

Vad är en bra projektmetod för små IT-projekt?

Ett försök att besvara frågan görs i kursen II1302 ”Projekt och projektmetoder” vid KTH EECS

Erik Holm^{#1}, Max Körlinge^{#2}, Nicole Othman^{#3}, Hampus Pukitis Furhoff^{#4}, Kim Säther^{#5}

<sup>#EECS, Royal Institute of Technology
Electrum Kistagången 16, Stockholm, Sweden</sup>

¹erih@kth.se

²korlinge@kth.se

³nothman@kth.se

⁴hfurhoff@kth.se

⁵ksather@kth.se

Abstract — Kurs Projekt och projektmetoder II1302 - KTH EECS. Den här studien hade i syfte att efterforska, reflektera och slutligen svara på frågan ”Vad är en bra projektmetod för små IT-projekt?” En kvalitativ forskningsmetod valdes, mer specifikt en fallstudie. Ett redan konfigurerat förslag på projektmetod blev undersöknings ansats. Den tilldelade utgångspunkten var Scrum. Därefter utvärderades projektmetoden tillsammans med projektets arbetsmetod iterativt, enligt den valda agila arbetsmetoden. Kursdeltagare delade upp i grupper om cirka fem deltagare. Var och en fick välja en av fem fördefinierade arbetsroller. Varje projektmedlem fick under kursens gång rapportera och utvärdera sin roll med hjälp av erfarenheter samt litteratur. Utöver några obligatoriska uppgifter var det upp till grupperna att prova på, alternativt läsa om andra projektmetoder än Scrum för att en jämförelse skulle vara möjlig. Det erhållna resultatet tar upp tydliga fördelar med den agila projektmetodiken som t.ex. Scrum/Kanban, men även vilka nackdelar metodiken medför, följt av en diskussion. Delar som diskuteras är bland annat de iterativa sprintarna och agila projektmetoders generella inverkan på ett projekt sett från en projektledares, arkitekts, kund- och kravansvarigs, utvecklingsansvarigs samt testansvarigs perspektiv. (Erik Holm)

Keywords - Scrum, Agile, Kanban, Iterative, Project

I. OM DETTA DOKUMENT OCH UNDERSÖKNING (ERIK HOLM)

Studien riktar sig till likväld studenter som IT-ingenjörer på arbetsmarknaden. Studien utgår från ett mindre IT-projekt vilket främst kan appliceras på startups och mindre företag. Efter en kort introduktion och förtäligande av relevanta begrepp går studien igenom en litteratur- och förstudie samt undersökningsmetod. Avslutningsvis sker en genomgång av projektets genomförande och resultat samt en diskussion kring studien och dess resultat, samtliga från ett rollperspektiv såväl som ett grupperspektiv.

Bilagor:

1. Projektdefinition (Kim Säther)
2. Kravspecifikation (Erik Holm)
3. Arkitekturbeskrivning (Max Körlinge)
4. Design- och utvecklingsplan (Hampus Pukitis Furhoff)

5. Testplan (Nicole Othman)
6. Teknisk dokumentation: Frontend (Kim Säther)
7. Teknisk dokumentation: Backend (Nicole Othman)
8. Teknisk dokumentation: Externals Controller (Hampus Pukitis Furhoff)
9. Teknisk dokumentation: External Integration (Max Körlinge)
10. Teknisk dokumentation: Arduino (Erik Holm)

II. INTRODUKTION

A. Bakgrund (Erik Holm)

Projekt och projektmetoder II1302 vid Kungliga Tekniska Högskolan ämnar ge kursdeltagare en inblick i hur projekt organiseras ute i arbetslivet. Kursen utgår från en teknisk konstruktionsuppgift, varpå studenten ska kunna jämföra och kritiskt analysera metoder som används i ett modernt IT-projekt. Sammanfattat är kursens mål att ge studenten en stabil bas för ett effektivt deltagande och ledning av IT-projekt på den befintliga och framtidia arbetsmarknaden.

Enda sen år 1968, då uttrycket ”software engineering” först introducerades, har IT-sektorn kritiseras för att inte vara tillförlitliga, kostsamma och levererade med problematiska förseningar. Sommerville ser två anledningar till varför IT-projekt fortfarande kantas av otillräckliga resultat (sid. 4 Software Engineering). För det första den ökade efterfrågan. Ingenjörer inom IT-sektorn förväntas bygga allt större och komplexa system som tidigare ansågs vara omöjliga. Därför är de befintliga ingenjörsmetoderna utdaterade. Nya metoder behöver arbetas fram på nytt och ur redan befintliga metoder, för att vara relevanta på dagens arbetsmarknad. Den andra delen av problemet är marknadens alltför låga förväntningar. Problemet här är inte att ingenjörsmetoder är utdaterade utan att de inte används överhuvudtaget. Följaktligen skadas projektets struktur vilket leder till den redan diskuterade bilden av IT-projekt som kostsamma och tidskrävande.

Den här studien har i syfte att besvara frågan ”Vad är en bra projektmetod för små IT-projekt?”. På så sätt ska studenten kunna få en uppfattning om, men även vara med att

påverka bilden av IT-projekt samt hur ett modernt IT-projekt utformas.

B. Problemformulering (Erik Holm)

Det är viktigt att ha en gemensam uppfattning om huvudfrågans innehörd innan en studie påbörjas. Så vad är då ”en bra projektmetod”? Enligt Sommerville ska ett projekt vara effektivt sett till resurser, leverera en produkt i tid och utan defekter (Sommerville, 2005, sid. 5). Själva målet sett till slutresultat är tydligt, men vad krävs under projektets gång för att uppnå det önskade resultatet?

En viktig del av ett projekt är möjligheten till fullständig insyn i arbetsflödet. Det krävs för att undvika missförstånd med en tänkt kund. På pappret är ett missförstånd nästan alltid utvecklarens fel och det leder till onödiga tid- och arbetskraftsresurser för att redigera felet. Det krävs även en fullständig överblick över arbetsflödet. På så sätt kan man snabbt lokalisera flaskhalsar och onödigt/duplicerat arbete (Kniberg and Skarin, 2010, sid. 10). Projektmetoden ska därför vara strukturerad på så sätt att missförstånd undviks mellan kund och utvecklare, samt mellan utvecklare i teamet. Den ska ta vara på projektgruppens resurser på bästa sätt, effektivitet utan att det påverkar kvalitén och strukturen på produkten och projektet.

C. Undersökningsstrategi/lösningsstrategi (Erik Holm)

En kvalitativ forskningsmetod valdes, mer specifikt en fallstudie. Där man i grupp om fem genomförde ett mindre IT-projekt med redan utvalda projektmetoder. En kvalitativ forskningsmetod är att föredra sett till den givna tidsresursen eftersom kvalitativa forskningsmetoden lämpar sig bäst när man ska pröva en befintlig teori. (Association for Information Systems, 2012). Det finns inte tid att successivt samla in data och göra en kvantitativ analys. Istället måste vi använda oss av specifika fall i en fallstudie för att få en detaljerad förståelse av enskildheter (Esaiasson et al., 2002, sid 9). Varje gruppmedlem analyserar och värderar sen sitt tilldelade ansvarsområde för att gemensamt finna ett svar på studiens frågeställning.

D. Relaterade arbeten (Erik Holm)

Projektledning och motivation I IT-projekt : Betydelse av projektmetod och teamroll

Bargot, Lisa

Karlstads Universitet/Handelshögskolan 2017

Agil Projektmetodik : En studie av den agila metodiken och Scrums inverkan på IT-projekt

Landström, O; Odervall, O

Blekinge Tekniska Högskola/Sektionen för planering och mediedesign, 2012

Faktorer som anses vara kritiska vid genomförandet av ett IT-projekt.

Mekonnen, Aida

Uppsala Universitet/Företagsekonomiska institutionen, 2016

E. Avgränsningar (Erik Holm)

Fallstudien använder sig främst av agila projektmetoder där men ett tillvägagångssätt i linje med Scrum är centralt för studien.

Läsaren bör även vara medveten om att den här fallstudien koncentrerar sig på mindre IT-projekt med begränsade arbetskraftsresurser. En projektmetods effektivitet och lämplighet är i högsta grad beroende av projektgruppens resurser. Därför ska en applicering av studiens resultat på ett större projekt, ske med aktsamhet.

III. TEORI OCH INGENJÖRSPRAXIS

Detta kapitel listar och i viss mån beskriver teorier och ingenjörspraxis som används i undersökningen. I det första delkapitlet, Litteraturstudie, beskrivs och görs en källkritisk studie av litteraturen som har används för undersökningen. I det andra, Förstudie, beskrivs vilken förstudie som har gjorts för att förbereda undersökningen.

F. Litteraturstudie (Max Köringe)

Detta kapitel listar, beskriver, och källkritiskt utvärderar, litteraturen som har används för undersökningen.

De övergripande källorna för undersökningen har främst varit litteratur kring agila projektmetoder av varierande vetenskaplig kvalité.

Bland de mer traditionellt vetenskapliga källorna sprungna ur akademien finns Sommerville 2011, Eklund 2010, OMG 2013, Azizyan et al. 2011, Katter 2015, Cobb 2015 och Elvsæter et al. 2012 & 2013. Dessa behandlar i huvudsak agila projektmetoder som generellt fenomen och jämför dem med vattenfallsmetoder, oftast med slutsatsen att agila projektmetoder är bättre. Den vetenskapliga litteraturen föreslår inte några helhetslösningar för agila projekt, så att säga från starten till slutet av ett projekt, och utredet heller inte specifika agila projektmetoder såsom Scrum, Kanban, XP, etc. Ett undantag är möjlichen litteraturen om Essence Framework (OMG 2013) som gör ett försök att vetenskapligen fastställa en specifikation för agila projekt.

När det gäller hur man tillämpar de specifika projektmetoderna som har används under fallstudien så har mer icke-vetenskaplig, praktisk, litteratur används. Litteraturen om Scrum (Kniberg 2015) och Kanban (Kniberg & Skarin 2010) är skriven mer som en beskrivning och personlig reflektion över hur en person använt sig av metoderna inom sitt egna företag, snarare än att man har gjort en vetenskaplig undersökning. Även om denna litteratur saknar vetenskaplig förankring har dess praktiska och lättläggbara karaktär varit till hjälp under fallstudien, då gruppen genomförde sitt IT-projekt.

Eftersom gruppen har använt sig av specifikationen Essence (OMG 2013) och medlemmarna därmed antagit olika roller så har de olika gruppmedlemmarna använt sig också av egen litteratur för att fullgöra sina uppgifter och skriva sina formella dokument som ingår i denna rapport. Alla ansvarsområden har inom detta uppdrag använt sig av

relevanta kapitel i Sommerville 2011, vilket är studentlitteratur om mjukvaruingenjörsskap.

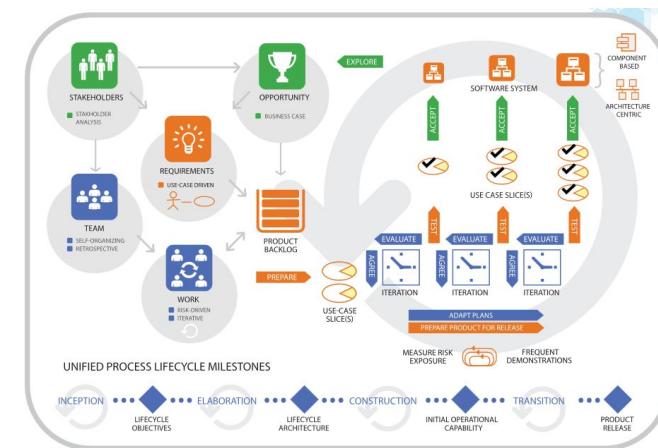
Mest övrig litteratur har det funnits på arkitektur- och testområdena. På arkitektursidan har metoden 4+1 använts, som läggs ut i Kruchten 1995. Där finns det också en specifikation från IEEE (ISO/IEC 2011) som har använts. För arkitekturenhetvecklingen har det alltså funnits gedigen litterär vetenskaplig grund.

Testansvarige har använt flera källor om hur man bedriver test inom IT-projekt: Eriksson 2004, Koskela 2008, och de Grood & Bruhn-Pedersen 2018, vilka alla är litteratur av mer praktisk än vetenskaplig natur, där man på olika sätt beskriver hur man kan bedriva testning i agila IT-projekt.

Källorna som används verkar vara i linje med hur den nuvarande vetenskapliga situationen ser ut när det gäller agila projekt och projektmetoder. Liknande projekt inom näringslivet utförs ofta utan stark vetenskaplig förankring (Andersson & Ekholm 2002: 1), och som Jacobson et al. (2016) påpekar så saknas det även ofta ordentliga vetenskapliga undersökningar när det gäller specifika projektmetoder. Däremot finns det vetenskaplig litteratur när det gäller agila metoder i allmänhet, och sådana har används i litteraturstudien för denna rapport.

G. Förstudie (Hampus Pukitis Furhoff)

I kursen har det inte funnits så mycket tid för att göra någon större förstudie eller göra en egen första ansats till projektmetod. Därför handlade mycket av dom första veckorna om att sätta sig in i den ansats som läraren har tagit fram och dom arbetssätt som används i den. För att göra det så läste vi en kort bok om hur man jobbar med scrum (Kniberg 2015) samt hade en del seminarier och föreläsningar där vi diskuterade ansatsen och våra olika roller i den. Lärarens ansats bygger till allra största del på Scrum med lite tillägg från traditionella projektmetoder (Fig. 1). Under kursens gång har vi sedan läst på mer om alternativa projektmetoder och testat dessa varteftersom.



Figur 1. Översiktlig modell över ansatsens upplägg.

Nedan följer en lista över viktiga punkter i lärarens ansats och en beskrivning av dom.

- Projektgruppen jobbar iterativt och inkrementellt för att minimera risker och kunna vara flexibla.

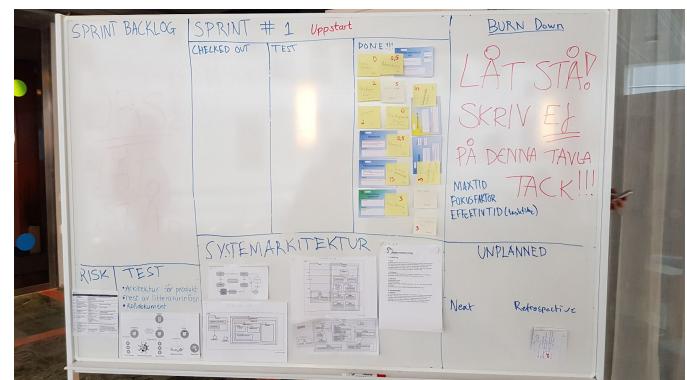
Till detta hör att projektgruppen får en vision om en produkt av en produktägare att jobba med och i kortare tidsperioder (s.k. sprintar) så levereras man olika funktionaliteter av produkten för att så småningom leverera den färdiga slutprodukten. Dessa mindre delar som man levereras bygger på backlog items och är kopplade till dom use cases som hör till en viss vision.

- Projektgruppen använder en arbetstavla som är strukturerad på ett visst sätt (Fig. 2 & Fig. 3).

Detta för att projektet ska bli överskådligt och alla gruppmedlemmar och andra intressenter lätt ska kunna se hur projektet ligger till, vad gruppen jobbar med och vad gruppen kommer göra härnäst.



Figur 2. Arbetstavlans publika sida.



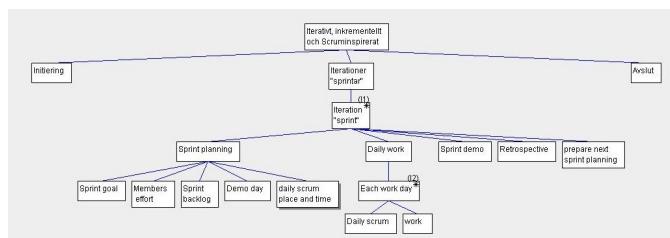
Figur 3. Arbetstavlans interna sida.

- Projektmedlemmarna tar olika roller i gruppen baserat på preferens och erfarenhet.

Dom roller som hade definierats för den här ansatsen var krav- och kundansvarig, arkitekt, konstruktions- och utvecklingsansvarig, testansvarig och projektledare. Detta är inget som ingår i Scrum men som var med i ansatsen eftersom det är vanliga roller i IT-projekt. Dessa roller var även inlagda i ansatsen för att vi skulle lära oss deras funktion antingen genom att praktisera dom själva eller iakta hur någon annan i gruppen tolkade och praktiserade den rollen.

- Projektgruppen har ett visst arbetsflöde med ett par specifika händelser och möten (Fig. 4).

Detta arbetsflöde bygger på det som i Scrum kallas för sprintar där dessa specifika händelserna sker innan, under och efter varje sprint. En sprint inleds med ett sprintplaneringsmöte där projektgruppen planerar vad dom ska jobba med under själva sprinten baserat på hur mycket tid som finns tillgängligt. Under sprinten har gruppen dagliga scrummöten för att alla ska veta hur gruppen ligger till och om det har uppstått några problem. I slutet av sprinten så levereras det gruppen har jobbat med under sprinten och gruppen håller i en kort demo för intressenter. Innan nästa sprint påbörjas har även projektgruppen ett retrospekt-möte för att diskutera hur sprinten har gått och om man bör ändra på något inför nästa sprint.



Figur 4. Schematisk bild över arbetsflödet.

IV. UNDERSÖKNINGSMETODER

Detta kapitel beskriver vilka metoder som används i undersökningen. Metoderna är valda och specificerade så att de skall kunna ge svar på ett antal följdfrågor som identifierats i denna undersökning. Först anges frågorna och sedan följer metodbeskrivning.

H. Frågor att besvara i undersökningen

1. Hur skall man bedöma/redovisa om en delprojektmetod eller praktik är bra?
2. Hur kan man kategorisera, välja, och namnge projektmetoder, projektpraktiker, och verklighetsbeskrivning så att diskussionen om dito blir begreppsmässigt konsistent för ingenjörer inom IT-området (s.k. ontologi)?

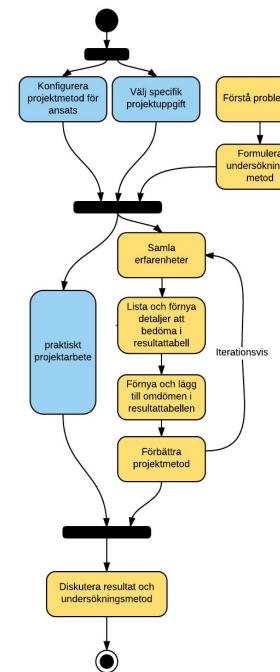
3. Vilka ansvarsroller skall användas som ansats i projektet?
4. Vad består ett projekt av och vilka metoder/praxis skall användas, undersökas och bedömas? Vilken ansats skall göras?

I. Metodbeskrivning (Nicole Othman)

Den centrala metoden i undersökningen bestod av att avgöra huruvida de olika valda projektmetoderna och arbetssättet var bra samt om de bidrog till att göra hela projektprocessen bra. Då erfarenheten i projektgruppen var liten så blev metoden för att samla data induktiv istället för deduktiv som ju annars är den klassiska vetenskapliga metoden där man, utifrån en given referensram (teori eller modell) formulerar hypoteser som sedan testas mot verkligheten.

Metod 1: Undersökningsmetod (Fig. 5). De gula fälten är aktiviteter som kopplar till själva undersökningen. Metoden följer principer för vetenskaplighet enligt Andersson och Ekholm (Andersson & Ekholm 2002: pp 17).

Genom det induktiva arbetssättet så samlade gruppen löpande på sig information under projektets gång, analyserade och drog därefter slutsatser baserade på erfarenheterna. Gruppen arbetade iterativt med fallstudien och samlade på sig resultat genom att testa olika metoder under varje iteration, med resultatet kunde gruppen därefter under retrospektiv-mötena utvärdera huruvida projektmetodiken fungerade bra respektive mindre bra.

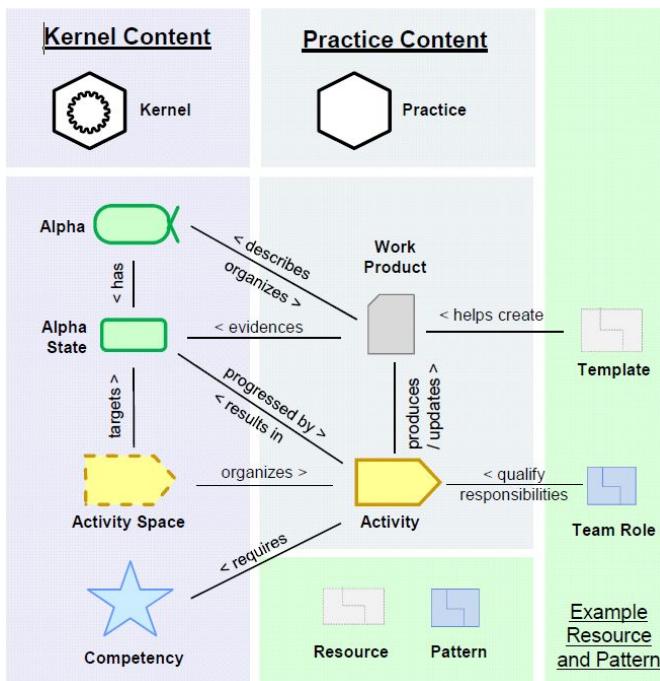


Figur 5. Undersökningsmetod för "Vad är bra projektmetod för små IT-projekt".

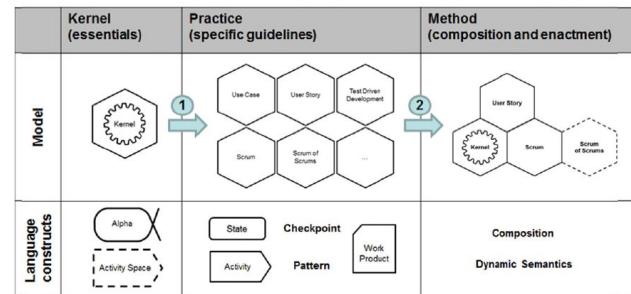
Metod 2: Begrepp/nomenklatur. Ontologi inom IT-området syftar till föremål, processer och begrepp som finns inom ett givet kunskapsområde och relationerna mellan dem. IT-system ska oftast hantera företeelser i den yttre världen och detta sätter krav på utvecklarna som skall definiera företeelserna, namnge dem, ange dess egenskaper och relationerna de har till varandra. I syfte att få en begreppsmässigt konsistent diskussion kring kategorisering, projektmetodiksvälj och dyligt så har projektgruppen genomfört en omfattande litteraturstudie där bland annat Scrum, Kanban och Xtreme Programming (XP) varit ett av flera huvudämnen. I samband med litteraturstudien har projektmedlemmarna tagit sig an quizar för att ytterligare försäkra sig om att litteraturstudien lett till ökad kunskap både vad gäller begreppsmässigt och arbetsmässigt relaterade ting.

De begrepp som används följer om möjligt OMGs standard *Essence - Kernel and Language for Software Engineering Methods Version 1.0* (OMG, 2013).

Följande bilder listar illustrativt centrala begrepp. I denna artikel kommer de engelska begreppen att fritt översättas till svenska då risken för missförstånd anses liten.



Figur 6. Begrepp (Elvesæter, Benguria, & Ilieva, 2013).



Figur 7. Practice och Methods (Elvesæter, Striewe, McNeile, & Berre, 2012).

Metod 3: Ansvarsroller. De ansvarsroller som används som ansats i projektet utgår från specifikationen Essence (OMG 2013). Genom att ha läst igenom rollernas olika innehörd och dess relaterade uppgifter, funderat och därefter valt ut två önskade roller per gruppmedlem har gruppen kunnat bilda en projektgrupp med följande nyckelroller: Projektledare, Kund- och kravansvarig, Arkitekt, Konstruktions- och utvecklingsansvarig och slutligen Testansvarig. Vad rollerna utfört under projektets gång samt hur deras respektive arbetssätt sett ut finns att läsa i mer detalj under V. Genomförande.

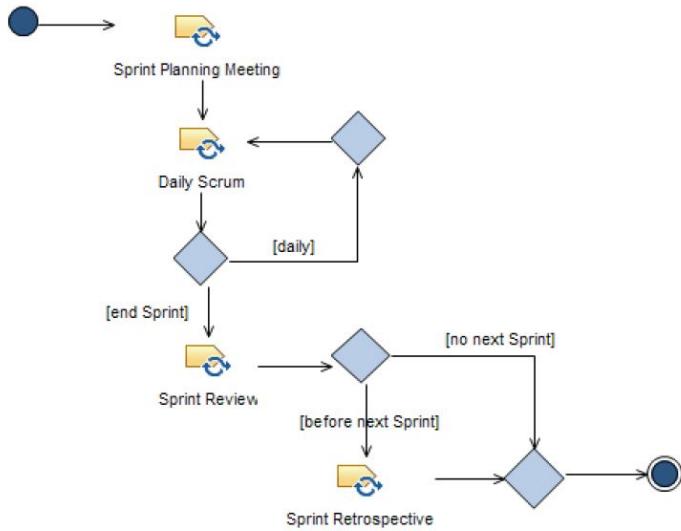
Metod 4: Genom deltagande på lärarledda föreläsningar samt, för var och en projektmedlem, aktivt deltagande på respektive rollmöten har gruppen kunnat ta del av diverse olika projektmetoder och praktiker som sedan kunnat testas under projektets gång.

V. GENOMFÖRANDE

I följande kapitel redovisas viktiga beslut, förändringar och anpassningar som gjorts i projektmetod, projektpraktiker, värderingar, beslut med mera som gjorts under studiens genomförande. Genomförandet beskrivs av vardera formell roll nedan.

J. Projektledning (Kim Säther)

I projektet har vi jobbat iterativt med delleveranser i varje iteration. Projektmetoden som valdes till genomförandet skulle utgå ifrån Scrum, men sedan modifieras utifrån våra erfarenheter av vad som funkar med inslag från andra projektmetoder. Genomförandet av iterationer följer mallen för hur en ”sprint” i Scrum fungerar och den beskrivs enklast med följande aktivitetsdiagram (Fig. 8).



Figur 8. Iteration/Sprint(Elvesæter et al., 2013)

Varje sprint inleddes med ett sprintplaneringsmöte, vid det mötet beslutades vilket mål vi skulle ha för den sprinten och efter det valde vi tillsammans vilka stories/Use case som vi skulle jobba med i denna sprint. När det var beslutat så började vi gemensamt i teamet att skapa våra tasks, samt att vi delade upp tasksen utifrån våra tekniska roller. Uppdelningen av tasksen tog vi bort till iteration 3 för att prova jobba mera enligt Scrum.

På kommande möten vi hade i teamet fortsatte vi med att ha ett dagligt Scrum mötet. Dessa möten är nog dem vi har utvecklat mest under projektets gång. I sprint 1 började vi med att ha ganska långa dagliga scrummöten där vi gick igenom alla våra tasks och flyttade dem på våran tavla om vi hade börjat jobba med dem eller testat eller helt färdigställt tasken. Vi pratade även om vi hade stött på några problem som vi kände att vi behövde hjälpas åt med. I iteration 2 insåg vi att våra möten drog ut på tiden och fortsatte därför bara med den delen då vi gick igenom vad vi hade gjort och flyttade tasksen på tavlan. Vi sa också att om vi hade några problem så behövdes inte dem tas upp på mötet utan kunde tas upp senare och eventuellt endast med någon i teamet. Detta gjorde vi för att kunna jobba effektivare. På detta sätt jobbade vi vidare även i iteration 3. Men i iteration 4 insåg vi att vi ville testa hålla mötet på ett tredje sätt. I den iterationen började vi med att fråga varje teammedlem vad dem har gjort sedan sist och om dem har stött på några problem, vi gick med andra ord inte igenom alla tasks, bara dem vi hade jobbat med. I iteration 4 införde vi även ett max antal för hur många tasks som fick ligga under pågående/checked out på våran scrumboard för att testa det arbetssättet som är kanban inspirerat.

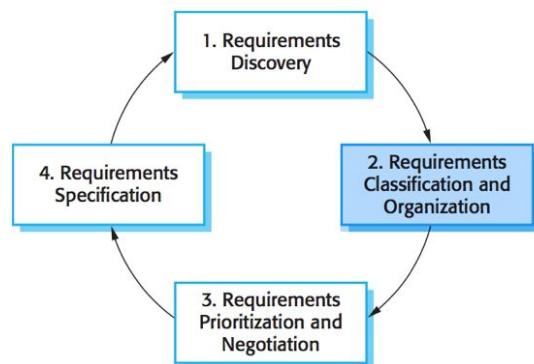
När sprinten hade nått sin sista dag så genomförde vi Retrospect mötet. Det mötet la vi upp på ett sådant sätt att var och en i teamet brainstormade på kring frågeställningen i metoden keep – problems – try under 5 minuter, sedan fick

var och en i teamet presentera vad dem hade kommit på och förklara hur dem hade tänkt. När det var gjort så beslutade vi gemensamt om vad vi ville prova och vad vi ville låta bli att göra till nästa sprint.

Utöver detta så har jag som projektledare även utvärderat projektets status enligt ”Alpha State Cards” i Essence mot slutet av varje iteration från iteration 2 och framåt. Detta gjorde jag på egen hand men tog vid behov hjälp av teamet.

K. Kund- och kravansvarig (Erik Holm)

Projektet utgick till en början från ett antal iterationer enligt Scrums ”sprints”. Det fanns tydliga delmoment som en kravansvarig såg över under iterationerna. Varje steg kommer sammanfattas för att sedan diskuteras i ”Resultatdiskussion”. Som kravansvarig var det under första sprinten som man var i kontakt med kund för första gången. Det var även här som produktkraven började formas. Man kan se det som en cirkel, illustrerad nedanför i Figur 9:



Figur 9. Iterativ kravhantering.

Krav från stories till Case

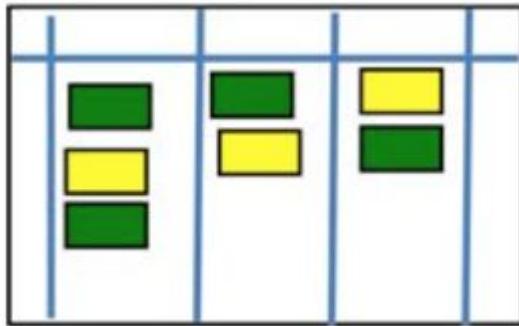
Kundens önskemål tolkades iterativt och skrevs ner som en lista med stories i punktform. Nästa steg var att klassificera och organisera kraven. Från kundens stories kunde man sedan ta fram så kallade use cases. Use cases är en metod för att upptäcka krav eller organisera krav som först introducerades i ”Objectory method”. Det beror helt på om man använder en Top Down eller Bottom Up approach (Jacobson 2005). I den här studien användes först en Bottom Up approach, där kundens stories grupperades tillsammans och formade Use Case för vidare utveckling längre fram. En Top Down kunde därefter användas för att tänka ut några alternativa flöden till use case main flow. Ur de alternativa flödena kunde man formulera sekundära use cases med lägre prioritet.

