

CDIO opgave forår 2014 - final

02324 Videregående programmering

Gruppe 51 - Area 51.

Alternativt projekt - dataopsamling fra tests af piezoelektriske krystaller.



Morten Hesselbjerg, s017704



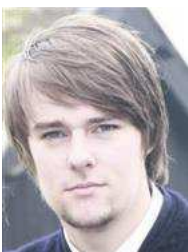
Christian Budtz, s134000



Martin Nielsen, 123064



Eirik Oterholm Nielsen, 123006



Rúni Egholm Vørmadal, s134004

Tidsregistrering

Dato	Navn	Design	Impl.	Test	Dok.	Andet	I alt	Bemærk
04/06/2014	Martin		4				4	
04/06/2014	Christian	2	4				6	
04/06/2014	Eirik		4				4	
04/06/2014	Morten		4				4	
06/06/2014	Eirik	2	4				6	
06/06/2014	Martin		8				8	
06/06/2014	Rúni	1	9				10	
06/06/2014	Christian	1	4	3			8	
06/06/2014	Morten		8				8	
07/06/2014	Morten		2				2	
07/06/2014	Rúni		6	2			8	
08/06/2014	Morten		2				2	
08/06/2014	Christian		0,5	0,5	1		2	
08/06/2014	Rúni		3	1	2		6	
09/06/2014	Rúni		4				4	
10/06/2014	Martin		8				8	
10/06/2014	Rúni		6	2			8	
10/06/2014	Eirik		7				7	
10/06/2014	Christian	1	3	2			6	
10/06/2014	Morten		1				1	Syg
11/06/2014	Martin		7				7	
11/06/2014	Eirik		4	2			6	
11/06/2014	Rúni		8	2			10	
11/06/2014	Morten		8				8	
11/06/2014	Christian		7				7	
12/06/2014	Eirik		5				5	
12/06/2014	Martin		6,5				6,5	
12/06/2014	Rúni		3			3	6	
12/06/2014	Morten		6,5				6,5	
12/06/2014	Christian		4	2			6	
13/06/2014	Morten		6				6	
13/06/2014	Martin				1	4	5	Læst kode og om Javascript/HTML
13/06/2014	Eirik		7				7	
13/06/2014	Christian		2	1	1		4	
13/06/2014	Rúni		3			1	4	
15/06/2014	Morten		2				2	
15/06/2014	Christian				2		2	

16/06/2014	Christian				2		2	Syg :(
16/06/2014	Rúni		4	2			6	
16/06/2014	Eirik		8				8	
16/06/2014	Martin		2	2		2	6	
16/06/2014	Morten		6				6	
17/06/2014	Martin				5	2	7	Fixed eclipse bugs
17/06/2014	Eirik				4		4	
17/06/2014	Rúni		5	1			6	
17/06/2014	Christian				2		2	Syg :(
17/06/2014	Morten		6				6	
18/06/2014	Christian		3				3	Forsøg med Cloud - stadig syg
18/06/2014	Rúni		5	5			10	kundebesøg
18/06/2014	Eirik		1	5			6	
18/06/2014	Martin				5		5	
18/06/2014	Morten		3	3			6	Kundebesøg
19/06/2014	Eirik				16		16	
19/06/2014	Martin				9	9	18	Lavet poster
19/06/2014	Morten		4	2	13		19	
19/06/2014	Christian				12		12	
19/06/2014	Rúni		3	1	14		18	
20/06/2014	Christian				5		5	
	I alt							
	Christian	4	25,5	7,5	24	0	61	
	Eirik	2	40	7	20	0	67	
	Martin	0	35,5	2	20	17	74,5	
	Morten	0	58,5	5	13	0	76,5	
	Rúni	1	59	16	16	4	96	
							375	

