

Linköpings universitet | Institutionen för datavetenskap
Examensarbete på grundnivå, 15hp | Data teknik
Vårterminen 2017 | LIU-IDA/LITH-EX-G--17/046--SE

3D-kopiering

Registrering och meshning av punktmoln för utskrift

**Hampus Dunström
Olof Holmberg
Gustav Jannering
Michael Karlsson
Martin Lundberg
Hannes Tuhkala
Fredrik Wallström**

Handledare : Sam Le
Examinator : Kristian Sandahl

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innehåller rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Hampus Dunström
Olof Holmberg
Gustav Jannerig
© Michael Karlsson
Martin Lundberg
Hannes Tuhkala
Fredrik Wallström

Sammanfattning

Tekniken för att kunna skriva ut 3D-objekt har funnits i många år men först på 2010-talet har 3D-skrivare blivit tillgängliga även för vanliga konsumenter. Det finns dock ett problem: för att kunna använda en 3D-skrivare måste man antingen kunna CAD-mjukvara eller förlita sig på andra människors 3D-modeller. Genom att använda ett system för 3D-kopiering kan ett verkligt objekt istället kopieras. Rapporten behandlar ett kandidatprojekt som genomfördes av sju studenter på datavetenskapliga civilingenjörsutbildningar på Linköpings universitet 2017. Målet med projektet var att utveckla ett system som kan ta separata punktmoln som indata för att sedan registrera dessa till ett komplett punktmoln. Utifrån detta kompletta punktmoln genereras sedan en 3D-mesh som kan skrivas ut av en 3D-skrivare. Tidigt i projektet var målet att vidareutveckla ett befintligt system. Detta mål omförhandlades med kunden efter att ett flertal fel identifierats med det befintliga systemet. Projektet resulterade i *3DCopy*, ett mjukvarusystem som registrerar punktmoln och utifrån dessa genererar en 3D-mesh.

Abstract

The technology to be able to print 3D objects has been available for many years, but it is only recently that 3D printers have been made available for regular consumers. There is one issue though: to be able to use the 3D printer either knowledge of CAD software or 3D models made by others are needed. By using a system for 3D copying a real object can instead be copied. This report presents a bachelor project that was done by seven students studying engineering programs in computer science or software technology at Linköping University, 2017. The goal of the project was to develop a system that could take several point clouds as input and then register them to a complete point cloud. Then use this point cloud to generate a 3D mesh to be printed on a 3D printer. The 3D printer will then be able to print the object. In the early stages of the project the main focus was to develop an already existing system. This goal was then renegotiated since the existing system contained several errors. The project resulted in *3DCopy*, a software system that registers point clouds and from these point clouds generates a 3D mesh.

Författarnas tack

Vi vill tacka institutionen för systemteknik vid Linköpings universitet och framförallt riktar vi ett tack till Maria Magnusson och Andreas Robinson för ett gott samarbete under projektets gång. Vi vill rikta ett stort tack till vår handledare Sam Le för hans stöd och feedback under projektets gång. Vi vill även tacka kursens examinator Kristian Sandahl för en givande kurs med många intressanta moment.

Innehållsförteckning

Sammanfattning	v
Författarens tack	vii
Innehållsförteckning	ix
Lista över figurer	xv
Lista över tabeller	xvi
1 Inledning	1
1.1 Motivering	1
1.2 Syfte	1
1.2.1 Vidareutveckling av TreeD	1
1.2.2 3DCopy	2
1.3 Frågeställning	2
1.3.1 Generella frågeställningar	2
1.3.2 Specifika frågeställningar	2
1.4 Avgränsningar	2
1.5 Definitioner	2
2 Bakgrund	5
2.1 Projektbakgrund	5
2.2 Tidigare erfarenheter	6
3 Teori	7
3.1 Punktmoln	7
3.2 Point Cloud Library	8
3.2.1 Filtrering	8
3.2.2 Registrering	9
3.2.3 Ytrekonstruering	9
3.3 3D-skrivare	9
3.4 Systemanatomi	10
3.5 Hållbar utveckling	10
4 Metod	11
4.1 Utvecklingsmetodik	11
4.2 Förstudie	11
4.3 Iteration 1	12
4.4 Iteration 2	12
4.5 Iteration 3	12
4.6 Metod för att fånga erfarenheter	13

5 Resultat	15
5.1 Systembeskrivning - vidareutveckling av TreeD	15
5.2 Systembeskrivning - 3DCopy	17
5.2.1 Arkitektur	17
5.2.2 Kommandoradsgränssnitt (CLI)	18
5.2.3 Grafiskt användargränssnitt (GUI)	18
5.2.4 Registrering	20
5.2.5 Meshning	20
5.3 Gemensamma erfarenheter	21
5.3.1 Övergripande projekterfarenheter	21
5.3.2 Erfarenheter gällande kommunikation	21
5.3.3 Erfarenheter gällande kvalitet	22
5.3.4 Erfarenheter av att bygga vidare på ett projekt	22
5.4 Översikt över individuella bidrag	22
6 Diskussion	23
6.1 Resultat	23
6.1.1 Lärdomar	23
6.1.2 Vad återstår för att uppnå fullt värde för kunden?	24
6.1.3 Tidigare projekt	24
6.1.4 Alternativa implementationssätt	24
6.2 Metod	25
6.3 Arbetet i vidare sammanhang	25
6.3.1 Hållbar utveckling	26
6.3.2 Specifika förbättringspunkter till vårt system	26
7 Slutsatser	29
7.1 Värde för kunden	29
7.2 Erfarenheter	30
7.3 Systemanatomi	30
7.4 Reviderad kravspecifikation	30
7.5 Vidareutveckling av ett system	31
A Hur påverkas ett team av sin arbetsmiljö?	33
— <i>Hampus Dunström</i> —	
A.1 Inledning	33
A.1.1 Syfte	33
A.1.2 Frågeställningar	33
A.1.3 Avgränsningar	33
A.1.4 Definitioner, akronym och förkortningar	33
A.2 Bakgrund	34
A.3 Teori	34
A.3.1 Belysning	34
A.3.2 Ljud	34
A.3.3 Kontor	35
A.3.4 Skillnader mellan små och stora företag	35
A.4 Metod	35
A.5 Resultat	36
A.6 Diskussion	37
A.6.1 Metod	37
A.6.2 Hur ser arbetsmiljön ut för teamen i kursen <i>Kandidatprojekt i programvaruveckling</i> ?	37
A.6.3 Hur påverkar arbetsmiljön sammanhållningen i teamet?	38

A.6.4	Hur påverkas gruppen av att få tillgång till ett eget rum där kandidatrabetet kan utföras jämfört med grupper som inte fått det?	38
A.7	Slutsatser	39
A.7.1	Hur ser arbetsmiljön ut för teamen i kursen <i>Kandidatprojekt i programvaruutveckling</i> ?	39
A.7.2	Hur påverkar arbetsmiljön sammanhållningen i teamet?	39
A.7.3	Hur påverkas gruppen av att få tillgång till ett eget rum där kandidatrabetet kan utföras jämfört med grupper som inte fått det?	39
B	Kontexters påverkan vid testning av GUI	41
<i>— Olof Holmberg —</i>		
B.1	Inledning	41
B.1.1	Syfte	41
B.1.2	Frågeställning	41
B.1.3	Avgränsningar	41
B.1.4	Definitioner	42
B.2	Bakgrund	42
B.3	Teori	42
B.3.1	Vad är interaktionskontext?	42
B.3.2	Betydelsen av kontext för interaktioner	42
B.3.3	Framtagning av testfall	43
B.3.4	Generering av testfall	43
B.4	Metod	44
B.4.1	Litteraturstudie	44
B.4.2	Testning av 3DCopys GUI med hänsyn till kontext	44
B.4.3	Testning av 3DCopys GUI utan hänsyn till kontext	46
B.5	Resultat	46
B.5.1	Litteraturstudie	46
B.5.2	Testresultat	46
B.6	Diskussion	47
B.6.1	Resultat	47
B.6.2	Metod	47
B.6.3	Källkritik	48
B.7	Slutsatser	48
B.7.1	Hur viktig är kontexten för interaktioner vid testning av GUI?	48
B.7.2	Hur bör testfall utformas för att ta hänsyn till kontexten?	48
B.7.3	Hade hänsyn till kontexter någon påverkan vid testning av 3DCopys GUI?	48
C	Hur kravhanteringsmetoder påverkar ett utvecklingsprojekt	49
<i>— Gustav Jannering —</i>		
C.1	Inledning	49
C.1.1	Syfte	49
C.1.2	Frågeställning	49
C.1.3	Avgränsningar	49
C.1.4	Definitioner, akronym och förkortningar	50
C.2	Bakgrund	50
C.3	Teori	50
C.3.1	IEEE standard 830	51
C.4	Metod	51
C.5	Resultat	51
C.5.1	Metoder för elicitering av krav	51

C.5.2	Erfarenheter	53
C.5.3	IEEE standard 830	53
C.6	Diskussion	54
C.6.1	För- och nackdelar med metoder för elicitering av krav	54
C.6.2	För- och nackdelar med IEEE std 830	55
C.6.3	Erfarenheter	55
C.6.4	Källkritik	55
C.6.5	Alternativa metoder	55
C.7	Slutsatser	56
C.7.1	Vilka metoder finns för kravhantering och elicitering av krav och vilka fördelar och nackdelar finns med dessa?	56
C.7.2	Vilka fördelar och nackdelar finns med att använda IEEE std 830 i ett programvaruutvecklings projekt av den storleken som detta projekt?	57
C.7.3	Vilka erfarenheter kan dokumenteras från kravhantering och elicitering av krav i projektet som kan vara intressanta för framtida projekt?	57
D	Analys av punktmolnsregistrering	59
<i>— Michael Karlsson —</i>		
D.1	Inledning	59
D.1.1	Syfte	59
D.1.2	Frågeställningar	59
D.1.3	Definitioner och förkortningar	59
D.1.4	Avgränsningar	59
D.2	Bakgrund	60
D.3	Teori	60
D.3.1	Registrering	60
D.3.2	Kinect Fusion	60
D.4	Metod	60
D.4.1	Litteraturstudie	60
D.4.2	Experiment	61
D.5	Resultat	61
D.5.1	Litteraturstudie	61
D.5.2	ICP vs. JR-MPC	62
D.5.3	Experiment	62
D.6	Diskussion	63
D.6.1	Metod	64
D.6.2	Resultat	64
D.7	Slutsatser	64
D.7.1	Hur skapas ett enhetligt punktmoln från bilder tagna med en fast avståndskamera?	64
D.7.2	Hur gör man för att välja algoritm och hur resonerade gruppen när de valde ICP?	65
E	Att bygga ett system i ROS	67
<i>— Martin Lundberg —</i>		
E.1	Inledning	67
E.1.1	Syfte	67
E.1.2	Frågeställning	67
E.1.3	Avgränsningar	67
E.2	Bakgrund	67
E.3	Teori	68
E.3.1	ROS	68

E.3.2	Pipe and filter-modellen	68
E.4	Metod	68
E.4.1	Arkitekturens framtagande	69
E.4.2	Systemet i ROS	69
E.4.3	Systemet utan ROS	69
E.4.4	Dokumentering av processen	70
E.5	Resultat	70
E.5.1	Arkitekturen i ROS	70
E.5.2	Arkitekturen utan ROS	70
E.6	Diskussion	71
E.6.1	Portabilitet	71
E.6.2	Att gå från ROS till klasser	72
E.6.3	Hållbarhet	72
E.7	Slutsatser	72
F	Verktyg som är lämpliga för att skriva stora dokument	75
<i>— Hannes Tuhkala —</i>		
F.1	Inledning	75
F.1.1	Syfte	75
F.1.2	Frågeställning	75
F.1.3	Definitioner och förkortningar	75
F.1.4	Avgränsningar	76
F.2	Bakgrund	76
F.3	Teori	76
F.3.1	L <small>A</small> T <small>E</small> X	77
F.3.2	Git	77
F.3.3	Apache Subversion	77
F.3.4	Mercurial	77
F.3.5	Kandidatrapport	77
F.4	Metod	77
F.4.1	Första frågeställningen	77
F.4.2	Andra frågeställningen	78
F.4.3	Tredje frågeställningen	78
F.5	Resultat	78
F.5.1	Litteraturstudie	79
F.5.2	Enkät 1	79
F.5.3	Enkät 2	81
F.6	Diskussion	82
F.6.1	Vilka fördelar och nackdelar finns det med att använda L <small>A</small> T <small>E</small> X eller Word för att skriva dokument?	82
F.6.2	Vilka verktyg lämpar sig för att skriva större dokument?	82
F.6.3	Vilka erfarenheter kan tas med ifrån det här projektet gällande framställning av större dokument?	83
F.6.4	Metod	83
F.6.5	Källkritik	83
F.7	Slutsatser	84
F.7.1	Vilka fördelar och nackdelar finns det med att använda L <small>A</small> T <small>E</small> X eller Word för att skriva dokument?	84
F.7.2	Vilka verktyg lämpar sig för att skriva större dokument?	84
F.7.3	Vilka erfarenheter kan tas med ifrån det här projektet gällande framställning av större dokument?	84

G Kvalitetsarbete i praktiken	85
<i>— Fredrik Wallström —</i>	
G.1 Inledning	85
G.1.1 Syfte	85
G.1.2 Frågeställning	85
G.1.3 Avgränsningar	85
G.2 Bakgrund	86
G.3 Teori	86
G.3.1 Kvalitetssäkring	86
G.3.2 Kodgranskning	86
G.4 Metod	86
G.5 Resultat	87
G.5.1 Litteraturstudie	87
G.5.2 Enkät	88
G.6 Diskussion	88
G.6.1 Källkritik	89
G.7 Slutsatser	89
Referenser	91
H Bilagor	95
H.1 Svar på enkät om arbetsmiljö	96
H.2 Projektdirektiv	97
H.3 Intervju guide	99
H.4 Svar på enkät om dokument	100
H.5 Svar på enkät om erfarenheter för dokument	101
H.6 Svar på enkät om kvalitetssäkrande process	102

Lista över figurer

1	Ett komplett punktmoln som representerar ett torusobjekt.	7
2	Ett icke komplett punktmoln som representerar en del av en kyrka.	8
3	Översikt över systemanatomin.	15
4	Diagram över ROS-noderna i den första versionen av systemet.	16
5	En skiss av det tänkta ROS-baserade systemet.	17
6	Översiktligt klassdiagram över arkitekturen.	18
7	Huvudvyn i 3DCopys webbaserade GUI.	18
8	Formuläret för att skapa ett registreringsjobb.	19
9	Formuläret för att skapa ett meshningsjobb.	19
10	Ett jobb i jobblistan.	19
11	Resultatet av registrering med 1, 5, 11, 17, 23, 29 respektive 35 punktmoln.	20
12	Mesh utan handpåläggning, objektet föreställer Einstein.	21
13	Mesh med handpåläggning, objektet föreställer Einstein.	21
14	GUI:t för 3DCopy i slutet av iteration 2.	44
15	EFG:n för 3DCopys GUI.	45
16	EIG:n för 3DCopys GUI.	45
17	Kyrkan som skannats samt ett resulterande punktmoln från en enkel skanning.	61
18	Till vänster: resultat från registrering med ICP. Till höger: resultat från registrering med JR-MPC.	63
19	Serie av delresultat från registrering med ICP.	63
20	Systemets arkitektur i ROS.	70
21	Systemets arkitektur utan ROS.	71
22	Visar resultatet från vilka verktyg som grupperna använder för de flesta större dokument.	79
23	Visar resultatet från vilka verktyg som deltagarna tycker är bra för större dokument.	81
24	Resultatet från om antalet deltagande män i en kodgranskningsprocess spelar någon roll.	88

Lista över tabeller

1	De sekvenser som ska användas för att ta fram testfallen.	45
2	De testfall som genomfördes på GUI:t med hänsyn till kontext.	46
3	De testfall som genomfördes på GUI:t utan hänsyn till kontext.	46



1 Inledning

1.1 Motivering

Tekniken för att kunna skriva ut 3D-objekt har funnits i många år men först på 2010-talet har 3D-skrivare blivit tillgängliga även för vanliga konsumenter. Även i industrin har det blivit allt vanligare att använda 3D-skrivare för att skapa prototyper. Det finns dock ett problem: för att kunna använda 3D-skrivaren måste man antingen kunna CAD-mjukvara eller förlita sig på andra människors 3D-modeller. Även om man kan CAD-mjukvara kan det vara svårt eller tidsödande att rita upp ett komplicerat objekt från grunden. Genom att använda ett system för 3D-kopiering kan ett verkligt objekt istället kopieras.

1.2 Syfte

Nedan beskrivs två syften med detta projekt. Det första syftet som beskrivs, vidareutveckling av TreeD, var syftet med projektet innan projektet reviderades medan det andra syftet, 3DCopy, var syftet med projektet efter revidering.

1.2.1 Vidareutveckling av TreeD

Syftet med detta projekt var till en början att konstruera en mjukvara för att styra ett system som kopierar tredimensionella objekt. Projektet skulle bygga vidare på ett tidigare kandidatarbete som utfördes våren 2016 och som kallas TreeD. Det kandidatarbetet resulterade i en mjukvara, också kallad TreeD, som styr ett rotationsbord och en linjärenhet. På denna linjärenhet finns det en avståndskamera som används för att generera punktmoln.

Syftet med projektet var alltså att bygga vidare på TreeD till ett system som kan kopiera ett tredimensionellt objekt. Tanken var att detta skulle göras genom att ett flertal skanningar tas med olika rotationer och lutningar. De genererade punktmolnen skulle sedan registreras till ett komplett punktmoln som sedan skulle omvandlas till en 3D-mesh. Denna 3D-mesh skulle sedan kunna skrivas ut med hjälp av en 3D-skrivare.

1.2.2 3DCopy

Med revideringen av projektet reviderades även syftet till att inte bero på TreeD. Det nya syftet var att utveckla ett mindre automatiserat system, framförallt med hänsyn till att skanna objekt. Systemet skulle kunna läsa in färdiga punktmoln som sparats på datorn. Genom att endast läsa in punktmoln från filer är programmet inte beroende av att det underliggande systemet för att styra hårdvaran är stabilt.

Efter att punktmolnen lästs in skulle de sedan behandlas genom att registrera alla punktmoln till ett komplett punktmoln och sedan generera en mesh utifrån detta. Den genererade meshen kan sedan skrivas ut med hjälp av en 3D-skrivare. På detta sätt skulle mjukvaran kunna användas till att kopiera tredimensionella objekt även om processen inte skulle vara helt automatiserad.

1.3 Frågeställning

De generella samt specifika frågeställningar som denna rapport kommer att behandla listas nedan.

1.3.1 Generella frågeställningar

1. Hur kan ett system för 3D-kopiering implementeras så att man skapar värde för kunden?
2. Vilka erfarenheter kan dokumenteras från ett 3D-kopieringsprojekt som kan vara intressanta för framtida projekt?
3. Vilket stöd kan man få genom att skapa och följa upp en systemanatomy?

1.3.2 Specifika frågeställningar

4. Hur påverkas projektet av en helt reviderad kravspecifikation efter hälften av projektets gång?
5. Hur har gruppen anpassat sitt tillvägagångssätt vid vidareutvecklingen av ett system?

1.4 Avgränsningar

Denna rapport hade innan projektets revidering avgränsningar att systemet inte skulle utveckla det tidigare systemet TreeD även om behov skulle finnas. Det visade sig att behov fanns och därfor reviderades projektet istället.

De nya avgränsningarna till projektet var att systemet enbart kommer behandla punktmoln tagna med den tillgängliga hårdvaran. Systemet ska enbart klara av att registrera dessa punktmoln och sedan generera en 3D-mesh för utskrift. Systemet ska inte behandla punktmolnen på något annat vis.

1.5 Definitioner

Här definieras vissa begrepp och förkortningar som används senare i rapporten.

- Iterative Closest Point (ICP) - En optimeringsalgoritm för att minimera avståndet mellan punkterna i två mängder av punkter. Vanligt använd för att registrera punktmoln.
- Point Cloud Library (PCL) - Ett C++-bibliotek för hantering av punktmoln.
- Robot Operating System (ROS) - Ett ramverk för att utveckla mjukvara för robotar.

- CLI - Fökrortning för det engelska uttrycket *command line interface*, kommandoradsgränssnitt på svenska.
- GUI - Fökrortning för det engelska uttrycket *graphical user interface*, grafiskt användargränssnitt på svenska.
- Pipe and filter-modellen - En vanlig modell inom mjukvaruutveckling. Går ut på att programmet är uppdelat i moduler där varje modul har strikt in- och utdata och där utdata från en modul kan skickas som indata i nästa modul.
- 3DCopy - Namnet på detta projekt samt namnet på den slutgiltiga produkten. En mjukvara som registrerar och meshar punktmoln.
- TreeD - Namnet på det tidigare projekt 3DCopy skulle bygga vidare på. Även namnet på en mjukvara för att styra ett rotationsbord och en linjärenhet med avståndskamera.
- Slack - Ett kommunikationsverktyg där kommunikationen kan delas in i kanaler. Används ofta av företag för att underlätta kommunikation inom projektgrupper.
- Trello - Ett onlineverktyg för att hantera listor. Användes i projektet för hålla reda på och följa upp aktiviteter som skulle utföras.
- Git - Ett distribuerat versionshanteringssystem.
- 3D-mesh - En vattentät modell av ett objekt som är uppbyggd av polygoner. Kallas även för bara mesh.
- Meshning - Att generera en 3D-mesh utifrån ett punktmoln.
- PCD-fil - Ett vanligt filformat för att spara punktmoln är PCD.
- STL-fil - Ett vanligt filformat för att spara en 3D-mesh är STL.
- CAD-mjukvara - Programvara för att designa fysiska objekt i datorn.
- Voxel - Ett punktmoln kan delas in i så kallade voxels vilket kan ses som kuber.
- Rotationsbord - En hårdvaruenhet som används för att rotera ett objekt.
- Linjärenhet - En hårdvaruenhet som används för att förflytta en avståndskamera eller annat föremål längs en axel.
- ROS-nod - En enskild process som kommunicerar med andra noder inom ramverket ROS.
- Nedampling - Metod för att minska antalet punkter i ett punktmoln med så liten förlust av information som möjligt.
- Parvis registrering - En teknik för att registrera punktmoln där man tar två punktmoln och registrerar de till ett resultatmoln.
- Styrkort - Ett kretskort som används för att styra hårdvara av något slag.



2 Bakgrund

2.1 Projektbakgrund

Kunden i detta projekt var avdelningen för datorseende (CVL) som är en del av institutionen för systemteknik (ISY) vid Linköpings universitet. CVL bedriver utbildning och forskning inom följande områden:

- signalbehandling
- bildanalys
- datorseende
- beräkningsfotografi
- detektion, följning och igenkänning av objekt
- skattning av pose och 3D-struktur
- robotseende och autonoma system
- medicinsk bildanalys och bildrekonstruktion.

CVL utlyste ett kandidatprojekt år 2016. Detta projekt hade som mål att utveckla ett system som skulle skanna tredimensionella objekt. Detta projekt och dess slutgiltiga produkt kommer häданefter att benämñas TreeD. Till förfogande för projektet fanns ett rotationsbord, en linjärenhet som innehåller en avståndskamera och två datorer för att styra dessa. Innan TreeD projektet, byggdes detta system av en doktorand som använde det i ett forskningsprojekt. Efter doktorandens forskningsprojekt beslutade CVL att systemet skulle användas vidare till kurser i bildsensorteknik och utlyste därför TreeD projektet. Projektet i sig bestod av att dekonstruera rotationsbordet, som var specialbeställt och levererades utan teknisk dokumentation, för att sedan implementera styrning av systemet.

Efter att TreeD utvecklats till att skanna tredimensionella objekt, utlyste CVL ett nytt kandidatprojekt. Det var detta projekt som denna rapport beskriver. Projekt och dess slutgiltiga produkt kommer hädanefter att benämñas 3DCopy. 3DCopy var en vidareutveckling av TreeD. Denna gång hade CVL som mål att använda 3DCopy i forskning kring punktmoln, 3D-skrivarteknik, avståndskamerateknik samt vidare forskning inom Point Cloud Library

(PCL), som är ett bibliotek till C++. CVL har även en tanke om att använda 3DCopy för att skapa hinder som kommer att användas i deras nya labb för obemannade farkoster och att konstruera laborationer som använder systemet.

Halvvägs in i 3DCopy projektet bestämdes i samråd med kund, handledare och examinatör att en omförhandling av 3DCopys mål borde göras. Detta på grund av att TreeD inte motsvarade de krav och förväntningar som var ställda på 3DCopy. Målet med 3DCopy reviderades därför till vidare forskning kring punktmoln, PCL och olika 3D-skrivartekniker. 3DCopy projektet genomfördes således oberoende av TreeD på grund av att TreeD systemet inte gick att använda på det sätt som krävdes för 3DCopys ändamål. Vid önskan om ett komplett system, ifrån att skanna ett objekt till utskrift, krävs en genomgång och revidering av TreeD.

Den största förändringen som skedde för 3DCopys projektgrupp var att de uppsatta kraven fick revideras till att inte bero på TreeD. Istället fick projektgruppen sätta upp krav som tog färdiga punktmoln som indata, utan att skanna hela objektet som en del av systemet.

2.2 Tidigare erfarenheter

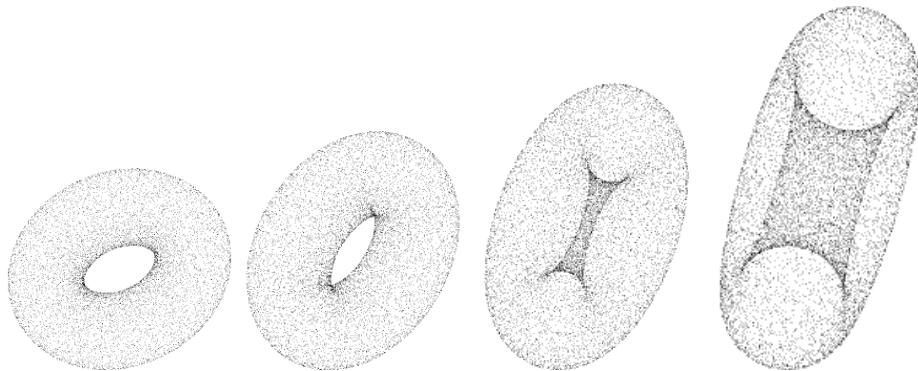
Projektgruppen består av studenter vid civilingenjörsutbildningen i data teknik och civilingenjörsutbildningen i mjukvaruteknik vid Linköpings universitet. Tidigare erfarenheter inom mjukvaruutveckling har alla gruppmedlemmar fått ifrån tidigare kurser inom respektive program. För de medlemmar som läser data teknik är projektkursen *Konstruktion med mikrodatorer* den kurs som gett mest erfarenhet av att jobba med ett utvecklingsprojekt i grupp. Kursen gick ut på att konstruera en robot tillsammans i grupper om sex studenter. För de medlemmar som läser mjukvaruteknik är det kursen *Artificiell intelligens - projekt* som gett mest erfarenhet av att jobba med ett utvecklingsprojekt i grupp. Kursen gick ut på att identifiera relevanta AI-tekniker och litteratur som beskriver dem för att sedan utvärdera och jämföra dessa tekniker relativt varandra. Detta gjordes i grupper om 4-6 personer. Slutligen implementerades och integrerades dessa AI-tekniker i ett valfritt system. Samtliga gruppmedlemmar har tidigare haft dåliga erfarenheter av projekt som styrs med en för lös hand. Gruppen beslutade därför i ett tidigt skede för en strikt uppdelning av roller med förutbestämda möten, kommunikationskanaler och aktivitetsuppdelning.

3 Teori

3.1 Punktmoln

Ett punktmoln är en mängd av punkter i ett tredimensionellt koordinatsystem. Varje punkt i punktmolnet består av tre värden som representerar punktens XYZ-koordinater. Tillsammans kan dessa punkter representera ett objekt. Detta görs ofta genom att punkterna representerar utkanterna av objektet. [42].

Det finns två typer av punktmoln, ett så kallat komplett punktmoln och ett icke komplett punktmoln. Ett komplett punktmoln, se figur 1, betyder att punktmolnet innehåller samtliga punkter som behövs för att representera objektet. Ett icke komplett punktmoln, se figur 2, betyder att punktmolnet endast representerar en del av objektet. Flera stycken icke kompletta punktmoln kan kombineras för att skapa ett komplett punktmoln. Denna process kallas registrering och kommer förklaras noggrannare senare i rapporten.



Figur 1: Ett komplett punktmoln som representerar ett torusobjekt.



Figur 2: Ett icke komplett punktmoln som representerar en del av en kyrka.

3.2 Point Cloud Library

Point Cloud Library (PCL) är ett bibliotek till C++ som bidrar med algoritmer till att behandla 3D-objekt. PCL innehåller bland annat algoritmer för:

- filtrering
- registrering
- ytrekonstruering.

PCL är alltså ett bibliotek avsett att ge stöd till att behandla punktmolnsdata, i form av olika bearbetningsalgoritmer [46].

PCL använder sig av ett eget visualiserbibliotek för att visualisera punktmoln. Visualiserbiblioteket i PCL integrerar med visualiseringssverktyget VTK [54][46].

Nedan beskrivs de olika algoritmerna från PCL som är relevanta för projektet.

3.2.1 Filtrering

PCLs filtreringsbibliotek innehåller algoritmer för att filtrera bort skräppunkter från ett punktmoln. Dessa skräppunkter kan uppstå vid skanning av ett objekt eftersom avståndskameran kan reagera på bakgrunden och inte bara på objektet. Detta gör att skräppunkter infaller i punktmolnet. För att kunna behandla punktmolnet måste dessa skräppunkter först filtreras bort.

PCL har stöd för flera olika metoder för filtrering. Ett sätt att filtrera punktmoln är att ta bort alla punkter som ligger utanför ett visst intervall av koordinater. Syftet med detta är att ta bort sådan som måste ligga utanför det skannade objektet. En annan metod är att för varje punkt i punktmolnet kontrollera hur många grannar den punkten har inom en viss radie. Alla punkter som har under ett visst antal grannar tas bort vilket leder till att endast punkter som inte ligger i närheten av något objekt tas bort. Ofta används dessa filtreringsmetoder tillsammans för att försäkra sig om att all skräpdata tagits bort [36].

För att undvika att punktmoln blir för stora, det vill säga att de får för många punkter vilket gör att de tar för lång tid att behandla, kan nedsampling användas för att ta bort punkter och samtidigt uppskatta den yta som representeras så nära som möjligt. För att göra detta i PCL delas punktmolnet som ska nedsampelas in i voxels, vilket kan ses som att punktmolnet delas in i ett antal kuber. Sedan ersätts alla punkter i varje voxel med en punkt som ligger i centrum för alla punkterna i voxeln. På detta sätt ges en nära uppskattning av den representerade ytan med mycket färre punkter än det ursprungliga punktmolnet [10].

3.2.2 Registrering

Att kombinera två punktmoln till ett punktmoln som innehåller samma information kallas för att registrera punktmoln. Detta är en komplicerad process där både translationen och rotationen, runt alla axlar, kan behöva ändras. Målet med parvis registrering är att flytta ett punktmoln till det andra punktmolnet så att alla punkter i de två separata punktmolnen överlappar med varandra. I PCL finns det ett registreringsbibliotek som innehåller algoritmer för registrering [39].

För att registrera punktmoln finns ett antal olika algoritmer med olika för- och nackdelar. En vanlig algoritm är *Iterative Closest Point* (ICP). Det är en så kallad parvis registreringsalgoritm som registrerar enbart ett punktmoln till ett annat. ICP går ut på att minimera summan av avståndet från varje punkt i det ena punktmolnet till varje punkt i det andra punktmolnet.

För att begränsa antalet punkter som måste beräknas används parametern *Max correspondence distance* så att enbart de punkter inom avståndet räknas in i summan för varje punkt. *Transformation epsilon* är ett mått på hur mycket transformationen för punktmolnet som ska matchas in får ändras för varje iteration. Genom att öka detta epsilon kan, vid behov, algoritmen flytta punktmolnet längre och därmed nå sitt mål på färre iterationer. Nackdelen med ett för högt epsilon är dock att punktmolnet kan flyttas förbi optimum som därmed slösar tid. Slutligen kan *Max iterations* användas för att avgöra hur många gånger algoritmen ska köras för varje parvis registrering. Genom att öka antalet iterationer ökar tidsåtgången men resultatet förbättras också [38].

3.2.3 Ytrekonstruering

För att återskapa ytor från en 3D-modell används PCLs ytrekonstrueringsbibliotek. I projektets sammanhang betyder det att ett punktmoln används för att skapa en vattentät modell. De algoritmer som biblioteket tillhandahåller kan användas för att generera en mesh utifrån ett komplett punktmoln [41].

Meshning av ett punktmoln görs efter att punktmolnen registrerats så de bildar ett komplett punktmoln. Att skapa en 3D-mesh av ett komplett punktmoln innebär att skapa en yta utan punkter. Punktmolnets 3D-mesh är alltså en vattentät 3D-modell uppbyggd av polygoner. En algoritm som kan användas för att mesha ett punktmoln är en trianguleringsalgoritm vid namn *Poisson surface reconstruction*. Denna algoritm körs på ett punktmoln med normaler för att erhålla en yta baserad på grannarna till respektive punkt. För detta krävs alltså ett punktmolnsobjekt som innehåller normalerna till varje punkt i punktmolnet. I detta punktmoln med normaler kan sedan önskade parametrar sättas innan själva trianguleringsalgoritmen körs för att generera den önskade meshen [41][40].

3.3 3D-skrivare

En 3D-skrivare kan användas för att skriva ut 3D-objekt i olika material vanligast är plast. Detta görs utifrån datormodeller som skapats i till exempel CAD eller med system som 3DCopy.. För att kunna göra detta har skrivaren ett styrkort som exekverar så kallad G-code [15]. Denna G-code beskriver vilka motorer som ska flyttas och hur mycket. En 3D-modell som är sparad i datorn, i form av en mesh, måste alltså behandlas genom att använda en algoritm för att generera G-code utifrån modellen. Detta görs ofta genom en typ av programvara som kallas för slicer. Namnet kommer av att programmet delar upp modellen i olika lager och sedan genererar hur skrivaren måste röra sig för att skriva ut ett lager. På detta sätt byggs sedan hela den ursprungliga modellen upp lager för lager. Två vanliga slicer-programvaror är *Cura* [7] och *Slic3r* [49].

3.4 Systemanatomi

En systemanatomi är en beskrivning av ett systems funktionalitet. Ofta består en systemanatomi av en graf, där noder representerar systemets olika funktioner och bågar representerar beroenden.

Systemanatomin säger ingenting om hur systemet ska implementeras men kan vara till stor hjälp för en utvecklingsgrupp då de lättare kan förstå vad systemet ska göra. En systemanatomi fungerar inte som ersättning för andra modeller men är ett bra komplement till dessa [51].

3.5 Hållbar utveckling

Hållbar utveckling är en relevant aspekt i dagens utveckling av programvaror eftersom de används kontinuerligt av samhället. En programvara som utvecklats kan både hjälpa samhället framåt men också påverka samhället negativt, beroende på hur den producerats och hur den används [44].

Enligt Lago et al. [26] definieras hållbarhet som "*förmåga att uthärda och bevara funktionen hos ett system under en utsträckt tidsperiod*". Lago et al. [26] definierar även fyra områden som de utmärkande områdena inom hållbarhet. Dessa fyra områden listas nedan.

- Ekonomiskt - Systemet ska bevara marknadsvärde och kapital.
- Socialt - Systemet ska underhålla samhället.
- Miljö - Systemet ska skydda den mänskliga välfärden genom att skydda naturens tillgångar.
- Tekniskt - Systemet ska utvecklas för att stödja långtidsanvändning.

En mer ingående beskrivning om hur 3DCopy främjar hållbar utveckling diskuteras i avsnitt 6.3.1.



4 Metod

4.1 Utvecklingsmetodik

Gruppen använde sig av en iterativ och agil process med vissa inslag av vattenfallsmodellen. Utvecklingsledaren ansvarade för att ta fram vad som skulle genomföras i varje iteration och planera iterationerna i samråd med övriga gruppmedlemmar. Teamledaren ansvarade för att planera de administrativa delarna av projektet. För att planera en iteration utgick gruppen från kravspecifikationen och bröt ner den till aktiviteter. Målet med varje aktivitet var att den skulle vara genomförbar på mindre än en vecka. När aktiviteterna var skapade gjorde utvecklingsledaren med inrådan från bland annat arkitekten en tidsuppskattning på varje aktivitet och hur många som borde jobba med aktiviteten. Efter planeringen gick gruppen igenom aktiviteterna för att alla i gruppen skulle bli informerade om vad som skulle genomföras under iterationen. När alla i gruppen var informerade om aktiviteterna valde gruppmedlemmarna vilka aktiviteter som de ville delta i. Vidare under iterationen planerades hur aktiviteten skulle utföras av de gruppmedlemmar som valt den aktiviteten.

4.2 Förstudie

Projektet inleddes med en förstudiefas där gruppen sammanställde diverse dokument. Där dessa dokument användes för att både gruppmedlemmarna själva och externa parter skulle få en bättre insikt i vad projektet gick ut på och vad dess mål var. Dokumenten utformades och togs fram av den eller de personer i gruppen som var bäst lämpade för ett visst dokument. Med det menas den person som passade bäst för ett visst dokument med avseende på dennes roll i gruppen.

Gruppen började sedan att studera de områden, verktyg och ramverk som skulle behövas för projektet. Den information som gruppen hade från början var den som kunden tillhandahållit och utifrån denna information upptäcktes en del nya områden som behövde undersökas. Under förstudien delade utvecklingsledaren ut olika områden att undersöka. Efter ett område undersökts skrevs ett kort dokument som beskrev slutsatserna och innehöll relevanta länkar från undersökningen. De största områdena som undersöktes var ROS och PCL. Först undersöktes ROS och PCL av några ur gruppen samtidigt som de andra arbetade på diverse dokument. Därefter undersökte alla i gruppen ROS och ett par stycken undersökte PCL. Alla

gruppmedlemmar genomförde de guider som rekommenderades på ROS hemsida. Efter att guiderna var genomförda skrev alla gruppmedlemmar varsin chattklient med hjälp av ROS.

4.3 Iteration 1

Iteration 1 omfattar de två första veckorna efter förstudiefasen. Huvudmålet för iteration 1 var att färdigställa basfunktionaliteten för produkten. Gruppen började med att skriva kod för att skapa ett gränssnitt mot TreeD och börja testa att registrera och mesha en del punktmoln som laddades ner från olika källor på internet. Efter en veckas utvecklande började majoriteten av gruppen att arbeta med de dokument som skulle lämnas in efter iteration 1. Några gruppmedlemmar fortsatte med att färdigställa arbetet på produkten även under andra veckan.

4.4 Iteration 2

Iterationen inleddes med fokus på registrering då detta var det stora frågetecknet i projektet vid början av iteration 2. Efter att först ha undersökt olika algoritmer beslutades det att gruppen skulle fortsätta med ICP då den hade bäst stöd i PCL-biblioteket. Efter det delades gruppen i två för att arbeta med två olika algoritmer för att registrera punktmoln. Den första algoritmen var att försöka vrida och placera punktmolnen rätt från början och på så sätt få det enklare att sedan registrera punktmolnen med ICP. Vrida och placera punktmolnen rätt innebär att först rotera dem så att de har rätt rotation i förhållande till varandra och sedan sätta origo till samma punkt för alla punktmolnen. Problemet med att få algoritmen att fungera var att den behövde vara väldigt generell vilket gjorde den svår att ta fram och fullända. Speciellt för de objekt som i början av projektet ansågs enkla att registrera av gruppen och kunden. Ett exempel på dessa enkla objekt var släta rätblock. Den andra algoritmen var mer anpassad för de då ansedda enkla objekten. Den gick ut på att i första hand endast lösa problemet med att registrera rätblocksliknande föremål. Detta gjorde den genom att hitta den plattaste ytan, anta att det var en sida och sedan ta fyra skanningar med 90 graders rotation mellan varje. Dessa punktmoln registrerades sedan på ett genererat rätblock med ICP.

Den andra algoritmen var den som utforskades mest tills den skulle börja testas ihop med hårdvaran. Då upptäcktes att det fanns mängder med fel på det underliggande systemet som tog fram punktmolnen. Det gick inte att genomföra tillräckligt många skanningar i rad utan att det underliggande systemet kraschade.

Detta gjorde att gruppen tillsammans med kunden fick revidera en stor del av projektet där gruppen bortsåg från det tidigare systemet helt och hållet. Gruppen beslutade också tillsammans med kunden att det borde vara lättare att registrera mer detaljerade objekt. Gruppen gick då tillbaka till algoritmen för att placera rätt och sedan registrera. Denna nya registrering placerades i ett helt nytt program, 3DCopy. Gruppen började då bygga ett CLI och integrera meshning och registrering i det programmet. Istället för att som tidigare bygga allt som enskilda ROS-noder.

4.5 Iteration 3

Iteration 3 omfattade fyra veckor efter iteration 2. Den första veckan arbetade gruppen med att färdigställa de individuella delarna av kandidatrapporten inför inlämning. En del arbete genomfördes också på den gemensamma rapporten som också skulle lämnas in. Rapporten lämnades sedan in i slutet av första veckan. Sedan under den andra veckan arbetade gruppen på den gemensamma delen av rapporten. En del komplettering av de individuella delarna gjordes också då gruppen fick feedback på första veckans inlämning redan i början av andra veckan i iteration 3.

Resterande två veckor arbetades det med att bryta ut registrering och meshning från ROS-systemet och integrera det med det nya programmet 3DCopy. Därefter färdigställdes 3DCopy med kommandoradsgränssnitt och grafiskt användargränssnitt. Det skrevs också mjukvara för att kunna ta fram och filtrera punktmoln från TreeDs system. Parallelt med detta genomfördes opponering samt förberedelser för redovisning av erfarenheter och slutpresentation.

4.6 Metod för att fånga erfarenheter

Erfarenheter fångades upp under projektets gång med hjälp av kontinuerlig utvärdering och diskussioner under gruppens gemensamma möten. Dessa möten skedde två gånger i veckan, varje måndag och torsdag. På måndagar deltog även handledaren och då låg fokus på en statusuppdatering och vad som skulle göras i veckan. Under statusuppdateringen diskuterades vad som gått bra, vad som gått dåligt och hur gruppen skulle kunna förbättra saker och ting. Under torsdagsmöten lades fokus på diskussion kring viktiga frågor som hela gruppen behövde vara del av. Erfarenheterna som kommit upp på dessa möten, framförallt måndagsmötena sammanfattades av teamledaren i varje veckas statusrapport.

Erfarenheter har också delats av gruppen under arbetets gång. Då gruppen arbetat större delen av tiden i samma rum har det skett mycket diskussion i gruppen kring olika problem öppet i rummet. Slutligen har gruppmedlemmarna under projektets gång samlat de erfarenheter som de fått i den här rapporten. Därmed så har erfarenheter samlats in kontinuerligt under genomförandet av projektet.

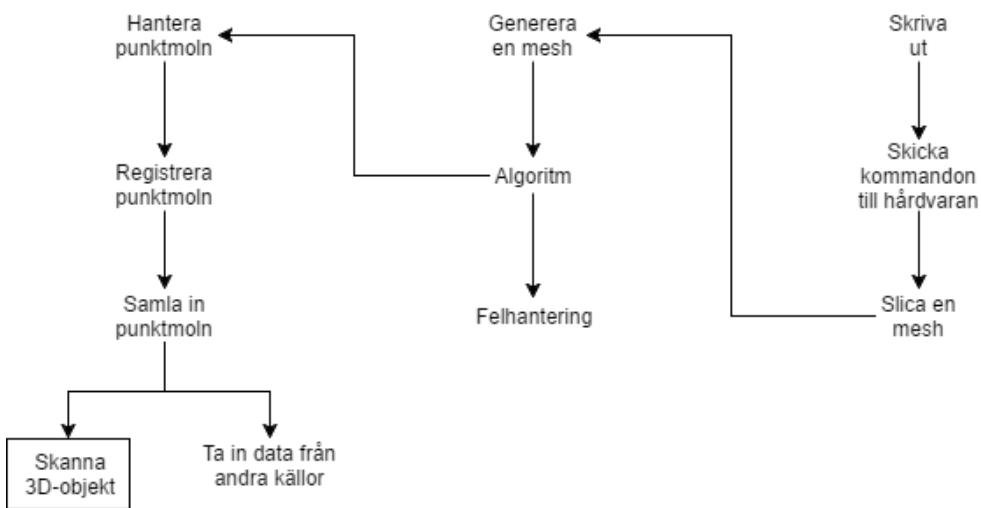
5 Resultat

5.1 Systembeskrivning - vidareutveckling av TreeD

Nedan följer en beskrivning av det ROS-baserade system som byggde vidare på TreeD och som utvecklades innan kraven i projektet omförhandlades med kunden.

Systemet bestod hårdvarumässigt av ett rotationsbord, en avståndskamera och en linjärenhet för att flytta avståndskameran längs en axel. För att styra detta fanns det sedan tidigare mjukvara, TreeD, som kan utföra en enskild skanning. Projektet skulle ha resulterat i en mjukvara för att utföra fler skanningar från olika vinklar och sammanfoga dessa till ett komplett objekt som kunde skrivas ut på en 3D-skrivare.

Tidigt i arbetet togs en systemanatomy fram, se figur 3, som beskriver vilken funktionalitet systemet skulle ha. Denna hjälpte även till med att dela in närliggande funktionalitet i moduler och låg som underlag för arkitekturbeskrivningen.



Figur 3: Översikt över systemanatomin.

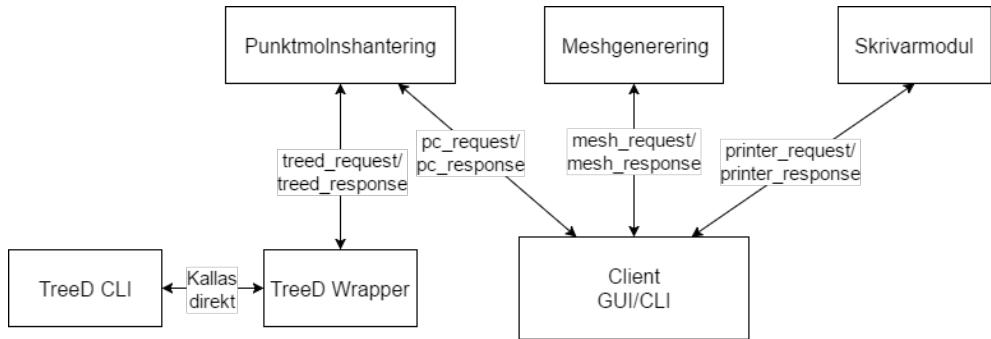
Arkitekturbeskrivningen som togs fram för systemet förklrarar hur systemet skulle fungera och vilka moduler som skulle finnas. Arkitekturen gav en bra utgångspunkt för

5. RESULTAT

hur systemet skulle implementeras. I arkitekturbeskrivningen finns det förklarat vilka noder systemet skulle bestå av och hur de skulle kommunicera med varandra.

Denna arkitektur var uppbyggd i ROS vilket innebär att de flesta noder i det tänkta systemet fungerade som en service som tog emot ett anrop för att utföra någonting och när den var klar svarade den med resultatet av arbetet. Detta gjorde att alla noder hade tydliga gränssnitt mellan varandra vilket resulterade i en bra separation mellan olika moduler.

Systemet var uppbyggt enligt en förenklad variant av *pipe and filter*-modellen, där resultatet av ett steg skickas vidare som indata till nästa steg i modellen. Detta flöde styrdes av klienten så att användaren kunde välja att köra hela eller delar av processen. Figur 4 visar hur systemet var uppbyggt.



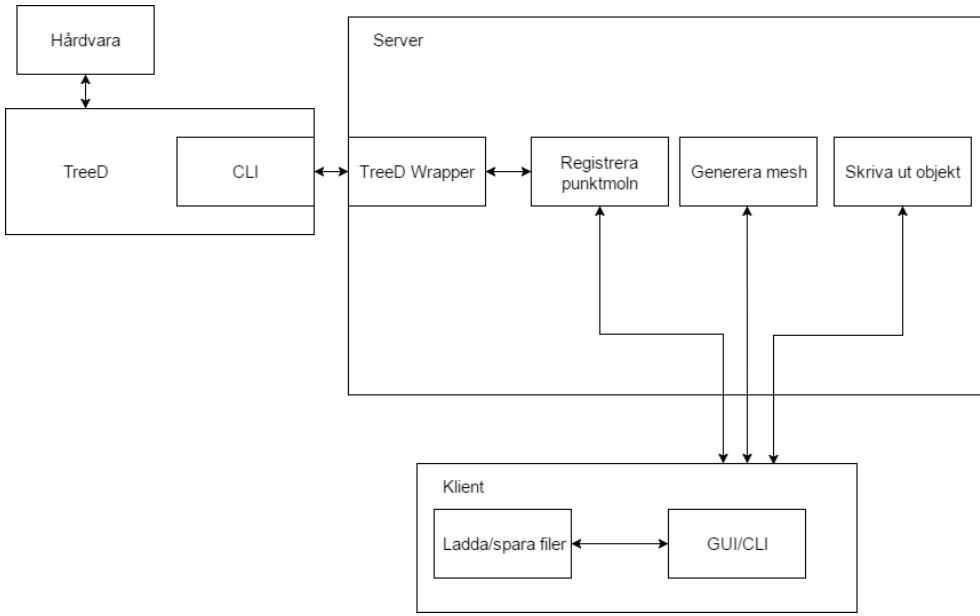
Figur 4: Diagram över ROS-noderna i den första versionen av systemet.

Det tänkta systemet kunde samla in fler punktmoln. Detta gjordes av punktmolnshanteringsnoden som tog in ett värde för hur noggrant objektet skulle skannas. Punktmolnshanteringsnoden utförde ett antal skanningar genom att noden TreeD-wrapper kallades. Sedan registrerade punktmolnshanteringsnoden dessa inkompleta punktmoln till ett punktmoln som representerade hela objektet.

När ett komplett punktmoln hade genererats skickades det vidare till meshgenereringsnoden som uppskattade en tredimensionell yta bestående av polygoner, även kallad mesh, utifrån punktmolnet. Denna mesh kunde sedan användas för att generera kod som kunde köras på en 3D-skrivare för att skriva ut objektet.

Mycket tid under de första iterationerna av projektet spenderades på att utforska och undersöka olika tekniker och algoritmer. Som resultat av det finns det mycket nyttig kunskap i projektgruppen och även en del testkod för att utföra olika delar.

Det ROS-baserade systemet var uppdelat i en serverdel och en klient. Tanken var att servern skulle köras på en dator som var fysiskt kopplad till hårdvaran och på så sätt kunde styra den genom TreeD-wrappern. Klienten skulle i första hand köras på samma dator och styra servern men med möjligheten att köras på en annan dator och styra servern över nätverket. Se figur 5 för en bild på hur det tänkta systemet var planerat att vara strukturerat.



Figur 5: En skiss av det tänkta ROS-baserade systemet.

Som kan ses i figurerna 4 och 5 var klienten den centrala delen i programmet. Den hanterade användarens indata och använde sig av de olika noderna för att utföra det användaren vill. Konceptet var att systemet skulle använda en *pipe and filter*-arkitektur där klienten skickade runt data i pipen men att användaren, genom klienten, hade möjlighet att granska resultatet och göra viss handpåläggning mellan de olika stegen. Detta genom att klienten alltid kunde spara resultatet från ett steg till en fil och använda andra program för handpåläggningen innan den skickades vidare i systemet.

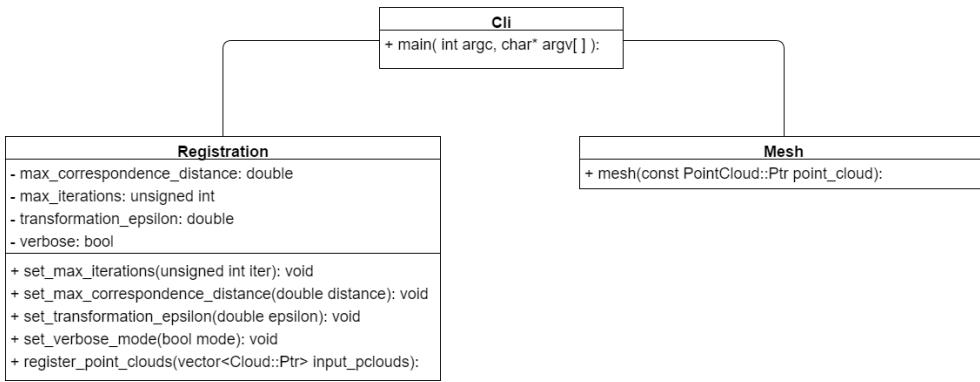
5.2 Systembeskrivning - 3DCopy

Som kan läsas tidigare i denna rapport omförhandlades projektets krav och mål med kunden under projektets gång. Detta ledde till att gruppen gjorde om systemets arkitektur. ROS-arkitekturen som presenterats i avsnitt 5.1 ersattes. Istället skapades ett objektorienterat program, skrivet i C++. Detta program utför både registrering, filtrering och meshning. När sedan meshen genererats kan programmet skriva ut objektet med hjälp av en 3D-skrivare. Till detta program finns också ett CLI och ett GUI.

5.2.1 Arkitektur

Programmet består av två större klasser, *Registration* och *Mesh*. Dessa två klasser används sedan av *Cli* vilket visas i figur 6. Det har lagts stor vikt vid modulärheten i programmet då *Registration* och *Mesh* ska fungera självständigt i olika program.

5. RESULTAT



Figur 6: Översiktligt klassdiagram över arkitekturen.

5.2.2 Kommandoradsgränssnitt (CLI)

Systemet kan kontrolleras av ett CLI som kan ta in olika parametrar för att anpassa registreringen. Det finns även alternativ för att bara registrera eller bara mesha objekt. Under en körning av programmet med hjälp av kommandoradsgränssnittet väljs vilka filer eller mappar med filer programmet ska läsa in och använda. Hur de används beror på alternativen. Ett exempel på ett anrop är:

```
3DCopy -v --max-corr-dist 15 path/to/register/ output
```

Det anropet kommer registrera och sedan mesha punktmolnsfilerna i mappen *path/to/register/* och döpa det kompletta punktmolnet respektive färdiga meshen till *output.pcd* och *output.stl*. Under registreringen och meshningen kommer programmet att skriva ut information om processen på grund av *-v* flaggan som står för *verbose mode*. Med flaggan *--max-corr-dist* sätts värdet på *Max correspondence distance*, se avsnitt 3.2.2. Om man undrar vilka alternativ som finns kan man använda *-h* flaggan. Då skriver programmet ut de alternativ som finns samt hur man använder programmet med hjälp av kommandoradsgränssnittet.

5.2.3 Grafiskt användargränssnitt (GUI)

För att lättare kunna styra systemet utvecklades också ett webbaserat GUI. GUI:t har samma funktionalitet som CLI:t och använder CLI:t för att köra meshning och registrering. Huvudvyn för det webbaserade GUI:t kan ses i figur 7.



Figur 7: Huvudvyn i 3DCopys webbaserade GUI.

De funktioner som finns i GUI:t är *Add Registration Job* och *Add Mesh Job*. *Add Registration Job* öppnar ett formulär där parametrar på registreringen ska sättas och de filer som ska registreras ska väljas. Formuläret för registrering kan ses i figur 8.

New registration job

Name

Log level

Max Correspondence

Max iterations

Transformation Epsilon

Leaf Size

Point Clouds

Välj filer | Ingen fil har valts

Create Back

Figur 8: Formuläret för att skapa ett registreringsjobb.

Add Mesh Job öppnar ett formulär där parametrar på meshningen ska sättas och den fil som ska meshas ska väljas. Formuläret för meshning kan ses i figur 9.

New Mesh job

Name

Log level

Point Cloud

Välj fil | Ingen fil har valts

Create Back

Figur 9: Formuläret för att skapa ett meshningsjobb.

Det finns även tre funktioner för varje jobb som finns i jobblistan. De funktionerna är *View Job*, *Start Job* och *Delete Job*. Ett jobb och dess tillhörande funktioner kan ses i figur 10.

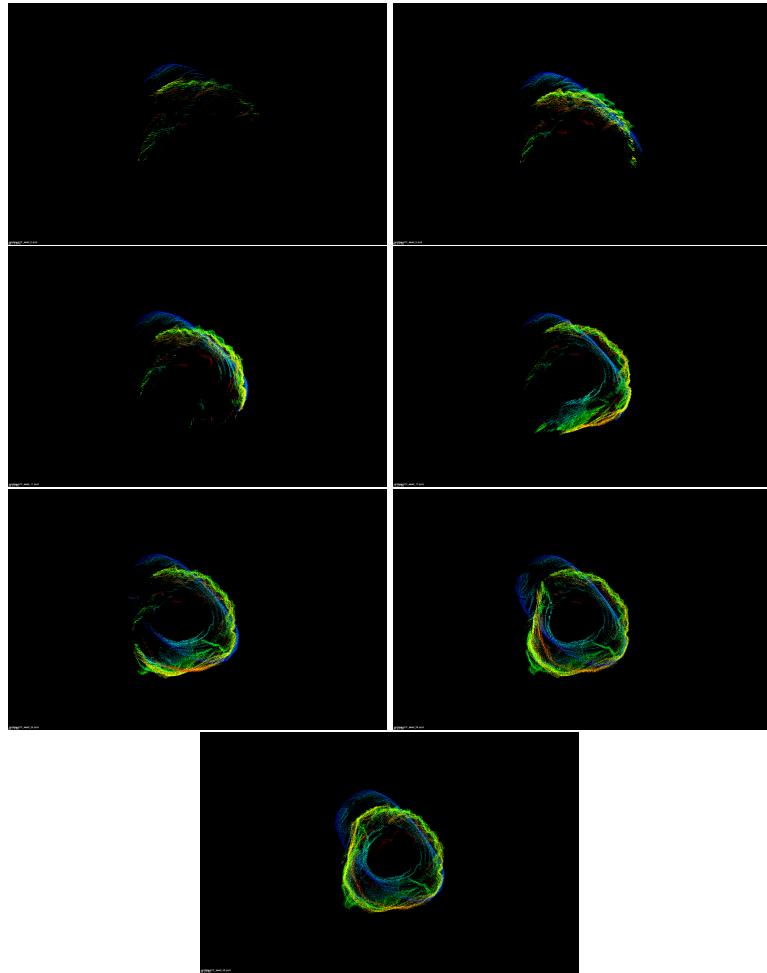
Registration	reg_ein	May 22, 2017, 7:56 a.m.	None	None	View Job	Start Job	Delete job
--------------	---------	-------------------------	------	------	--------------------------	---------------------------	----------------------------

Figur 10: Ett jobb i jobblistan.

View Job visar information om det specifika jobbet, bland annat de filer som är associerade och de parametrar som valdes för jobbet. *Start Job* startar jobbet på servern, jobben körs inte automatiskt efter de har skapats utan de måste manuellt startas. Slutligen kan *Delete Job* väljas. *Delete Job* tar bort jobbet, det vill säga all information om jobbet i databasen och alla filer på servern som associeras med jobbet.

5.2.4 Registrering

Systemet registrerar punktmoln parvis genom att använda PCLs implementation av ICP. De punktmoln som läses in i antingen CLI:t eller GUI:t behandlas av registreringklassen och utför registreringen. För att registreringen ska fungera för olika fall kan användaren själv sätta värden på parametrarna *Max correspondence distance*, *Transformation epsilon* och *Max iterations*. I figur 11 kan en bildserie ses. Denna bildserie föreställer resultatet från registrering med 1, 5, 11, 17, 23, 29 respektive 35 punktmoln. Punktmolnen som har använts vid registreringen i bilden föreställer en byst av Einstein och är tagna med tio graders skillnad. Bildserien visar hur bystens konturer växer fram allt efter som att fler punktmoln registreras. Efter den sista registreringen har ett punktmoln som föreställer bysten, utan lutning, framställdts.



Figur 11: Resultatet av registrering med 1, 5, 11, 17, 23, 29 respektive 35 punktmoln.

5.2.5 Meshning

Den ytrekonstruering som systemet använder, PCLs *Poisson surface reconstruction*, är väldigt beroende av att det registrerade punktmolnet inte har några punkter som har hamnat fel. Om några punkter är fel eller om det är för glest mellan punkterna så kan inte algoritmen räkna ut ytnormalerna korrekt. Om ytnormalerna blir fel kommer inte meshen att stämma överens med det objekt som punktmolnet föreställer.

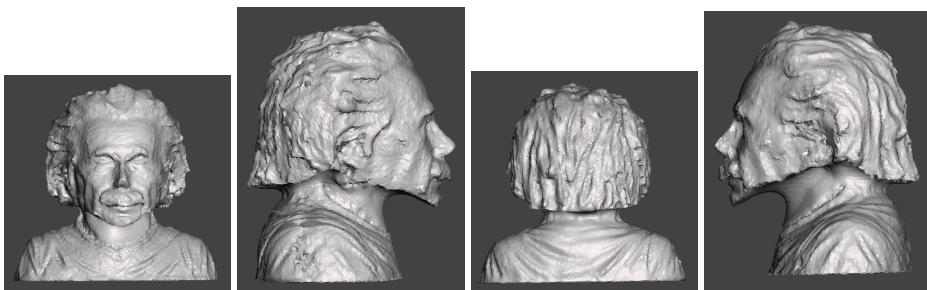
I figur 12 visas en mesh, som föreställer Einstein, gjord med 3DCopy. Det syns tydligt att olika problem uppstår. Dels blir ytorna på den genererade meshen grynpiga och liknar därfor

inte det skannade objektet till 100 %. Det sista och största problemet med meshen är att det uppstår öönskad massa, vilket syns tydligt på Einsteins undre del av hakan.



Figur 12: Mesh utan handpåläggning, objektet föreställer Einstein.

I figur 13 visas resultatet efter manuell hantering av den genererade meshen i figur 12. Den öönskade massan som fanns under Einsteins haka har plockats bort. Det syns även att de grynniga ytorna har slätts till, vilket i sig leder till att man går miste om detaljer i objektet.



Figur 13: Mesh med handpåläggning, objektet föreställer Einstein.

5.3 Gemensamma erfarenheter

I detta avsnitt presenteras de erfarenheter som gruppen har samlat på sig under projektets gång.

5.3.1 Övergripande projekterfarenheter

Att göra efterforskningar innan man börjar med någonting är väldigt viktigt och det märktes direkt i projektets början. Både genom att större delen av efterforskningarna kom till stor hjälp direkt i projektet och bara några dagar in i första iterationen märktes det att det fanns några områden som behövdes undersökas mer innan kod kunde implementeras. Ett exempel på bra efterforskning som gjordes i förstudien var efterforskningarna kring ROS där alla gruppmedlemmar gick igenom handledningsexempel och läste dokumentation som utvecklingsledaren gått igenom och tagit fram. I slutet av förstudien gjordes en gemensam kodutmaning där alla fick skriva varsin chattklient med hjälp av ROS. Kodutmaningen ledde till att alla kom in i ROS, Git, Python och andra verktyg som skulle användas under resten av projektet.

5.3.2 Erfarenheter gällande kommunikation

Betydelsen av kommunikation var stor under projektets gång. Verktygen *Slack* och *Trello* har använts flitigt och varit givande för gruppen. Med *Slack* har gruppmedlemmarna alltid kunnat nå varandra snabbt och smidigt. Informationsflöden har kunnat hållits skilda och

sorterade i olika kanaler. Funktioner som påminnelser och trådar har också varit till stor hjälp för att lätt kunna komma åt den informationen som man vill åt i flödena.

För att ha en översikt i hur det går med olika delar av projektet har gruppen använt *Trello*. Genom att ha olika listor för statusen av dokument och funktioner under utveckling gjorde att varje gruppmedlem lätt kunnat se hur det går och vad som behöver göras.

5.3.3 Erfarenheter gällande kvalitet

Kvalitet har varit viktigt för gruppen och kvalitetssamordnare har gjort ett bra jobb med att se till att projektets kod och dokument håller en hög standard. Detta framförallt genom granskning av dokument och kod. All kod i projektet har genomgått granskning för att säkerställa god kvalitet. Till detta har Githubs pull request-funktion använts där alla gruppmedlemmar kunnat kommentera och diskutera koden innan den går in i master branchen på repositoriet. Dokumenten har också granskats, då genom korrekturläsning. All denna granskning har varit mycket bra för att se hur andra gruppmedlemmar skriver dokument och kod.

5.3.4 Erfarenheter av att bygga vidare på ett projekt

I mitten av projektet upptäckte gruppen att det system som skulle vidareutvecklas inte fungerade tillräckligt stabilt för att det skulle vara möjligt att integrera med det system som gruppen utvecklade. Gruppen hade inte heller tillgång till någon slags testdokumentation från det tidigare projektet. Istället för att lita blint på det tidigare systemet borde gruppen i ett tidigt skede ha testat systemet utförligt. Då hade dessa fel kunnat hittas tidigare och en bättre plan för att arbeta runt dem kunde ha tagits fram. Detta hade lett till att alla parter blivit nöjdare med projektets resultat.

5.4 Översikt över individuella bidrag

Här listas de individuella bidrag som gruppens medlemmar har bidragit med till rapporten:

- Dunström, Hampus - Hur påverkas ett team av sin arbetsmiljö?
- Holmberg, Olof - Kontexters påverkan vid testning av GUI
- Jannering, Gustav - Hur kravhanteringsmetoder påverkar ett utvecklingsprojekt
- Karlsson, Michael - Analys av punktmolnsregistrering
- Lundberg, Martin - Att bygga ett system i ROS
- Tuhkala, Hannes - Verktyg som är lämpliga för att skriva stora dokument
- Wallström, Fredrik - Kvalitetsarbete i praktiken



6 Diskussion

6.1 Resultat

I detta avsnitt diskuteras de resultat som projektet kommit fram till.

6.1.1 Lärdomar

Att förbereda sig inför projekt med ordentliga förundersökningar och träningsuppgifter är definitivt något att ta med sig till kommande projekt. Det märktes i skillnaderna på hur lätt projektgruppen hade det med ROS och hur svårt gruppen hade det med PCL. Där alla gruppmedlemmar hade genomgått en grundlig utbildning i ROS men bara några få hade undersökt PCL men då inte särskilt noggrant. I framtiden kommer gruppen undersöka allt mer noggrant och vid tidsbrist dela upp undersökningarna mer så att allt undersöks grundligt av någon i gruppen istället för att alla undersöker ett område.

När projektgruppen i framtiden stöter på ett projekt som bygger vidare på existerande mjukvara eller hårdvara kommer ingen av oss glömma att testa och dokumentera testerna på det existerande systemet innan någon påbyggnad eller vidareutveckling kommer ske.

Vikten av att kunna uppskatta tiden olika aktiviteter tar är också något som gruppen tar med sig från projektet. Framförallt höll inte planeringen av undersökningarna inom PCL. Det var till stor del på grund av att PCL krävde mycket teoretiska förkunskaper för att kunna förstå hur de funktioner som PCL tillhandahöll ska användas. Det stora kravet på förkunskaper var inte uppenbart vid planeringen av undersökningarna och resterande aktiviteter fick hela tiden planeras om med hänsyn till dessa undersökningar. Därför kommer gruppmedlemmarna i framtida projekt att ta hänsyn till om de nödvändiga förkunskaper finns för att kunna genomföra projektet på det sätt som är planerat.

De rollspecifika dokument som skrevs i förstudien av projektet var bra då varje gruppmedlems erfarenhet inom deras specifika roll var varierande. Vissa gruppmedlemmar hade bra erfarenhet av rollen och visste hur de skulle gå tillväga. Andra gruppmedlemmar hade roller där de saknade eller hade väldigt lite erfarenhet inom de rollspecifika uppgifter som skulle genomföras. De rollspecifika dokumenten som skulle skrivas tvingade gruppmedlemmarna att sätta sig in i sin roll och snabbt få en uppfattning om vad som förväntades av dem. Det som gruppen kan ta med sig av denna erfarenhet är att tidigt i projekt

tilldela uppgifter som tillhör den roll som projektmedlemmarna har för att de snabbt ska komma in i deras roll och få bättre insikt i vad som förväntas av dem inom projektet.

Erfarenheterna inom *Slack* kommer bli användbara i det framtida arbetslivet och kommande projekt då det är ett vanligt förekommande verktyg inom kommunikation. Även *Trello* är ett vanligt förekommande verktyg även om det finns en del annan mjukvara med liknande funktionalitet. Grundprincipen med att dela upp arbetet i aktiviteter och uppdatera statusen på dem är definitivt något gruppen kommer ta med oss. Vi lärde oss också att försöka göra aktiviteterna så små som möjligt och involvera hela gruppen i processen att skapa aktiviteter. Det resulterade i en bättre användning av *Trello* samtidigt som det blev lättare att komma igång med arbetet.

Under projektets gång har projektgruppen använt oss av kodgranskning, även om några i gruppen var lite skeptiska till att någon måste läsa deras kod och godkänna den innan den togs in i den slutgiltiga kodbasen. Men genom att läsa varandras kod har gruppmedlemmarna lärt sig nya saker och förbättrat koden som gruppmedlemmarna skriver. I programering blir man aldrig fullärd och vetskapen om att läsa andras kod kan vara väldigt lärorikt är definitivt något vi tar med oss. Det kommer vara till stor hjälp i vår utveckling som utvecklare.

6.1.2 Vad återstår för att uppnå fullt värde för kunden?

Det är en definitionsfråga. Om man definierar *fullt värde för kunden* som det värde som kunden förväntade sig innan projektet eller i projektets tidiga skede krävs minst ett nytt projekt, vars mål är att lösa problemen med TreeD och sedan rekonstruera produkten från detta projekt så att den fungerar som den skulle innan omförhandlingen av kraven. Om man istället definierar "fullt värde för kunden" som det värde som kunden förväntar sig efter omförhandlingen av kraven skulle gruppen hävda att väldigt lite återstår. Meshgenereringen blev inte så automatiserad som gruppen hade hoppats och kunden förväntat sig, men detta beror mest på svårigheter att implementera en sådan process helt automatiskt då flera olika fall kan uppstå. Målet med projektet var att använda existerande algoritmer och inte skapa nya eftersom det hade tagit mycket längre tid och krävt helt andra kunskaper. Utöver detta tycker projektgruppen att kunden har fått ut det värde som kan förväntas av projektet efter omförhandlingen av kraven.

6.1.3 Tidigare projekt

Något som gruppen har tagit med sig från de tidigare projekt gruppmedlemmarna har genomfört är tydlig struktur på interna dokument samt gruppkontrakt. Strukturen på dokumenten har hjälpt till att hålla en hög standard även på interna dokument, det vill säga dokument som enbart är till för gruppen själva. Gruppkontraktet var uppskattat i tidigare projekt och ger tydliga riktlinjer för bland annat vad som förväntas av gruppens medlemmar, hur konflikter ska lösas och hur gruppen ska arbeta för att nå projektmålet.

Något som förbättrades från tidigare projekt är kommunikationen. Under vissa projekt som gruppens medlemmar tidigare genomfört har kommunikationen ibland varit bristande. Det har ibland varit svårt att få tag i medlemmar eller svårt att få en överblick av kommunikationen då många olika kommunikationskanaler används. Projektgruppen löste det snabbt genom att använda *Slack* för all kommunikation inom gruppen och delade upp kommunikationen i olika kanaler beroende på ämne. Gruppen såg också till att allas telefonnummer fanns tillgängliga ifall man behövde kontakta någon omedelbart.

6.1.4 Alternativa implementationssätt

Då projektets krav behövde omförhandlas långt in i projektet togs två olika system fram. Det första systemet som togs fram var byggt på ROS och hade en tät koppling till det tidigare

systemet även om det skulle kunna användas fristående med annan hårdvara. Det andra systemet var inte lika beroende på hårdvaran som det första utan läser istället in punktmoln från filer på datorn. Det är inte heller byggt på ROS utan är implementerat helt med hjälp av klasser i C++. Båda de implementerade systemen uppfyller i princip samma systemanatomi, se figur 3, med den enda skillnaden att det andra systemet inte kan skanna 3D-objekt utan endast läsa in punktmoln från filer.

För att registrera punktmoln finns det många olika algoritmer och sätt att implementera dessa. I slutändan använde gruppen ICP då den gav bäst resultat och har bra stöd i PCL. Det finns dock flera alternativa algoritmer för registrering. Detta utforskas mer i Michael Karlssons individuella utredning, se kapitel D.

Även för meshgenerering finns fler alternativa metoder. I vårt system används algoritmen *Poisson surface reconstruction* då det är en vanlig algoritm som fungerar bra i många fall. Även den har bra stöd i PCL. På den här punkten skulle en hel del kunna förbättras, antingen genom att använda andra algoritmer eller genom att implementera användandet av algoritmen bättre, exempelvis genom att justera de parametrar som sätts.

6.2 Metod

Som tydligt kan ses i denna rapport har inte projektet gått som det var planerat. Projektgruppen har under projektets gång (speciellt under projektets andra halva) varit kritiska över hur vi gått tillväga. I avsnitt 5.3 listas de erfarenheter som har dokumenterats under projektets gång. Bland dessa finns erfarenheter om vad som borde ha gjorts annorlunda. Gruppen borde ha testat det befintliga systemet, TreeD, mer rigoröst tidigt istället för att lita på att det fungerade felfritt.

När det gäller metoden som har använts i projektet anser projektgruppen att själva kärnan, eller idén bakom metoden har varit bra men att några av detaljerna av implementeringen av denna metod har varit mindre bra. Som grupp håller alla med om att gruppen borde fokuserat mindre på ROS i början av projektet. I efterhand tycker gruppen att tanken var bra, att alla gjorde handledningsexempel och en kodutmaning tillsammans stärkte gruppens sammanhållning och skapade en bra anda. Gruppen borde istället fokuserat på till exempel PCL eller registrering av punktmoln eftersom systemet, både innan och efter omförhandlingen, inte var så beroende av ROS som vi trodde att det skulle vara.

Efter förstudiefasen och iteration 1 började fler och fler problem uppstå med projektet. De flesta av dessa problem hade sin grund i TreeDs system. Detta gjorde att gruppen mindre och mindre höll fast vid den metod som använts tidigare och sammanhållning blev sämre.

Något som förändrades genom projektets gång var längden på aktiviteterna, det vill säga deras omfattning mätt i tid. De stora undersöknings- och efterforskningsaktiviteterna som gruppen skapade i början var svåra att överblicka och få en uppfattning om hur långt arbetet hade kommit. Det förändrades då gruppen valde att skapa mindre aktiviteter när problemet med de större upptäcktes. Hade gruppen från början gjort mindre och mer lättöverskådliga aktiviteter hade eventuellt undersökningarna i början gått bättre och inte påverkat tidsplanen så negativt som de gjorde.

6.3 Arbetet i vidare sammanhang

Systemet är tänkt att användas i akademiska syften och därfor ses inga etiska aspekterrelaterade till projektet. Nedan beskrivs kopplingen mellan projektet och de samhälleliga aspekterna samt vilka specifika förbättringspunkter det finns till vårat system med avseende på hållbar utveckling.

6.3.1 Hållbar utveckling

Ett system för 3D-kopiering har många positiva effekter på samhället. Systemet gynnar framförallt miljön eftersom att tillverka en önskad produkt istället för att köpa en fabrikstillverkad motsvarighet sparar avsevärt på naturens tillgångar. Dels kommer 3D-kopieringssystemet att använda mindre energi och dessutom kommer transporterna till och från affären att minska, vilket leder till minskade koldioxidutsläpp. Kreiger och Pearce [25] har gjort en studie där de skriver ut 3D-produkter i plast och jämför kostnaden för att tillverka produkter av plast i en fabrik. Med kostnaden menas den energi som går åt från råmaterial till färdig produkt samt kostnaden som går åt för transport. Det visar sig att tillverka en produkt i en 3D-skrivare kräver mellan 41-64 % mindre energi än att fabrikstillverka produkten. Förklaringen till detta är att produkter som skrivs ut i en 3D-skrivare kan göras mer ihåliga och således kräver de också mindre material.

Det finns dessutom relaterade studier som visar att det blir billigare att skriva ut en produkt i en 3D-skrivare istället för att köpa en fabrikstillverkad [55]. Det här främjar samhället i positiv beaktning eftersom det helt enkelt blir billigare för konsumenter att införskaffa sig de produkter de önskar.

6.3.2 Specifika förbättringspunkter till vårt system

Vårt system för 3D-kopiering är som tidigare nämnts ett generellt bra system för att främja hållbar utveckling. Det finns dock en del aspekter som skulle kunna gjorts annorlunda vid systemets uppbyggnad för att ytterligare främja hållbar utveckling. För att utveckla detta har gruppen valt att titta på hur våra krav framställdes genom att besvara dessa frågor:

- Vad skulle vi kunna göra annorlunda i kravprocessen?
- Vad har vi tagit hänsyn till i kravprocessen?
- Hur kan vi bedöma de krav vi satt på systemet med hållbar utveckling i åtanke?

Vid framställning av kraven till systemet fanns det inga tankar på att ta fram krav som främjar hållbar utveckling. Detta berodde på att ingen i projektgruppen hade erfarenheter från hållbar utveckling tidigare och således tänkte inte någon på det. Vid framtagning av krav till systemet har alltså ingen hänsyn tagits till att främja hållbar utveckling. Det som har tagits hänsyn till i kravprocessen är endast funktionaliteten av systemet. Att utveckla icke-funktionella krav för att främja hållbar utveckling är något som gruppen kunde ha gjort annorlunda. Det är också ett bra tillvägagångssätt enligt Raturi et al. [44].

En förbättring som gruppen kunde ha gjort är att systemet kunde ha haft ett icke-funktionellt krav att systemets beräkningar får ta en viss maximal tid. Detta är bra för energi- och miljösynpunkt eftersom hög processoranvändning under en lång tid leder till hög energiförbrukning för systemet.

Det finns även en relation mellan hur systemets energianvändning påverkas av hårdvaran i systemet. Den största energianvändande komponenten är processorn och därför skulle systemet även kunna ha ett icke-funktionellt krav att optimera processoranvändningen. Detta leder till att energianvändningen för systemet minskar vilket också är bra ur miljösynpunkt. Enligt Fan et al. [13] ökar energiförbrukningen linjärt med processoranvändningen vilket inte är helt rättvisande och givetvis beror det på vilken processor som används.

För 3DCopys system innebär det här att gruppen skulle vilja kombinera de två aspekterna köringstid och processoranvändning till ett gemensamt icke-funktionellt krav. En kombination av dessa innebär att vi som grupp skulle behövt göra en studie över hur vi kan optimera processoranvändningen med avseende på energiförbrukning samtidigt som systemet inte får alltför lång körtid. Det optimala ur energisparsningssynpunkt kanske skulle vara att låta systemet använda 70 % av processorn och istället få lite längre körtid. Detta är någonting som gruppen skulle kunna undersökt i förstudiefasen för senare användning till kravprocessen.

För att bedöma de krav som vi har med hållbar utveckling i åtanke kommer endast de krav som inte berör kärfunktionaliteten att bedömas, eftersom de är nödvändiga för systemets funktionalitet. Andra krav bedöms ur energisynpunkt, det vill säga om koden är tillräckligt optimerad för att uppfylla kravet eller om det blir för tungt beräkningsmässigt som gör att kravet inte uppfylls. Det skulle även vara möjligt att sätta upp icke-funktionella krav med avseende på det sociala området som gör att användaren känner sig tillfredsställd med produkten och på så vis bidrar till samhället. Detta bedöms genom att testa systemet med några användare och se huruvida det uppfyller användarens förväntningar eller inte.



7

Slutsatser

Det inledande målet med projektet var att konstruera en mjukvara för att styra ett system som kopierar tredimensionella objekt. Målet reviderades efter halva utvecklingstiden eftersom det system som vi skulle bygga vidare på inte höll de förväntade kraven. Det nya målet med projektet blev istället att utveckla ett system som kan hantera, det vill säga registrera och mesha, punktmoln och på det viset försumma det tidigare projektets mjukvara. Målet med projektet nåddes, ett system för hantering av punktmoln har skapats. Det finns dock förbättringsmöjligheter med systemet. Att registrera punktmolnen så de bildar ett komplett punktmoln visade sig vara svårare än vad vi trodde men vi har lyckats registrera ett visst antal punktmoln som gör systemet användbart. Det stora hindret med registreringen är en lång körtid, detta försvårade utvecklingsarbetet samt testningen.

Det finns även en del förbättringsmöjligheter angående meshningen. Den genererade meshen ifrån systemet är inte användbar för att skriva ut med hjälp av en 3D-skrivare. Meshen består av oönskad massa som inte är en del av det verkliga objekten. För att ta bort denna oönskade massa måste meshen hanteras manuellt. Detta problem är något som skulle kunna automatiseras så att det önskade objekten genereras direkt. Avsnitten nedan besvarar forskningsfrågorna som ställdes i avsnitt 1.3.

7.1 Värde för kunden

För att skapa värde för kunden med produkten är det framförallt två delar som är viktiga. För det första, att tidigt sammanställa de krav som kunden har på produkten så att arbetet tidigt går i rätt riktning. För det andra, att kontinuerligt följa upp dessa krav under projektets gång för att säkerställa att de inte har ändrats. Om dessa två steg följs kommer kunden att i slutändan få en produkt som uppfyller dennes krav.

Något som också tagits hänsyn till i utvecklingen av systemet är dess ursprungliga syfte, att vara del av ett större system. Därför har det fokuserats på att göra 3DCopy modulärt för att det ska vara enkelt för kunden att anlita en annan grupp med liknande programmeringsvana som kan integrera 3DCopy i ett större system, vidareutveckla 3DCopy eller använda 3DCopy för att forska kring registrering och meshgenerering.

7.2 Erfarenheter

Att göra efterforskningar i ett tidigt skede är en avgörande del i ett projekt. Under projektets gång har vi insett att man behöver prova nya områden inte bara i teorin utan även med praktiska övningar. Vi har också insett att man inte kan undersöka en enda sak allt för mycket utan bör sprida ut sin tid och undersöka flera områden. Detta gäller lika mycket för ramverk, språk och teori som för de system man bygger vidare på eller som ligger till grund för det man ska utveckla. Skenet kan bedra och man bör alltid ha god dokumentation för sina antaganden.

Det är viktigt att ha strikta kommunikationskanaler och att utnyttja de tjänster och program som finns är till stor hjälp under ett projekt. Kunskaper i *Slack* kommer att komma till bra användning i våra fortsatta arbetsliv, precis som att arbeta med aktiviteter och bryta ner problem i så små bitar som möjligt.

Granskning av kod och dokument är också något som har spelat stor roll i detta projekt och kan appliceras på framtida projekt. Granskningsprocessen är ett utmärkt verktyg då den bidrar med framförallt två viktiga aspekter. För det första kontrolleras allt gruppen producerar och kvaliteten på det producerade håller därför hög standard. För det andra hjälper den till att sprida kunskap mellan gruppens medlemmar då man tar del av det andra gruppmedlemmar har gjort och lär sig av varandra.

Dessa erfarenheter är av intresse för framtida projekt då 3DCopys projektgrupp har fått stor hjälp av väl genomförda efterforskningar, tydlig kommunikation och noggranna granskningar av dokument. För framtida projekt innebär det en ökad effektivitet om de tar del av och applicerar dessa erfarenheter.

7.3 Systemanatomi

För att svara på fråga 3 i avsnitt 1.3 så var projektets systemanatomi användbar vid projektets början. Systemanatomin gav ett bra stöd under projektets gång eftersom alla i projektgruppen fick en enkel helhetsbild över vilken funktionalitet systemet skulle innehålla. Projektgruppen skulle bygga ett stort och komplext system. här var systemanatomin till stor hjälp för projektgruppen då den gav en bättre förståelse för produkten och dess olika funktioner. Systemanatomin gav oss alltså en gemensam förståelse över produkten samt över funktionerna och dess beroenden vilket ledde till att alla i projektmedlemmar var på samma nivå kunskapsmässigt vid projektets start.

Systemanatomin var även till stor hjälp då arkitekturen togs fram då den tydliggjorde vilka funktioner som systemet skulle innehålla och hur närliggande funktionalitet kunde delas in i relevanta moduler. På så sätt hjälpte systemanatomin till att göra systemet så modulärt som möjligt vilket i slutändan ledde till en högre kvalitet på produkten.

7.4 Reviderad kravspecifikation

Efter omförhandling med kunden och samtal med handledaren och examinatorn ändrades målet med projektet tillsammans med kraven. Under omförhandlingen med kunden kom vi överens om att det nya målet med projektet var likt det gamla fast vi kopplade bort TreeD. Tillsammans med det skrevs kraven som direkt eller indirekt byggde på TreeDs mjukvara om. Som tur var så hade vi kod som kunde återanvändas efter denna pivotering av projektet. Det är denna kod som det nya systemet bygger på. Det som förvärrade situationen var att omförhandlingen kom vid helt fel tidpunkt eftersom planeringen av den andra halvan av projektet inkluderade mycket mer dokumentskrivande än de tidigare iterationerna.

Utöver detta så påverkades gruppen av de problem som uppstod innan omförhandlingen. Precis innan omförhandlingen så var stämningen i gruppen relativt dålig. Det kändes

som att vi tog ett steg framåt och två bakåt nästan varje dag. Gruppens stämning och sammanhållning blev gradvis bättre efter omförhandlingen.

Projektet och projektgruppen påverkades negativt av den revision som gjordes. Att revidera kravspecifikationen så kraftigt som gjordes i detta projekt lägger en stor börd på utvecklingsgruppen. Efter omförhandlingen gick mycket tid åt till att skapa en ny förståelse för hur systemet skulle utformas och för att uppdatera alla dokument så att de reflekterade projektets nya mål. Tidsbristen som resulterade av detta gjorde att gruppen inte hade så mycket tid som vi hade önskat för att utveckla det nya systemet.

7.5 Vidareutveckling av ett system

För att svara på fråga 5 i avsnitt 1.3 anpassades tillvägagångssättet för arbetet markant. Då vi skulle utveckla ett system som skulle vara tätt integrerat med det tidigare systemet som använde ROS lade vi mycket fokus i början på att lära oss det. Hela arkitekturen för systemet togs fram med ROS i åtanke. Det underliggande systemet innebar alltså att vi kunde utveckla saker som vi inte hade kunnat göra utan TreeD eftersom det låg mycket arbete bakom det.

Däremot var det inte helt problemfritt att vidareutveckla ett system. Det har lagts ner mycket tid och energi för att utforska och lära oss systemet. Detta var någonting som vi trodde vi skulle undvika eftersom vi byggde arkitekturen med tanken att det gamla systemet skulle fungera som en separat modul. Det visade sig dock att det gamla systemet inte höll de förväntade kraven vilket då betydde att gruppen behövde utforska och lära sig det för att lösa de problem som uppstod vilket inte låg under projektets ramar.



A

Hur påverkas ett team av sin arbetsmiljö?

— *Hampus Dunström* —

A.1 Inledning

Examen kommer bara närmare och närmare och uppvakningen av olika företag är stor. Företagen erbjuder många olika miljöer att arbeta i. Vilken ska studenterna välja, var arbetar och mår de bäst? I den här rapporten undersöks arbetsmiljön för studenterna i kursen *Kandidatprojekt i programvaruutveckling* för att ta reda på det.

A.1.1 Syfte

Ta reda på hur arbetsmiljön för studenterna i kursen *Kandidatprojekt i programvaruutveckling* ser ut och hur de påverkar dem. Detta för att skapa förståelse i hur arbetsmiljön påverkar ett team som arbetar med mjukvaruutveckling. Då detta kan underlätta framtida val inom arbetsmiljö, både för studenterna själva och arbetsgivarna som vill anställa dem.

A.1.2 Frågeställningar

1. Hur ser arbetsmiljön ut för grupperna i kursen *Kandidatprojekt i programvaruutveckling*?
2. Hur påverkar arbetsmiljön sammanhållningen i gruppen?
3. Hur påverkas gruppen av att få tillgång till ett eget rum där kandidatarbetet kan utföras jämfört med grupper som inte fått det?

A.1.3 Avgränsningar

Denna rapport grundar sig på en undersökning om arbetsmiljön under arbetet i kursen *Kandidatprojekt i programvaruutveckling*. Rapporten blir därför begränsad av arbetet, de arbetssätt och arbetsmiljöer studenterna som deltagit i kursen utfört, använt och arbetat i.

A.1.4 Definitioner, akronym och förkortningar

Följande definitioner och förkortningar används på flera ställen i denna del av rapporten:

- Småföretag - Företag med 50 eller färre anställda.

A. HUR PÅVERKAS ETT TEAM AV SIN ARBETSMILJÖ?

- Stora företag - Företag med fler än 50 anställda.
- Allmänbelysning - Medelbelysningsstyrkan mätt i horisontalplanet 85 cm över golvet.
- lux - SI-enhet för illuminans, också kallad belysningsstyrka.
- Creative - Ett öppet kontorslandskap i närheten av universitetet som studenter har tillgång till.
- Slack - Kommunikationsverktyg för strukturerad kommunikation i olika chattar.

A.2 Bakgrund

Under studietiden har det genomförts en del projekt och grupperbeten. Där alla hade varierande resultat. Alla dessa projekt har haft olika förutsättningar i form av arbetsmiljö. Vissa har utförts på olika platser varje gång gruppen träffats, andra på samma plats varje gång. I vissa fall kan privata föremål lämnas på arbetsplatsen, ibland inte. Vissa grupperbeten har skett i högljudda miljöer, andra i tysta. Vad har detta för betydelse för arbetets resultat och effektivitet? Denna undersökning görs i ett försök att ta reda på hur den optima arbetsmiljön ser ut.

A.3 Teori

Nedan följer information kring arbetsmiljö i småföretag och kontorsmiljö. Informationen kommer från Arbetsmiljöverkets hemsida och en rapport *Arbetsmiljö- och hälsoarbete i småföretag* publicerad av Arbetslivsinstitutet [21][50].

A.3.1 Belysning

Vid kontorsarbete och framförallt bildskärmsarbete är bra belysning viktigt. Arbetsmiljöverket rekommenderar att en arbetsplats är belyst med 500 lux och allmänbelysningen är 300 lux. När belysningen mäts upp är det mängden ljus som träffar ytan som mäts. Belysningen är inte det enda att tänka på när det kommer till belysning. Reflektioner, skuggor och kontraster kan också vara dåligt för arbetsmiljön. Arbetsmiljöverket tar upp arbetsplatsplacering i förhållande till fönster när bildskärmsarbete ska utföras vid arbetsplatsen. De föreslår att skärmarna placeras vinkelrätt mot fönstren för att minimera reflektioner [53][8].

Konsekvenserna av dålig belysning är koncentrationssvårigheter, huvudvärk och ansträngningsskador på ögonen. De som arbetar mer än en timme i veckan med bildskärmsarbete bör tillgodoses med regelbundna synundersökningar i intervall på två till fem år beroende på ålder. Där yngre mäniskor inte behöver det lika mycket som äldre. Arbetsgivaren ska dessutom införskaffa glasögon vid behov då detta är ett mycket viktigt arbetsredskap [53].

A.3.2 Ljud

Enligt arbetsmiljöverket är det viktigt med en tyst miljö när arbete som kräver koncentration ska utföras. Det är viktigt att avskärma störande ljudkällor såsom skrivare och ventilation. I kontorsmiljö är andra mäniskor en stor källa till störande ljud. Detta är inte på grund av ljudnivån utan på grund av att tal distraherar. Då mäniskor ofta automatiskt börjar lyssna på vad som sägs. Då är det viktigt med respekt eller avskilda rum. Där de som behöver diskutera något kan prata ostört och utan att störa [29].

A.3.3 Kontor

Det finns flera typer av kontor. Cellkontor, öppna kontor, kombikontor och flexkontor är några exempel. På ett cellkontor har alla arbetare ett eget rum eller bås. Det ger avskildhet, minimerar störande ljud och underlättar koncentration. Informationsflödena blir dock sämre och det kan bli svårt att överblicka lokalen.

Öppna kontor är kontor där personalen arbetar i stora gemensamma rum. Här är det lättare att arbeta i grupper och flexibiliteten är större. Större blir också risken för störande ljud och det kan vara svårt och koncentrera sig, speciellt för arbetstagare med vissa kognitiva svårigheter.

En kombination av cellkontor och öppna kontor kallas kombikontor. Här finns både öppna utrymmen och avskilda kontor.

I de tidigare kontorstyperna har arbetstagarna oftast sin egen plats. Det är inte fallet i flexkontor, även kallade aktivitetsbaserade kontor. Här har inte arbetstagarna en fast plats utan de väljer en plats anpassad för arbetet som ska utföras. Det är i dessa flexkontor eller cellkontor som medarbetarna mår bäst enligt en forskningsstudie, men det kan finnas fler orsaker än bara kontorstypen [35].

A.3.4 Skillnader mellan små och stora företag

En god arbetsmiljö är väldigt viktigt och något som lätt försummas, speciellt i mindre företag. Arbetsolyckor och arbetsskador är vanligare i småföretag än större men sjukfrånvaron är högre i större företag än i småföretagen. Detta beror bland annat på att småföretag är beroende av enskilda anställda i större utsträckning än stora företag [50].

A.4 Metod

För att besvara frågeställningarna gjordes en empirisk studie i form av en enkät utskickad till de andra deltagare i kursen *Kandidatprojekt i programvaruutveckling*. Enkäten skickades ut som ett Google Formulär [48] med frågorna:

- Beskriv kort projektet du arbetar med.*
- Hur intresserad av projektet är du?*
- Hur många gånger i veckan träffas hela gruppen?*
- Hur stor del av ditt arbete gör du ensam?
- Föredrar du att jobba ensam eller tillsammans med någon?*
- Vart brukar du utföra ditt arbete, hemma, i skolan eller...?*
- Hur är ljudvolymen på dessa platser?
- Hur är belysningen på dessa platser?, trivs du med den?
- Är det folk runt omkring dig där, isåfall vad gör dom?
- Har ni fått tillgång till någon lokal, isåfall hur ofta används den?*
- Hur tycker du sammanhållningen i ditt team är?*
- Varför tror du sammanhållningen är som den är?

A. HUR PÅVERKAS ETT TEAM AV SIN ARBETSMILJÖ?

Frågorna med ** på slutet var obligatoriska. Svaren på enkäten kan ses i bilaga H.1. Utifrån denna data och de personliga erfarenheterna som samlats under utförandet av Kandidatprojektet gjordes sedan en diskussion och slutsatser drogs. Dessa finns beskrivna under Diskussion A.6 och Slutsatser A.7.

För att kunna utforma och tolka denna enkät gjordes en litteraturstudie om arbetsmiljö. Där eftersöktes främst grundläggande information och forskning inom arbetsmiljö i kontorsmiljö. Eftersom det är där mjukvaruutveckling sker. För att finna denna information användes sökmotorn Google med sökorden "arbetsmiljö", "kontor", "dator", "små företag" i olika kombinationer. Den främsta källan som snabbt hittades var arbetsmiljöverkets hemsida. Där navigerades det i deras menyer för att finna all relevant information om arbetsmiljön på kontor. En rapport publicerad av Arbetslivsinstitutet hittades också och ansågs som en mycket trovärdig källa tack vare sin utgivare.

A.5 Resultat

Det var 27 av de 98 (27,5%) studenter som läste kursen *Kandidatprojekt i programvaruutveckling* som svarade på enkäten. Sammanfattade svar kan ses nedan:

Hur intresserad av projektet är du?

Genomsnittsintresset låg på 3,8 på en skala 1-5.

Hur många gånger i veckan träffas hela gruppen?

Grupperna träffades i genomsnitt 3,5 gånger per vecka.

Hur stor del av ditt arbete gör du ensam?

Större delen av arbetet gjordes i grupp, 16 studenter svarade 25-50 % 9 svarade mindre än 25% och 2 studenter svarade att de gjorde en större del än så ensamma.

Föredrar du att jobba ensam eller tillsammans med någon?

Det allra flesta studenterna föredrog att arbeta tillsammans med någon.

Vart brukar du utföra ditt arbete, hemma, i skolan eller...?

Skolan var den plats som de flesta gjorde större delen av sitt arbete.

Hur är ljudvolymen på dessa platser?

Större delen av de svarande tyckte ljudvolymen var låg eller bra.

Hur är belysningen på dessa platser?, trivs du med den?

Studenterna var nöjda med belysningen, flera uttryckte att de föredrog fönster med naturligt ljus och enstaka studenter uttryckte specifikt misstycke för lysrör.

Är det folk runt omkring dig där, isåfall vad gör dom?

Runt omkring arbetsplatsen svarade större delen att det var andra studenter som också arbetade med sina studier.

Har ni fått tillgång till någon lokal, isåfall hur ofta används den?

De flesta hade inte fått tillgång till någon specifik lokal men de som hade det använde den flitigt. Av de som inte fått någon lokal uttryckte flera att de bokade olika lokaler i skolan nästan varje dag.

Hur tycker du sammanhållningen i ditt team är?

Sammanhållningen var hög, genomsnittet på de som svarade var 4,2.

Varför tror du sammanhållningen är som den är?

Kick-off och att arbeta tillsammans tyckte de svarande studenterna var viktiga faktorer till bra sammanhållning. De som kunde försämra sammanhållningen var att studenterna upplevde att de andra i gruppen inte gjorde sin del eller att projektet i allmänhet inte gick särskilt bra.

För mer detaljerade resultat se bilaga H.1.

A.6 Diskussion

Diskussionen är uppdelade utifrån frågeställningarna A.1.2.

A.6.1 Metod

Att använda sig av en enkät var ett bra sätt att samla in en större mängd data. Det skulle inte ha varit rimligt att hålla ca 30 intervjuer. Däremot hade ett par kompletterande intervjuer kunnat vara givande i tolkningen av svaren. Enkäten skulle också ha kunnat förbättras. Det hade varit en fördel om enkäten hade med en fråga om vilken grupp studenten var i. Då skulle jag kunnat se bättre hur sammanhållningen var i de olika grupperna. En fråga om hur studenterna tyckte deras sammanhållning påverkades av arbetsmiljön hade också gett användbar information. Men jag tror att det var bra att hålla enkäten kort. En för lång enkät hade troligen resulterat i färre svar.

Enkäten skickades ut i ett tidigt skede. Flera andra enkäter skickades ut långt senare. Det hade nog blivit fler svar om en påminnelse hade skickats ut i samband med många av de andra enkaterna. Ett problem där var dock att jag ville kunna börja arbeta med min rapport tidigt. Det hade dock gått att börja på rapporten och sen göra ändringar ifall att påminnelsen gav nya svar med data som visade på nya saker eller motbevisade saker jag redan hade skrivit om.

A.6.2 Hur ser arbetsmiljön ut för teamen i kursen *Kandidatprojekt i programvaruutveckling*?

Majoriteten av arbetet i kursen gör studenterna i skolan. Där trivs studenterna även om lysrören kanske inte alltid ger optimalt arbetsljus och ibland kan lysrören låta irriterande. Detta minskar dock risken för reflektionen som Arbetsmiljöverket talar om när det kommer till fönster och skärmarbete. Ljudvolymen är de flesta också nöjda med även om det ibland på de öppna ytorna kan bli lite störande. För precis som Arbetsmiljöverket skriver så är samtal distraherande trots att ljudvolymen inte är störande hög. Lösningen för många är hörlurar. Att lyssna på musik stänger ute distraherande samtal och underlättar för koncentrationen. Däremot när studenter bokar egna rum är sällan ljudvolymen ett problem. När studenterna inte arbetar i skolan gör de det oftast hemma. Där svarade studenterna att de hade skrivbor-dslampor som gjorde ett bra jobb och de största problemen var störande grannar men det var inte särskilt vanligt.

Jag håller med svaren till enkäten. Det är en bra arbetsmiljö på universitetet. Det enda problemet är att det kan vara svårt och få tag på salar. Detta har dock studenter löst genom att ta sig till Creative, ett öppet kontorslandskap utanför campus. De har också valt att jobba hemifrån och kommunicera över Slack då detta passat bättre. Det arbetssättet liknar det Arbetsmiljöverket beskriver när de skriver om aktivitetsbaserade kontor. Vilket det finns forskningsstudier som visar att de som arbetar i denna typ av miljö mår bäst enligt Arbetsmiljöverket.

A.6.3 Hur påverkar arbetsmiljön sammanhållningen i teamet?

Det flesta studenterna kände att de hade en bra sammanhållning i sitt team. Bara en person tyckte att sammanhållningen var dålig (sämre än 3 på en skala 1-5) i sitt team. Detta gjorde att det inte gick att hitta något samband mellan arbetsmiljön och sammanhållningen i teamet. Istället såg jag att studenternas intresse för projekten också var hög. Detta fick mig att fundera på huruvida arbetsmiljön kanske inte riktigt hunnit sjunka in. I korta projekt är arbetsmiljön kanske mindre viktig än i längre projekt. För om projektdeltagarna får göra något de tycker är intressant och roligt hålls moralen i projektgruppen ändå hög. När projekten blir längre eller mindre intressanta kan arbetsmiljön eventuellt bli mer påtaglig. Detta kan liknas med arbetsmiljön i små företag. Där arbetas det ofta med saker personalen och framförallt företagaren är passionerad i. Vilket kan leda till att arbetsmiljön överses. Detta är riskabelt enligt Carl-Göran Ohlson och Peter Westerholm som skriver om hur små företag är beroende av företagaren och dennes anställda [50]. På samma sätt är kortare projekt i små grupper ofta beroende av varje gruppmedlem och skador i arbetet eller sjukskrivning kan få drastiska konsekvenser.

För att få en bättre inblick i hur arbetsmiljön är borde ytterligare en undersökning göras. I den undersökningen bör några grupper ges en bra arbetsmiljö, några en sämre och en kontrollgrupp med en vanlig arbetsmiljö. Efter de projekten genomförts skulle en liknande enkät som denna rapport är grundad på kunna ge bättre resultat.

Det kan ha funnits ett mörkertal med studenter som kände att sammanhållningen i deras grupper varit sämre. Dessa studenter kan ha avstått från att svara på enkäten då deras motivation och känsla av deltagande i kursen inte varit särskilt stor.

A.6.4 Hur påverkas gruppen av att få tillgång till ett eget rum där kandidatarbetet kan utföras jämfört med grupper som inte fått det?

En av de första tydliga skillnaderna mellan olika grupper är att vissa fick tillgång till egna rum på universitetet. Rum som endast den gruppen använde. Detta resulterade i att dessa grupper utförde mycket av sitt arbete i öppna kontorsutrymmen. Andra grupper som inte blev tilldelade rum att arbeta i har arbetat i något som mer kan liknas vid aktivitetsbaserade kontor som Arbetsmiljöverket beskriver på sin hemsida. Då de bokat salar i skolan, arbetat på Creative eller hemma. Creative kan ses som ett stort öppet kontorslandskap med arbetsplatser och ytor avsedda för umgänge och samtal. Vår projektgrupp fick ett rum på universitetet då vårt projekt till en början var knuten till en viss hårdvara som bara fanns där. Det var en stor fördel i början av projektet när mycket samarbete behövdes. Lite längre fram i projektet uppstod en del störningsmoment på grund av att vi var alldelens för många på en för liten yta. Att inte få ett rum kan alltså ha varit en fördel i vissa situationer. Arbetsmiljöverket skriver till exempel om att det finns forskning som visar att aktivitetsbaserade kontor är bättre för medarbetarnas välmående än öppna kontorslösningar.

Men detta kan också bero på helt andra saker. Vårt projekt fick en hel del problem vilket gjorde att moralen i gruppen blev sämre. Det fanns till exempel en annan grupp som fick ett liknande rum vars projekt inte stötte på sådana problem och där har moralen varit högre och sammanhållningen bättre. Det är precis som Arbetsmiljöverket även skriver att välmåendet inte bara beröver bero på utformningen av kontoret utan kan bero på många faktorer. Min undersökning visar att över lag är sammanhållningen hög precis som intresset för de olika projekten över lag är hög. Detta i kombination med teambuilding som lyfts fram i undersökningen kan vara några av de andra faktorer som påverkar sammanhållningen. Det var dessutom bara 27,5% av studenterna som läser kursen som svarade på enkäten. Därför kan undersökningen ha missat en hel del.

A.7 Slutsatser

Mina slutsatser utifrån frågeställningarna A.1.2.

A.7.1 Hur ser arbetsmiljön ut för teamen i kursen *Kandidatprojekt i programvaruutveckling?*

Arbetsmiljön för studenterna liknar den i aktivitetsbaserade kontorsmiljöer där de arbetande väljer arbetsplats utifrån de arbete som ska utföras. Trots små störningsmoment som dåliga lysrör och samtalande studenter runt om kring har studenterna löst dessa problem. Resultatet stödjer den forskning Arbetsmiljöverket tar upp på sin hemsida om att välmåendet är högt i aktivitetsbaserade kontor.

A.7.2 Hur påverkar arbetsmiljön sammanhållningen i teamet?

Det gick inte att dra en direkt relation mellan arbetsmiljön och sammanhållningen i teamen. Det var alldelvis för många faktorer och mycket mer efterforskningar krävs. Att det dessutom bara var 27,5% av de som läste kursen som svarade på enkäten gjorde att undersökningen skulle behövt mer data för att kunna dra mer konkreta slutsatser. Däremot kan det konstateras att arbetsmiljö är viktigt i små projekt precis som stora.

A.7.3 Hur påverkas gruppen av att få tillgång till ett eget rum där kandidatarbetet kan utföras jämfört med grupper som inte fått det?

Grupperna som blivit tilldelade rum har enligt undersökningen använt dem flitigt och varierat sin arbetsmiljö mindre än de som inte fått något rum. De som inte fått tillgång till något rum har varit mer flexibla och valde att boka salar, jobba hemifrån eller i andra lokaler beroende på vad som passat bäst.

De grupper som flitigt använt sina rum har dock haft det lättare med planering och kommunikation. Eftersom de flesta var på plats i rummet under arbetstid kunde man i dessa grupper lätt få till ett samtal öga mot öga. I början av arbetet var detta till stor hjälp. Vilket gjorde att grupper med rum kom igång med sitt arbete fortare än andra grupper.



B

Kontexters påverkan vid testning av GUI

— Olof Holmberg —

B.1 Inledning

När människor interagerar med mjukvara gör de det oftast genom ett grafiskt användargränssnitt. Då byggs en sekvens av interaktioner upp som bara blir längre och längre. Eftersom varje interaktion påverkar mjukvaran kan det uppstå fel som beror på hur denna sekvens är uppbyggd. Dessa fel behöver inte uppstå förrän sekvensen består av flera hundra olika interaktioner.

Kontexten är det som medför problem vid GUI-testning. Det är nästintill omöjligt att testa alla olika kombinationer av interaktioner som användarna kommer att utsätta mjukvaran för. Därför är det viktigt att vid test av GUI:t ta fram testfall som maximerar antalet testade sekvenser utan att testningen tar för lång tid. Den här utredningen undersöker hur sekvenser som är mest kritiska att testa kan tas fram samt hur det tillämpades på projektet.

B.1.1 Syfte

Syftet är att undersöka hur viktig kontexten för interaktioner är vid testning av GUI. Något som också undersöks är hur dessa testfall kan utformas för att inte få en testning som är alldelvis för tids- och resurskrävande. Syftet är också att undersöka resultatet av att tillämpa testning som tar hänsyn till kontexten jämfört med testning som inte tar någon hänsyn till kontexter.

B.1.2 Frågeställning

1. Hur viktig är kontexten för interaktioner vid testning av GUI?
2. Hur bör testfall utformas för att ta hänsyn till kontexten?
3. Hade hänsyn till kontexter någon påverkan vid testning av 3DCopys GUI?

B.1.3 Avgränsningar

Denna rapport fokuserar enbart på att undersöka testning av grafiska användargränssnitt där hänsyn har tagits till kontext. Denna begränsning tillämpas främst på de artiklar som

har använts i denna rapport. Själva undersökningen är begränsad till en litteraturstudie samt resultatet av testningen som genomfördes på gruppens GUI.

B.1.4 Definitioner

Följande definitioner används på flera ställen i denna del av rapporten:

- GUI (Graphical user interface) - Ett grafiskt användargränssnitt för ett program eller system.
- Interaktion - När användaren av GUI:t interagerar med GUI:t genom att till exempel trycka på en knapp.
- Kontext - Kontexten för en interaktion är sekvensen av interaktioner som har utförts på GUI:t innan den aktuella interaktionen.
- Systeminteraktion - En interaktion som inte enbart påverkar och använder GUI kod.
- EFG (Event-flow-graph) - En graf som visar alla möjliga interaktioner som kan utföras på GUI:t.
- EIG (Event-interaktion-graph) - En EFG där alla interaktioner som enbart påverkar GUI:t är borttagna.

B.2 Bakgrund

Ett av kraven för 3DCopy projektet var att det skulle finnas ett grafiskt användargränssnitt för mjukvaran som utvecklades. Då projektet skulle utföras inom ett visst antal timmar måste tiden som en uppgift, exempelvis GUI testning, fås ta begränsas. Eftersom GUI testning kan ta i princip obegränsad tid är det viktigt att hitta en gräns för när GUI testningen kan anses vara klar. Det är också viktigt att mjukvaran som kunden ska få är vältestad, därför är kontexter vid testning något som kan undersökas för att testa tillräckligt mycket på kort tid.

Eftersom att testledaren har ansvar för alla tester som ska utföras på 3DCopy är detta ett relevant område att undersöka då testningen är en viktig del av projektet.

B.3 Teori

Detta kapitel tar upp relevanta teoriaspekter om varför kontexten är viktig samt hur kontexten kan utnyttjas för att ta fram testfall.

B.3.1 Vad är interaktionskontext?

Kontexten för en interaktion är sekvensen av tidigare interaktioner med GUI:t. Till exempel för en interaktion i_i är kontexten följande sekvens: $\langle i_0, i_1, \dots, i_{i-1} \rangle$. Problemet som uppstår är att kontexten kan göras godtyckligt lång och att varje interaktion ska testas i alla möjliga kontexter. Det innebär att ett GUI kan testas oändligt länge [57].

B.3.2 Betydelsen av kontext för interaktioner

Enligt Yuan et al. [57] är kontexten i vilken en interaktion med GUI:t utförs extremt viktigt och medför problem gällande testning av ett GUI. Kontexten kommer då att påverka framtida interaktioner och deras resultat. Ordningen av dessa interaktioner är essentiell för kontexten till en interaktion. För testning av ett GUI innebär detta att varje interaktion måste testas i flera kontexter. Till exempel kanske en viss interaktion genererar ett fel men enbart i en speciell kontext.

För att kunna identifiera fel är det framförallt tre saker som är viktigt. För det första är kontexten till en interaktion eller sekvens viktig, den kan vara avgörande för att hitta vissa fel. För det andra är positionen av interaktionerna i sekvensen viktig, en viss ordning av interaktionerna kan krävas för att hitta ett visst fel. För det tredje är ordningen som interaktionerna och sekvenserna testas i viktig då den påverkar vilka fel som hittas vid testning av GUI [57].

B.3.3 Framtagning av testfall

Den metod som föreslås av X. Yuan et al. [57] innehåller följande steg:

- 1 Generera EFG.
- 2 Generera EIG.
- 3 Välja styrka på testen (Förklaras mer ingående i avsnitt B.3.4).

Modellering av GUI

Det första steget är att representera GUI:t som en event-flow-graph (EFG). En EFG representerar alla möjliga sekvenser av interaktioner som är möjliga att göra med GUI:t. Ett exempel på en EFG återfinns senare i rapporten i figur 15. Subvägarna i EFG:n representerar olika sekvenser av interaktioner. Men antalet subvägar i EFG:n ökar exponentiellt med längden på sekvensen. På grund av den exponentiella ökningen är det i princip omöjligt att kunna testa alla olika subvägar för sekvenser som består av fler än två interaktioner. För att minimera antalet noder i EFG:n omvandlas den istället till en event-interaktion-graph (EIG) [57].

Andra steget är alltså att generera en EIG utifrån sin EFG. En EIG är en EFG utan noder (interaktioner) som inte påverkar mjukvaran bakom GUI:t, d.v.s. öppnandet av exempelvis menyer och fönster. Ett exempel på en EIG återfinns senare i rapporten i figur 16. Metoden för att gå från en EFG till en EIG beskrivs av Xie och Memon [56]. Metoden går ut på att alla interaktioner som enbart påverkar GUI:t tas bort med hjälp av regler för grafomskrivning [57].

B.3.4 Generering av testfall

Första steget när testfallen ska genereras är att välja styrka. Styrkan kan vara två eller högre och säger hur många interaktioner som är intressanta i sekvensen. Till exempel skulle en styrka av två innebära att alla par av interaktioner från EIG:n som har en både mellan sig väljs. Eftersom att det är riktade bågar måste interaktionen som bågen utgår från vara först i paret. Dessa tupler som bildas efter att styrkan är vald utnyttjas sedan för att kunna ta fram kriterier för tillräcklighet hos testfallen [57].

Det första som ska uppnås med testfallen är tillräcklighet för något som kallas "t-cover" där t representerar styrkan. "T-cover" innebär att testfallen exekverar alla tupler av interaktioner, som erhölls när styrka valdes på testfallen, utan att andra interaktioner bryter den sekvensen [57].

Det andra som ska uppnås är "t⁺-cover". "T⁺-cover" innebär att tuplerna exekveras, men med andra interaktioner mellan de som ingår i tupeln. När ett sådant test ingår för varje tupel är "t⁺-cover" uppfyllt till 100 % [57].

Enligt Yuan et al. [57] om både "t-cover" och "t⁺-cover" är uppfyllt anses testfallen vara "t^{*}-cover". Det går även att utöka ytterligare med hjälp av arrayer men då ökar antalet testfall väldigt mycket och riskerar att ta för lång tid för att passa i projektets tidsplan. Därför kommer testerna enbart att uppfylla "t^{*}-cover".

B.4 Metod

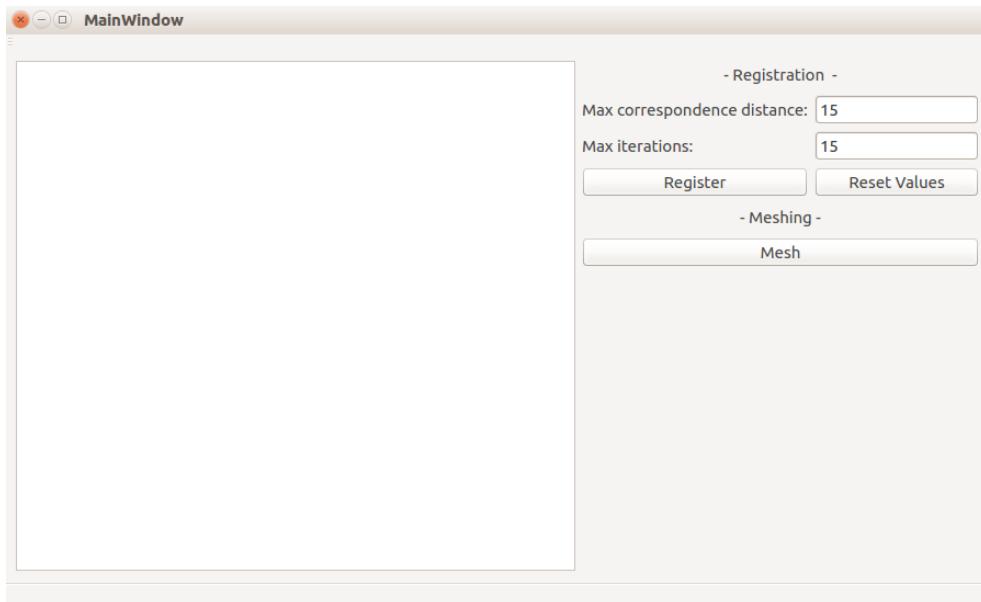
Här beskrivs den metod som har använts vid litteraturstudien och de tester som har utförts på GUI:t. Först genomfördes en litteraturstudie för att samla information. Sedan testades 3DCopys GUI först med och sedan utan hänsyn till kontext för att se vilka fel som hittades av vilken testning. Dessa tekniker utvärderades sedan baserat på testresultatet.

B.4.1 Litteraturstudie

Som första steg utfördes en litteraturstudie för att hitta information om kontexters påverkan vid GUI testning. De sökningar som utfördes under litteraturstudien har genomförts med antingen Google Scholar eller den söktjänst som tillhandahålls av Linköpings Universitets bibliotek. De sökord som användes i litteraturstudien är "GUI testing context" och "context GUI testing". Typ av källa (bok, artikel m.m) har inte medvetet begränsats i litteraturstudien. Endast de två första sidorna med resultat har tagits hänsyn till i litteraturstudien.

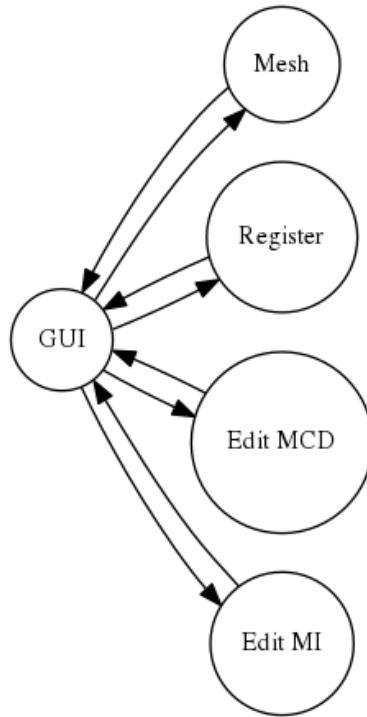
B.4.2 Testning av 3DCopys GUI med hänsyn till kontext

Som test av GUI:t med hänsyn till kontext tas testfallen fram via den metod som presenteras i avsnitt B.3.3.



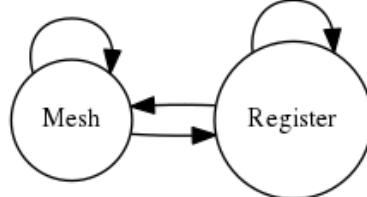
Figur 14: GUI:t för 3DCopy i slutet av iteration 2.

GUI:t har 5 olika interaktioner som kan utföras. Det går att trycka på någon av de tre knapparna eller skriva i någon av de två input fälten. Det stora vita fältet är till för att visa meddelanden från mjukvaran och kan ej interageras med. Utifrån dessa 5 interaktioner kan vi ta fram EFG:n för GUI:t.



Figur 15: EFG:n för 3DCopys GUI.

Utifrån EFG:n ska sedan EIG:n skapas. Det som händer vid skapandet av EIG:n är att noderna GUI, Edit MCD och Edit MI tas bort eftersom att de inte interagerar med mer än enbart GUI:t.



Figur 16: EIG:n för 3DCopys GUI.

Nu när EIG:n är framtagen ska styrka väljas på testen. För att testerna ska skilja sig lite mer från de testerna utan hänsyn till kontext men ändå inte ta för lång tid väljs styrka 3 på dessa tester. Det innebär att vi får följande sekvenser som ska användas för att ta fram testfallen.

Table 1: De sekvenser som ska användas för att ta fram testfallen.

Nr	Sekvens
1	<Mesh, Mesh, Mesh>
2	<Mesh, Mesh, Register>
3	<Mesh, Register, Mesh>
4	<Mesh, Register, Register>
5	<Register, Mesh, Mesh>
6	<Register, Mesh, Register>
7	<Register, Register, Mesh>
8	<Register, Register, Register>

Dessa sekvenser kombineras sedan för att testfallen ska uppnå 3-cover och 3^+ -cover och därfor också 3^* -cover. Först genererades två stycken olika mängder med testfall som enskilt uppfyllde antingen 3-cover eller 3^+ -cover. Dessa kombinerades sedan för att få fram den mängd testfall som ska utföras på GUI:t.

Table 2: De testfall som genomfördes på GUI:t med hänsyn till kontext.

Nr	Testfall
1	<Register, Mesh, Register, Mesh, Register, Mesh>
2	<Mesh, Register, Mesh, Register, Mesh, Register>
3	<Register, Mesh, Mesh, Mesh, Register, Mesh>
4	<Mesh, Mesh, Register, Register, Register, Mesh>
5	<Mesh, Mesh, Register, Mesh, Register, Mesh>
6	<Register, Mesh, Mesh, Register, Mesh, Register>
7	<Register, Mesh, Register, Register, Register, Mesh>
8	<Mesh, Register, Register, Mesh, Mesh, Register>
9	<Mesh, Register, Register, Mesh, Register, Mesh>
10	<Mesh, Register, Mesh, Register, Register, Mesh>
11	<Register, Mesh, Mesh, Register, Register, Mesh>
12	<Register, Register, Mesh, Register, Mesh, Register>

GUI:t testas sedan genom att utföra dessa testfall i den ordning som visas i tabell 2. GUI:t startades om mellan varje testfall. Dessa tester utfördes manuellt på GUI:t.

B.4.3 Testning av 3DCopys GUI utan hänsyn till kontext

Som test av GUI:t utan hänsyn till kontext kan full-branch coverage kan vara tillräcklig. Det vill säga att alla olika vägar i EIG:n för GUI:t testas. Det blir alltså totalt $N!$ antal test där N är antalet interaktioner som kan utföras på GUI:t.

Table 3: De testfall som genomfördes på GUI:t utan hänsyn till kontext.

Nr	Testfall
1	<Register, Mesh>
2	<Mesh, Register>

GUI:t testas sedan genom att utföra dessa testfall i den ordning som visas i tabell 3. GUI:t startas om mellan varje testfall. Dessa tester utfördes manuellt på GUI:t.

B.5 Resultat

Här presenteras de resultat som erhölls från både litteraturstudien och de tester som genomfördes på GUI:t.

B.5.1 Litteraturstudie

Litteraturstudien visar ett tydligt samband mellan kontexten och hur många fel som testerna kunde hitta. Yuan et al. genomförde tester med både varierande styrka och tillräcklighet, alltså de olika formerna av "t-cover" och arrayer. Generellt för dessa tester ökar antalet funna fel när styrkan eller tillräckligheten ökas. Yuan et al. anser att genom ändra dessa parametrar, styrkan och tillräckligheten, kan fel som inte hittas av svagare parametrar identifieras [57].

B.5.2 Testresultat

I detta avsnitt presenteras de resultat som erhölls vid den testning av GUI:t som genomfördes.

Med hänsyn till kontext

Testfallen med hänsyn till kontext lyckades identifiera ett fel i meshningen. Den klarar bara av att mesha en viss fil, *first_registered_church.pcd*. Eftersom att den tar lång tid att mesha utfördes enbart test nummer ett i tabell 2 med denna fil. Resterande tester utfördes med andra filer som fick fel i meshningen. Alltså exekverades inte meshningskoden i de flesta fall utan bara tills felet uppstod.

Utan hänsyn till kontext

De tester som utfördes utan hänsyn till kontext hittade samma fel som de med hänsyn till kontext. Alltså att meshningen enbart klarade av en av filerna. Dessa tester utfördes båda med filer som klarade eller fick fel i meshningen.

B.6 Diskussion

I detta avsnitt diskuteras resultatet, den metod som användes samt källkritik.

B.6.1 Resultat

Från litteraturstudien kan det konstateras att kontexten är avgörande för hur en interaktion påverkar mjukvaran och därmed vilka fel som kan identifieras under testningen.

Som resultatet visar lyckades de båda testen identifiera samma fel trots att det med hänsyn till kontext var mycket mer omfattande. Detta resultat kan bero på många olika saker. Dels hade 3DCopy som testades enbart två systeminteraktioner och dessa två använder ingen delad kod. Därför är sannolikheten för att de påverkar något i koden som förstör för den andra interaktionen väldigt liten.

Dessa interaktioner är också testade sedan tidigare i andra systemtest och därför kan alla fel i koden redan vara lösta. Sannolikheten för att det skulle finnas något fel kvar minskar därför ytterligare. Resultatet som erhölls är därför ganska sannolikt med tanke på omständigheterna. Hade 3DCopy varit större vid testningen med fler interaktioner som delvis använder samma kod hade testningen med hänsyn till kontext eventuellt hittat fler fel.

B.6.2 Metod

En stor brist i litteraturstudien är avsaknaden av ytterligare källor. "Context", i kombination med GUI och testningsrelaterade termer, var de sökord som användes under sökandet av ytterligare källor. Det hade eventuellt varit fördelaktigt att utforska andra termer som beskriver kontext då inga ytterligare relevanta källor hittades. Att söka efter andra termer var dock något som inte genomfördes på grund av tidsbrist.

Den främsta bristen i metoden är att felet i meshningen inte hade hittats och blivit löst innan dessa tester. Eftersom att den enda filen som kunde meshas tog väldigt lång tid var det inte realistiskt att genomföra alla tester med den och därför är inte alla testfall med hänsyn till kontext genomförda med den filen. Det medför i sin tur att de testerna aldrig exekverade koden som utför meshningen och det kan därför finnas kontextkänsliga fel i den biten av koden som inte hittades.

Den kontextkänsliga testningen begränsades också vid t^* -cover. Det finns ytterligare ett steg som är ännu kraftfullare som kanske skulle ha hittat fel som inte identifierades.

Den kontextkänsliga testningen borde också jämförts med en annan GUI testnings metod, inte bara en testning som uppnår full path coverage. Detta för att kunna jämföra olika GUI testnings metoder.

B.6.3 Källkritik

I litteraturstudien hittades bara en relevant artikel som berörde den bit som var mest kritisk för att utföra testerna, nämligen en som tog upp hur GUI:n testas med hänsyn till kontext. Den artikeln är skriven av personer med lång erfarenhet inom just GUI testning vilket framgick när sökandet efter ytterligare artiklar genomfördes eftersom många artiklar som dök upp var skrivna av samma personer. En anledning till att det var svårt att hitta ytterligare artiklar kan vara att Yuan et al. [57] har valt att kalla det för kontext. Då de också definierar begreppet kontext i sin artikel indikerar det att de eventuellt är en av få eller den enda artikeln inom just detta ämne. Andra artiklar kan också använda sig av en annan benämning än just kontext.

B.7 Slutsatser

Här presenteras slutsatserna för frågeställningarna i avsnitt B.1.2.

B.7.1 Hur viktig är kontexten för interaktioner vid testning av GUI?

Att ta hänsyn till kontexten vid GUI testning är definitivt något som bör göras. Den är ofta avgörande för att hitta vissa fel. Det hade varit intressant att skapa exempel GUI:n med medvetet placerade fel som beror på andra interaktioner för att kunna undersöka detta noggrannare. Det fanns dock inte tid att göra en sådan undersökning i denna rapport. Därför får betydelsen av kontexter begränsas till litteraturstudien.

B.7.2 Hur bör testfall utformas för att ta hänsyn till kontexten?

Gällande metoden för att ta fram testfall med hänsyn till kontext användes samma metod som i Yuan et al. [57]. Något som blev tydligt då testerna utfördes manuellt är att metoden inte tar hänsyn till hur resurs- eller tidskrävande en interaktion är när den placeras i sekvensen. Det är något som kan vara intressant för en framtida undersökning. Att undersöka hur de mest krävande interaktionerna ska placeras i sekvenserna för att maximera antalet testade kontexter och samtidigt minimera tids- och resurskostnader.

B.7.3 Hade hänsyn till kontexter någon påverkan vid testning av 3DCopys GUI?

Testerna visade att för projektets GUI, i det stadie som GUI:t befann sig i vid testningen, spelade inte kontexten någon roll för att hitta fel. Det beror sannolikt på att mängden interaktioner var liten samt att systeminteraktionerna var väl separerade och inte delade någon kod. Det indikerar att hänsyn till kontexten blir viktigare ju större mängd interaktioner GUI:t har samt hur stor del av koden som interaktionerna har gemensamt. Hade mer tid funnits skulle en undersökning med olika GUI:n med varierande antal interaktioner varit intressant för att vidare kunna undersöka denna teori.



C

Hur kravhanteringsmetoder påverkar ett utvecklingsprojekt

— Gustav Jannering —

C.1 Inledning

Denna del redogör för projektets analysansvarige Gustav Jannerings utredning kring kravhantering och kravinsamling.

C.1.1 Syfte

Syftet med denna rapport är att undersöka hur olika metoder för kravhantering och elicitering av krav påverkar ett utvecklingsprojekt av samma storlek som det projekt som övriga rapporten beskriver (ett projekt med sju till åtta utvecklare med en budget på 400 timmar vardera). Samt vilka för- och nackdelar som finns med den metoden som användes och vad som kunde gjorts annorlunda.

C.1.2 Frågeställning

Generella frågeställningar

1. Vilka metoder finns för kravhantering och elicitering av krav och vilka fördelar och respektive nackdelar finns med dessa?

Specifika frågeställningar

2. Vilka fördelar och nackdelar finns med att använda IEEE std 830 i ett programvaruutvecklingsprojekt av den storleken som detta projekt?
3. Vilka erfarenheter kan dokumenteras från arbetet med krav i projektet som kan vara intressanta för framtida projekt?

C.1.3 Avgränsningar

Denna rapport kommer att begränsas till de frågeställningar som presenterats i avsnitt C.1.2. För metoder för elicitering av krav har följande avgränsning gjorts: endast generella metoder, metoder som går att applicera i alla situationer och som har studerats vetenskapligt ingår. Inga andra metoder (icke generella eller metoder som inte studerats vetenskapligt) kommer

att presenteras i denna rapport. De erfarenheter som dokumenteras senare i denna rapport kommer uteslutande från kursen **TDDD96 Kandidatprojekt i programvaruutveckling** och mer specifikt från det projekt i kursen som presenterats tidigare i denna rapport.

C.1.4 Definitioner, akronym och förkortningar

Följande definitioner och förkortningar används på flera ställen i denna del av rapporten:

- Elicitering av krav - Identifiering och uppsamling av krav, översatt från det engelska ordet *requirements elicitation*.
- Kravhantering - arbetet med krav som ett teknisk system ska uppfylla. Översatt från det engelska ordet *requirements engineering*.
- IEEE - Institute of Electrical and Electronics Engineers, en icke-vinstorienterad organisation bestående av ingenjörer och vetenskapspersoner. Upprätthåller standarder inom ingenjörsvetenskap.
- Gamification - Processen att lägga till spel eller spellika element till något (som en uppgift) för att uppmuntra deltagande [31].
- User story - En informell beskrivning av funktioner i ett mjukvarusystem.
- Acceptance test - Ett test utfört för att avgöra om kraven i en kravspecifikation är uppfyllda.

C.2 Bakgrund

Arbetet med kravhantering och elicitering av krav i projektet (som presenteras tidigare i denna rapport) började vid det första mötet med kunden. Detta var ett möte, dels för att träffa kunden efter att projektgruppen hade blivit tilldelade projektet, och dels för att starta kravinsamlingsfasen. Kunden i projektet, CVL, sade vid detta möte att de helst ville styra projektet med en lös hand och att det var mer eller mindre upp till projektgruppen att komma med en lista på förslag på krav (i form av ett första utkast till en kravspecifikation). För att skapa denna lista med förslag till krav på den slutgiltiga produkten gick gruppen gemensamt igenom det projektdirektiv som CVL tidigare hade presenterat (återfinns i bilaga H.2), för att identifiera vilka funktioner som produkten skulle ha. Utifrån dessa funktioner kom gruppen med förslag på krav. Denna metod kallas i facklitteratur för *introspektion* [17]. Användandet av denna metod innebar att kunden bara var involverad i att godkänna de krav som gruppen föreslog och var minimalt involverade i kravidentifieringsprocessen.

Examinatoren i kursen (TDDD96 Kandidatprojekt i programvaruutveckling, Linköpings universitet), som detta projekt genomfördes under, hade bestämt att IEEE std 830 var den standard som kravspecifikationen skulle följa (för mer information om IEEE std 830 se kapitel C.3). Detta projekt var första gången då någon i projektgruppen använde standarden, vilket medförde att vi inte hade tillräcklig kunskap för att skräddarsy standarden till projektet, istället följde vi standarden så gott vi kunde. Målet med en kravspecifikation är att identifiera kundens vision på hur den slutgiltiga produkten ska se ut och sedan tolka huruvida detta faktiskt är det som kunden vill ha eller behöver och specificera de krav som produkten ska vara bunden av.

C.3 Teori

Kravhantering, kan som tidigare nämnts, definieras som *arbetet med krav som ett teknisk system ska uppfylla*. Detta arbete ligger ofta tidigt i ett projekt, då det är viktigt att specificera

de krav som skall uppfyllas. Kravhanteringsprocessen är ofta uppdelad i fyra delar: elicitering, analys, specifikation och godkännande. Detta arbete tar främst upp de tre första delarna. I detta arbete definieras elicitering som: "insamlingen av krav eller information för senare specificering av krav från användare, kunder och andra intressenter". I analysfasen analyseras de kraven och den informationen som samlades in under eliciteringsfasen för att undersöka huruvida dessa krav är relevanta och huruvida informationen går att använda för att specificera krav. Under specificeringsfasen ska de slutgiltiga kraven specificeras, detta innebär att varje krav numreras och att en kort text skrivs som detaljerar kravet.

C.3.1 IEEE standard 830

IEEE standard 830 är en standard, publicerad av IEEE, som specificerar innehållet och kvaliteterna hos en "bra" kravspecifikation [6]. Standarden specificerar både hur en kravspecifikation borde skrivas (hur involverad kund/intressenter ska vara, hur kravspecifikationen borde utvecklas med tiden m.m.) och vilka avsnitt som borde finnas med och deras syften. Enligt IEEE ska kraven som skapas med standarden ska vara: korrekta (eller relevanta), entydiga, kompletta, konsekventa, rangordnade efter betydelse, verifierbara, modifierbara och spårbara [6].

C.4 Metod

För att undersöka vilka metoder som används för kravhantering har en litteraturstudie av publicerade vetenskapliga rapporter genomförts för att identifiera vilka metoder som har studerats och hur dessa används idag. Litteraturstudien inleddes med en litteratursökning på universitetets biblioteksdatabas och på Googles databas, Google Scholar [18]. Svårigheten med denna ansats har varit att begränsa antalet källor och försöka göra relevanta avgränsningar för att passa arbets begränsningar. De rapporter som ingår i arbetet har uteslutande fokuserat på eliciteringsprocessen av kravhantering. Anledningen till att dessa har valts är att de uppfyller kriterierna som presenterats ovan och i avsnittet C.1.3. Vidare har endast de artiklar som presenterat eller studerat generella metoder använts. Dessa avgränsningar är gjorda för att göra denna rapport relevant för både mindre och större utvecklingsprojekt.

För att identifiera för- och nackdelar med IEEE std 830 [6] har dels gruppen och dels analysansvariga från andra projektgrupper i denna kurs intervjuats i en öppen intervju. Se bilaga H.3 för den intervjuguide som användes vid intervjuerna.

För att dokumentera erfarenheter från arbetet med kravhantering och elicitering av krav har projektgruppen gått igenom den kravspecifikation som framställdes i projektet och utvärderade vårt arbete. Erfarenheter har också dokumenterat under arbetet med denna rapport.

C.5 Resultat

C.5.1 Metoder för elicitering av krav

På en vetenskaplig konferens i San Diego 1993 presenterade C. Linde och J. A. Goguen en rapport som beskrev författarnas undersökning och utvärdering av olika metoder för att elicitera krav [17]. I rapporten skriver författarna om följande metoder: Introspektion, öppna intervjuer, frågeformulär och fokusgrupper. I det följande hänvisas huvudsakligen till C. Linde och J. A. Goguen rapport [17].

Introspektion

Introspektion kallas den metod som användes i projektarbetet som presenterats tidigare i denna rapport. För att elicitera krav med introspektion, föreställer sig personen som eliciterar

krav vilket system som hen skulle vilja ha om hen var kunden. Författarna skriver i sin rapport att introspektion är den mest självklara metoden för att specificera krav men att det finns fall då denna metod misslyckas med att specificera viktiga krav. Detta kan ofta bero på personliga fördomar (från eng. *bias*). Två personer med olika bakgrund (både personlig och professionell) kan komma med olika krav då de använder introspektion för att elicitera krav. Det är även svårt för en enskild person att identifiera samtliga krav som bör ställas på ett system. C. Linde och J. A. Goguens slutsats är att introspektion är en användbar metod för elicitering av krav om den kombineras med andra metoder och att de eliciterar krav med hjälp av introspektion kommer från olika bakgrunder.

Frågeformulär

Frågeformulärsintervjuer används inom många forskningsområden. Formulär där försökspersoner får svara på frågor med förutbestämda svar är ett bra sätt för forskare att få statistisk information. Detta tillvägagångssätt kan anpassas för att användas för att elicitera och specificera krav. Tänkta användare, kunder och andra intressenter får, med hjälp av den som eliciterar krav, fylla i ett formulär, vars data senare analyseras för att identifiera vad som är relevant för de som svarade på formuläret. Liksom i forskningsvärlden så är detta ett sätt att få statistisk information som kan visas för kunden och användas som underlag i en mängd situationer under utvecklingen av ett system. Författarna poängterar att det finns ett fundamentalt problem med detta tillvägagångssätt: - Vad händer om den som skapar formuläret och frågorna, och den som svarar på frågorna inte har samma referensram?.

Öppna intervjuer

C. Linde och J. A. Goguen kommer fram till att ett sätt att lindra problemet med frågeformulärsmetoden är att istället hålla s.k. öppna intervjuer. Intervjuaren ställer en fråga och låter försökspersonen svara på frågan med egna ord. Intervjun fortskrider med ett antal förutbestämda frågor som ska ställas och besvaras men intervjuaren kan ställa följdfrågor och egna frågor emellan de förskriva frågorna. Denna metod används ofta av psykologer och andra som forskar inom liknande områden. För att anpassa denna metod för elicitering av krav ändrar man vilka frågor man ställer och vilka personer man ställer frågorna till. Frågorna kommer att handla om hur försökspersonen använder system som liknar det system som ska utvecklas. Det är viktigt att frågorna är konstruerade så att de inte leder till att försökspersonen använder introspektion för att själv föreställa vilka krav som personen tycker systemet bör ha. Försökspersonerna kommer, liksom med frågeformulärsmetoden, att vara tänkta användare, kunder och andra intressenter.

Fokusgrupper

Den sista metoden för elicitering av krav som C. Linde och J. A. Goguen presenterar i sin rapport är fokusgrupper (från eng. *focus groups*). Fokusgrupper är mycket vanligt i marknadsundersökningar. Tanken bakom fokusgrupper är att samla en grupp intressenter (potentiella kunder om en marknadsundersökning utförs) med olika bakgrunder. Gruppen får sen diskutera ett ämne som är av intresse (t.ex. en ny produkt om en marknadsundersökning utförs). Arrangören för protokoll på vad som sägs i gruppen och använder senare denna data som underlag vid beslut [20]. För att applicera denna metod för att elicitera krav skapas JAD- (*Joint Application Development*) eller RAD- (*Rapid Application Development*) grupper bestående av utvecklare, kunder och andra intressenter som i grupp diskuterar det system som ska utvecklas. Författarna poängterar att det är svårt att inkludera personer som saknar teknisk bakgrund i dessa samtal då de har svårigheter att bedöma betydelsen av tekniska beslut. Författarnas slutsats är att denna metod är lovande men att dess begränsningar bör studeras.

GREM

I en rapport som presenterades på International Working Conference on Requirements Engineering: Foundation for Software Quality i Göteborg 2016 skriver P. Lombris et al. [31] om en ny metod för elicitering av krav som bygger på gamification. Rapporten beskriver metod som ska öka engagemanget hos interressenter vid elicitering av krav som författarna kallar *gamified requirements engineering model* eller GREM. I rapporten presenterar författarna ett experiment där en webbaserad plattform, som har spellika element, används för att elicitera krav genom att skapa user stories och acceptance tests. Författarnas slutsats är att deras experiment visar att användandet av spellika element kan ha en positiv inverkan på intressenters engagemang men att denna inverkan kan variera baserat på vilka spellika element som används.

Kravkategorisering

I sin doktorsavhandling definierar M. Karlsson tre olika kategorier för krav: *fångade krav* (från eng. captured requirements), *eliciterade krav* (från eng. elicited requirements) och *framväxande krav* (från eng. emergent requirements) [22]. *Fångade krav* är krav som är relativt lätt att specificera eller som är relativt uppenbara. *Eliciterade krav* är krav som identifierats efter att grävt djupare i vad som tänkta användare, kunder och andra intressenter förväntar sig av systemet. Dessa krav kan klassas som icke uppenbara och kräver en grad av utredning för att identifieras. *Framväxande krav* är krav som växer fram under utvecklingen av systemet allt efter att tänkta användare, kunder och andra intressenter fått testa systemet. Dessa krav är svåra att identifiera i ett utvecklingsprojekts tidiga stadier om utvecklingsgruppen inte håller prototypdemonstrationer där tänkta användare får komma med feedback.

C.5.2 Erfarenheter

Följande är en sammanfattning av de erfarenheter om kravhantering och elicitering av krav som dokumenterats under projektets gång.

- Att skapa en *bra* kravspecifikation är svårt.
- För att identifiera relevanta krav gäller det att både utvecklingsgruppen och kunden är tillräckligt insatta i det som ska utvecklas. (speciellt om projektets mål är att vidareutveckla ett system).
- Att blint följa IEEE std 830 för att skriva en kravspecifikation kan vara frustrerande.
- Kunden (och/eller tänkta användare) borde vara mer involverad vid specificering av krav än de var i projektet.
- Vid vidareutveckling är det viktigt att testa det befintliga systemet ordentligt innan man specificerar krav.

C.5.3 IEEE standard 830

Följande är en sammanfattning av två öppna intervjuer gjorda med två analysansvariga från två olika grupper i kurserna **TDDD96 Kandidatprojekt i programvaruutveckling**. Intervjuguiden som användes vid dessa intervjuer återfinns i bilaga H.3. Svaren på intervjuerna är anonymiserade.

Båda intervjuobjekten använde introspektion för att elicitera krav i sina projekt. De började med att analysera det projektdirektiv som gavs för respektive projekt, eliciterade krav utifrån dessa och skapade ett första utkast till en kravspecifikation som de sedan skickade till sina respektive kunder. När det gäller användandet av IEEE std 830 svarade det ena intervjuobjektet att de inte använt sig av denna standard vid skrivandet av kravspecifikationen.

Intervjuobjektets grupp baserade sin kravspecifikation på en mall från en tidigare projektkurs och undersökte IEEE std 830 först efter att en motiverande grupp hade påpekat att IEEE std 830 var standarden som skulle användas. Det andra intervjuobjektets grupp använde sig av IEEE std 830 från början. Efter utfrågning om vad de tyckte om standarden höll båda intervjuobjekten med om att standarden gav en bra mall att följa när det gällde vilka rubriker som bör vara med i en kravspecifikation men att standarden kändes för omständlig för ett projekt av denna omfattning och att de kände att det tog för lång tid att läsa igenom och sätta sig in i standarden.

C.6 Diskussion

C.6.1 För- och nackdelar med metoder för elicitering av krav

Nedan används M. Karlssons kravkategorisering för att diskutera vilka metoder som är kapabla att specificera vilka krav [22]. Den tredje kategorin av krav, *framväxande krav*, är mycket svåra att korrekt identifiera tidigt i ett projekt varför den kommer inte att användas.

Introspektion

Min åsikt om för- och nackdelarna med introspektion stämmer överens med C. Lindes och J. A. Goguens slutsats om introspektion [17]. Introspektion är en bra metod för elicitering av krav om den kombineras med andra metoder och att de som eliciterar krav med hjälp av introspektion kommer från olika bakgrunder. Ren introspektion kräver mycket erfarenhet för att kunna elicitera alla relevanta krav som bör ställas på ett system. Introspektion klarar i nästan alla fall att identifiera alla *fångade krav*, eftersom de kraven är mer eller mindre uppenbara. Men det krävs en betydande mängd erfarenhet för att identifiera samtliga *eliciterade krav*.

Frågeformulär

Frågeformulär, liksom introspektion, är beroende av att den som eliciterar krav (och i detta fall skapar frågorna och svaren) har en korrekt bild av vad kunden, tänkta användare och andra intressenter vill ha för system. Om denna bild inte är korrekt finns det en stor risk att de krav som eliciteras inte är relevanta. Liksom introspektion klarar frågeformulär av att elicitera *fångade krav*, men om den som skapar frågorna och svaren, och försökspersonen inte delar samma referensram finns risken att vissa *eliciterade krav* missas.

Öppna intervjuer

Användandet av öppna intervjuer kan minska det tidigare nämnda problemet med referensramar som frågeformulärsmetoden har. Om intervjuaren och försökspersonen inte har samma uppfattning om vad som ska utvecklas eller vilka systemets fundamentala aspekter är finns risken att kraven som specificeras med hjälp av intervjun inte är relevanta. Öppna intervjuer kommer nästan alltid att identifiera alla *fångade krav*, men liksom frågeformulär finns det en risk att vissa *eliciterade krav* inte specificeras. Ett annat problem med denna metod är tidsaspekten. Öppna intervjuer tar tid. Eftersom intervjuerna görs en och en kommer tiden det tar att elicitera krav från samma mängd intressenter att ta mycket längre tid om man använder öppna intervjuer istället för frågeformulär.

Fokusgrupper

En risk med fokusgrupper är att deltagarna i gruppen inte representerar verkligheten, d.v.s. att deltagarna inte lyckas representera en eller flera grupper av intressenter, vilket leder till att vissa relevanta krav inte identifieras. Det finns också andra risker med fokusgrupper.

Eftersom en fokusgrupp ska starta en diskussion kring ett ämne finns risken att en minoritet av gruppen inte blir hörd eller att delar av gruppen hörs för mycket. Det kan vara lätt för en majoritet att utesluta en minoritet från diskussionen eller att en minoritet tar över diskussionen. Detta kan leda till att de som representerar en grupp intressenter inte blir hörd och att relevanta krav inte blir eliciterade. Fokusgrupper har enligt min uppfattning störst chans att korrekt identifiera samtliga *eliciterade krav* så länge gruppen fungerar korrekt.

GREM

GREM verkar vara en lovande metod men den är väldigt ung (rapporten som presenterade metoden skrevs 2016) och är beroende av extern mjukvara. GREM har potential men bör studeras mer och/eller standardiseras innan den används mer brett. I teorin har GREM potentialen att korrekt identifiera alla *eliciterade krav* i form av user stories och acceptance tests skapade av kunder, eftersom GREMs mål är att involvera kunder, användare och andra intressenter i eliciteringsprocessen.

C.6.2 För- och nackdelar med IEEE std 830

Att intervjuva två personer för att identifiera för- och nackdelar med en standard som är skapad av en organisation med tusentals medlemmar, och som har mycket större kunskap och erfarenhet av ämnet, är inte optimalt, men det kan ge en bra bild av hur personer som försöker sätta sig in i standarden känner. Efter intervjuerna och efter att ha reflekterat över mina egna erfarenheter av standarden kan jag konstatera att våra åsikter om standarden är väldigt lika. IEEE std 830 ger en bas att bygga en välkonstruerad kravspecifikation på men i detta fall, ett mindre projekt med relativt få projektmedlemmar, känns standarden överflödig.

C.6.3 Erfarenheter

Erfarenheterna som finns listade i avsnitt C.5.2 är specifika för detta projekt, ett projekt där kraven fick skrivas om efter det att fel med den befintliga mjukvaran som skulle vidareutvecklas hittats. Detta är ett tveeggat svärd, dels ger det bra erfarenheter att se tillbaka på om hur vi borde ha gjort för att identifiera dessa fel tidigare. Men det som ledde till att vi inte identifierade felet i ett tidigt skede var en serie av val (både från vår och från kundens sida) som där och då verkade försumbara, men som i efterhand visar att vi, från gruppens sida, borde ha ställt högre krav på kundens kunskap om systemet innan vi specificerade krav på vår vidareutveckling av den befintliga mjukvaran som implicit antog att den befintliga mjukvaran var relativt felfri. Med detta i åtanke skulle jag hävda att de erfarenheter som är listade i avsnitt C.5.2 alla är relevanta erfarenheter för framtida projekt.

C.6.4 Källkritik

De källor som använts i denna rapport är alla publicerade källor som citerats i tidigare arbeten. Den första källan som har använts är publicerad av IEEE, som nämns i avsnitt C.1.4 är IEEE en icke-vinstdrivande organisation som upprätthåller standarder inom ingenjörsvetenskap. Den andra källan är publicerad av ett stort förlag och presenterades dessutom på en internationell konferens. Den tredje och sista är en doktorsavhandling från Chalmers. Ovanstående leder mig till att anse att dessa källor är pålitliga.

C.6.5 Alternativa metoder

Det finns en uppsjö med metoder som skulle kunna ha använts för att besvara de frågor som ställs i avsnitt C.1.2 inom ramen för detta arbets begränsningar. Problemet i detta fall har varit att finna metoder för att besvara frågorna som passade detta arbets tidsbegränsningar.

Av denna anledning valdes just en begränsad litteraturstudie tillsammans med öppna intervjuer och en utvärdering för att besvara frågorna. Om tidsbegränsningen för detta arbete varit annorlunda kunde man utvidgat både litteraturstudien och de öppna intervjuerna.

C.7 Slutsatser

I det följande kommer slutsatserna att redovisas i samma följd som frågorna i avsnitt C.1.2

C.7.1 Vilka metoder finns för kravhantering och elicitering av krav och vilka fördelar och nackdelar finns med dessa?

De metoder för elicitering av krav som presenterats är: introspektion, frågeformulär, öppna intervjuer, fokusgrupper och GREM.

Introspektion

Introspektion är den vanligaste metoden för att elicitera krav men min slutsats är att i många fall är detta inte den bästa metoden att använda. Det finns för många felkällor med denna metod för att säga att man borde använda ren introspektion (bara introspektion) för att elicitera krav. Metoden bygger på att den som eliciterar krav lyckas få en komplett bild av det system som kunden/användaren vill ha. Denna bild kommer att vara komplex, även i mindre projekt, vilket betyder att många eliciterade krav kan falla bort på vägen.

Frågeformulär

Frågeformulär har fördelen att kunna samla in statistisk information om vad kunder/användare vill ha för system men som tidigare presenterats finns ett ganska fundamentalt problem med denna metod: Vad händer om den som skapar formuläret och frågorna och den som svarar på frågorna inte har samma referensram?. Min slutsats är att frågeformulär är en bättre metod för att elicitera krav än introspektion då man involverar kunder och användare mer, men att även den är bristfällig.

Öppna intervjuer

Öppna intervjuer kan vara en bättre metod för elicitering av krav eftersom kunder och användare är mer engagerade i elicitingsprocessen. Dock finns det två problem med metoden. Dels samma problem som med frågeformulär (och introspektion) och dels tidsaspekten. Att utföra tillräckligt många öppna intervjuer för att identifiera alla relevanta krav kan ta för lång tid och kosta för mycket för ett mindre utvecklingsprojekt.

Fokusgrupper

Fokusgrupper har potentialen att vara den bästa metoden för elicitering av krav (så länge en eller flera gruppen, med rätt komposition, kan samlas). Fokusgrupper lider inte av samma problem med tid som öppna intervjuer och den referensram som gruppen har borde vara uppenbar efter att gruppen har träffats. Problemen med fokusgrupper (som presenterats tidigare) och C. Lindes och J. A. Goguens slutsats, att metodens begränsningar bör studeras, får mig att tveka att rekommendera metoden [17].

GREM

GREM är den metod som är svårast för mig att rekommendera. Utifrån den data som presenterats av P. Lombriser et al. [31] verkar metoden lovande men eftersom metoden är beroende av extern mjukvara för att elicitera krav och dessutom är så ny kan jag inte komma till slutsatsen att denna metod kan rekommenderas.

Sammanfattning

Alla metoder har sina egna styrkor och svagheter.

Det bästa tillvägagångssättet för att elicitera krav är att använda flera av de metoder som presenterats för att säkerställa att alla krav som bör ställas på ett system identifieras och specificeras.

C.7.2 Vilka fördelar och nackdelar finns med att använda IEEE std 830 i ett programvaruutvecklings projekt av den storleken som detta projekt?

IEEE std 830 ger en bra översikt över vad som bör vara med i en välkonstruerad kravspecifikation, men att dessa fördelar inte väger upp hur lång tid det tar att sätta sig in i standarden för utvecklare i ett mindre projekt. Efter att ha läst igenom standarden förstår man hur väluttänkt den är. Problemet med standarden är att det tar lång tid att sätta sig in i. Tid som kan vara knapp i början av ett projekt.

C.7.3 Vilka erfarenheter kan dokumenteras från kravhantering och elicitering av krav i projektet som kan vara intressanta för framtida projekt?

De erfarenheter som har kunnat dokumenteras kring ämnet kravhantering och elicitering av krav är följande:

- Att skapa en välkonstruerad kravspecifikation är svårt.
- För att identifiera relevanta krav gäller det att både utvecklingsgruppen och kunden är tillräckligt insatta i det som ska utvecklas (speciellt om projektets mål är att vidareutveckla ett system).
- Att blint följa IEEE std 830 för att skriva en kravspecifikation kan vara frustrerande.
- Kunden (och/eller tänkta användare) borde vara mer involverad/de vid specificering av krav än de var i projektet.
- Vid vidareutveckling är det viktigt att testa det befintliga systemet ordentligt innan man specificerar krav.

D

Analys av punktmolnsregistrering

— Michael Karlsson —

D.1 Inledning

I denna rapport behandlas undersökningen av olika registreringsalgoritmer utförd av Michael Karlsson i samband med TDDD96 *Kandidatprojekt i programvaruutveckling* samt de problem som projektgruppen stött på gällande registrering av punktmoln.

D.1.1 Syfte

Syftet med den här delen är att väga olika algoritmer mot varandra och undersöka hur väl de fungerade för det projekt som beskrivs i grupprapporten samt vilka problem gruppen hade med de olika algoritmerna som testades. Gruppens tillvägagångssätt för val av algoritm undersöks också.

D.1.2 Frågeställningar

1. Hur skapas ett enhetligt punktmoln från bilder tagna med en fast avståndskamera?
2. Hur gör man för att välja algoritm och hur resonerade gruppen när de valde ICP?

D.1.3 Definitioner och förkortningar

Här listas de definitioner och förkortningar som används i den här rapporten.

- Point Cloud Library (PCL) [37] - Ett C++ bibliotek för hantering av punktmoln.
- Iterative Closest Point (ICP) - En algoritm för punktmolnsregistrering.
- JR-MPC [11] - Joint Registration of Multiple Point Clouds.

D.1.4 Avgränsningar

För registrering finns en ofantlig mängd algoritmer och tillvägagångssätt. Denna utredning begränsas till de metoder som använts inom projektet.

D.2 Bakgrund

I ett tidigt stadio av projektet upptäckte gruppen att en stor del skulle handla om att få registrering av punktmoln att fungera bra. Tidiga tester visade dessutom att meshningen, en annan del som tidigt verkade omfattande, som görs efter registreringen var förhållandevis enkel att utföra. Meshning innebär att det, utifrån det kompletta punktmolnet, skapas en vattentät 3D-modell. Författaren blev intresserad efter att ha jobbat med tidiga försök till registrering och påbörjade en djupdykning i ämnet.

D.3 Teori

För att återskapa fysiska objekt digitalt saknas lämpliga verktyg, såsom tex en 3D-kamera. Istället behöver flera bilder av objekten tas från olika vinklar för att kunna sammanfoga dessa till det kompletta objekten. Det finns många metoder för detta, exempelvis Kinect Fusion, se avsnitt D.3.2. Den metod projektgruppen använt sig av har dock bestått av en avståndskamera på en linjärenhet med objekten monterat på ett rotationsbord med två rotationsaxlar. Med hjälp av detta rotationsbord kan hela objekten ses från en fast punkt. Det primära målet har varit att enhetligt registrera de punktmoln som kommit av skanningar med denna uppställning till en komplett representation av objekten.

D.3.1 Registrering

Registrering är den generella metoden att sätta ihop två eller flera punktmoln till ett i ett gemensamt koordinatsystem med all information från punktmolnen. Det finns väldigt många olika algoritmer för att utföra detta såsom till exempel ICP, eller GMMReg, *Gaussian Mixture Model Registration*. De flesta tillgängliga algoritmer har någon form av svaghet eller nackdel. Till exempel lider ICP av så kallad *felutbredning*, där ett litet fel i ett stadio orsakar större fel i senare stadier och JR-MPC, som utnyttjar GMM, lider av längre körtid [11].

D.3.2 Kinect Fusion

Kinect Fusion använder sig av Microsofts egna avstånds- och RGB-kamera, Kinect, för att mappa upp ett 3D-objekt i den verkliga världen. Avståndskameran mäter upp punktmolnet för scenen framför den samtidigt som en synkroniserad RGB-kamera ger färgdata till varje punkt. Varje punktmoln som Kinect-kameran skickar består av ca. 307 000 punkter. Det kan jämföras med hårdvaran som används av gruppen där maximala antalet punkter i en skanning är ca. 786 000 punkter. De punktmoln som används har dock generellt innehållit ca 30-60 000 punkter efter filtrering av skräpdata. Skräpdata är de punkter som inte är intressanta. Registreringen som görs av Kinect Fusion är en variant av ICP.

D.4 Metod

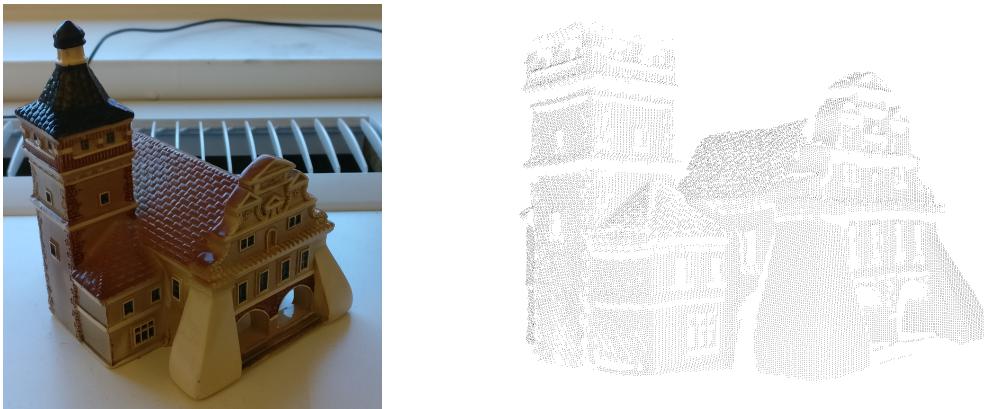
Här presenteras metoden som används för att besvara frågeställningarna i den här rapporten. De är uppdelade i hur arbetet för att besvara frågorna gick till i förstudien samt under utvecklingsarbetet för rapporten.

D.4.1 Litteraturstudie

Initialt gjordes en litteraturstudie för att hitta information om vilka tillvägagångssätt för att registrera punktmoln som finns. De söktermer som används är "registration pointcloud" samt "registration algorithm". Sökmotorn som används är Google Scholar. De källor som undersöks har begränsats till att vara relaterade till JR-MPC samt ICP.

D.4.2 Experiment

Ett experiment utfördes för att jämföra ICP och JR-MPC. Experimentet gick ut på att registrera en uppsättning av 36 punktmoln. Vart och ett av dessa punktmoln var en skanning av kyrkan, se figur 17, där kyrkan skannats och roterats 10 grader efter varje skanning. 3DCopy användes för att testa ICP medan JR-MPC testades med kod som tillhandahålls av skaparna, skriven i MatLab. För JR-MPC är enbart inläsningen av indata modifierad. För ICP valdes *Max Correspondence Distance* till 15, *Max Iterations* sattes till 100 samt *Transformation Epsilon* var satt till $1e - 8$. Övriga parametrar i PCLs implementation av ICP användes utan modifikation. För beskrivning av dessa parametrar, se avsnitt D.5.1. Varje algoritm kördes tre gånger och resultatet mellan körningarna jämfördes. Testdatan finns tillgänglig på GitHub [52]. PCD-filerna användes för 3DCopy medan TXT-filerna användes för JR-MPC. TXT-filerna skapades genom att kopiera PCD-filerna, byta filändelse samt ta bort den *header* som finns i PCD-filerna. Punkterna i TXT-filen är därför identiska med PCD-filerna.



Figur 17: Kyrkan som skannats samt ett resulterande punktmoln från en enkel skanning.

D.5 Resultat

Här presenteras av resultatet av den litteraturstudie som genomförts samt den jämförelse som gjorts baserat på studien. Dessutom presenteras resultatet av det experiment som utförts.

D.5.1 Litteraturstudie

PCL [37] ger många handledningsexempel med exempelkod som var utgångspunkten för gruppens registreringsalgoritm. PCL ger också en bra sammanfattning men utöver det saknas relevanta källor. Då utredningen är begränsad till ICP och JR-MPC begränsas antalet källor ytterligare. Det finns enormt många varianter av ICP, såsom SequentialICP, Point-to-Plane ICP, Point-to-Point ICP m.fl. Den variant som utvärderas här är den som implementeras av PCL, Point-to-Point.

JR-MPC [11] är resultatet av en forskningsstudie som tagit fram algoritmen och därmed finns enbart den källa som används här för den algoritmen.

Bellekens et al. [3] presenterar en genomgående studie där olika algoritmer jämförs.

ICP

ICP är en optimeringsalgoritm som försöker minimera det totala avståndet från varje punkt i ursprungsmolnet till varje punkt i målmolnet. Det ger algoritmen en tidskomplexitet $O(n * m)$ där n är antalet punkter i ursprungsmolnet och m är antalet punkter i målmolnet. ICP ett ursprungsmoln som aldrig ändras samt ett målmoln som algoritmen försöker passa in så att det stämmer överens med ursprungsmolnet.

ICP har ett antal parametrar för att bestämma hur registreringen ska gå till. De parametrar som undersöks inom gruppen beskrivs nedan.

- **Max Iterations** anger hur många registreringsiterationer som algoritmen får utföra innan den tvingas avsluta.
- **Transformation Epsilon** avgör hur stor förflyttning algoritmen får göra på målmolnet inom en iteration.
- **Max Correspondence Distance** anger hur stort avståndet från en punkt i ursprungsmolnet till motsvarande punkt i målmolnet får vara. Är avståndet större än detta kommer algoritmen ignorera punkten i målmolnet vid registreringsförsöket.
- **RANSAC Outlier Rejection Threshold** anger hur stort avstånd som en punkt tillåts existera från den antagna matematiska modellen för att användas i registreringen. RANSAC, *Random Sample Consensus* är en metod för att avgöra om en punkt följer det matematiska mönster som algoritmen letar efter.
- **Euclidean Fitness Epsilon** representerar det fel som accepteras. Är skillnaden i felet mellan två iterationer mindre än det här kommer algoritmen avsluta.

JR-MPC

JR-MPC använder en så kallade GMM, *Gaussian Mixture Model*, för att registrera objektet. Exakt hur matematiken bakom algoritmen ser ut och fungerar är utanför omfattningen av denna rapport men grundläggande principer följer. GMM är en sannolikhetsmodell som används för att lista sannolikheten att ett kluster av punkter i punktmolnet är en del av objektet som ska registreras, detta motsvarar RANSAC filtreringen som görs i ICP. Själva registreringen görs genom en villkorlig maximering, ECM, *Expectation Conditional Maximization* som optimerar registreringen med begränsade variationer i registreringsparameterar. För fördjupning om ECM hänvisas till Roche [45].

D.5.2 ICP vs. JR-MPC

Vid jämförande av ICP och JR-MPC finns ett par signifikanta skillnader. Dessa presenteras här.

På grund av att ICP är en mer rudimentär algoritm än JR-MPC blir den också lättare att implementera. JR-MPC bygger på mer avancerad matematik som måste förstås för att kunna implementera den.

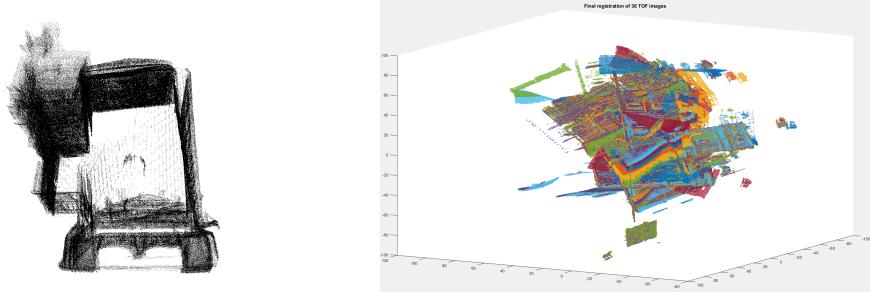
Enligt Evangelidis et al. [11] är ICP en aning snabbare än JR-MPC vilket dock inte har replikerats i projektets begränsade tester. Då tidsåtgången för komplett registrering av 36 punktmoln med ICP är i närheten av tre timmar har inte någon ingående studie genomförts.

Som nämntes i avsnitt D.3.1 påverkas algoritmerna väldigt olika av fel, såsom till exempel avrundningsfel som är oundvikligt med flyttal. Då ICP¹ bara tillåter parvis registrering, det vill säga att ett punktmoln registreras med ett annat kommer dessa avrundningsfel växa och bli större ju fler punktmoln som registreras. Dessutom sparas i 3DCopy det parvis registrerade punktmolnet som ett färdigt oföränderligt punktmoln och ev avrundningsfel kan bara elimineras genom att köra algoritmen igen med andra parametrar.

D.5.3 Experiment

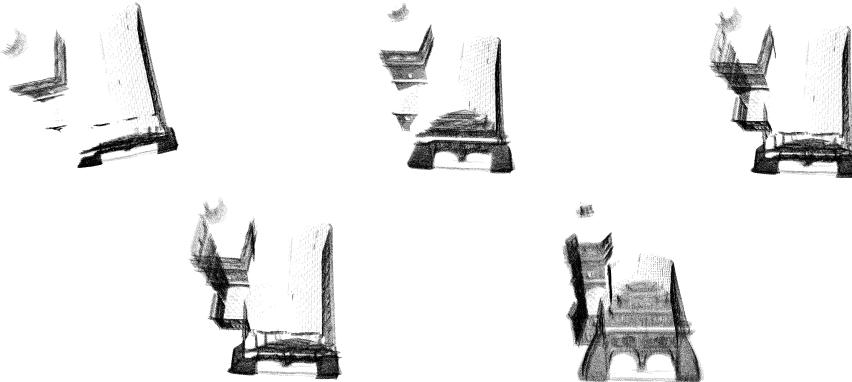
Resultatet av experimentet som utfördes presenteras här med hjälp av ett antal bilder.

¹Notera att detta endast gäller vissa ICP algoritmer



Figur 18: Till vänster: resultat från registrering med ICP. Till höger: resultat från registrering med JR-MPC.

Figur 18 visar resultatet av experimentet. ICP algoritmen har lyckats producera ett resultat som återspeglar kyrkan med några få felaktigheter. JR-MPC däremot har placerat alla punktmoln i varandra på ett sätt som varken liknar punktmolnen innan registrering eller den kyrka som förväntades. I figur 19 ses från övre vänstra till nedre högra bilden delresultatet av 12, 13, 18, 19 samt 20 punktmoln extraherat från en och samma körning av 3DCopy. Notera till exempel resultatet för 20 punktmoln där det tydligt i främre vänstra hörnet på kyrkan syns dubbla hörn. Det här kommer av att punktmolnen som registreras ligger i varandra vid starten. Efter 12 punktmoln behöver därmed nästa moln roteras 130 grader och sannolikheten för ett falskt optimum ökar.



Figur 19: Serie av delresultat från registrering med ICP.

Den poäng (eng. *Fitness score*) som beräknas av ICP visade på att resultatet från de olika körningarna var identiskt. För JR-MPC gjordes en visuell granskning av resultatet och någon skillnad kunde inte identifieras.

D.6 Diskussion

JR-MPC funkade väldigt dåligt för den typ av data som vi hade. På grund av att algoritmen är väldigt komplicerad och inte ger något delresultat är det svårt att göra några ändringar för att uppnå bättre resultat.

3DCopy använder sig av *Max Iterations*, *Transformation Epsilon* samt *Max correspondence distance*. Då PCLs standardvärden för *RANSAC outlier rejection threshold* samt *Euclidean fitness epsilon* funkade bra för gruppen användes dessa och de kan inte ändras i 3DCopy. Det är dock väldigt lätt att lägga in stöd för det om programmet skulle vidareutvecklas.

D.6.1 Metod

Litteraturstudien som genomfördes som förstudie hade kunnat vara större. Avsaknaden av relevanta källor är något som märkts vid flera tillfällen, både under projektet samt denna rapportens framtagande. Man bör dock inte underskatta utvecklingsarbete i utbildningssyfte som var litteraturstudiens primära mål. Då de källor som hittades inte hade mycket överlappande information kunde få jämförelser göras och Evangelidis et al. [11] fick stå för det mesta av faktan. Dock är det ett publicerat forskningsdokument som funnits tillgängligt under längre tid och som presenterar en bra metod och ett väl dokumenterat tekniskt resultat.

Experimentet gav både bra och identiskt resultat vid varje köring. Det som saknas är en tidsjämförelse men eftersom att resultatet inte är så bra som önskat vid registrering av 36 punktmoln lades tillgängliga resurser på att förbättra resultatet mer än att förkorta körtiden. Efter att experimentet var utfört skapades dock ett filter som filtrerar bort redundant information som både ger bättre resultat, eftersom äldre punktmoln inte får mer och mer vikt under registreringens gång, samt reducerade körtiden signifikant. På grund av tidsbrist kunde dock experimentet inte göras om.

D.6.2 Resultat

En ordentlig jämförelse mellan algoritmerna som fanns tillgängliga försvarades av att JR-MPC är så matematiskt avancerad. För ordentlig analys uppskattar författaren att det krävs en kandidat i matematik. På grund av detta var det svårt att överhuvudtaget förstå och analysera JR-MPC på samma sätt som gjordes, förhållandevis enkelt, med ICP.

Experiment

Experimentet visar att ICP har kommit i närheten av ett bra resultat, se figur 18. Genom att granska delregistreringen i figur 19 framgår att de första 12 punktmolnen har gått bra att registrera. I delresultatet för 13 punktmoln syns det första punktmolnet som hamnat fel, uppskattningsvis 1-5 grader. Detta fel blir mer och mer tydligt ju fler punktmoln som registreras då alla punktmoln som kommer efter passar bättre med det punktmoln som sitter fel är de tidigare registrerade punktmolnen.

Första analysen av registreringsresultat resulterade i ett antagande om att ett punktmoln runt nr. 18-20 hade hamnat fel, vilket senare reviderades till resultatet som presenterades här. JR-MPC har dock misslyckats med registreringen som ledde till att projektgruppen helt övergav vidare försök att använda algoritmen. Då JR-MPC saknar delresultat eller tydliga parametrar att ändra är det svårt att analysera vad som orsakar beteendet som syns här.

D.7 Slutsatser

Nedan presenteras svaren på de forskningsfrågor som ställdes i avsnitt D.1.2.

D.7.1 Hur skapas ett enhetligt punktmoln från bilder tagna med en fast avståndskamera?

Vid digitalt återskapande av fysiska objekt skapas generellt flera punktmoln som ska registreras, dvs sys ihop. För att utföra denna registrering väljs en registreringsalgoritm som är lämpad för uppgiften, framförallt beroende på hur punktmolnen som samlats in ser ut. För ICP handlar det mest om att punktmolnen ska vara någorlunda rätt placerade i ett tänkt gemensamt koordinatsystem. När detta är uppfyllt kan punktmolnen registreras, och förutsatt att man har tillräckligt mycket data kan ett enhetligt punktmoln skapas som representerar det fysiska objektet som skannats.

D.7.2 Hur gör man för att välja algoritm och hur resonerade gruppen när de valde ICP?

För gruppens mål funkade ICP bäst men tidiga tester som gjordes gav skäl att undersöka andra algoritmer. Därmed är det svårt att dra någon konkret slutsats om vilken algoritm som är bäst. Det material som tagits fram är inte tillräckligt för att dra en konkret slutsats. Slutssatsen som Bellekens et al. [3] kommer fram till, att valet av algoritm är beroende av indata och tillämpning, är dock väldigt rimlig och med mer testdata till grund för deras slutsats är författaren av denna rapport benägen att hålla med.



E

Att bygga ett system i ROS

— Martin Lundberg —

E.1 Inledning

I ROS finns det många verktyg för att bygga upp ett system. ROS gör det enkelt att köra flera processer och kommunicera mellan dessa. Det hjälper även till att tydliggöra uppdelningar av ett program i separata moduler som enkelt kan återanvändas och vidareutvecklas separat utan att påverka de övriga modulerna.

E.1.1 Syfte

Syftet med den här rapporten är att utforska olika metoder för att bygga upp ett system i ROS. Rapporten ska även jämföra lösningarna i ROS med andra metoder för att ge fler perspektiv på hur ett system kan byggas upp.

E.1.2 Frågeställning

Följande frågeställningar ska behandlas och besvaras i denna rapport.

1. Hur kan en arkitektur implementeras i ROS?
2. Hur påverkade valet att använda ROS uppbyggnaden av systemet?

E.1.3 Avgränsningar

ROS är ett stort system med många möjliga implementationer av arkitekturen med i grunden samma funktionalitet. Denna rapport behandlar endast de delar av ROS som gruppen har kommit i kontakt med under implementationen av systemet som beskrivs i avsnitt 5.1 och utforskar inte någon annan funktionalitet i ROS.

E.2 Bakgrund

I det projekt som utvecklats och beskrivits i huvuddelen av denna rapport användes ROS redan i ett tidigt skede. En anledning till att just ROS valdes var för att projektets kund hade det som önskemål, dock såg projektmedlemmarna tidigt att ROS förde med sig många

fördelar. Användningen av ROS planerades redan i arkitekturen som är utformad helt med förutsättningen att ROS skulle användas för implementationen.

I denna rapport utredes hur ROS kan användas i implementationen av en arkitektur. Arkitekturen hade som mål att vara så modulär som möjligt då det skulle underlätta för projektets kund som vill kunna vidareutveckla olika delar av systemet i framtiden.

I ett sent skede av projektet behövde stora ändringar göras i arkitekturen vilket gjorde att ROS uteslöts helt. Detta gav ytterligare ett perspektiv att utgå från i den här utredningen.

E.3 Teori

I detta kapitel presenteras den teori som ligger till grund för rapporten.

E.3.1 ROS

ROS står för *Robot Operating System* men är inte ett fullt operativsystem i den traditionella meningen. Istället är det ett system som körs i ett vanligt operativsystem och som ger användaren möjlighet att på ett enkelt sätt köra fler processer och kommunicera mellan dessa [43]. En process som körs i ROS kallas för en nod och ROS tillhandahåller verktyg för att styra dessa noder. Det är ofta en bra tumregel att tänka att en nod motsvarar en modul i arkitekturen, men en modul skulle även kunna bestå av ett flertal noder.

ROS har även verktyg för att på ett enkelt och strukturerat sätt kommunicera mellan noder genom meddelanden (eng. messages). Dessa meddelanden är strikt typade och kan skickas mellan noder på två olika sätt:

1. Genom att en nod publicerar på en namngiven kanal (eng. *topic*) som en annan nod kan lyssna på.
2. En nod kan tillhandahålla en service som en annan nod kan kalla på för att utföra något arbete och få tillbaka resultatet i form av ett meddelande.

Den här rapporten ska utforska hur dessa verktyg kan användas för att bygga upp ett system.

E.3.2 Pipe and filter-modellen

En välkänd modell för att bygga upp mjukvarusystem är *pipe and filter*-modellen. Den beskrivs av Garlan och Shaw [16] som en modell där varje komponent i systemet har bestämda indata och utdata. Komponenterna behandlar indata och producerar utdata som kan skickas vidare till nästa steg i modellen.

I en traditionell *pipe and filter*-modell behandlas indata kontinuerligt så att utdata produceras innan hela strömmen av indata har nått fram, därav namnet filter [16]. I detta projekt användes dock ett specialfall av modellen där varje komponent kräver ett helt block av indata som sedan behandlas för att producera ett block av utdata. Det ska noteras att trots att varje modul endast tar kompletta block av indata snarare än strömmar så kan modulerna internt arbeta med strömmar av data och inkrementellt bygga upp resultatet.

E.4 Metod

I detta kapitel beskrivs metoden som användes för att ta fram systemets arkitektur och hur beslut fattades om systemets uppbyggnad. Hur arkitekturen togs fram ligger även som grund till att besvara frågeställningarna.

E.4.1 Arkitekturens framtagande

Arkitekturen för systemet har tagits fram med en iterativ process. Det första steget var att tänka igenom olika use cases för systemet och utifrån det ta fram de funktioner som programvaran måste innehålla. Detta resulterade i en systemanatomi som användes i det kommande arbetet.

Nästa steg var att se vilka funktioner som var relaterade till varandra och bestämma vilka noder systemet skulle bestå av och även hur dessa skulle kommunicera med varandra. Det var tydligt att programmets funktionalitet kunde delas upp i tre distinkta moduler:

1. punktmolnshantering
2. meshgenerering
3. hantering av 3D-skrivare.

Utöver de delarna behövdes även ett sätt att styra processen. För detta ändamål infördes ett grafiskt användargränssnitt och även ett kommandoradsgränssnitt som skulle kunna initiera hela eller delar av processen.

Då projektet byggde vidare på ett system kallat TreeD från förra året för att styra hårdvaran behövde även det tidigare systemet tas i åtanke när arkitekturen togs fram. Fler alternativ för att kommunicera med det tidigare systemet utforskades men i slutändan beslutades det att en egen nod skulle kalla på det tidigare systemets kommandoradsgränssnitt för att på så vis slippa sätta sig in i tidigare kod för att lista ut vilka funktioner som behövde kallas.

Ett viktigt mål med arkitekturen var att systemet skulle vara modulärt. Tidigt i arbetet bestämdes det att en variant av *pipe and filter*-modellen skulle vara lämplig. Där skickas resultatet från ett steg vidare som indata till nästa steg. En stor fördel med denna modell är att den gör det möjligt att byta ut enskilda delar utan att göra om hela systemet [16]. Detta är något som uppfyller de krav på moduläritet och underhållbarhet som ställdes på systemet.

E.4.2 Systemet i ROS

För att bygga systemet i ROS togs ett första förslag på en arkitektur fram där de moduler som beskrevs ovan motsvarade varsin ROS-nod. För att kommunicera med varandra skulle noderna använda sig av kanaler för att ta emot kommandon och skicka resultatet av utförda beräkningar. Att använda kanaler har fördelen att det är ett enkelt och snabbt sätt att komma igång med kommunikation mellan noder. Det var dock ett naivt sätt att implementera systemet i det här fallet då en kanalbaserad lösning gör väldigt lite för att upprätthålla den tänkta arkitekturen. Eftersom vilken nod som helst kan lyssna på en kanal blir det svårare att se kommunikationsvägarna mellan olika noder och därför blir det även svårare för en utomstående att förstå systemets uppbyggnad och vidareutveckla systemet.

För att göra systemet tydligare togs en ny arkitektur fram som istället var baserad på service-anrop. I denna arkitektur tillhandahåller modulerna en service som en annan nod kan kalla på för att utföra ett specifikt arbete. När arbetet är utfört returneras resultatet till noden som utförde service-anropet. Detta system gör det mycket tydligare vad de olika noderna gör och hur de kommunicerar med varandra och det är den slutgiltiga arkitekturen i ROS som togs fram för projektet.

E.4.3 Systemet utan ROS

Då projektets förutsättningar ändrades långt in i utvecklingsfasen behövde en ny arkitektur tas fram. Den nya arkitekturen behövde inte kommunicera med hårdvaran på något sätt utan skulle endast läsa in punktmoln från filer på disk. För att underlättा utvecklingsarbetet och för att det inte längre var lika aktuellt att använda ROS bestämdes det att den nya arkitekturen inte skulle vara baserad på ROS. Systemets uppbyggnad på modulnivå behölls dock så

lik den tidigare arkitekturen som möjligt där den största skillnaden är att någon koppling till TreeD inte längre fanns.

För att göra det fortsatta arbetet så enkelt som möjligt då det var begränsat med tid kvar beslutades det att systemet skulle byggas upp med klasser i C++, där varje modul motsvarar en klass som tillhandahåller modulens funktionalitet.

E.4.4 Dokumentering av processen

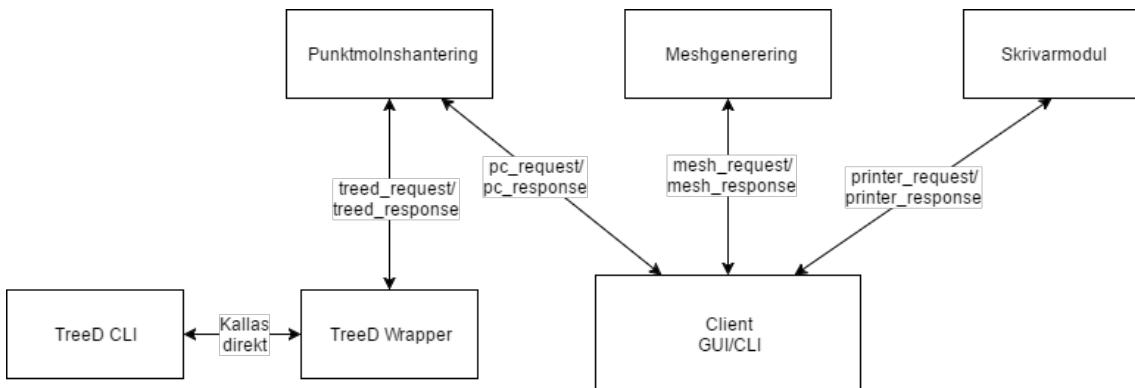
Under framtagandet av arkitekturen dokumenterades arbetet kontinuerligt i form av mötesprotokoll och diverse skisser och blockscheman som togs fram. Stora delar av arbetet med arkitekturen finns dokumenterat i projektets arkitekturbeskrivning där alla beslut som tagits om arkitekturen finns. Den innehåller även motiveringar till de beslut som fattats och, i relevanta fall, alternativa lösningar som diskuterades men som inte kom med i den slutgiltiga arkitekturen. Denna dokumentation används som underlag för att kunna svara på frågeställningarna i denna rapport.

E.5 Resultat

Sent i projektet stötte gruppen som tidigare nämnts på problem som inte gick att lösa inom en rimlig tidsram. Därför behövde projektets mål omdefinieras och på grund av det togs även en ny arkitektur fram, denna gång utan ROS. I detta kapitel visas båda de arkitekter som togs fram för projektet.

E.5.1 Arkitekturen i ROS

Den ROS-baserade arkitektur som togs fram för projektet visas i figur 20. Varje block i figuren motsvarar en nod i ROS och kommunikationsvägarna mellan noderna illustreras med pilar. All kommunikation mellan noderna utförs i form av service-anrop i ROS.



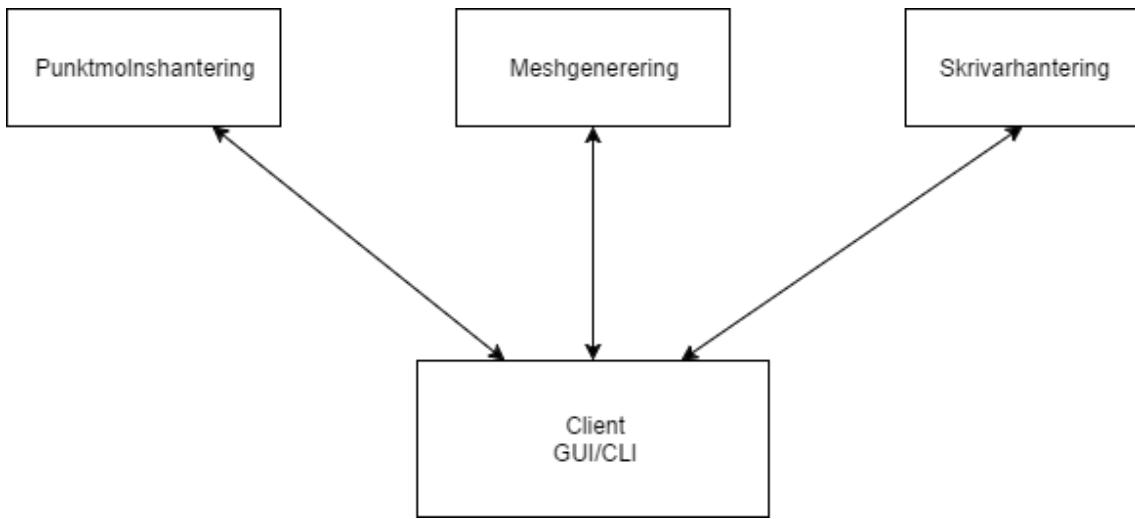
Figur 20: Systemets arkitektur i ROS.

Tidigt i projektet togs ett annat förslag på en arkitektur fram i ROS som är väsentligen samma som den i figur 20 men med den enda skillnaden att kommunikationen skulle skötas via kanaler istället för service-anrop. Detta skulle innebära att en egen kanal skapades för varje kommunikationsväg som visas i figuren. Detta arkitekturförslag är också intressant för denna rapport då det visar på en del skillnader mellan olika sätt att implementera samma system i ROS.

E.5.2 Arkitekturen utan ROS

Den nya arkitekturen som togs fram utan ROS är uppbyggd enligt samma modell i grunden. Dock är den uppbyggd helt objektorienterat i C++ med hjälp av klasser som tillhandahåller

den funktionalitet som tidigare låg i ROS-noder. Den nya arkitekturen använder inte heller det tidigare projektet TreeD så wrapern för den mjukvaran är borttagen. I detta system läses punktmolnsdata in från filer av klienten som sedan använder funktionaliteten i klasserna för att behandla punktmolnen. Klassindelningen och kopplingen mellan dem beskrivs i figur 21.



Figur 21: Systemets arkitektur utan ROS.

E.6 Diskussion

Resultatet av arbetet var två olika arkitekturen och implementationer av i grunden samma arkitekturidé. Detta ger möjlighet att jämföra ett system byggt i ROS med i grunden samma system byggt utan ROS.

Under arbetet med att ta fram och implementera dessa arkitekturen framkom en del fördelar och nackdelar med båda alternativen. Dessa diskuteras i detta avsnitt.

Det ROS-baserade systemet implementerades med service-anrop som beskrivs i kapitel E.5. Detta sätt att kommunicera gör det tydligt hur en nod kommunicerar med andra noder och hur systemet hänger ihop. Det undviker även att programflödet blir rörigt eftersom varje modul har en avgränsad uppgift och inte går utanför sina avgränsningar genom att till exempel kalla på funktioner i andra modular eller skicka data via andra kommunikationsvägar. Sådan kod blir väldigt svår att förstå och underhålla.

Dock märktes det senare under utvecklingsarbetet att den service-baserade strukturen gjorde systemet stelt och det resulterade i mycket mer arbete för att till exempel kunna kommunicera delresultat från en nod som tar väldigt lång tid att slutföra sitt arbete. För att kunna se delresultat från en nod som körs genom ett service-anrop skulle noden behöva publicera delresultaten på en kanal som en annan nod kan lyssna på. Detta gör dock att systemets modell bryts vilket gör att många av de fördelar som finns med att använda service-anrop går förlorade.

Det klassbaserade systemet däremot har mer flexibilitet i hur det implementeras. Det krävs dock lite tanke för att bibehålla modulärheten i en sådan implementation. För att göra ett sådant system modulärt är det viktigt att se till att klasserna kan användas helt fristående från det övriga systemet.

E.6.1 Portabilitet

En viktig fördel med ROS är då man vill använda systemet tillsammans med andra ROS-baserade system. Eftersom kommunikation inom ROS är tydligt definierad kan man skriva

en egen ROS-nod som använder sig av andra existerande noder och på så vis enkelt använda och bygga vidare på existerande system. Det kräver dock att det existerande systemet antingen är utförligt dokumenterat eller har en så pass tydlig struktur att det går att förstå hur man ska kommunicera med noderna.

Den klassbaserade implementationen kan dock ses som mer portabel då den inte ska integreras med andra ROS-system. Klasserna som systemet är uppbyggt av kan importeras och användas i ett annat projekt oavsett om det använder ROS eller inte.

E.6.2 Att gå från ROS till klasser

En stor fördel med att systemet byggs upp så modulärt redan från början var att det var relativt smärfritt att gå från den ROS-baserade arkitekturen till den klassbaserade. Mycket av den kod som hade skrivits kunde återanvändas och det var inga problem att implementera i grunden samma arkitektur både med och utan ROS.

Om det ROS-baserade systemet istället hade implementerats med kommunikation genom kanaler hade det varit svårare att separera modulerna och bryta ut funktionaliteten då modulerna hade haft en högre grad av koppling mellan sig. All kommunikation i det ROS-baserade systemet sköttes via service-anrop vilka kan liknas väldigt mycket vid klassiska funktionsanrop där någon indata ges och något resultat returneras när funktionen är klar. Detta gjorde det relativt enkelt att gå från ROS-systemet till det klassbaserade systemet.

E.6.3 Hållbarhet

Då olika ROS-noder körs som olika processer i operativsystemet och därför kan köras parallellt har ett ROS-baserat system en teoretisk möjlighet att vara mer energieffektivt än det klassbaserade system som implementerades i projektet (detta under förutsättning att stora beräkningar inte paralleliseras genom exempelvis flertrådad köring). I detta projekt skulle det dock ha minimal inverkan då varje större steg i processen var beroende av resultatet i det tidigare steget och modulerna kunde därför väldigt sällan köras parallellt.

E.7 Slutsatser

I denna rapport har några olika alternativa arkitekturen för att uppnå samma funktionalitet utvärderats och jämförts. Arkitekturen ger ett par förslag på hur ett system kan byggas upp och implementeras i ROS vilket besvarar frågeställning 1 i avsnitt E.1.2. Det finns så klart fler sätt att implementera systemet.

Ett ROS-system byggt helt på service-anrop ger god separation mellan noderna och definierar tydligt hur kommunikation ska ske. Det förhindrar även att snabba fullösningar implementeras där data skickas utanför den fastställda strukturen för att underlätta utvecklingen på kort sikt men gör systemet svårare att underhålla på lång sikt. Ett helt service-baserat system är dock stelt och det är svårt att lägga till ytterligare kommunikationsvägar i efterhand. Det är därför tydligt att en balans mellan service- och kanalbaserade system måste hittas då ett system designas.

I arbetet blev det tydligt att ett system som är modulärt uppbyggt och där mycket eftertanke lagts på systemets uppbyggnad redan från början har stora fördelar jämfört med ett system som har en tätare koppling mellan modulerna. I detta projekt specifikt gjorde moduläriteten så att beroendet av ROS kunde tas bort utan större förändringar i strukturen.

För att svara på frågeställning 2 kan det utnyttjas att i stort sett samma system implementerades både i ROS och utan ROS. Då arkitekturen som implementerades i ROS gick att översätta så gott som rakt av till ett system där varje modul istället motsvarade en klass i C++ blev det tydligt att användandet av ROS hade en väldigt liten påverkan på hur systemet byggdes upp på modulnivå. Användandet av ROS hade dock större påverkan i implementationsstadiet då arkitekturen i ROS använde meddelanden och utnyttjade de redan fastställda

kommunikationssätten som finns i ROS. I den rena C++-implementationen krävdes istället att den enskilda programmeraren satte upp ett tydligt och lättanvänt gränssnitt för klassen med vettiga metoder och parametrar.



F

Verktyg som är lämpliga för att skriva stora dokument

— Hannes Tuhkala —

F.1 Inledning

Att skapa bra och stiliga dokument är viktigt för att det ska se professionellt ut. I den här rapporten kommer en undersökning som är gjord av Hannes Tuhkala om vilka fördelar och nackdelar som L^AT_EX och Word har för att skriva stora dokument. Rapporten tar också upp vilka verktyg som lämpar sig för att skriva stora dokument.

F.1.1 Syfte

Syftet med den här rapporten är att ta reda på vilka för- respektive nackdelar som finns med att använda L^AT_EX och Word för att framställa dokument. Syftet är också att undersöka vilka asynkrona eller synkrona verktyg som är lämpliga att använda för att skriva stora dokument samt vilka erfarenheter projektgruppen kan ta med sig ifrån projektet gällande dokumenthantering.

F.1.2 Frågeställning

De frågeställningar som rapporten behandlar är:

1. Vilka fördelar och nackdelar finns det med att använda L^AT_EX eller Word för att skriva dokument?
2. Vilka verktyg lämpar sig för att skriva större dokument?
3. Vilka erfarenheter kan tas med ifrån det här projektet gällande framställning av större dokument?

F.1.3 Definitioner och förkortningar

Följande definitioner och förkortningar används på flertalet ställen i den här rapporten:

- Ett större dokument - Definieras här som ett dokument som två eller flera personer arbetar på och är åtminstone sju sidor långt.

- Synkront verktyg - Ett program eller en tjänst som flera personer kan skriva samtidigt på utan att någon behöver vänta på att någon annan skrivit färdigt.
- Asynkront verktyg - Ett program eller en tjänst som en person skriver text på och skickar det till andra personer efteråt.
- SVN - Apache Subversion
- Word - Microsoft Word
- IDE - Integrated Development Environment
- WYSIWYG-editor - WYSIWYG står för (What You See Is What You Get) och med en WYSIWYG-editor kan innehåll ändras i ett program, oftast text, där programmet sedan visar hur det slutgiltiga dokumentet kommer att se ut.
- Separata filer - Se avsnitt F.3.5.
- Gemensam fil - Se avsnitt F.3.5.
- Linter - En lint [32] är ett program som letar efter misstänkta fel i kod, som leder till att det inte kompilerar. Det letar efter fel i programmeringskod som är skrivet i C. En mer generell benämning för detta, oavsett vilket programmeringsspråk som används, är en linter.

F.1.4 Avgränsningar

Denna rapport kommer endast att ta upp de två vanligaste typsättningssystemen, \LaTeX och Microsoft Word eftersom de är de mest använda och kända. Det finns liknande system som dessa, till exempel OpenOffice eller LibreOffice och de kommer inte att behandlas i rapporten för att de inte är lika välhanvända och kända. Rapporten kommer också att ta upp olika asynkrona verktyg som Git, SVN och Mercurial samt synkrona verktyg som Google Docs, ShareLaTeX och Overleaf. Dessa valdes också för att de är de vanligaste tjänsterna och mest använda.

F.2 Bakgrund

Gruppen har arbetat med olika verktyg för att framställa större dokument för projektet. De verktyg som gruppen har använt sig av är \LaTeX för alla större dokument. Gruppen har också använt både asynkrona och synkrona verktyg när dessa dokument. För alla större dokument utom kandidatrapperten användes ett synkront verktyg, Overleaf. För kandidatrapperten använde gruppen sig av ett asynkront verktyg, Git. Den här studien har genomförts för att se vilket av de olika asynkrona och synkrona verktygen som kan vara bättre att använda sig av för större dokument. Flera andra verktyg som andra grupper kan ha använt har också tagits upp för att se hur de framställer sina dokument.

F.3 Teori

Nedan visas information om det som kommer att behandlas i rapporten. Det ger lite teori om bland annat vad \LaTeX , Git, SVN och Mercurial är för något.

F.3.1 L^AT_EX

L^AT_EX [1][24] är ett typsättningssystem som är baserat på TeX och skapades för att göra det enklare att skriva böcker och artiklar. Till skillnad från till exempel Microsoft Word, programmas dokumentet, istället för att skriva formaterad text. L^AT_EX bygger på att den som skriver inte ska formatera texten, utan ska bara skriva det den vill få sagt, utan att behöva bry sig om hur dokumentet ska se ut. Det är därför väldigt enkelt att få stiliga dokument i L^AT_EX genom att endast specificera dokumentets struktur, istället för att krångla med hur dokumentet ska se ut.

F.3.2 Git

Git [5] är ett versionshanteringsprogram som skapades av Linus Torvalds för att hantera källkoden till Linuxkärnan. Git är ett distribuerat versionshanteringssystem. Det innebär att alla som använder ett Git repository har tillgång till allt som det innehåller. Man behöver således inte kontakta en centraliserad server varje gång en ändring sker. Alla ändringar som skett sparas. Det finns alltså en komplett historik över allt som ändrats, tagits bort och lagts till. Det går också att gå tillbaka till tidigare versioner av dokument och källkod.

F.3.3 Apache Subversion

Subversion (SVN) [2] är också ett versionshanteringsprogram för mjukvara, likt Git, förutom att SVN är ett centraliserat versionshanteringssystem.

F.3.4 Mercurial

Ett annat versionshanteringsprogram är Mercurial. Det är ett distribuerat versionshanteringsprogram som Git [34].

F.3.5 Kandidatrapport

Kandidatrapporten är skriven i L^AT_EX och använder mallen för avhandlingar skapad av Ola Leifler [27]. Mallen är upplagd på sådant sätt att det finns separata filer för varje del av rapporten istället för en enstaka stor fil. Det innebär att introduktion, bakgrund, metod, de individuella bidragen samt de andra delarna i rapporten är separata filer.

F.4 Metod

För att kunna besvara de olika frågeställningarna har dem delats upp i tre olika sektioner för att beskriva hur det gick tillväga att besvara dem.

F.4.1 Första frågeställningen

För den första frågeställningen gjordes en litteraturstudie för att samla information om vilka för- och nackdelar som finns med att använda L^AT_EX eller Word för att skriva dokument. Detta för att se hur tidigare arbeten ställer sig till de två olika typsättningssystemen. Litteraturstudien gjordes genom att leta efter relevant litteratur i databaserna: Linköpings universitets biblioteksdatabas [4], Google Scholar [18] och på DiVA [9]. De söktermer som används för att hitta relevant litteratur var *Latex*, *Microsoft Word* och *Document Preparation System*. Endast de artiklar som varit relevanta för frågeställningen har valts. De artiklar som har används är Knauff och Nejasmic [23], Loch et al. [30] samt Henrik Henrikssons bidrag i Haavisto et al. [19].

F.4.2 Andra frågeställningen

För min andra frågeställning gjordes en enkätundersökning som skickades ut till alla som läser kursen **TDDD96, Kandidatprojekt i programvaruutveckling**. I enkäten ställdes först dessa frågor:

- Vilken grupp är du i?
- Vilket verktyg använder ni när ni skriver de flesta större/viktiga dokument?

Beroende på hur de svarade på föregående fråga, "Vilket verktyg använder ni när ni skriver de flesta större/viktiga dokument?", får de olika frågor. Om de valde alternativ 1 (Asynkrona verktyg som Git, SVN och liknande) fick de följande frågor:

- Vad för slags versionshanterare använder ni er av?
- Hur nöjd är du med hur ni versionshanterar era dokument?
- Om du fick ändra vilket asynkront versionshanteringsverktyg ni använde, vilket hade du då valt?
- Skulle du vilja byta hantering av dokument till synkrona verktyg som Google Docs, ShareLaTeX och liknande?
- Om du svarade ja, vilket verktyg skulle du då vilja använda?

Om de valde alternativ 2 (Synkrona verktyg som Google Docs, ShareLaTeX och liknande) fick de besvara dessa frågor istället:

- Vilken webbsida/tjänst använder ni er av för större/viktiga dokument?
- Hur nöjd är du med att använda en tjänst som denna för hantering av dokument?
- Om du fick byta den tjänst ni använde, vilken hade du då valt?
- Skulle du vilja byta hantering av dokument till asynkrona verktyg som Git och SVN?
- Om du svarade ja, vilket verktyg skulle du då vilja använda?

Till sist frågades också detta oberoende av vilka alternativ som valts.

- Vilka verktyg som har listats här tycker du är bra för större dokument?

F.4.3 Tredje frågeställningen

För min tredje frågeställning gjordes också en enkätundersökning, men enbart för medlemmarna i projektgruppen. Där ställdes dessa frågor:

- Överlag hur tycker du att hanteringen av större dokument har gått i gruppen?
- Vilket verktyg tycker du har varit bäst att använda?
- Varför tycker du det?
- För kandidatrapporten, tycker du att det var bättre med separata filer än en gemensam fil?

F.5 Resultat

Resultatet från litteraturstudien och de två enkäter som beskrevs i metoden visas nedan.

F.5.1 Litteraturstudie

Knauff och Nejasmic [23] visar med hjälp av ett mjukvaruanvändbarhetstest hur effektivt det är att använda sig av \LaTeX jämfört med Microsoft Word för vetenskapliga texter. Det användbarhetstestet gick ut på var att 40 olika deltagare skulle skriva olika texter. Det författarna kom fram till var att \LaTeX -användare var längsammare än Word-användare, skrev mindre text och gjorde flera olika sorters fel. Dessa fel inkluderar grammatik, formattering och typsättning. Det visar sig att \LaTeX -användare tycker om att använda \LaTeX mer än vad Word-användare tycker om att använda deras program. Författarna kom också fram till slutsatsen att erfarna \LaTeX -användare kan drabbas av produktivitetsförlust när de skriver dokument.

Loch et al. [30] undersöker om Word är ett lämpligt verktyg för studenter att typsätta matematiska formler och andra sorters matematiska texter. Detta som ett alternativ för att använda sig av \LaTeX . Författarna kommer fram till att nyare versioner av Word kan användas för att skapa matematiska texter med hög kvalitet. Att det inte heller är någon större inlärningskurva för att lära sig Word är också något som författarna kom fram till.

Henrik Henrikssons bidrag i Haavisto et al. [19] tittar på vilka problem som nybörjare i \LaTeX stöter på och hur det går att förhindra sådana problem i framtida projekt. Han kommer fram till att det är mycket konsistensproblem i ett \LaTeX dokument. Många skriver text olika och det finns ingen standard. Ett par förslag som Henrik kommer med på hur dessa problem kan minskas eller elimineras är att använda en linter eller att sätta upp en standard i början av projektet som sedan följs av projektgruppen.

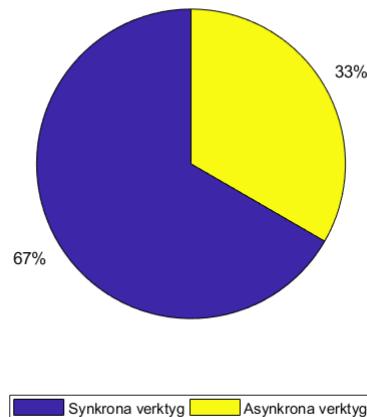
F.5.2 Enkät 1

Resultaten från den första enkäten, enkäten om hur det är att använda sig av synkrona respektive asynkrona verktyg när större dokument behöver framställas visas nedan. Om mer detaljerade resultat önskas, se bilaga H.4.

Totalt sett var det 36 av 98 personer som svarade på enkäten. Det ger ett deltagande på 36,7 %.

Vilket verktyg använder ni när ni skriver de flesta större/viktiga dokument?

I figur 22 ses att en tredjedel av grupperna använder sig av ett asynkront verktyg, alltså Git, SVN och liknande. Resterande grupper använder sig av ett synkront verktyg som ShareLaTeX, Google Docs och liknande.



Figur 22: Visar resultatet från vilka verktyg som grupperna använder för de flesta större dokument.

De som valde alternativ 1 (Asynkrona verktyg som Git, SVN och liknande) visas nedan. Totalt sett var det en tredjedel av grupperna som använde detta varav 15 personer som svarade.

Vad för slags versionshanterare använder ni er av?

Alla grupper som använder sig av ett asynkront verktyg använder sig av Git.

Hur nöjd är du med hur ni versionshanterar era dokument?

Från en skala från 1 till 5 där 1 är inte nöjd och 5 är nöjd, är medelvärdet $4,466 \approx 4,5$.

Om du fick ändra vilket asynkront versionshanteringsverktyg ni använder, vilket hade du då valt?

Ingen av deltagarna vill byta det asynkrona verktyg de använder till ett annat, utan de vill fortsätta använda Git.

Skulle du vilja byta hantering av dokument till synkrona verktyg som Google Docs, ShareLaTeX och liknande?

40 % av deltagarna vill byta till ett synkront verktyg istället för det asynkrona verktyg de använder nu. Resterande 60 % vill fortfarande använda sig av Git.

Om du svarade ja, vilket verktyg skulle du då vilja använda?

Av de som vill byta till synkront verktyg skulle två tredjedelar vilja byta till ShareLaTeX och en tredjedel byta till Google Docs.

De som valde alternativ 2 (Synkrona verktyg som Google Docs, ShareLaTeX och liknande) visas nedan. Totalt sett var det två tredjedelar av grupperna som använde detta varav 21 personer som svarade.

Vilken webbsida/tjänst använder ni er av för större/viktiga dokument?

3 av grupperna som använder ett synkront verktyg använder ShareLaTeX. Lika många grupper använder sig istället av Overleaf. En grupp använder sig av Google Docs och en annan grupp använder sig av Office 365 Word.

Hur nöjd är du med att använda en tjänst som denna för hantering av dokument?

Från en skala från 1 till 5 där 1 är inte nöjd och fem är nöjd, är medelvärdet på hur nöjda de är 3,8.

Om du fick byta den tjänst ni använder, vilken hade du då valt?

13 av de som fick frågan ville inte byta den synkrona tjänsten som de använder. Två vill använda sig av Google Docs istället. En person vardera ville byta tjänst till Overleaf och ShareLaTeX. En annan ville ha en kombination av en "online-tjänst" och Git. Två vill byta till L^AT_EX och Git.

Skulle du vilja byta hantering av dokument till asynkrona verktyg som Git och SVN?

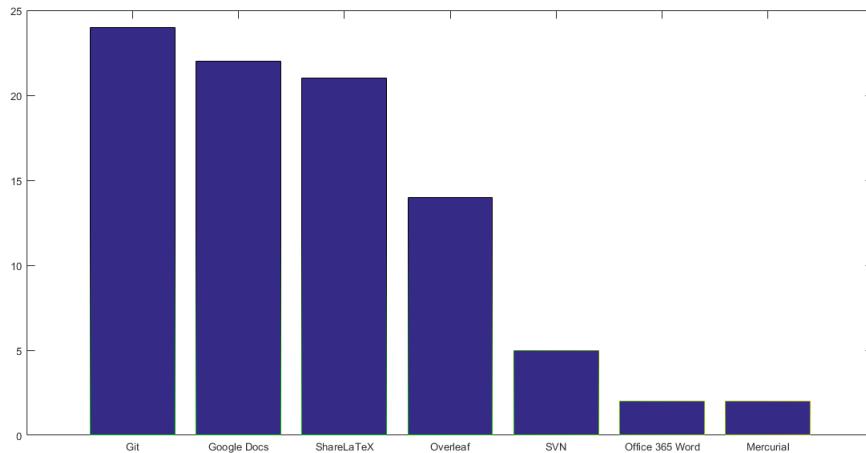
15 deltagare vill inte byta till ett asynkront verktyg medan sex personer ville byta verktyg.

Om du svarade ja, vilket verktyg skulle du då vilja använda? De som ville byta till ett asynkront verktyg vill använda sig av Git istället.

Den sista frågan som ställdes till alla deltagare i undersökningen oberoende av vad de tidigare valt:

Vilka verktyg som har listats här tycker du är bra för större dokument?

Figur 23 visar resultatet för frågan. På x-led finns de olika alternativen som fanns att välja och y-led visar hur många som valt det alternativet. Från figuren ses att Git, Google Docs och ShareLaTeX är det verktyg som de flesta har valt.



Figur 23: Visar resultatet från vilka verktyg som deltagarna tycker är bra för större dokument.

F.5.3 Enkät 2

Resultaten från den andra enkäten, den om vilka erfarenheter gruppen kan ta med sig ifrån det här projektet visas nedan. Om mer detaljerade resultat önskas, se bilaga H.5. Totalt sett var det 6 personer som svarade på enkäten.

Överlag hur tycker du att hanteringen av större dokument har gått i gruppen?

Fem av sex personer tycker att hanteringen av större dokument har gått bra, och en av dem tycker att det gått väldigt bra. En person tycker att det gått helt okej men kunde ha varit bättre.

Vilket verktyg tycker du har varit bäst att använda?

En person har ingen åsikt på frågan och en annan person tycker att båda verktygen varit lika bra att använda för större dokument. Fyra personer tycker att kombinationen \LaTeX och Git var det bästa att använda.

Varför tycker du det?

Personen som inte har någon åsikt i frågan och den som tyckte att båda alternativen var bra eftersom båda verktygen hade sina egna problem. \LaTeX och Git fungerade smidigare, hade versionshantering, historik och vem som skrivit vad, är de bra delarna av de som tyckte att \LaTeX och Git var bättre att använda.

För kandidatrapporten, tycker du att det var bättre med separata filer än en gemensam fil?

Alla deltagare i enkäten tyckte att separata filer var bättre att använda än en stor gemensam fil för kandidatrapporten.

F.6 Diskussion

Nedan finns en diskussion om resultaten som fåtts. De är indelade efter frågeställningarna.

F.6.1 Vilka fördelar och nackdelar finns det med att använda \LaTeX eller Word för att skriva dokument?

Att \LaTeX -användare var längsammare att skriva dokument än Word-användare tycker jag inte är jättemärktligt, eftersom att de som använder \LaTeX skriver mer än de som skriver Word. Detta för att \LaTeX -användarna måste skriva kommandon för att åstadkomma en viss sak medan Word-användare behöver bara använda sig av WYSIWYG-editor som finns inbyggd i Word där användaren klickar på knappar och drar tabeller i dokumentet. Word-användare använder också ett anpassat program för att skriva dokument, likt en IDE för programmering. \LaTeX har inte lika många anpassade editorer som Word, utan där kompileras dokumenten istället och texten skrivs i någon annan editor som vanligtvis inte har stöd för \LaTeX kommandon.

Från resultatet kan det också ses att inlärningskurvan för Word inte är väldigt stor och kräver inte mycket av användaren för att förstå och använda programmet. Det håller jag med om eftersom användaren behöver inte lära sig kommandon, hur kompilerar går till och hur en \LaTeX -miljö sätts upp. Microsoft Word är en WYSIWYG-editor, där endast texten som användaren vill få fram skrivs.

Det Henrik Henriksson tog upp i sitt bidrag om att ha en standardisering på \LaTeX dokument skulle ha gjort att våra dokument såg likadant ut överallt. Det hade varit bättre, för till exempel när gruppen använde Overleaf var det några som använde mellanslag för indentering och andra som använde tabbar för det. För en del av medlemmarna i gruppen var det första gången de använde \LaTeX och det hade kunnat vara lättare för dem om en standardisering fanns och instruktioner för hur \LaTeX fungerar. Att använda en linter hade också hjälpt att hitta fel i dokumentet fortare än att behöva kompilera och få kompileringsfel. Overleaf hade ingen linter vilket hade gjort det svårt att införa det där, men det finns som sagt alternativ till det.

F.6.2 Vilka verktyg lämpar sig för att skriva större dokument?

Att alla grupper som använder sig av asynkrona verktyg använder Git förvånar mig inte, just eftersom det är ett bra verktyg för att versionshantera dokument och källkod bland annat. Jag förväntade mig dock att någon grupp använde sig av SVN eller Mercurial. De som använt sig av Git har varit väldigt nöjda och gett på en skala från 1 till 5 medelvärdet 4,5 på hur nöjda de varit. Det tycker jag låter rimligt för att det är ett väldigt bra verktyg. Eftersom enkäten tog upp ett par snarlika alternativ som till exempel Git, SVN och Mercurial samt ShareLaTeX och Overleaf där funktionaliteten inte skiljer sig mycket åt kanske det hade varit bättre att bara fråga mer generellt om asynkrona verktyg är bättre att använda än synkrona verktyg.

För synkrona verktyg är medelvärdet på hur nöjda deltagarna var 3,8. Det skiljer sig med 0,7 jämfört med 4,5 som var medelvärdet hos deltagarna för de asynkrona verktygen. Det är också väldigt bra och jag tycker också det är bra att kunna ändra på dokument samtidigt i Overleaf och Google Docs. Majoriteten vill inte byta synkrona verktyg eller byta till ett asynkront. Det tror jag kan bero på att de inte testat att använda sig av ett asynkront verktyg tidigare för dokument.

På den sista frågan i resultatet finns det flera alternativ som ligger i topp: Git, Google Docs, ShareLaTeX och Overleaf. Ett av dem är asynkront och resten är synkrona. Jag har själv använt alla förutom ShareLaTeX fast jag tror inte det skiljer sig mycket med Overleaf, och jag tycker att alla är väldigt bra att använda för att skapa stora dokument. Om det fanns ett verktyg som kunde kombinera versionshantering med att skriva samtidigt i samma dokument tror jag det skulle ha varit bättre att använda. Detta för att kunna få allt i båda världar:

Versionshantering och historik samt att kunna skriva samtidigt. Hur det skulle fungera vet jag dock inte.

En del av alternativen jag hade till frågorna hade inte valts alls, och jag hade inget som frågade varför dessa inte valdes. Det kan ju ha varit för att personen inte visste vad verktyget var för något, till exempel för Mercurial. Det kan också vara för att de inte visste vad skillnaden var mellan verktygen, om de till exempel endast använt ett av dem. Det kan också ha varit för att de inte tyckt att det var något bra alternativ alls. Det är svårt att veta utan att ha frågat om det.

F.6.3 Vilka erfarenheter kan tas med ifrån det här projektet gällande framställning av större dokument?

Att majoriteten tycker att sättet gruppen arbetade med kandidatrapporten jämfört med de andra stora dokument är bättre håller jag med om. Det var enklare att ha versionshantering och historik, för att kunna se vad som ändrades eller lades till vid varje version och vem som utförde det. Man kan gå tillbaka till en tidigare version med ett enkelt kommando. Detta kunde inte göras när vi använde Overleaf. Fördelen med det var att alla kunde skriva samtidigt och ingen behövde installera någon speciell mjukvara för det. Det jag tycker var mindre bra med Git och L^AT_EX var att stycken oftast skrevs på en rad vilket ger många konflikter när sammanfogning av varandras delar sker. Om något ändrats, stort eller litet.

F.6.4 Metod

Litteraturstudien för frågeställning 1 kunde kanske ha gett ett annat eller mer utförligare svar om fler artiklar, bloggar och liknande hittats. Litteraturstudien som genomfördes kunde ha varit mer omfattande, men eftersom den här rapporten hade tidsbegränsningar gjordes inte en mer omfattande studie. Nu när det är få källor som används kan det ge en missrepresentativ bild av resultatet.

Enkätundersökningen som gjordes för den andra frågeställningen kunde ha gjorts mer utförligare, genom att ställa bättre frågor eller fråga om mer utförligare svar. Till exempel "Rangordna de olika alternativen från det du hade helst använt till det du helst inte velat använda." istället för att endast bara fråga om "Vilka alternativ hade du använt i framtiden för större dokument?". Om fler personer hade svarat på den första enkäten kunde det också ha gett en bättre bild på hur det ser ut. Om deltagarna testat de olika verktygen innan enkäten hade även det nog gett ett bättre resultat. Jag tror inte att de flesta som gjorde enkäten testat alla alternativ utan möjligtvis två stycken.

Urvalsgruppen är alla projektgrupper som läser kandidatprojektet och det kanske inte ger en representativ generell bild av arbete på större dokument än om urvalsgruppen innehållit andra yrken eller liknande. Vissa av alternativen kan förekomma mer inom företag än andra ställen.

Den andra enkätundersökningen kunde ha haft en till fråga "Vad var det som var dåligt med hanteringen av större dokument?". Som det ser ut nu kommer endast de bra sakerna med verktygen. Kunde också ha frågat om det finns något annat verktyg än de vi använde som kunde ha varit bättre. Att använda sig av en enkät för att ta reda på vad gruppmedlemmar tycker kan både vara bra och dåligt. Det dåliga är att det bara är mina frågor som kommer fram medan andra frågor kan komma upp om en intervju hade gjorts. Det blir lite skillnad på en enkät och en intervju. Det som är bra är att det går fort att göra och att det är enklare att sammanställa svaren.

F.6.5 Källkritik

Alla källor som har användts i litteraturstudien är publicerade källor. De första två källorna har citerats av andra arbeten. Den första källan jag använde, Knauff och Nejasnic [23], lät

bara testanvändarna skriva i 30 minuter. Det tycker jag är för lite tid för att säga att ett typsättningssystem är bättre än ett annat. Testanvändarna fick också bara skriva en liten del text och några matematiska ekvationer vilket inte är helt representativt i hur många arbetar med dokument. Ryan Schuetzler skriver om denna artikel i en bloggpost på sin webbsida [28]. Metoden Knauff använde för att ta reda på vilken av typsättningssystemen som är effektivast är därför inte den bästa. De andra två källorna verkar vara mer pålitliga.

De flesta källor som använts i teorin är tagna direkt från deras egna webbsidor för varje program. Programmen är också öppen källkod. Jag påstår att dessa källor är pålitliga.

F.7 Slutsatser

Nedan visas slutsatserna som kommit fram från resultatet för de tre frågeställningarna.

F.7.1 Vilka fördelar och nackdelar finns det med att använda \LaTeX eller Word för att skriva dokument?

Det som kommit fram genom litteraturstudien är att båda alternativen har fördelar och nackdelar samt att det inte skiljer sig mycket mellan dem. Word-användare skriver dokument snabbare än vad \LaTeX -användare gör. Inlärningskurvan är inte särskilt stor för att lära sig att använda Word. För att skriva större matematiska texter är det ingen skillnad mellan alternativen eftersom båda gör det med hög kvalitet. Att ha en standardisering och utbildning i \LaTeX kan hjälpa nybörjare.

F.7.2 Vilka verktyg lämpar sig för att skriva större dokument?

Det som kommit fram med hjälp av enkätundersökningen är att Git, Google Docs och Share-LaTeX kan vara bra verktyg att använda sig av när större dokument skrivs. Overleaf kan också vara ett bra verktyg att använda sig av. Mindre lämpliga verktyg att använda är SVN, Office 365 Word och Mercurial. Verktygen som användes i alla grupper var gruppmedlemarna nöjda med.

F.7.3 Vilka erfarenheter kan tas med ifrån det här projektet gällande framställning av större dokument?

De erfarenheter som kan tas med ifrån detta projekt är att \LaTeX och Git har varit det bästa verktyget att använda vid framställning av större dokument. \LaTeX och Overleaf har också varit bra men fördelarna med Git har övervägt det. Separation av dokumentet i flera mindre filer har också varit bättre än att ha en stor gemensam fil.



G

Kvalitetsarbete i praktiken

— Fredrik Wallström —

G.1 Inledning

De flesta företag ute på marknaden idag strävar mot att utveckla bättre produkter och tjänster. Detta leder till att kvalitetsarbetet i utvecklingen av dessa produkter och tjänster spelar en alltmer betydande roll. Kvalitetsarbetet är ett problem att effektivisera för att hinna med dagens utveckling och är generellt ett komplext ämne att behandla. Det står dock klart att det är en viktig komponent i utvecklingen av bättre produkter och tjänster.

G.1.1 Syfte

Syftet med denna rapport är att ta reda på vad det innebär att kvalitetssäkra ett mjukvarusystem och vilka metoder det finns för ändamålet. Syftet är också att se över hur kvalitetsarbetet används i praktiken och om det finns möjligheter att utveckla och effektivisera detta arbete. Detta undersöks eftersom det blir alltmer aktuellt på grund av den snabba utvecklingen av produkter och tjänster.

G.1.2 Frågeställning

De frågeställningar som denna rapport kommer ta upp är:

1. Vad är effekterna av att kvalitetssäkra mjukvaran för en produkt med den specifika metoden kodgranskning?
2. Kommer effekterna av en kodgranskningsprocess kompensera för den faktiska tidsåtgången?

G.1.3 Avgränsningar

Denna rapport kommer endast att undersöka vad den specifika metoden kodgranskning har för påverkan på kvaliteten hos ett mjukvarusystem. Det finns flera metoder för detta ändamål men dessa kommer alltså inte att undersökas i denna rapport eftersom just kodgranskning har använts under det relaterade kandidatprojektet. En annan avgränsning är att den enkät som genomfördes under experimentet endast skickades ut till 30 personer.

G.2 Bakgrund

Kodgranskning är en vanlig process för att öka kvaliteten på koden och för att dela kunskap bland medlemmar inom ett projekt.

Under detta kandidatprojekt i programvaruutveckling har vi arbetet med just kodgranskning som en del i att kvalitetssäkra vår produkt. Vi anser, utan att ha något konkret bevis, att kodgranskning har hjälpt oss att säkerställa kvaliteten på vår produkt. Därför är det intressant att se vilken påverkan en kodgranskningsprocess har på kvaliteten hos ett mjukvarusystem och vilka egentliga problem som lösas med denna process.

G.3 Teori

Nedan följer information om de ämnen som behandlas i rapporten. Informationen beskriver vad kvalitetssäkring innebär och vad kodgranskning är för något.

G.3.1 Kvalitetssäkring

Att genomföra en kvalitetssäkrande process innebär, enligt Feldman Stuart [14], att "*tillhålla en försäkran om att programvaruprodukten och processer i produktens livscykel överensstämmer med deras specifika krav och håller fast vid de uppsatta planerna*". Kvalitetssäkring är en process som genomförs under hela utvecklingen av ett system [14].

Det finns flera aktiviteter som kan ingå i en kvalitetssäkrande process beroende på projekt samt dess kvalitetsmål. Processen kan alltså bestämmas utifrån projektets syfte och mål. En aktivitet i de flesta kvalitetssprocesserna är testning. Att kontinuerligt testa en produkt leder till att buggar kan hittas och åtgärdas. En annan ofta vanlig aktivitet i en kvalitetssäkrande process är kodgranskning som beskrivs mer ingående i avsnitt G.3.2 [14].

G.3.2 Kodgranskning

Kodgranskningsprocessen definierades av Fagan [12] och innebär att "*granskarna som granskar en kods ändringar ska metodiskt följa checklistor för att sedan medverka i gruppmöten tillsammans med kodens författare och diskutera de förslagna ändringarna*". En kodgrankningsprocess syfte är enligt McIntosh et al. [47] att "*upptäcka fel och fixa dessa fel innan koden integreras med kodbasen*".

Fagans [12] kodgranskingsprocess har under åren utvecklats till en så kallas modern kodgranskingsprocess. Den moderna kodgranskingsprocessen behåller Fagans ursprungliga metod för processen med skillnaden att checklistorna och gruppmötena istället har digitalisering. Detta har antagligen blivit följden eftersom att även samhället har modernisering till att bli mer beroende av internet [47].

Att utföra en kodgranskingsprocess idag är enkelt. Det finns idag verktyg som underlättar en kodgranskingsprocess. Ett kodgranskingsverktyg hjälper till att få en enkel och snabb överblick över statusen i processen. Kodgranskning har dessutom integrerats i olika versionshanteringsverktyg för att underlätta processen [47].

För att kort sammanfatta en kodgranskingsprocess så innebär det, enligt McIntosh et al. [47], att en utvecklare föreslår en kodändring, denna kod integreras sedan med kodbasen om den godkänns av de utnämnda granskarna.

G.4 Metod

För att förstå effekterna av att genomföra en kodgranskingsprocess samt för att inse vad de egentliga kostnaderna är med att kvalitetssäkra en produkt har en litteraturstudie genomförts. Litteraturstudien bestod av att analysera och bearbeta olika vetenskapliga rapporter som var relevanta för ämnet ifråga. De utvalda rapporterna som användes i resultatet hade genomfört intressanta experiment och utredningar. De utvalda rapporterna som användes i

teorin gav bra information om vad både kvalitetssäkring och kodgranskning innebär. Nyckelorden kodgranskning och kvalitet användes vid sökningen efter relevanta rapporter.

För att undersöka dessa frågor med en koppling till praktiken så har även ett praktiskt experiment gjorts på studenter ifrån Linköpings universitet. Dessa studenter har genomfört ett projekt där fokus var att leverera ett färdigt system med hög kvalitet till kund. Experimentet bestod av att studenterna skulle svara på en enkät om hur det är att jobba med en kvalitetssäkrande process. Frågorna som studenterna fick svara på var följande:

- Hur anser du det är att jobba med en kvalitetssäkrande process, är det positivt eller negativt, om man tar den faktiska tidsåtgången i åtanke?
- Har du jobbat med den specifika metoden kodgranskning för att säkerställa kvaliteten på en produkt?
- Tycker du att kodgranskning hjälper att kvalitetssäkra en produkt?
- Anser du graden av deltagande män i kodgranskningen hjälper eller försämrar kvaliteten hos produkten? Blir det för komplicerat att anordna möten, tidsåtgången blir för stor, det tar för lång tid att få koden godkänd med mera.
- Har du några övriga tankar om att arbeta med en kvalitetssäkrande process eller med kodgranskning?

Detta experiment valdes eftersom att enbart utgå ifrån litteraturstudier inte ger en representativ bild av de relevanta kvalitetsfrågorna. Experimentets syfte var att se hur kvalitetsarbete fungerar i praktiken. Det är också intressant att se hur detta experiment skiljer sig ifrån resultatet av litteraturstudien. Det valdes studenter från Linköpings universitet eftersom jag själv studerar aktivt på detta universitet. Vilket gjorde det enkelt att nå ut med mitt experiment till studenter. Experimentet genomfördes även bara av studenter som har jobbat med en kvalitetssäkrande process under deras kandidatarbeten.

G.5 Resultat

Nedan följer resultatet ifrån den litteraturstudie som genomförts och ifrån den enkät som genomfördes på studenter ifrån Linköpings universitet.

G.5.1 Litteraturstudie

McIntosh et al. [33] visar med sin studie vilken påverkan en modern kodgranskningsprocess har på kvaliteten hos ett mjukvarusystem. McIntosh analyserar huruvida det finns en relation mellan kodgranskning och kvalitet genom att studera tre stycken projekt. Studierna går ut på att undersöka hur kvaliteten påverkas beroende på hur stor del av koden som har blivit kodgranskad och hur många granskare som deltagit i kodgranskningen. McIntosh et al. [33] kommer fram till att kodgranskning och kvalitet har en relation med varandra. Det visar sig att en dåligt granskad kod kan skapa mellan två till fem stycken defekter.

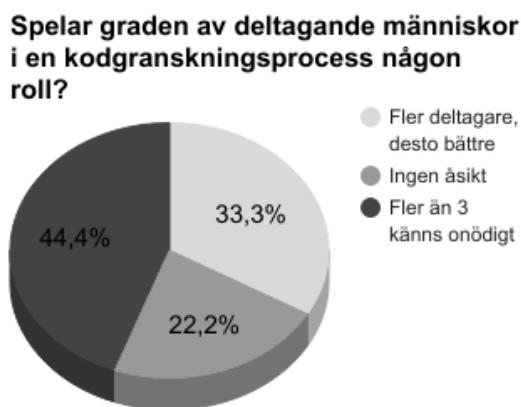
Beller et al. [3] visar med en studie vad de praktiska fördelarna är med en modern kodgranskningsprocess. Det visar sig att många av kommentarerna i en kodgranskning är onödiga, hela 7-35 % kasseras. Beller et al.'s [3] studie visar också att hela 10-22 % av de ändringar som gjorts, på grund av en kodgranskningsprocess, inte berodde på kommentarer ifrån kodgranskningen. Beller et al.'s [3] slutsats är att en homogen kodgranskning, det vill säga en likartad kodgranskning beroende på vem som utför den, leder till samma antal ändringar i koden oberoende på granskaren.

G.5.2 Enkät

De sammanfattade resultaten ifrån den genomförda enkäten presenteras nedan, för mer detaljerade resultat, se bilaga H.6.

- 100% anser att det är positivt att jobba med en kvalitetsäkrande process.
- 88% har jobbat med den specifika metoden kodgranskningen för att säkerställa kvaliteten hos en produkt.
- 100% anser att kodgranskning hjälper till att kvalitetssäkra en produkt.

Figur 24 visar resultatet från frågan om graden av deltagande människor i en kodgranskingsprocess hjälper eller försämrar kvaliteten hos en produkt.



Figur 24: Resultatet från om antalet deltagande människor i en kodgranskingsprocess spelar någon roll.

G.6 Diskussion

Resultatet av den genomförda enkäten visar att effekterna av en kodgranskingsprocess är positiva. 100 % av de tillfrågade anser att kodgranskning hjälper till att kvalitetssäkra en produkt. Det är endast få personer som inte använt sig av den specifika metoden kodgranskning för att kvalitetssäkra en produkt, utan istället använt sig av parprogrammering. Datat ifrån enkäten är inte tillräcklig och är en avgränsning i denna studie, för en mer representativ bild skulle det krävas större mängd data.

Det intressanta med resultatet ifrån enkäten är att 44,4 % av de som deltog anser att om fler än tre personer deltar i en kodgranskingsprocess finns det risk att tidsåtgången inte kompenseras för den faktiska vinsten med granskningen. Detta är också något som Beller et al. [3] kommer fram till i sin studie angående vilka problem som lösas med hjälp av kodgranskning. Beller anser att en homogen kodgranskning, det vill säga en likartad kodgranskning beroende på vem som utför den, leder till samma antalet ändringar i koden oberoende av vem som granskar. Det spelar alltså inte någon roll vem som granskar koden, den kommer heller inte blir bättre desto fler som granskar. Resultatet tyder alltså på att en kodgranskingsprocess inte bör innehålla fler deltagande personer än tre för att kvalitetssäkra produkten.

Samtidigt visar McIntosh et al. [33] att kodgranskingsprocesser med lågt deltagande uppskattningsvis innehåller upp till fem stycken defekter, vilket kan låta som en kontradiktion till Beller et al.'s [3] resultat. McIntosh specificerar dock inte att det finns ett optimalt antal personer som ska delta i en kodgranskingsprocess för att göra den så effektiv och

kvalitetssäkrande som möjligt. Den intressanta fråga som då ställs är om det finns ett optimalt antal deltagande personer i en kodgranskingsprocess? Det är en fråga som utifrån egna slutsatser inte finns något svar på. Det är en bred fråga och beror på hur stort projektet är och vad det finns för kvalitetsmål på produkten. Ju högre kvalitetsmål, desto fler personer bör delta i kodgranskingsprocesserna känns intuitivt som ett bra tillvägagångssätt. Denna studie visar dock att så inte är fallet, kodgranskningen riskerar att bli ineffektiv vilket således kan leda till en icke komplett produkt. Det här är en intressant frågeställning som kräver vidare efterforskning. Även här finns det såklart avgränsningar som bör tas med i åtanke. Dels innehåller inte enkäten tillräckligt med data och därtill skulle det även behövas göra en mer omfattande litteraturstudie för att få en representativ bild över situationen.

G.6.1 Källkritik

Det finns många artiklar som handlar om just kodgranskning och vad denna process har för inverkan på kvalitet. De två artiklar som användes i resultatet för denna studie valdes för att de experiment som de genomfört var relevanta för min studie.

McIntosh et al.'s [33] studie har i skrivandets stund citerats av 84 stycken andra publicerade artiklar. McIntosh har även en bred repertoar med publicerade artiklar och flera av dem handlar om just kodgranskning och dess påverkan av kvalitet.

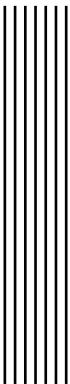
Bellers et al.'s [3] studie har i skrivandets stund citerats av 55 stycken publicerade artiklar. Beller har inte heller en lika bred repertoar i publicerade artiklar som McIntosh har, han har dock flera artiklar som blivit väl citerade som handlar om testning. Som nämnts tidigare är testning en stor del i att kvalitetssäkra en produkt. Jag tycker att Bellers och McIntosh artiklar anses pålitliga och har därför använts i denna studie.

De källor som användes till teorin stämmer inte överens med de källor som representerar studiens resultat. Anledningen till detta är att de artiklar som representerar resultatet i studien inte presenterar vad kvalitetssäkring och kodgranskning innebär på ett elementärt vis. Detta leder till att resultatet kan tyckas bli mindre trovärdigt. Jag ser det dock som att både McIntosh et al.'s [33] och Bellers et al.'s [3] artikel förutsätter att läsaren vet vad kvalitetssäkring och kodgranskning innebär och behöver således inte förklaras mer ingående. På grund av detta har jag valt att använda olika artiklar till teoridelen och resultatdelen.

G.7 Slutsatser

Utifrån den genomförda enkäten men framförallt utifrån den gjorda litteraturstudien visar denna rapport att kvalitet och kodgranskning har en stark relation. Att genomföra en kodgranskingsprocess visar sig tydligt ha positiva effekter på produktens kvalitet. För att svara på fråga 1 i avsnitt G.1.2 så tyder resultatet från denna studie på att effekterna av en kodgranskingsprocess är positiva ur kvalitetssynpunkt. McIntosh et al. [33] visar att en dåligt granskad kod, det vill säga kod som granskats av få personer, kan skapa mellan två till fem stycken defekter.

För att svara på fråga 2 i avsnitt G.1.2 så tyder resultatet från denna studie att kostnaden för en kodgranskingsprocess kompenserar för den faktiska tidsåtgången. Den genomförda enkäten visade att 100 % av deltagarna anser att kodgranskning hjälper till att kvalitetssäkra en produkt. Enkäten visar också att antalet deltagande män i kodgranskningen inte behöver överstiga tre stycken, detta kan leda till onödig tidsåtgång. Det här är även något som bekräftas av Bellers et al.'s [3] studie. Slutsatsen av detta är att effekterna av en kodgranskingsprocess kompenserar den faktiska tidsåtgången så länge antalet personer som granskas inte överstiger tre stycken.



Referenser

- [1] *An introduction to LaTeX*. <https://www.latex-project.org/about/>. Senast uppdaterad ej angiven. Hämtad 2017-05-08.
- [2] *Apache Subversion*. <https://subversion.apache.org/features.html>. Senast uppdaterad ej angiven. Hämtad 2017-05-08.
- [3] Moritz Beller, Alberto Bacchelli, Andy Zaidman, and Elmar Juergens. "Modern code reviews in open-source projects: Which problems do they fix?" In: *Proceedings of the 11th working conference on mining software repositories*. ACM. 2014, pp. 202–211.
- [4] *Biblioteket: Linköpings universitet*. <https://www.bibl.liu.se/?l=sv>. Hämtad 2017-05-29.
- [5] Scott Chacon and Ben Straub. *Pro Git, 2nd Edition*. Apress, 2014.
- [6] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. "Ieee recommended practice for software requirements specifications". In: Institute of Electrical and Electronics Engineers. 1998.
- [7] *Cura 3D Printing Slicing Software*. <https://ultimaker.com/en/products/cura-software>. Hämtad 2017-04-28.
- [8] *Dator- och bildskärmsarbete*. <https://www.av.se/inomhusmiljo/dator--och-bildskärmsarbete/>. Senast uppdaterad 2016-10-26.
- [9] *DiVA*. <http://www.diva-portal.org/>. Hämtad 2017-05-29.
- [10] *Downsampling a PointCloud using a VoxelGrid filter*. http://pointclouds.org/documentation/tutorials/voxel_grid.php. Hämtad 2017-05-02.
- [11] G.D. Evangelidis, D. Kounades-Bastian, R. Horaud, and Psarakis E.Z. "A generative model for the joint registration of multiple point sets". In: *European Conference on Computer Vision (ECCV)*. 2014.
- [12] Michael E Fagan. "Design and code inspections to reduce errors in program development". In: *IBM Systems Journal* 38.2/3 (1999), p. 258.
- [13] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer". In: *ACM SIGARCH Computer Architecture News*. Vol. 35. 2. ACM. 2007, pp. 13–23.
- [14] J Feldman Stuart. *Quality Assurance: Much More than Testing*. 2005.

- [15] *G-Code*. <https://en.wikipedia.org/wiki/G-code>. Senast uppdaterad 2017-04-25. Hämtad 2017-04-28.
- [16] David Garlan and Mary Shaw. "An introduction to software architecture". In: *Advances in software engineering and knowledge engineering* 1.3.4 (1993).
- [17] Joseph A Goguen and Charlotte Linde. "Techniques for requirements elicitation". In: *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*. IEEE. 1993, pp. 152–164.
- [18] *Google Scholar*. <https://scholar.google.se/>. Hämtad 2017-05-28.
- [19] Felix Haavisto, Henrik Henriksson, Niklas Häddy, Johan Jansson, Fabian Petersen, David Pop, Viktor Ringdahl, and Sara Svensson. "Modernisering av ett 3D-scanningssystem : Utmaningar och lärdomar av ett projekt". In: (2016). ISRN: LIU-IDA/LITH-EX-G-16/039-SE, Linköping University.
- [20] Ingrid Hylander. *Fokusgrupper som kvalitativ datainsamlingsmetod*. Linköping University Electronic Press, 1998.
- [21] *Inomhusmiljö*. <https://www.av.se/inomhusmiljo/>.
- [22] MariAnne Karlsson. *User requirements elicitation : a framework for the study of the relation between user and artefact*. Doktorsavhandlingar vid Chalmers tekniska högskola: Ny serie 1181. Göteborg : Chalmers tekniska högsk., 1996 ; 1996. ISBN: 9171973133.
- [23] Markus Knauff and Jelica Nejasmic. "An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development". In: *PLoS one* 9.12 (2014), e115069.
- [24] Stefan Kottwitz. *LaTeX beginner's guide*. Packt Publishing Ltd, 2011.
- [25] Megan Kreiger and Joshua M Pearce. "Environmental life cycle analysis of distributed three-dimensional printing and conventional manufacturing of polymer products". In: *ACS Sustainable Chemistry & Engineering* 1.12 (2013), pp. 1511–1519.
- [26] Patricia Lago, Sedef Akinli Koçak, Ivica Crnkovic, and Birgit Penzenstadler. "Framing sustainability as a property of software quality". In: *Communications of the ACM* 58.10 (2015), pp. 70–78.
- [27] *LaTeX class for LiU Thesis*. <https://gitlab.ida.liu.se/olale55/liuthesis>. Hämtad 2017-05-08.
- [28] *LaTeX vs Word (Again)*. <http://www.schuetzler.net/blog/latex-vs-word-again/>. Senast uppdaterad 2015-01-04. Hämtad 2017-05-08.
- [29] *Ljud och akustik*. <https://www.av.se/inomhusmiljo/ljud-och-akustik/>. Senast uppdaterad 2015-06-18.
- [30] Birgit Loch, Tim W Lowe, and Ben D Mestel. "Master's students' perceptions of Microsoft Word for mathematical typesetting". In: *Teaching mathematics and its applications* (2014), hru020.
- [31] Philipp Lombriser, Fabiano Dalpiaz, Garm Lucassen, and Sjaak Brinkkemper. "Gamified requirements engineering: model and experimentation". In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer. 2016, pp. 171–187.
- [32] *Man page for Lint*. <http://www.unix.com/man-page/FreeBSD/1/lint>. Senast uppdaterad: Ej angivet. Hämtad 2017-06-06.
- [33] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. "The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects". In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM. 2014, pp. 192–201.

- [34] *Mercurial source control management*. <https://www.mercurial-scm.org/about>. Senast uppdaterad 2017-05-02. Hämtad 2017-05-08.
- [35] *Olika typer av kontorslokaler*. <https://www.av.se/inomhusmiljo/lokaler-och-arbetsutrymme/lokalernas-storlek/olika-typer-av-kontorslokaler/>. Senast uppdaterad 2016-07-01.
- [36] *PCL: Filtering*. http://docs.pointclouds.org/1.7.0/group__filters.html. Senast uppdaterad 2013-12-15. Hämtad 2017-04-26.
- [37] *PCL: Home*. <http://pointclouds.org>. Hämtad 2017-05-05.
- [38] *PCL ICP documentation*. http://docs.pointclouds.org/trunk/classpcl_1_1_registration.html. Senast uppdaterad 2017-05-03. Hämtad 2017-05-18.
- [39] *PCL: Registrering*. http://pointclouds.org/documentation/tutorials/registration_api.php. Hämtad 2017-05-01.
- [40] *PCL: Trianguleringsalgoritm*. http://www.pointclouds.org/documentation/tutorials/greedy_projection.php. Hämtad 2017-04-27.
- [41] *PCL: Ytrekonstruktion*. http://docs.pointclouds.org/trunk/group__surface.html. Senast uppdaterad 2017-04-24. Hämtad 2017-04-27.
- [42] *Punktmoln*. https://en.wikipedia.org/wiki/Point_cloud. Senast uppdaterad 2017-02-22. Hämtad 2017-04-28.
- [43] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe. 2009, p. 5.
- [44] Ankita Raturi, Birgit Penzenstadler, Bill Tomlinson, and Debra Richardson. "Developing a sustainability non-functional requirements framework". In: *Proceedings of the 3rd International Workshop on Green and Sustainable Software*. ACM. 2014, pp. 1–8.
- [45] Alexis Roche. "EM algorithm and variants: An informal tutorial". In: *arXiv preprint arXiv:1105.1476* (2011).
- [46] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–4.
- [47] Junji Shimagaki, Yasutaka Kamei, Shane McIntosh, Ahmed E Hassan, and Naoyasu Ubayashi. "A study of the quality-impacting practices of modern code review at sony mobile". In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM. 2016, pp. 212–221.
- [48] *Skapa dina formulär*. <https://www.google.se/intl/sv/forms/about/>.
- [49] *Slic3r - About*. <http://slic3r.org/about>. Hämtad 2017-04-28.
- [50] Carl-Göran Ohlson och Peter Westerholm Sten Bornberger-Dankvardt. *Arbetsmiljö- och hälsoarbete i småföretag*. ISBN 91-7045-661-5. 2003.
- [51] *Systemanatomi*. https://en.wikipedia.org/wiki/System_anatomy. Senast uppdaterad 2017-04-13. Hämtad 2017-05-02.
- [52] *Testdata till M. Karlsson's experiment*. <https://github.com/mickwald/Testdata>. Hämtad 2017-05-30.
- [53] *Vilka krav kan man ställa på kontorsbelysning?* <https://www.av.se/inomhusmiljo/ljus-och-belysning/belysning-pa-kontor/>. Senast uppdaterad 2016-07-01.
- [54] K. Martin W. Schroeder and B. Lorensen. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. Kitware, 2006.

- [55] Ben T Wittbrodt, AG Glover, J Laureto, GC Anzalone, D Oppliger, JL Irwin, and Joshua M Pearce. "Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers". In: *Mechatronics* 23.6 (2013), pp. 713–726.
- [56] Qing Xie and Atif M Memon. "Using a pilot study to derive a GUI model for automated testing". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 18.2 (2008), p. 7.
- [57] Xun Yuan, Myra B Cohen, and Atif M Memon. "GUI interaction testing: Incorporating event context". In: *IEEE Transactions on Software Engineering* 37.4 (2011), pp. 559–574.



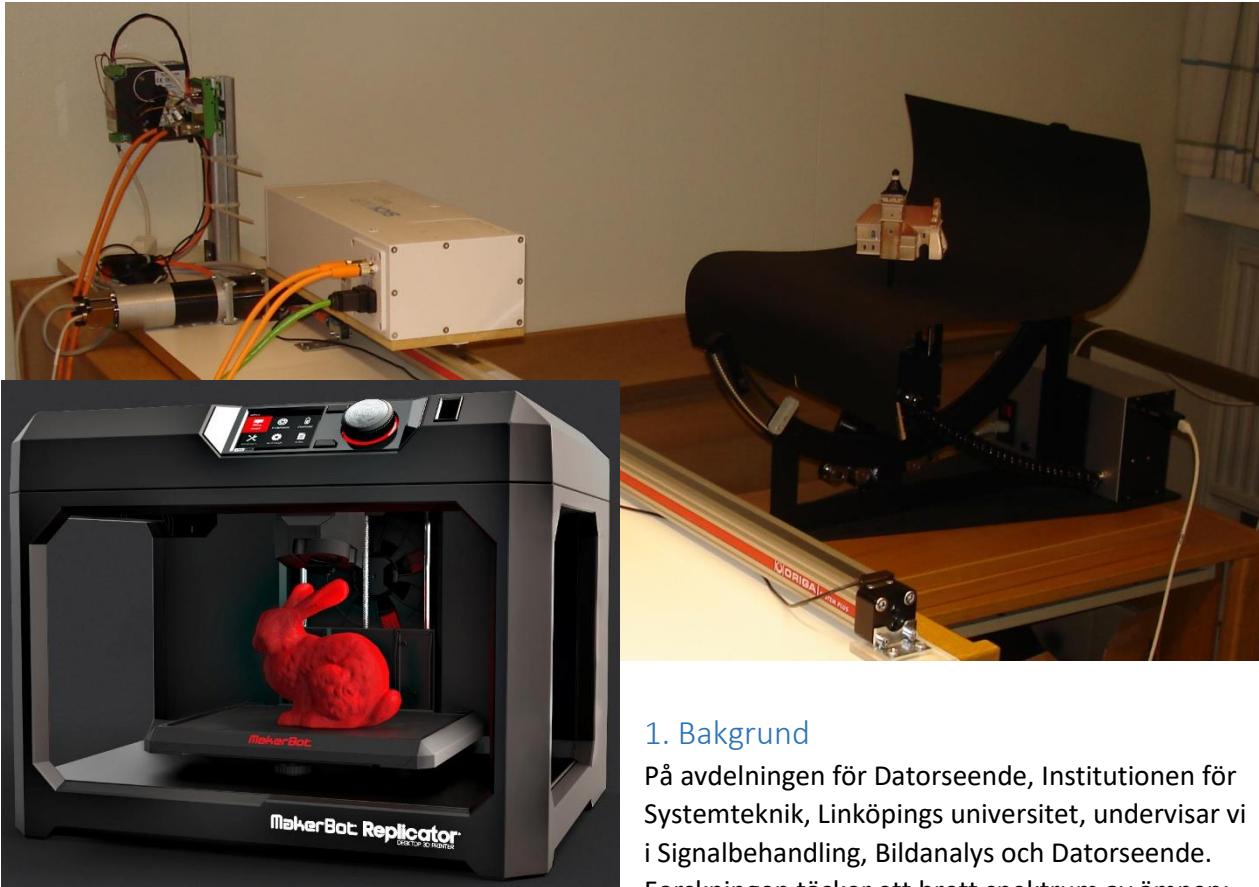
H Bilagor

H.1 Svar på enkät om arbetsmiljö

Tidstamps	Beskriv kort projektet du arbetar med.	Hur intresserad av projektet är du?	Hur många gånger i veckan träffas hela gruppen?	Hur stor del av ditt arbete gör du ensam? Föredrar du att jobba ensam eller tillsammans med någon?	Vart brukar du utföra ditt arbete, hemma, i skolan eller ...?	Hur är belysningen på dessa platser, finns du med?*	Hur är belysningen på dessa platser, finns du med?*	Ar det folk som ordnar dig till att jobba här?	Har ni fått tillgång till någon lokal, sådär hur ofta används den?	Hur tycker du om sammanhållningen i ditt team?	Vårfrör du sammanhållningen är som den är?
2017-03-30 13.25.06 3d kopiering		4		2 25% - 50%	Med någon	Skolan	Lagom förutom hårdvaran	Bra belysning och bra plats			5 Kommer lätt överens och har inte haft några större konflikter.
2017-04-03 13.55.49 Visualisering av Continuous integration				25% - 50%	Tillsammans	Skolan	Lite för högt ofta	Bra	Pluggar eller samtalar		3 Vissa har inte gjort sin besärda del av arbetet
2017-04-03 13.57.49 3D copy		4	Nästan varje dag	25% - 50%	Tillsammans med någon	Skolan	Aceptabel	Adekvat	Beror på var i skolan jag sitter Andra studenter som pluggar detta projektet, särskilt de som är från mina lokaler	4 Vi är bäst	
2017-04-03 13.59.48 Webbaserad visualisering av handelser i CI-flöden		4		5 25% - 50%	Tillsammans	I skolan	Oftast högt	Mestadels bra belysning!	Oftast är det det. Det inte vid de gör, oftast arbetar med sitt, antalet arb	Ja, ca 15 minuter per dag för scrum standup.	2 Vi behöver göra mer saker tillsammans utanför skolan.
2017-04-03 14.05.01 Utvecklar en webbapplikation för Region Östergötland			2 Varietet. Allt från 1 till 10 gånger	25% - 50%	Beror på, men oftaft med någon	I skolan, på Creative, ibland hemma		Den funkar			4 Gruppen består av individer med bra inställning
2017-04-03 14.12.21 Videanalyseverktyg åt Polisen.			4 Varietet. Allt från 1 till 10 gånger	25% - 50%	Jag tycker det är bra att ha sin egen del i projektet att göra, dvs enskilt arbete, men efter givna i rammen lokaler som resterande gruppmedlemmar att komma dit och jobba tillsammans, så då är vi oftaft ändå gruppmedlemmar på dessa tillfällen.	Sitter ofta på Creative, ibland universitet eller Creative.	Ljudvolymen är låg och behaglig på både dessa platser. Creative har väldigt bra belysning, med mycket fönster.	Belysningen är full tillräcklig på både dessa platser. Creative har väldigt bra belysning, med mycket fönster.		Nej	
2017-04-03 14.12.51 Automatiserade videonalyser på NF-C		4 3-4		25% - 50%	Tillsammans	I skolan	Ganska bra, inte för skräckigt	Den är bra	Bladet på universitet och på Creative är det mycket studenter runt omkring		5 Vet faktiskt inte
2017-04-03 14.13.06 Avancerad webbprogrammering		5 4-5		25% - 50%	Tillsammans	Hemma och i skolan, mest skolan nu	Stolen är låg om man har bokat sal	Bra, jag trivs. Har en läslampa hemma så den är bra.	Bara projektgruppen. Vi jobbar mest och snicker sitt ibland.		5 Vi är däre och trevliga grabbar. Tur kanske
2017-04-03 14.15.42 Akutplattan		4 3-5		<25%	Ensam	Skolan	Ganska låvit. Lyssnar dock på musik i headset.	Den är bra nästan överallt. Jag trivs jämfört med den.	Oftast ja. Studenter olika kurser.		4 Vi har bra kommunikation och ingen person som avviker från gruppkontakten.
2017-04-03 14.22.42 Grupp 7. Kartläggning av flygdjurarnas hälsa- och kognitivställdom		3 1 regel alla schemalagda pass		25% - 50%	Tillsammans	Skolan	Platser som Blå torget i b-huset eller studieplats Topp		Ett helhetsintryck utveckles samt två deltagare.		5 Kick-off, många gemensamma arbetspass och har diskuterat fram ett gemensamt mål.
2017-04-03 15.26.59 VR Moduleri av ansträngande student situationer		4		5 >75%	Med den kunskapsklyfta som finns så ensam	I skolan	Låg	Bra	Förstör lyssn. Jag skulle försöka naturligtvis via ett förster	Delvis, vi använder den när ingen annan lokal finns	5 Alla arbetar på samma plats och vi har haft 2 kickoffer redan
2017-04-03 15.31.04 En plugin till boardearers platform		2		7 <25%	Ensam på en specifik uppdrag men andra att diskutera med	I grupprum i skolan	Inte så högljutt	Ingen utanför gruppen			4 Alla motiverade och verkar ha samma mål.
2017-04-03 15.43.33 VR-presentationer		3		5 25% - 50%	Mycket av den individuella delen för kandidatrapporten är skräck att sitta ensam och skriva, men utöver det spelar det ingen större roll för min del.	I grupprum i skolan	Rött laga, samtalensva inom gruppen med eventuella lyssnare emellan.	Dagslus är nice nu när våren är här.	Arbetar, hoppar jag.	Näst intill alla arbetarstider vi sätter tillsammans gruppens sättar vi i vår lokal, så väldigt ofta.	5 Bra teamledare, bra gruppmedlemmar. Kaffe är godt.
2017-04-03 16.27.06 3D kopiering		3		2 25% - 50%	Jag föredrar att jobba tillsammans med någon	I skolan	Det är skräck, ibland kan 3D-skripten samt tekniken vara svår att hantera, vilket nog borrar på att jag själv är intresserad.	Mycket bra	Jag jobbar på samma projekt som mig		Det kan kanske inte alltid är positivt att jobba så sätt in på varandra hänta tiden. Folk stor sig på andra där de prövades och folk lägger sig i andras arbete av intresse men får då ingenting själv gjort. Självklart är det viktigt att jobba med varandra här ibland med starken att man kan hjälpa varandra ständigt om som är det.
2017-04-03 19.40.24 Akutplattan inom värden		5 >3		<25%	Tillsammans med någon	Skolan eller Creative	Låg		Den pluggar också.		3 nägen kör fast
2017-04-03 21.06.35 Grupp 3 - Hur är det att vara en robot?		4		3 <25%	Tillsammans	Skolan	Oftast på kontoret (i skolan), ibland hemma	Bra, jag trivs bra	Arbetar med kontoret i särskilt särskilt vad som behövs.		5 Vi trivs tillsammans och jobbar på bra.
2017-04-03 21.77.03 3D kopiering		5	De festa dagar, dock saknas ofta någon	25% - 50%	Beror mycket på uppgiften men generellt tillsammans		Oftast bra. Mycket bättre sen vi fick tillredning	Tolv bra med den.	För det mest är det bara folk som jobbar på projektet där.		5 Vi är öppna och har kul
2017-04-04 07.30.45 Akutplattan		5	En gång på möte, sitter dock ofta tillsammans och jobbar ifallfall	25% - 50%	Tillsammans	Skolan framst, ibland hemma	Okej för det mest	Okej	För det mest sitter vi i grupprum		3 Motgångar i arbetet och när vissa inte dyker upp på kontoret trots att det skulle behövas.
2017-04-04 10.41.13 Karakterisering av en mykvara för att jämföra olika teknologier ekonomiska tillstånd.		4 Flera gånger varje dag ofta		<25%	Beror på arbetet, oftaft föredrar jag grupparbete	Hemma och jag jobbar själv, på campus om jag jobbar med folk, ibland Creative.	Somliga salar på campus ger mig huvudvärk efter för lång tid, trots det har mest belysning att göra.	Hemma är det ingen, på campus främst folk studera, ibland kontoret.	Jag jobbar på samma projekt som mig		4 Alla vill samma sak och är dessutom trevliga mänriskr. :)
2017-04-04 11.22.02 studievärden		5		2 50% - 75%	Bada, mest med någon annan nära	I skolan	Låg	Bra förstör, trevligt jus			
2017-04-04 16.05.28 Konkurrentanalys		4		5 <25%	Tillsammans	Skolan	Ljus jag gillar belysningen				4 Bra start och alla respekterar varandra?
2017-04-04 16.15.42 Konkurrentanalys		4 2-3		<25%	Tillsammans	Skolan	Bra	Jag, de jobbar			5 Vi skön stämning i grupper!
2017-04-04 18.05.14 Videanalyseprogram		4 3-5		25% - 50%	Tillsammans	hemma, i skolan eller creative	bra	Jobbar			5 Samma ambitionsnivå
2017-04-04 21.21.05 Smart lässystem		4	Möte 1 gång. Jobbar alla tillsammans 2-3 ggr.	25% - 50%	Tillsammans med någon (t.ex. parprogrammering)	Skolan, Creative	Låg-medel	God arbetsbelysning, just			
2017-04-05 15.03.38 Kunderbjudanden i molnet		4	4 25% - 50%	Jag föredrar att jobba tillsammans med någon.	Först i skolan	Då vi ofta bokar salar är ljudnivån väldigt bra Belysning.					4 Värt tydiga med att vi har högt i tak och att alla kan läsa på att gruppmedlemmar drar sitt strå till ståcken.
2017-04-07 10.45.58 Ett smart lässystem		4 Minst 1 gång		<25%	Tillsammans	På creative	Låg				4 Värt väldigt nog med att jag faktiskt arbetar som led till transparens och ett gemenskap åtgärdsavskap.
Stapa en analys av ett företags konkurrenter på ett enkelt och pedagogiskt sätt, som är tillgänglig i redan tillgänglig form.					Mestadels i skolan, men en del hemma	Bra	Ja	I skolan har vi ofta gruppum så då är det entart projektmedlemmar			4 Kodbasen, vilket gör att alla känner sig investerade i projektet och stöder över sin egen och andras insat.
Parallel utveckling på hårdvara och mobilapplikation, IoT-relaterat.		2 -4 -5 ggr		<25%	Kvällar	Både och, för jag arbetar bra ensam, men gillar att kunna diskutera det jag jobbar om	Duglig	Dålig hemma, bra Creative, trivs med både.	Jag, avhörs också		
2017-04-10 12.52.19 Skapa ett konkurrentanalysverktyg		4 5 gånger		50% - 75%	Både hemma och i skolan	Skolan men rätt mycket hemma (dubbelklickar för the win)	Bra	Jobbar			4 God struktur i projektet.
2017-04-10 19.20.35 Konkurrentanalys		4	5 25% - 50%	Parprogrammering är föredra. Ensam går bra ibland med							5 Vi fungerar bra ihop
2017-04-21 14.03.23 Vi konstruerar en hemsida med webshop i Wordpress		3 1 till 4 gånger		25% - 50%	tillsammans med någon	i skolan eller hemma		belysningen är bra på alla de nämnda platserna			
Sammanfattnings		3.81414815	3.5 ggr per vecka	35.55555556	Tillsammans	Skolan	Bra	Bra, gillar förstör, lyfter intet opmatt	Mestadels andra studenter som bokar pluggar		4 Går bra så länge alla gör sitt jobb. När de känner som att någon/några inte drar sitt strå till ståcken eller det inte kan. Efferson, vilket gör att de under den idén de har arbetat.
											3 Sökt att veta om jag faktiskt arbetar.
											4.222222222. Inte gär sa blir det sittningar.

H.2 Projektdirektiv

System för 3D-kopiering



1. Bakgrund

På avdelningen för Datorseende, Institutionen för Systemteknik, Linköpings universitet, undervisar vi i Signalbehandling, Bildanalys och Datorseende. Forskningen täcker ett brett spektrum av ämnen:

beräkningsfotografi; detektion, följning och igenkänning av objekt; skattning av pose och 3D-struktur; robotseende och autonoma system; medicinsk bildanalys och bildrekonstruktion.

Vårt avståndskamerasytem används i kursen Bildsensorer. Med detta kan man mäta upp ett objekt i 3D. Systemet byggdes ursprungligen av en doktorand för användning i hans forskningsprojekt. Systemet består av: avståndskamera (sheet-of-light laser); linjärenhet för att förflytta avståndskameran; rotationsbord för rotering av det objekt som ska avbildas. Systemet med dess tre olika delar styrs från ett ROS-baserat system som har utvecklats under 2016 i kursen TDDD96.

ROS (Robot Operating System) är ett programvaru-bibliotek för utveckling av mjukvara till robotiksystem. ROS ger standardiserade operativsystemstjänster såsom hårdvara-abstraktion och kommunikation mellan olika processer. ROS innehåller även bryggor till point-cloud library (PCL), visualisering och ett JavaScript bibliotek.

2. Mål och Vision

Vi vill komplettera vårt avståndskamerasytem med en 3D-skrivare och mjukvara för att kunna skriva ut uppmätta enkla och, om möjligt, medelsvåra 3D objekt. Med detta vill vi erhålla en 3D-kopiator som kan användas både inom forskning och undervisning.

3. Viktiga krav

Systemet ska fungera robust och vara lätt att använda (GUI). Utskrifternas noggrannhet ska motsvara den uppmätta 3D-modellen och skrivarens förmåga. Definition av *enkla objekt* är: konvexa objekt, ca 10-15 cm stora. *Medelsvåra objekt* liknar enkla objekt, men får delvis ha konkava ytor eller hål som är synliga från sidan. Systemet ska bygga på ROS i största möjliga omfattning.

En del av projektarbetet består av att rotera objektet till lämpliga positioner, scanna och erhålla flera set av punktmoln. Dessa ska sedan matchas ihop (registrering). Skymda ytor ska kunna identifieras och åtgärdas (gäller medelsvåra objekt).

4. Programomgivning

Hårdvara:

- avståndskamera (med sheet-of-light laser princip), modell: SICK-IVP RULER E600 (Ruler-E2111)
- linjärenhet för att förflytta avståndskameran, modell: ORIGA SYSTEM PLUS
- rotationsbord med två rotationsaxlar för att rotera de objekt som ska avbildas, unikt exemplar beställt av konsult, styrs via serieport RS232, fullständig dokumentation saknas, men fungerande styrning via ROS finns.
- Dator
- 3D-skrivare kommer inhandlas enligt behov och budget

Mjukvara:

- operativsystem Linux (Ubuntu)
- programmeringspråk Python eller C++
- programvarubiblioteken ROS och PCL
- MeshLab och forskningskod för registrering av punktmoln
- ROS-baserade skannermjukvara som producerar punktmoln

5. Kontaktpersoner

Maria Magnusson

Andreas Robinsson

6. Administrativt

Genom att skicka in detta förslag lovar jag att jag läst informationen på
<http://www.ida.liu.se/~TDDD96/inbjudan/index.sv.shtml>.

Om projektet blir valt kan jag träffa studenterna vecka 4, 2017.

Om projektet blir valt kan skriva ett avtal med studenterna senast 2017-02-01.

(Stryk det som inte gäller dig.)

- Jag avser att teckna ett eget avtal med studenterna som går ut på att licensiera den resulterande programvaran under öppen källkods-licens: (GPL)

H.3 Intervju guide

Intervju guide

1. Hur gjorde ni för att samla in och specificera krav i ert projekt?
2. Hur mycket använde ni IEEE std 830 när ni gjorde er kravspecifikation?
3. Vad är dina åsikt om IEEE std 830?
4. Hur mycket har ni behövt revidera er kravspecifikation?

H.4 Svar på enkät om dokument

Tidstämpe	Vilken grupp är du i?	Vilken webbsida/tjänst använder ni er av för större/viktiga dokument?	Hur nöjd är du med att använda en tjänst som denna för hantering av dokument?	Om du fick byta den tjänst ni använde, vilken hade du då valt?	Skulle du vilja byta hantering av dokument till asynkrona verktyg som Git och SVN?	Om du svarade ja, vilket verktyg skulle du då vilja använda?	Vilket verktyg använder ni när ni skriver de fiesta större-viktiga dokument?	Vad för slags versionshanterare använder ni er av?	Hur nöjd är du med hur ni versionshanterar era dokument?	Om du fick ändra vilket asynkront versionshandteringsverktyg ni använde, vilket hade du då valt?	Skulle du vilja byta hantering av dokument till synkrona verktyg som Google Docs, ShareLaTeX och liknande?	Om du svarade ja, vilket verktyg skulle du då vilja använda?	Vilka verktyg som har listats här tycker du är bra för större dokument?
2017-04-24 12:56:52	10						Asynkrona verktyg som Git, SVN och liknande	Git	5 Git	Nej			Git
2017-04-24 12:57:45	1						Asynkrona verktyg som Git, SVN och liknande	Git	4 Git	Ja		Google Docs	Google Docs;ShareLaTeX;Git
2017-04-24 12:57:58	2						Asynkrona verktyg som Git, SVN och liknande	Git	4 Git	Ja		ShareLaTeX	ShareLaTeX;Git
2017-04-24 13:02:21	2						Asynkrona verktyg som Git, SVN och liknande	Git	4 Vill inte byta	Ja		ShareLaTeX	Google Docs;ShareLaTeX;Overleaf;Git
2017-04-24 13:11:59	2						Asynkrona verktyg som Git, SVN och liknande	Git	4 Vill inte byta	Nej		Google Docs;Git	
2017-04-24 13:14:55	10						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Ja		Google Docs	Google Docs;Git
2017-04-24 13:15:18	4 Overleaf		3 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Office 365 Word;ShareLaTeX;Overleaf
2017-04-24 13:15:28	4 Overleaf		4 Google Docs		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;ShareLaTeX;Overleaf;Git
2017-04-24 13:22:04	12 ShareLaTeX		5 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;ShareLaTeX;Git
2017-04-24 13:55:07	9						Asynkrona verktyg som Git, SVN och liknande	Git	4 Vill inte byta	Nej			Git
2017-04-24 13:59:33	10						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Google Docs;ShareLaTeX;Git
2017-04-24 14:00:57	4 Overleaf		3 Om man hade kunnat kombinera online-tjänster med Git så hade det förenklat vissa problem		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX;Overleaf;Git
2017-04-24 14:01:13	11 Google Docs		4 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Overleaf
2017-04-24 14:16:45	4 Overleaf		4 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;ShareLaTeX;Overleaf
2017-04-24 14:36:06	1						Asynkrona verktyg som Git, SVN och liknande	Git	3 Git	Ja		ShareLaTeX	Google Docs;ShareLaTeX;Git
2017-04-24 14:43:17	13						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Google Docs;ShareLaTeX;Git
2017-04-24 14:44:41	13						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Google Docs;Git;SVN
2017-04-24 15:31:27	12 ShareLaTeX		5 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Office 365 Word;ShareLaTeX;Overleaf
2017-04-24 17:08:58	7 ShareLaTeX		4 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX
2017-04-24 17:29:55	7 ShareLaTeX		4 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX;Overleaf;Git;SVN;Mercurial
2017-04-24 17:48:38	11 Google Docs		5 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Overleaf
2017-04-24 18:37:26	6 Office 365 Word		1 Overleaf		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Git
2017-04-24 19:49:46	10						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Git
2017-04-25 07:33:42	13						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Google Docs;Git
2017-04-25 08:06:32	6 Word Online		3 ShareLaTeX		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX;Overleaf
2017-04-25 09:04:59	9 Overleaf		3 Vet ej		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Git
2017-04-25 10:49:32	11 Google Docs		4 Git		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Overleaf;Git
2017-04-25 17:31:59	11 Overleaf		5 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Overleaf
2017-04-26 11:01:17	1						Asynkrona verktyg som Git, SVN och liknande	Git	4 Git	Ja		ShareLaTeX	ShareLaTeX;Git;SVN
2017-04-26 16:32:30	8 Overleaf		5 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;Overleaf
2017-04-26 16:33:08	2						Asynkrona verktyg som Git, SVN och liknande	Git	5 Vill inte byta	Nej			Google Docs;ShareLaTeX;SVN;Mercurial
2017-04-27 10:54:40	9 Overleaf		4 Vill inte byta		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX;Overleaf;Git;SVN
2017-04-27 14:48:14	9 Overleaf		2 LaTeX och git		Ja	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Git
2017-04-27 15:48:12	12 ShareLaTeX		4 Google Docs		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;ShareLaTeX
2017-04-28 10:09:11	3 ShareLaTeX		4 Vill inte byta		Nej	Git	Synkrona verktyg som Google Docs, ShareLaTeX och liknande						ShareLaTeX;Git
2017-04-30 15:16:09	12 ShareLaTeX		4 Vill inte byta		Nej		Synkrona verktyg som Google Docs, ShareLaTeX och liknande						Google Docs;ShareLaTeX;Overleaf

H.5 Svar på enkät om erfarenheter för dokument

Tidstamps	Fråga	Vilket verktyg tycker du har varit bäst att använda när vi skrivit större dokument?	Varför tycker du det?	För kandidatrapporten, tycker du att det var bättre med separata filer än en gemensam fil?
2017-05-04 10:27:15	bra	ingen åsikt	git har sina problem, overleaf har sina, tycker inte att nån av dom har varit "bäst"	Separata filer
2017-05-04 10:27:18	Bra	Git och LaTeX	Man har enkelt kunna arbeta självständigt samtidigt som man tar del av andras ändringar på ett smidigt sätt.	Separata filer
2017-05-04 10:27:38	Helt okej men kunde varit bättre	Latex på git	Har funktat smidigast, har inte uppstått så många problem. Bra att ha historiken i git och kunna se diffar och liknande	Separata filer
2017-05-04 10:28:49	Generellt har det gått bra. Några enstaka problem med git hanteringen pga latex	Tycker både overleaf och git har funkat bra.	Overleaf hade fördelen att man slapp mergeconflicts och att flera kunde jobba på samma del samtidigt. Git ger en bättre versionshistorik och kontroll på version.	Separata filer
2017-05-04 10:29:02	Jag tycker att hanteringen av större dokument har gått väldigt bra. Vi har tagit hänsyn till hur stort och viktigt dokumentet är och hanterat det därefter.	Git och TeXStudio	Lättare att versionshantera och se ändringar än Overleaf.	Separata filer
2017-05-04 10:35:56	Jag tycker att det har gått bra, det gick bättre med att använda sig av Git än med Overleaf.	LaTeX med Git	För att man kan se historik, vem som skrivit vad, ifall något försvinner eller om man vill gå tillbaka till en tidigare version av dokumentet.	Separata filer

H.6 Svar på enkät om kvalitetssäkrande process

Tidstämpel	Hur anser du det är att jobba med en kvalitetssäkrande process, är det positivt eller negativt, om man tar den faktiska tidsåtgången i åtanke?	Har du jobbat med den specifika metoden kodgranskning för att säkerställa kvalitén på en produkt?	Tycker du att kodgranskning hjälper att kvalitetssäkra en produkt?	Ansar du graden av deltagande mäniskor i kodgranskningen hjälper eller försämrar kvalitén hos produkten? Blir det för komplicerat att anordna möten, tidsåtgången blir för stor, det tar för lång tid att få koden godkänd med mera.	Har du några övriga tankar om att arbeta med en kvalitetssäkrande process eller med kodgranskning?
2017-04-03 11.38.39	Positivt!	Ja	Ja	Jag tycker det fungerar bra. Inga direkta problem.	Jag tycker det är ett bra sätt att säkerställa kvalitén på koden, framförallt läsbarheten av koden.
2017-04-03 11.39.40	Positiv	Ja	Ja	Det räcker med 2 personer	Nej
2017-04-03 11.41.29	Det är positivt då det bidrar till att höja kvalitén hos produkten samtidigt som man lär sig något av att genomföra kvalitetssäkringen.	Ja	Det tycker jag, framförallt gör den koden lättförståelig och mer genomtänkt.	Fler än 3 känns onödigt då det mest väsentliga borde ha hittats och korrigerats efter 3st kodgranskningar. D.v.s. att 3 personer har utfört en individuell granskning. Att ha flera personer som genomför samma granskning hjälper nog inte lika mycket för kvalitén som om man gör den individuellt.	Nej
2017-04-03 11.47.22	Positivt, en person missar ofta många detaljer som kunde göras bättre. Processerna kan även göras så att tidsåtgången inte blir så stor, t.ex. om de ger andra gruppmedlemmar en chans att sätta in sig i koden.	Nej, i vårt kandidatprojekt används dock parprogrammering för ungefär samma orsaker.	Helt klart.	Mer än en eller två personer (beroende på kodens storlek) är antagligen för mycket vad gäller tidsåtgång. De fel som inte ses av 3 personer kommer antagligen inte ses av 5...	
2017-04-03 13.02.44	det är positivt, även om det tar tid så tjänar man i långa loppet om man kan få mindre buggar.	ja	ja	Ja ansar att desto fler desto bättre, man får fler inblickar. man behöver inte sitta tillsammans	nej
2017-04-03 14.19.26	Positivt	Ja	Ja	Vet ej	
2017-04-04 07.28.00	Bra! Man lär behöva sitta mindre tid i slutet på projektet för att lösa alla de konflikter och snygga till koden som annars skulle uppstå.	Ja	Ja!	Vet inte? Kanske det blir väl tidkrävande? Å andra sidan blir fler insatta i all kod. Inga "superstars" som är ensamma om det.	Det är bra!
2017-04-04 09.11.38	positivt	ja	ja	det hjälper	nej
2017-04-04 09.14.59	Positivt. Tror det hjälper väldigt mycket med kvaliteten.	Ja	Ja	Jag tycker det är lagom om ca 2 pers granskar en bit kod. Blir det för många blir det inte så mycket bättre men tar upp mer tid	Tycker det är en väldigt bra sak