Parallella projekt

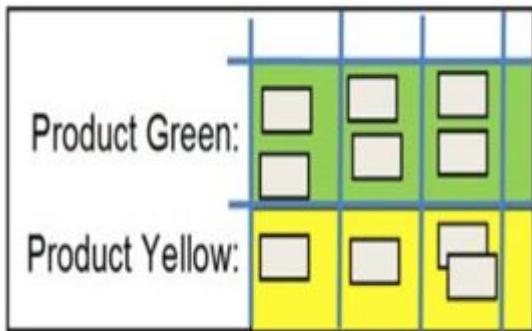
Som kund- och kravansvarig hade man två projekt att ta hänsyn till. Dels projektet i sig, men även rapporten och utvärderingen som var betygsgrundande. Man fick se gruppens Product Backlog som en Team Backlog (Kniberg &

Skarin 2010). Produkterna lades ihop i en och samma sprint, vilket tvingade gruppen att prioritera mellan de båda.

En strategi hade varit att fokusera på en produkt per sprint, den andra var att man arbetade parallellt med de båda produkterna. Eftersom det löpande kom delmoment i båda projekten blev det andra alternativet ett självklart val som kravansvarig. I bilden nedanför kan man se två förslag på grupperingar av en Team Backlog, "colored cards" och "swim lanes". Valet kan beskrivas som agilt då det är i linje med både Kanban och Scrum. Skillnaden mellan de båda metoderna visar sig snarare senare vid planering av gruppens backlog. Vår grupp valde att implementera "swim lanes".



Figur 10: Backlog enligt "Colored Cards"

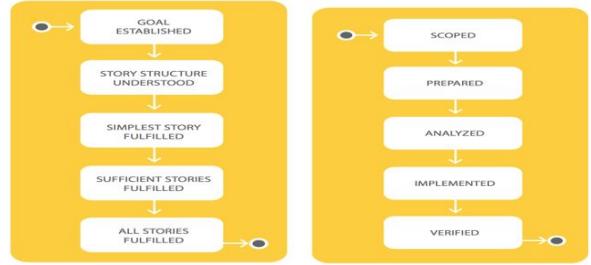


Figur 11: Backlog enligt "Swim Lanes"

Kravspecifikation och User Slices

Use Cases för kundens funktionella krav hade nu valts ut till projektets produkt. Det var viktigt att tydligt modellera upp kopplingarna mellan use case och den story som låg till grund för det valda use caset. Därför hade kravspecifikationen en allt viktigare roll, desto längre tid det gick på projektet. Ändringen i prioritering berodde främst på att man inte alltid hade tidsresursen att bli klar under en sprint. Ett case kunde då brytas ner i så kallade user slices. En user slice hjälpte till att dela upp ett use case i lättanterliga delar (Jacobson 2005). Det kunde till exempel bero på att två olika delar av ett use case behövde implementeras parallellt. Man delade då upp

casen i slices som blev en "tårtbit" av det ursprungliga caset. Teamet kunde därefter implementera delarna av de båda system – delar som hörde ihop i maskineriet, utan att nödvändigtvis ha en uppenbar koppling från användarens perspektiv.



Figur 12: Modell för Use Case och Slice

Varje Case och Slice utvärderades enligt dokumentationen i Figur 12. Punkter "Simplest Story Fulfilled" och "All Stories Fulfilled" krävde att man hade en bra översikt över kopplingen mellan stories och cases/slices i sin kravspecifikation alternativt på Scrum boarden. I Kravspecifikationen skrevs även de icke-funktionella kraven in under kategorier, som till exempel "Prestanda", "Användarvänlighet" och "Säkerhet. Kraven var formulerade efter en natural language specification (Sommerville 2005). Till skillnad från structured specification var den valda specifikationen en struktur som mer liknar vanlig text än kod.

Use Case till Tasks i iterationer

Rollen som kravansvarig fortsatte även när den tekniska delen påbörjades eftersom kraven behövdes ses över och redigeras löpande. Use Cases som hade valt ut för en sprint (iteration) delades upp i tasks (arbetsysslor) för teamet. Tasks skulle vara av rätt storlek för att inte utgöra en alltför stor arbetsbörd sett till den tillgängliga tiden. Varje task fick ett antal story points, beroende på hur lång tid det estimerades ta. Som kravansvarig krävdes en dokumentation av alla tasks för att veta vilka tasks som hörde till vilket use case (implementation) i fall att något gick fel, eller behövde skjutas på. Till en början fick varje gruppmedlem arbeta på ett obegränsat antal tasks. Mot projektperiodens slut valde vi däremot att implementera en av Kanbans restriktioner på arbetsflödet, en så kallad WIP limits (Kniberg & Skarin 2010). Det fanns nu ett maxantal task som man fick påbörja innan de andra var avklarade. En kundansvarig behövde härmed tänka över fördelningen av tasks till ett use case för att undvika flaskhalsar. Avslutningsvis hade gruppen ett Retrospective i slutet av varje sprint där kravansvarig kunde anpassa kraven efter ändrade förutsättningar.

L. Arkitekt (Max Köringe)

Fallstudien använde sig av rollindelning enligt Essence (OMG 2013), där rollen som arkitekt innehåller huvudkompetensen analys. Under första iterationen sattes rollerne och det blev genast uppenbart att det var viktigt att snabbt ta fram en första arkitektur att utgå ifrån för att påbörja

utvecklingen av produkten, vilket också gjordes. Dock slutfördes första versionen enskilt av arkitekten utan använda sig av till exempel brainstorming för att ta hjälp av gruppen, vilket medförde att det mot slutet av iterationen visade sig att det fanns vissa skillnader inom gruppen om hur de tänkt sig att arkitekturen skulle se ut. Utifrån detta anpassades den ursprungliga lösningen till en ny version så att hela gruppen var nöjd till slutet av iterationen.

Arkitekturen byggdes med hjälp av 4+1-metoden (Kruchten 1995) och en IEEE-specifikation angående IT-arkitektur (ISO/IEC 2011), för att använda sig av vetenskapliga metoder vid framställningen. Produkten är uppbyggd av flera hårdvarudelar och mjukvaran var tänkt att köras multitrådat, vilket gjorde att alla vyer enligt denna 4+1-metoden hade en användning. Se Bilaga 3 för närmare beskrivning av arkitekturen.

Eftersom gruppens fallstudie baserades på projektmetoden Scrum avsattes en del av whiteboardtavlan åt arkitekturen, så att teamet enkelt kunde komma åt den. Arkitekturen underhölls sedan under övriga iterationer av analytikern under projektets gång, men inga större förändringar ansågs nödvändiga.

M. Konstruktions- och utvecklingsansvarig (Hampus)

Som konstruktions- och utvecklingsansvarig så var jag tvungen att ha goda kunskaper om programmering och teknologi, kunna designa och refaktorera samt ha ett kritiskt tänkande. Mitt ansvar var att se till att alla i projektgruppen jobbade med utvecklingen av produkten på samma sätt så att vi skulle kunna ha välskriven och levererbar kod i slutet av varje sprint. För att försäkra mig om detta så ställde jag mig själv följande frågor:

- Vad är välskriven kod?
- Hur kan jag som utvecklingsansvarig underlätta för mina gruppmedlemmar i kodskrivandet?
- Hur ska vi leverera vår produkt?
- Hur ska vi dela upp jobbet med att utveckla produkten på ett bra sätt?
- Hur ska vi versionshantera i vårt projekt?

För att besvara dessa frågor så läste jag på mycket om vad andra personer hade för erfarenhet från liknande projekt och försökte se framåt och tänka ”Om vi väljer det här tillvägagångssättet, hur blir det då?” kring olika alternativ gällande bland annat kodspråk, versionshantering och kodsemantik. Jag diskuterade också med gruppmedlemmar om vad dom skulle vilja ha i en designplan som dom skulle vara hjälpta av samt hade brainstormingmöten för att kunna fånga in idéer, förslag och erfarenheter. Vi hade även två seminarier för alla utvecklingsansvariga där vi diskuterade vår roll, vad som förväntades av oss och hur vi hade tolkat vår roll.

När allt detta var gjort kunde jag skriva ner det material jag hade samlat in i det som skulle komma att bli vår designplan. I

denna designplan innehållas sådant som gruppmedlemmarna tyckte var viktigt för dom i utvecklingen, såsom beslut kring kodstandard, språk och framework, versionshantering samt detaljerat klassdiagram baserat på arkitekturen med metoder. Allt detta gjordes i iteration 1 så att vi sedan i iteration 2 skulle kunna dela ut tekniska ansvarsområden och direkt börja utveckla produkten. Resten av iterationerna behövde jag i min roll mest bara uppdatera designplanen om det kom upp några ändringar, framförallt gällde detta metodnamnen i koden.

N. Testansvarig (Nicole Othman)

En del av undersökningen bestod av att testa att arbeta enligt TDD (test driven development) som beskrivs i Ian Sommervilles bok Software Engineering (Sommerville I, 2005). TDD innebär i grund och botten att man arbetar ”förtestat”, det vill säga att man börjar med att identifiera den inkrementella funktionaliteten som skall implementeras vilket normalt sett bör vara ett litet område och således kunna implementeras med några få rader kod. Därefter skriver man ett test för den nya funktionaliteten som tar form av ett automatiserat test och vid en exekvering så får man reda på huruvida testet blev godkänt respektive icke godkänt. Initiat, innan själva källkoden är skriven, så kommer testet att misslyckas och allteftersom utvecklaren lägger till mer källkod så kommer testet att så småningom bli godkänt och vid detta skede kan utvecklaren gå vidare och påbörja implementering utav nästa del.

För projektgruppen innebar detta främst att det första gruppen gjorde inför en ny iteration var att skapa tester som verifierade iterationsmålet och man skrev således testet innan man påbörjade själva kodandet. Som en naturlig del av denna metodik tog testansvarige fram så kallade Test Cases som utformades enligt projektmodellen ”EssUP”, dessa togs fram i samband med sprintplanerings-mötena och baserades på de, för sprinten aktuella, User Case:n. På baksidan av dessa Test Cases formulerades en testspecifikation som täckte nedanstående fem frågor. Dessa frågor tog testansvarige fram efter aktivt deltagande på rollmöte två som ägde rum 2018-04-12:

- Vad som skall testas
- Hur skall det testas
- Testets förväntade resultat
- När testet kan påbörjas

Testspecifikationerna fungerade som ett stöd för projektmedlemmarna då de innehöll information om vad som behövdes testas för att eliminera eventuella risker samt säkerhetsställa att den del av arkitekturen fungerande som den skulle. De var även bra för att säkerställa att gruppen de facto kunde demonstrera att User Case:t uppfyllde produktägarens krav.

Test Case:n placerades med tillhörande User Case på Sprint Backlog:n och miniatyrer av Test Case:n placerades på arkitekturmodellen i syfte att ge en snabb överblick på produktens teststatus. När ett test för ett specifikt Test Case var genomfört kunde testansvarige grönmarkera Test Case i

fråga. På dessa Test Cases fanns även små rutor till vänster som fylldes i allt eftersom kriterierna uppfylldes, på detta sätt kunde projektgruppen utvärdera testernas framsteg.

Av de olika testernas karaktär så skiljde sig testmetodiken något men vad gäller själva kodandet av logiken (det vill säga källkoden) så arbetade projektgruppen främst enligt TDD, resultatet av denna metodik presenteras och diskuteras vidare under kapitel VI. Resultat respektive VII. Diskussion. Att arbeta på detta sätt med testerna var inte en självklar idé i början av projektets gång utan ett arbetssätt som vuxit fram allteftersom projektet fortlöpt. Projektgruppen testade innan framtagandet av ovannämnda testspecifikationsfrågor att skriva utförliga testspecifikationer på varje programmeringsdel som samtliga projektmedlemmar skrivit. Detta visade sig vara tidskrävande och ett icke hållbart arbetssätt vilket resulterade i att testansvarige tog fram och formulerade de fem testspecifikationsfrågorna.

Vad gäller de tester som skrevs i avseende att testa källkoden så arbetade gruppen alltså enligt TDD och samtliga projektmedlemmar skrev sina enhetstester själv. Då projektgruppen använde sig utav Pythons ramverk för testning så kunde gruppen enkelt utföra regressionstester på källkoden allteftersom ny funktionalitet lades in. Regressionstester innebär att man repetitivt utför tester under projektets gång i syfte att försäkra sig om att eventuella defekter som tidigare existerat inte återuppstår allt eftersom mjukvaran utvecklas. Även automatiserade tester (som utfördes genom att i terminalen skriva in kommandot pytest) utav källkoden kunde implementeras tack vare ramverket. På detta sätt har projektgruppen arbetat med testning under projektets gång, för utförligare detaljer kring testmetodiken hänvisas läsaren till Testplanen som testansvarig tagit fram som en del av sitt formella dokumentationsunderlag. Testplanen finns i Bilaga 5.

VI. RESULTAT

Tabellen nedan listar bedömda ”områden” och subjektiv värdering gentemot deras bidrag till Åstadkommer/skapar projektet rätt saker (validitet) och konstrueras lösningar på bästa sätt (reliabilitet)? Alternativa arbetssätt och del-projektmetoder anges också.

TABELLER I
”KEEP – PROBLEMS –TRY”
PROJEKTLEDNING - KIM SÄTHER

Utvärdera Projektstatusen	Det blir tydligt hur projektet ligger till och vilka delar man måste jobba mer med. Det i sin tur gör det även lättare att veta vad man behöver jobba med för att fortsätta utvecklas.	Börja med detta redan i första iterationen så arbetsättet hinner implementeras mera innan projektet är slut. Samt att tydliggöra definitionerna för de olika statusarna.	Essence-specifikation (OMG 2013)
Dagligt Scrum-möte enligt metoden i iteration 4 (beskriven under genomförande)	Detta är enligt mig det bästa sättet att hålla ett dagligt möte eftersom det var både effektivt och alla fick gehör för eventuella problem som dem ville ha hjälp med.	Försöka att inte börja lösa problemen under mötet. Det vore då bättre att delegera ansvaret att hjälpa till att lösa problemet till någon.	-
Scrumboarden	Det blir väldigt visuellt och enkelt att se vad som behöver lösas, samt att ansvaret blir ganska jämnt fördelat mellan team-medlem marna eftersom alla jobbar med något hela tiden.	Ha tavlan digital, så att man kan komma åt den och uppgifterna även om man inte sitter vid den fysiska tavlan. Idag finns det verktyg för det som ex. Jira.	

”Keep”			
”Keep”	Motivation	Förbättringar	Referenser

Max-antal för pågående tasks	Detta är ett bra arbetssätt då man enkelt vet vad man ska fokusera på och inte hoppa mellan flera olika uppgifter. På detta sättet blir jobbet mer kvalitativt eftersom man kan hålla fokus på en sak.	Testa vilket max antal som funkar bäst utifrån vilket arbete man utför och hur stort team man jobbar i.		Avsätta tid för kursinnehåll och uppdatera dokument	Avsätta tid för att gå på föreläsningar och hålla befintliga dokument uppdaterade i varje sprint.	För att undvika att det ska bli svårt att hinna med befintliga tasks utan att jobba övertid och kärra stress över mängden arbete. Samt att uppdateringen av dokumenten ska göras kontinuerligt så att dem fyller ett syfte både för oss i teamet, produktägarna och ev. utomstående intressenter.	-
Arbata riskfokuserat	För att kunna känna sig trygg i att man alltid kommer ha något att leverera efter varje iteration.	Ingen.					

"Problems – Try"			
"Problem"	"Try what?"	Motivering	Referenser
Kommunikation vid gränserna i de tekniska rollerna	Par-programmering eller tydligare definition av vem som gör vad i gemensamma filer.	För att undvika merge-konflikter och därmed dubbelt jobb.	-
Otydlig definition av Story-points eller bestämmelser i överlag	Definiera med en bestämd tidsangivelse. Ex. en dag skulle för vissa kunna betyda en hel arbetsdag 8 timmar men för medarbetare som jobbar halvtid kanske en dag betyder 4 timmar och andra kanske en dag som 24 timmar.	För att undvika missförstånd och olikheter i uppskattning av tid, samt konflikter i teamet.	-

"Problems – Skip"			
"Problem"	"Skip or replace, why"	Motivering	Referenser
Flera projekt ihopslagna till ett	Replace. Ska flera projekt utföras samtidigt så vore det bättre att behandla dem som skilda projekt på olika scrumboards med egna möten.	För att det inte ska bli rörigt med vad som bör jobbas med och hur mycket tid man bör lägga på dem olika projekten.	-
Inläsning av litteratur	Replace. Byt ut till att se filmklipp om ämnet.	Alla lär sig inte lättast av att läsa, speciellt inte när det är dem mängderna text på så kort tid. Vissa tar in lättast av att höra, vissa av att se och andra av att prova. Främja allas olikheter genom att möjliggöra annan typ av inlärning.	-

KUND- OCH KRAVANSVARIG - ERIK HOLM

"Keep"			
"Keep"	Motivation	Förbättringar	Referenser
Story Points	Story Points gav en kravansvarig en kontinuerlig feedback i utformning och ändring av befintliga krav. Det är snabbt en indikator på om ett Case borde delas upp i mindre delar.	Tydlig värdering av Story Points från projektets början. Hur mycket är ett story point värt?	
Rollindelning	Det var ett bra sätt att dela upp ansvaret och för att påbörja ett projekt utan alltför många frågor kring	Med reservation för att arbetsbördan blir alltför stor, visade det sig dock	OMG 2013

		ringen (tekniska delen), vilket kan kompenseras med projektmetoder, till exempel iterationer och standup meetings, som uppmuntrar till dialog.		
Swim Lanes	Swim lanes var ett effektivt sätt att hantera flera parallella projekt. Team backlog är nödvändigt om samma projektgrupp ansvarar för flera projekt	Inga. Utnyttjade arbets tavlan på bästa sätt.	Kniberg & Skarin 2010	under den föregående iterationen. Ur ett studiesyfte är det väldigt önskvärt. Ett flöde enligt Kanban hade krävt mer av en kravansvarig. Ansvarig hade själv behövt kategorisera och utvärdera avklarade tasks för att få en uppfattning om arbetstakt. Det hela beror på kompetensen och erfarenheten i gruppen. Inte bara den egna kompetensen utan även den gemensamma och erfarenheten att arbeta tillsammans. Kanbans val att ha en period kan säker optimera projektet i viss mån när man väljer bort iterativa reflektionsmöten, men med det kommer risker som är beroende av gruppens dynamik och förkunskap. Iterationer gav även kravansvarig en bra översikt över icke-funktionella krav som implementerades. Det kan även bero på den tekniska rollen som kravansvarig har. Läs VIII - Diskussion.
Variation mellan Bottom Up och Top Down Approach	Det var en bra utgångspunkt att börja med att tänka ut olika stories som kategoriseras enligt Bottom Up, för att därefter gå över till en Top Down för att hitta sekundära use cases i det alternativa flödet. Det var lätt att kunden fastnade vid en lösning på problemet när man utgick från ett Use Case redan från början. I bland kunde två skilda use cases vara lösningar på problemet.	Inga. Men en prioritering borde vara att skriva alternative flows till ett use case enligt en Top Down approach om det finns en tidsbrist. Hellre att få klart en modul som löser kundens problem än att ha två som inte har hunnit räkna med/behandla alla alternativa flöden som kan uppstå.	Jacobson 2015	
Iterationer med Burndown Chart	Iterationerna gav kravansvarig tillfälle att tänka över och analysera valda kravprioriteringar och metoder		Kniberg & Skarin 2010	

"Problems – Try"			
"Problem"	"Try what?"	Motivering	Referenser
Risk att formulerin g av user slices blir ett stressmom ent	Planera reservplaner i form av slices redan vid första iterationsmötet, även om gruppen är säkra på att de hinner implementera hela caset.	Finns tid att arbeta fram ett fungerande fall vid eventuell tidsbrist så har man snabbt en lösning tillhands	

		komplettera r de stories som ligger till grund för use cases, men man borde konsekvent utgå från ett av de båda hjälpmedle n inledningsvis i mindre projekt.	
Tid att implementera en lösning planerades inte alltid in bland tasks vilket visade sig i burndown	Ha gemensamma lappar för implementation av olika gruppmedlem mars komponenter	När en komponent delas upp mellan olika gruppmedle mmars krävs det inte bara en estimering av tiden det tar för respektive person att göra sin del, utan även den tid det tar att lägga ihop delarna. Annars blir Burndown chart väldigt missvisande	

"Problems – Try"			
"Problem"	"Try what?"	Motivering	Referenser
Förvirrand e terminolo gi i kravhanter ing	Prova att välja antingen Scrums krav med user stories eller Jacobsons stories till Use Case. Var i alla fall noga med att förtydliga skillnaden i dokumentation.	Båda komplettera r inte nödvändigt vis varandra särskilt bra. Förvirrande terminologi kan bara försvåra kravprocess en. Även kravspecifi kationen blir rörig. En tumregel kan vara att använda use case i större projekt i större behov av dokumentat ion (Model) eller där projektet behöver delas upp i slices. En kombinatio n används ibland på arbetsmark naden där user stories	Jacobson 2015 Kniberg & Skarin 2010

"Problems – Skip"			
"Problem"	"Skip or replace, why"	Motivering	Referenser
En alltför Trevande och vardaglig formulering av de icke-funkt ionella kraven	Replace Natural language specification med constructed specification	Kraven i det här fallet är ämnade för gruppens deltagare. Det är effektivare att kommunicera genom constructed specificatio n när kraven ska implemente	Sommerville I, 2005

		ras som kod. Förkunskaperna får ses över men i ett mindre IT-projekt ska de vara tillräckliga	
WIP limits fungerar inte lika bra i Scrums iterationer som nämndt i "Try"	Replace Scrum Board med Kanban board, i fall av WIP limits. Eller Skip helt.	Det kunde känna onödigt med ytterligare en restriktion utöver Scrums uppdelning av tasks i iterationer. Tänk till exempel att tre tasks är beroende av varandra men WIP limit är satt till två. Ska kravansvarig dela upp en implementering i två delar, men riskera att de är alldeles för stora, eller dela upp det i fler delar och riskera att det uppstår en flaskhals när två är påbörjade och resterande inte får påbörjas på grund av WIP limit? Ett scenario som lätt kan undvikas om man antingen väljer en Kanban approach	Kniberg&Skarin, 2010

Roller gick in i varandra	Replace Rollindelning enligt essence med Scrums roller eller prova att slå ihop två roller.	utan sprintar. Läs Rollindeling i "Keep"	

ARKITEKT - MAX KÖRLINGE

"Keep"			
"Keep"	Motivation	Förbättringar	Referenser
Rollindeling	Tydliga ansvarsområden säkerställer att viktiga delar av projektet färdigställs, och att det är tydligt vem som ska göra vad.	Inga.	Essence-specifikation (OMG 2013)
Arkitektur	Att ha en genomarbetad arkitektur tidigt i projektet gjorde utveckling enkel och strukturerad.	Förankra arkiteturen hos teamet från början.	ISO/IEC 2011 Bilaga 3
Olika arkitekturytyper	Olika typer ger tydlighet både när det gäller fysisk utformning av produkten, hur processer kommunicerar, samt en plan för mjukvaruutvecklingen. Enbart en ensam ritning kan inte göra alla dessa saker.	Tydligare definition av vad den logiska vyn ska åstadkomma.	Kruchten 1995 Bilaga 3
Brainstorming	Brainstorming gjordes efter att en första arkitektur togs fram, och det ledde till att teamet fick vara med och skapa en tydlig bild av vart de skulle med produkten.	Göra brainstormingen redan innan första versionen av arkiteturen skapades.	-

Arkitektur beskrivning	Att ha ett dokument med lite mer utförlig beskrivning av de olika vyerna i text var bra för att tvinga sig själv att kunna förklara ritningarna på ett tydligt sätt, och bra för att referera till för utomstående.	Håll texten mer uppdaterad.	Bilaga 3			använde texten ändå, utan tittade på ritningarna.	
------------------------	---	-----------------------------	----------	--	--	---	--

"Problems – Try"			
"Problem"	"Try what?"	Motivering	Referenser
Enskilt arbete	Brainstorma med teamet innan påbörjad arkitektur	Att försöka göra första versionen av arkitekturen själv gjorde bara att man behövde ändra den sedan. Bättre börja förankra med teamet från början.	-
Sent bestämd kodsemantisk	Ha namnkonventioner etc. färdigt innan påbörjad arkitektur	Att ändra i namn på klasser och liknande i efterhand är onödigt, om man kommit överens om namnkonventioner i förväg kan man skriva rätt från början.	-
Arkitektur beskrivningens text uppdaterad	Sätt av tid i varje iteration till att uppdatera texten	Aven om ritningarna (vyerna) hölls uppdaterade så liggade själva texten efter, troligen pga brist på motivation eftersom ingen	-

"Problems – Skip"			
"Problem"	"Skip or replace, why"	Motivering	Referenser
Logiskt vy otydlig	Skip, pga kan vara användbar för andra typer av projekt.	Den logiska vyn för arkitekturen var, för vårt projekt, mycket lik andra vyer och upplevdes därför som obsolet.	Kruchten 1995
Tidsbrist i iteration 1	Skip, uppstod pga att iteration 1 blev speciell i och med att den innehöll uppstartsdelar, och att man även var tvungen att lära sig hur allting fungerade, vilket man inte behöver göra nästa gång.	Analytikern fick jobba extra mycket i iteration 1 för att ha en färdig arkitektur att utgå ifrån.	-
Dubbelt projekt	Skip, pga speciell situation med kursen som pågick samtidigt som projektet.	Att hålla isär projektet "Prototyp" med projektet "Rapport" gjorde det svårare att applicera och studera projektmetoden på ett verkligt sätt.	-

KONSTRUKTIONS- OCH UTVECKLINGSANSVARIG - HAMPUS PUKITIS FURHOFF

"Keep"			
"Keep"	Motivation	Förbättringar	Referenser
Arbetstavlan	Arbetstavlan var en väldigt bra arbetsplats, oavsett vart den stod eftersom den förde	Man hade kunnat fotat den kontinuerligt, kanske i samband	Kniberg, 2015

	gruppen samman, det var VÅR tavla, och man kunde väldigt snabbt få svar om det var något man undrade kring projektet bara genom att titta upp på den.	med daily scrum eller retrospekt och sprintplanning, för att få spårbarhet och kunna gå tillbaka för att se vad projektstatus va då.			kan göra en designvy som hen tror på. Annars finns en risk att man jobbar mot olika saker där den ena har en bild i huvudet över hur det kommer se ut medan den andra har en annan.														
Design- och utvecklingsplan	Det var en bra ide att samla alla våra gemensamma regler kring koden i en design- och utvecklingsplan eftersom om man blev osäker på något kring utvecklingen kunde man enkelt kolla upp det där. Det hade säkert varit användbart om man hade fått in någon ny medlem för att den snabba skulle kunna sätta sig in i hur vi jobbade.	-	Kursseminarier	Krav på att utveckling sansvarige ska ha övergripande bild	Det var nog väldigt bra att den utvecklingsans variga var tvungen att ha en övergripande bild över produkten eftersom det gjorde att alla kunde fråga mig något om produkten och veta att jag antingen kunde svara direkt på deras fråga eller hitta svaret ganska fort.	-													
Designvyn	Designvyen var till en stor hjälp när vi skulle dela ut tekniskt ansvar och när vi skulle koda eftersom alla kunde se direkt vad som innefattas i dom olika klasserna och hur man skulle använda varandra klasser.	Inte en förbättring men en sak för utvecklings ansvarig att tänka på i samband med designvyen är att kommunicaera mycket med arkitekten så att hen kan komma fram till en arkitektur som utvecklings ansvarige	Lindbäck, 2018	<p style="text-align: center;">"Problems – Try"</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #cccccc;"> <th style="padding: 2px;">"Problem "</th> <th style="padding: 2px;">"Try what?"</th> <th style="padding: 2px;">Motivering</th> <th style="padding: 2px;">Referenser</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">Otydlig gräns mellan arkitekten s och utveckling sansvarige s ansvar</td> <td style="padding: 2px;">Diskutera tidigt vem som gör vad så att alla är på samma sida.</td> <td style="padding: 2px;">Detta var egentligen aldrig något aktuellt problem i just vårat fall men det hade kunnat bli om inte jag och Max hade diskuterat detta tidigt.</td> <td style="padding: 2px;">Eklund, 2010</td> </tr> <tr> <td style="padding: 2px;">När oförutsedd a saker dök upp</td> <td style="padding: 2px;">Överskatta tiden för tasks och underskatta antalet</td> <td style="padding: 2px;">Var alltid väldigt noggrann i planeringen</td> <td style="padding: 2px;">Kniberg, 2015</td> </tr> </tbody> </table>				"Problem "	"Try what?"	Motivering	Referenser	Otydlig gräns mellan arkitekten s och utveckling sansvarige s ansvar	Diskutera tidigt vem som gör vad så att alla är på samma sida.	Detta var egentligen aldrig något aktuellt problem i just vårat fall men det hade kunnat bli om inte jag och Max hade diskuterat detta tidigt.	Eklund, 2010	När oförutsedd a saker dök upp	Överskatta tiden för tasks och underskatta antalet	Var alltid väldigt noggrann i planeringen	Kniberg, 2015
"Problem "	"Try what?"	Motivering	Referenser																
Otydlig gräns mellan arkitekten s och utveckling sansvarige s ansvar	Diskutera tidigt vem som gör vad så att alla är på samma sida.	Detta var egentligen aldrig något aktuellt problem i just vårat fall men det hade kunnat bli om inte jag och Max hade diskuterat detta tidigt.	Eklund, 2010																
När oförutsedd a saker dök upp	Överskatta tiden för tasks och underskatta antalet	Var alltid väldigt noggrann i planeringen	Kniberg, 2015																

blev det snabbt stressigt att få klart innan deadline.	storypoints som kan klaras av under iterationen. Diskutera tasksens grundligt för att försöka upptäcka oförutsedd arbete	och överskatta gärna tiden det tar att göra saker för att kunna ta hand om oförutsedda saker om dom skulle dyka upp	
---	--	--	--

		skärmar på möten och fick på det sättet mycket mer aktiva och fokuserade möten.	
--	--	---	--

TESTANSVARIG - NICOLE OTHMAN

"Problems – Skip"			
"Problem "	"Skip or replace, why"	Motivering	Referenser
Datorer på möten	Tillåt inte datorer på daily scrum, sprint planning och retrospektmöte n eftersom det är väldigt lätt att man tappar fokus då.	Vi märkte att om alla hade sina datorskärm ar uppe så hände det flera gånger att man kanske diskuterade något problem och att någon plötsligt sa "Jag ska bara ..." för att sedan söka efter någon lösning eller göra en snabb fix. Detta stal tid från våra möten och ledde ofta till att övriga gruppmedlemmar blev tvungna att invända personen som fixade/sökt e då det oftast inte gick så snabbt som personen trodde. Till slut gjorde vi så att vi förbjöd öppna	-

”Keep”			
”Keep”	Motivation	Förbättrin gar	Referenser
TDD	Hjälper programmerare n att hålla fokus på en mindre del utav funktionaliteten . Man kan inkrementellt bygga vidare på koden när testet väl passerat istället för en ”happy hacking”-stil.	Introducera TDD i tidigare programme ringskurser.	Sommerville, 2011
Fästa Test Cases på arkitektur beskrivnin g.	Ett lätt sätt att snabbt checka teststatus och få bra överblick.	Sätt referens-ID på de små miniatyrbil derna och referera till Test Casen	Bilaga 5
Ett försök till automatiserade tester.	Bra att få igång automatiserade tester därfor att man då lättare kan göra regressionsteste r efter varje ny införd funktionalitet och på så sätt hitta ev. buggar.	-	Koskela, 2008
Test cases (med testspecifi kation på baksidan) som fästs på User Casen.	Genom att ha ett Use Case kopplat till Test Caset så var det lättare att förstå kraven och utifrån det skriva test	-	Bilaga 5

	specifikationen		
	.		