Indholdsfortegnelse

Tidsregistrering	2
Initialisering og opstart af projektet.....	5
Import/installation.....	5
Brugerguide.....	5
Anvendelse af måleprogram	5
Anvendelse af Web applikation.....	7
Indledning	11
Projektforløb.....	12
Vores prioriterede backlog:.....	12
Kravspecifikation	13
Use Cases.....	14
Hent data.....	16
Administrer brugere	16
Design.....	16
Designovervejelser- og afgrænsning	17
Design processen i projektet.....	18
Database Model	18
Design patterns	19
Implementering	24
.(default package)	25
.view	26
LoginGui	26
BatchMeasureGui.....	26
NewBatchGui	27
CalibrationGui.....	28
.control	29
LoginController.....	29
BatchMeasureController	29
NewBatchController.....	29
CalibrationController	30
DirectoryListener	30
DasyFileReader	30
DataBaseController	30
CConnector	30
JavaMeasure.....	31
Data Transfer Objects.....	31
Data Access Objects.....	31
C# Komponent	32
C# komponent - Asynkron socket listener (SocketASync-klasse).....	33
C# komponent - USBConnection klasse.....	33
Web-delen	33
Test	35
Konklusion.....	35
Muligt fremtidigt projekt	36
Litteraturliste.....	37
Bilag	37
Bilag 1 - Java/C# protokol	37
Bilag 2 Bidragydere til projekt	38

Initialisering og opstart af projektet

Import/installation

Projektet importeres som et eclipse projekt. For at køre java programmet, startes MeasureMainTestMode. Dette starter programmet i en testmode hvor USB-DAQ'en ikke skal tilsluttes, men bruger en slags simulator, som returner nogle tilfældige værdier. Web applikationen kan køres efter man har sat op sin egen tomcat server. Det anbefales at man kører web applikationen på sin browser i stedet for i eclipse, da det lader til at den ikke understøtter alle javascript funktioner.

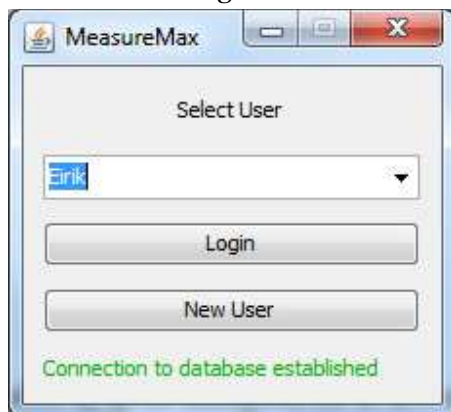
Brugerguide

Selve programmet er delt i to - et Java måleprogram og en Web-applikation.

Anvendelse af måleprogram


For at lave målinger skal PROGRAMMET startes. Hvis du er registreret bruger skal du indtaste dit brugernavn og trykke "log in" ellers skal du indtaste ønskede brugernavn og trykke "new user", hvorefter du bliver automatisk logget ind. Oprettelse af ny bruger skal gennem en godkendelse før du kan logge ind, og hvis du ikke bliver godkendt får du en fejlbesked på skærmen.

Herunder ses i figur 1 det første billede en bruger bliver mødt med - loginbilledet.



Figur 1. Login

Efter brugeren er logget ind kan hovedvinduet hvor målinger bliver lavet ses (fig.2). Til venstre er to vinduer, øvre vindue viser de målinger der bliver taget, og ellers de målinger der tilhører batchet. Får hvert element der tilføjes, kommer et afkrydsningsfelt, hvor elementer kan markeres som godtaget eller ikke godtaget. Vinduet under udskrifter hvad der sker i programmet, her bliver fejlmeddelelser også skrevet, hvis for eksempel USB stikket til måleudstyr ikke er sat til. Til højre i vinduet vises specifikationerne som tilhører batchet.


 Logged in as: Runi

verified

Element

Stroke Value

Leak Current

New Batch

Load Batch

Edit Settings

Leak Current Measurement

Stroke Measurement

Approve batch

Delete last stroke

Delete last leak

Log out

Customer: -

Drawing number: -

Item description: -

Specification: -

Item code: -

Visual inspection: -

Internal order number: -

Specifications	norm value	tolerance	Insp lvl
outer diameter:	-	±	-
inner diameter:	-	±	-
thickness:	-	±	-
elekt. marg. OD:	-		-
elekt. marg. ID:	-		-
number of active layers:	-	±	-
layer thickness:	-	±	-
capacitans:	-	±	-
stroke	-	±	-
blocking force:	-	±	-
measure voltage:	-		-
dynamic test:	-		-
leak current, max	-		-
DC resistance:	-	±	-
special visual inspection:	-		-
spectrum:	-		-

Event log:

I midten er så valgmulighederne for brugeren. For at komme i gang med at lave målinger er det nødvendigt at have et batch at tilføje målingerne til. Dette gøres ved enten at hente et batch fra databasen ved klik på “load Batch” og derefter skrive hele batchnavnet, eller ved at oprette et nyt batch ved klik på “new Batch”.

