

./Billeder/0_1_ForsideBillede.jpg

SILVER BULLET SORT

4. SEMESTERPROJEKT

Procesrapport for Silver Bullet Sort

Author:

Gruppe 5

Supervisor:

Poul Ejnar ROVSING

14. december 2013

Versionshistorie:

Ver.	Dato	Initialer	Beskrivelse
1.0	7-05-2012	RBH	Opsætning af skabelon til rapportskrivning
1.1	19-05-2012	MBH og CSJ	Udkast til de fleste afsnit påbegyndt.
1.2	21-05-2012	CSJ	Forbedringer og fortræffeligheder tilføjet
2.0	28-05-2012	Gruppe 5	Rapporten mangler kun konklusion, små tilføjelser og rettelser
3.0	29-05-201	Gruppe 5	Endelig Rapport til aflevering

Godkendelsesformular:

Forfatter(e):	Michael Bojsen-Hansen (MBH) Kasper Vinther Andersen (KVA) Lars Anker Christensen (LA) Lasse Hansen (LH) Christian Smidt-Jensen (CSJ) Rasmus Bækgaard (RAB) Christoffer Lousdahl Werge (CW) Lasse Sørensen (LS)
Godkendes af:	Poul Ejnar Roving.
Projektnr.:	4. semesterprojekt.
Filnavn:	Procesrapport.pdf
Antal sider:	??
Kunde:	Poul Ejnar Roving (PER).

Sted og dato: _____

09421 Lasse Lindsted Sørensen

10063 Lasse Hansen

10648 Lars Anker Christensen

10719 Michael Bojsen-Hansen

10750 Kasper Vinther Andersen

10770 Christian Smidt-Jensen

10832 Christoffer Lousdahl Werge

PER Poul Ejnar Røvsing

10893 Rasmus Bækgaard

1 Resumé

I forbindelse med 4. semesters eksamensprojektet er der blevet udviklet styring til en robotarm, der tillader at sortere klodser efter materialetype. Systemet giver desuden brugeren mulighed for at udvikle simple programmer, og teste disse på både den fysiske robot, såvel som til en simuleret version af denne. For at sikre persistering af data, samt give brugeren mulighed for at gemme alle hændelser i systemet, er en relationel database blevet implementeret. Det overordnede kodesprog i projektet er C# og som udviklingsmiljø er Microsoft .NET framework blevet anvendt. Under .NET frameworket er WPF blevet brugt til, at udvikle den grafiske brugergrænseflade. Som projektstyring er der bl.a. brugt dele af Scrum, hvor principper fra Extreme Programming og GRASP er blevet fundet brugbare med henhold til kodeskrivning.

Det er lykkedes at udvikle et program, der gør robotten i stand til at sortere klodser efter materialetype. Derudover kan der via en udviklet IDE, skrives programmer til selvbestemte formål. Desuden er programmet blevet udstyret med nogle ekstra funktioner, som giver brugeren en række muligheder med henhold til opsætning af systemet og manipulering med data.

In relation with the fourth semester project, a controlsystem to a robotic arm has been developed. The purpose of the system is to sort bricks in according to their material. Furthermore the system holds a facility that allows the user to develop simple programs and test these on both the real robot as well as a simulation of the robot. To guarantee persistence of the data and allow the user to save events from the system, a relational database has been implemented. The primary programming language used in this project is C# and the development environment used was Microsoft's .NET framework. Within the .NET framework WPF has been used to develop the graphical user interface. As project management method parts of Scrum have been used where principles from Extreme Programming and GRASP have showed useful when writing the source code.

A system which was able to sort bricks according to material type was implemented. Additional, an integrated development environment was developed which allows the programmer to create custom software for the robot. Furthermore functions that gives the user opportunity to setup the system and manipulate the data have been added.

Indhold

2 Forord

Dette projekt er udarbejdet af otte studerende på Ingeniørhøjskolen i Aarhus. Projektet er et eksamensprojekt på fjerde semester for IKT-linjen. Det er primært udarbejdet på baggrund af undervisning fra fag på fjerde semester; Windowsprogrammering, Databaser, Softwaretest og Interfacing. Således danner viden fra disse fire fag, samt fag fra de tre foregående semestre, baggrund for udviklingen af dette semesterprojekt.

3 Indledning

Sortering af elementer kan være en vigtig opgave i industriverdenen, og en automatisering af denne mekanisme kan spare meget arbejdskraft og dermed spare både hænder, penge, fejl, arbejdstimer og arbejdsskader. Dette dokument omhandler udviklingen af et softwareprodukt, der kan kontrollere en robot, samt gøre denne i stand til at sortere en række klodser alt efter deres materialetype. Det er en forholdsvis lille robot, der kun kan sortere små klodser, men det samme software ville kunne modificeres til brug på en større robot, som er i stand til at sortere så store elementer, at de ikke ville kunne flyttes med håndkraft.

Program, Silver Bullet Sort, er blevet udviklet til at håndtere ovenstående problemstilling for et firma, der har indkøbt en Scorbot ER-4u¹ med tilhørende programmel. Systemet skal fungere således, at en klods registreres på et transportbånd, hvorpå robotten måler og vejer klodsen. Herefter placerer klodsen i et materialespecifikt rum i en boks, der indeholder klodser med den givne materialetype. Programmet er lavet på baggrund af en række krav fra et udstedt produktoplæg. Desuden har en række møder med virksomheden også været med til at danne grundlag for fastlæggelse af programmets endelige funktionalitet. Programmet skal indeholde en database, som kan persistere diverse data i systemet, herunder data fra de elementer, der er blevet sorteret samt logge informationer om, hvordan processen er forløbet. Ligeledes skal programmet indeholde en IDE, som gør det muligt at lave brugerdefinerede programmer til sortering af elementer. Denne kan implementeres på flere måder, men den anbefalede metode indbefatter at omprogrammere robotten ved brug af en række funktioner, som firmaet har udleveret til udviklerne. Dog er disse

¹http://www.intelitek.com/admin/Products/uploads/File/File1_17.pdf

funktioner meget dårligt dokumenteret, så det har været op til udviklerne at undersøge anvendelsen af dem. Derudover skal det være muligt, at simulere robotten således, at et program kan testes, når robotten ikke er til rådighed. Desuden var det et krav, at der skulle laves en brugergrænseflade, hvorigennem alt kommunikation med robotten kunne kontrolleres. Det eneste hardware-mæssige krav kunden havde var, at få udviklet en vægt på baggrund af en vejecelle, som skulle indgå i processen, når materialetypen skulle findes.

Inden selve koden blev påbegyndt udviklet, blev der brugt en del ressourcer på at få tilegnet sig en fundamental viden om det, fra producenten, udleverede hardware og software. En domænemodel blev lavet på baggrund af kravene, hvilket skulle give et overblik over de specifikke dele i projektet. Herefter blev en kravspecifikation udarbejdet. Opbygning med at fastlægge de fundamentale krav før der blev udviklet kode, er primært kendt fra udviklingsmetoden *Unified Process*.²

For at det endelige produkt skulle stemme overens med firmaets krav og ønsker, er der gennem hele arbejdsprocessen holdt mange møder med virksomheden samtidig med, at virksomheden undervejs er blevet præsenteret for noget fungerende software, så det fra firmaets side har været muligt, at komme med ønsker til forbedringer og modificeringer af det endelige produkt. Da projektet er udviklet ved hjælp af agile arbejdsmetoder og principper, herunder primært *Scrum*³ og *Extreme programming*⁴, er det en naturlig proces at ændre kravene undervejs og revurdere projektet alt efter kundens behov.

Begreber og forkortelser

- UC: Use Case
- IDE: Integrated Development Environment
- UP: Unified Process.
- DLL: Dynamic-link library

²http://en.wikipedia.org/wiki/Unified_Process

³[http://en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))

⁴http://en.wikipedia.org/wiki/Extreme_programming

- WPF: Windows Presentation Foundation
- MVVM: Model View ViewModel

Læsevejledning

Nedenfor er listet en kort beskrivelse af hvert afsnit i dette dokument:

- Abstract og Resumé
 - Disse afsnit giver en kort beskrivelse af projektet på både dansk og engelsk.
- Indledning
 - Dette afsnit fortæller baggrunden for projektet, kravene til projektet, samt hvilke arbejdsmetoder og processer, der er anvendt. Herefter følger begreber og forkortelser samt en læsevejledning.
- Projektafgrænsning
 - Afsnittet fortæller kort om de begrænsninger, der er blev lavet, da projektet blev fastlagt.
- Projektgennemførelsen
 - Her præsenteres de forskellige iterationer, hvorigennem projektet er udviklet, samt erfaringerne med disse.
- Metoder
 - Dette afsnit beskriver de forskellige arbejdsmetoder, der er anvendt i udarbejdelsen af det endelige produkt.
- Specifikation- og analysearbejdet
 - Analysearbejdet, der ligger til grund for opbygningen af projektet, præsenteres i dette afsnit. Til dette hører kravspecifikation, domænemodel osv.
- Designprocessen

-
- Her beskrives selve designprocessen og erfaringerne med denne. Hertil hører lagdeling, klassediagrammer osv.
 - Udviklingsværktøjer
 - Dette afsnit giver en beskrivelse af de mest væsentlige udviklingsværktøjer, der er blevet brugt til udviklingen af projektet.
 - Resultater
 - De mest væsentlige resultater præsenteres objektivt i dette afsnit.
 - Diskussion af opnåede resultater
 - I dette afsnit diskuteres der på baggrund af de opnåede resultater.
 - Opnåede erfaringer
 - Dette afsnit giver en beskrivelse af de opnåede erfaringer gruppen har gjort sig, primært på baggrund af selve arbejdsprocessen.
 - Projektets fortræffeligheder
 - Her præsenteres de dele af projektet, som er blevet fundet særdeles velfungerende.
 - Forslag til forbedringer af projektet eller produktet
 - Afsnittet giver en beskrivelse af, hvad der kunne have været forbedret i selve produktet eller produktudviklingen.
 - Konklusion
 - Der laves her en konklusion på baggrund af de opnåede resultater og diskussionen af disse, samt på baggrund af de opnåede erfaringer.
 - Referencer
 - Afsnittet lister de forskellige materialer og værker, der refereres til.

4 Projektafgrænsning

Projektet tager udgangspunkt i det udleverede produktoplæg⁵ og herfra er der ikke blevet foretaget synderligt mange afgrænsninger af projektet, udover småændringer, der er kommet i forbindelse med kontakt med firmaet. I produktoplægget optrådte der dog referencer til både et kamera, stemmestyring og en såkaldt manualpandant. Stemmestyningen og kameraet viste sig at være et levn fra tidligere projekter og blev derfor skrottet, uden der var tale om en egentlig afgrænsning. Den såkaldte manuelle pandant blev erstattet af IDE-delen, igen uden der er tale om en egentlig afgrænsning.

5 Projektgennemførsel

Gennem projektet er der fokuseret på mange og korte iterationer. Dette har resulteret i 6 iterationer med en varighed på 14-20 dage, hvilket har hjulpet med til at dele projektet op i mindre dele.

Desuden var ønsket, at hver iteration skulle resultere i en implementeret funktionalitet f.eks. en IDE eller muligheden for at sortere en enkelt klods. Fordelen ved denne fremgangsmåde er, at der kunne planlægges på erfaringer og feedback fra tidligere iterationer samtidig med, at det ville give en forsikring om, at projektet ville blive færdigt til tiden. I praksis var dette dog svært at overholde, da der af og til opstod problemer, som forhindrede at funktionaliteten var færdig ved fuldendt sprint. I nogle sprint var dokumenteringen af projektet sat i højsædet, så derfor var der ikke decideret moment for at fremvise funktionelt kode.

Nu skal det ikke hede sig, at der slet ikke blev afholdt nogle af Scrum's sprint reviews. Det bør nævnes, at der blandt andet blev holdt en demonstration af b.la. vægt-komponenten for kunden, da denne var færdig og funktionelt klar. Desuden er der løbende blevet vist kørende software, samt præsenteret de færdige resultater omkring componenten fra sprintet. Tilmed kan det nævnes at små delmål i sorteringen af en klods er blevet forelagt for kunden.

⁵Produktoplægget er vedlagt som bilag under "Reference Dokumenter".

5.1 Sprint 1

Sprint 1 fungerede som en typisk elaboration/inception fase fra udviklingsmetoden Unified Process. Her blev sprintet naturligvis brugt til at undersøge, hvad projektet i bund og grund gik ud på, så gruppen fik syn for sagen. Herefter blev produktoplægget gransket til hudløshed, og der blev udformet krav som projektet er bygget på. Med et overblik over kravene blev det muligt, at lave en domænemodel og aktørbeskrivelser, samt planlægge sprintet (vist på figur ??).

Dernæst blev det første udkast til kravspecifikationen påbegyndt.

Figur 1: Burndown chart for sprint 1 Billeder/Sprint1_burn.png

Her benyttede gruppen Use Case-teknikken for at finde aktører og deres mål. Undervejs blev selve robotten udforsket, så det var nemmere at danne et overordnet overblik over hvilke funktionaliteter der var nødvendigt at implementere for at opretholde kravene.

Med et godkendt Use Case-diagram blev en Accepttest påbegyndt og undervejs blev det ligeledes opdaget, at der manglede Use Cases, der tog højde for andre scenarier.

I dette sprint blev designet af databasen også påbegyndt.

5.2 Sprint 2

I sprint 2 blev databasen designet med tabeller og system sekvens diagrammer.

Figur 2: Burndown chart for sprint 2 Billeder/Sprint2_burn.png

Der blev også åbnet op for de .dll-filer, som der skulle bruges til at tilgå robotten, og et simpelt program blev lavet, så de forskellige funktioner kunne tilgås. Yderligere dokumentation blev udformet til Use Case-diagrammet og Domænemodellen blev opdateret til, at passe med Use Case-diagrammet.

Kravspecifikationen blev også opdateret og var tæt på et færdigt stadie, hvorefter Systemarkitekturen blev påbegyndt.

5.3 Sprint 3

I Sprint 3 blev Accepttesten rettet til, så den passede med Use Case-beskrivelserne samtidig med, at nye emner blev påbegyndt:

Her kan nævnes at der blev påbegyndt en IDE til at definere grup-

pens eget programmeringssprog. Efter at have diskuteret programmeringssproget til denne, endte valget med at falde på *Iron Python*, hvilket også gav muligheden for at implementere *Intellisense*.

Figur 3: Burndown chart for sprint 3
Billeder/Sprint3_burn.png

Vægten der skulle bruges til at finde klodsernes densitet blev påbegyndt, og der blev lavet et udkast til, hvordan simuleringen af robotten kunne se ud.

Derudover blev et funktionelt program lavet, der kunne tage klodser fra transportbåndet og flytte dem til vægten. Alle disse ting blev fremvist for kunden i et sprint review.

5.4 Sprint 4

Sprint 4 blev kort, grundet påskeferie. Men sprintet gik primært ud på udvikling af programmer og algoritmer. Meget få emner blev færdiggjort, da mange af emnerne havde særdeles store udfordringer, der ikke var forudset.

Følgende gav de største problemer:

Figur 4: Burndown chart for sprint 4
Billeder/Sprint4_burn.png

- Algoritme til at kunne rykke på transportbånd.
- Database synkronisering.
- Det funktionelle programs funktioner i en sekvens.
- Vægten fejlede grundet defekt fumlebræt.

Det resulterede i, at sprintet og flere af dets opgaver blev flyttet til Sprint 5.

5.5 Sprint 5

I sprint 5 blev programmet samlet. Dette var en længerevarende proces, som primært bestod af, at samle de forskellige mindre projekter og pakker, som udgjorde det store projekt.

Med de funktionelle komponenter samlet, blev GUI'en omskrevet

via MVVM-mønstret⁶, så forretningslogikken og præsentationslaget blev bundet sammen. Dette kostede en del tid.

Samtidig opstod der nye udfordringer i projektet mht. de nye komponenter, som ligeledes kostede nogen tid at debugge.

Med komponenterne samlet og test af programmet påbegyndt, nærmede deadline for aflevering sig og en skabelon til selve rapporten, godkendt af kunden, blev udformet.

5.6 Sprint 6

Sprint 6 blev fastsat til de sidste 3 uger, inden afleveringsfristen. Der blev foretaget en mere detaljeret analyse af, hvad der manglede for den endelige aflevering, og gruppen aftalte, at der de kommende uger skulle arbejdes lidt længere end normalt for, at sikre at produktet levede op til kundens forventninger, behov og ikke mindst deadline.

Derfor blev der brugt tid på, at få de sidste funktionaliteter i pro-

grammerne finpudset samtidig med, at de sidste dele blev implementeret. Et designproblem med databasen og dennes samspil med log-funktionaliteten udløste en mindre udfordring med systemet, men efter nogle diagrammer over de enkelte filers afhængigheder var skitseret, blev udfordringen løst relativt gnidningsfrit.

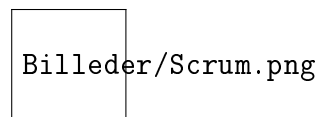
De sidste 14 dage stod primært på dokumentering af systemet. Her blev accepttesten tilrettet efter kravspecifikationen, samtidig med at processrapporten og designdokumentet for alvor blev udviklet.

⁶http://en.wikipedia.org/wiki/Model_View_ViewModel

Der var også planlagt, at der de sidste 14 dage skulle være et "kodestop", dvs. at alt udvikling af kode skulle ophøre. Men da nogle endelige tests blev udført viste det sig, at der var mangler i programmet, så systemet ikke overholdt acceptttesten. Der blev derfor brugt yderligere 3 dage for 2-3 mand på, at teste koden for småfejl og få disse rettet.

6 Metoder

Som nævnt i indledningen, er der blevet arbejdet under det agile processframework *Scrum* i dette projekt. Alle i projektgruppen har arbejdet med *Scrum* i tidligere semestre, generelt med meget gode erfaringer. Derfor er *Scrum* blevet valgt som udviklingsprocess-ramme.



Figur 7: Scrum model

Figur ?? er en model af nogle artefakter indenfor *Scrum*. Modellen afspejler også, hvordan der er blevet arbejdet under *Scrum* i projektgruppen. Der er blevet arbejdet i sprints, (se *afsnit 5 Projektgennemførsel*) og i hvert sprint har der været en sprintplanlægning. Her er der blevet udtaget de vigtigste opgaver fra productbackloggen, hvor de overordnede opgaver i projektet er listet, og derudfra dannet en sprintbacklog med opgaver der skal løses i sprintet.

For at holde øje med sprintets process, er et burndown-chart blevet benyttet, som giver et overblik over, hvor langt sprintet er i forhold til, hvor langt det burde være. Burndown-chartet er genereret i værktøjet *Scrum Wise* (se *afsnit 9 Udviklingsværktøjer*). Det skal nævnes, at *Scrum Wise*'s burndown-chart-værktøj er lavet ud fra et estimat om, hvor lang tid hver opgave tager, men chartet tager ikke højde for, at der er flere udviklere i projektgruppen. Chartet kan dog sagtens bruges til at se om sprintets opgaver overholder den givne deadline.

Hver dag har der været et Daily Scrum Meeting, hvor hvert gruppemedlem har præsenteret, hvad der er blevet arbejdet med siden sidst, og hvilke problemer der har været. Dette sikrer en hvis transparens og inspektion, så gruppen kan tilpasse sig mulige udfordringer. Desuden blev der også ved disse møder diskuteret store fælles beslutninger, som var meget

afgørende for projektet, det kunne f.eks. være selve opbygningen af lag, og hvordan de forskellige klasser skulle spille sammen.

Efter hvert sprint har der ligeledes været et sprint retrospective møde, hvor det forløbne sprint blev evalueret, både med hensyn til arbejdsprocessen og selve det resulterende produkt fra sprintet. Dette gav feedback, som kunne bruges i planlægningen af det næste sprint.

Under selve implementeringen af koden var metoder fra Extreme Programming til stor hjælp - navnlig det at kode i par, blev anvendt. Her skulle den ene kode, mens den anden stillede kritiske spørgsmål til, hvad modparten lavede. På den måde blev koden løbende revideret, og både ansvar og viden blev delt ud.

Ligeledes blev der nævnt i indledningen, at der i projektet er udviklet efter software-udviklings frameworket UP. Inception- og Elaborationfasen har primært bestået af, at fastsætte de overordnede krav til projektet, samt at få adgang til robotten og få den til at udføre simple funktioner. Det var nødvendigt at få en forståelse for robotten og dens virkemåde, før det var muligt at fastlægge en række krav til den endelige funktionalitet. Constructionfasen, som har fyldt størstedelen af projektudviklingen, har fungeret således at der er blevet implementeret ny funktionalitet til robotten (fra produktbacklog), mens det sideløbende er blevet dokumenteret og revurderet. Transitionfasen, som er den sidste fase, har primært bestået i at produktet er blevet finpudset og udgivet (afleveret) samtidig med, at det sidste dokumentation er blevet skrevet.

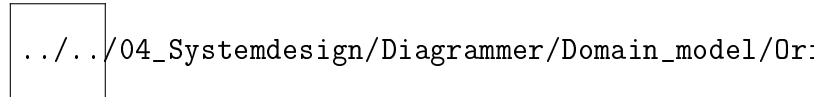
Til at verificerer koden, er der blevet benytte automatiserede tests⁷. Der er lavet tests til de vigtigste dele af projektet, og disse dele omhandler selve kommunikationen til robotten, gennem kald til funktionerne i DLL-filen⁸. Derudover er der ligeledes lavet test af kommunikationen til vægten. Det havde været optimalt at lave tests af hele systemet, men dette blev nedprioriteret i forhold til at få en funktionelt produkt med mange funktioner og en god opbygning. Desuden blev programmets mere overordnede funktionalitet testet gennem selve accepttesten, og grundet dette og tidspress, blev der ikke afsat tid til integrationstest, selvom det selvfølgelig havde været optimalt.

⁷Se *NUnit* under afsnit 9

⁸USBC.dll: fil der indeholder de funktioner, der kan udføres på robotten.

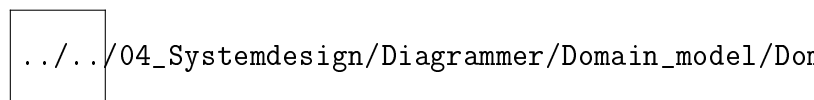
7 Specifikations- og analysearbejdet

Ved påbegyndelse af projektet blev der udarbejdet en domænemodel, for at skabe et overblik over systemet. Denne blev dannet ud fra de informationer, der blev givet i produktoplægget og var derfor langt fra komplet (modellen ses på figur ??).



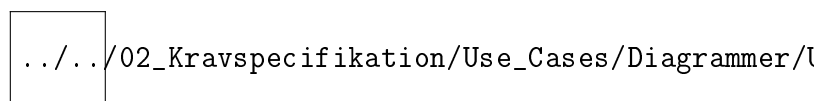
Figur 8: Første udkast af domænemodellen

Efter tilegnelse af mere viden blev domænemodellen udvidet og specificeret. Der blev derudover gjort overvejelser om, hvordan de forskellige elementer interagerer med hinanden. Dette ledte til en model med den første anskuelse af det system, der skulle udvikles (modellen ses på figur ??)



Figur 9: Andet udkast af domænemodellen

Ved hjælp af denne domænemodel, blev der udtænkt brugssituationer af systemet samt overvejet, hvilke aktører systemet ville have. Under denne proces blev et Use Case diagram udviklet, hvilket gav et overblik over disse brugssituationer samt hvilke aktører, der interagerer med Use Casene (se figur ??).



Figur 10: Use Case diagram

Use Case diagrammet der ses på ovenstående figur, er det endelige diagram. I takt med at systemet er blevet udviklet, er der blevet gjort erfaringer og indsamlet viden. Denne viden har gjort, at diagrammet er blevet forbedret og dermed afspejler det system, der blev udviklet.

Ud fra Use Casene i diagrammet, samt kravene i produktoplægget, blev der udformet en kravspecifikation. Her blev alle kravene til Use Casene specificeret, og disse indbefatter

bl.a. scenarier af, hvorledes Use Casene opfylder de mål, de er sat til. Denne kravspecifikation er ligesom Use Case diagrammet så vidt muligt blevet holdt opdateret. Dette er naturligvis gjort i samarbejde med kunden.

Til at bekræfte, at kravene der blev stillet er opfyldt, blev der specificeret en accepttestspecifikation, der beskriver hvordan alle kravene skal testes. Denne er ved færdiggørelsen af systemet kørt igennem sammen med kunden.

Før udvikling på hver del af systemet blev påbegyndt, er der blev gjort betragtninger over, hvorledes systemet snakker sammen med lag og aktører via simple sekvensdiagrammer. På denne måde er et overblik før udvikling blevet dannet, hvilket har hjulpet til en mere solid og problemfri udvikling. Dette er kun gjort ved de dele af systemet, hvor det er fundet fordelagtigt.

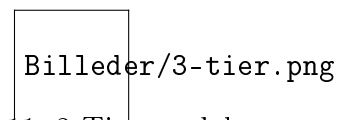
Generelt har analyse- og specifikationsarbejdet fungeret rigtig godt. Overblikket over domænet og systemet har været bibeholdt gennem hele projektet. Det skyldtes bl.a., at designkritiske beslutninger løbende er blevet diskuteret af hele gruppen, hvilket medførte at fokus på den rigtige funktionalitet blev bibeholdt. Det skal dog nævnes, at der også er blevet udviklet ekstra funktionaliteter, der ikke direkte er specificeret, da der ikke er fundet tid og overskud hertil. Måden hvorpå kravene er delt op i Use Cases, har også gjort det muligt at opdele opgaverne blandt gruppemedlemmerne på en god og effektiv måde.

8 Designprocessen

Da udkastet til en kravspecifikation var færdig, skulle der naturligvis fastsættes nogle rammer for,

hvordan systemarkitekturen skulle være. Det overordnede design blev fastlagt som det første. Der var flere overvejelser og diskussioner om dette design, da det danner grundlag for hele systemet. 3-Tier modellen endte med at blive valgt som model for systemet (se figur ??).

Denne model blev fundet simpel og solid, hvilket er to egenskaber der er blevet prioriteret



Figur 11: 3-Tier model

højt i systemet. Grundidéen i modellen er at opdele systemet i uafhængige moduler, og som følge af det, vil der komme en lav afhængighed samt en høj samhørighed, hvilke er mål, der tilstræbes i ethvert it-system. Dette resulterer også i at et lag, som for eksempel grænsefladen, kan udskrives, uden at hele systemet skal ændres. Disse elementer samt simpelheden gjorde, at valget af den overordnede model faldt på 3-Tier modellen.⁹

Da 3-Tier modellen blev valgt, var der ikke mange overvejelser omkring, hvilken model brugergrænsefladen skulle udvikles efter. MVVM er et meget benyttet mønster inden for WPF, og er også blev benyttet til opbygning af dette systems brugergrænseflade. Opdeling af logikken og det grafiske, samt binding imellem disse, stemmer godt overens med 3-Tier modellen, hvilket også er hovedgrunden til at dette mønster er valgt. Derudover sikres en god opdeling af koden til brugergrænsefladen med dette mønster, og det grafiske kan dermed for eksempel nemt skiftes ud. Det opretholdes ved at GUI elementerne i Viewet's data er bundet til properties i MVVM's viewmodel og ikke hardcodet til modellen. Dermed er der kun afhængigheder nedad i systemet.

Tilgangen til databasen, blev i første omgang lavet som en ren tilgang. I videreudviklingen af denne tilgang, blev der diskuteret, hvilken metode der vil være bedst. Det endte ud med en beskedkø, hvor databasekaldene blev gemt. Denne kø underretter videre til databasetilgangs koden via *observer*-mønstret. Køen sikrer, at systemet kan køre videre indtil databasen skal kontaktes, selvom forbindelsen til databasen skulle mistes.

I hele koden er der generelt gjort stor brug af interfaces. Dette er gjort, da interfaces gør det nemt at teste kode, og da interfaces bruges i forbindelse med mange forskellige designmønstre¹⁰, samtidig med, at det bliver muligt at skifte objekter ud på run-time f.eks. vha. strategy mønstret. Et andet eksempel på et designmønster, er singleton. Dette designmønster sikrer at en bestemt klasse kun kan oprettes en gang, og dette er b.la. brugt på klassen RobotEvent.¹¹ Flere klasser benytter denne, til at fortælle, at der er sket en hændelse med robotten, men da der kun er en robot, skal det naturligvis sikres at samme objekt bliver benyttet.

⁹Se afsnit 5 *LOGISK VIEW* i systemarkitekturdokumentet, for en detaljeret model af systemet

¹⁰Se afsnit 10.2 Arkitektur mønstre i systemarkitekturen for oversigt over designmønstre.

¹¹Se afsnit 8.2.2 Komponent 2: Simulering i systemarkitekturen for information om klassen RobotEvent

I det hele taget er forskellige designmønstre blevet benyttet, til at sikre en solid og god kode.

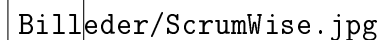
De forskellige designløsninger der undervejs er blevet overvejet og diskuteret, har gjort at et solidt, fleksibelt og vedligeholdelsesvenligt program så vidt er opnået.

9 Udviklingsværktøjer

Som nævnt har den primære udviklingsprocess til dette projekt været *Scrum*.

Under hele forløbet er værktøjet *Scrum Wise*¹² blevet brugt til at håndtere projektbackloggen og sprintbackloggen. Værktøjet er online-baseret, og det er meget nemt at gå til. Der er mulighed for at oprette, redigere, afslutte og sætte statussen på de forskellige opgaver, via det indbyggede taskboard. Derudover er der en burndown-chart, der gør det nemt at overskue, hvordan det går med det aktuelle sprint.

Dette værktøj har gjort det nemt at benytte *Scrum* som udviklingsprocess og virkelig givet den gennemsækelighed i projektetprocessen, hvilket er en af grundpillerne i *Scrum*. Et lille eksempel på *Scrum Wise*, kan ses på nedenstående billede (Ligger i stor størrelse i bilaget under Billeder/Scrum):

The image is a placeholder for a screenshot of the Scrum Wise tool interface, which would show a taskboard, sprint backlog, and burndown chart.

Figur 12: Scrum Wise Overblik

Til versionsstyring og revisionskontrol er der brugt *SVN*. Programmerne *SmartSVN* / *TortoiseSVN* er benyttet til at styre den overordnede filhåndtering. Derudover er *AnkhSVN* (et plugin til *Microsoft Visual Studio*) blevet benyttet, hvilket sikrer en nem styring af solution-filer. Dette har givet en flydende versionsstyring uden store problemer.

Til test af koden er *NUnit* blevet benyttet, sammen med frameworket *Rhino Mocks*, der var en hjælp til testskrivning. For at overskueliggøre hvor meget af koden der var testet, er *DotCover* blevet brugt sammen med de ovenstående programmer.

Derudover har hjælpepluginet *ReSharper* ligeledes hjulpet med syntax, opbygning af programmer og rettelser af navne på attributter, variabler osv.

Alt tekstredigering er foregået i *LaTeX*, hvilket har været meget brugbart, da større dokumenter kunne opdeles i forskellige filer, som kunne arbejdes på individuelt. Dette har tilladt, at flere gruppemedlemmer kunne redigere samme dokument uden der opstod konflikter. Desuden sikres, at de nyeste billeder og lignende altid er opdateret, da *LaTeX* linker til disse i stedet for at indsætte dem direkte ligesom i fx word.

¹²<http://www.scrumwise.com/> - for nærmere indblik i Scrum Wise

Brugen af *LaTeX* har dog givet en del ekstra arbejde, da det var et nyt program for de fleste gruppemedlemmer. I sidste ende er den tabte tid dog blevet genvundet, da samlingen har fungeret smertefrit, og den grundlæggende opsætning ikke er blevet ødelagt i takt med, at der er tilføjet mere tekst.

Til brug til at lave diagrammer er programmet *Visual Paradigm For UML*, samt *Star UML* blevet benyttet. Disse programmer er fundet rigtig gode og nemme at arbejde med og dermed blevet de primære valg som diagramværktøjer frem for for eksempel *Microsoft Visio*.

Microsoft Visio er dog blevet benyttet, da det har været glimrende til relationelle skemaer. Derudover er programmet *DIA* blevet brugt til udformning af ER-diagrammer.

10 Resultater

Nedenfor er de mest væsentlige resultater listet, hvorefter de kort beskrives. Der refereres til accepttestspecifikationen, hvis det ønskes at få et større overblik over de udførte tests.

Overordnede resultater

- Sortering af klodser efter matrialetype via Scrobot-robotarm.
- Implementering af IronPython som via scripting kan kontrollere robotarmen.
- Interfacing af den udleverede strain gauge således, at signalet på 0-100mV blev forstærket til 0-5V.
- Opbygning af en relationel database til persistent lagring af data samt alle informationer fra processen.
- Opbygning af en grafisk brugergrænseflade der kan styre systemet.

Sortering af klodser

Robotarmen kan samle en klods op fra transportbåndet, måle alle sider, placerer klodsen på vægten, og veje den, udregne densiteten, og slutteligt placerer den i tilsvarende kasse.

IDE

Det kan lade sig gøre, at skrive simple programmer til robotten ved brug af IronPython, og benytte et IntelliSense-lignende system til at autoudføre kode.

Vægtcellen

Den udleverede strain gauge gav et for svagt signal, der derfor skulle forstærkes. Der blev designet et simpelt print, som kan forstærke signalet til værdier, som interfacer med me-gal16 microcontrolleren. Der blev også skrevet et program, som A/D konverterer signalet og sender det til PCen via en RS232-forbindelse.

Grafisk brugergrænseflade

Den grafiske brugergrænseflade tager udgangspunkt i et login vindue, hvor man kan logge ind som operatør eller programmør. Derefter bliver man præsenteret for en hovedmenu, hvori der kan åbnes forskellige vinduer med forskellige funktionaliteter. Det omhandler blandt andet, at man kan se log beskeder på databasen, se en liste over de forskellig klod-sers placering og starte/stoppe samt pause systemet i en given frekvens, fx mens den er i gang med at sortere en klods.

Database

Den relationelle database er skabt med Microsoft SQL server og består af fem tabeller til at holde data om systemet. I systemet gemmes der en del persistente data, herunder brugere, klodser, positioner i en boks, systemevents og densiteter. Data bliver både gemt og hentet flere steder i systemet, og til det er der blevet implementeret et beskedkø-system som håndterer database tilgangen. Fra systemet side er det muligt at opdatere, hente fra, skrive til, og slette fra databasen. Desuden håndterer databasen selv vedligeholdelse af databasen med diverse triggers og identity kolonner.

11 Diskussion af opnåede resultater

Som det kan ses i det ovenstående resultat afsnit, er de overordnede krav til produktet blevet implementeret. Det er lykkedes at udvikle et softwareprodukt, som kan sortere et antal klodser alt efter deres materialetype, hvilket selvfølgelig er den vigtigste funktion i

produktet. Undervejs i sorteringen persisteres de indsamlede data på en database, så de til enhver tid kan tilgås gennem den grafiske brugergrænseflade. Alt kommunikationen til robotten styres også gennem brugergrænsefladen, så det er muligt for brugeren at starte og stoppe en sortering, og det er også muligt at tilgå en række ekstra funktionaliteter. De ekstra funktionaliteter vil blive diskuteret senere i afsnittet.

Selve sorteringsmekanismen virker stabilt, og der er også implementeret fejlhåndtering således, at klodser, der ikke kan genkendes, frasorteres. Mekanismen sorterer klodser med en rimelig stor præcision, og denne præcision er også inden for den med kunden aftalte præcision. Under kørsel af funktionen, er det altid muligt at udregne klodsens densitet, og dette er ved hjælp af den implementerede vægt, hvis præcision også stemmer overens med den fra starten fastlagte præcision. Der er dog opstået et problem, idet at boksen til sortering af materialetyper indeholder fem rum, men da det sidste rum har været uden for robotarmens rækkevidde, har det kun været muligt at sortere fire ud af de fem udleverede materialetyper. Men da dette er en fejl i det udleverede hardware, har det ikke været muligt at udbrede denne fejl.

Det er selvfølgelig en subjektiv vurdering, om en grænseflade er pæn og overskuelig, men det er tilstræbt at lave den så brugervenlig som muligt. Knapperne har en passende størrelse og teksten på dem er kort og præcis. Desuden er der anvendt et tema, som går igen på alle vinduer, hvilket skal give et godt helhedsindtryk. Selve implementeringen af GUI'en bygger som nævnt på MVVM, og dette gør det muligt at udskifte GUI'en, uden at skulle skifte hele modellaget, og dette må siges at være en stor fordel.

Den relationelle database indeholder alle de persisterede data, undtagen de brugerdefinerede programmer, da disse er gemt lokalt. Måden at database tilgangen er implementeret på, bevirker at en afbrydelse af forbindelsen til internetadgangen, ikke nødvendigvis stopper programmet. Det er kun i de tilfælde, hvor data på databasen, er nødvendig for programmets videre kørsel, at brugeren vil blive afbrudt i programkørslen, og her får han mulighed for at vente på at forbindelsen genetableres. Dette er en stor fordel, da det automatiserer systemet i forhold til databasetilgangen.

IDE'en giver programmøren mulighed for at lave et brugerdefineret program, og det indbyggede IntelliSense, gør det nemt for programmøren at få et overblik over tilgængelige funktioner og anvendelsen af disse. Dog skal det nævnes, at IntelliSense-funktionen bliver aktiveret, så snart der sættes et punktum, så det fremkommer også, selvom der skrives noget kode, der ikke er gyldigt. Selvom det er muligt at gøre dette, vil der komme fejlmeddelser så snart der compiles, hvis der ikke er rigtig syntax.

De brugerdefinerede programmer virker naturligvis på den rigtige robot, men det er også muligt at lave en simulering af dem, så de testes før de køres på den fysiske robot. Simuleringen er lavet forholdsvist simpelt, da det er en log der udskrives, men det vigtigste er, at de relevante informationer udskrives på brugergrænsefladen, så forløbet kan følges. Det havde været en mulighed at lave en form for grafik til denne simulering, men da den nuværende simulering overholder kravene fra kunden, blev dette nedprioriteret i forhold til andre funktionaliteter.

Undervejs i udarbejdelsen af programmet er der blevet tilføjet ekstra funktionaliteter. F.eks. er systemet blevet implementeret således, at der er forskellige brugerprofiler i systemet, som har forskellige rettigheder og rådighed over forskellige funktioner. Dette skal være med til at optimere systemets sikkerhed, da det kun er personer med en bestemt autorisation, der kan anvende funktioner, som kan være skadelige for systemets sikkerhed. Disse funktionaliteter er f.eks. at redigere i eksisterende materialetyper og lave brugerdefinerede programmer. Det er også en ekstra funktionalitet, at der kan redigeres systemets eksisterende materialetyper samt tilføjes nye materialetyper gennem brugergrænsefladen. Dog er denne funktionalitet ikke helt optimalt, grundet de fastlåste koordinater i algoritmen for sortering af klodser.¹³ Desuden er der implementeret en funktionalitet, der gør det muligt, at se en liste over alle klodser, der er blevet sorteret. Det er også muligt at fjerne disse klodser fra systemet, men klodsen vil naturligvis stadig ligge i boksen med de sorterede materialetyper. Ligeledes er der implementeret en funktionalitet, der gør det muligt at hente gemte logbeskeder for en given dato.

Det er også gjort muligt, at ændre på systemets opstilling gennem brugergrænsefladen.

¹³Dette er nærmere beskrevet i systemarkitekturen afsnit 5.3.5 Use Case 3 : Rediger materialetype realisering.

Dvs. at det er muligt at ændre transportbåndets positionering, dog kun på en enkelt akse. Dette gøres ved blot at indtaste tre værdier, der angiver nogle mål mellem transportbåndet og selve robotten, og programmøren bliver igennem brugergrænsefladen guidet igennem disse indtastninger med illustrative billeder.

12 Opnåede erfaringer

Som tidligere nævnt er Scrum Wise blevet brugt til at håndtere arbejdsfordelingen, primært gennem sprintbackloggen. Det hændte dog i få tilfælde, at der blev arbejdet på noget, som ikke var beskrevet i sprintbackloggen. Værktøjerne kunne således have været brugt mere konsekvent, hvilket ville resultere i, at der havde været et lidt bedre overblik over arbejdsfordelingen. Selvom Scrum Wise kunne have været benyttet i større omfang, skal det nævnes, at det også har bidraget med mange gode elementer. Det gav især et godt overblik over, om deadlineen for det aktuelle sprint kunne overholdes.

At dele ansvaret ud er som regel en positiv ting, da mere arbejde bliver udført på kortere tid, men under projektet hændte det, at en enkelt person sad med alt ansvar og alt viden om en vigtig komponent i projektet. Det havde været en fordel altid at have to personer afsat til et vigtigt arbejdsområde. I få tilfælde opstod der situationer, hvor udviklingen af en funktionalitet måtte sættes på standby, da personen med ansvaret for dette område var udeblevet grundet sygdom. Generelt er det også erfaret, at det er meget vigtigt at arbejdsopgaverne bliver uddelt på hensigtsmæssig måde, og at der hele tiden kommunikeres mellem gruppemedlemmerne omkring udviklingen af disse. Hertil blev det erfaret at det daglige Morning Scrum Meeting var rigtig godt til dette.

En anden erfaring der er gjort under udviklingen af projektet er, at der skal afsættes meget tid til håndtering af fejl, især når store dele af systemet sættes sammen. Det gav f.eks. anledning til store problemer, da IDE'en skulle omskrives til at passe til MVVM. Generelt var der også ofte problemer, da de forskellige funktionaliteter skulle testes på robotten, og der blev brugt rigtig meget tid på at få den initialiseret, hver gang noget nyt funktionalitet skulle testes.

Gennem projektudviklingen er der naturligvis brugt meget viden fra de forskellige fag. Eftersom projektet er skredet frem, er der tilegnet sig mere viden gennem undervisningen i de forskellige fag, og denne viden er så blevet benyttet i projektet. Den iterative udviklingsprocess bidrager generelt godt til dette. F.eks. blev MVVM først introduceret i undervisningen efter at udkastet til grænsefladen, men alligevel blev der brugt tid på at få dette implementeret, da det sikrer en god opdeling mellem de forskellige lag. Det samme gjorde sig gældende mht. databasen, da der blev afsat tid til at lave en normalisering af denne meget sent i forløbet, da normalisering ligeledes først blev introduceret meget sent i projektudviklingen.

13 Projektets fortræffeligheder

Nedenfor er en række funktionaliteter og arkitekturer i projektet, som gruppen er specielt stolt af, listet og beskrevet.

IDE - Specielt IntelliSense

Der var lagt op til at IDE-delen af projektet ville blive meget tidskrævende. Dog blev en løsning fundet ved hjælp af IronPython scriptsproget, som både sparede tid, samt resulterede i en god, funktionel løsning. Den tid der blev sparet blev istedet brugt til at gøre IDE'en bedre, og der blev blandt andet implementeret IntelliSense, der kan foreslå valgmuligheder løbende mens koden skrives. Selvom IDEen ikke altid fungerer helt fejlfrit, gør den det ekstremt simpelt at skrive programmer.

GUI - MVVM

I første omgang blev brugergrænsefladen udviklet uden nogen egentlig arkitektur, andet end forms and controls. Da gruppen blev introduceret til MVVM, stod det hurtigt klart at denne arkitektur måtte implementeres. Det var ikke nogen nem opgave, da den eksisterende struktur var rodet, men ved hårdt arbejde blev der implementeret MVVM samt et message system til at håndtere beskeder mellem ViewModel og Views.

Databasetilgang

Det kan vække mange problemer at tilgå en database, og derfor blev der lagt meget arbejde i at udvikle en fornuftig implementering af denne tilgang. Derfor blev der lavet et message system baseret på observer-pattern, der sikre at data gemmes trådsikkert, og uafhængigt af om der er forbindelse til databasen eller ej.

Interfacing til vægtcellen

Interfacing arbejdet til vægtcellen var relativt hurtigt overstået uden problemer, og igennem hele processen fungerede dette uden modifikationer. Således har en har udviklingen af en virksom vægt ikke kostet meget. Derfor betragtes den som en fortræffelighed, selvom den er simpelt udviklet. Der skal dog knyttes en kommentar til at en brændt IC gjorde vægten upræcis under accepttesten.

14 Forslag til forbedringer af projektet eller produktet

Under udarbejdelsen af et softwareprojekt er det næsten altid plads til forbedringer, både mht. til arbejdsprocessen og selve produktet. Det gælder også for dette projekt. Nedenfor er en række elementer i projektet, som kunne have været optimeret, listet og beskrevet.

Klodser i kassen

I programmet er det hardkodet hvor klodserne skal placeres. Dette betyder, at det vil være svært at udvide systemet med flere rum, eller for den sags skyld flytte på hvor klodskassen er. Hvis algoritmen ikke havde være hardcodet, havde det også været muligt at angive en boksplacering, til de tilføjede materialetyper. Denne funktionalitet blev ikke prioriteret særligt højt, da kassen er fastmonteret.

Simulering

Den implementerede simulator er meget simpel. Der udskrives beskeder til en log, og på den måde kan brugeren følge med i programmets forløb. Dette kunne evt. have været forbedret vha. en grafisk simulator. Men da denne funktion ikke var en del af kravspecifikation, blev det vurderet, at den ville være for tidskrævende i forhold til betydningen af

denne.

Vinduer i GUIen

Brugergrænsefladen består af en række vinduer. Disse vinduer er kommet til et af gangen og eksisterer derfor uafhængigt af hinanden. Dog er der elementer som fx en statusbar og en menulinje, der er fælles for alle vinduer. Der kunne være sparet en del arbejde og kode ved at lægge alle views i samme vindue og indsætte indholdet som en UserControl. Dette ville betyde at redundant kode, som er fælles for alle vinduer, kunne placeres i hovedvinduet. Desværre gjorde den iterative udvikling, at dette ikke kunne lade sig gøre.

Opdatering af kravspecifikation

Et andet forslag til forbedring kunne være at få tilføjet alt funktionaliteten i det endelige program til kravspecifikationen. F.eks. er funktionaliteten med håndtering af klodser i systemet ikke en del af kravspecifikationen. I det endelige program er det også muligt at flytte på transportbåndet og andre dele af systemet, og en Use Case kunne også have været udformet på baggrund af denne funktionalitet. Det skal dog nævnes, at disse Use Case ville være forholdsvis små, så dette er en af grundene til, at de ikke er blevet tilføjet til kravspecifikationen.

15 Konklusion

Silver Bullet Sort programmet er resultatet af en velfungerende arbejdsproces, og dette kan også ses på det endelige produkt. De agile udviklingsprocesser og rammer har været en stor hjælp til udviklingen af softwaren, og de har været med til at give et overblik over arbejdsgangen. At der i starten af projektet blev brugt tid på at fastlægge kravene og undersøge implementeringsmulighederne resulterede i, at der blev dannet et godt fundament for udviklingen af det samlede system. Hertil kan der knyttes denne kommentar at der fortløbende er blevet tilføjet funktionalitet til selve produktet, som ikke er blevet defineret som krav.¹⁴ Systemet er opbygget efter en lagdeling, således at koden bliver overskuelig og let kan videreudvikles, samtidig med at det er muligt at udskifte de forskellige lag.

¹⁴Se afsnit 14: Forslag til forbedringer af projektet eller produktet.

Denne lagdeling har også resulteret i, at det var let at arbejde forholdsvis uafhængigt af hinanden, hvilket var en stor fordel.

Det kan konkluderes, at det endelige produkt lever op til de opstillede krav, hvilket kan ses i afsnittene *Resultater* og *Diskussion af opnåede resultater*. Programmet er brugervenligt, og det er nemt at få robotten til at sortere en klods efter materialetype vha. det indbyggede standardprogram. Desuden er det overskueligt at følge med i selve sorteringsprocessen, og informationer om denne process kan til enhver tid tilgås, da de som nævnt er persisteret på en database. Det kan yderligere konkluderes, at IDE'en opfylder sit mål idet, at det er muligt at lave brugerdefinerede programmer. Disse programmer kan bruges på den fysiske robot, og hvis denne ikke er til rådighed, kan de simuleres, således at informationer om programmets kørsel bliver vist. Heraf kan det konkluderes, at simuleringen også virker efter hensigten.

Hvorom alting er, kan det konkluderes at de mange møder med virksomheden, har medført at det endelig produkt stemmer overens med virksomhedens ønsker og forventninger.

16 Referencer

Der er ikke nogen nævneværdige referencer i dette dokument, da de hjemmesider der henvises til, er vedhæftet som fodnoter. Tilgængæld refereres der til en række dokumenter, som er udviklet af projektgruppen. Disse er listet nedenfor:

- SBS_Kravspecifikation.pdf
- SBS_Accepttestspecifikation.pdf
- SBS_Systemarkitektur.pdf