VII. DISKUSSION

N. Metoddiskussion (Kim Säther)

Det utförda testet har hjälpt mig att kunna svara på frågan ”Vad är en bra projektmetod för små IT-projekt?”. Den metoden vi har använt beskrivs under metodbeskrivning, den är en bra metod enligt vår projektgrupp.

Reliabiliteten på svaret måste dock anses vara ganska låg p.g.a. att svaret kommer ifrån enskilda individers bedömningar och inte mätbar data. Det kan därför vara så att andra inte anser att den metoden som används är bra även om den fungerade bra för den här gruppen.

Och validiteten måste också anses låg eftersom studien var väldigt kort och testet inte har genomförts med någon annan ansats än den valda i detta projektet. Samt att studien endast genomfördes av en liten grupp med individer. För ett mer pålitligt resultat hade det varit bättre att genomföra testet på flera individer. Det skulle därför kunnat vara så att ansatsen inte ledde till en bra projektmetod och då hade man inte kunnat svara på frågan. Det skulle också kunna finnas flera projektmetoder som är bra att använda vid små IT-projekt som inte berördes av denna studie.

O. Resultatdiskussion

Projektledning - Kim Säther

Projektet har haft både bra och mindre bra metoder, men som helhet har projektet fortskrivet med en bra metod. Det har gått bra eftersom vi har lyckats ha något att leverera i varje iteration och slutprodukten uppfyllde dem obligatoriska funktionerna samt en extra önskvärd funktion som vår produktägare ville ha.

I denna diskussion kommer jag försöka besvara frågan om rollen projektledare behövs för att kunna svara på den övergripande frågan för projektet, ”Vad är en bra projektmetod för små IT-projekt?”.

Min roll som projektledare har delvis kännts meningsfull och delvis inte alls meningsfull för projektet. Detta tror jag beror på vilket team man jobbar i och vilka egenskaper dem olika individerna i teamet har. I detta teamet var alla väldigt drivna och ville vara med och planera och fatta beslut vilket gjorde att min roll som projektledare fick mera ansvar i att hejda och leda tillbaka på rätt spår när teamet var för ivriga och ville lösa alla problem på en gång. Jag fick även hjälpa till med dem lite mera känsliga bitarna som att se till att alla fick bli hördas och känna sig delaktiga i teamet.

Jag anser absolut att denna roll behövs, men inte alltid för att ro projektet i hamn utan mera för att se till att projektet blir gjort på ett mer effektivt och mindre kostsamt sätt. Reliabiliteten på svaret måste dock anses vara ganska låg eftersom svaret kommer från en enskild individens bedömning och inte mätbar data. Och validiteten måste också anses vara låg eftersom studien var väldigt kort och testet inte har genomförts med flera olika personer.

”Problems – Try”			
”Problem”	”Try what?”	Motivering	Referenser
Test Cases	Tidigarelägg a rollmötena, gärna innan början av nya iterationer.	Svårt att förstå användningsområdet i början. Vissa Use Cases gick det inte att ha Test Case till.	-
Testrelaterade dokument	Snabbt komma fram till vilka dokument som skall användas. Brainstorma kring dessa och gemensamt komma fram till vad som känns rimligt inom projektets ramar.	Tar onödig tid att fundera och prova olika dokument/mallar, denna tid kan läggas på att t.ex. lära sig att arbeta enligt TDD.	-

”Problems – Skip”			
”Problem”	”Skip or replace, why”	Motivering	Referenser
Test sektion på Sprint backlog	Skip	I gruppens fall så flyttades tasken oftast från ”Checked out” till ”Done” direkt i.o.m. TDD.	-
Testspecifikationer per metod	Skip	Alltför omfattande arbete som inte ger tillräckligt bra utfall som gör att det i längden inte är värt det.	-

Kund- och kravansvarig - Erik Holm

Projektets var en kvalitativ studie, där resultatets validitet och reliabilitet måste ses utifrån studiens förutsättningar i fråga om tid och arbetsresurser. Även om en kvalitativ studie inte är lika beroende av en omfattande insamling av data så begränsas studien av tidsramen, empirisk förankring och studiens författares samfällda utgångspunkt. För ökad tillförlitlighet skrevs egna åsikter ned från början, för att fastställa förutfattade meningar, så kallad perspektivmedvetenhet (Hedin 1996)

Diskussionen utgår från en kund- och kravansvarigs perspektiv för att besvara studiens fråga. Det bör dock tilläggas att rollen är något okonventionell i den mening att ansvarig även har en såpass stor teknisk roll, i en gruppkonstellation utformad i studiesyfte. Ute på arbetsmarknaden och i större projekt kan den rollen bli alltför omfattande. Önskvärda egenskaper hos ett projekt så som den övergripande kommunikationen och kravansvarigs förståelse för projektet behöver därför inte nödvändigtvis bero på projektmetoden utan även den mångsidiga roll som deltagaren har i studiesyfte.

Sammanfattningsvis hjälpte Scrums iterationer med strukturen av kravarbetet. Det var visserligen tidskrävande moment som hade kunnat läggas på utveckling av produkt. Ett projekt enligt Kanban lägger mer tid på den nämnda utvecklingen när man inte arbetar i iterationer med samma prioritering av avstämning. Med den lösare strukturen kommer dock ett större ansvar hos den enskilda utvecklaren. Det är viktigt att alla har erfarenheten och kompetensen av IT-projekt, inte bara generellt utan även i den befintliga projektgruppen. Vad har man för resurser att arbeta med, hur mycket brukar man få gjort och var behöver flest resurser läggas. Man får väga den lösare och mer tidseffektiva Kanban mot Scrums struktur som säkerställer ett fungerande ”maskineri” men som lägger mer tid på arbete som inte är direkt knutet till slutprodukten. Studiens fråga berör mindre IT-projekt och med ett mindre IT-projekt blir det lättare att få en översikt över arbetet, vilket talar för ett mindre behov av hjälpmittel för att få just en översikt över sitt projekt. Med det sagt borde inte Scrum avfärdas som onödigt tidskrävande för mindre projekt. Studien utgjorde ett kursmoment, med deltagare som inte hade någon större förkunskap i ämnet och som inte kände varandra från början och det blev där tydligt hur lätt missförstånd och delade meningar kan uppstå. Det var nödvändigt att stämma av i iterationer för att lära känna gruppen och det blev snabbt ett essentiellt verktyg i IT-projektet. I ”Arbete i Projekt” poängterar Sven Eklund just vikten av att ”lära känna gruppen”. Att man inte enkelsidigt fokuserar på vad som ska göras utan även hur man tillsammans ska göra det, i vad han kallar processen (Eklund 2002). Inledningsvis handlar det främst om att man ska etablera gruppen med de olika individerna. Något som Scrums iterativa upplägg fungerar utmärkt för. Eklund delar även upp projektet i olika faser. Tillhörasfasen följs av rollsökningfasen och därefter samhörighetsfasen. Eklund nämner

problematiken med att många grupper fastnar i rollsökningfasen, vilket till stor del kan bero på missförstånd och brist på dialog. Scrums avstämningar och uppmuntran till dialog svarar på frågor om det egna inflytandet och vilket ansvar man har i gruppen, frågor som uppstår i projektgrupper oberoende av gruppens storlek. I studiens projekt var rollerna fördefinierade, men trots detta uppstod frågor som till exempel var linjen gick mellan två roller eller vem som skulle ha huvudansvar för ett visst område. På en arbetsplats finns inte samma tid för att formulera pedagogiska specifikationer och utförlig rolldokumentation, vilket kommer leda till fler oklara avgränsningar mellan olika roller och deras ansvar. Det stödjer om möjligt än mer vikten av Scrums iterationer.

Avvägningen får helt enkelt ske i varje specifikt fall. På samma sätt som WIP gav varje enskild person en chans att evaluera den egna arbetstakten så är valet av projektmetod en avvägning hur stort behov gruppen har av hjälpmittel för att bibehålla struktur, inte bara sett till gruppens storlek utan även till individernas kompetens samt erfarenhet av projekt och varandra.

Arkitekt - Max Köringe

Resultaten består av både positiva och negativ upplevelser av projektmetoderna. Generellt kan man säga att metoderna som används har gett ett gott intryck och gott resultat, och det är därför det under de positiva delarna ligger större saker såsom rollindelning och arkitekturens olika vyer, medan de negativa bitarna är mer specifika saker som upplevs som att de inte fungerade lika bra.

Utifrån resultatet kan man se att själva arkitekturen upplevdes som positiv och nödvändig för att projektet skulle ros i land. Arkitekturen gav en stark grund att stå på för utvecklingen av produkten från start till slut, vilket leder till slutsatsen att åtminstone utifrån arkitektenperspektiv så var rollindelningen enligt Essence-specifikationen (OMG 2013) en mycket bra projektmetod att använda.

När det gäller att bedöma det vetenskapliga värdet av resultatet krävs en mer utförlig diskussion. Reliabiliteten får anses vara låg i denna studie. Reliabilitet innehåller hur väl som studien mäter det som den mäter (NE 2018), och i och med att resultaten ej mäts med hjälp av något exakt bestämbart, så som till exempel siffror, utan mäts med hjälp av personliga motiveringar, så är resultatet ej pålitligt sett ur detta perspektiv. Man kan också säga att reliabilitet innehåller att resultatet av testet ska bli detsamma vid upprepade mätningar av olika människor, vilket inte stämmer i detta fall då resultatet är subjektivt. Resultatets validitet däremot, det vill säga att det mäter det som det är avsett att mäta (NE 2018), får anses som medelmåttigt. Det kommer faktiskt fram ett visst mått av hur en ”bra” projektmetod ser ut, men måtten är inexacta, tolkande, ord, såsom ”bra” och ”dåligt”, vilket understryker den subjektiva naturen av resultatet.

Både reliabilitet och validitet är termer som används mest i kvantitativ forskning, det vill säga när man använder statistiska och matematiska metoder för att studera stora

mängder data, med en önskan om att uppnå objektiva resultat. Därför är det inte underligt att om man försöker analysera denna studies resultat utifrån reliabilitet och validitet så kan man inte undgå att komma fram till att resultatet har låg pålitlighet, då denna studie varken använt sig av kvantitativ data eller statistik. Utgångspunkten för studien har heller inte tillåtit ett sådant tillvägagångssätt, då kvantitativ data ej varit tillgänglig, utan studien utgår ifrån en enda fallstudie, och deltagarnas upplevelser av denna. Därför kan man dra slutsatsen att studien som görs i denna rapport snarare är kvalitativ än kvantitativ, det vill säga att forskaren samlar in och tolkar data simultant (NE 2018). Även om kvantitativ forskning är vanligast när det gäller naturvetenskaplig och teknisk forskning, så är kvalitativ forskning utbrett inom humaniora och sociala vetenskaper. Då använder man oftare en induktiv metod, till exempel genom en fördjupad analys av en enda fallstudie, vilket också är det som har gjorts i denna studie.

Sett ur ett kvalitatitv perspektiv är dock denna studies resultat tyvärr ej heller särskilt pålitliga, då den kvalitatitva metoden stipulerar en djupgående fallstudie, medan denna studies resultat kommer av endast mycket korta försök när det gäller olika projektmetoder. Detta ger ett undermåligt underlag för tolkning och induktiva slutsatser.

Slutsatsen blir således i slutändan att oavsett om man ser på studiens resultat ur ett kvantitatitv eller kvalitatitv perspektiv så får de anses som ej vetenskapligt pålitliga.

Konstruktions- och utvecklingsansvarig Hampus Pukitis Furhoff

Jag tolkar mitt och dom andra medlemmarnas resultat som så att dom flesta av dom projektmetoder vi testat, inklusive rollfördelningen, har hjälpt oss att leverera en bra produkt på relativt kort tid. Om vi skulle få i uppdrag att genomföra ett ungefärligt stort projekt så skulle vi förmodligen kunna estimera projektets triangelhörn (kostnad-tid-omfattning) i förväg med gott resultat.

Ur ett undersökningsperspektiv kan man fundera på om vi undersöker vi det vi vill undersöka, d.v.s. svarar resultatet på frågeställningen? Det tycker jag det gör, vi får fram många bra förslag i resultatet på vad som ska ingå i en bra projektmetod och vad vi tycker har fungerat för oss.

Däremot skulle jag inte säga att det är något resultat som man ska lita på alls då det helt baseras på våra subjektiva åsikter och om någon annan skulle göra samma undersökning så skulle dom kanske komma fram till helt andra resultat.

Testansvarig - Nicole Othman

Att döma av slutresultatet så hittade projektgruppen till slut ett fungerande sätt att arbeta med testning, de tester som gruppen tagit fram har alla blivit godkända och en slutprodukt har således kunnat framställas. Att testningsarbetet är en vital del utav ett IT-projekt framgår tydligt när man studerar resultatet, enhetstesterna som skrevs och exekverades gav en indikation på huruvida produkten/funktionaliteten fungerade

eller ej. På ett sätt var det ett bra verktyg för att hålla koll på projektets status vad gäller framtagning av produkten. Såsom tidigare nämnt under V. Genomförande så bestod en del utav undersökningen av att testa TDD (test driven development) och den generella uppfattningen var att det stundtals var svårt att tillämpa en TDD-metodik.

Att skriva TDD är svårt därför att man som utvecklare (speciellt juniora utvecklare) kan ha svårt att på förhand veta vad som skall testas och hur det skall testas innan källkoden väl finns på plats. Det är en teknik som kräver tid samt erfarenhet (övning ger färdighet) och med tanke på den korta perioden som projektet var igång så var det stundtals svårt att finna tid till att prioritera och sätta sig in i ett TDD-mässigt arbetsätt. Emellertid, när gruppen väl fått grepp om ett TDD-orienterat arbetsätt så fungerade det smidigt och tack vare en utförlig arkitekturbeskrivning samt detaljerad utvecklingsplan (UML-diagram) kunde gruppen lättare ta sig an utmaningen att skriva tester och källkod enligt TDD.

Med det sagt så skulle man kanske omformulera frågeställningen ”Vad är en bra projektmetod för små IT-projekt?” så att hänsyn tas till projektmedlemmarnas erfarenhet vad gäller programmering och arbetet med att skriva enhetstester för källkod. Finns det avsatt tid till att lära sig TDD på djupet så är TDD bra att tillämpa då det bland annat hjälper programmeraren att hålla fokus på en mindre del av funktionaliteten istället för att riskera att förvirra sig i produkten som helhet. Man kan inkrementellt bygga vidare på koden när testet väl passerat istället för att försöka ge sig på att koda hela projektet vilket troligtvis resulterar i fler buggar som i sin tur leder till att det tar längre tid att ta fram produkten/få färdigt projektet. Har projektgruppen svårt att finna tid till att lära sig TDD på djupet så kanske det inte är ett optimalt sätt att arbeta med testning för ett litet IT-projekt då det är tidskrävande och risken för att TDD orsakar en flaskhals är stor. Beroende på gruppens erfarenhet så kan frågeställningen således besvaras med olika synvinklar där det ena svaret skiljer sig från det andra.

Vidare kan man också diskutera och jämföra hur pass bra testningsarbetet är i ett projekt som tillämpar TDD och Scrum, kontra ett projekt som tillämpar Vattenfallsmodellen och i de fall där tester görs i efterhand. Till skillnad mot TDD, där man skriver testet innan själva implementationen, så påbörjar man i det senare fallet testning vid slutet, när själva utvecklingsarbetet är genomfört. Såsom namnet lyder så löper arbetet i ett Vattenfallsprojekt i en nedåtgående mekanism precis som ett vattenfall. När ett steg i Vattenfallsmodellen är avklarat och nästa steg i utvecklingsprocessen har påbörjats så kan man inte återgå till det tidigare steget för att återutveckla eller utföra eventuella ändringar. Fördelarna med Vattenfallsmodellen är att det bland annat är enkelt att dokumentera i och med att det inte kräver samma underhållning som ett agilt IT-projekt med TDD-tillämpning där man är mer flexibel för förändringar. Det är också kostnadseffektivt om man från början vet exakt hur projektet skall fortlöpa och vilka delar som skall finnas på plats efter

avslutat projekt, utrymme och sannolikheten för förändringar är väldigt liten med andra ord. Sedan är det troligtvis också lättare att sätta sig in i arbetsättet och man som utvecklare inte på förhand behöver fundera på vad som skall testas, i och med att testning sker efter att funktionaliteten/källkoden finns på plats. Det stora problemet med Vattenfallsmodellen är emellertid att det som projektgruppen specificerat under projektet och de krav som tagits fram under den mest övergripande nivån är också de krav som testas och utvärderas vid slutet av utvecklingsmodellen. Detta kan utgöra ett hot för projektets slutresultat därfor att man eventuellt upptäcker de mest övergripande och grundläggande behoven allra sist, vilket leder till att en stor del av arbetet är bortkastat. Här är TDD och Scrum helt klart ett bättre arbetsätt eftersom man i ett tidigt skede kan eliminera eventuella risker som i värsta fall skulle kunna vara föroldande för ett IT-projekt.

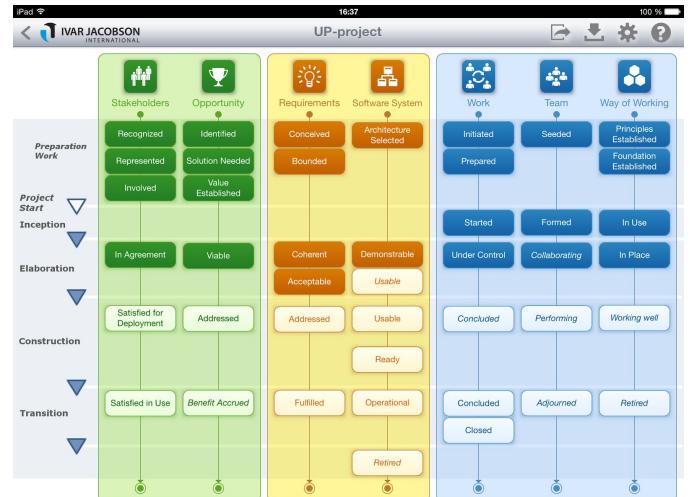
P. Bidrag till vetenskaplighet, ingenjörserfarenhet (Kim Säther)

Vetenskap kan definieras som ”Vetenskapen är den samhälleliga sanktionerade auktoriteten i så kallade sakfrågor. Till det ovetenskapliga räknas mystiska eller religiösa insikter som vilar på övernaturliga fenomen. Vidare håller sig vetenskapen fri från fördömar och forskarens personliga värderingar. Resonemanget leder in på gränsdragningen mellan vetenskap och det som kan kallas sunt förnuft.” enligt (Andersson & Ekholm 2002: sida 13)

Enligt detta skulle man kunna säga att vi har jobbat vetenskapligt i och med metod 1 som innebär att vi empiriskt samlade på oss information under projektets gång och analyserade och drog därefter slutsatser baserade på erfarenheterna.

I det stora hela så har vi inte jobbat speciellt vetenskapligt eftersom detta är baserat på våra åsikter och värderingar. Det är inte heller vetenskapligt då det inte är säkert att om någon annan person gjorde studien så kanske denna inte skulle få samma resultat. Och det finns inte heller någon annan metod för att testa våran metod på (Andersson & Ekholm 2002: sida 16).

Framtida förbättringar skulle möjligen kunna vara att planera bättre från början så att det skulle gå lättare att involvera våra produktägare i processen och därmed hunnit bli klara med fasen elaboration (Fig. 13).



Figur 13. Projektets faser (OMG 2013).

SŁUTORD

Produkten vi konstruerade är ett fjärrstyrт bevattningssystem för blommor. Vi mäter fuktvärdet med en fuktsensor och när fuktvärdet har kommit under en viss gräns så får man en notis via epost som säger att det är dags att vattna sin blomma. Då kan man gå in på vår hemsida och vattna sin blomma. På hemsidan kan man även se aktuellt fuktvärde samt ställa in vid vilket fuktvärde man vill få notisen skickad med epost till sig.

Ytterligare funktioner som produktägarna ville ha men som inte hanns med var:

1. Att kunna ställa in vilken vattenmängd blomman skulle vattnas med.
2. Att vattningen skulle ske automatiskt utan att man behöver gå in på hemsidan och trycka på knappen när fuktvärdet var lågt.
3. Att kunna stoppa bevattningen.
4. Att skapa en mobilapplikation för systemet.

Nu är systemet gjort för att hantera en blomma, men en annan önskan som vi i projektgruppen skulle velat få till var att systemet skulle klara av att hantera flera blommor.

Länken till vårat github repository är: <https://github.com/fongie/projectgaming>

REFERENSER

- Andersson, N., Ekholm, A. (2002). *Vetenskaplighet - Utvärdering av tre implementeringsprojekt inom IT Bygg och Fastighet 2002*. Institutionen för Byggande och Arkitektur, Lund.
- Azizyan, G., Magarian, M., Kajko-Matsson, M. (2011). Survey of Agile Tool Usage and Needs. *10.1109/AGILE.2011.30*.
- Cobb, C.G. (2015). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*. Wiley.
- Eklund, S. (2010). *Arbeta i projekt: individen, gruppen, ledaren*. Studentlitteratur.
- Elvesæter, B., Benguria, G., & Ilieva, S. (2013). *A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods*. Paper presented at the Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering.
- Elvesæter, B., Striewe, M., McNeile, A., & Berre, A.-J. (2012). Towards an Agile Foundation for the Creation and Enactment of Software Engineering Methods: The SEMAT Approach. *Second Workshop on Process-based approaches for Model-Driven Engineering (PMDE 2012)*.
- Eriksson, U. (2004). *Test och kvalitetssäkring av IT-system*. Studentlitteratur.
- ISO/IEC/IEEE 42010. (2011). *Systems and software engineering - Architecture description*.
- de Groot, D., Bruhn-Pedersen, M. (2018). *Testing and Quality in the Scaled Agile Framework for Lean Enterprises*. EuroSTAR eBook
- Hedin, A. (1996) En liten lathund om kvalitativ metod. Retrieved from <https://studentportalen.uu.se/uusp-filearea-tool/download.action?nodeId=459535&toolAttachmentId=108197>.
- Jacobson, I., Spence, I., Seidewitz, E. (2016). Industrial-Scale Agile: From Craft to Engineering. *ACM Queue 2016 Sept/Oct issue*.
- Katter, A. (2015). *Förslag till en ekonomiskt hållbar projektmetod: En fallstudie vid Sydweb*. Examensarbete vid Raseborg, Helsingfors.
- Kniberg, H. (2015). *Scrum and XP From the Trenches: How we do scrum*. InfoQ.
- Kniberg, H., Skarin, M. (2010). *Kanban and Scrum: making the most of both*. InfoQ.
- Koskela, L. (2008). *Test Driven: Practical TDD and Acceptance TDD for Java Developers*. Manning Publications Co.
- Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, 12(6), 42-50.
- Lindbäck, L. (2018). *A First Course in Object Oriented Development A Hands-On Approach*. KTH.
- NE. (2018). Nationalencyklopedin, kvalitativ metod. <http://www.ne.se/uppslagsverk/encyklopedi/lång/kvalitativ-metod> (hämtad 2018-05-15)
- NE. (2018). Nationalencyklopedin, reliabilitet. <http://www.ne.se/uppslagsverk/encyklopedi/lång/reliabilitet> (hämtad 2018-05-15)
- NE. (2018). Nationalencyklopedin, validitet. <http://www.ne.se/uppslagsverk/encyklopedi/lång/validitet> (hämtad 2018-05-15).
- OMG. (2013). Kernel and Language for Software Engineering Methods (Essence). 1.0. Retrieved 7 May, 2015, from <http://www.omg.org/spec/Essence/1.0/>
- Sommerville, I. (2011). *Software Engineering*. 9th ed. Pearson.

FlowerPower

Projektdefinition

Abstract

Detta dokument är en projektdefinition (Eklund, 2010) för ett studentprojekt KTH ICT.

En projektdefinition är inte en projektplan utan föregår ofta en sådan. Projektdefinitionen kan vid behov utvecklas till en projektplan. För examensarbetet är det lämpligt att projektdefinitionen fungera som ”överenskommelse” mellan projektets huvudintressenter vilka oftast är ett företag, studenten som gör arbetet och akademien varifrån studenten kommer. Förändras projektet i något viktigt avseende så uppdateras och förankras projektdefinitionen.

Dokumentversion, senaste överst

Date	Version	Author	Description
08/04/2018	<Version 1.0>	Kim Säther	Framtagning av dokument
26/05/2018	<Version 2.0>	Kim Säther	Påfyllning av det som ej gjordes i version 1.0

Innehållsförteckning

1	Introduktion.....	3
1.1	Dokumentets syfte.....	3
1.2	Dokumentets omfattning	3
1.3	Dokumentöversikt	3
2	Projektöversikt – bakgrund, syfte och mål	4
2.1	Bakgrund	4
2.2	Syfte	4
2.3	Mål	4
2.4	Funktionella krav - användningsfallsmodell	5
3	Organisation	6
3.1	Personer i projektet.....	6
3.2	Möten	6
3.3	Arbetsplats.....	6
3.4	Arbetsutrustning	6
3.5	Meddelanden	6
3.6	Webbplatser.....	6
4	Projektets olika mål.....	7
4.1	Uppgiftsägaren	7
4.2	Kursmål och examensmål	7
4.2.1	Vetenskaplighet.....	7
4.3	Hållbarhetsaspekter	9
4.4	Etik, jämställdhet och likabehandling (JML)	9
4.5	Arbetsmiljöaspekter	10
5	Fas-, tids- och arbetsplan	11
5.1	Arbetsplan och arbetsvolym per projektdeltagare.....	11
5.2	Fas- och tidsplan.....	12
6	Intressenter	13
7	Riskanalys	14
8	Förändringsplan	16
9	Kostnadsplan.....	17
10	Dokumentplan.....	18
11	Utbildningsplan.....	19
12	Rapport- och granskningsplan	20
	Appendix A - Referenser	21

1 Introduktion

1.1 Dokumentets syfte

Dokumentets syfte är att få en ökad förståelse för projektet. Det ska bland annat skapa förståelse för uppgiften, ge en bättre överblick av projektet såväl som de resurser som behövs under projektets gång. Utöver detta har dokumentet som syfte att ge studenterna praktisk erfarenhet och övning i att skriva dokument samt arbeta efter projektprocessen Scrum.

1.2 Dokumentets omfattning

Detta dokument behandlar följande:

- Vad projektet innehåller.
- Hur projektets process går till.
- Varför projektet genomförs.

Detta dokument behandlar *inte* följande:

- Projektets detaljplanering.
- Projektets lösning.
- Projektets presentation.

1.3 Dokumentöversikt

Detta dokument innehåller följande delar:

- **Projekt- eller uppgiftsbeskrivning**
- **Organisation**
- **Projektmål**
- **Fas- och tidsplan**
- **Intressenter**
- **Riskanalys**
- **Förändringsplan**
- **Kostnader**
- **Dokumentplan**
- **Utbildningsplan**
- **Rapport- och granskningsplan**
- **Referenser**

2 Projektöversikt – bakgrund, syfte och mål

2.1 Bakgrund

I dagens stressiga samhälle glöms det oftast bort att vattna de blommor och växter man har i sitt hushåll, detta resulterar i att växterna dör. För att lösa detta problem har projektgruppen som målbild att ta fram ett hjälpmittel för blomvattning. I takt med att vi går mot ett allt mer digitaliserat samhälle där IOT (Internet Of Things) blir mera populärt vill även vi jobba i den riktningen.

2.2 Syfte

Projektets syfte är att få ökad kunskap i hur SCRUM fungerar, öva oss i rapportskriving och kunna hjälpa människor i vardagen genom att ta fram en användbar produkt.

2.3 Mål

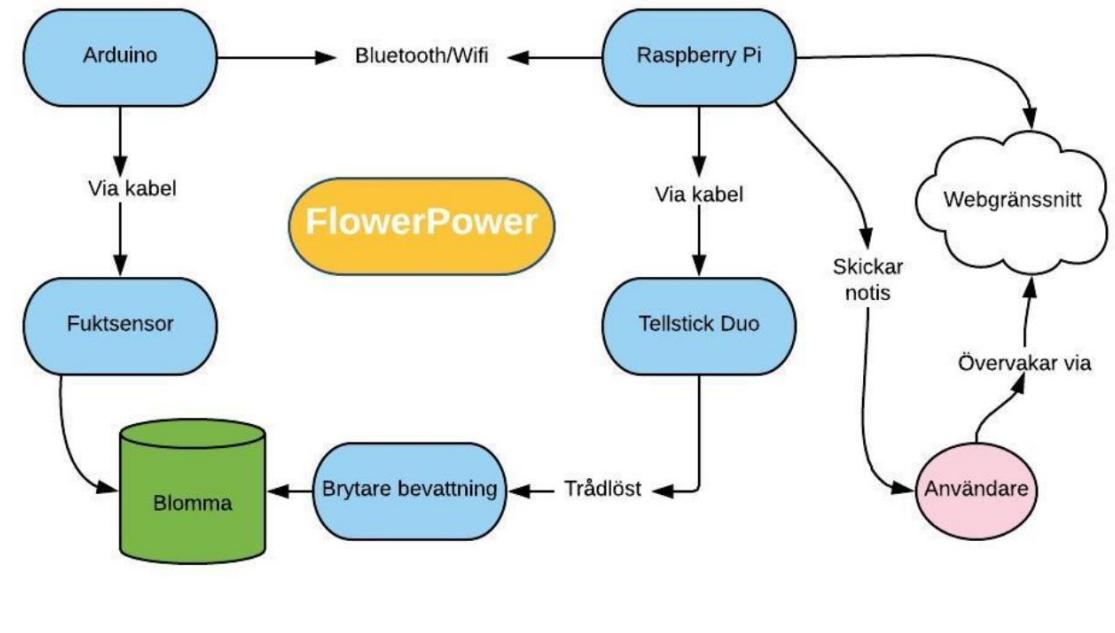
För att uppnå detta ska våran projektgrupp ta fram ett hjälpmittel som ska underlätta blomvattningen i sitt hushåll. Utvecklingen av denna produkt ska ske med metoden SCRUM och i projektets slutfas ska vi skriva en rapport om vårat arbete.

Projektets resultatmål är att skriva en Kursrapport och att ta fram en produkt som vi själva fick välja vad det skulle vara. Vi kan nå detta mål genom att använda en projektmetod vid utvecklingen av våran produkt, det kommer göra utvecklingen lättare och vi kan sedan reflektera över vårat jobb i en kursrapport.

Projektmålet är att lära sig mera om hur man jobbar i ett projekt som mjukvaruutvecklare och bli bättre på rapportskrivning. Båda dessa delar kommer uppfyllas genom att öva vid utvecklingen av vår produkt.

2.4 Funktionella krav - användningsfallsmodell

- När jorden är torr skall användaren få en notis.
- Man skall kunna stoppa bevattningen.
- Man skall kunna slå på bevattningen fjärrstyrts.
- Man skall kunna se fuktstatusen på en hemsida.
- Man skall kunna reglera vattenmängden som blomman vattnas med.
- Blomman skall bli vattnad automatiskt utan användarens signal.
- Man skall kunna reglera känsligheten på fuktsensorn.
- Det skall finnas en mobilapplikation.



3 Organisation

3.1 Personer i projektet

Person	Kontaktinformation och beskrivning (adress, telefon ..)
Erik Holm	erih@kth.se - Kund- och kravansvarig
Hampus Pukitis Furhoff	hfurhoff@kth.se Konstruktions- och utvecklingsansvarig, MHU-ansvarig
Kim Säther	ksather@kth.se Projektledare, Arbetsmiljöansvarig
Max Körlinge	korlinge@kth.se Arkitekt
Nicole Othman	nothman@kth.se Testansvarig

3.2 Mötens

Scrum-möte, Retrospekt och Scrumdemo.

3.3 Arbetsplats

Främst KTH campus Kista, därutöver valfri plats.

3.4 Arbetsutrustning

SCRUM-board, personliga datorer, Raspberry Pi, TellStick Duo, Fjärrströmbrytare (Kompatibel med TellStick Duo), Arduino och Fuktsensor.

3.5 Meddelanden

Om medlemmarna befinner sig på olika platser sker kommunikationen främst genom Facebook Messenger. I denna gruppkonversation sker all typ av kommunikation, där utöver kommunicerar gruppen via e-mail.

3.6 Webbplatser

Webbplatser som kommer användas är kth.se, kth.instructure.com samt en egen hemsida som ska skapas. Webbverktyg som kommer användas är Lucidchart och Messenger.

4 Projektets olika mål

4.1 Uppgiftsägaren

Slutprodukten bör bestå av delprodukter som projektgruppen, vid slutet av projektet, knutit ihop. Detta för att undvika att leverera en inkomplett slutprodukt. Den iterativa utvecklingen bör därför ske genom att man levererar en färdig delprodukt vid varje iterations slut, så att denna går att bruka oavsett om vissa delar av slutprodukten ännu inte är färdig.

4.2 Kursmål och examensmål

Likheter i detta projekt och det självständiga examensarbetets framgång är tydligt bl.a. genom att projektmedlemmarna, gemensamt såväl som självständigt, analyserar och diskuterar berörda frågeställningar. Vid varje iterations slut sker presentation av arbetet och produkten, under projektets gång sker kontinuerlig dokumentation – båda punkterna presenteras i ”KTH:s gemensamma mål för kursens examensarbete om 15hp för högskoleingenjörsexamen”. Utöver detta, har projektgruppen valt att använda sig av samma mall som används vid projektdefiniering under examensarbeten.

Vad gäller de uppsatta kursmålen i den berörda kursen behandlar projektet utvecklingsprocessen SCRUM. Genom detta projektarbete kommer studenten att få ökad kunskap inom bland annat:

- **Arbete inom projekt i IT-branschen**
- **Tillämpa viktiga verktyg och metoder som stödjer utvecklingsprocessen**
- **Reflektera över genomfört projekt**
- **Skriva en teknisk rapport**

4.2.1 Vetenskaplighet

Projektets vetenskaplighet är värderad utifrån rapporten från LTH (Lunds Tekniska Högskola) vetenskaplighet/ingenjörsmässighet (Andersson & Ekholm, 2002).

I vårat projekt anses att vi har följt nedanstående punkter till viss mån, se figur 1. Vi använder begrepp och teoretiska strukturer genomgående under hela projektet. Vi jobbar systematiskt och empiriskt när vi testar våra teorier. Vi identifierar och styr de variabler som påverkar vårat projekt under retrospektmötena och planeringsmötena. Relationerna mellan fenomen som berör problemställningen är identifierat. Och vi använder förklaringsmodeller för att visa våra produktägare samt andra intressenter. Detta visar på att vi har använt oss av en vetenskaplig metod jämfört med om vi inte hade följt nedanstående punkter då vi hade använt sunt förnuft.

Vetenskap kontra ”sunt förnuft”, se citat:

Kerlinger (1973) poängterar på samma sätt vikten av den vetenskapliga metoden och gör en distinktion mellan vetenskap och sunt förnuft utifrån fem punkter, vilka alla kretsar kring begreppen systematik och kontroll. Vetenskap inkluderar, till skillnad från sunt förnuft:

1. Användandet av en enhetlig begreppsapparat och teoretiska strukturer
2. Att systematiskt och empiriskt testa teorier och hypoteser
3. Kontroll, t.ex. att identifiera och styra de variabler som påverkar den studerade företeelsen och samtidigt systematiskt utesluta de variabler som inte berör den aktuella problemställningen
4. Att metvetet och systematiskt identifiera relationer mellan fenomen som berör problemställningen
5. Användningen av förklaringsmodeller.

Av punkterna framgår att vetenskap hanterar observerade fenomen och förklarar relationer mellan fenomen, vilka kan testas och utvärderas.

Figur 1: citat ur (Andersson & Ekholm, 2002)

Vidare kan man säga att vi har följt nedanstående punkter i den vetenskapliga metoden för att utföra teknologisk forskning, se figur 2. Vi började med att identifiera vad vi har för problem. Sedan tog vi fram en lösning för problemet. När det var gjort började vi kolla på vilka underlag som redan fanns. Däribland kom det upp att det till exempel redan fanns ramverk för att utveckla i python och arduinos egna språk och skapade det underlag som inte fanns som exempelvis en arkitektur för hur vi skulle bygga systemet. Sedan började vi utveckla vår produkt och den visade sig fungera bra så vi började med att skapa vår modell av produkten. Konsekvenser som modellen har är att den till exempel bara kan hantera en blomma vilket vi nöjde oss med i detta skede och modellen är därmed tillfredsställande. Sedan utvärderade vi resultatet i förhållande med befintlig kunskap och praxis och konstaterade att vi har skapat en bra produkt som uppfyller det mål som den var avsedd att fylla, det går att vattna sina blommor utan att vara hemma och fysiskt göra det. Vill man fortsätta med denna forskning så kommer det finnas minst 2 nya problem att lösa. Det första är att systemet ska kunna hantera flera blommor. Det andra är att man skulle vilja komma åt funktionerna på ett smidigare sätt, exempelvis med en app. Det finns även lite mera funktionalitet att implementera som skulle kunna ses som ett problem att lösa. Detta tyder också på att vi har använt ett vetenskapligt tillvägagångssätt vid projektet.

Teknologisk forskning (se Andersson & Ekholm för "generell vetenskaplig metod")

Den generella vetenskapliga metoden kan anpassas för teoretisk, experimentell och teknologisk forskning. För den senare inriktningen, teknologisk forskning, följer den vetenskapliga forskningsprocessen följande steg (Bunge 1983):

1. Hur kan den aktuella problemställningen lösas?
2. Hur kan en teknik/produkt utvecklas för att lösa problemet på ett effektivt sätt?
3. Vilket underlag/information finns och erfordras för att utveckla tekniken/produkten?
4. Utveckla tekniken/produkten utifrån underlaget/informationen i steg 3. Om tekniken/produkten visar sig fullgod, gå till steg 6.
5. Försök med ny teknik/produkt.
6. Skapa en modell/simulering av den föreslagna tekniken/produkten.
7. Vad medför, alltså vilka är konsekvenserna av, modellen/simuleringen i steg 6?
8. Testa tillämpningen av modellen/simuleringen. Om utfallet inte är tillfredsställande gå till steg 9, annars gå till steg 10.
9. Identifiera och korrigera för brister i modellen/simuleringen.
10. Utvärdera hur resultatet i förhållande till befintlig kunskap och praxis, samt identifiera nya problemområden för fortsatt forskning.

Figur 2: citat ur (Andersson & Ekholm, 2002)

Enligt detta skulle man kunna säga att vi jobbar vetenskapligt i projektet i och med att vi empiriskt samlade på oss information under projektets gång och analyserade och drog därefter slutsatser baserade på erfarenheterna.

Jag vill samtidigt förespråka att vi inte har jobbat speciellt vetenskapligt eftersom studien är baserad på våra åsikter och värderingar. Det är inte heller vetenskapligt då det inte är säkert att om någon annan person gjorde studien så kanske denna inte skulle få samma resultat (Andersson & Ekholm 2002: sida 16).

4.3 Hållbarhetsaspekter

- Vi har under projektets gång försökt att inte skriva ut så mycket papper. Vi har istället hänvisat till våran Google drive där allt finns sparat.
- Vi i teamet som bor tillräckligt nära har även tänkt på att cykla till och från våra möten om vädret tillåtit det för att inte förstöra vår miljö mera än nödvändigt, i övrigt har vi alla åkt kommunalt.
- När vår produkt ska avvecklas så går alla delar att återanvända eller sälja vidare begagnat.

4.4 Etik, jämställdhet och likabehandling (JML)

- I våran projektgrupp har alla i teamet varit lika behandlade av varandra.
- Vår grupp har även bestått av en ganska jämn fördelning mellan könen då vi har haft 3 män och 2 kvinnor i teamet.

4.5 Arbetsmiljöaspekter

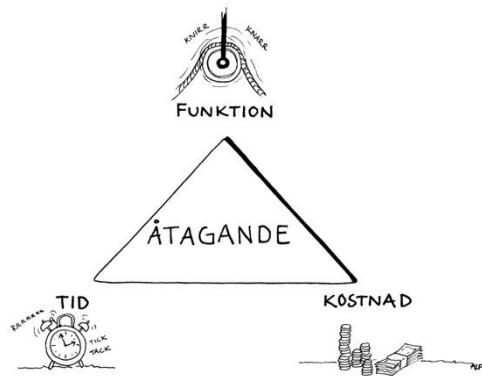
- Vi har valt att sitta utanför grupprum 320 vid våra möten, delvis för att det är en ljus och trevlig arbetsmiljö där och dels för att det är bättre luft där än om vi hade trängt in oss i ett grupprum.
- Vi har även haft en fika tillsammans för att lära känna teamet bättre och för att trivseln skulle öka.
- Det skulle vara bra att se över stolar och bord för att arbetsmiljön ska bli bättre.

5 Fas-, tids- och arbetsplan

Projektet kommer fortlöpa under ca 10 veckor med en beläggning på halvtid.

Dem flexibla hörnen i ”projekttriangeln” är kostnaden och funktionen, dessa går alltså att spela med medan tiden är fast bestämd.

Totalt ska varje projektmedlem lägga någonstans mellan 180-200 timmar på projektet. Fördelningen av tiden ligger ganska jämnt fördelad mellan veckorna med största skillnaden på 10 timmar från ena veckan till andra.

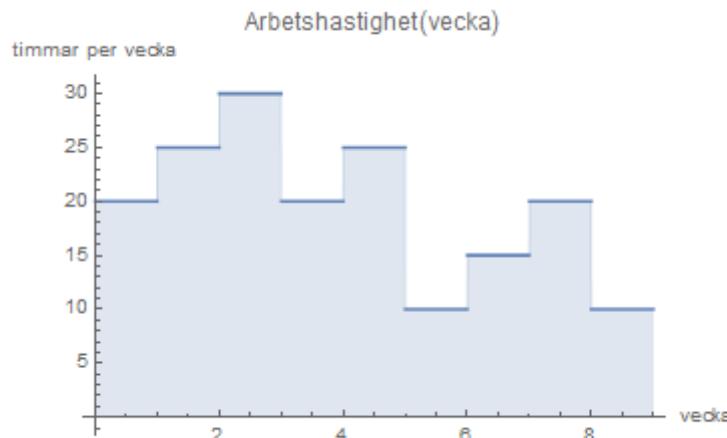


Figur 3. Ur boken "Projekt och projektmetoder" (Eklund 2010)

5.1 Arbetsplan och arbetsvolym per projektdeltagare

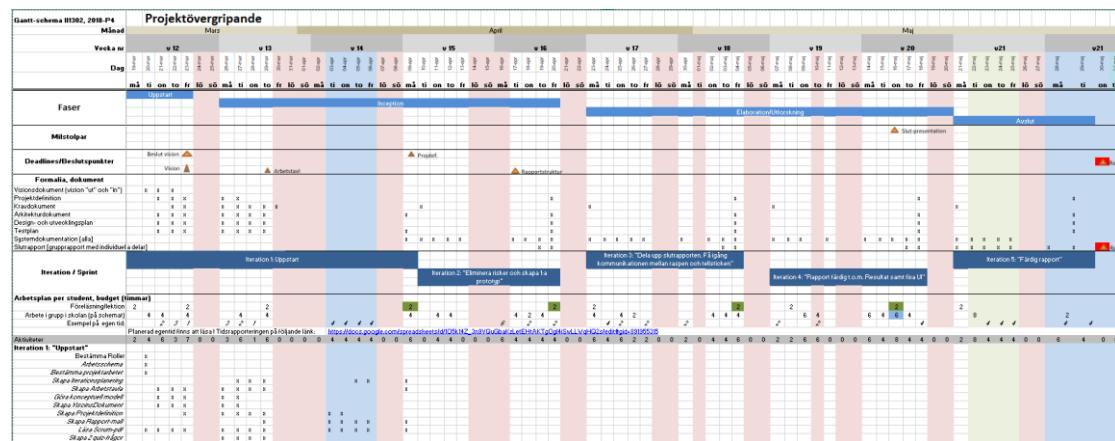
Följande diagram och beräkning är skapat med verktyget Mathematica och visar planerade arbetstimmar för en av projektdeltagarna.

Integralen av funktionen i diagrammet nedan och över de visade veckorna ger den totala planerade arbetsvolymen som skall bli ca 200h per student.



Figur 4 Exempel (ej alla veckor)

5.2 Fas- och tidsplan



Figur 5 Originalark finns på url:
<https://drive.google.com/open?id=17ZoupSeLPyxF3wa-V9USd04SWu5yx788>

6 Intressenter

Lista vilka som är projektets intressenter, deras förväntningar och ambition att uppfylla dessa förväntningar och hur.

Inressent	Namn	Förväntningar	Uppfyllande av förväntningar
1	Anders Sjögren	Få en kursrapport och en fungerande produkt.	Jobba iterativt så kommer vi att ha en fungerande produkt med mer eller mindre funktion vid projektets slutfas. Då kan vi också skriva en kursrapport.
2	Vår projektgrupp	Underlätta blomvattningen i våra egna hushåll.	Jobba iterativt så kommer vi att ha en fungerande produkt med mer eller mindre funktion vid projektets slutfas.
3	Projektgrupp 4	Att User story nummer 1, 4 och 7 är genomförda vid projektets slut.	Jobba iterativt så kommer vi att ha en fungerande produkt möjligt även med några önskvärda funktioner vid projektets slutfas.

7 Riskanalys

Nedan beskrivs identifierade risker.

Risklista

ID	Risk	Förebyggande åtgärd	Åtgärder vid riskutfall
R1	All trådlös kommunikation	Börjar med detta för att i tidigt skede kunna utesluta denna risk.	Koppla upp oss med kablar.
R2	Nya programme-ringsspråk	Kolla på tutorials på nätet vid behov.	Byta språk.
R3	För mycket tid på dokument vs. Systemet	Se till att ha alla dokument klara i sprint 1.	Minska kraven på systemet.
R4	Problem att få hårdvaran att fungera	Googla & läsa litteratur om hur man ska gå tillväga.	Köpa ny hårdvara.
R5	Beställning av hårdvara sker för sent	Beställa snarast, senast i iteration 2.	Använda privat utrustning eller minska kraven på systemet.
R6	Hög fränvaro i gruppen	Prata med fränvarande deltagare.	Prata med fränvarande deltagare.
R7	Konflikter i gruppen	Prata med varandra.	Projektledare har utslagande röst.
R8	Tidsnöd	Planera och scrummötena ska klargöra var vi ska lägga vår fokus.	Minska kraven på systemet.
R9	Projektdeltagare hoppar av kursen	Dokumentera vår kod väl.	Minska antalet story points per sprint.
R10	Uppkopplingen till nätet ligger nere		Använda mobilt nät, byta lokal eller implementera automatisk bevattning.
R11	Raspberry pi får virus	Sätta upp brandvägg.	Reboota raspen alternativt köpa in en ny.
R12	Att vi har sönder hårdvaran	Vara försiktiga.	Köpa in ny.

Riskbedömning

	Hög sannolikhet			
Liten påverkan		R3	R8	Stor påverkan
	R2	R7	R1, R4, R6	
	R5	R10, R11, R12	R9	
	Låg sannolikhet			

8 Förändringsplan

Skulle det ske någon viktig förändring i projektet så meddelar vi det via messenger och på nästa gemensamma möte vi har.

9 Kostnadsplan

Kostnader i projektet

ID	Betalare	Kostnad	Vad?
1	KTH	199,00 kr	Vattenpump
2	KTH	89,00 kr	Slang
3	KTH	139,00 kr	Slangklämma
4	KTH	49,00 kr	Backventil till vattenpump
5	KTH	119,00 kr	Förvaringslåda 32 l, 50x39x26 cm
6	KTH	29,90 kr	Förvaringslåda 2 l, 21x17x11 cm
7	KTH	29,90 kr	Förvaringslåda 1 l, 21x17x6 cm

10 Dokumentplan

Nedanstående dokument är dem som vi i gruppen har kommit fram till att vi ska arbeta med och underhålla under projektet.

ID	Ansvarig	Dokumentets namn
D1	Kim	Projektdefinition
D2	Kim	Gantt
D3	Erik	Visionsdokument
D4	Erik	Story Cards
D5	Max	Arkitekturbeskrivning
D6	Max	Development View
D7	Hampus	Design- och utvecklingsplan
D8	Hampus	Kod-modell
D9	Nicole	Testplan
D10	Alla	Slutrapport
D11	Alla	Tidsrapportering

11 Utbildningsplan

ID	Person	När?	Utbildning?
U1	Alla	9 April	Läsa: Introduktion till agil projekthantering
U2	Alla	9 April	Läsa: Scrum and XP from the trenches
U3	Alla	9 April	Läsa: Survey of Agile Tool Usage and Needs
U4	Alla	9 April	Läsa: Förslag till en ekonomiskt hållbar projektmetod
U5	Dem som behöver	20 April	Prova på nya programspråken.
U6	Den som behöver	20 April	Googla och/eller läsa litteratur om hur hårdvaran fungerar.

12 Rapport- och granskningsplan

Datum	Klockslag	Mötestyp	Syfte
9 April	15.00-17.00	Scrumdemo	Visa kunden hur långt vi har kommit, samt stämma av att vi är på rätt spår.
9 April	12.30-15.00	Retrospective & Sprintplanning	Utvärdera & förbättra till nästa iteration. Samt planera nästa iteration.
19 April	10.00-12.00	Retrospective & Sprintplanning	Utvärdera & förbättra till nästa iteration. Samt planera nästa iteration.
20 April	8.00-10.00	Scrumdemo	Visa kunden hur långt vi har kommit, samt stämma av att vi är på rätt spår.
4 Maj	8.00-10.00	Scrumdemo	Visa kunden hur långt vi har kommit, samt stämma av att vi är på rätt spår.
4 Maj	10.00-12.00	Retrospective & Sprintplanning	Utvärdera & förbättra till nästa iteration. Samt planera nästa iteration.
16 Maj	10.00-12.00	Scrumdemo / Slutpresentation	Visa kunden hur långt vi har kommit, samt stämma av att vi är på rätt spår.
17 Maj	10.00-12.00	Retrospective & Sprintplanning	Utvärdera & förbättra till nästa iteration. Samt planera nästa iteration.
30 Maj	8.00-18.00	Scrumdemo / Rapportinlämning	Examination på att vi har lärt oss enligt kursmålen.

Appendix A - Referenser

Andersson, N., & Ekholm, A. (2002). Vetenskaplighet - Utvärdering av tre implementeringsprojekt inom IT Bygg & Fastighet 2002.

Eklund, S. (2010). *Arbete i projekt: individen, gruppen, ledaren*: Studentlitteratur.

Bilaga 2

KTH Projekt och projektmetoder II-1302

Kravspecifikation Flower Power

Skapad datum: 180427

Senast ändrad: 180521

Dokumentansvarig: Erik Holm

Version: [4]

Godkänd av:

Bilagor

1. Use case Model sprint 3
2. Exempel på use Case uppdelat i tasks för arbetsgrupp
3. Exempel på use case i fall av anpassad user slice i sprint 3

Innehåll

1. Introduktion
2. Definitioner och förkortningar
3. Projektets syfte
4. Kund och andra intressenter
5. Användare
6. Avgränsningar
7. Fakta och förutsättningar
8. Krav
9. Osäkerheter och risker
10. Användardokumentation och utbildning
11. Tidsuppskattning

Referenser

1. Introduktion

1.1. Syfte

Syftet med kravspecifikationen är att ge en inblick i projektet "FlowerPower":s projektmetod och hur projektet har arbetat utifrån kundens krav. På ett kort och konkist sätt ska en eventuell efterträdare snabbt kunna sätta sig in i det utförda arbetet och kunna fortsätta i linje med den hittills utarbetade strukturen. Projektet använder ett agilt arbetsätt enligt Scrum vilket kommer sammanfattas och kopplas till det aktuella projektets utformning under punkt 8 - "Krav".

2. Definitioner och förkortningar

2.1. Prescriptive approach

Antalet begränsningar. En mer prescriptive approach har inte samma valfrihet, färre öppna val.

2.2. Skikt

Ett skikt är en grov indelning av en applikation på fysisk nivå. Ett skikt ansvarar för en grupp av funktioner.

3. Projektets syfte

3.1. Projektets bakgrund

KTH vill utforma en produkt som kan underlätta bevattning av växter i deras lokaler. En förstudie visade växters naturliga del i det svenska hemmet. Plantagen är nordens ledande kedja för försäljning av växter. De är därför en indikator kring trender och försäljning i Sverige. Mellan år 2016 och 2017 ökade försäljningen med 2%.¹ Med en ökad försäljning ökar även behovet till skötsel och FlowerPowers tänkta kundkrets.

Tid. Vi är alltid i behov av mer tid. Tid att tvätta, tid att laga mat, tid att vattna blommor. Marknadens att optimera/effektivisera den tid vi spenderar till varje syssla har blivit en marknad i sig. Familjer får livsmedel hemkördta till dörren, husdjur får tillfälliga ägare när deras husse och matte reser bort. Flower Power är ytterligare ett naturligt steg i den riktningen. Behöver man jobba över, stanna kvar på affärsresan ytterligare några dagar så ska man ändå inte behöva mötas av en vissen växt när man kliver innanför farstun. Det blir extra svårt när man bor själv och inte kan dela upp ansvaret. Det är ett problem eftersom ensamboende utgör 38% av alla hushåll². Flower Powers applikation skickar påminnelser och sköter bevattningen åt en.

3.2. Mål

Produkten ska senare kunna implementeras i alla hem/företagslokaler. En sensor ska kopplas till vald växt, sensorn är i sin tur kopplad till en databas där användaren ska se värden för samtliga växter i hemmet. Ett enkelt paket ska kunna beställas för att på egen hand kunna sättas upp i det egna hemmet. För utförligare implementeringsdokumentation se punkt 6.1.

3.3. Effekt

Den långsiktiga effekten av FlowerPower är en modul med låg kostnad och behov av skötsel som underlättar bevattningen i svenska hem. Det kommer i sin tur leda till en ökad livskvalitet för den tänkta användaren då FlowerPower avlastar från den dagliga arbetsbördan. Ökad livskvalitet minskar stress och den övergripande folkhälsan, vilket är i statligt intresse.

4. Kund och andra intressenter

4.1. Lista över kund och intressenter

KTH – Första intressent som har påbörjat projektet i form av en kravformulering

Statlig intressent – I och med främjandet av folkhälsa kan en möjlig intressent vara statliga företag, i form av subventioner utan vinstintresse.

5. Användare

5.1. Användargrupper

Den tänkta målgruppen är bred och innefattar både privatpersoner (stora familjer samt ensamstående) och företag av varierande storlek. Det är viktigt att utvärdera varje användares förkunskaper. Ett företag kan till exempel ha en teknikansvarig som har betydligt lättare för att implementera modulen än vad en privatperson har. Det är viktigt att ha i åtanke vid uppdelningen av tasks till varje use case.

6. Avgränsningar

6.1. Implementation

Implementationen innefattar en Arduino Uno med tillhörande sensor, en ESP8266 wifi-modul samt en raspberry pi.

- ESP8266 kopplar upp sig till ett lokalt nätverk och kommunicerar genom serial till arduinon.
- Raspberry Pi har en databas med alla värden och en tråd för varje planta, tråden skickar en HTTP request som plockas upp av ESP8266
- Requesten skickas vidare till Arduino. Arduinon väntar på requesten i en loop-funktion
- Svaret blir en string i ett text/html svar som skickas till ESP, vidare till raspberry.
- Raspberry Pi får en string som den kan hantera och skicka vidare till en SQL-databas samt till front end.

För ytterligare dokumentation, se teknisk rapport för implementation av wifi samt arkitekturdokumentation.

7. Fakta och förutsättningar

8. Krav

8.1. Introduktion av krav

Projektets krav- och kundansvarig arbetar utifrån en agil arbetsmetod enligt Scrum. Projektet har kantats av sen kontakt med kund och allmänt vaga kravformuleringar, därför har scrum hjälpt till med sin flexibla anpassning till förändringar. En vattenfallsmeddelad hade låst projektet på en väg, vilket kräver en större flexibilitet i den initiale planeringen. KanBan hade även kunnat fungera i projektet, samtidigt var det viktigt att ha en tydligare struktur av roller och uppgiftsfördelning, en mer preskriptiv approach då gruppmedlemmarna har en relativt kort introduktion i metodlära och behövde guidas i studiesyfte. Det var bra att välja en "batch" med tasks till varje sprint samt ha en rollfördelning i gruppen för att få igång arbetet snabbt, något som inte ingår i KanBans metod utan snarare Scrum (KanBan and Scrum Info Q sid.11).

Det har under projektets gång varit viktigt att undvika alltför tekniska cases/stories, det hör hemma i tasks som man arbetar fram för teamet. På så sätt kan vår kund vara med i avvägningar kring tradeoffs (Scrum and XP from the trenches, sid. 47). Se bilaga 3 för exempel vid uppdelning av tasks samt punkt "punkt" för ytterligare förklaring. Observera även att kraven är uppdelade i funktionella samt icke-funktionella krav. Alla krav är inte alltid riktade till användarens direkta interaktion med produkten utan även "de restriktioner och krav som finns på en systemtjänst" (Sommerville). Trots att användaren inte ser kraven direkt så kommer de ändå vara viktiga för hur kunden uppfattar produkten. Därför är det minst lika viktigt att de icke-funktionella kraven uppmärksamas i det fortsatta arbetet (se punkt 8.4).

8.2. User Stories

- Som användare vill jag gå in på hemsidan när jag är på semester och se fuktvärden för att få en snabb överblick
- Som användare vill jag få en notis vid ett visst fuktvärde för att slippa hålla koll på det själv, helst till min mail
- Som användare vill jag kunna ändra fuktkänslighet för att anpassa bevattning till olika växter då jag har kakthusar som inte kräver lika mycket vatten
- Som användare vill jag kunna starta bevattning för vald blomma för ökad kontroll över bevattningsprocessen
- Som användare vill jag kunna nödstoppa bevattningen för ökad kontroll

Såg listan ut så här från början? Svaret är nej, den har redigerats löpande under projektets gång. Som Kinberg föreslår (Scrum and XP from the trenches, sid. 41) fanns det en checklist för "definition of ready" och "definition of done" där hemsidan inte platsade i vår checklist "definition of done". Det kan liknas vid avvägningen man gör mellan stories till use case (Jacobson 2015). Det

får dock inte blandas ihop med stories som Jacobson pratar om. Story att gå in på hemsidan ansågs till exempel inte vara redo att appliceras som use case eftersom den var för övergripande och var en del av vattna blommorna/se över fuktvärdens main flow. Den kunde istället läggas in i use case:et se fuktvärden utan att bli ett eget case, vilket blev andra sprintens huvudmål. Hemsidan blev istället en task för det arbetande teamet. Allt för att möta kundens förväntningar. Projektgruppen arbetar både Top Down samt Bottom Up, men till en början Bottom Up (Use Case 2.0 sid. 17). Två stories togs fram för att undvika att blommor vissnade och definierades sen som UC. Därefter definierades ett main flow och alternative flow. Vissa alternative flows blev ett eget UC, men ett sekundärt sådant. Det visades genom en tydlig rangordning. I punkt 8.3 presenteras de i den nämnda rangordningen.

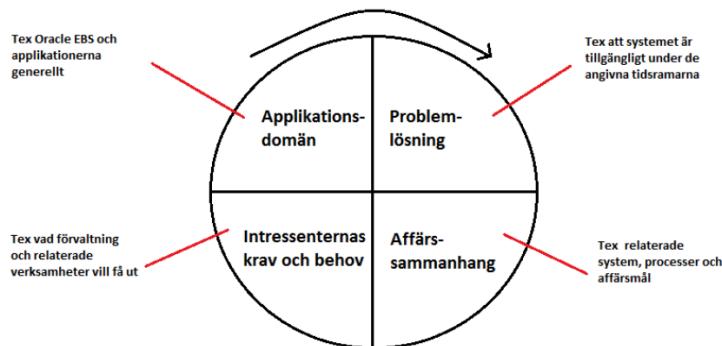
8.3. Use Cases av funktionella krav.

Prioritering i samrådan med kund togs fram utifrån kundens stories och rangordnades på följande sätt. Kraven delades upp i nödvändiga krav och "förbättringar" som i samrådan med kund fick finslipa produkten i mån om tid. Uppdelningen var ett grepp på metoden att dela upp ett primärt UC:s alternative flow till sekundära UC.

1. Översikt av fuktvärden
2. Få notis vid fuktvärde
3. Ändra sensorkänslighet
4. Starta bevattnings
5. Nödstoppa bevattnings
6. Kunna se över värden på språng (mobil)

Det är viktigt att fortfarande koppla varje story till sitt case eftersom en ändring kan komma att ändra prioritering och struktur av kraven. Den iterativa kravhanteringen kan leda till många hinder som illustrerat i bilden nedanför.

Kravhantering ur fyra dimensioner



Problemlösning är hur produkten löser kunden krav på en produkt, men för att nå dit krävs även en krav hantering kring applikationsdomänen, t.ex. systemkrav. De här kraven kommer oftast upp under projektets gång. Därför är det extra viktigt att kopplingen mellan Case och Stories dokumenteras i en kravspecifikation. Bara för att man uppfyller kravet i en stund betyder inte att kravet är uppfyllt för kommande iterationer av produkten då förutsättningar ändras. En Bottom approach användes för att formulera första UC. Därefter skrevs main flow samt alternative flow till UC för att tillåta en Top Down approach. Krav som att stänga av vatten upptäcktes genom alternative flow och kunde formuleras som ett sekundärt UC. Det är viktigt att uppdatera specifikationen för att ha en översikt över kopplingen mellan UC och stories eftersom ändringar kan behöva en ny story eller till och med slice. Se punkt 8.5.3 för exempel och förtydigande.

8.4. Icke-funktionella krav

8.4.1. Användbarhet

- 8.4.1.1. Lättförståelig applikation ska eftersträvas genom att tydligt visa och guida användaren till respektive blomma, med pedagogiska färg- och typsnittsval

- 8.4.1.2. Data

Värden ska endast presenteras som heltal.

8.4.2. Felhantering

8.4.2.1. Alla programmerares tester ska skrivas i pytest för ökad kompatibilitet och för debug.

8.4.2.2. Väntan för exception vid wifi-connection mellan Arduino och Pi

Ett exception ska slängas fem sekunder efter klassen ArduinoConnections försök till server request mot arduinon/sensor. Det är för att underlätta den kommande felhanteringen i trådarna "plant" till huvudträden. Vid ändring av tid, var tydlig med att dokumentera tid och motivering av ändring.

8.4.3. Säkerhet

8.4.3.1. Datatermer får inte användas av användaren

För att undvika cross site-scripting måste indata från front end hanteras.

8.4.3.2. Bara auktoriserade användare kan se sina växters värden.

Det här säkerställs genom användarnamn och lösenord för att ta del av de värden som är knutet till kontot.

8.4.4. Användargränssnitt

8.4.4.1. Pedagogiskt gränssnitt

Pedagogiskt färg- och typsnittsval med val av språk. Extern style guide bifogas och dokumenterar eventuella ändringar.

8.5. Förtydliganden och tillägg av krav

8.5.1. Kopplat till punkt 8.1 - "Alltför tekniska krav"

Under Sprint 2 var ett krav att kunna kommunicera mellan raspberry pi och arduino uno. Det är ett alltför tekniskt task som inte borde presenteras för kund, inte ens som slice. Den lades därför in som ett task under use case:et "se översiktlig vy av fuktvärdet". Ett case som är utifrån användarens perspektiv och tydligt knutet till vår nämnda tidslinje. Uppdelningen sker enbart i ett task för arbetsgruppen att bry sig om. User Case ska vara av direkt värde för användaren.

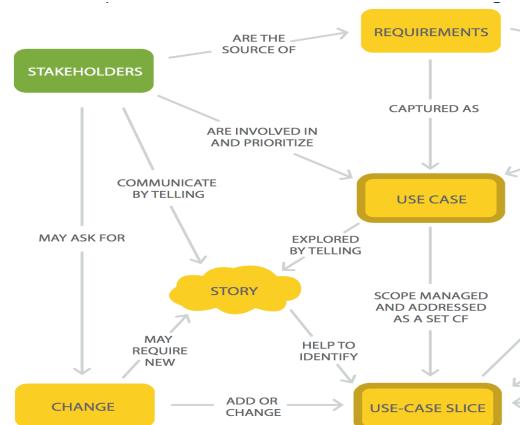
8.5.2. Kopplat till punkt 8.2 - Val av stories och "Definition of done"

Beslutet att slå ihop hemsidan med översikten av fuktvärdet togs bland annat på grund av resurserna. Kunskapen kring utveckling av hemsida samt tidsramen ansågs vara tillräcklig för att minimera risk att inte uppnå iterationens mål. Ponerar man andra förutsättningar hade hemsida-storyn kunnat vara en del av use case i form av en slice, där vi hårdkodade värden till vår hemsida.

Att föra in hemsidan som ett case, även om det är sekundärt, öppnar för ett missförstånd med kunden, eftersom det frångår de förväntningar och krav som de har på modulen. Hur ett Use Case ska uppnås är upp till teamet att arbeta fram.

8.5.3 Kopplat till 8.3 - Ändringar från kund med exempel

Ett problem, följt av en avvägning, skedde i sprint 3. Ett case hade ett task som skulle koppla samman en Telstick med en pump för att användaren skulle kunna starta en bevattning. På grund av hårdvaruförseningar var projektgruppen osäkra på om det var ett möjligt mål under sprinten. Use caset delades därför i samråd med kund upp i en slice för kommunikation med en enklare modul. Slicen var en tydlig tårtbit till målet kunden förväntade sig som kunde visas upp vid ett sprintdemo. Här hade kravspecifikationen en central roll när man behövde se över vilka Case som påverkas av ändringen.



Figuren ovanför (Jacobson 2005) visar just beroenden i kravprocessen. Vid ändring kan en ny story behöva skrivas samt en slice läggas till/ändras, vilket kommer att påverka relaterade use case. För att lokalisera detta caset krävs dokumentation. Naturligtvis är det ett större problem ju fler komponenter (slices) ett use case har delats upp i när ändringen av UC sker. (Jacobson 2005, 16-17).

Den slice som kom ur beslutet är ytterligare ett steg mot den fullständiga insynen i ett projekt som diskuterades i inledningen. Färre missförstånd med kund sparar på tidsresurser och arbetskraft.

9. Osäkerheter och risker

9.1. Beroenden

9.1.1. Vår front end är beroende av att hårdvaran kopplas rätt hos leverantör. Det är därför viktigt att felsökning är möjligt på varje nivå för att kunna lokalisera problemet. Tester skrivs i pytest som standard för projektet.

9.2. Stabilitet

9.2.1. Brandvägg och säkerhet hos det valda nätverket kan försvåra uppkopplingen hos ESP8266. Uppkopplingen tar för givet att den är auktoriserad efter lösenord och användarnamn har angivits.

9.3. Prestanda

9.3.1. Arduinon kan ha svårt att serva fler än en sensor, vilket kommer bli en flaskhals för projektet. Vid implementering av ytterligare sensorer, utfärda ett task att se över möjligheter till strömförsörjning samt tillägg av pins innan implementeringen blir en task i en kommande sprintiteration.

10. Användardokumentation och utbildning

10.1. Rollutbildning

Till varje roll ingår ett flertal seminarium och kunskapstester för att underlätta samarbetet i projektgruppen

11. Tidsuppskattning

11.1. Tidsuppdelning i sprintmoment

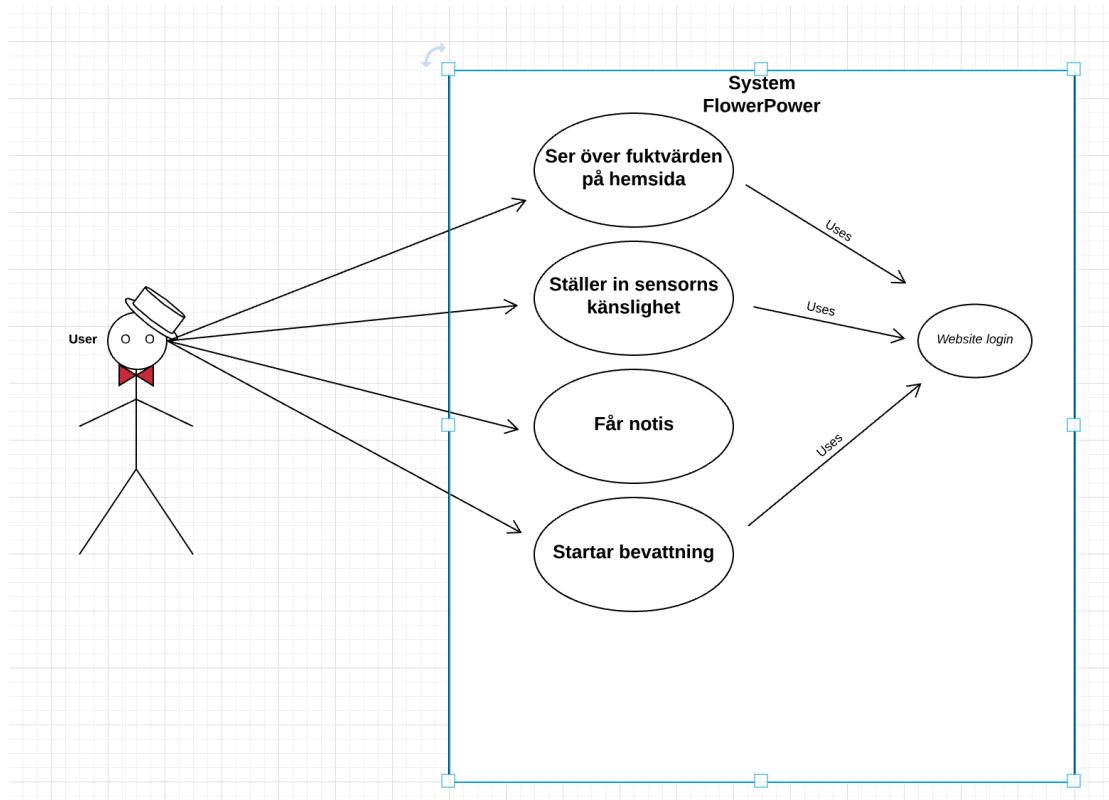
Projektets tidsuppskattning löper under en högskoletermin. Enligt Scrum skedde sen uppdelningen i iterationer. Första och andra sprinten hade en burndown på 20 poäng. Medan tredje hade en uppskattad tid på 30 poäng. Varje poäng är 4 timmar. Anledningen till en ökad tidsuppskattning berodde på att arbetskraften behövde delas upp på två parallella projekt (kursrapport).

Referenser

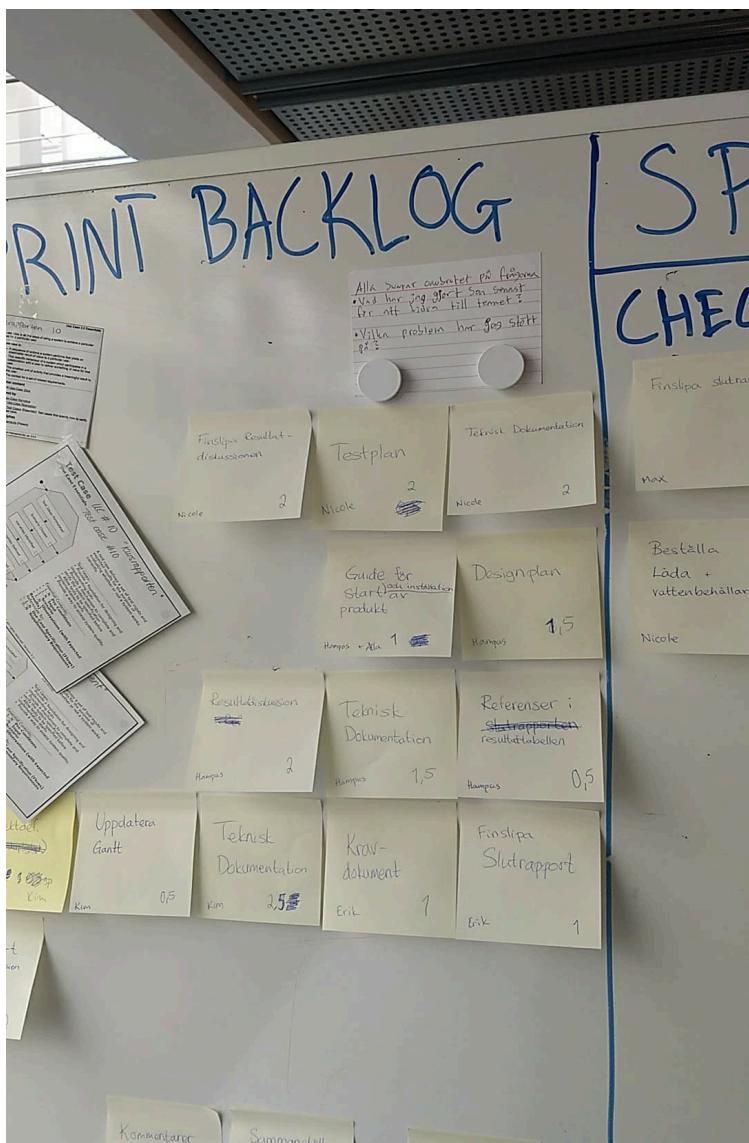
[1] https://www.ratos.se/globalassets/global/05_investor-relations/delarsrapporter/2017_q1/delarsrapport-q1-sve.pdf

[2] https://www.scb.se/sv/_Hitta-statistik/Artiklar/Atta-av-tio-delar-hushall-med-nagon/

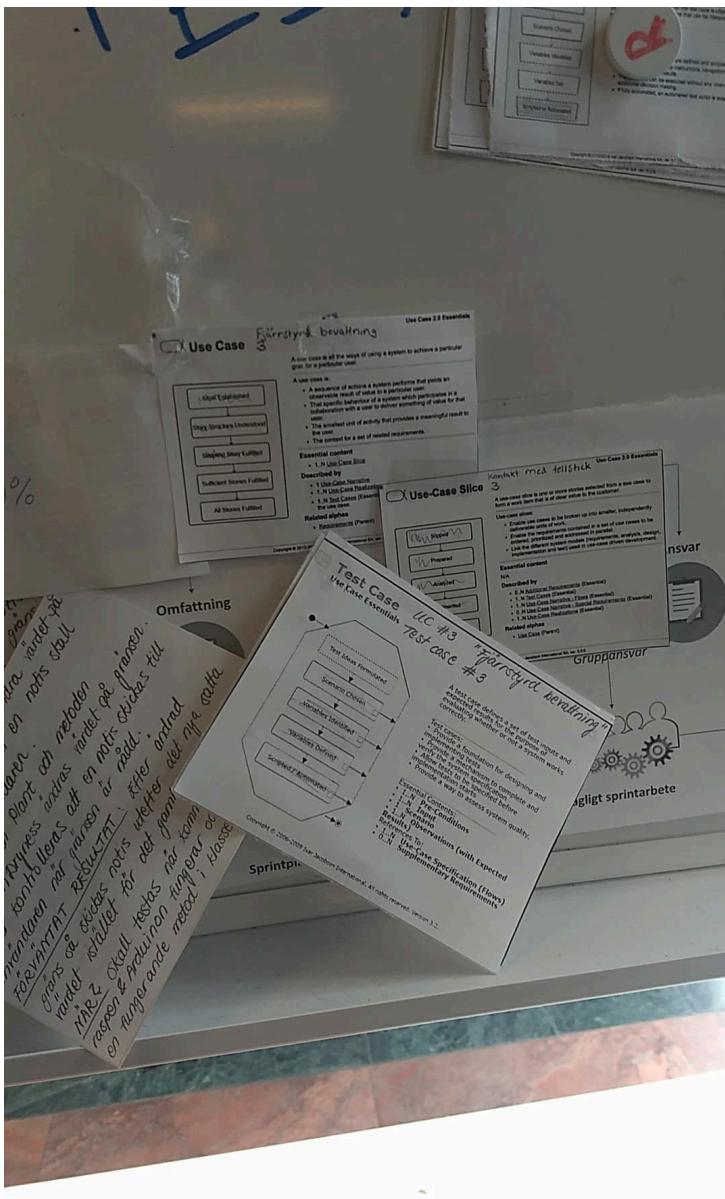
Bilaga 1



Bilaga 2



Bilaga 3



Arkitekturbeskrivning

Författare: Max Körlinge

1. Inledning

1.1 Syfte

Arkitekturen ska användas för att först skapa en plan för projektets utförande att utgå ifrån, samt för att sedan behålla en överblick över helheten av projektet under tiden men arbetar med de olika delarna. Därefter används arkitekturbeskrivningen även för att möjliggöra för nya utecklare att arbeta vidare med projektet. Detta dokument ska ge en beskrivning av produktens arkitektur med hjälp av olika vyer för att beskriva olika delar av systemet ur olika synvinklar enligt 4+1-modellen 4+1 [1].

1.2 Omfattning

Dokumentet omfattar arkitekturen för produkten "FlowerPower" [3], som utvecklas av projektgrupp 10 på KTHs kurs "Projekt och projektmetoder" VT2018. Det omfattar ej tester av arkitekturen, som i stället finns i projektets testplan [5]. Det omfattar ej heller koddesign på detaljnivå, det vill säga klassers specifika metoder, attribut, etc. Denna detaljnivå kan i stället hittas i projektets designplan [6].

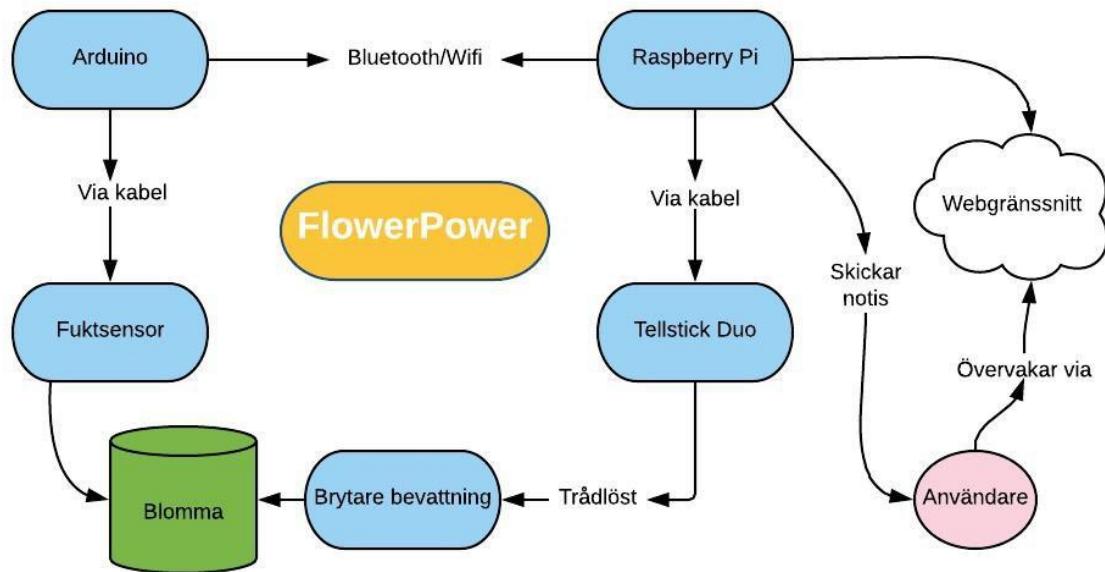
1.3 Referenser

- [1] Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, 12(6), 42-50.
- [2] Sommerville, I. (2011). *Software Engineering*. 9th ed. Pearson.
- [3] Projektdefinition FlowerPower. <https://github.com/fongie/projectgaming/wiki/6-Projektledning> (Hämtad 2018-05-22).
- [4] Kravdokument FlowerPower. <https://github.com/fongie/projectgaming/wiki/2.-Krav> (Hämtad 2018-05-22).
- [5] Testplan FlowerPower. <https://github.com/fongie/projectgaming/wiki/5-Test> (Hämtad 2018-05-22).
- [6] Designplan FlowerPower. <https://github.com/fongie/projectgaming/wiki/4-Design> (Hämtad 2018-05-22).

2. Användarfall

2.1 Bakgrund

Användarfallen ska beskriva viktiga sätt som användaren interagerar med produkten på [4]. Figur 1 visar en konceptuell modell över produktens beståndsdelar och hur den interagerar med användaren. Raspberry Pin agerar alltså server och servar användaren, tar in och bearbetar data från fuktsensorn via en Arduino, samt kan skicka signaler för att stänga av och sätta på bevattning via en Tellstick Duo.



Figur 1. En illustration av konceptet för produkten FlowerPower. Tjänar även som illustrering av fysisk vy. Modellen beskriver systemet när det används med endast en blomkruka.

2.2 Användarfall

- Användaren ska kunna övervaka fuktvärdena på användarens blommor via ett webgränssnitt.
- Användaren ska få en notis om jorden torkar ut.
- Användaren ska kunna starta bevattning av en blomma via internet.
- Användaren ska kunna ställa in systemet för att automatiskt bevattna blomman när den blir torr.

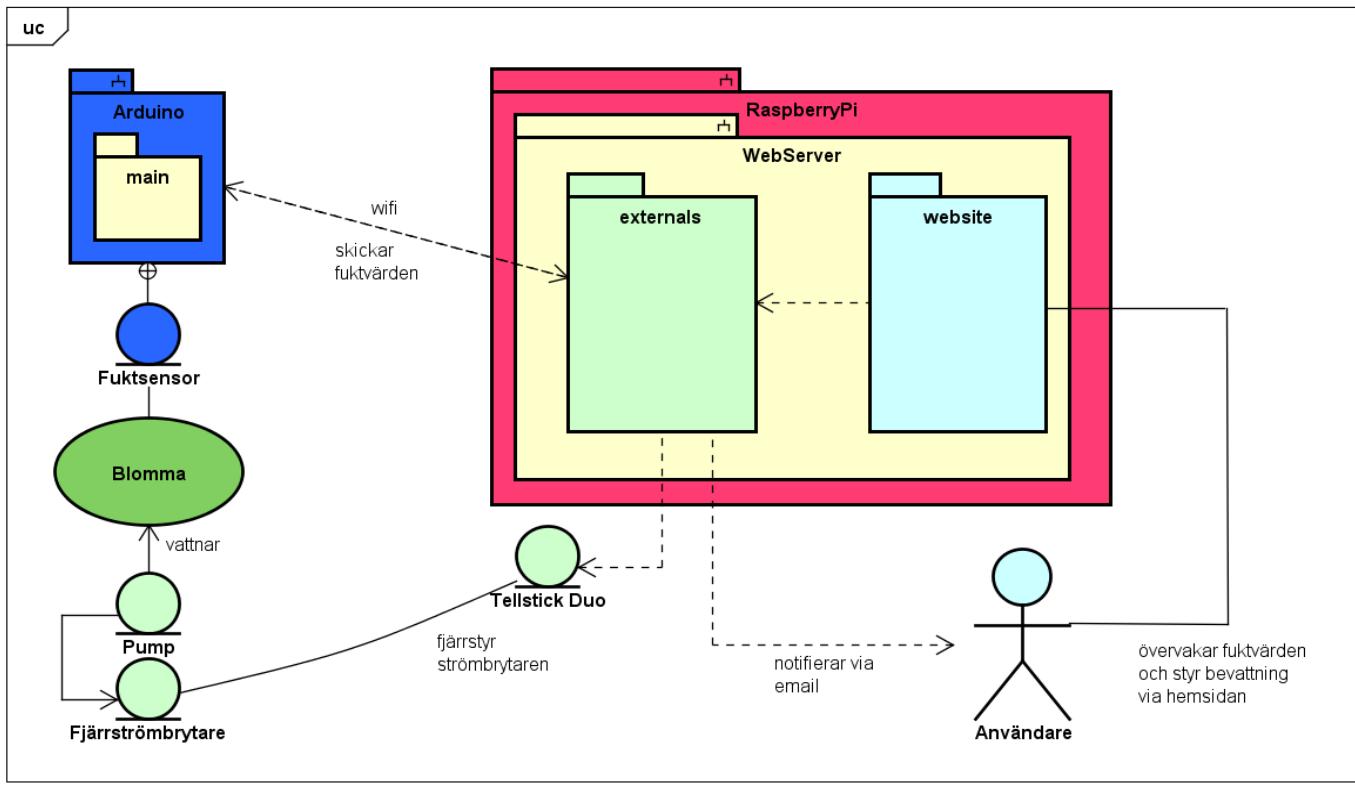
Intressenten som agerar i dessa fall är en ensam användare av produkten, som använder den på en eller flera blomkrukor. Det behövs en Arduino för varje blomkruka, men endast en Raspberry Pi.

3. Logisk vy

3.1 Bakgrund

Den logiska vyn beskriver hur de viktigaste delarna hänger ihop logiskt. Den ger en överblick av hur systemet passar ihop, och fungerar som en blandning och översikt av de andra vyerna.

3.2 Logisk arkitektur



powered by Astah

Figur 2. En logisk vy över fuktkontroll- och bevattningssystemet. Färgkodning för att försöka beskriva vilka komponenter som tillsammans bildar logiska subsystem.

3.2.1 Arduino

Detta subsystem hanterar interaktion med fuktsensorn, samt anslutningen till centralservern (Raspberry Pi) via wifi. Det kommer att sända information om blommans fuktstatus till **externals**.

3.2.2 website

På Raspberry Pi ligger subsystemet **website** som hanterar gränssnittet som Användaren ser och kan utnyttja. Därifrån kan användaren övervaka fuktstatus samt starta bevattnings.

3.2.3 externals

På Raspberry Pi ligger också subsystemet **externals**, som hanterar allt som rör blomman och externa komponenter: mottagande av fuktstatus från Arduinon, startande av bevattning via Tellstick Duon, samt sändandet av notifikationer till användaren vid behov.

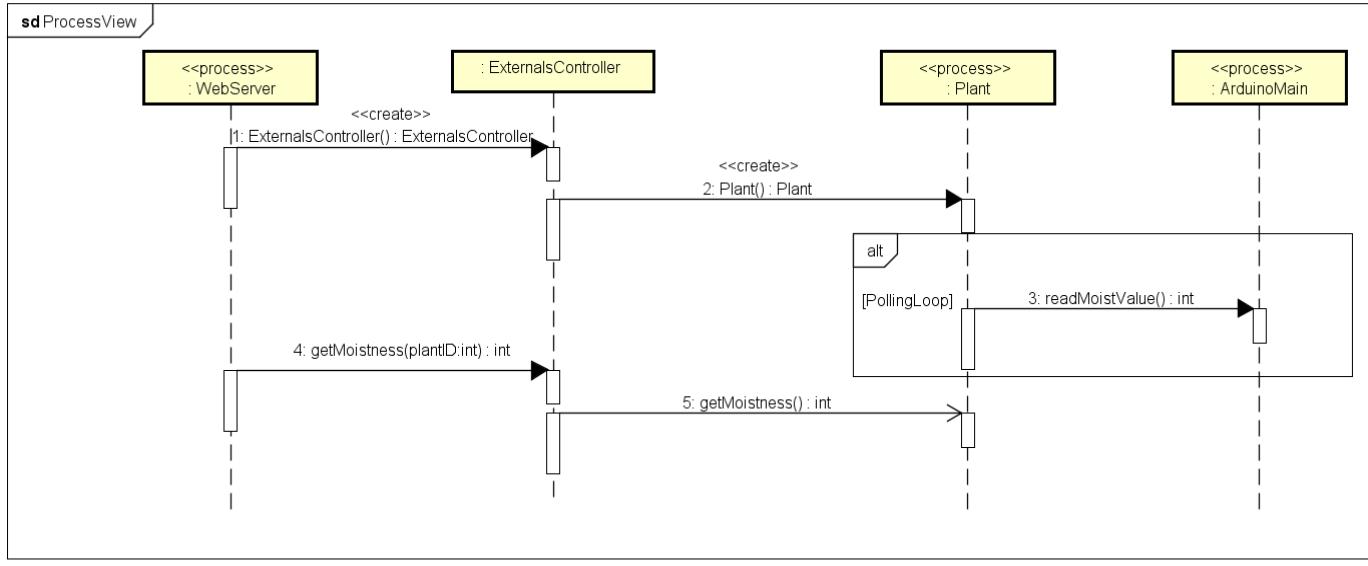
4. Processvy

4.1 Bakgrund

Processvyn beskriver vissa icke-funktionella krav så som optimeringar, parallelism och distribution. Den beskriver processerna som körs på samma eller på olika

fysiska enheter och ger en överblick över hur de interagerar med varandra. För en närmare beskrivning över de fysiska enheterna, se den konceptuella modellen och den fysiska vyn.

4.2 Beskrivning



Figur 3. Ett sekvensdiagram över processerna och exempel på hur de kommunicerar med varandra.

4.2.1 WebServer

Webservern som driver hemsidan är också startpunkten för systemet, som initierar de externa systemen via ExternalsController. Denna controller startar och hanterar alla processerna av typen Plant. Webserverprocessen körs på Raspberry Pin.

4.2.3 Plant

En Plant-process per blomma hanterar all kommunikation med extern hårdvara, det vill säga Arduinon med fuktsensorn, och Tellstick Duon som hanterar bevattningen. Plantprocesserna och kommunikationerna från dem hanteras av klassen ExternalsController.

4.2.3 Arduino

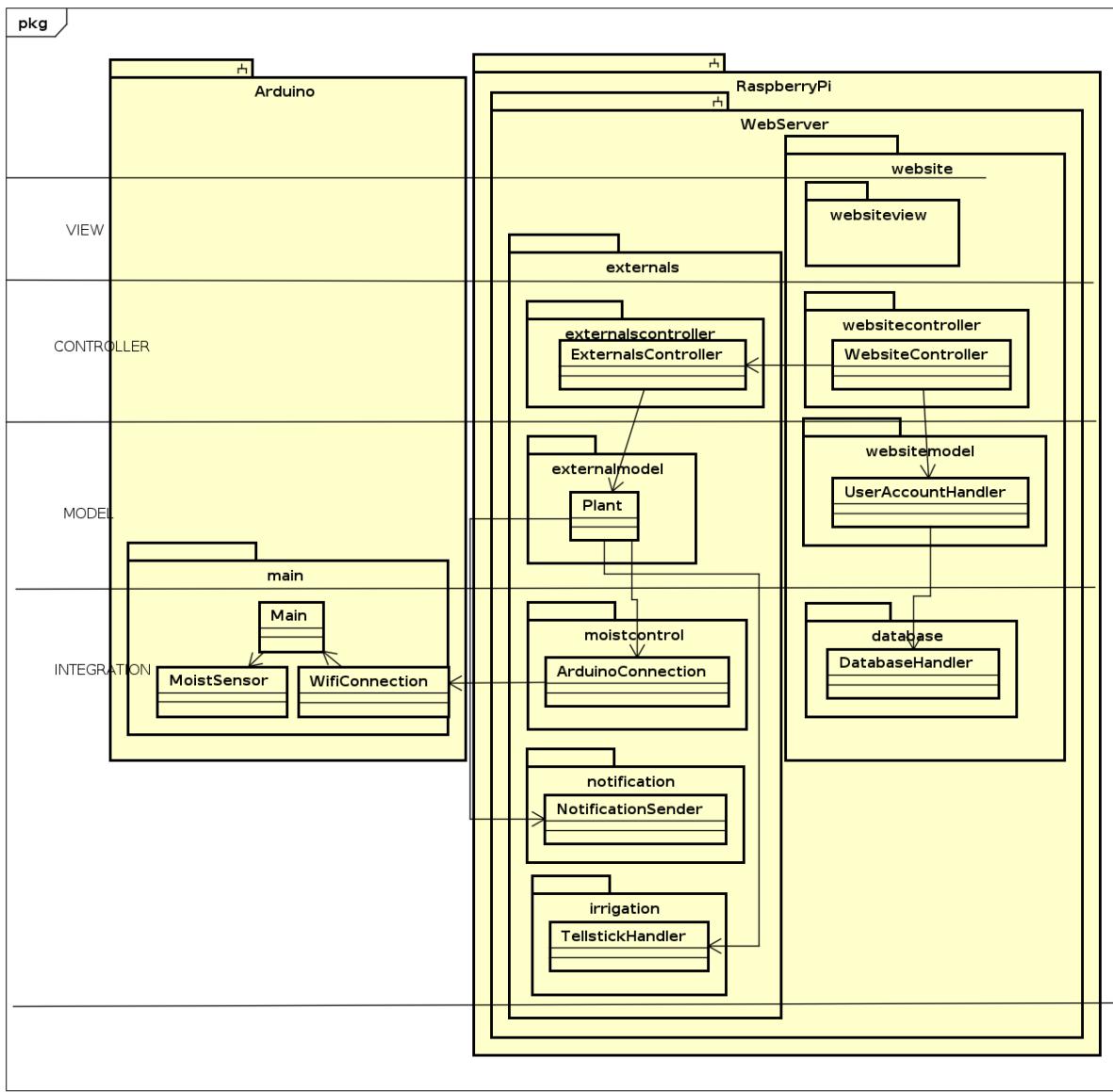
På arduinoen körs en process som läser av fuktsensorn och kommunicerar via wifi med Raspberry Pin. En Plant-process pollar denna process för nya uppdaterade fuktvärden vid vissa intervall.

5. Utvecklingsvy

5.1 Bakgrund

Utvecklingsvyn ger en mer detaljerad beskrivning av systemets kodstruktur, indelat i subsystem, lager, och klasser. Koden ska vara indelad så att den i lagom stora delar kan läggas ut på de olika utvecklarna av projektet.

5.2 Beskrivning



Figur 3. En ritning över utvecklingsvyn för systemet.