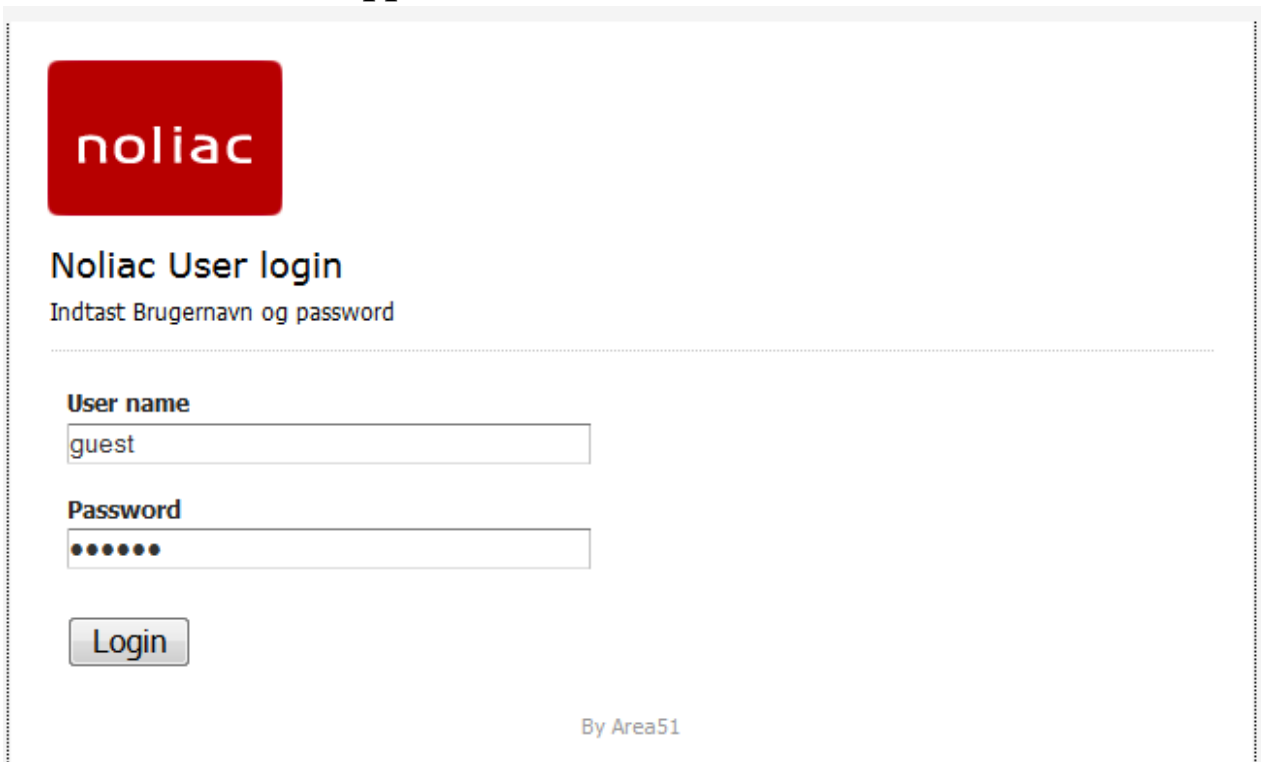
Vinduet hvor et nyt batch kan oprettes har også nogle ekstra knapper: “Delete Settings”, “load Batch Settings” og “Save Batch Settings”. Disse kan bruges til at gemme bestemte profiler, så indstillingerne kan genbruges på et nyt batch, uden at brugeren behøver at indtaste alle værdierne igen. Disse profiler kan så selvfølgelig også slettes igen, hvis det ønskes.

hvor DasyLab dumper sine dasyLab filer. Hvis man kun vil teste denne funktion, kan man kopiere og indsætte dasyLab filer i en den mappe man har valgt. Hver fil svarer til én måling, så husk at tage stroke målinger imellem hver fil man tilføjer. Ved tryk på “Stroke measurement” hentes en måling fra mikrometer skruen.

Når man er færdig med sine målinger kan man starte på nyt, hente et andet batch eller bare lukke programmet. Alle målinger, batch og profiler bliver gemt løbende, således at når programmet lukkes, så er alt allerede gemt. Hvis man ikke har internet forbindelse når programmet kan man ikke logge ind. Hvis internet forbindelsen forsvinder imens man tager målinger bliver fejlbeskeder sendt til vinduet nede i venstre hjørne, så man bliver opmærksom på at målinger ikke bliver gemt.

Batch profilen på det aktive batch kan ændres ved at trykke på “edit batch profile”, her kommer samme vindue frem, som når man oprettede nyt batch, men her er ikke muligt at ændre batchnavnet, og mængden af knapper er også blevet begrænset. Enten kan du gemme de nye indstillinger eller gå tilbage.

Anvendelse af Web applikation



The screenshot shows a web application login interface. At the top left is a red square logo with the word "noliac" in white lowercase letters. Below the logo, the text "Noliac User login" is centered. Underneath that, the instruction "Indtast Brugernavn og password" is centered. A horizontal dashed line separates the header from the form. The form contains two input fields: "User name" with the text "guest" and "Password" with masked characters represented by dots. Below the password field is a "Login" button. At the bottom right of the form area, the text "By Area51" is displayed.

Figur 3. Login

Når man starter programmet bliver man introduceret til loginsiden, til demobrug er det muligt at logge på med den forudfyldte bruger guest med passwordet 123456. Hvis du prøver at logge på med en ikke eksisterende bruger, "alæsdkaælsdkf", på en inaktiv en "Holger" eller med det forkerte kodeord, så vil du blive sendt tilbage til denne side med en fejlmeddelelse.



Noliac Batch-udtræk

Indtast Batch ID eller vælg Batch ID fra rullemenu

Indtast Batch navn <input type="text" value="ch"/>	Batch navn	Oprettet af	Oprettet dato	Godkendt af	Godkendt dato
Dato: <input checked="" type="radio"/> Oprettet <input type="radio"/> Godkendt	charles	Martin2	2014-06-18 15:24		
Start dato 18 / 6 / 2014 DD MM YYYY	charles22	Martin2	2014-06-18 15:25		
Slut dato 18 / 6 / 2014 DD MM YYYY	EKSAMEN BATCH	Eirik	2014-06-18 15:46		
<input type="button" value="I dag"/> <input type="button" value="Denne uge"/> <input type="button" value="I går"/>					
<input type="button" value="Sidste uge"/>					
<input type="button" value="Søg"/> <input type="button" value="logout"/> <input type="button" value="Edit User"/>					

By Area51

Figur 4. Menu

Når du er logget ind kan du søge på batches efter navne bliver det søgt og en tabel bliver lavet løbende med batches der matcher. Man kan også skrive start og/eller slutdato, men så må man trykke søg for at få genereret en liste. Det er også muligt at kombinere de to. Man vælger en batch ved at klikke på den, og så bliver man sendt videre til følgende side.

Noliac Batch report

Final Inspection and Test Report Certificate of Conformance				Batch id:	EKSAMEN BATCH			
Customer	Area-51							
Item description:	Cirkler			Drawing number:	2			
Item code:	6213-161			Specification:	VIGTIGT			
Internal order number:	651			Visual Inspection:	YES			
Parameter	Specification			Inspection level	Measured			
	Norm value	Min	Max		Average	Min	Max	Approved
Length (mm):	10	9.9	10.1					
Width (mm):								
Thickness (mm):								
Capacitans (nF):								
Stroke (um):								
Leakage current:	10	9.9	10.1					
Visual inspection:	10	9.9	10.1					
Comments:								
					approved date			
					number approved			
					approved by			

Ny rapport

gem som Excel

By Area51

Figur 5. Report

Her kan du se den relevante information vedrørende den valgte batch. Der mangler at blive fyldt ud nogle steder, men det er fordi vi følger Noliacs rapport form, men genererer ikke al informationen i vores løsning. Trykker du på gem som Excel, så får du hentet rapporten som en csv fil. Denne csv fil giver ikke 100% mening, men den indeholder den rigtige information, og det skal vores med venner ude på Noliac køre igennem en Excel behandler og lave en flot rapport ud af. Så trykker vi på Ny rapport, og kommer tilbage til menuen. Nu trykker vi på Edit User og får følgende side:

The screenshot shows a web form titled 'noliac' with the subtitle 'Editing guest'. It contains a 'Password' input field with the text '123456'. Below the input field are two buttons: 'Save' and 'Cancel'. At the bottom right, there is a small text attribution 'By Area51'.

Figur 6. UserEdit

Her kan vi se hvad koden er i øjeblikket, og vi kan ændre den til hvad du vil. Trykker vi på save, så gemmes koden og vi bliver sendt tilbage til menu'en, siden med logout knappe, eller trykker vi på cancel og kun sendt til menu'en. Nu trykker vi på logout knappen, kommer tilbage til login siden, og logger os ind som Eirik, og koden 13. Vi trykker på Edit User og bliver sendt til følgende side.

The screenshot shows a web form titled 'noliac' with the subtitle 'Choose user'. It features a 'User name' dropdown menu with 'Eir' selected and 'Eirik' visible in the suggestion list. Below the dropdown are two buttons: 'Choose' and 'Done'. At the bottom right, there is a small text attribution 'By Area51'.

Figur 7, Choose User

Vi vælger lige en bruger at ændre i, for sjovs skyld ændrer vi i den bruger vi er logget på med, Eirik. Læg lige mærke til den sjove autocomplete funktion, hvis du bruger en html 5 venlig browser. (Hvis du trykker choose uden at der er en gyldig bruger indtastet, så bliver du på siden.) Choose sender os videre til følgende side.



Figur 8. EditUser

Her kan vi så ændre Eirik til at være inaktiv, eller fjerne administrator statusen, eller blot give brugeren en ny kode. Save gemmer vores ændring, og sender os til den sidste side, medens cancel sender os til menu'en. Det skal lige bemærkes, at vi bliver sendt tilbage til menuen vis vi fjerner administratorstatusen og gemmer, og vi bliver loget af og sendt til loginsiden.

Nu har vi været igennem hele web-delen og dens funktionaliteter.

Indledning

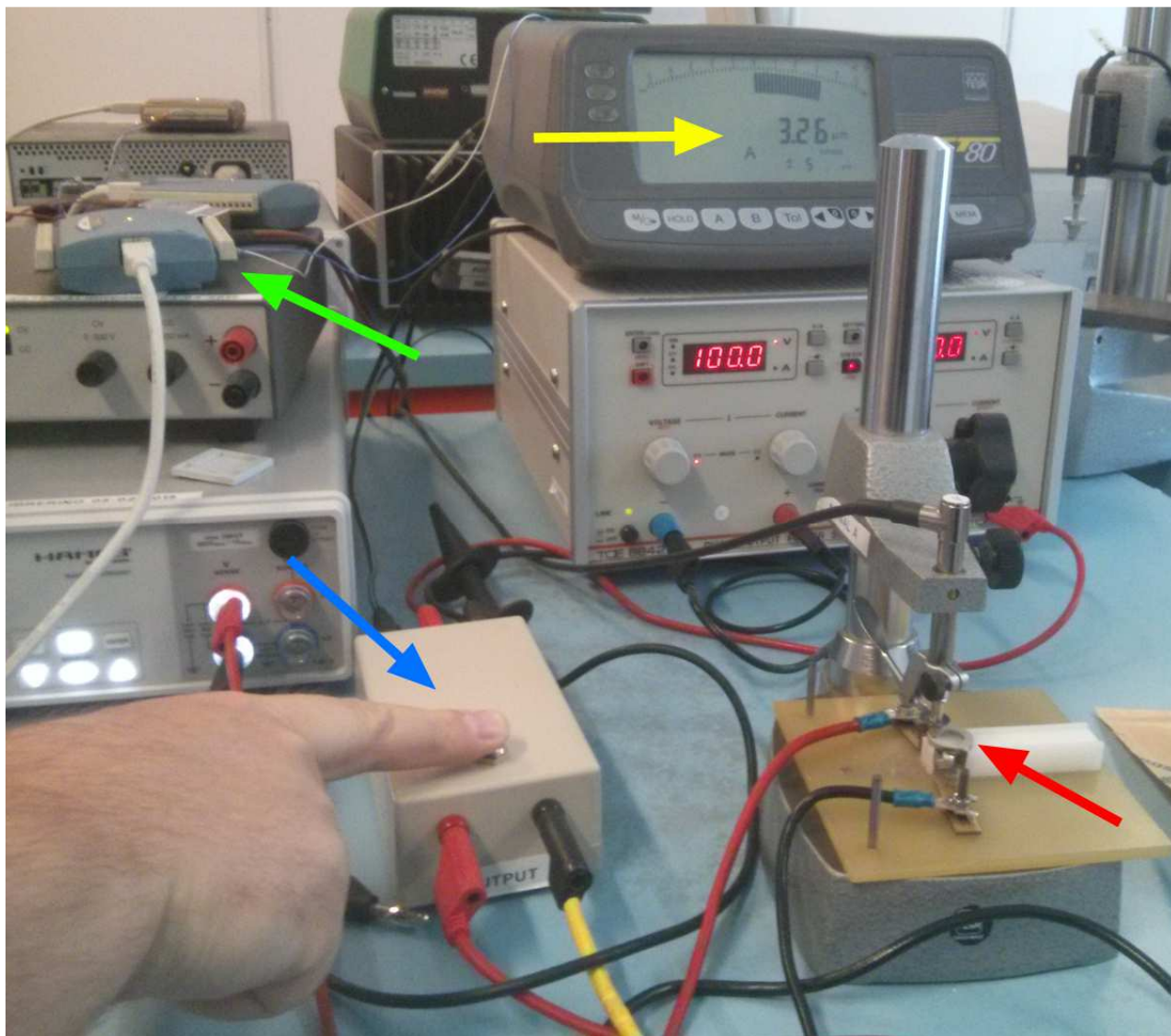
Vi har som projekt valgt at arbejde med en alternativ opgave. Opgaven løste vi for firmaet Noliac, der fremstiller piezokeramiske elementer til en lang række industrier. Noliac udfører kvalitetskontrol på en delmængde af de producerede elementer, for at sikre at de lever op til specifikationerne. Disse tests/målinger inkluderer bl.a.:

- Måling af udbøjning - dvs. den udvidelse der genereres, når elementet udsættes for en spænding
- Måling af lækstrøm

Målingerne udføres en af gangen og de opsamlede data indtastes manuelt i et Excel-ark, for derefter at blive brugt til en batch-rapport og statistik.

Noliac ønsker at få måleprocessen automatiseret i videst mulige omfang, således at kvalitetskontrollen kan foregå hurtigere, men også så den udføres mere ensartet.

Vores projekt blev tænkt som et 'proof of concept', hvor vi søgte at automatisere en del af arbejdsgangen, mhp. evt. fremtidig fuldstændig automatisering af hele kvalitetskontrollen.



Figur 9. Billede fra laboratoriet. Den røde pil angiver elementet der testes. Den gule pil er den målte udbøjning (i μm) på mikrometerskruen. Den grønne pil angiver USB måleopsamlingsenheden, der er forbundet til den analoge udgang på mikrometerskruen. Den blå pil angiver knappen, der påfører spænding, til elementet, så længe den holdes nede.

Projektforløb

Fra starten afgrænsede vi projektets omfang således at vi kunne være sikre på at kunne levere et funktionsdygtigt og robust produkt til deadline. Var tiden til det havde vi også et par ekstra features vi gerne ville implementere. Vi etablerede en prioriteret Backlog med ansvarlige på hver delopgave og ajourførte fremdriften fortløbende for at sikre os at vi nåede essentielle dele af vores projekt.

Vores prioriterede backlog:

1. Fortsat udvikling af dataopsamlingsmodul - Rúni/Martin

- a. Færdigudvikling af DAL - Martin
 - b. Færdigudvikling af save/edit/approve - Martin
 - c. Implementering af 'settings' funktionalitet - Christian
- 2. Implementering af mikrometerskrue - Runi/Morten
 - a. implementering af kalibrering og aflæsning.
- 3. Web del - Rapportgenerering Morten/Christian/Eirik
 - a. Udsøgning af rapporter
 - i. AJAX
 - 1. HTML - Christian
 - 2. JS + XML + JSP - Morten
 - ii. Generering af CSV til Excel-ark - Eirik
- 4. Web del - Brugeradministration
 - a. Implementering af superbrugere -Eirik
 - b. Implementering af administrationsmodul - Eirik
- 5. Web del - servlet
 - a. MVC design - Christian
 - b. Servlet + JSP -Eirik/Christian
 - c. Clean up - refaktorering og kommentarer
- 6. Poster - Martin
- 7. Input validering (regular expressions) - Christian/Rúni
- 8. Bedre exception handling - Alle
- 9. Fortsat testing - Alle
- 10. Evt. Implementering af hot folder funktionalitet - Runi
- 11. