I4PRJ4 – IHA

RoboGO

Dato: 01-06-2012

Sorting Industrial Robot

Projektrapport

Vejleder:

#1 Projektvejleder:	Navn: Poul Ejnar Rovsing

Deltagere:

#1 Studienummer: 10778	Navn: René Høgh Thomsen
#2 Studienummer: 10517	Navn: Cong Thanh Dao
#3 Studienummer: 10430	Navn: Søren Howe Gersager
#4 Studienummer: 10791	Navn: Michael Batz Hansen
#5 Studienummer: 10898	Navn: Sam Luu Tong
#6 Studienummer: 10754	Navn: Nicolaj Quottrup
#7 Studienummer: 10568	Navn: Yusuf Tezel
Dato: 1. juni 2012	Godkendt af:

Resumé

Formålet med projektet er at implementere software til en Scorbot ER-4u for firmaet RoboGO. Denne robot med tilhørende programmel har til formål at automatisere en eksisterende manuel proces, der angår sortering og registrering af dimensioner af forskellige klodser med forskellige masser samt massefylder.

Registreringsdelen foregår i en database, der er udviklet til projektets formål om automatiseret registrering. Måden, som registrering bliver fortaget på, er gennem målinger af klodsens tre dimensioner, hvorefter klodsen bliver flyttet over til en vægt for at skaffe klodsens masse.

Det ligger endvidere i projektgruppens ansvar at udvikle en vægtforstærker, som er en vigtig del af systemet. Det er muligt ud fra klodsens masse og dimensioner at beregne en massefylde for den enkelte klods, som herefter kan kategoriseres i en kategori, som der kan sorteres ud fra.

For at opnå succes med dette system anvendes erfaring opnået ud fra Ingeniørhøjskolen Aarhus Universitets kurser ”Software test”, ”Windows Programmering”, ”Interfacing” samt ”Databaser”. Alle nævnte fag har hver især spillet en speciel rolle i projektet.

Derudover har der været anvendt UP og TDD som vores udviklingsproces, og under arbejdet er der anvendt arbejdsmetode i form af XP. Scrum benyttet vi som framework.

Projektet mundede ud i en automatiseret proces som fungerede efter hensigten.

MANGLER RESULTATER OG KONKLUSION, SOM FØRST KAN LAVES SENERE!!!

Abstract

The purpose of the project is to implement a Scorbot ER-4u for the company RoboGO. This robot and its associated software is to be designed with the purpose to automate an existing manual process of sorting and recording the dimensions of different blocks with different colors, masses and densities.

The recording takes place in a database, which is developed specifically for the project on automated recording. The way the recording is done, is by measuring the blocks sizes, after which the blocks are moved to the weight cell to measure weight and the rest of the values.

Furthermore is it the project groups responsibility to develop an amplifier for the weight cell, which is an important part of the system. It is possible by taking the blocks masses and dimensions to calculate the density for one specific block, which afterwards can be placed in a category to be sorted after.

To achieve success with this system, gained knowledge from Engineering Aarhus Universitet courses ”Software test”, ”Windows Programmering”, ”Interfacing” and ”Databaser” is being put to use. All of the mentioned courses have individual played a speciel role.

We have used UP and TDD as our software development processes, and as our work method we've used XP, further more we used Scrum as framework.

The project culminated in an automated process which worked according to plan.

Indholdsfortegnelse

1. Indledning

Dette projekt har til formål at udvikle et automatiseret system til en robot, som kan foretage sorteringer af elementer med forskellige massefylder. Systemet er implementeret til en Scorbot ER-4u robot. Der er stillet krav til, at der skal være en visuel brugergrænseflade til Windows udviklet i C#, samt at der skal persisteres data i en database. Projektets problemformulering er givet af underviserne på ASE på studieretningen IKT 4. semester og ses i næste punkt.
Valget af litteratur læner sig opad kurserne på 4. semester for IKT-studerende, som danner grundlag for arbejdet i projektet. Der henvises til, at den brugte litteratur i form af bøger kan findes i kravspecifikation punkt 1.2. Til dette punkt står, hvilke bøger der er brugt i de diverse kurser, samt skal det siges, at der er brugt diverse, hurtige internetopslag til f.eks. programmering af Windows brugergrænsefladen.
Sidst skal det nævnes, at Scrum er valgt som framework til projektarbejdet. De forskellige elementer i Scrum blev til dels fulgt, men der var visse dele, som blev fravalgt eller ikke vedligeholdt. Arbejdet i gruppen har dels været på enkelt- eller tomandshånd, medens de store beslutninger ang. f.eks. arkitektur, dokumentation, standarder, kodeskrivning m.m. er taget i fællesskab.

1.1 Læsevejledning

- Punkt 2 refererer til et link, hvor der er opgivet, hvad der forventes af os.
- Punkt 3 beskriver beskriver størstedelen af projektet overordnet og den proces, der er anvendt gennem projektet.
- Punkt 4 beskriver den nåede konklusion i projektet.

2. Opgaveformulering

Til dette punkt vælges der at blive referere til følgende link, hvor opgaveformuleringen ligger: http://kurser.iha.dk/eit/i4prj4/Projektoplaeg.doc

3. Projektbeskrivelse

3.1 Afgrænsning

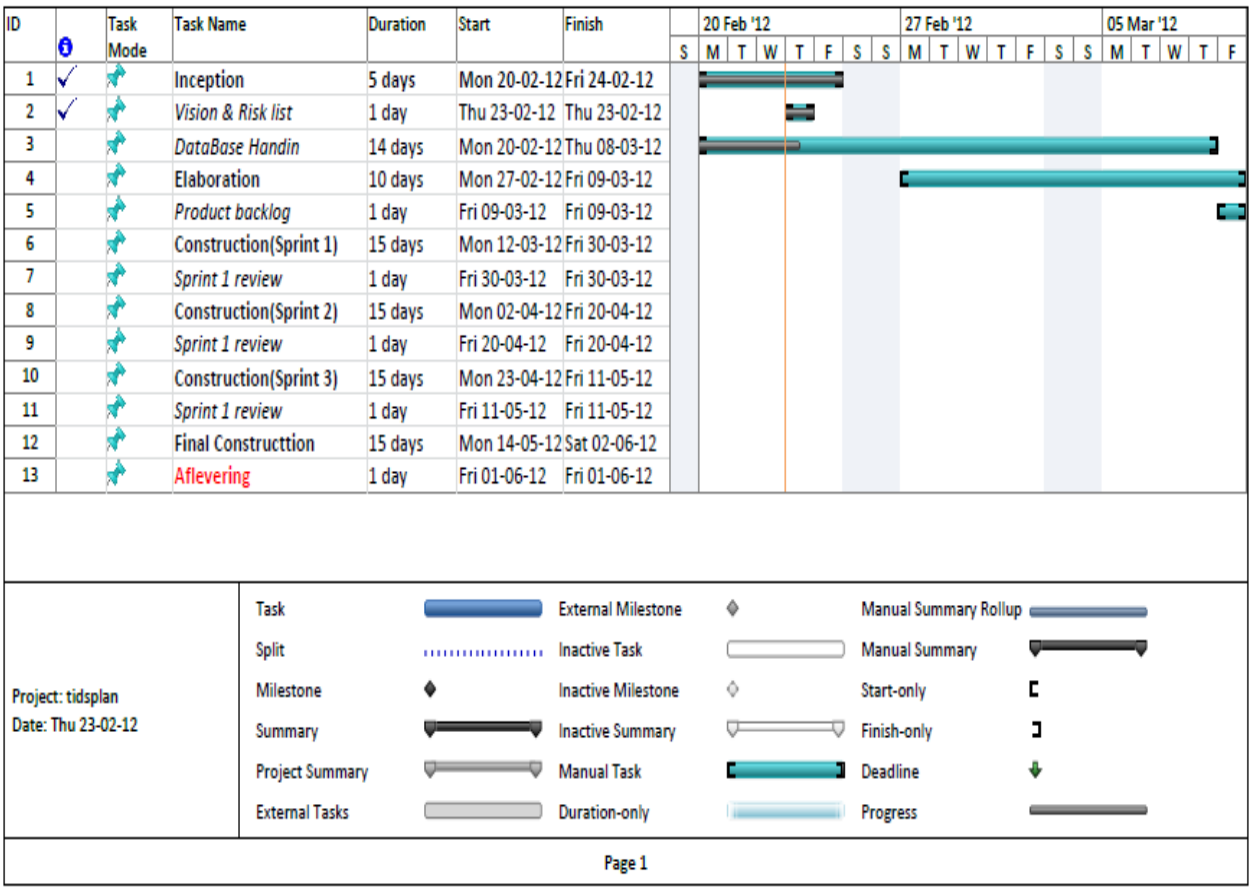
Der skal minimum implementeres

- En database, hvori alle relevante data fra processen kan opbevares og manipuleres.
- En IDE, hvori man kan fremstille programmer til styring af robot-processen (Til dette er der valgt IronPython, en Python fortolker som knytter sig tæt op til .NET frameworket).
- En simulator (et program der simulerer robotten), som kan kobles på systemet i stedet for den fysiske robot. Dette vil blandt andet i et vist omfang kunne anvendes i forbindelse med test.
- Et styreprogram med grafisk brugergrænseflade (programmeres i C#). Via brugergrænsefladen skal man blandt andet kunne starte/stoppe systemet, aflæse status for sorteringsprocessen, få vist relevante alarmer, arbejde med data i databasen og lave programmer til styring af robotten.
- Et system samt programmel til at kunne bruge vejecellen.

3.2 Projektgennemførelse

For at kunne gennemføre et projekt af denne kaliber, har der været nogle ting vi har været nødt til at tage højde for lige fra starten.
Vi lagde ud med at uddele roller, såsom projektleder, scrummaster og product owner, samt diverse ansvarsområder, dvs. vi hver især har haft et område. Vi har derfor alle sammen været med til at styre slagets gang og til sidst at kunne opnå vores mål. Ligeledes blev der i starten besluttet, hvilke softwareudviklingsmetoder vi ville gøre brug af, og valgene faldt på UP med Scrum som framework, idet vi har haft gode erfaringer med disse. Herefter blev der lavet en tidsplan over hele forløbet, da UP har timeboxed iterationer. Vi har altså hele tiden haft et godt overblik over projektet. Vigtigt for vores projektstyring var, at vi holdt os strengt inden for de rammer, vi havde fastlagt. Dette betød dog nogle gange, at vi var nødt til at opgive noget arbejde for at afslutte et sprint. Ved hvert sprint opstart afholdes et sprint planning. I dette fik vi diskuteret de use cases, vi skulle have med igennem, og på den måde fik vi sikret, at alle havde en klar forståelse for, hvordan en specifik use case skulle laves. Samtidig blev der defineret et mål for hvert sprint, f.eks. i det første sprint

”Implementeret delvise funktionaliteter af de højst prioriterede use cases.” Disse mål for hvert sprint var med til at give en motivationsfaktor, da vi havde et mål til sidst at kunne opnå vores mål. Ligeledes blev der i starten besluttet, hvilke softwareudviklingsmetoder vi ville gøre brug af, og valgene faldt på UP med Scrum som framework, idet vi har haft gode erfaringer med disse. Herefter blev der lavet en tidsplan over hele forløbet, da UP har timeboxed iterationer. Vi har altså hele tiden haft et godt overblik over projektet. Vigtigt for vores projektstyring var, at vi holdt os strengt inden for de rammer, vi havde fastlagt. Dette betød dog nogle gange, at vi var nødt til at opgive noget arbejde for at afslutte et sprint. Ved hvert sprint opstart afholdes et sprint planning. I dette fik vi diskuteret de use cases, vi skulle have med igennem, og på den måde fik vi sikret, at alle havde en klar forståelse for, hvordan en specifik use case skulle laves. Samtidig blev der defineret et mål for hvert sprint, f.eks. i det første sprint ”Implementeret delvise funktionaliteter af de højst prioriterede use cases.” Disse mål for hvert sprint var med til at give en motivationsfaktor, da vi havde noget at arbejde os hen imod.



ID	Task Mode	Task Name	Duration	Start	Finish
1		Inception	5 days	Mon 20-02-12	Fri 24-02-12
2		*Vision & Risk list*	1 day	Thu 23-02-12	Thu 23-02-12
3		*DataBase Handin*	14 days	Mon 20-02-12	Thu 08-03-12
4		Elaboration	10 days	Mon 27-02-12	Fri 09-03-12
5		*Product backlog*	1 day	Fri 09-03-12	Fri 09-03-12
6		Construction(Sprint 1)	15 days	Mon 12-03-12	Fri 30-03-12
7		*Sprint 1 review*	1 day	Fri 30-03-12	Fri 30-03-12
8		Construction(Sprint 2)	15 days	Mon 02-04-12	Fri 20-04-12
9		*Sprint 1 review*	1 day	Fri 20-04-12	Fri 20-04-12
10		Construction(Sprint 3)	15 days	Mon 23-04-12	Fri 11-05-12
11		*Sprint 1 review*	1 day	Fri 11-05-12	Fri 11-05-12
12		Final Constructtion	15 days	Mon 14-05-12	Sat 02-06-12
13		Aflevering	1 day	Fri 01-06-12	Fri 01-06-12

Figur 1: Delvis tidsplan. Se CD´en for fulde version.

3.3 Metoder

For at projektet kunne realiseres, har vi arbejdet efter iterative metoder. Dette har givet os et bedre overblik, da projektet er blevet brudt ned i mindre dele.
Vi har derfor kunnet fokusere på enkelte dele af projektet frem for projektet som helhed.
Vi har arbejdet efter UP, som overordnet består af fire faser.

- Inception
- Elaboration
- Construction
- Transition

Idet UP er fleksibel og åben, har vi samtidig også brugt noget fra andre agile metoder, især XP. Her har vi benyttet os af pair programming og Test-driven development (TDD). Vi har haft en product backlog, hvor alle vores use cases havde forskellige prioriteter afhængigt af, hvor vigtigt de var for at opnå vores mål. En scrum tavle, så vi hele tiden vidste, hvad der skulle laves, samt har vi kunnet følge udviklingen på burndown chart. Endvidere har vi så tit som muligt holdt daily scrum, samt evalueret os selv efter hvert sprint ved at holde retrospektiv.
Vi har prøvet at estimere vores focus factor ud fra de antal mandedage, vi havde til rådighed, men idet der løbende har været afleveringer og andre ting, endte det ikke med at være så brugbart.

Inception

Den korteste af alle faserne – herunder beskrev vi visionen for vores firma og de risici, der var forbundet ved udarbejdelse af projektet.

Elaboration

Målet for denne fase var at analysere problemstillingerne og fastlægge en grundlæggende arkitektur for systemet. Dette blev gjort igennem en kravspecifikation og en accepttest.
I kravspecifikation arbejdede vi efter use case driven design for at fastlægge de overordnede krav til systemet, og accepttesten blev udformet og skrevet ud fra denne.

Construction

Den største fase i projektet, som foregår over flere iterationer. Her bygges systemet ud fra det fundament, vi havde lagt i Elaboration-fasen.
Construction-fasen bestod af fire iterationer, hvor hver iteration var timeboxed på 3 uger. I opstart af hver iteration holdte vi sprint planning, hvor de use cases i vores product

backlog med højest prioritet blev taget ud, herefter blev der lavet fully dressed for dem, så vi havde et udgangspunkt. Vi tog altså en lille del af systemet, som vi implementerede, testede og integrerede. Systemet er altså blevet inkrementeret efter hver iteration i den forstand, at det har fået udvidet funktionalitet. Ligeledes har vi fået feedback efter hver iteration af vejlederen ved brug af accepttesten. Vi har således kunne tilpasse os selv og forøge vores arbejdsværdier.

Transition

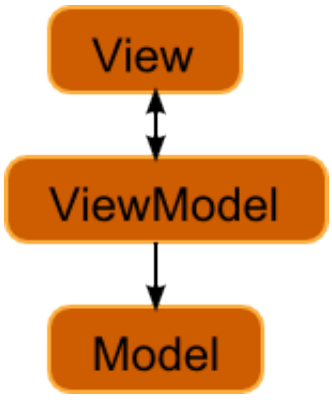
Den sidste fase, hvor det færdige system overleveres til slutbrugeren.

3.4 Designprocessen

Brugergrænsefladen:

Til at starte med blev der skabt et overblik over, hvilke vinduer der skulle laves (se evt. kravspecifikation for første udkast af brugergrænsefladen). Derefter blev vinduerne én for én skitseret ud fra de funktioner, de skulle indeholde. Ud fra skitserne blev vinduerne så modelleret i Visual Studio 2010 ved hjælp af WPF, og til sidst samlet i et hovedvindue med tabkontrol. Elementerne i hvert vindue er organiseret i forhold til hinanden med et grid panel som i det omfang, det var muligt, er blevet genbrugt fra vindue til vindue. Grunden til den store brug af grid panelet gør det lettere at sætte tingene ind ved brug af designeren(VS), men i løbet af senere design kan det skabe problemer, hvis noget skal resizes. Dette betyder, at nogle af de store tekstbokse (brugt som editorer) er blevet sat de samme sted. Desuden er der blevet tænkt på, at menulinjen er meget ens for hinanden, så disse er blevet placeret de samme steder i de samme vinduer.

Programmet er blevet implementeret med MVVM pattern:



Figur 2: MVVM pattern

Vi valgte dette, da det var en af de designmønstre, som der var godt understøttet i WPF, samt fjernede det afhængighed mellem komponenterne.

3.5 Database:

Designet af databasen har hængt sammen med undervisningen i kurset DAB1, som gruppen har været undervist i sideløbende med projektet. Det har i starten været en del af undervisningen at designe en løsning til projektet. Den endelige database er blevet ændret gennem de forskellige sprints i Scrum-frameworket.

Løsningen koncentrerede sig i starten om tre komponenter:

- System log komponent – omhandler alle løbende informationer omkring kørslen af systemet.
- Inventory komponent – varetager alle informationer omkring opsætning af systemet samt omfatte informationer om hardware og brugere af systemet.
- Converting komponent – varetager informationerne om de data, der skulle bruges til sorteringsprocessen, samt konverterer den dataene.

Det endelige design ser således ud:

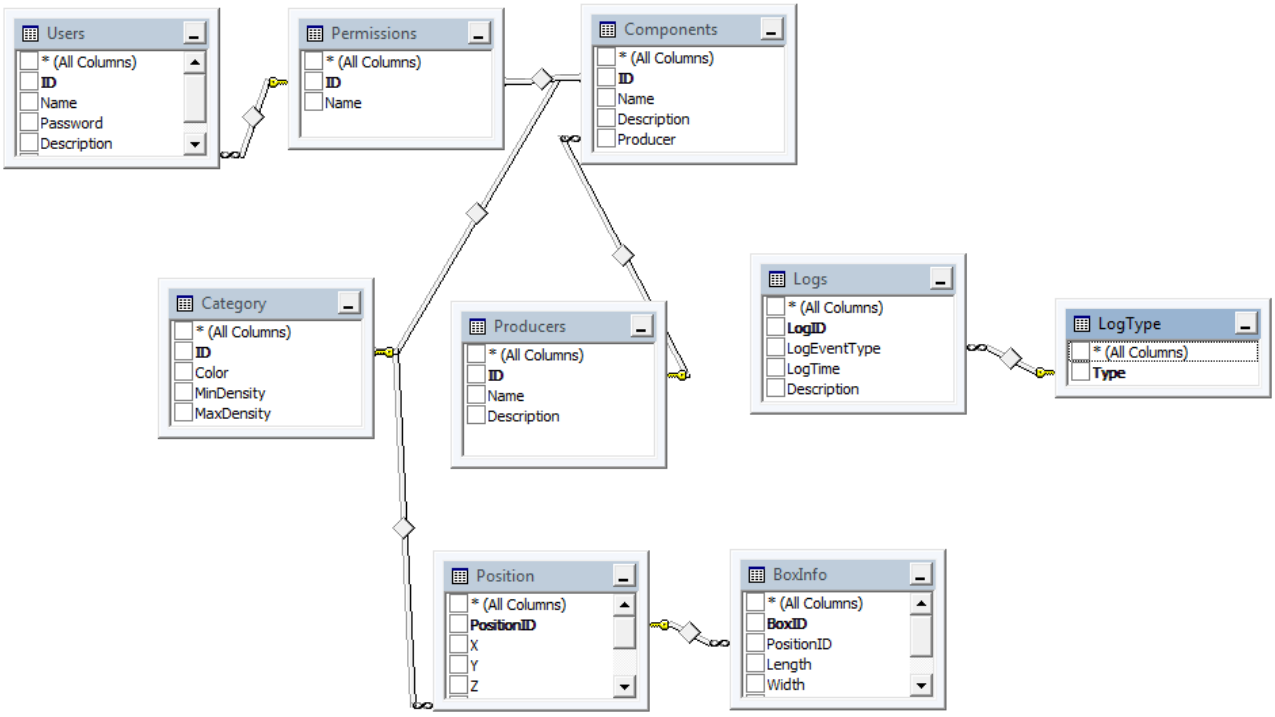


Illustration 1:Diagram for databasen

Som det kan ses i designet, så er der relationer mellem det hele undtagen Logs og LogType, der bruges i den forstand, at de hele tiden skal have direkte logning af informationer.

Grunden til denne løsning er, at det i starten af designløsningen var meget omfattende, men jo længere man trådte ind i, hvad der krævedes i systemet, kunne de forskellige tabeller og kolonner indskrænkes. Den 'gamle' løsning havde en masse ubrugte tabeller og ville blot fylde i databasen uden at blive brugt.

3.6 Testing

Vi har på dette semester beskæftiget os meget med software-testing metodikker, bl.a. unit-testing, integration-testing, continuous integration, versionsstyring og TDD. Mange af disse metodikker har vi anvendt i dette semesterprojekt til unit testing af vores klasser. Vi har introduceret seams vha. interfaces, der gør vi kan udskifte dependencies med vores egne "falske" objekter stubs/mocks, som isolation framework brugte vi RhinoMocks. Testning af koden gør den mere robust, samt at flere bugs opdages tidligt i udviklingsforløbet. Vi oplevede at kvalitetsfaktoren for koden, ved at bruge de førnævnte metoder, blev forøget.

Vores integrationstest er lavet efter "Big Bang" metoden. I denne fremgangsmåde er alle udviklede moduler koblet sammen til dannelse af et komplet softwaresystem og derefter anvendt til integrationstest.

"Big Bang" metoden er meget effektiv til at spare tid i integrations testprocessen, hvis testen ikke fejler. I tilfælde af at testen fejler, vil hele integrations processen være mere kompliceret og måske vil integrationstesten være mere tidskrævende end forventet. Der anbefales derfor at man enten bruger "Top-Down" eller "Buttom-Up" når man laver integrationstest.

Grunden til at vi valgte "Big Bang" er at vi var under tidspres og tog chancen (gamblede). Det har kostet os mange ekstra timer, fordi programmet fejlede.
For at finde fejlen begyndte vi med at integrationsteste med "Buttom-Up" metoden, det resulterede i at vi fandt fejlen og stoppede integrationstesten straks. Derefter kørte vi endnu en "Big Bang" integrationstest som var vellykket.

3.7 Udviklingsværktøjer

Kode:
Microsoft Visual Studio 2010 har været det primære værktøj under udviklingen af programmet, da programmet skulle implementeres i C#-sproget, samt skulle biblioteket WPF bruges som krav til programmet.

Unit testning:
Til det formål har vi brugt NUnit sammen med Rhino Mocks, samt har vi brugt ReSharper og dotCover samtidigt til refactoring og udføre testene, men er ikke musts for at kunne bruge NUnit og Rhino Mocks.

Design:
Til design er der blevet brugt alt fra papir og tavler ved de første skitser, men derefter til dokumenterende tegninger er der blevet brugt Visio, da det kunne integreres og laves med Visual Studio.

Version kontrol:
Til version kontrol har vi brugt Git sammen med Github.
Dette er valgt, da vi ville se fordelene ved at bruge noget andet end standard SVN, som vi har brugt i projekter før og ville så samtidig eksperimentere med fordelene ved branching.

Kommunikation:
Til kommunikation i gruppen, når en eller flere ikke var samlet, blev der brugt Skype og TeamViewer, da vi førhen har haft gode oplevelser med dem, og folk kan finde ud af at bruge dem.

Dokumentation:
Alle vores tekstdokumenter er enten blevet udviklet i LibreOffice eller OpenOffice, da vi har brugt dem før, samt alle kan bruge dem, da de er gratis.

Opgaver:
Til at holde styr på opgaver, der skal laves i løbet af et sprint, har vi brugt værktøjet

ScrumWise[1], som er et online værktøj til Scrum. Anvendelse for studerende er gratis, hvis blot man skriver til personen, som har udviklet værktøjet.

3.8 Resultater

Den 1 juni endte vi op med et resultat, der bestod i, at robotten kunne sortere forskellige klodser af forskellige massefylder. Dette vil sige, at de forskellige dele i systemet virket som de skulle. Vægten sørger for at måle klodserne, databasen sørger for at persistere de data vi får ind, samt logge eventuelle hændelser.

3.9 Diskussion

I projektet har vi haft mulighed for at afprøve den teoretiske del og den praktiske del fra kurserne DAB1, WIN1, INF1 og SWT1.
Vi har arbejdet med databaser, programmering af den grafiske brugergrænseflade, implementering af en vægtforstærker og udført diverse enhedstest.
Under hver af fagene har det været dels interessant at anvende det brugte undervisningsmateriale til at opfylde produktets krav – dette gælder bl.a. andet, at der er tilgang til en database, hvorved brugeren kan se data, og at der er en grafisk brugergrænseflade implementeret og designet i WPF. En brugbar ting gennem software test har været at teste de enkelte funktioner/enheder hver for sig, hvorefter der foretages en integrationstest af systemet. Dette trak dog lidt ud, da det til tider endte i problemer at kunne teste de forskellige ting.
Der er endvidere under udviklingsforløbet til tider anvendt TDD til XP-programmering, og dette har fungeret godt, men det kan diskuteres, om det kom for sent i undervisningen på trods af, at det blev anvendt før kurset SWT1 introducerede det.

3.10 Fortræffeligheder

I forbindelse med projektet, har der været nogle ting, som vi synes der er værd at fremhæve, herunder kode, men samtidig også hvordan vi har valgt at gribe nogle ting an på, og hvad vi har gjort for at til sidst opnå vores endelige mål.

Kodemæssigt er factory værd at fremhæve. Factory står for at oprette singleton af ThreadHandling, Robot, Simulator osv. Den er altså derfor en central del af projektet, da en stor del bliver oprettet igennem den. Blev brugt meget for at gøre tilgang til klasserne

1 www.scrumwise.com

nemmere, samt sørge for der kun bliver brugt en version af hver.

Manual styring, da var meget effektiv til test af funktioner og robotten. Blev også senere brugbar i udviklingen af script til automatisering, da man kunne indstille robotten til positionerne, man ville flytte klodserne til for at få koordinaterne.

Vi har brugt IronPython som DSL. Python gør det nemt for vores brugere selv at interagere med systemet og få robotten til at få udført den ønskede funktionalitet, da sproget har en nem syntaks.

Derudover vil vi fremhæve Intellisense, som vi er stolt af. Vi har et .txt dokument, som vi bruger til at tilføje nye funktionsnavne og keywords til vores intellisense. Når der bliver skrevet i vores IDE i RoboGO, vil Intellisense forslå funktionsnavne eller keywords, som den kender til via dokumentet. Hvis man laver et ny script ind på vores IDE, er det en fordel, at man altid lige kan se, hvilke muligheder man har, og hvilke parameter man kan sende med.

For at varetage et projekt af denne størrelse og ende op med noget, der virker, har vi brugt softwareudviklingsmetoden UP, som fungerer rigtigt godt, og som der har været erfaringer med. Der blev sørget så tit som muligt for at holde daily scrum, hvilket har bevirket, at vi hele tiden vidste, hvad hver den enkelte person i gruppen gik og lavede, de enkelte gruppemedlemmer kunne altså holde øje med, hvilke problemer folk havde, således de hurtigt blev taget hånd om. Udover de daglige Scrum møder, holdt vi også retrospektiv efter hvert sprint. Vi evaluerede altså hele tiden os selv, forbedrede punkter hvor vi haltede som gruppe. Vi blev derfor løbende som gruppe bedre og bedre.

3.11 Opnåede erfaringer

Vi har igennem projektet fået flere nyttige erfaringer. Både for kodeskrivning til måden vi planlagde vores arbejde. Vi har selvfølgelig alle fået mere erfaring med at skrive kode, men i et gruppeperspektiv er vi i løbet af projektet specielt blevet bedre til at gøre vores kodebase mere homogen.
Dette har igennem de forskellige sprints været sådan, at der holdes et møde i slutningen af hvert sprint, hvor der snakkes om forbedringer, det dårlige og det gode. Dette har i særdeleshed været godt, da gruppen kan opremse, hvad der kunne være af

problematikker.
Det endte ud med, at der skulle laves en del forbedringer, hvorved gruppen kunne holde sig mere optimale til at arbejde med projektet.
I starten blev der planlagt mange møder, og efterfølgende blev der bestemt, at der skulle anvendes Scrum som framework. Dette var effektivt, idet man kunne sætte sig på en opgave, sende den videre til en anden og gå i gang med den næste. Mere arbejde, mindre spildtid.
Der har også været tider med upræcise mødetider fra gruppens medlemmer, hvilket har resulteret i manglende motivation, koncentration og til tider spild af værdifuld tid.
Det kan derfor overordnet siges, at gruppens opnåede erfaringer har været, at vi i projektet er blevet gode til at styre tiden, men pga. manglende fokus har gruppen været ude af stand til at fuldføre diverse opgaver, det enkelte gruppemedlem er blevet tildelt.

3.12 Forbedringer

I gruppen

Efter en afstemning i gruppen valgte vi at bruge Git, herunder Github som der er blevet anvendt til versionstyring. Vi havde muligheden mellem Git eller Subversion og grunden til, at vi valgte Git, er at vi ville prøve at benytte en decentraliseret form for versionsstyring i kontrast til Subversion. At den er decentraliseret betyder at hver gruppemedlem har et lokalt repository de uploader til, dette kan gøres offline og man kan derefter pushe til Github, når man har fået forbindelse til internettet. Mange af gruppens medlemmer synes dog at Git i modsætning til Subversion er sværere at bruge. Idet gruppen har vænnet sig til at anvende Subversion under hele tredje semester, så har visse medlemmer i gruppen haft problemer med at finde ud af, hvilken rækkefølge man skal anvende til at uploade sine filer. Visse gruppemedlemmer har brugt en del tid på at lære at bruge det, og det er pga. manglende øvelse, at der har været en del merge problemer. Ud fra dette eksperiment med anvendelse af Git er vi nået frem til, at det skal indlæres bedre.

Begrænset mulighed for fysisk test:

Der har været stor interesse for robotten, derfor har der været begrænset adgang til den. Det har resulteret i, at vi har brugt lange nætter på skolen for at få robotten for os selv, så vi kunne teste og debugge. Man kunne godt have flere robotter (mindst en mere), når en hel klassen arbejder på den samme fysiske enhed. Den anden begrænsning ligger på, at

visse grupper har 'bestemt', at i slutningen af projektet er det muligt at reservere robotten i længere end en halv time, og der er sågar nogen, der har reserveret i op til to timer!

Produktet

Der kunne godt laves designmæssige ændringer på vores UI, så man fik et moderne UI. Flere tilføjelser til IDE'en såsom linjenummer. Mulighed for at debugge direkte i vores IDE og en forbedret simulator. Hvis vi skulle lave en ny version af vores nuværende RoboGO program vil det helt klart være disse punkter, som vi vil starte med at forbedre, med mindre robotten får nye funktionaliteter.

4. Konklusion

Vi synes vi er endt ud med et produkt der har god funktionalitet. Det er lykkedes for os, at udvikle et automatiseret system til Scorbot ER-4u robotten, som er i stand til at sortere klodser af forskellige massefylder. Udover dette blev der til systemet udviklet en brugervenlig brugergrænseflade, hvori vi har alle relevante informationer tilgængelige, såsom informationer omkring de forskellige klodser som er persisteret i vores database. Den automatiseret proces kan nemt varetages med vores DSL, idet brugeren kan skrive scripts deri, afhængig af hvilken proces han ønsker fuldført, samtidig bliver han hjulpet på vej af vores Intellisense. Udover automatisering, har vi også en manuel del, hvilket gør det nemt at bevæge robotten rundt.

5. Referencer

Se det litteratur, der er anvendt i kravspecifikation.