Utvecklingen av systemet sker i form av flera mindre subsystem, som var för sig är inordnat enligt MVC-mönstret (Model-view-controller). Figur 3 är en översikt över subsystemen och paketen som ingår där i. Mer detaljerad beskrivning av klassernas publika interface och implementation finns i projektets Designplan [6].

6. Fysisk vy

6.1 Bakgrund

Den fysiska vyn beskriver framförallt icke-funktionella krav såsom vilka hårdvarudelar systemet är uppbyggt av och hur de är sammankopplade.

6.2 Beskrivning.

Figur 1 visar den fysiska sammansättningen av systemet. Ett av huvuddragen i designen är att kommunikation sker trådlöst mellan systemdelarna som är nära blomman (Arduinon samt Tellstick Duon) och Raspberry Pin. Detta för att man ska kunna använda systemet till flera blommor över ett längre avstånd. Se nedan för mer detaljerad beskrivning.

6.3 Hårdvara

- Raspberry Pi 3B med Raspbian OS (+ strömkälla)
- Arduino (+ strömkälla)
- Wifi-modul ESP8266 till Arduino
- Fuktsensor kompatibel med Arduino, till exempel Art.nr 41015738 på electrokit.com (hämtad 22-05-2018).
- Tellstick Duo
- Fjärrstyrd strömbrytare kompatibel med Tellstick Duo.
- Eheim Pump Compact ON 300 eller annan vattenpump som styrs med elektricitet

6.3.1 Raspberry Pi 3B

Raspberryn är ansluten till:

- Vägguttag för el, via sladd
- LAN-port, för att ansluta till internet och få en statisk IP
- Arduinon, trådlöst via wi-fi
- Tellstick Duo, via kabel

6.3.2 Arduino

Arduinon är ansluten till:

- Raspberry Pin, via wi-fi
- Vägguttag för el, via sladd
- Fuktsensorn, via sladd

6.3.3 Fuktsensor

Fuktsensorn är ansluten till Arduinon via sladd, och dess placering ska vara i blomjorden som ska mätas.

6.3.4 Tellstick Duo

Tellstick Duons sändare är kopplad till Raspberry Pin via USB. Sändaren kommunicerar trådlöst med fjärrströmbrytaren.

6.3.5 Pump

Pumpen är kopplad till fjärrströmbrytaren, som sitter i ett elnätuttag. Pumpen ska befina sig i en behållare med vatten, och dess slang ska leda vattnet till blomjorden. När fjärrströmbrytaren sätts på, ska vattenpumpen pumpa vatten.

Bilaga 4

Design- och utvecklingsplan

Grupp 10 - FlowerPower
Hampus Pukitis Furhoff
Senast ändrad 2018-05-21

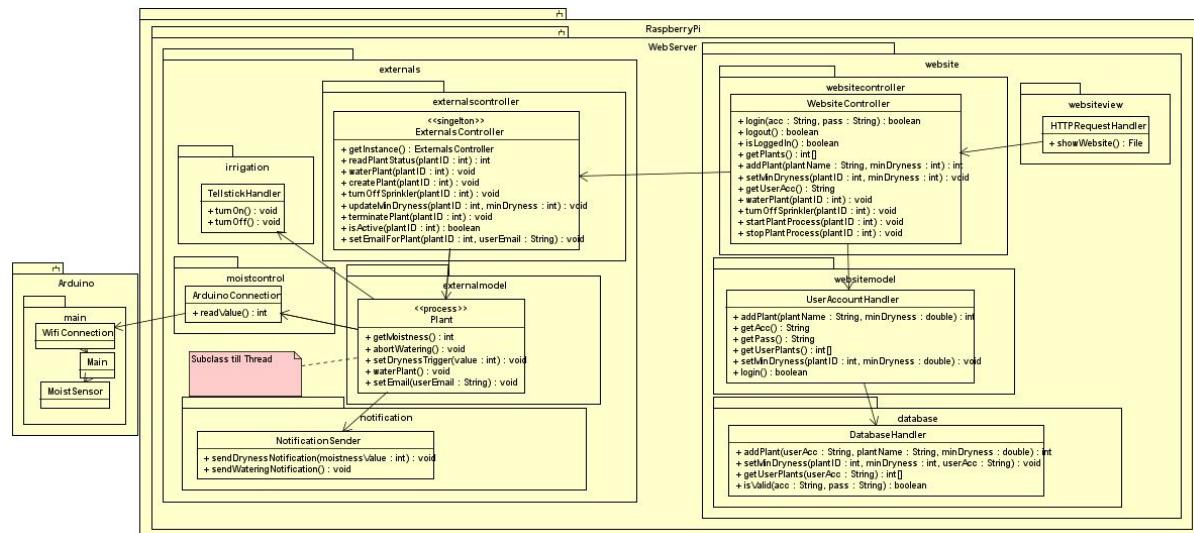
Innehåll

1. Innehåll
2. Syfte
3. Design/modell
4. Språk och framework
5. Kodstandard
6. Versionshantering

Syfte

Syftet med detta dokument är att sammanfatta hur vi jobbar med koden i vårt projekt så att alla i projektgruppen jobbar på samma sätt och så att projektmedlemmar kan kolla upp saker dom är osäkra på.

Design/modell



Språk och framework

Arduino

På vår arduino har vi kommit fram till att vi ska koda i arduinos egna miljö med arduino-språket eftersom det finns mycket stöd för det och det är lätt att komma igång med. Vi övervägde att koda i C på arduinon men kom fram till att det mest skulle krångla till saker

och att dom hasighetsförbättringar man eventuellt skulle få inte var så viktiga för oss i nuläget.

Raspberry Pi

På raspberry pi'en övervägde vi att koda vår webserver i java eftersom alla i gruppen är bekanta med det men valde istället python eftersom det ska vara väldigt lätt att lära sig och likt java. Andra skäl till varför vi valde python istället för java var för att det ska finnas väldigt mycket stöd och dokumentation just om att göra saker på en raspberry pi med python.

Frameworks

Vi kör med Flask på serversidan, detta eftersom det är lätt att lära sig, lättviktigt och det går bra att göra MVC-struktur på med det. Vi kör inte med något framework på klientsidan eftersom vi inte har så stort behov av det.

Kodstandard

Variabler och funktioner skrivas med kamelnotation med inledande liten bokstav (camelCase). Klassnamn skrivas med kamelnotation med inledande stor bokstav (CamelCase). Varje funktion bör ha förklarande kommentarer men man bör undvika kommentarer inne i funktionen, skriv istället förklarande kod och undvik hårdkodade värden. Här skriver jag bara väldigt kort så om man vill gå in mer i detalj så kan man läsa "A First Course in Object Oriented Development A Hands-On Approach" av Leif Lindbäck.

Smelly code

Smelly code gör det svårare att förstå koden och kan innehåra mycket onödigt arbete. För att förhindra smelly code så ska vi refaktorera när vi stöter på nedanstående företeelser.

Duplicated code

Om man har samma kod på flera ställen så är det ett problem eftersom om man behöver ändra i den koden så måste man ändra på flera ställen. Bättre då att refaktorera och lägga koden i en egen metod.

Long method

En metod är för lång om man som användare av metoden inte kan utläsa vad metoden gör bara på metodnamnet. Ett bra sätt att veta om en metod är för lång är om man som programmerare behöver lägga in kommentarer i metoden för att förklara vad metoden gör. Då är det bättre att bryta upp metoden i olika metoder.

Large class

På samma sätt som att en metod kan vara för lång så kan en klass vara för lång. Om en klass gör annat än det som utlovats av klassnamnet så bör man se över klassen.

Meaningless names

Om man inte använder namn som förklarar vad en variabel, metod eller klass är så blir koden snabbt mycket mer svår förstådd. Istället för att skriva tex pp så kan man skriva plantProcess och man förstår direkt vad det är för något.

Unnamed values

Om man använder sig av primitiva data så bör man lägga dessa i en variabel med ett förklarande namn, tex så bör man skriva plantID = 3; getPlant(plantID); istället för getPlant(3); eftersom det blir mycket tydligare vad man menar.

Versionshantering

Vi använder github för att versionshantera vår kod. Vår repository finns på <https://github.com/fongie/projectgaming>. Varje gång man ska jobba med en ny grej så gör man en ny branch med ett förklarande namn i camelcase. När man är klar så mergar man med test/huvud-branchen eller med en annan branch som också är klar där den nya featuren testas och till slut när alla brancher är klara och testade tillsammans merges huvud/test-branchen med master. Mergningen till master görs enbart med godkännande av utvecklingsansvarig. Om testning visar att vissa saker behöver fixas till så görs detta i en egen branch och sedan mergar man igen till test/huvud-branchen. Som sista steg så tas arbetsbranchen bort. I master ska vi endast ha fungerande och testad kod, så därför mergar vi endast till master från test/huvud-branchen.

Gul - Kalles branch
Grön - Lisas branch, huvud/test-branch
Blå - Charlies branch



FlowerPower

Test Plan

Abstract

Detta dokument är en testplan för studentprojekt vid KTH EECS. Dokumentet behandlar viktiga områden som behöver konkretiseras och fastställas så att projektgruppen kan genomföra lyckade tester genomgående under projektets gång - målet är att säkerställa att slutprodukten är kvalitetssäkrad och levererbar inom avtalad tid.

Dokumentversion, senaste överst

Date	Version	Author	Description
31/04/2018	1.3	Nicole N.O. Othman	Uppdaterat, kompletterat
18/04/2018	1.2	Nicole N.O. Othman	Uppdaterat, kompletterat
08/04/2018	1.1	Nicole N.O. Othman	Uppdaterat, kompletterat
25/03/2018	1.0	Nicole N.O. Othman	Upprättat mall för testplan

Student/författare: Nicole N.O. Othman på Företag: KTH EECS Kista

Innehållsförteckning

1	Unik identifiering	3
2	Introduktion.....	3
2.1	Dokumentets syfte	3
2.2	Dokumentets omfattning.....	3
2.3	Dokumentöversikt.....	3
3	Testöversikt – bakgrund, syfte och mål	4
3.1	Bakgrund.....	4
3.2	Syfte	4
3.3	Mål	4
4	Plan	5
4.1	Varför?	5
4.2	Vad (testobjekt)?.....	5
4.3	Var?.....	6
4.4	När?.....	6
4.5	Vem?	6
4.6	Hur (tillvägagångssätt)?	6
4.7	Hur mycket (iterationer)?.....	8
4.8	Avgränsning.....	8
5	Start- och slutkriterier	9
5.1	Kriterier för att inleda testarbetet	9
5.2	Kriterier för att avsluta testarbetet	9
6	Avbrytande- och återupptagandekriterier	9
6.1	Kriterier för att avbryta testning	9
6.2	Kriterier för att återuppta testning.....	9
7	Testdokumentation.....	10
8	Testaktiviteter	10
9	Riskanalys.....	11
10	Förändringsplan	11
11	Appendix A - Referenser	11

1 Unik identifiering

FPTP01 (FlowerPower Test Plan 01).

2 Introduktion

2.1 Dokumentets syfte

Detta dokument har som syfte att ge en ingående beskrivning av hur projektgruppen FlowerPower, löpande under projektets gång, arbetar med samtliga tester. Dokumentets målgrupp är projektgruppens medlemmar, produktägaren samt kursens examinator. Författare är Nicole Othman, testansvarig för FlowerPower.

2.2 Dokumentets omfattning

Detta dokument behandlar följande:

- Bakgrund till testning
- Testmetodik
- Ansvarsfördelning tester
- Risker

Detta dokument behandlar *inte* följande:

- Detaljer kring de olika specifika testerna inför varje sprint

2.3 Dokumentöversikt

Detta dokument innehåller följande delar:

- Projekt- och uppgiftsbeskrivning
- Riskanalys
- Förändringsplan
- Referenser

3 Testöversikt – bakgrund, syfte och mål

Detta kapitel ger en översikt av de tester som ska utföras i projektet samt beskriver syfte och mål med dokumentet.

3.1 Bakgrund

Att löpande utföra tester under ett projekt är ett vitalt och framförallt ett viktigt och nödvändigt sätt att försäkra sig om att slutprodukten fungerar som den skall och håller kvalitet. I traditionella mjukvaruutvecklings-projekt så spelar således testplanen en betydande roll.

Testplanen är ett dokument som systematiskt beskriver testnings-approachen och behövs i syfte att försäkra sig om att viktiga komponenter och frågeställningar behandlas. Dokumentet upprättas normalt sett under ett tidigt skede av projektet och används som stöd för att se till så att tester utförs noggrant, korrekt och enligt tidsschema.

FlowerPower är en projektgrupp som utformats genom aktivt deltagande i kursen II1302 Projekt och Projektmedoder. Gruppens mål är att ta fram en fungerande prototyp för ett automatiskt bevattningssystem.

3.2 Syfte

Syftet med detta dokument är att fastställa en plan gällande de tester som utförs löpande under projektets gång. Testerna används för att utvärdera om produkten uppfyller satta krav som finns beskrivna i User Stories.

3.3 Mål

Målet är att ta fram ett underlag som skall underlätta vid arbetet av testning och tydliggöra vad som behöver göras för att kvalitetssäkra produkten. Det slutgiltiga målet ska resultera i att projektgruppen tagit fram en fullt fungerande prototyp för ett bevattningssystem. Följande frågor skall kunna besvaras med hjälp av detta dokument:

- Varför?
- Vad?
- Var?
- När?
- Vem?
- Hur?
- Hur mycket?

4 Plan

4.1 Varför?

Testerna utförs i syfte att kvalitetssäkra underlaget och se till så att slutprodukten håller upp till de krav som tagits fram.

4.2 Vad (testobjekt)?

Tester med avseende på produkten (listade utan inbördes ordning):

- Testa att det går att styra/reglera fuktsensor till specifik växt.
- Testa att det går att mäta fukten i den specifika blomman, kontrollera att fukt-sensorn fungerar korrekt.
- Kontrollera att Tellstick Duo:n och bevattningsbrytaren kan kommunicera trådlöst.
- Kontrollera att kommunikation mellan Arduino och Raspberry Pi fungerar.
- Kontrollera att det går att skicka notiser till användare.
- Testa och analysera design utav webgränssnitt enligt gällande heuristik¹.

Tester med avseende på den mer tekniska delen (listade utan inbördes ordning):

- Utifrån designmodellen, som finns i ett UML-format, skall varje kodstycke som skrivs testas.
- Testa att metoderna fungerar korrekt
- Testa att felhantering (exceptions) fungerar korrekt
- Testa mot eventuella buggar
- Testa säkerheten på hemsidan. Detta inkluderar tester av metoder och funktionalitet som förebygger bland annat:
 - Cross Site Scripting (XSS)
 - Lösenordskryptering
 - Impersonation
- Kontrollera att databasen innehåller korrekt data.
- Testa att det går att mata in nya data korrekt i databasen.
- Testa att det går att hämta data från databasen korrekt.

Tester med avseende på givna kurskriterier:

- Testa att de delar som finns med i den skriftliga rapporten uppfyller kursens uppsatta krav².

¹ <https://www.nngroup.com/articles/ten-usability-heuristics/>

² <https://www.kth.se/student/kurser/kurs/II1302>

4.3 Var?

Skolans (KTH EECS Campus Kista) lokaler kommer främst att utnyttjas vid testtillfällena. Testerna kommer att genomföras på projektmedlemmarnas datorer, mjukvaran för testning varierar beroende på vad respektive projektmedlem föredrar att arbeta i.

4.4 När?

Testcykeln börjar officiellt 2018-03-19 och kommer att fortskrida fram till 2018-05-30 då projektrapporten skall inlämnas. Observera att perioden endast omfattar arbetsdagar, det vill säga att perioden inte omfattar helger samt övriga röda-/helgdagar.

4.5 Vem?

Samtliga projektmedlemmar kommer att utföra tester på det underlag (kod) som projektmedlemmen själv tagit fram. Testansvarig har det yttersta ansvaret och övriga projektmedlemmar rapporterar kontinuerligt teststatus till denne person. Det är inte förrän ett test anses vara 100% genomgånget och kvalitetssäkrat som testansvarig kan markera testen i fråga som avklarad.

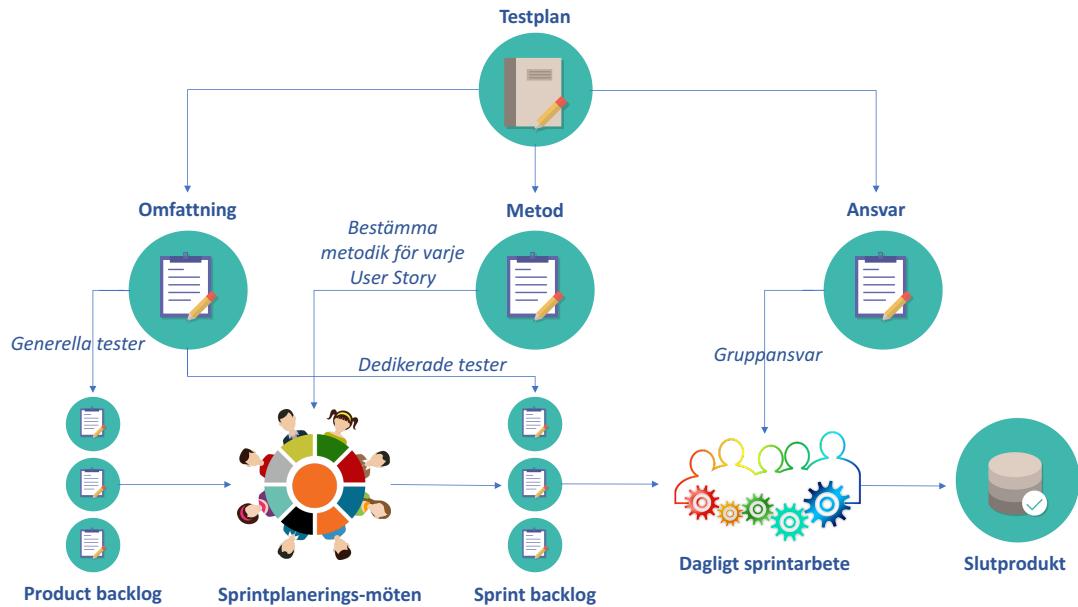
4.6 Hur (tillvägagångssätt)?

I projektet så dokumenteras kundens samtliga krav i så kallade ”user stories”, dessa prioriteras av produktägaren och sedan placeras dem på den så kallade ”product backlog”:n. Det är dessa ”user stories” som utvecklas och testerna skall säkerställa att ”user story”:n uppfyller de krav som kunden satt. De ”user stories” som skall vara avklarade inom den närmsta tiden, det vill säga innan avslutad sprint, flyttas från ”product backlog” till ”sprint backlog” och det är dessa som i huvudsak prioriteras av testarna under varje iteration.

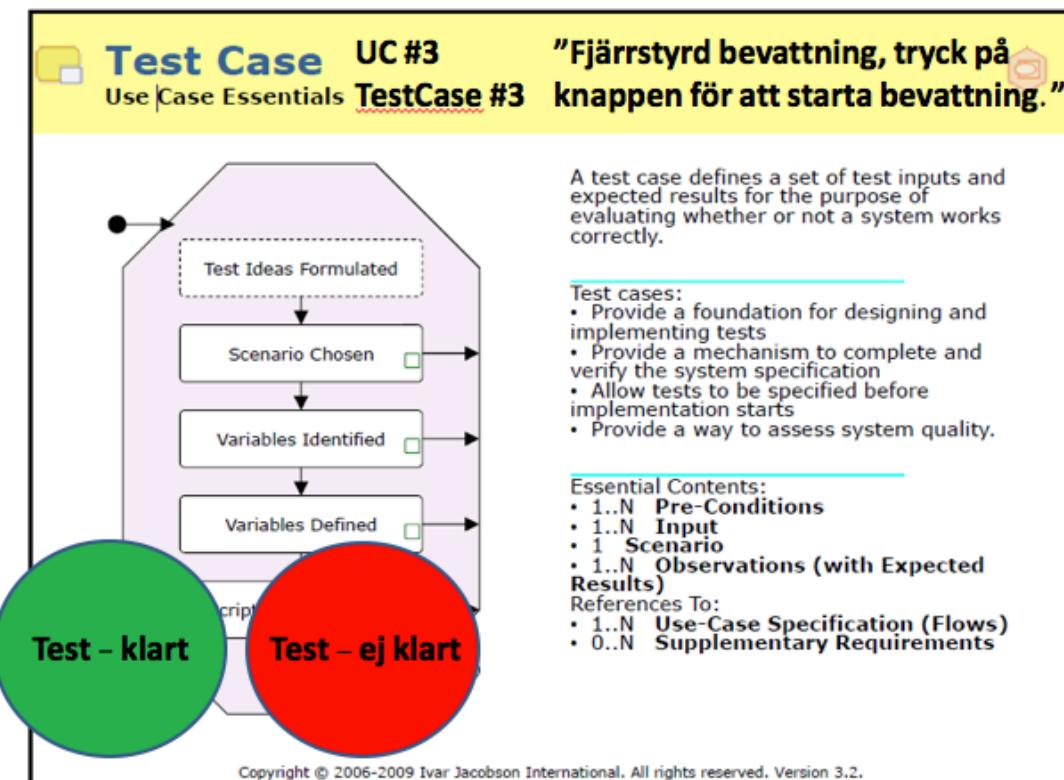
Inför varje sprint och mer specifikt under varje sprintplaneringsmöte så kommer samtliga aktuella ”user stories” att analyseras utifrån ett testing-perspektiv, denna analys ska resultera i ett så kallat Test Case. Test Case:et ska definiera ett set av test-input samt de förväntade resultaten, denna information används sedan i syfte att evaluera huruvida produkten/funktionaliteten/system fungerar korrekt.

Kopior på Test Case:n tas fram i mindre format och placeras vid arkitekturen som finns på Sprint-backloggen, dessa används för att visa att vilken komponent som testas. Röd markering avser ännu ej godkänt test och en grön markering avser godkänt test. Varje Test Case har så kallade ”Checkpoints” som beskriver olika nivåer för Test Casets status. Dessa nivåer utvärderas kontinuerligt och fylls i allteftersom caset uppfyller de krav som finns listade på Test Caset.

Vad gäller testmetodik så bestäms detta för varje test case och kan således variera beroende på vilken del av produkten som skall testas. Däremot när det kommer till källkoden, det vill säga mjukvaran och dess logik så kommer projektgruppen att skriva så kallade enhetstester (unit tests). Projektgruppen har valt att använda sig utav Python som programmeringsspråk och kommer således att använda ramverket ”pytest” för att testa att källkoden fungerar som förväntat. Då projektgruppen i huvudsak ska tillämpa en TDD-approach (Test Driven Development) så skrivs testerna innan man utvecklar själva källkoden.



Figur 1 - Översikt utav testplanen samt hur testplanen ligger till grund för projektgruppens arbetssätt när det kommer till tester. Design Nicole Othman.



Testspecifikation på baksidan

Figur 2 – exempel på framtaget Test Case, Test Case #3

Testspecifikation för User Story #3 "Fjärrstyrda bevattning"

- **VAD?** Testa att det, via hemsidan och mer specifikt via en knapptryckning, går att sätta på bevattning.
- **HUR?** När knappen är tryckt skickas information via klasserna WebsiteController → ExternalsController → ExternalsModel → Plant → Irrigation till Tellsticken som sätter på bevattningssystemet.
- **FÖRVÄNTAT RESULTAT:** När användaren tryckt på knappt på hemsidan så startas bevattningen av plantan.
- **NÄR?** Skall testas när hemsidan är utökad med en knapp som är avsedd för att starta bevattning och metoden waterPlant i klasserna WebsiteController, ExternalsController, ExternalsModel, Plant är skriven samt metoden turnOn i klassen TellstickHandler.

Figur 3 – Testspecifikation för Test Case #3, se figur 2 för Test Case.

4.7 Hur mycket (iterationer)?

Testarbetet sker iterativt och enligt den procedur som beskrivs i ovan stycke 4.6 ”Hur (tillvägagångssätt)?”. Arbetet kan i huvudsak delas in i följande faser:

- Enhetstester - för varje ny klass och dess tillhörande metoder så skrivs det enhetstester.
- Omtest – då projektgruppen arbetar enligt TDD så kommer testet att inte godkännas vid första körning. Varje projektmedlem skriver kod fram tills att testet blir godkänt.
- Regressionstest – för varje ny kod funktionalitet som läggs till så skall så kallade regressionstester göras. Detta i syfte att försäkra sig om att produkten fortfarande fungerar i sin helhet.

4.8 Avgränsning

Följande tester kommer inte att genomföras:

- Extremtester – det vill säga tester som har som syfte att testa hur produkten hanterar extrema situationer såsom till exempel vid strömbrott eller nätverksstörningar.
- Tester på beställda komponenter – det vill säga produkter som projektgruppen beställt från externa leverantörer, dessa förväntas gruppen att leverantören redan testat. Däremot testar projektgruppen att komponenter fungerar i sin helhet och tillsammans med andra delar som tillsammans utgör slutprodukten.

5 Start- och slutkriterier

5.1 Kriterier för att inleda testarbetet

Nedanstående punkter skall vara uppfyllda för att testerna skall kunna påbörjas:

- Nödvändig programvara (Python, Flask, Pip, Pytest m.m.) ska vara installerat på datorn.
- User Cases och Test Cases skall vara framtagna så att varje projektmedlem vet vad denne skall göra och testa under sprintarbetet.
- Testplanen och teststrategi framtagen och godkänd av FlowerPower.

5.2 Kriterier för att avsluta testarbetet

Nedanstående punkter skall vara uppfyllda för att testarbetet skall kunna avslutas:

- Det finns enhetstester för samtliga klasser och metoder.
- Samtliga enhetstester godkända.
- Övriga planerade testfall genomförda och godkända.

6 Avbrytande- och återupptagandekriterier

Skulle projektgruppen stöta på problem har FlowerPower tagit fram avbrytande- och återupptagandekriterier. Dessa behövs ifall gruppen skulle komma fram till situationer där de stöter på många fel eller andra allvarliga saker som gör att testarbetet inte kan fortskrida. Vid ett sådant tillfälle kommer problemet att utredas och om möjligt åtgärdas innan ett testningsarbete kan återupptas, beslutet tas gemensamt utav samtliga projektmedlemmar. Om något av nedanstående följande kriterier påträffas så avbryts/återupptas testningsarbetet.

6.1 Kriterier för att avbryta testning

- Flera viktiga komponenter, såsom till exempel Arduinon, går sönder.
- Projektmedlemmarna får ej tid till att skriva enhetstester.
- Svårigheter med att enas om ett testningsarbete.

6.2 Kriterier för att återuppta testning

- Problemet åtgärdat och resultatet av de regressionstester som genomförs visar godkänt.
- Samråd mellan testledare och övriga projektmedlemmar.

7 Testdokumentation

Följande dokument tas fram och underhålls löpande under projektets och mer specifikt testarbetets gång:

Dokument	Beskrivning	Ansvarig
Testplan	Detta dokument.	Testansvarig
Testspecifikation	Innehåller mer detaljerad information kring varje test.	Testansvarig
Test Cases	Anger teststatus och relaterad User Case.	Testansvarig
Checklista/sektion	Test-sektionen som finns på sprint-backlog, visar vilka tasks som skall testas.	Samtliga projektmedlemmar

8 Testaktiviteter

Aktivitet	Ägare
Ta fram Test Cases för varje User Case	Testansvarig
Ta fram testspecifikationer för varje Test Case	Testansvarig
Skriva Testplan	Testansvarig
Genomföra tester/skriva enhetstester	Samtliga projektmedlemmar

9 Riskanalys

Nedan beskrivs identifierade risker, listade utan inbördes ordning.

Risklista:

ID	Risk	Förebyggande åtgärd	Åtgärder vid riskutfall
R1	Lång tid att lära sig TDD	Läsa in teori	Slopa TDD-metodik
R2	Hinner inte testa innan nästa sprint	Se till att checka status kontinuerligt, hjälpa till vid tester där det behövs, skriva enkla men effektfulla tester.	Avklara test innan man påbörjar framtagning av ny funktionalitet.
R3	Otillräckliga tester	Samma som ovan samt att evaluera testresultatet och jämföra mot uppsatta krav.	Gör om gör rätt.
R4	Svårtestat	Skriver enligt TDD-metodik.	Bryta ner det som skall testas i mindre delar.

Riskbedömning:

	Hög sannolikhet			
Liten påverkan		R1	R2	Stor påverkan
			R3	
		R4		
	Låg sannolikhet			

10 Förändringsplan

Den sortens förändring som innefattar ändringar av själva produkten (vilket i sin tur påverkar testerna och testmetodiken) bestäms tillsammans under ett möte och meddelas därför på plats, beslut sker efter konsultation med samtliga projektmedlemmar samt eventuellt även kursens handledare. För ändringar kring det praktiska arbetet (det vill säga var testerna skall utföras) så kommuniceras och planläggs detta främst via Facebook Messenger om projektgruppen inte är samlad, i annat fall kommuniceras detta muntligt vid ett arbetspass.

11 Appendix A - Referenser

Kniberg H. (2015) - *SCRUM AND XP FROM THE TRENCHES*, kap. 14 "How do we do testing".

Eriksson U. (2004) – *Test och kvalitetssäkring av IT-system*.

Bilaga 6

<Organization>

FlowerPower

Front-end websida – Systemdokumentation

Abstract

Det här dokumentet beskriver dem behövde funktionerna och gränssnitten av front-end websidan.

Version History

Date	Version	Author	Description
28/05/2018	<Version1>	<Kim Säther>	Skapat teknisk dokumentation för front-end websida



 IVAR JACOBSON
CONSULTING

An Essential Unified Process Document

Innehåll

1	Introduktion.....	3
1.1	Dokumentets Syfte.....	3
1.2	Dokumentomfattning	3
1.3	Dokument Översikt.....	3
2	Obligatoriska Funktioner	Error! Bookmark not defined.
3	Implementeringsbegränsningar	4
4	Tillhandahållna Gränssnitt.....	5
5	Obligatoriska Granssnitt	5
6	Intern Struktur.....	5
	Referenser	6

1 Introduktion

1.1 Dokumentets Syfte

Syftet med detta dokument är att beskriva beteendet och gränssnittet för komponenten <Front-end websida> för att:

- Fastställa beroenden som finns med andra komponenter
- Stödja den oberoende utvecklingen av denna komponent
- Ge en grund för att testa att komponenten fungerar efter behov.

1.2 Dokumentomfattning

Omfattningen av detta dokument är begränsat till:

- Specifikationen av de nödvändiga funktionerna för denna komponent
- Specifikationen av gränssnitten som denna komponent måste genomföra
- Listar de gränssnitt som krävs för denna komponent
- Ange komponentens interna struktur och design.

Dokumentet omfattar inte följande:

- Enhetstester som krävs för att verifiera att denna komponent överensstämmer och utför det som är specificerat
- Specifikationen eller genomförandet av andra komponenter som är beroende av denna komponent eller denna komponent beror på.

1.3 Dokument Översikt

Det här dokumentet innehåller följande sektioner:

- **Obligatoriska Funktioner** - funktioner som krävs av komponenten när det gäller de ansvarsområden som den måste uppfylla och relaterade krav.
- **Implementeringsbegränsningar** - begränsningar som implementeringsmiljö, genomförandespråk, relaterade mönster, riktlinjer och standarder.
- **Tillhandahållna gränssnitt** - de gränssnitt som komponenten måste tillhandahålla specificerad med avseende på deras signaturer och begränsningar.
- **Obligatoriska gränssnitt** - listar de gränssnitt som komponenten beror på.
- **Intern struktur** - specificerar komponentens interna designstruktur med avseende på dess beståndsdelar och deras relationer.
- **Referenser** - ger fullständiga referensuppgifter för alla dokument, böcker och länkar som refereras av detta dokument.

2 Obligatoriska Funktioner

I det här avsnittet anges de funktioner som krävs för komponenten i termer av:

- De beteende eller tjänster som den måste tillhandahålla för sina kunder
- De icke-funktionella kraven som är relaterade till var och en av dessa
- Övriga krav på underhåll, dokumentation, återanvändning, kontroll eller loggning som gäller komponenten.

Dessa funktionella funktioner tillhandahåller websidan:

- Visa blommans fuktvärde
- Ange fuktkänslighetsvärde för notis
- Vattna blomman

Dessa icke-funktionella funktioner tillhandahåller websidan:

- Logga in
- Logga ut

Koden behöver inte underhållas annat än vid genomförandet av ändringar. Det finns inget krav på dokumentation av koden som skrivs i språket HTML eller CSS, är det ändå önskvärt med en kommentar. Det får inte finnas någon duplicerad kod i systemet. Systemet ska också vara säkert med kontroller på sådant som kan skada användaren eller systemet, exempelvis inmatningen av användaruppgifter ska kontrolleras och krypteras. Systemet har inget som behöver loggas i nuläget, men det kan komma att byggas till senare.

3 Implementeringsbegränsningar

Detta avsnitt anger de begränsningar som påverkar komponentens genomförande, såsom:

- Utvecklingsmiljöerna inom vilka den måste fungera
- Vilket språk det ska skrivas i
- De ramverk den ska fungera inom
- Designmönster den ska följa
- Riktlinjer som ska följas, inklusive modelleringsriktlinjer, kodningsriktlinjer och standarder
- Miljömässiga begränsningar som minnesfotavtryck och andra resursanvändningsbegränsningar.

Den använda utvecklingsmiljön är Visual Studio Code men det finns ingen anledning att begränsa vilken ide som ska användas, så vill man fortsätta utveckla systemet så kan man välja ide efter egen smak och tycke. Språket som används för hemsidan är HTML och det grafiska är med CSS. Sedan används även en fil som är gränssnittet mot back-end websidan och den är skriven i python. Den filen ingår inte i front-end websidan, men den nämns här för att det kan vara bra att ha koll på den om man

jobbar med front-end websidan. Front-end websidan ska fungera i ramverket flask och ska tillsammans med back-end websidan följa designmönstret MVC. Riktlinjer som bör följas är att koden bör vara inkapslad, ha hög sammanhållning och låg koppling [1]. Front-end websidan är inte utvecklad på ett miljövänligt sätt då systemet måste ladda all information på nytt varje gång och inte bara det som är nytt på sidan. Detta är något att jobba med framöver om man vill fortsätta utveckla produkten.

4 Tillhandahållna Gränssnitt

I det här avsnittet anges de gränssnitt som komponenten måste tillhandahålla när det gäller deras signaturer och begränsningar.

Grafiskt gränssnitt och gränssnitt mot back-end websidan tillhandahålls av front-end websidan. Det grafiska är det som användaren ser och det är designat med avseende på dem 10 användbarhets heuristikerna för användargränssnitt [2]. Det som gränsar till back-end är skapat av back-end utvecklaren som kommunicerade med front-end utvecklaren för att få dem olika delarna av systemet att fungera tillsammans.

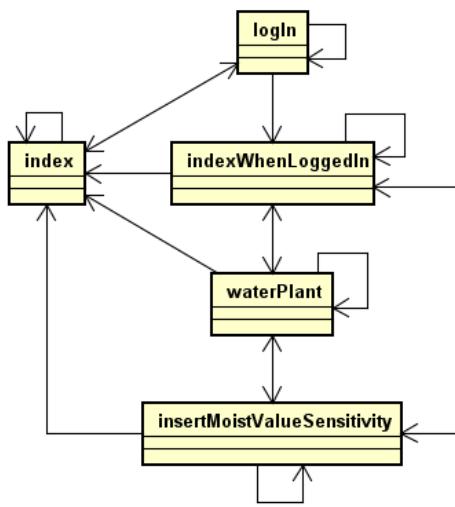
5 Obligatoriska Gränssnitt

I det här avsnittet anges de gränssnitt som komponenten beror på för att uppfylla sitt angivna beteende.

För att alla funktioner ska fungera krävs det att hela systemet är på plats. Det innebär att alla fem komponenter, front-end websida, backend, externals controller, external integration och arduino kommunicerar och att en användare använder det grafiska gränssnittet för att styra systemet.

6 Intern Struktur

Det här avsnittet anges komponentens interna designstruktur i termer av dess ingående komponenter och hur dessa samverkar för att leverera dem nödvändiga komponentbeteenden.



När front-end websidan öppnas så visas först index-sidan, sedan kan man klicka sig vidare till logIn-sidan. Från logIn-sidan kan man antingen navigera tillbaka till index-sidan eller till indexWhenLoggedIn-sidan. Därifrån kan man navigera sig till sidorna waterPlant, insertMoistValueSensitivity eller tillbaka till index. När man sedan står på waterPlant eller insertMoistValueSensitivity så kan man navigera fritt mellan dem eller till indexWhenLoggedIn eller index, se figur 1.

Med denna design får man ett användarvänligt och lättanvänt system, det är lätt att veta var man ska navigera utifrån vad man vill göra.

Figur 1. UML-diagram över komponent

Referenser

Använd det här avsnittet för att ge fullständiga referensuppgifter för alla dokument, böcker och länkar som refereras av detta dokument.

- [1] Leif Lindbäck 4 May 2018. Kapitel 5.2 Design Concepts.
A First Course in Object Oriented Development A Hands-On Approach
KTH, Kungliga Tekniska Högskolan
- [2] Jakob Nielsen 1 Januari 1995. 10 Usability Heuristics for User Interface Design
<https://www.nngroup.com/articles/ten-usability-heuristics/>
NN/g, Nielsen Norman Group

FlowerPower Project

Backend Component Description

Abstract

This document describes the required behavior and interfaces of the Backend component.

Version History

Date	Version	Author	Description
27/05/2018	1.4	Nicole Othman	Final update
18/05/2018	1.3	Nicole Othman	Updated with component description
12/05/2018	1.2	Nicole Othman	Updated with component description
10/05/2018	1.1	Nicole Othman	Layout established

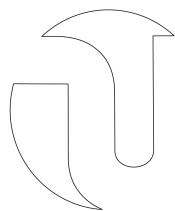


Table of Contents

1	Introduction.....	3
1.1	Document purpose	3
1.2	Document Scope	3
1.3	Document Overview	3
2	Required Behavior	4
3	Implementation Constraints	5
4	Required/Provided Interfaces.....	6
5	Internal Structure	7
6	Tests	9
	Appendix A - References.....	9

1 Introduction

1.1 Document purpose

The purpose of this document is to describe the behavior and interfaces of the backend component in order to:

- Enable the implementation effort to be sized and managed
- Establish the dependencies that exist with other components
- Support the independent development of this component
- Provide a basis for testing that the component operates as required.

1.2 Document Scope

The scope of this document is limited to consideration of:

- The specification of the required behaviors for this component
- The specification of the interfaces that this component must implement
- Listing the required interfaces that this component depends upon
- Optionally specifying the internal structure and design of the component.

This scope of this document does not include consideration of:

- The unit tests that required to verify that this component conforms and performs to its specification
- The specification or implementation of any other components that depend upon this component or that this component depends upon.

1.3 Document Overview

This document contains the following sections:

- **Required Behavior** – behavior that is required from the component in terms of the responsibilities that it must satisfy and related requirements
- **Implementation Constraints** – constraints like implementation environment, implementation language and related patterns, guidelines and standards
- **Internal Structure** – optionally specifies the internal design structure of the component in terms of its constituent components and their relationships
- **References** – provides full reference details for all documents, white papers and books referenced by this document.

2 Required Behavior

Every user interaction on the website has to spark an event, this is where backend coding plays an important role. There are currently three different user actions that can be performed on the website, these required functions are:

- Login functionality – a user must be able to login
- Watering functionality – a user must be able to start the watering of a plant
- Set minimum dryness level functionality – a user must be able to update the minimum dryness level

Among these functions there's a current dryness level displayed and the user is able to view different pages on the website. All these actions require a backend component that can check whether or not an action has been made.

Whenever a user navigates to the login site and performs a login, typed user credentials such as username and password needs to be fetched and matched with a record in the database. If the entered data matches with the database record, the user is routed to a logged in site, from here the user can perform actions such as water a plant and set the minimum dryness level.

If the user presses a button specifically designed for watering a plant, it'll activate the watering function and a notice will be shown indicating that the plant has been watered. If the user presses a button specifically designed for setting the minimum dryness level, it'll update the current value and a notice will be shown indicating that the minimum dryness level has been set.

3 Implementation Constraints

For this project, we've used an architectural pattern called MVC (Model-View-Controller). The view is responsible for the UI (User Interface), i.e. responsible for displaying information for the user. The controller is responsible for managing the user inputs, i.e. interpreting them in order to understand what should be done and act accordingly. Finally, the model is the layer in which all the business logic is added.

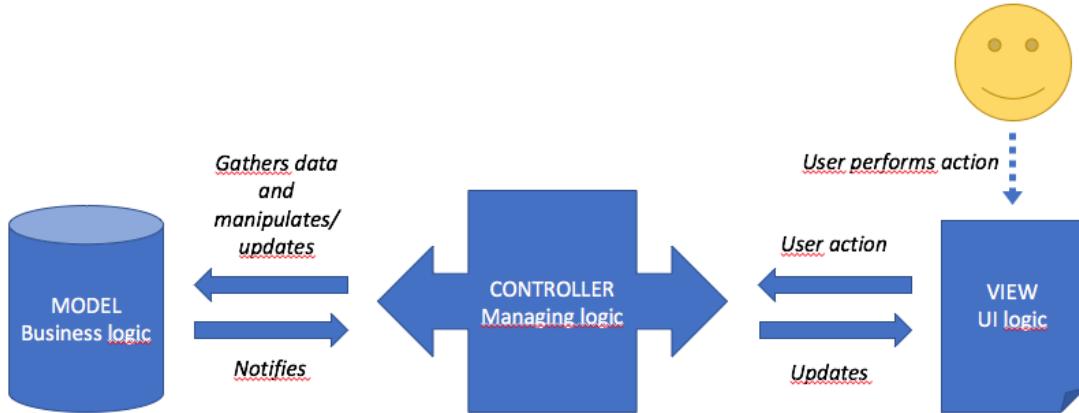


Figure 1: MVC design pattern, design Nicole Othman

The backend component consists therefore of four different packages with their belonging classes:

- database
Consists of a class named “DatabaseHandler”
- websitecontroller
Consists of a class named “WebsiteController”
- websitemodel
Consists of a class named “WebsiteModel”
- websiteview
Consists of a class named “WebsiteView”

The source code of the backend component is written in Python. In order to build the web application and running the server I've used Flask [1], a micro web framework for Python.

The database for storing user credentials is created in MySQL [2]. Our database is called “flowerpower” and the table in which we've stored the user credentials is called “Användare” and it consists of two columns named “namn” and “lösenord”. The column named “namn” is the primary key, this value is unique and therefore used to uniquely identify the rows in the table. AN1 in the conceptual model (figure 2 below) indicates that the two columns together are a composite key, this means that the combination of the two columns can also be used to uniquely identify a row.

Database flowerpower

Användare		
namn	1...1	UNIQUE AN1
lösenord	1...1	AN1

Figure 2: Conceptual model of the database

By installing and configuring the MySQL Connector/Python, which is a self-contained Python driver used for communication with MySQL servers [3], we're able to create SQL-queries towards our database.

4 Required/Provided Interfaces

In order to get the backend component to work completely it has to establish a connection with a database (MySQL), the database is used to fetch user records whenever a user performs a login. It also has to be able to establish a connection to the web server, which is a Python-powered web server built with the framework Flask. Finally, in order to react to user actions and perform accordingly, the backend component must be able to synchronize with the source code of the frontend component as well as the source code in the externalscontroller-package.

As this project is built using the MVC-structure, the backend component provides an interface between the frontend component and the more internal components such as the externalscontroller-class, these components are described more further in their respective documents.

5 Internal Structure

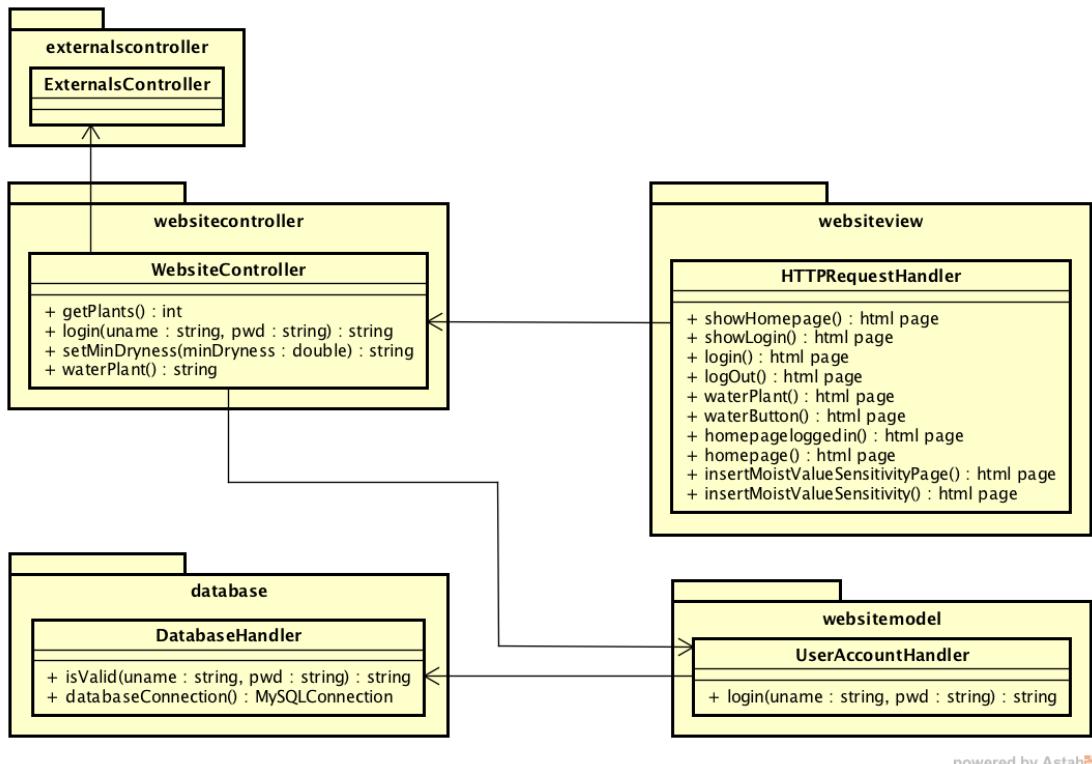


Figure 3: UML design of the backend component

powered by Astah

HTTPRequestHandler-class

The key package in the backend component is the “websiteview”, this package contains a class called “HTTPRequestHandler” and whenever a user performs a user action on the website it sparks an event in that specific class. As mentioned in section 4 Implementation constraints, the framework Flask is used for managing the website, which is designed by our frontend-developer.

By the use of the route()-decorator we’re able to bind a function to a specific URL, i.e. the route()-decorator tells the Flask-framework what URL should trigger our function. In order to be able to shift between the different web pages I’ve used a Flask-function called render_template() which, instead of returning hardcoded HTML, returns a static HTML-file. This function also enables data to be passed to the specific HTML-file which we’re taking advantages of by passing values such as the current moistness value to be displayed on the website. The return_template()-function is used in all methods inside the “HTTPRequestHandler”-class.

As mentioned in section 2 Required Behavior, we’ve three main functions that can be performed by a user. In the source code of this class I’ve added an if-statement that checks if a specific login-button has been pressed. When the button is pressed, the typed username and password are stored in variables which in return are passed as arguments in a method-call to the WebsiteController-class. If the result from the method call matches with the typed username it indicates on a successful login and a session-

variable is set to true, when the user then presses the logout-sign on the website, the session-variable is set to false. Similarly, there's an if-statement that checks whether or not the watering-button and the button for setting the minimum dryness value has been pressed.

In order to implement some security-aspects I've added a session-variable (mentioned above) which is set to true whenever a user is logged in. With the help of an if-statement, a control of the current value of the session-variable is made before rendering the correct web page. This prevents unauthenticated users from navigating to pages from where they can perform tasks such as watering the plant and setting the minimum dryness value.

All method calls in this class are made towards methods in the WebsiteController-class.

WebsiteController-class

This class is responsible for interpreting the user actions in order to understand what should be done and act accordingly. As there are currently three functions a user can perform from the website and a value indicating a plant's current moistness value, naturally there are four corresponding methods in this class.

The getPlants()-method calls upon a method in the ExternalsController-class which returns the current moistness value, this value is returned to the view which in turn renders a HTML-file with the specific data.

The login()-method takes two arguments, a username and a password, and makes a method call to a method in the UserAccountHandler-class. The result from this call is returned to the view which in turn renders a HTML-file with the specific data.

The setMinDryness()-method takes one argument, which is the value that the user has typed in, and calls upon a method in the ExternalsController-class. setMinDryness()-method returns a string value to the view which in turn renders a HTML-file with the specific data.

The waterPlant()-method calls upon a method in the ExternalsController-class. waterPlant()-method returns a string value to the view which in turn renders a HTML-file with the specific data.

UserAccountHandler-class

The UserAccountHandler consists of a method called login() which takes two arguments, a username and a password. These are passed as arguments to a method call in the DatabaseHandler-class which returns a string value. This string value is returned to the view which in turn renders a HTML-file with the specific data.

DatabaseHandler-class

This class is separated from the model-layer since it makes calls to an external database. The class consists of two methods, the `isValid()`-method and the `databaseConnection()`-method. The `databaseConnection()`-method tries to set up a connection and establish a session with the MySQL server. If the connection is successful, a `MySQLConnection` object is returned.

What the `isValid()`-method does is the following: It tries to establish a connection to the MySQL server, with the returned object from the `databaseConnection()`-method call it makes a method call to the `cursor()`-method in the `MySQLCursor` class. The `MySQLCursor` class instantiates an object that can execute SQL-queries, this object is then used when creating a SQL-query towards our database. The SQL-query fetches a username for the given password and if the fetched username matches with the typed username then the method will return the username in string format, otherwise it'll return a string with the value 'False'. Before returning the result, the connection is closed.

6 Tests

Throughout this project, we've had the ambition of writing code according to TDD (Test Driven Development) [4]. This simply means that the tests are written before the actual source code and as the backend component consists of four classes, naturally there are four test-files for each class and their belonging methods.

When working with the tests we've used a Python framework specially designed for tests called `pytest` [5]. Pytest makes it easy to build simple and scalable tests, the framework also enabled us to implement automated tests which was extremely time saving when performing regression tests whenever new functionality was added.

Appendix A - References

- [1] *Flask* (fetched 27/05/2018)
<http://flask.pocoo.org/docs/1.0/>
- [2] *MySQL* (fetched 27/05/2018)
<https://www.mysql.com>
- [3] *MySQL-connector* (fetched 27/05/2018)
<https://dev.mysql.com/doc/connector-python/en/>
- [4] Ian Sommerville, chapter 8.2 "Test-driven development", Software Engineering, Ninth Edition, Addison Wesley, 2011
- [5] *Pytest* (fetched 27/05/2018)
<https://docs.pytest.org/en/latest/contents.html>

Bilaga 8

Tekniskt Dokument - ExternalsController

FlowerPower (Grupp 10)

Hampus Pukitis Furhoff - hfurhoff@kth.se

Datum	Version	Författare	Beskrivning
2018-05-22	1.0	Hampus Pukitis Furhoff	Första version

Innehåll

1. Bakgrund
2. Syfte
3. Nuvarande design
4. Krav på komponenten
5. Testning
6. Gränssnitt
7. Beroende
8. Intern struktur
9. TODO

1. Bakgrund

Vi i grupp 10 har jobbat med en produkt i några veckor nu som vi kallar FlowerPower. FlowerPower är en produkt som ska hjälpa privatpersoner eller företag att ta hand om sina växter genom att avläsa fuktvärden från växter och skicka emailnotis när växten håller på att torka ut. På en hemsida ska man även kunna se dessa fuktvärden samt sätta av och på en bevattningspump. Vi har därför tänkt att på vår server där vi har vår hemsida ska vi även ha ett antal trådar som håller koll på dom olika planterna och det är där som behovet för ExternalsController kommer in. ExternalsController är tänkt att vara den komponenten som håller koll på dom olika plant-trådarna och som övriga komponenter måste gå igenom för att kunna utföra operationer på plant-trådarna.

2. Syfte

Syftet med detta dokument är att det ska finnas samlad information om komponenten ExternalsController så att vem som helst ska kunna använda komponenten eller utveckla vidare på den.

3. Krav på komponenten

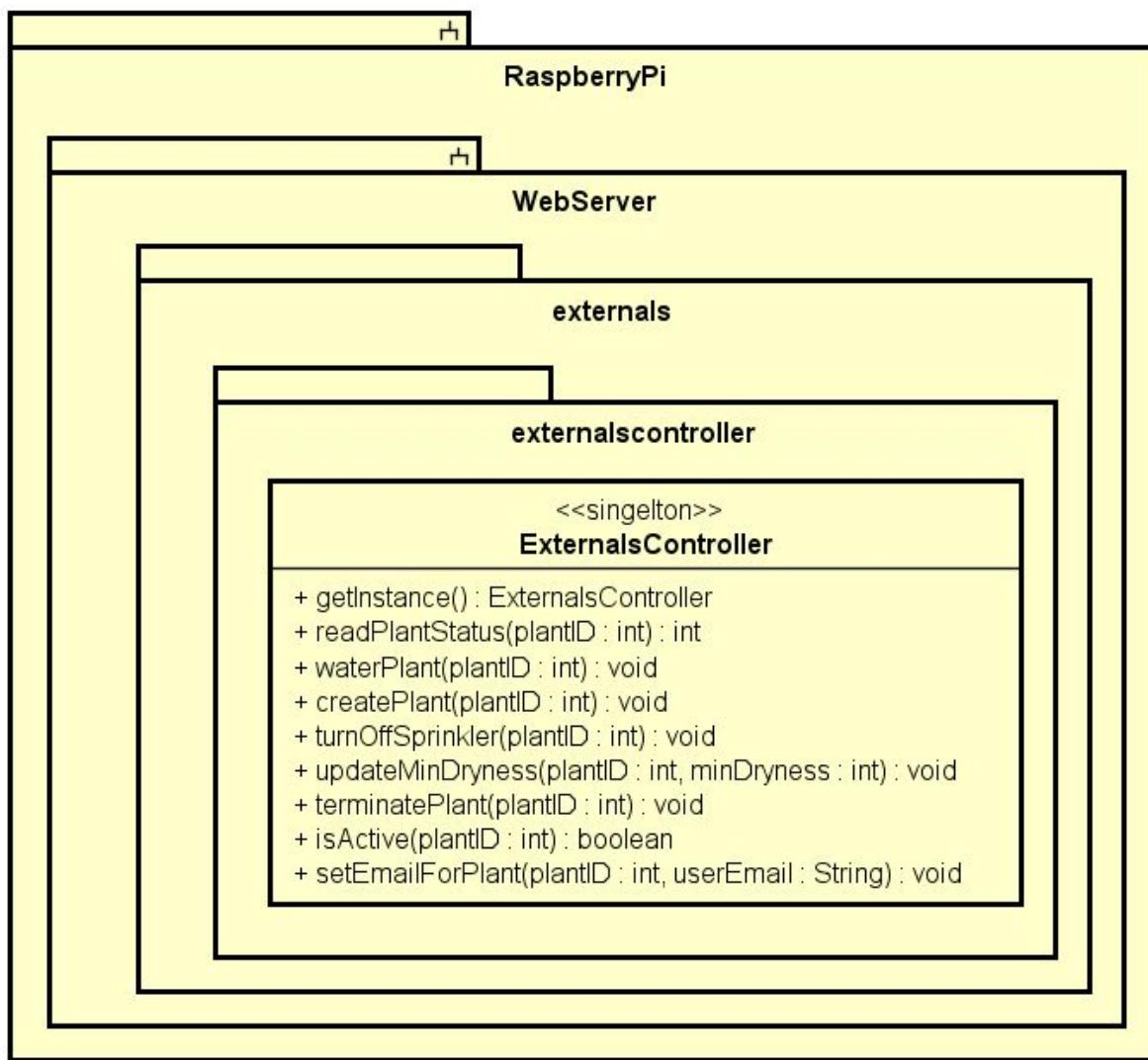
Användarkrav

Komponenten ska kunna hantera flera plant-trådar och göra operationer på dessa så att någon anropare av dess metoder ska kunna utläsa plantstatus, starta och stoppa plantprocesser, ändra när och till vem notis skickas när plantan är torr, samt styra bevattning.

Utvecklingskrav

Komponenten skrivs i python 3.5 för att köras med en flask-server på en RaspberryPi 3B med operativsystemet Raspbian.

4. Nuvarande design



Det är utifrån denna design som utveckling har skett hitintills. Till en början resonerade vi om att även göra ExternalsController till en egen tråd som skulle köras oberoende av servern men det visade sig att det inte var nödvändigt utan att det funkade ändå.

5. Testning

För varje metod skrivs minst ett unit-test som körs med ramverket PyTest. Detta är gjort med alla nuvarande implementerade metoder och ska göras med alla ännu icke implementerade metoder.

6. Gränssnitt

Som man kan se designen i under rubrik 3 så är det ett antal publika metoder som andra komponenter förlitar sig på.

- getInstance() - statisk metod som returnerar den enda instansen av klassen.
- readPlantStatus(plantID) - returnerar plantans nuvarande fuktvärde. Om plantan inte blivit skapad tidigare så skapas den också.
- createPlant(plantID) - skapar en ny planta.
- waterPlant(plantID) - påbörjar bevattning av plantan.
- turnOffSprinkler(plantID) - stänger av bevattning till plantan.
- updateMinDryness(plantID, minDryness) - sätter ett nytt börvärdet för plantans fuktvärde då den skickar notis till användaren.
- terminatePlant(plantID) - stoppar den angivna plantprocessen.
- isActive(plantID) - returnerar true eller false beroende på om det finns en aktiv process för den angivna plantan.
- setEmailForPlant(plantID, userEmail) - anger till vilken email som notis ska skickas när den angivna plantan är torr.

7. Beroenden

Dom beroenden som komponenten har är gentemot plant-modulen i projektet och en av pythons standardmoduler threading.

8. Intern struktur

Komponentens huvudsakliga uppgift är i botten att hålla reda på dom olika plant-trådarna. Hur detta görs är genom att man har gjort klassen till en singelton så att den enda instansen som finns skapar alla andra plant-trådar och sparar dom i en av pythons standard-datastrukturer, dict, som är en slags key-value-map. När någon annan då vill utföra någon operation på en planta så plockar externalscontroller då fram den plant-tråden som operationen ska utföras på och utför sedan operationen på den plant-tråden.

9. TODO

Just nu finns det inga fler publika metoder som behöver implementeras eller något som är oklart. Om man i framtiden vill ändra på designen så är det fritt fram men då får man se till att skriva nya tester och metoder baserat på dom ändringar som görs, samt uppdatera detta dokument.

FlowerPower Project

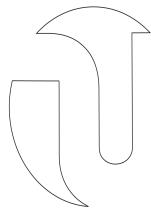
External Integration Component

Abstract

This document describes the External Integration component in the FlowerPower project, its provided and required interfaces, and its dependencies and implementation constraints.

Version History

Date	Version	Author	Description
18/05/18	1	Max Körlinge	First complete version.



An Essential Unified Process Document

Table of Contents

1 Introduction.....	3
1.1 Document Purpose.....	3
1.2 Document Scope.....	3
1.3 Document Overview.....	3
2 Required Behavior.....	4
3 Implementation Constraints.....	4
4 Provided Interfaces.....	5
5 Required Interfaces.....	5
6 Testing.....	5
7 Internal Structure.....	6

1 Introduction

1.1 Document Purpose

The purpose of this document is to describe the behavior and interfaces of the External Integration component in order to:

- Establish the dependencies that exist with other components
- Describe the dependencies that exist with external hardware
- Support further development of this component
- Describe how the component can be tested

1.2 Document Scope

The scope of this document is limited to:

- Specifying the required behaviors for this component
- The specification of the interfaces that this component must implement
- Listing the required interfaces that this component depends upon
- Listing the dependencies that must be in place for the component to function
- Describe the internal structure of the component using UML

This scope of this document does not include consideration of:

- Specifications of the unit tests that are implemented to test the component
- The specification or implementation of any other components that depend upon this component or that this component depends upon

1.3 Document Overview

This document contains the following sections:

- **Required Behavior** – behavior that is required from the component.
- **Implementation Constraints and Dependencies** – constraints like implementation environment, implementation language, architectural patterns, code guidelines and standards, dependencies on external hardware, and code dependencies.
- **Provided Interfaces** – the interfaces that the component must provide.
- **Required Interfaces** – the interfaces that the component depends upon
- **Internal Structure** – describes the internal structure of the component using UML.
- **References**

2 Required Behavior

The External Integration component of the FlowerPower system handles the Plant class abstraction and all its communication with external hardware, as well as sending notifications via email. This means that for each plant connected to the system, you are able to instantiate a new Plant object, which runs in its own thread, rely on it to handle internal logic concerning care of this particular plant, and you call upon the object to interact manually with the hardware connected to the plant.

The externals which the component is responsible for handling are:

- Moist readings from the Arduino component.
- Watering (turning the pump on and off) via the Tellstick Duo hardware component.
- Sending notifications on plant status via email.

The component is required to enable:

- Getting updated moist readings from the plant at reasonable intervals
- Turning on the watering pump
- Turning off the watering pump
- Sending an email notification when the plant is drying up
- Setting what email to send notifications to
- Setting at which level of dryness the notification will be sent

Non-functional requirements on the component:

- Being able to run the component in a separate thread, to enable a parallel system handling several plants in different threads at the same time.

3 Implementation Constraints

The component is implemented in Python and must be able to run on a Raspberry Pi 3B using the Raspbian OS. Since the component is responsible for handling external communication and integration, it has dependencies on external hardware:

- A Tellstick Duo and a compatible electric switch, with the telldus-core Linux package installed on the Raspberry Pi [1].
- An electric water pump connected to the Telldus switch.
- An email account to receive notifications from.
- The Arduino component with the moist reading sensor.

The component also has several dependencies on Python libraries, listed here as Class Name : Dependency name:

- Plant: threading, time, requests
- TellstickHandler: tellcore-py
- ArduinoConnection: requests

- NotificationSender: SMTPLib

It is recommended to handle the project's dependencies using the Python tool PipEnv. Guidelines on how to use this tool is provided in Attachment 1: Virtualenv Plan at the end of this document, a guide written for the team by the author of this document. In short, the tool provides easy command line operations which handle Python's virtual environments and code dependencies.

The component is written in Python following the code standards stated in the Design plan, which can be found in the Github repository or in the final report of the project.

The component is architecturally placed within a MVC layered design pattern, where its classes belong to the Model and Integration layers, more info can be found in the Architecture description, to be found at the github repository or the final report of the project. As such, all communication has to be made by the Plant class, which belongs to the Model layer, calling on the other classes, which all belong in the Integration layer, and no method calls are allowed the other way around.

4 Provided Interfaces

This section specifies the interfaces that the component must provide in terms of their signatures and constraints.

- start() method to start the component in new thread.
- getMoistness() method to get the current moist value from plant.
- waterPlant() method, to water the plant.
- abortWatering() method, to stop ongoing watering.
- setDrynessTrigger((int) value) method, to set at which moistness value the component will send a notification.
- setEmail((string) userEmail) method, to set which email to send a notification to.

5 Required Interfaces

The component requires an IP address to request the current moistness value from the Arduino component. It must receive this value in an object from the module *requests*, and the value must be a *string*.

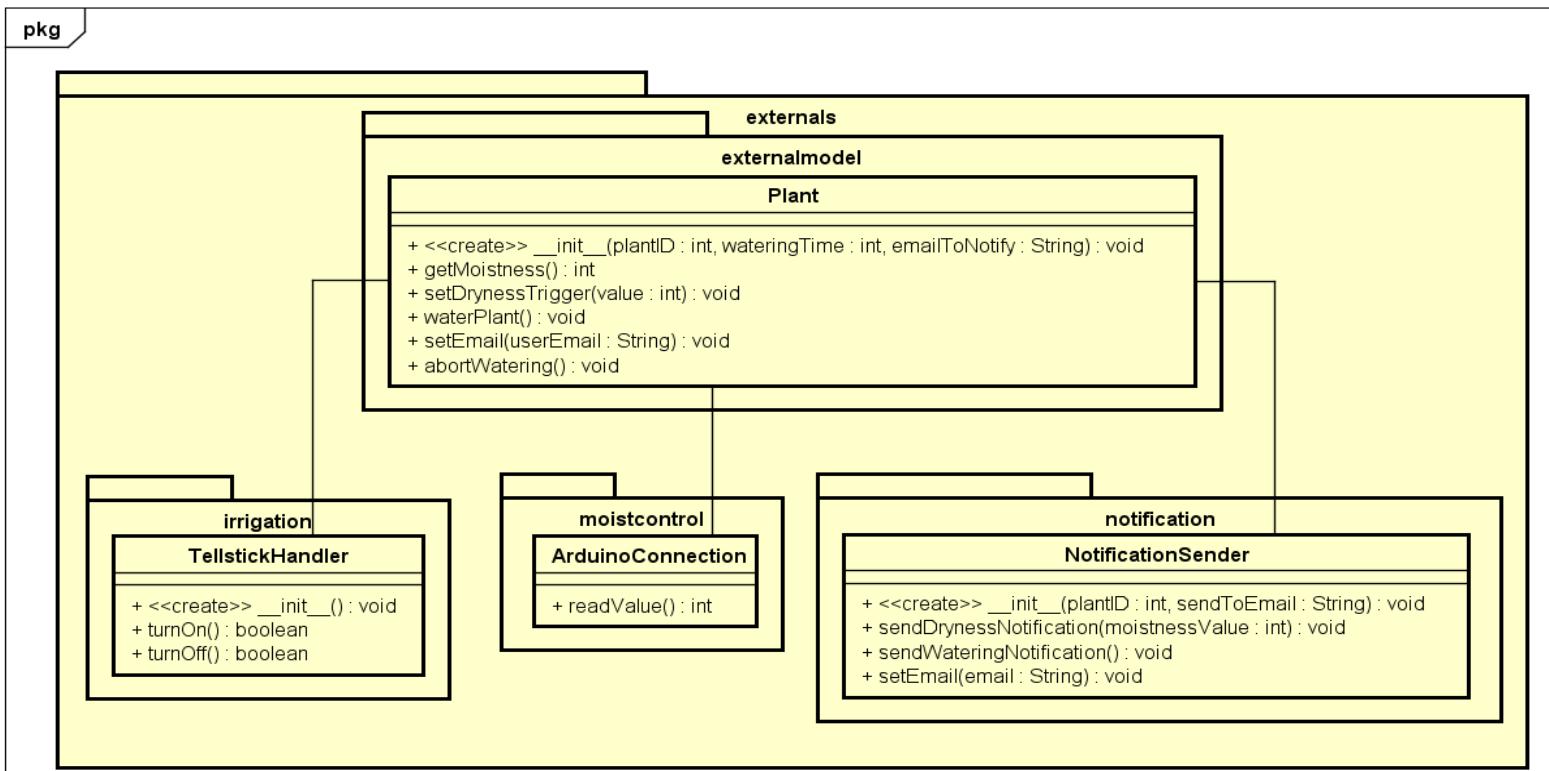
The component has no other interfaces it depends on.

6 Testing

The component is tested using unit tests and the PyTest framework. Unit tests are fully implemented for all methods and must continue to be so if development would continue. See the project's Test Plan provided at the github repository or in the final report of the project for more information.

7 Internal Structure

This section describes the internal structure of the component using a class diagram (Figure 1).



powered by Astah

Figure 1. A class diagram describing the internal structure of the External Integration component.

Appendix A - References

- [1] Telldus Developer Wiki. (2018). *Tellstick Installation Ubuntu*.
<http://developer.telldus.com/wiki/TellStickInstallationUbuntu> (hämtad 2018-05-18).

Virtualenv-plan

Syfte

Detta dokument ska ge en gemensam plan för hur vi hanterar pythons virtual environments i projektet FlowerPower.

Introduktion

PipEnv är en python-modul som du installerar i din globala pythoninstallation för att hantera virtual environments. En virtual environment är en lokal pythoninstallation som har sina egna moduler (bibliotek). Det används till exempel när man ska utveckla ett program och vill installera bibliotek och moduler för att använda i det programmet, men man vill inte att alla dessa moduler och bibliotek ska installeras lokalt på ens ”globala” pythoninstallation. Om man installerar fel grejer globalt kan det leda till problem med operativsystemet. Man kan även använda dessa virtualenvironments senare för att någon annan person enkelt ska kunna installera samma moduler och bibliotek för att kunna köra din kod på en annan dator.

Installation

Installation kräver Python3.x och pip(3), som är pythons pakethanterare. Se <http://docs.python-guide.org/en/latest/starting/installation/> för installation av dessa.

Öppna kommandotolken och skriv **pip install --user pipenv**. Om du kör Linux eller Mac och har en Python2 installation på datorn (samtidigt som din Python3 installation) kan du behöva använda **pip3** som kommando i stället för pip, alltså skriva pip3 install --user pipenv.

Om du stöter på problem här, se <http://docs.python-guide.org/en/latest/dev/virtualenvs/>

Användning

- Navigera till roten av vår Gitmapp och pulla senaste versionen, eller klona den till din dator om du inte har den än. I mappen finns det en fil som heter *Pipfile*, vilket är den filen som PipEnv använder för att hantera moduler och bibliotek som projektet ska använda. Just nu (i skrivande stund) finns enbart modulen py.test där, men när systemet fungerar och ska köras på Raspberry Pi så ska alla moduler och bibliotek som används finnas med här, så att det går lätt att installera på Raspberryn (och andra platser man vill). Du ska inte behöva bry dig om den här filen så mycket, den ska kunna sköta sig själv.
- Kör kommandet **pipenv install** i kommandotolken. Detta skapar ett virtualenv för detta projekt på din dator.
- För att installera nya paket, skriv **pipenv install PAKETNAMN**, t.ex *pipenv install flask*. Detta installerar det nya paketet i din virtualenv, och lägger till det i Pipfilen.

- Paketet (modulen) går nu att använda i din kod. T.ex ska du nu i dina pythonfiler kunna skriva ”import flask” och använda modulen.
- När du ska köra din kod måste du se till att du kör din i din ”virtual environment”. Det kan du göra på två sätt:
 1. Skriv **pipenv run FILENAMN.py**. Då körs programmet med hjälp av din virtualenvironment, där alla dina moduler ligger och kan användas.
 2. Skriv först **pipenv shell** för att ”logga in” på ditt virtual environment. Detta **måste** göras i rotmappen, där din Pipfile ligger. I ditt shell bör det nu indikeras att du har loggat in på ett virtualenv. För mig går jag från att se | **max@SUPERLAPTOP ~/ProjectGaming/gitmap \$** | där jag skriver, till att se | **(gitmap-Owbcew-k) max@SUPERLAPTOP ~/ProjectGaming/gitmap \$** |. Nu kan du navigera runt i filsystemet och köra filer med det vanliga kommandet, dvs *python SCRIPTNAMN.py* eller *python3 SCRIPTNAMN.py*, och den kommer alltid att köra scripten med modulerna i ditt virtualenv. För att logga ut ur ditt virtualenv, skriv **exit**.
- För att avinstallera ett paket, skriv **pipenv uninstall PAKETNAMN**.

För mer info om pipenv se <https://docs.pipenv.org/basics/>.

Commit-regler

När du har jobbat med din kod och vill commita något, **comitta inte pipfile.lock**, comitta **enbart Pipfile**. När man sedan ska萌ga olika Pipfiles till masterbranchen kan den ansvarige för detta behöva gå in manuellt och redigera Pipfile så att allas moduler kommer med.

Bilaga 10

Flower Power

Projekt och projektmetoder II-1302

KTH Projekt och projektmetoder II-1302

Teknisk dokumentation FlowerPower

Wifi-kommunikation mellan Arduino Uno och Raspberry Pi

Date	Version	Author	Description
22/04/2018	1	Erik Holm	Första draft
28/04/2018	2	Erik Holm	Komplettering med beskrivande bilder
14/05/2018	3	Erik Holm	Redigering av text
22/05/2018	4	Erik Holm	Sista draft

1. Introduktion
2. Material
3. Utvecklingsmiljö
4. Implementeringen
 - 4.1 Kommunikation mellan ESP8266 och Arduino
 - 4.2 Kommunikation mellan Arduino och Sensor
 - 4.3 Anslutning till nätverk
 - 4.4 Mottagandet av request från client
5. Testspecifikation
6. Sammanfattning
 - 6.1 Eventuella risker vid fortsatt implementation
 - 6.2 Miljö- och hållbarhetsaspekter och Etik

Referenser

1. Introduktion

Det huvudsakliga ansvarsområdet är kommunikation mellan:

Sensor->Arduino->ESP8266->Raspberry

Dokumentets syfte är att underlätta fortsatt underhåll av befintlig modul och implementering av ny modul. Dokumentet går först igenom materialval och utvecklingsval för att därefter gå igenom implementeringen steg för steg. Avslutningvis diskuteras eventuella risker vid underhåll och reflektioner kring val av hårdvara.

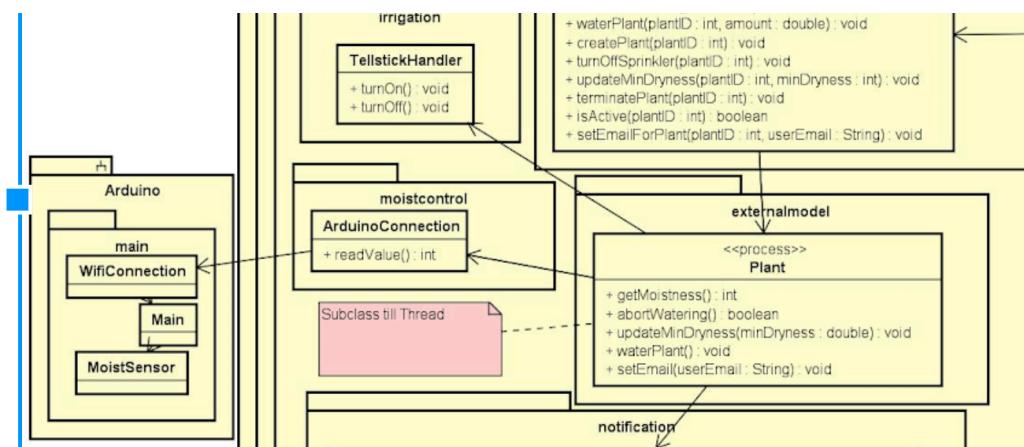


Bild 1 - Arkitektur för kontakt mellan Arduino->Raspberry Pi

Som man ser i arkitekturen i bild 1 kommunicerar Arduino Unos MoistSensor med ArduinoConnection på vår Raspberry Pi. När metoden `readValue()` kallas på returneras värdet. Kommunikationen mellan sensor, wifi-modul och och Arduino sker genom koppling på breadboard. Kommunikationen till Raspberry Pi sker genom wifi.

2. Material

- Arduino Uno MicroController
- ESP8266 Wifi-modul
- Breadboard
- Female/Male Wires
- Raspberry Pi
- Elektrokit fuktmätare

3. Utvecklingsmiljö

- Arduinos egna utvecklingsmiljö och språk
- Python för raspberry Pi
- Tester i pytest

4. Implementering

4.1 Kommunikation mellan ESP8266 och Arduino

Vi vill skicka ett värde från Arduino till Raspberry genom wifi så naturligt var det första steget att kommunicera mellan Arduino Uno och ESP8266. På så sätt skulle vi kunna skicka vidare det värde som plockades upp från sensorn. Arduinons standardinställning är att kommunicera med wifi-modulen via seriellporten på GPIO-stift 0 och 1 (RX och TX). Här sker en krock eftersom man använder den för att skicka data till Serial monitor på datorn och det går inte att dela på porten. Det löste jag med library SoftwareSerial som leder kommunikationen till andra stift än de redan nämnda stiften. Det gjorde man genom att öppna upp serials kommunikationsfönster och ange kommandot:

```
AT+UART_DEF=9600,8,1,0,0
```

Kommunikation kunde nu ske med ändrad Baud(hur många gånger per sekund som vår signal ändras).

Kopplingen mellan ESP8266 kopplades sedan som i bild 2 där pin 6 samt 7 nu kan kommunicera med ESP:n:

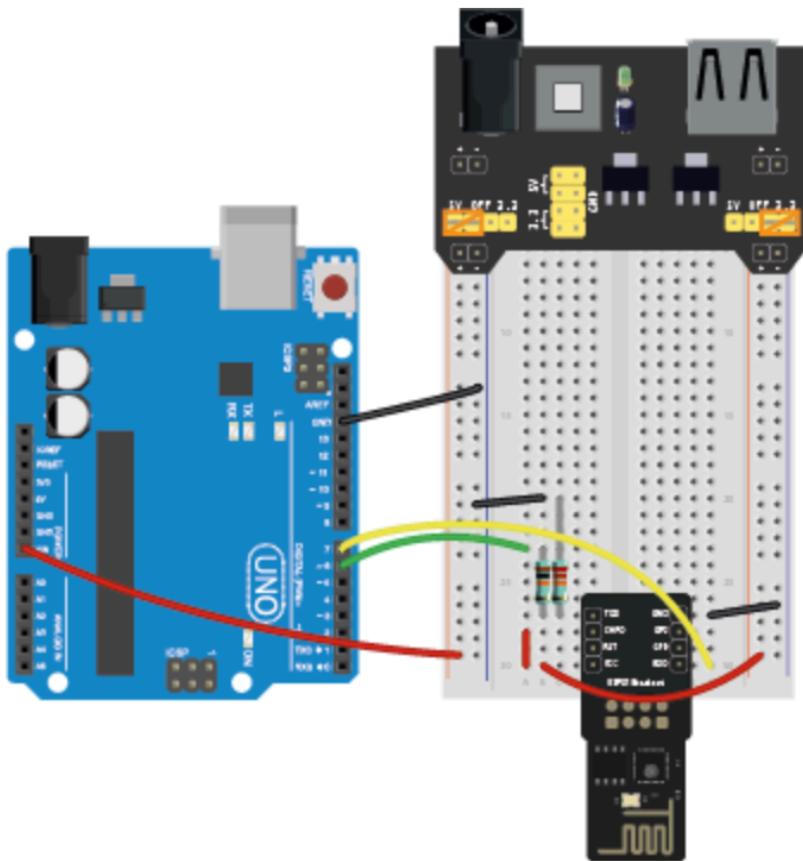


Bild 2 - Kopplingsschema mellan wifi-modul och Arduino

Slutligen sätter vi in två resistorer på 10 kohm samt 22 kohm för att sänka spänningen till 3.3V istället för den ursprungliga 5V. Det är inte bra att driva ESP:n med 5V, vilket är alldelens för högt.

4.2 Kommunikation mellan Arduino och Sensor

Härnäst behöver Arduino plocka upp ett värde från Sensorn. Sensorn kopplades till analog pin A0 och kunde plockas upp direkt till Arduinon. Det krävdes inga fler förberedelser vid användning av en sensor. Vid implementering av fler än en sensor kan det uppstå brist på output pins. Läs kravspecifikation av icke-funktionella krav för förslag på tillvägagångssätt. Sammanfattning (punkt 6.1) sammanfattar en eventuell lösning.

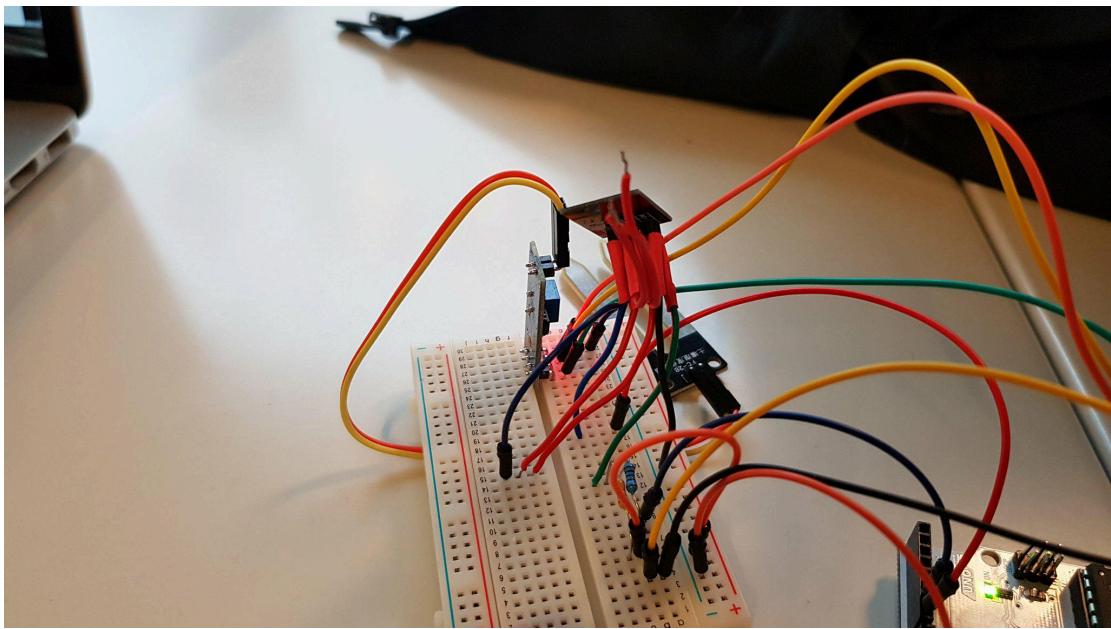


Bild 3 - Breadboard för sensor/wifi-modul/arduino sett från sidan

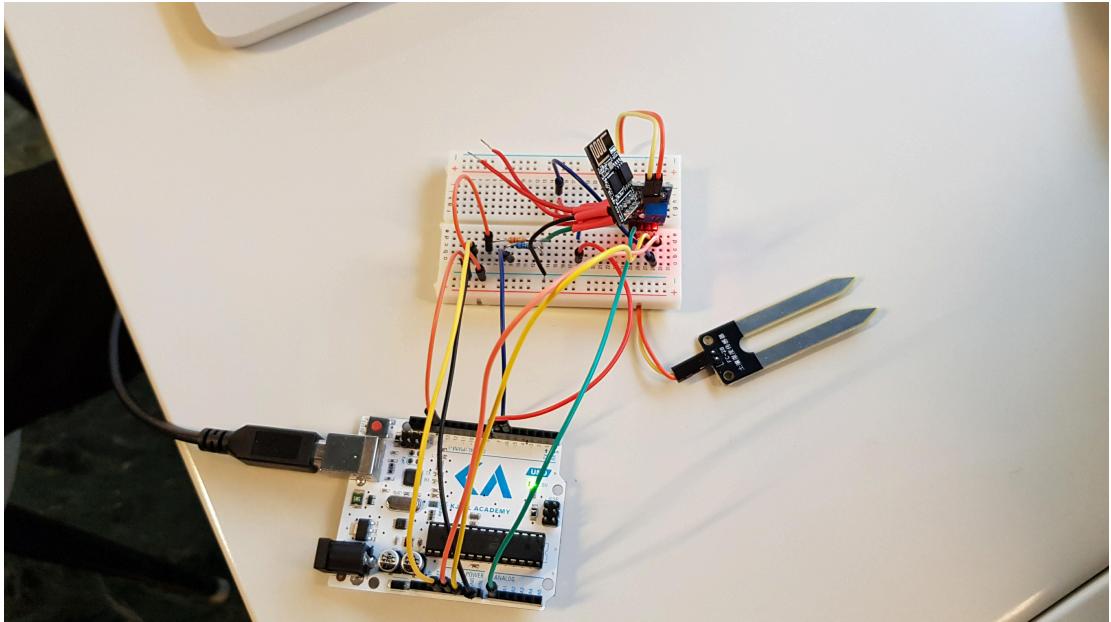


Bild 4 - Breadboard för sensor/wifi-modul/arduino sett ovanifrån

4.3 Anslutning till nätverk

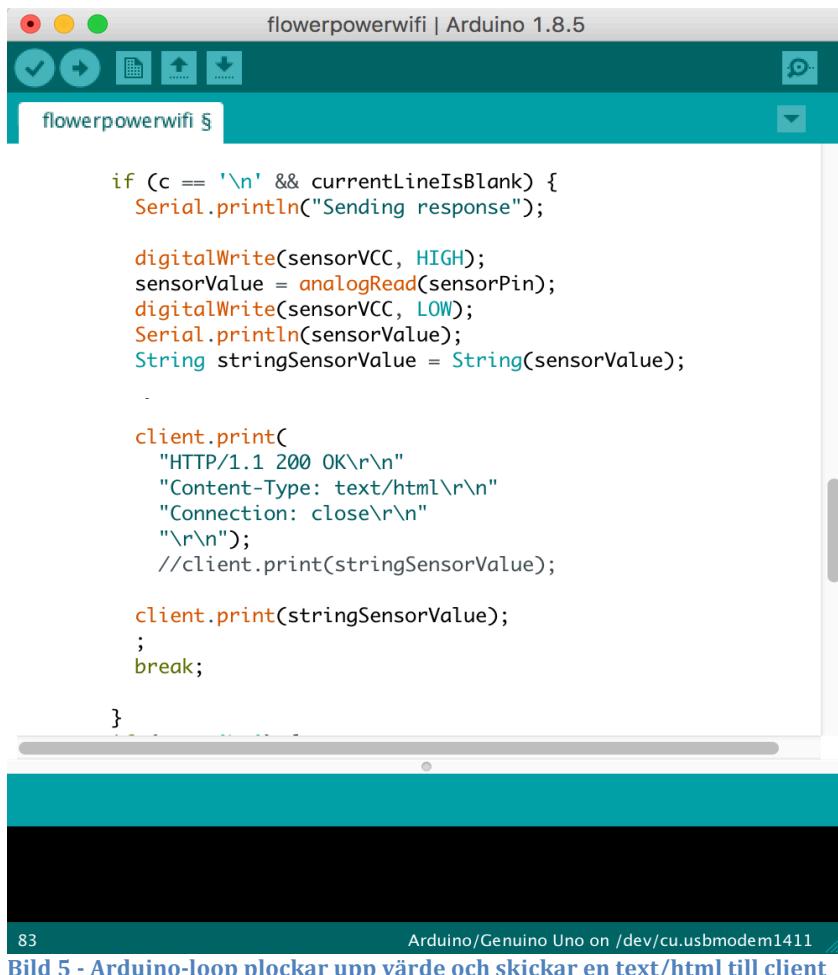
När värdet väl hade plockats upp från sensorn krävdes det en wifi-anslutning genom ESP:n. Vi hade nu en baud rate att serial-kommunicera med ESP-modulen. Därefter använde jag mig av ett bibliotek WiFiESP.h. På så sätt kunde man skicka en initialize request till ESP-modulen. Kommando var följande:

Init(Serial) – Parameter är den påbörjade serial-kommunikationen mellan ESP och Arduino.

Begin(SSID,PASS) – Nätverket lösenord och användarnamn är parameter för att ansluta till WPA/WPA2 nätverket.

4.4 Mottagandet av request från client

Vi har nu kontakt med nätverket och en web server kan startas för att ta emot requests. Arduinon har en loop-funktion som väntar på en http request från client. Svaret blir det värde som Arduinon plockar upp från sin sensor.



```
if (c == '\n' && currentLineIsBlank) {
    Serial.println("Sending response");

    digitalWrite(sensorVCC, HIGH);
    sensorValue = analogRead(sensorPin);
    digitalWrite(sensorVCC, LOW);
    Serial.println(sensorValue);
    String stringSensorValue = String(sensorValue);

    client.print(
        "HTTP/1.1 200 OK\r\n"
        "Content-Type: text/html\r\n"
        "Connection: close\r\n"
        "\r\n");
    //client.print(stringSensorValue);

    client.print(stringSensorValue);
    ;
    break;
}

83 Arduino/Genuino Uno on /dev/cu.usbmodem1411
```

Bild 5 - Arduino-loop plockar upp värde och skickar en text/html till client

Informationen görs om till en string och skickas som html när en request görs från raspberry pi:n. Pi:n gör sin request på följande sätt, med hjälp av biblioteket requests.

```

import json
import requests

class ArduinoConnection:
    #def __init__(self):
    #wlan = WLAN(mode=WLAN.STA)
    #wlan.connect("Isabels iPhone", auth=(net.sec, "ifpejnn09q53y"), timeout=5000)

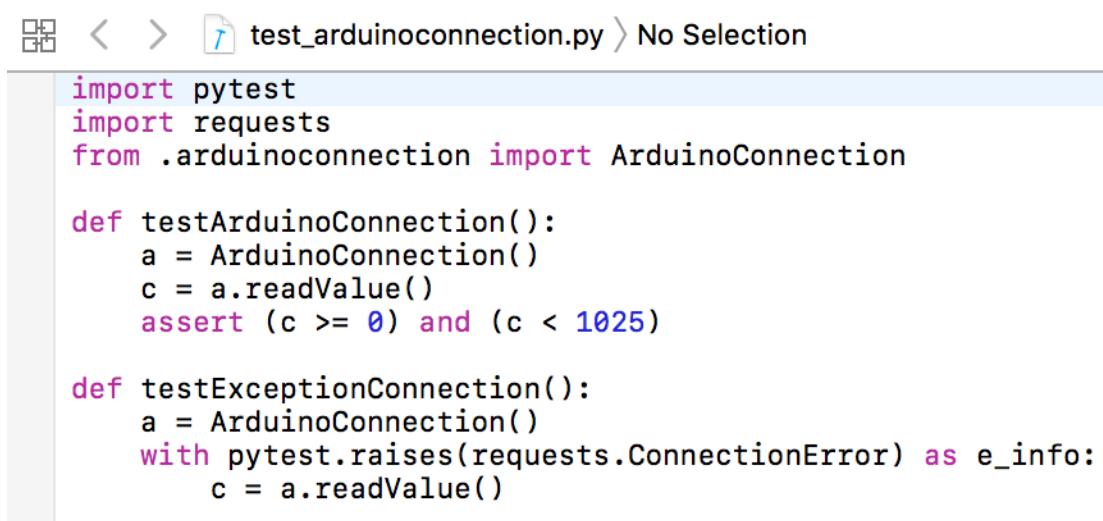
    def getValue(self):
        # r = requests.get('http://172.20.10.3/getValues.json')
        r = requests.get('http://172.20.10.3/')

```

Bild 6 - Raspberrykod för request där svaret sparas som "r"

5. Testspezifikation

Tester för kommunikation skrevs i pytest. Som man ser i bild 7 testas att värdet har ett rimligt värde och att vi inte har fått något fel i anslutningen med server (Arduino Uno). Läs pytests dokumentation och se exempel nedan vid skrivandet av ytterligare testkod.



```

import pytest
import requests
from .arduinoconnection import ArduinoConnection

def testArduinoConnection():
    a = ArduinoConnection()
    c = a.readValue()
    assert (c >= 0) and (c < 1025)

def testExceptionConnection():
    a = ArduinoConnection()
    with pytest.raises(requests.ConnectionError) as e_info:
        c = a.readValue()

```

Bild 7 - Pytest-kod med sensorvärdet som svar på request

Test för Arduino fixas internt genom kommunikationen med logg. Det fungerar mest som intern felsökning. All kommunikation som berör användaren hanterar redan pytest på clientsidan. Notera att det är viktigt att skriva ut status till serial för att få en överblick för varje moment (se bild 8).

```
You're connected to the network
SSID: Xperia Z5 Compact_2634
IP Address: 192.168.43.172
[WiFiEsp] Server started on port 80
[WiFiEsp] New client 0
New client
GET / HTTP/1.1
Host: 192.168.43.172
Accept-Encoding: gzip, deflate
Connection: keep-alive
User-Agent: python-requests/2.18.4
Accept: */*

Sending response
999
[WiFiEsp] Disconnecting 0
```

Bild 8 - Exempel på utförlig dokumentation i Arduinokod

6. Sammanfattning

6.1 Eventuella risker vid fortsatt implementation

Vid användning av fler än en sensor kan det behövas en extern modul för fler analoga inputs. En lösning vore en extern power supply alternativt en parkoppling mellan micro controllers. Ett övervägande som får ta med projektets budget i beräkning.

Spänningsförsörjning till Arduinon når inte alltid den utlovade 5V vilket gör att GND måste kopplas in igen efter en första uppstart, det här behöver göras för att det sker en initierad ökning i spänning-/strömförsörjning. För att slippa det krävs en extern strömkälla.

6.2 Miljö- och hållbarhetsaspekter och Etik

Vid val av en extern strömkälla och övriga komponenter är det viktigt att kontrollera inköpskällor, både ur ett etiskt och även ett miljöperspektiv.

E-waste eller E-skrot är utdaterad hårdvara som skickas till utvecklingsländer för återanvändning/ihopsamlande av komponenter. Behovet att byta ut elektronik även om komponenterna funkar ökar naturligtvis de mängder E-skrot som vi producerar årligen. Det syns även i statistiken då vi varje år exporterar 20-50 miljoner ton E-skrot. Anledningen till osäkra siffror beror på att mycket

sker illegalt. Om det här sker på ett okontrollerat sätt kan det påverka människors hälsa när farliga komponenter tas isär.



Som man kan se i bild 9 så är exporten ensidigt från I-länder till utvecklingsländer. Det är därför ett gemensamt ansvar att se över var ens hårdvara tillverkas och under vilka förhållanden. I vårt projekt kom de flesta komponenter från Kjell & Co. Även om Kjell&Co. bara eftersträvar en fullständig dokumentation så har de ett dagligt arbete för att garantera en CE-märkning. För mer information så hänvisas man besöka deras hemsida². CE-märkning visar att produkten överensstämmer med grundläggande krav på exempelvis hälsa, säkerhet, funktion, miljö enligt en EU-standard.

Referenser

- [1] <https://www.greenpeace.org/archive-international/en/campaigns/detox/electronics/the-e-waste-problem/>
- [2] <https://www.kjell.com/se/om-oss/hallbarhet>