema så att man kan distribuera tiden till projektet baserat på vilka medlemmar som kan arbeta mer respektive mindre under veckorna, eftersom alla har andra kurser parallellt med projektkursen.

-[A -> B]

I början av alla sprintar borde man lagt upp ett slags schema på veckan där alla kan diskutera och komma överens om vad alla personer ska ha gjort och hur mycket tid man ska lägga ner. Sedan behöver man inte alltid följa sin tidsplanering till punkt och pricka ifall andra förhinder uppstår, men det är bra att ha som grund så man vet på ett ungefär. Banana. Milkshake. Potatoes. Adding words to make it look more impressive, hope you won't read this and realize that we actually aren't saying anything useful but rather we are just typing meaningless words for the fun of it. How funny our life can become once we stop thinking about money and physical objects and rather just focus on the spiritual side of things.

3 Design decisions and product structure

3.1 How your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

-[A]

JavaFX är en mjukvaruplattform som vi valde eftersom det erbjuder robusta funktioner när det kommer till användargränssnitt samt kopplingen mellan det och resten av applikationen. Då de flesta av oss är väldigt bekväma med Java blev det ett självklart val att använda JavaFX, och med hjälp av Scenebuilder som utvecklingsverktyg blev arbetet mer smidigt. På så sätt kunde vi uppnå en någorlunda attraktiv och modern design på applikationen.

-[B]

Med facit i hand så borde vi använt en något nyare och mer utbrett sätt att utveckla gränssnitt, exempelvis React eller Vue. Det hade nog tillåtit oss att implementera mer visuell feedback när användaren gör saker i programmet, förslagsvis mer animationer. För att ge användaren ännu en feedback så hade vi kanske även kunna implementerat ljudeffekter av olika slag.

-[A -> B]

Om vi innan projektets start hade forskat mer i olika sätt att bygga gränssnitt så hade det kunnat bidra till att vi får ett mer moderniserat program. Vi kanske även borde tagit inspiration från andra sidor för att plocka ut de komponenter som hade funkat bra med vårt projekt. Istället för att direkt bestämma sig för att använda sig utav JavaFX hade man kunnat testa olika sätt att bygga applikationen, för att hitta det som passade oss bäst.

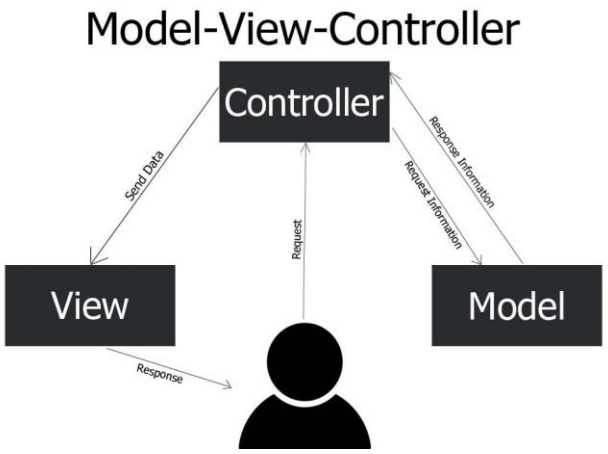
3.2 Which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

-[A]

Vår applikation är skrivet enlig Model-View-Controller modellen eller MVC förkortat. Det innebär att man delar upp programmet i tydliga delar, där modellen är datan, controllern är hur man modellerar datan och viewern är hur datan visualiseras för användaren.

Vi använde oss inte av några klassdiagram eller ER-diagram för databasen vilket gör att vi framför allt i koden har onödigt hög koppling på många ställen. Vilket hade kunnat visualiserats av ett klassdiagram.

Koden är dokumenterad med javadoc för varje public method. Så att det ska vara lätt att se vad metodens syfte är och hur den är tänkt att användas. Vi har dock inte genererat någon javadoc-fil.



https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

-[B]
Med tydligare klassdiagram så hade vi kunnat visualisera skrivit bättre kod. Ett problem med att inte ha det är att man inte har någon lätt överblick över programmet och måste vara väldigt insatt i varje detalj för att förstå hur saker interagerar eller bör interagera med varandra. Detta har lett till onödigt hög koppling som hade kunnat undvikas.

Ett tydligt ER-diagram för databasen var inte för tillfället nödvändigt. Men i takt med att vi hade utökat programmet så hade det varit nödvändigt för att visualisera databasen.

-[A -> B]
För att undvika de problem som uppstått hade vi i ett tidigt stadium innan vi börjat koda på en ny klass eller ny del av databasen kunnat göra lagt till den nya klassen eller delen av databasen till vårt klassdiagram eller ER-diagram. Om vi hade arbetat på det här sättet istället hade vi tidigt kunnat haft en utstuderad plan hur man vill att framförallt java-klasserna ska interagera med varandra. Istället för att komma på lösningar on the fly.

3.3 How you use and update your documentation throughout the sprints

-[A]

Vi har inte direkt använt oss utav någon utomstående dokumentation under projektet. Vi har dock kommenterat koden vi skrivit med kommentaren gällande vad just den delen gör samt variabler och liknande. Klasserna är kommenterade med JavaDoc då tanken var att vi skulle gjort ett JavaDoc med beskrivningar och ladda upp till vårt repository.

-[B]
Med en tydligare dokumentation av till exempel klassdiagram som nämnts innan hade vi som grupp haft bättre översikt vilket hade underlättat i projektet.
Att starta projektet kräver att man behöver välja till några bibliotek som körs med programmet, så att skriva en beskrivning hur man gör det underlättar mycket för utomstående användare som ska starta programmet.

-[A -> B]
Ett sätt att lösa detta på är vi efter varje sprint hade kunnat uppdatera vårt klassdiagram med hur vårt program ser ut, och på så sätt haft en rullande dokumentation av projektet som även hade varit up to date med hur programmet ser ut för tillfället.
Skrivit hur man kör programmet i README-filen på gruppens repository.

3.4 How you ensure code quality and enforce coding standards

-[A]

Det främsta sättet som vi har försökt att säkerställa kodkvalitet var att ha med i vår Definition of Done att koden skulle bli ordentligt testad. Inte bara av personen som skrivit koden utan också av två till tre andra gruppmedlemmar som skulle inspektera koden med en checklista. Checklistan innehåller att ingen onödig kod (debugging kod eller gammal utkommenderade kod) ligger kvar, samma kod stil användes, inga fel uppkom och alla varningar är inspekterade.

Kod standard är inget som vi har lagt ner mycket tid på att kontrollera, detta eftersom vi märkte att det inte fanns något större behov av det. Vi alla i gruppen skriver kod på ett tydligt och liknande sätt så att en specifik standardisering inte direkt har behövts.

-[B]

Ett problem som vi insåg under tiden vi jobbade med projektet var att våran Definition of Done kanske inte var tillräcklig. Detta på grund av att det som testades var specifikt nyskriven kod och inte alltid hur väl den fungerar med resten av programmet och hur lätt den är att bygga vidare på. Ny kod testades med andra ord inte alltid hur självständig den var utan oftast endast om det fanns några konkreta fel vid implementeringen.

-[A -> B]

Detta hade kunnat lösas genom att utöka vår Definition of Done med fler punkter i checklistan. Det skulle till exempel kunna finnas en punkt som säger att koden ska inspekteras ur ett perspektiv om hur självständig den är. Detta skulle också kunna lösas genom bättre definierade user stories som tvingar fram kod att bli självständig.

4 Application of Scrum

4.1 The roles you have used within the team and their impact on your work

-[A]
När vi påbörjade projektet tänkte vi inte på att man kan ha roller inom projektet. Det var först när vår handledare påpekade detta efter en vecka eller två som vi funderade på det. Då hade dock så lång tid gått att vi kände att det inte behövdes och valde därför att fortsätta som vi gjort, utan några roller. Alla blev helt enkelt ansvariga för allting. Detta ledde till att ingen hade speciellt bra koll på någonting och gjorde det svårare för oss att under projektets gång vara så effektiva som vi hade önskat att vi kunde vara.

-[B]
Till nästa gång kommer vi dela ut olika roller till olika personer vid projektets start. Några roller som starkt bör övervägas att ha med är scrum master och product owner. Utöver dessa borde vi funderat över att ha med någon roll som håller koll på våra KPI:er och möjligen en roll för att se över så att koden håller den standard vi satt upp.

Som nämndes ovan under social contract är även detta en roll som bör tas med så att det faktiskt efterföljs och uppdateras under projektets gång. Mycket av våra problem bottnar i att vi planerade dåligt och därför tog saker som de kom istället för att i början, innan vi hade trasslat in oss tog oss tiden att planera ordentligt. Då hade det varit enklare för oss att se vilka roller som kan behövas också.

-[A -> B]
Under första eller andra mötet kan man göra en rollfördelning så att alla har koll på vad som ska göras och så att det alltid finns någon som har koll på det rollen innefattar. Detta ska helst ske när hela gruppen är närvarande alternativt alla utan om person så får den personen ta den rollen som blir över.

För att lösa planeringen borde man besluta inom gruppen att ingen eller väldigt lite kod skrivs under de första dagarna, eller kanske till och med första veckan. Istället ska gruppens fokus ligga på att tänka över hur man tror att projektet kommer utveckla sig, vad som kommer krävas av oss och hur vi bäst hanterar det vi tror kommer framöver.

4.2 The agile practices you have used and their impact on your work

-[A]
Under projektets gång har Hope använt sig av scrum möten, en scrumtavla och user stories för att få en överblick av hur projektet går, samt planera hur arbetet ska gå tillväga för att nå alla uppsatta mål och deadlines. Vi satte upp från början att vi skulle hålla ett möte varje dag vid 12.00 för att diskutera vad som skulle göras och vem som skulle göra det. Detta blev även ett tillfälle att be om hjälp om det var någonting man undrade. Klockslaget 12 valdes då det inte krockade med de olika kurser som alla medlemmar läste.

-[B]

Vår scrum-board kunde även haft ett bättre upplägg än vad som blev. Vi skulle ha haft en bättre user story hantering, där man lägger till vilka personer som arbetat med vilka user stories, och hålla alla sprinter uppdaterade.

Som tidigare nämnt hade vi problem med våra user stories i början, då vi inte förstod riktigt vad våra user stories skulle hjälpa oss med. Med korrekta stories hade vi haft ett korrekt arbetsupplägg direkt och på så vis sluppit strukturera upp det senare.

-[A -> B]

Värdera alla user stories och tasks är ett bra och enkelt sätt att hjälpa till med användandet av stories då prioriteringar hjälper projektet framåt och vad som är det viktigaste att få gjort.
Istället för att köra vidare utan några bra user stories hade man kunnat ta hjälp av vår handledare innan man började arbeta med projektet. På så sätt hade vi direkt haft koll på hur upplägget ska vara i ett agilt projekt, och därför sparat tid.

4.3 The sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

-[A]
I slutet av varje sprint pratade vi lite om vad vi hade gjort den veckan men mest fokus låg på planeringen av nästa sprint. Vi gick igenom vad som behövdes göras den kommande veckan och diskuterade vad vi skulle ta upp på handledningsmötet i början av veckan. När vi planerade nästa sprint gick vi också igenom vad som skulle ha högst prioritet.

Vi hade ingen product owner utan hela gruppen var delaktig i våra sprint reviews. Vi gick igenom vilka user stories som var klara och vilka som fortfarande var under utveckling och behövde fortsätta utvecklas under kommande sprint. Vi satte högst prioritet för de user stories med funktioner vi kände var viktigast för att komma så nära vår idé av applikationen som möjligt och få med de fundamentala funktionerna av vår applikation. Vi gick inte igenom DoD specifikt för varje user story under vår review utan det var underförstått att de som gjorde klart en user story gick igenom DoD.

-[B]
Inför nästa projekt kommer vi att använda oss av en product owner. Det hade hjälpt för att få till mer strukturerade och effektiva reviews. En product owner hade kunnat ha en bättre översikt av projektet och se vilka user stories som var viktigast för att uppnå vårt mål om hur vår färdiga applikation skulle se ut och då sköta prioriteringen av user stories på ett bättre sätt. PO:n kunde också ha gått igenom alla färdiga user stories under våra sprint reviews för att se till så att alla kriterier är uppfyllda och att definition av done har följts. Vårt arbete hade blivit effektivare och vi hade kunnat komma närmare vår bild av hur den färdiga applikationen skulle se ut.

-[A -> B]
I framtiden är det rekommenderat att ha en product owner till projektet. Det hjälper till med planeringen av projektet, bland annat prioriteringar av tasks och user stories, håller i sprintarna och ser till att det som sagts hålls, hjälper till med reflektionerna genom att tillsammans med gruppen gå igenom sprintens händelser från ett annat perspektiv än endast gruppen själva.

4.4 Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

-[A]

De nya verktygen och teknikerna vi har använt har främst varit SceneBuilder, Git och en scrum board. Att lära oss att använda SceneBuilder var ganska enkelt, en person i gruppen hade redan använt det och kunde dela med sig av sin erfarenhet och hjälpa resterande gruppmedlemmar att förstå verktyget. Att lära sig att använda Git och en scrum board var däremot svårare. För att få en bättre förståelse för Git så testade vi tillsammans olika funktioner som att skapa en branch och merge:a med master och vi diskuterade tillsammans för att alla skulle få en god uppfattning om hur det funkade.

För att förstå bättre hur vi skulle använda scrum board:en var det mest diskussion om det som sagts på föreläsningarna.

-[B]

Ett bättre sätt hade kunnat varit att en person tar på sig att lära sig hur då till exempel Git fungerar för att senare lära resten av gruppen. Detta hade kunnat spara tid som nu lades på att hela gruppen testar sig fram och ingen hade heller någon riktig expertis. Vi märkte att det fungerade bättre att lära sig använda Scene Builder eftersom det då fanns en person som kunde visa hur det användes. Detta gjorde så att minimalt med tid lades på att förstå och lära sig verktyget och istället kunde läggas på att använda verktyget.

-[A -> B]

Till framtida projekt kommer vi att ta med oss vetskapen om hur fördelaktigt det är att delegera ut uppgiften att lära sig ett nytt verktyg till en person istället för att ha det som ett grupparbete.
Det är smart att fråga handledaren om relevanta källor för att hjälpa gruppen att komma igång, med tanke på att handledaren förmodligen gjort ett liknande projekt om man tänker på agile-aspekterna och repositories.

4.5 Relation to literature and guest lectures (how do your reflections relate to what others have to say?)

-[A]
Nedan följer en lista över källor som använts under projektet för att leta information eller för att lära oss implementera något i koden. Utöver de hemsidor som är länkade nedan har även google använts flitigt för att hitta information. I början var det främst information om hur agilt arbetssätt och scrum fungerade men även hur vi skulle komma igång med Git och Github.

Under de omständigheter som varit har kursen inte haft någon gästföreläsare när vi läst den. Vi har därför inte kunnat få någon information den vägen.

Hemsidor som information hämtats från:

https://www.jetbrains.com/help/idea/discover-intellij-idea.html

https://docs.oracle.com/en/java/javase/14/docs/api/index.html

https://guides.github.com/

https://git-scm.com/docs

https://stackoverflow.com/questions

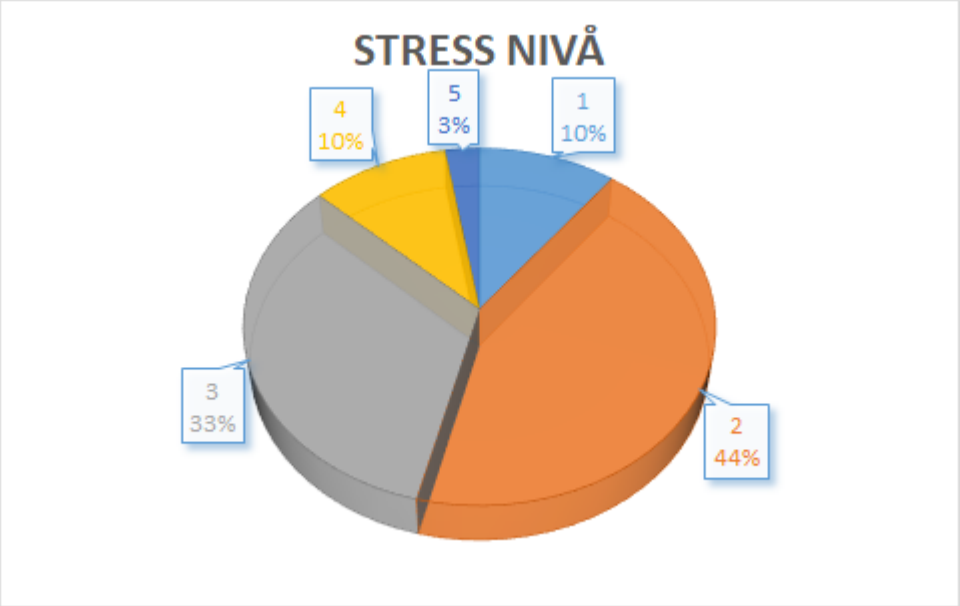
-[B]
Till nästa projekt vill vi tidigt sammanställa vad vi behöver ta reda på eller vill veta mer om som ligger nära i tiden. Det var lätt att börja söka information om hur vi skulle komma igång med Scenebuilder och Javafx när vi inte var helt införstådda med hur versionshantering fungerade, eller ännu värre hur vi skulle jobba agilt och hur scrum fungerade. Detta kan sedan vara en process som man fortsätter utvärdera under projektets gång så att man söker relevant information till det man ska göra härnäst, inte det som kommer efteråt.

-[A -> B]
I rapporten har vi hela tiden tryck på planering och det är även vad vi kommer göra här på sista punkten. Till nästa gång ska vi planera bättre så att vi har koll på vad som ska göras härnäst. En fallgrop har varit att försöka göra för mycket parallellt, speciellt i början innan man riktigt kommit igång med projektet. När några höll på att söka information om JavaFX och Scenebuilder hade vi inte ens fått igång Git än.

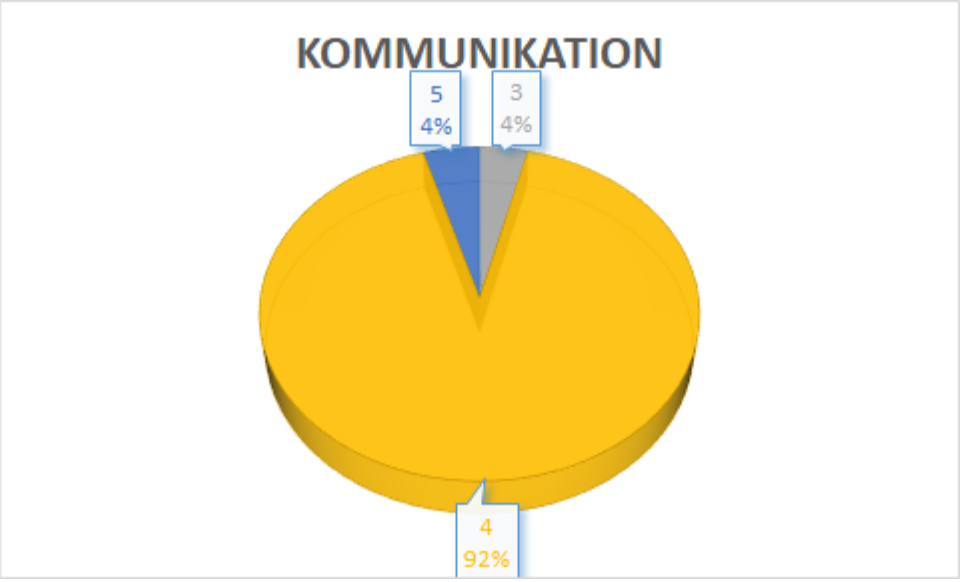
Vi borde också utnyttja utomstående kunskap bättre. Även om det inte var någon gästföreläsning i år så hade vi en handledare som inte bara gått kursen tidigare utan hade handlett tidigare. Honom borde vi ha frågat mer om sådant vi var osäkra på.

5 Bilagor

Bilaga 1



Bilaga 2:



Bilaga 3: