Vad är en bra projektmetod för små IT-projekt?

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

Antonio Morales[#1], August Ronne[*2], Mahad Hassan[*3], Simon Salas Amnér[#4]

[#]*ICT, Kungliga Tekniska Högskolan*
Isafjordsgatan 22, 164 40 Kista, SE

[1]antmc@kth.se
[2]aronne@kth.se
[3]mahadh@kth.se
[4]amner@kth.se

***Abstract*— Kurs II1302 vid KTH ICT.**
Denna rapport inkluderar en kvalitativ undersökning som har som mål att undersöka och besvara frågan – ”Vad är en bra projektmetod för små IT-projekt?” Undersökningens validitet är låg då undersökningen utförts av en grupp studenter i en universitetsmiljö.

För att driva den kvalitativa undersökningen framåt så utfördes ett mindre IT-projekt med Essential Unified Process som den övergripande agila projektmetoden.

***Nyckelord*—** IT-projekt, projektmetoder, agilt, inkrementell utveckling, Scrum, EssUP

I. Om detta dokument och undersökning

Denna rapport är skriven i examinerande syfte för kursen *Projekt och projektmetoder* [1] som ges av Kungliga Tekniska Högskolan. Således är rapporten huvudsakligen skriven i syfte att läsas av kursens examinatorer. Rapportens disposition följer *Institute of Electrical and Electronics Engineers* (IEEE) riktlinjer för en vetenskaplig rapport [2]. Rapporten dokumenterar författarnas försök att utföra en mindre vetenskaplig undersökning av frågeställningen som återfinns i rapportens Abstract. Den vetenskapliga granskningen, översynen och kritiken av undersökningens slutsatser är, av naturliga skäl, inte tillräckligt rigorös för att kunna hävda att undersökningens slutsatser är trovärdiga. Projektets resultat framgår av bilagorna.

II. Introduktion

Detta kapitel ger en summarisk bild av projektet som utförts och syftar till att möjliggöra en förståelse av bakgrunden till rapportens uppbyggnad.

A. Bakgrund

Kursen *Projekt och projektmetoder* lär studenterna att applicera flera olika projektmetoder för att genomföra ett mindre IT-projekt i grupp. Under projektarbetets gång utvärderar och dokumenterar gruppmedlemmarna kontinuerligt de olika projektmetodernas arbetsgång och utfall för att försöka svara på frågeställningen som definieras i rapportens Abstract.

Kursens mål definieras som följer,

Effektmål
- Att förbättra en situation som beskrivits som otillfredsställande av projektets kund med hjälp av IT-teknik.

Resultatmål
- Att ta fram IT-teknik med tillhörande tjänster som förbättrar situationen som kunden är missnöjd med.

Projektmål
- Att studenterna i projektgruppen ska lära sig förstå och applicera olika projektmetoder för IT-projekt och avsluta kursen med minst ett godkänt betyg.
- Skriva en slutrapport som försöker besvara frågan *Vad är en bra projektmetod för små IT-projekt?*

Koncepten *effektmål, resultatmål, projektmål* och deras definitioner är tagna från boken *Arbeta i projekt: individen, gruppen, ledaren* av Sven Eklund [3].

Inom IT-branschen är så gott som alla färdigställda produkter ett resultat av ett projekt. I ett IT-projekt arbetar projektets deltagare mot förutbestämda mål. IT-projekt är till viss mån unika och är tillfälliga arbeten till sin natur. Projekten är oftast svåra att kontrollera. För att försöka uppnå kontroll över projektet används oftast projektmetoder, vilka är verktyg för att skapa och kontrollera projektets omfattning [4]. Kursarbetet som rapporten dokumenterar skiljer sig inte från IT-branschen i övrigt när det kommer till syftet med att använda projektmetoder. Även studenterna i projektgruppen använde sig av projektmetoder för att kunna kontrollera och behärska sitt projektarbete.

B. Problemformulering

Frågeställningen *Vad är en bra projektmetod för små IT-projekt?* är utgångspunkten för hela kursen och därför även denna rapport och bör förklaras. Inom definitionen "software engineering" eller mjukvaruutveckling är "ingenjörsdisciplinen" det begrepp som innefattar de verktyg, projektmetoder och teorier för att uppnå projektmålen [10, s 8]. Den andra komponenten som definierar mjukvaruutveckling är alla aktiviteter som behöver göras för att kunna stödja en framgångsrik mjukvaruproduktion [10, s 8]. Då är frågan som bör ställas, vad är det för projektmetoder som behövs för att uppnå ett lyckat projekt? Vilka verktyg, teorier och aktiviteter är dessutom viktiga för att projektet ska lyckas.

C. Undersökningsstrategi/lösningsstrategi

Fallstudie genom att i grupp försöka genomföra ett mindre IT-projekt med utvalda projektmetoder och där varje gruppmedlem värderar sitt ansvarsområde i projektet för att ur detta försöka besvara den övergripande frågeställningen.

III. Teori och ingenjörspraxis

Detta kapitel listar och i viss mån beskriver teorier och ingenjörspraxis som använts i undersökningen. Det finns två underkapitel, Litteraturstudie och Förstudie.

D. Litteraturstudie

I detta kapitel anges litteratur och andra källor som har använts för att hitta ingångar och möjligheter i undersökningen. Förutom de källor som anges finns förmodligen andra, och kanske bättre, källor som denna studie inte använts sig av.

Undersökningens görs utifrån övergripande projektmetoder men också utifrån specifika metoder och arbetssätt som används av olika kompetenser [5] i projektets team. Vilka dessa kompetenser är framgår av texten nedan.

Litteraturen listad under *Övergripande källor för hela projektet* har varenda gruppmedlem studerat och arbetat med.

Övergripande källor för hela projektet

- Projektets hälsa, status och gemensamma standard [5]
- Projektets användning av Scrum, sprintplanering, sprintdemonstrationer och sprint retrospectives [6]
- Fasindelning [3, s. 84]
- Projektgrunder, inklusive konflikthantering, gruppteori, ledarskap och person- och gruppdynamik [3]
- Verktyg för agilt projektarbete [7]
- Definitioner av och skillnader mellan Vattenfallsmetoden och den agila metoden för att organisera projektarbete [8]
- Projektmetoder: användning och värde [4]

Kundrepresentant - kompetens enligt [Essence 1.0]

- Vision [9], [10]
- Kravspecifikation [9], [10]
  - "Stories" [9]
  - "Use Case" [9]

Analytiker - kompetens enligt [Essence 1.0]

- Arkitekturer (4+1) [11], [10]

Utvecklare - kompetens enligt [Essence 1.0]

- Utvecklingsplan [10], [12]
- Github
- Modellering av webbapplikationer [13]

Testare

- Testplan

Ledning och styrning - kompetens enligt [Essence 1.0]

- Projektplanering
  - Projektdefinition

E. Förstudie

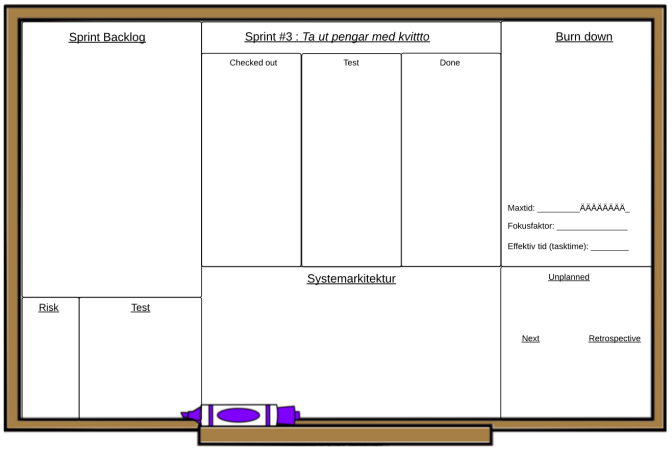
Enligt *undersökningsstrategin* (Sektion C) så skall en eller flera projektmetoder användas av gruppen för att utföra ett mindre IT-projekt och utifrån de erfarenheter som fås och dokumenteras skall sedan en utvärdering av projektmetoderna ske. Eftersom studenternas erfarenheter av att planera och utföra projektarbeten inom IT är otillräckliga så erbjuder kursen ett omfattande ramverk när det kommer till ansatsen av projektmetod.
Denna ansats är baserad på tidigare kursomgångar och lärarens förslag. Ansatsen tillhandages projektgrupperna både genom färdigdesignade upplägg för arbetstavlor, där designen på tavlorna hjälper till att organisera ett agilt arbetssätt, samt tydliga instruktioner om vilken

litteratur som bör utgöra projektarbetets teoretiska grund. Ansatsen som givits till projektgruppen kan sedan gruppmedlemmarna själva bestämma om de vill modifiera.

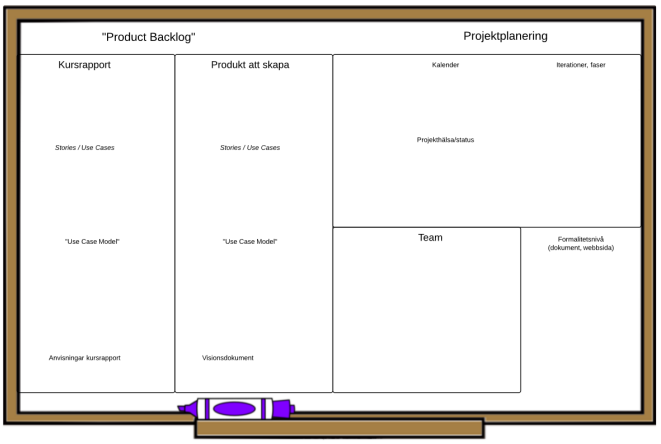
Resultatet av denna förstudie är att projektmetoderna, som anges i följande kapitel, har valts för att genomföra undersökningen.

Figurer 1: Arbetstavla iteration 2 och iteration 4

Arbetstavla "Arbetssida"



Arbetstavla "Publik sida"



IV. UNDERSÖKNINGSMETODER

Detta kapitel beskriver vilka metoder som använts 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.

F. Frågor att besvara i undersökningen

Frågorna kategoriseras i följande kategorier …(eventuellt).

1. Hur skall man bedöma/redovisa om en projektmetod eller praktik är bra?
2. Hur kan man kategorisera, välja, och namnge projektmetoder (projektpraktiker) och (verklighetsbeskrivning) så att diskussionen om dito blir begreppsmässigt konsisten 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?
5. (Vad och hur skall eller behöver jag som student redovisa i rapporten för att bli godkänd på kursen? Denna fråga tas bort i den slutliga rapporten)

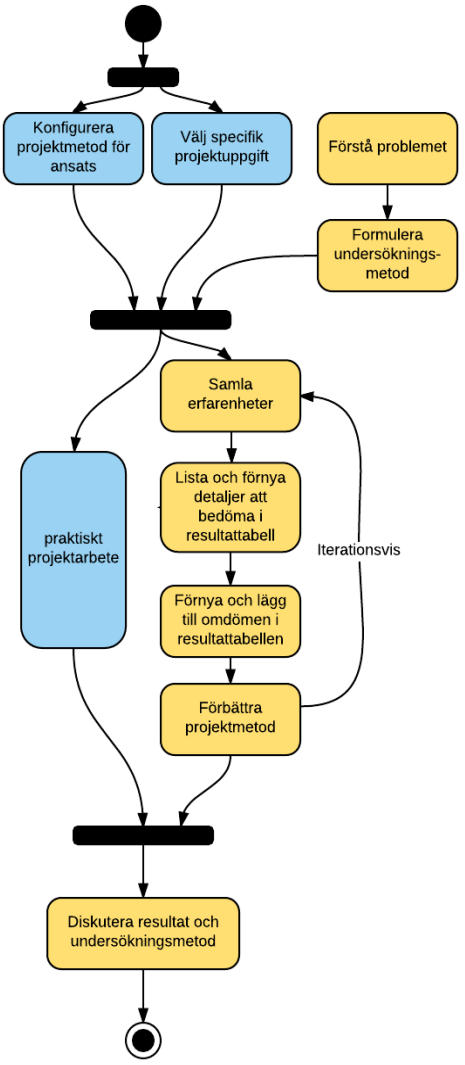
G. Metodbeskrivning

Den centrala metoden i undersökningen är att avgöra om olika valda projekt-praktiker och arbetssätt är ”bra” och om de bidrar till att göra hela projektprocessen bra. Åstadkommer/skapar projektet rätt saker och konstrueras lösningar på bästa sätt?

Metoden för att samla data i denna fråga blir induktiv då erfarenheten i gruppen är ytterst liten. Arbetssättet blir att efterhand som projektet fortskrider så förs bedömningsområden som anses vitala och bedömningskriterier in i en tabell kontinuerligt. Tabellen, se kapitel ”Resultat”, och dess utformning förbättras också hela tiden.

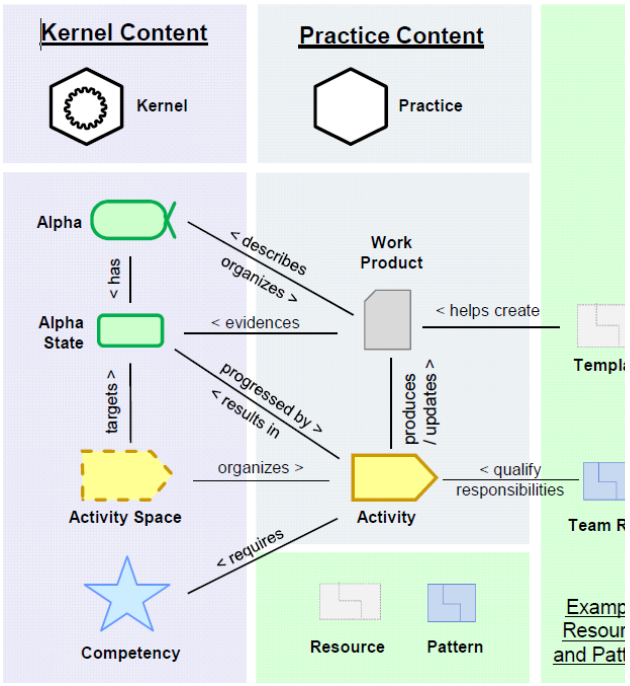
Som hjälp för att utvärdera om en projektmetod är bra eller inte så utgår man från åtagandetriangeln [3, Fig. 10.1] som har modifierats av kursansvarig genom att lägga till kvalité som en fjärde faktor.

Metod 1: Undersökningsmetod, se figur. De gula fälten är aktiviteter som kopplar till själva undersökningen. Metoden följer principer för vetenskaplighet enligt [14, s. 17].

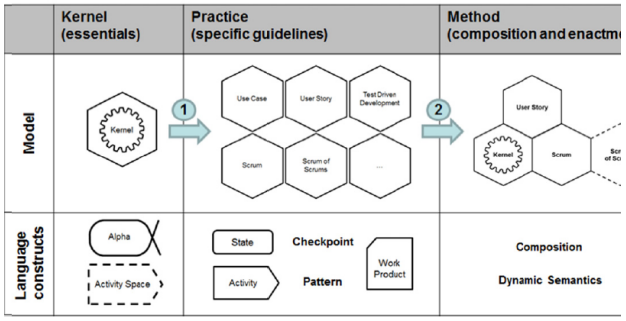


Figur 2: Undersökningsmetod för ”Vad är bra projektmetod för små IT-projekt”.

Metod 2: Begrepp. Begrepp som används följer om möjligt OMGs standard [5].
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 3: Begrepp [15]



Figur 4: Practice och Methods[16]

V. GENOMFÖRANDE

I följande kapitel redovisas viktiga beslut, förändringar och anpassningar som gjorts i projektmetod, projektpraktiker, värderingar, beslut mm som gjorts under studiens genomförande.

Projektet utgick från Essential Unified Process (EssUP) [17] som övergripande projektmetod och hade Scrum, DevOps, komponentbaserad och inkrementell utveckling samt arkitektur och användarfall som del-projektmetoder, eller "Practices". Andra projektmetoder och "Practises" testades också på och beskrivs mer i följande sektioner som är indelade efter olika ansvarsområden inom projektet.

För att lösa problem kopplade till gruppdynamik, konflikter, mötesteknik och värderingar hos gruppmedlemmarna har boken *Arbeta i projekt* [3] av Sven Eklund använts som verktyg av projektgruppen under hela projektets gång. Trots förändringar i övriga arbetsmetoder under iterationerna har koncepten från denna bok varit styrande för gruppens interaktioner med varandra.

H. Projektledning (Simon)

Projektet bestod av fem iterationer som delades in i olika kategorier som representerade de huvudmål för varje iteration:

1. Projektplanering och strukturering
2. Hårdvara och Mjukvara
3. Testning och konstruktion
4. Dokumentering och presentation
5. Rapportinlämning

Huvudmålen fungerade som riktlinjer för projektet om vad som bör göras vid varje iteration för att nå våra slutgiltiga mål. Då projektmetoden baserades på agila metoder och principer som Scrum [6] och EssUP [17] så var ingenting i iterationerna satta i sten. Arbetsuppgifter i iteration 1 kunde genomföras i iteration två och vice versa, beroende på uppgiften och erfarenheten hos projektmedlemmarna. Som nämnt, de agerade sin roll enbart som huvudmål.

Teori för tillvägagångssättet för projektledningen under projektet har tagits från kurslitteraturen Sommerville [10, St. 22-23.] och exempel från kursansvarig i kursen Projekt och Projektmetoder [1].

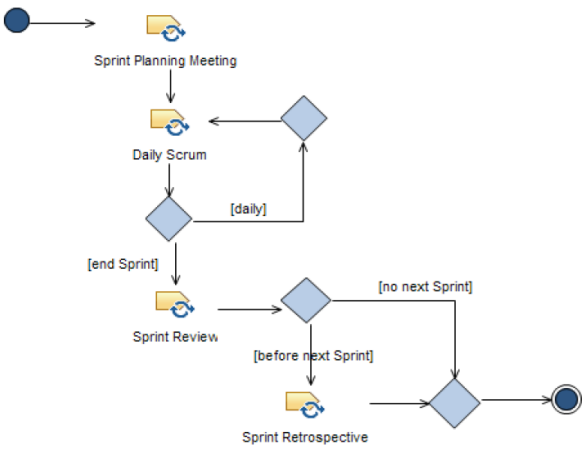
Iteration 1 innefattade det mest av arbetet för rollen som projektledare. Under denna iteration skapades den underliggande planeringen för projektet som projektdefinition, visionsplan, iterationsmål och indelning av projektroller och ansvarsområden. Projektdefinitionen och visionsplanen skrevs gemensamt då detta bidrog med att projektmedlemmarna skapade gemensamma mål om vad detta projekt faktiskt skulle uppnå samt bidrog detta till diskussion och varje medlem fick lära sig om vad de andra i gruppen tyckte och vad de hade för avsikt med projektet. Detta i sin tur bidrog till att gruppen började att jobba emot att bli en "cohesive group" [10, St. 22.2.1], vilket är extremt viktigt för progressionen av projektet.

Iterationsmålen skrevs av projektledare med huvudsyfte att dokumentera kommande risker, objektiv och uppgifter för iterationen. Se bilaga Iterationsplan 4. Att ha informationen både digitalt, iterationsplan i google drive, och fysiskt, arbetstavlor, gav gruppen

förutsättningar att jobba med projektet utan att vara på arbetsplatsen.

Indelning av projektroller var den första potentiella konflikten som projektgruppen stod inför. Projektrollerna som skulle delas ut var: Projektledare, Krav- & Kundansvarig, Utvecklingsansvarig, Arkitekt och Testansvarig. Dessa fem roller skulle delas in på de fyra medlemmarna i gruppen. Innan rollerna delades ut fick alla i gruppen säga sin behov om vilken roll som passade just dem och vad de skulle vilja få ut av en specifik roll. Processen slutade med att alla fick den roll de önskade och de som inte önskade någon särskild roll fick ta det som var över. Testansvarig blev ej vald men huvudansvaret låg hos projektledaren att se till att test genomfördes men blev utfört på ett väldigt enkelt och grundläggande sätt. Utifrån detta möte fick gruppen reda på allas sätt att se på konflikter. I boken, Arbeta i projekt [3] beskriver Sven Eklund hur konflikter kan ses utifrån två sätt: konfliktsyn och harmonisyn. Där gruppens sätt att samarbeta genom att berätta, lyssna och fråga ansluter mer till konfliktsyn. Utöver indelning av av roller delades själva gruppen in i två ansvarsområden, hårdvara och mjukvara, då gruppen bestod av två studenter från TIDAB och två studenter från TIEDB. Eftersom programmet TIDAB, datateknik, har kurser som fokuserar på mjukvara så var det ett logiskt val att de två studenterna fick ta ansvaret för webbapplikationen och DevOps. Den fysiska produkten där elektroniska komponenter och design av kretskort var inblandade var mer lämpad för de två studenterna i TIEDB.

Varje iteration kan visualiseras med illustrationen av aktivitetsdiagrammet, se figur 5. Alla iterationer började med ett sprint möte där övergripande information gällande projektet diskuterades, exempelvis förändringar kring produktkraven, projekthälsa och uppdateringar kring webbapplikationen och elektroniska apparaten delades. Arbetstavlor uppdaterades, nya tasks togs fram, överblivna tasks från föregående iteration uppdaterades och schemaläggning av gemensamma arbetspass. Vid varje planerad arbetspass utfördes daily scrum [6, s. 73] framför arbetstavlorna där varje medlem fick berätta vad de gjorde igår och vad de planerade att göra idag, vilket bidrog med att frågorna gällande daily scrum besvarades, vad gjorde jag igår som förde projektet närmare sprint målet?, vad kommer jag att göra idag för att gruppen ska nå sprint målet? och ser jag några hinder eller risker som hindrar jag eller min grupp att nå sprint målen? [6, s. 74], det blev tydligt att få en förståelse för vad alla i gruppen bidrog med för att få projektet att nå sprint målen.



Figur 5: Iteration/Sprint [15]

I slutet av varje iteration hölls en sprint retrospective med målet att utvärdera genomförd sprint. Retrospectives har utgått ifrån tre punkter: vad har varit problematiskt som kan förbättras?, vad ska teamet börja att göra? och vad ska teamet fortsätta göra? [referens samt bilaga]. Gemensam utvärdering av utfört arbete, arbetsmetoder och självreflektion genomfördes för att se över potentiella förbättringar. Förbättringar kunde naturligtvis implementeras vid vilken tidpunkt som helst men att hålla schemalagda retrospectives gav teamet en formell chans att inspektera och adaptera arbetet.

I kurslitteraturen Summerville [10, St. 22] så rekommenderas det att en projektgrupp kan och bör ha så kallade “away days”. Away days menas med att gruppen byter arbetsmiljö för att umgås och diskutera projektet. Aktiviteter för grupp moral är inte sällsynta i detta sammanhang och i detta projekt infördes fika på de gemensamma arbetspassen för att representera dessa “away days” samt för att förbättra arbetsmiljön.

Iterationsplaneringar, sprintmöten, daily scrum och retrospectives gav kategoriserade inblickar av projektet och för att skapa en helhet av hur projektet låg till så infördes alphastates [18] i iteration 2 som gavs av kursansvarig. Korta möten angående projektets hälsa hölls tillsammans med retrospective i slutet av varje vecka för att se till att projektet rörde sig framåt och att varje medlem hade koll på var projektet befann sig.

I. Utveckling (Antonio)

Utvecklingen i projektet har följt en iterativ inkrementell utvecklingsmodell med tydliga iterationsmål som redovisats vid varje Scrum demo. Den inkrementella utvecklingen har inneburit att en första implementation (en första version) har utvecklats och sedan med iterationerna utökats till nya versioner för att sedan bli en färdig version som kunden är nöjd med. Detta följer

även principerna för en inkrementell utvecklingsmodell [10, St. 2.1.2]. Till skillnad från en plan-driven arbetsprocess som vattenfallsmodellen för utvecklingen där kravspecifikation, system och mjukvarudesign, implementation och testning, integrering och system testning, och operation och underhåll sker i isolerade faser [10, St. 2.1.1] så har det mesta av dessa steg skett vid olika tillfällen vid varje iteration, mer än en gång, och på så sätt bidragit till en agil projektmetodik och följt en inkrementell utvecklingsmodell.

Innan utvecklingen började så utvärderades gruppmedlemmarnas kompetenser. Då gruppen bestod av fyra medlemmar, två med studier inriktad mot datateknik och två med studier inriktad mot datorteknik, så var det tydligt från början hur man skulle dela upp arbetet. Uppdelningen av arbetet var komponentbaserad och därför var utvecklingen också komponentbaserad. Gruppens storlek gjorde det enkelt att dela upp arbetet och ansvarsområden för det mindre IT-projektet och att även ha koll på allas arbetsuppgifter. Arbetsuppgifterna och hur de gick togs upp som en del av den så kallade *daily scrum* [6, s. 75]. Andra saker som tagits upp vid daily scrum har också varit frågor om design och utveckling.

Efter uppdelning av ansvarsområden och komponenter så kunde utvecklingen börja. Kravspecifikationen var väldig tydlig från projektets start, vilket gjorde det lätt att snabbt sätta upp iterationsmål redan vid första iterationen för utvecklingen av mjukvara, skapa en första utvecklingsplan och börja designa. Detta bidrog till en snabb start för utvecklingen i projektet där mycket fokus kunde läggas på implementationen och följa den inkrementella utvecklingen.

Med iterationsmålen upptäcktes det snabbt att det fanns brist på kunskap om exempelvis JavaScript och Node.js då det var nytt för alla i gruppen. Eftersom att genomförandet av iterationerna följde Scrum i grunden så löstes detta genom att planera in teori-*stories* vid varje sprint planering som sedan kunde brytas ned till *tasks* för att ha med dessa i gruppens *sprint backlog* [6, s. 15]. Teori-stories och dess tasks räknades som en arbetsuppgift och innebar att studera teori för att kunna implementera andra stories och tasks som krävde kunskap man inte haft tidigare. På så sätt planerades teori och kunskap som krävdes för utvecklingen in med resten av gruppens sprint backlog.

Den inkrementella utvecklingen följde principerna för "Development and Operations" (DevOps). Detta inkluderade *Continuous Integration* [19] och *Continuous Delivery* [20], som är några DevOps praxis [21]. Detta gjorde så att utvecklingen lättare kunde ske kontinuerligt på ett iterativt och inkrementellt sätt. Genom att tidigt sätta upp en gemensam github förvar (*repository*) med individuella *branch* för gruppen, koppla det till IBM Cloud-tjänsten som erbjuder anpassningsbara verktyg samt värdtjänster av internetapplikationer, så kunde alla i gruppen kontinuerligt slå samman sin egen kod med resten av koden (*Continuous Integration*) samt kontinuerligt leverera dessa framsteg genom att låta verktyg i IBM automatiskt bygga, testa och distribuera till "runtime"-miljö (*Continuous Delivery*) genom kopplingen med Github förvaringen.

Resultatet av utvecklingen efter varje iteration presenterades under en så kallad *sprint demo* [6, s. 81].

J. Arkitektur (Mahad)

Projektet indelades tidigt i två tekniska grupper, en grupp som ansvarade för hårdvaran och en grupp som ansvarade för webbapplikationens mjukvara. Detta skedde kring den första iterationen. Under denna period utformades dessutom en konceptuell arkitektur över produkten. Målet var att ge en översiktlig visualisering av produkten och dess funktioner [10, s 119]. Vid denna fas var det fortfarande för tidigt för att gå in i detalj över abstraktionerna som uppkom från arkitekturen. Kraven som ställdes på produkten blev tidigt givna till teamet, så under iteration 1 spenderade teamet på vilka funktionella egenskaper som behövdes för att Office Display skulle kunna levereras. Det är viktigt att dessa val bestäms tidigt för att det leder till en minskning av systemets kostnad och tiden som krävs för att förstå systemet bakom produkten [10, s 120].

En konceptuell arkitektur börjar i princip alltid som en start-arkitektur för IT-projekt [10, s 154]. Fördelen med arkitekturstilen är att designern kan utforma arkitekturen på ett informellt sätt och den påbörjar diskussioner med resten av teamet angående vad produkten ska innehålla och hur olika funktioner och objekt är relaterade till varandra. Vanligtvis brukar en kund och krav-ansvarig förmedla användarfall som en arkitekt ska försöka att arbeta utifrån när projektet är i sin första fas [22, s. 3] , men beslut om vad produkten bestod av beslöts gemensamt inom gruppen. Exempel på detta var diskussionen om produkten skulle ha en högtalare som skulle sättas på när sensorn reagerade. Hårdvarugruppen beslutade gemensamt med webbapp-gruppen att den inte behövdes.

Under sprintdemonstrationen i iteration 2 kommenterade kursansvarige att det var svårt att se hur arkitekturen urskiljde hårdvara och mjukvara. Varje komponent i arkitekturen hade samma färg men var löst grupperade beroende på vilken sida de var i. När den konceptuella

arkitekturen utformades var det viktigt att designen var lätt att förstå, så stilen gick ut på att rektangulära figurer skulle representera en komponent eller funktion. Som exempel var en utav de högst satta objekten “hårdvara”[Bilaga C]. Det objektet pekade nedåt på andra hårdvarukomponenter som hårdvaran bestod av, sedan pekade de komponenterna ned på konceptuella funktioner som de skulle uppnå. Den enda relationen mellan rektanglarna som användes var svarta pilar. Speciella språkval i arkitekturen som Booch, UML eller ADL applicerades inte än. Fördelen med detta var att gruppen använde ett informellt språk som de förstod, och att processen att utforma en arkitektur gick snabbare när designval bestämdes själv. Färgkoder sattes för att separera webbapplikationen från hårdvaran. Grönt kom att representera funktioner kopplade till webbapplikationen och rosa representerade hårdvarans relaterade funktioner [Bilaga C].

När alla i teamet började komma in i sina roller var det tydligt att arkitekturen behövde utökas för att dokumentera det existerande systemet som utformades. Målet var alltså i det skedet att beskriva produkten på ett formellt vis utan att göra några ändringar på systemets beståndsdelar [10, s 119]. Ett sekvensdiagram skulle först implementeras för att beskriva hur användaren skulle bemötas och interagera med produkten [10 s, 126]. Det första sekvensdiagrammet som gjordes visade bara hur produkten skulle interagera med användaren, men diagrammet fick konstruktiv kritik för att inte använda sig av några scenarion och sedan koppla dem till de tekniska uppgifterna som som skulle implementera processen. På grund av tidsbrist under den iterationen (iteration 2) pausades arbetet med sekvensdiagrammet. Med tiden var det tydligt att det fanns andra alternativ för att visa tidsberoende samverkande processteg i en arkitektur istället för ett diagram. Dessa kunde göras med en så kallad “pipe and filter”-arkitekturmodell [10, 163]. Det är en designmodell för arkitekturer där data flödar från process till process. Fördelen med arkitekturen är att den är lättförståelig [10, s 162] för att den bara använder två objekttyper, data och processer, samt att objekten rör sig bara med svarta pilar. Arkitekturmodellens idéer sammanfogades senare ihop med processvyns arkitektur som en del av den större “4+1”-modellen av Philippe Kruchten [11].

4+1-Modellen utvecklades av Philip Kruchten för att kunna ge en helhetlig abstraktion över produkten från fem perspektiv eller så kallade vyer [11, s 1]. Vyer är olika tillvägagångssätt som ett team kan se en arkitektur från för att sedan implementera den i verklighet [11, s 1]. 4+1-Modellen har fyra specifika vyer: den fysiska vyn, logiska vyn, utvecklingsvyn och processvyn. Den femte vyn (+1) är scenariovyn (användarfallet) [11, s 2]. Scenariovyn är vyn som sätter ihop alla fyra vyer och implementerar det från användarens perspektiv genom att införa ett eller flera användarfall. Se mer om detta i Arkitekturbeskrivningen [Bilaga C].

Den fysiska vyn som implementerades hade som syfte att visa vilka hårdvarukomponenter som är viktiga för systemet och på så vis förhindra dåliga ändringar av komponenter som degraderar systemet. Dess design utformades som ett fysiskt blockschema [23, Kap. 6.1] . Den fysiska arkitekturen var den första vyn som implementerades, främst för att den var lättast att utforma. Genomförandet bestod av att rita varje komponent och beskriva hur den hängde samman med andra komponenter [Bilaga C]. En hög grad av partitionering rekommenderas inte som designval när ett blockschema utformas [23, Kap 6.1], då kan man få en mer global vy över systemet. Processvyn implementerades som en hybrid mellan ett sekvensdiagram [10, s 126] och en pipe-and-filter-arkitektur [10, s 163]. Femhörniga gråa figurer användes som representerade processer och aktörsfigurer användes för att visa hur en klient skulle kunna interagera med produkten, likt ett sekvensdiagram [10, s 127].

Logikvyns syfte är att visa vilka funktioner som styr systemet och hur de olika funktionerna är beroende av varandra [11, s 3]. Arkitekturen använde sig av Booch-notation för att kommunicera mellan objekt [11, s 3]. Det är en modelleringsmetod med ett objektorienterat generellt språk för arkitekturer och diagram [10, s 120]. Pilar och figurer har en bestämd definition som måste följas så att andra läsare kan förstå designen utana att ge en detaljerad förklaring.

En arkitektur för utvecklingsvyn skulle ge en beskrivning av vad mjukvaran bestod av i systemet [11, s 6]. Den skulle bestå av moduler och segment i koden och deras relationer mellan varandra. Processen att utforma den visade sig dock vara komplicerad. Det första hindret var att arkitekturen för logikvyn och utvecklingsvyn var så lika. Det verkade inte fullständigt nödvändigt att då ha den. Det andra hindret var att avgränsningar gjordes för att de tekniska rollerna var separerade mellan hårdvaru och mjukvarugruppen, och eftersom arkitekten var en medlem av hårdvarugruppen var det svårt att skriva en arkitektur för webbmjukvaran. Det tredje var att UML som designspråk prövades och den var svår att implementera när mjukvaran som utvecklades inte var objektorienterad utan imperativ. UML hade dessutom testats i logikvyn, men den modellen övergavs för att Booch-notation passade bättre [24, Kap. 4] . Till slut övergavs arkitekturen för utvecklingsvyn.

Slutligen gjordes en arkitektur för scenarion som skulle representera användarfall. Dess syfte var att sätta ihop alla fyra vyer och implementera dem för ett eller flera användarfall [11, s 9]. Med dessa vyer kunde en arkitektur utformas som bidrog med att ge en formell beskrivning av produktens system.

K. Kund- och kravansvarig (August)

Projektgrupp 3 hade kursansvarig som kund. Således var kundens önskemål om vilka problem produkten skulle kunna lösa kända i ett tidigt skede, till skillnad från i de projektgrupper där gruppmedlemmarna själva fick agera kund.

Kursansvariges önskemål användes sedan som utgångspunkt för att formulera produktens grundläggande funktionella krav, enligt instruktioner i [10, Kap. 4.1].

Kund- och kravansvarig läste sedan om användningsfallsmodellering i [9], som var en av två obligatoriska teoretiska grunder för arbetet i den formella rollen. Utifrån dessa grundläggande funktionella krav producerades en första användningsfallsmodell för produkten som skulle utvecklas.

I detta tidiga skede av projektet så formulerades även projektgruppens vision i form av ett visionsdokument, där dokumentmallen tillhandahölls av kursen. Framtagandet av visionsdokumentet gav upphov till viktiga diskussioner och beslut om projektets intressenter (översatt från engelskans *stakeholders*), enligt OMG:s standarder [5, Kap. 8.2.2.1], samt projektets möjlighet (översatt från engelskans *opportunity*), även denna diskussion grundad i OMG:s standarder [5, Kap. 8.2.2.2].

Denna första användningsfallsmodell byggdes om från grunden i början av iteration 2, efter konstruktiv kritik från både kursansvarig och projektgruppens medlemmar. Kritiken grundade sig i att användningsfallsmodellen innehöll tekniska lösningar och specifikationer som kunden inte uttryckligen bett om.

Kundens önskemål om vad produkten skall kunna lösa för problem skall formuleras som användningsfall som är förståeliga även för intressenter utan den nödvändiga tekniska utbildningen för att utveckla produkten [9, s. 6]. På grund av att användningsfallsmodellen innehöll dessa tekniska specifikationer blev kravbilden tilltrasslad och otydlig för projektgruppens medlemmar. Tanken är att användningsfallsmodellen skall tydliggöra de funktionella kraven för projektgruppen så att tekniska lösningar för kraven kan tas fram. En användningsfallsmodell som är felbyggd på tidigare nämnda sätt förhindrar denna arbetsgång.

Att ta fram iterativa uppgifter (*tasks* på engelska) att arbeta mot i iteration 1 fungerade dock någorlunda väl trots den bristfälliga användningsfallsmodellen tack vare att kursmaterialet innehöll tydliga instruktioner om uppgifter för iteration 1.

Den nya användningsfallsmodellen som togs fram under iteration 2 innehöll inga tekniska lösningar på de funktionella kraven och var skrivet på ett språk som kunden kunde förstå även om han helt saknade den tekniska utbildning som projektgruppen besitter. Till denna iteration togs även en användningsfallsmodell inriktad på mjukvaruutveckling fram, samt en användningsfallsmodell för hårdvaruutveckling. Dessa modeller beslutade sedan gruppen att inte använda i arbetet eftersom de lämpade sig bättre som *slices* [9, s. 8].

Den senaste användningsfallsmodellen användes som grund för att planera det iterativa arbetet under iteration 3. Medlemmarna diskuterade användningsfallen och vilka *stories* dessa kunde leda till. Dessa stories försökte gruppen sedan dela upp enligt ett horisontellt arbetssätt till *slices* att arbeta med under iterationen. Projektgruppen var nöjd med att arbeta baserat på en tydlig koppling mellan de funktionella kraven och iterationsplaneringen, men efter diskussion med kursansvarig om möjliga förbättringar så genomfördes vissa förändringar till iteration 4.

Förändringar gjordes därför att kursansvarig påpekade att även fast projektgruppen formulerade *stories* utifrån användningsfallsmodellen, och sedan delade upp dessa *stories* i *slices* så dokumenterades aldrig denna arbetsgång. Stegen som projektgruppen tog för att ta sig från användningsfallsmodell till *slices* “fanns” enbart i gruppmedlemmarnas huvuden.
Därför tog projektgruppen beslutet att till iteration 4 att skapa och skriva ut användningsfallskort som sattes upp på gruppens arbetstavla. När dessa användningsfallskort sedan delades upp i *slices* under iterationsstartsmötet så sattes dessa *slices* upp på en horisontell linje efter användningsfallet som givit upphov till dem. Denna förändring möjliggjorde att även personer som inte ingår i projektgruppen kan bilda sig en uppfattning om hur arbetet från kundens krav på produkten till iterativa arbetsuppgifter har gått till.

VI. Resultat

Resultaten presenteras i form av tabeller där bedömda projektmetoder som använts är listade tillsammans med gruppmedlemmarnas subjektiva värdering av deras

bidrag till att konstruera “En bra projektmetod för små IT-projekt”.

Fullständiga tabeller finns i bilaga A. Där finns också den evidens som ligger till grund för listade omdömen.

VII. Diskussion

Vad säger resultatet om ”Vad är en bra projektmetod för små IT-projekt”? Resultatet av undersökningen består av projektgruppens subjektiva omdömen av projektmetoderna som använts. Dessa omdömen och metoderna som har valts för att ta fram omdömena diskuteras här i syfte att identifiera svagheter, förbättringar och för att dokumentera eventuella slutsatser som kan dras baserat på resultatet.

L. Metoddiskussion

Kvalitativ undersökning:

Gruppen har använt delar av teorin som studerats till projektet och har på så sätt kunna undersöka och analysera dessa delar till undersökningen. Viktigt att notera är att tidsresurserna varit begränsade och därmed har inte alla projektmetoder som studerats kunnat praktiseras och undersökas. Den övergripande projektmetoden under den kvalitativa undersökningen har varit *EssUp*, där projektet har fokuserat mestadels på den agila projektmetoden *Scrum*. Den kvalitativa undersökningen har varit nödvändigt för att samla på sig erfarenheter och komma fram till ett resultat och har även tydliggjort mycket av teorierna som inlärts i litteraturstudien genom att dessa praktiserats.

Litteraturstudier:

Litteraturstudier har inneburit att studera in nödvändig kunskap för att kunna utföra projektmetoderna och även att studera in tidigare forskning och dokumentation om projektmetoder för att få flera perspektiv. Litteraturstudierna har även hjälpt med att få bredare kunskap om flera projektmetoder, och även då projektmetoder som inte praktiserades, för att på så sätt kunna jämföra med praktiserade projektmetoder och eventuellt förbättra dessa.

Validitet för undersökningsmetod:

Metodens validitet bedömer projektgruppen vara låg. Undersökningen är inte utformad av studenterna som utför den och förståelsen för undersökningsgenomförandet är relativt svag i början av projektet. Undersökningen utförs även i en universitetskursmiljö, vilken av naturliga skäl skiljer sig på många plan från exempelvis en arbetsplatsmiljö. En rimlig kritik av metodens validitet är därför att den snarare söker svar på frågan “Vad är en bra projektmetod för små IT-projekt för en universitetskurs?”.

Gruppmedlemmarna som har genomfört undersökningen är även oerfarna vetenskapsmän och har inte fått sitt arbete kritiskt granskat under undersökningens gång. Eftersom studenterna ansvarar för att genomföra projektet blir opartiskheten och dokumentationen lidande. En metod med högre validitet hade troligtvis kunna uppnås genom att låta studenterna enbart granska en annan projektgrupps arbete, men detta är inte gångbart i en universitetskurs av andra anledningar.

Reliabilitet för undersökningsmetod:

Ifall en undersökningsmetod är reliabel kan andra vetenskapsmän upprepa undersökningen med hjälp av samma instruktionsmaterial och komma fram till liknande resultat. Det är projektgruppens bedömning att även metodens reliabilitet är låg. Tar man i beaktande undersöknings snäva tidsram tillsammans med det extensiva teoretiska underlaget bedömer projektgruppen det vara troligt att de subjektiva bedömningarna av projektmetoderna som används kan variera rejält.

M. Resultatdiskussion

Resultatet visar att de grundläggande principerna som utgör utvecklingsramverket *scrum* har bedömts som positiva bidrag till projektarbetet av samtliga medlemmar i projektgruppen. *Scrum* är ett agilt ramverk som alla gruppmedlemmar kände till innan kursarbetet inleddes.

Enligt en undersökning utförd av *the Project Management Institute* (PMI) tredubblades användningen av agilt projektarbete mellan åren 2008 till 2013 [8, Kap. 1]. År 2013 publicerade även *VersionOne* att 88% av företagen de undersökt använde agila projektmetoder [8, Kap. 1]. Undersökningens resultat ligger således i linje med IT-industrins bedömning av agila projektmetoder.

Gällande övriga projektmetoder är det svårare att formulera en lika enhetlig slutsats. Projektgruppens indelning i formella roller ger upphov till en märkbar spridning av delar av projektmetoder som recenserats. Dessa omdömen kan vara av värde att begrunda men de ger inte en lika tydlig bild jämfört med de omdömen av projektmetoder som varit övergripande för hela projektet.

N. Framtida förbättringar

Framtida förbättringar som skulle kunna utveckla projektet skulle kunna vara att gruppen tar tid åt att reflektera kring hur deras projekt bidrar till FNs globala

mål[källa]. Många projekt som har gjorts kanske bidrar till att något av de 17 globala målen uppfylls, utan att de vet om det. Om det fanns en föreläsning som tog upp detta skulle de kunna undersöka hur deras projektmetoder eller slutgiltiga produkt leder till en mer hållbar värld.

En mindre studie om relationerna mellan olika roller skulle också vara väldigt lärorikt. Som exempel finns det en relations-kedja som börjar med en kund och kravansvarig som beskriver krav till en arkitekt som tillsammans med en utvecklingsansvarig ska kunna producera arkitekturer och konceptmodeller innan projektet startar. Just nu verkar det som att varje roll är som en isolerad ö med väldigt liten kontakt med de andra rollerna. Detta skulle dock medföra att mer tid skulle spenderas under den första iterationen.

En annan möjlig förbättring skulle kunna vara att studenterna i varje grupp utför någon form av kvantitativ undersökning. Detta kan göras genom att till exempel fråga andra studenter i kursen om hur de bedömde vissa projektmetoder eller att skicka enkäter till studenter från förra årets kursomgång med liknande frågor. Ett mer intressant alternativ skulle kunna vara att en ingenjör som faktiskt arbetar som arkitekt, projektledare, utvecklingsansvarig eller kund och kravansvarig. Intresset skulle nog växa mer från studenternas sida.

Slutord

Produkten som konstruerades är ett system, kallad Office Display, och består av en webbkomponent (webbapplikation) och hårdvara (MCU med en display, sensor och wifi-modul). En användare kan nyttja webbapplikationen för att skriva in meddelanden som kommer att synas i webbapplikationen och i hårdvarans skärm. All kod som skrevs under detta projekt finns i gruppens github repo, se [25].

Bilagor

Bilaga A: Tabeller till resultat
Bilaga B: Use Case Survey
Bilaga C: Arkitekturbeskrivning
Bilaga D: Utvecklingsplan
Bilaga E: Iterationsplan
Bilaga F: Komponentbeskrivning databas och databasintegrering
Bilaga G: Komponentbeskrivning STM32F303RCT7
Bilaga H: Komponentbeskrivning EA DOGM204-A
Bilaga I: Komponentbeskrivning Webbkomponenten

Referenser

[1] ”Projekt och projektmetoder (II1302) | KTH”. [Online]. Tillgänglig vid: https://www.kth.se/social/course/II1302/. [Åtkomstdatum: 09-maj-2019].

[2] V. Kumar, ”Sample IEEE Paper for A4 Page Size”.

[3] S. Eklund, *Arbeta i projekt: individen, gruppen, ledaren*, 4:e uppl., vol. 2011. Studentlitteratur.

[4] A. Katter, ”Förslag till en ekonomiskt h\a allbar projektmetod: En fallstudie vid Sydweb”, 2015.

[5] ”Kernel and Language for Software Engineering Methods (Essence). 1.0”, *OMG*. [Online]. Tillgänglig vid: https://www.omg.org/spec/Essence/1.0/. [Åtkomstdatum: 09-maj-2019].

[6] H. Kniberg, *SCRUM AND XP FROM THE TRENCHES - How We Do Scrum*, 2:a uppl. InfoQ.com.

[7] G. Azizyan, M. K. Magarian, och M. Kajko-Matsson, ”Survey of agile tool usage and needs”, i *2011 Agile Conference*, 2011, s. 29–38.

[8] C. G. Cobb, *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*. John Wiley & Sons, 2015.

[9] ”Use-Case 2.0 ebook”, *Ivar Jacobson International*, 21-juli-2014. [Online]. Tillgänglig vid: https://www.ivarjacobson.com/publications/white-papers/use-case-ebook. [Åtkomstdatum: 14-maj-2019].

[10] I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

[11] P. Kruchten, ”The 4 + 1 view model of architecture”, 1995, s. 42–50.

[12] ”Software Development Plan Template”. Systems Engineering Process Office, Code 20203 Space and Naval Warfare Systems Center San Diego.

[13] J. Conallen, ”Modeling Web Application Architecture with UML”, 1999.

[14] N. Andersson och A. Ekholm, ”Vetenskaplighet - Utvärdering av tre implementeringsprojekt inom IT Bygg & Fastighet 2002”, Institutionen för Byggande och Arkitektur, Lunds Universitet, 2002.

[15] B. Elvesæter, G. Benguria, och S. Illieva, ”A comparison of the Essence 1.0 specifications for software engineering methods”, presenterad vid Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering, 2013.

[16] B. Elvesæter, M. Striewe, A. T. Mcneile, och A.-J. Berre, ”Towards an Agile Foundation for the Creation and Enactment of Software Engineering Methods: The SEMAT Approach”, presenterad vid Second Workshop on Process-based approaches for Model-Driven Engineering, 2012.

[17] ”Essential Unified Process”, *Ivar Jacobson International*, 16-maj-2016. [Online]. Tillgänglig vid: https://www.ivarjacobson.com/services/essential-unified-process. [Åtkomstdatum: 21-maj-2019].

[18] ”Alpha State Cards | Agile Software Development | Ivar Jacobson International”. [Online]. Tillgänglig vid: https://www.ivarjacobson.com/alphastatecards.

[Åtkomstdatum: 03-juni-2019].
[19] ”What is Continuous Integration? – Amazon Web Services”, *Amazon Web Services, Inc.* [Online]. Tillgänglig vid: https://aws.amazon.com/devops/continuous-integration/. [Åtkomstdatum: 13-maj-2019].
[20] ”What is Continuous Delivery? – Amazon Web Services”, *Amazon Web Services, Inc.* [Online]. Tillgänglig vid: https://aws.amazon.com/devops/continuous-delivery/. [Åtkomstdatum: 13-maj-2019].
[21] ”What is DevOps? - Amazon Web Services (AWS)”, *Amazon Web Services, Inc.* [Online]. Tillgänglig vid: https://aws.amazon.com/devops/what-is-devops/. [Åtkomstdatum: 06-maj-2019].
[22] L. Chung, D. Gross, och E. Yu, ”Architectural Design to Meet Stakeholder Requirements”, i *Software Architecture*, vol. 12, P. Donohoe, Red. Boston, MA: Springer US, 1999, s. 545–564.
[23] P. Wilson och H. A. Mantooth, ”Chapter 6 - Block Diagram Modeling and System Analysis”, i *Model-Based Engineering for Complex Electronic Systems*, P. Wilson och H. A. Mantooth, Red. Oxford: Newnes, 2013, s. 169–196.
[24] ”Five Things Every Developer Should Know about Software Architecture”, *InfoQ*. [Online]. Tillgänglig vid: https://www.infoq.com/articles/architecture-five-things/. [Åtkomstdatum: 02-juni-2019].
[25] A. Morales, A. Ronne, S. Amner, och M. Hassan, ”II1302 Grupp 3 Github repo”, *GitHub Enterprise*. [Online]. Tillgänglig vid: https://gits-15.sys.kth.se/antmc/II1302. [Åtkomstdatum: 06-maj-2019].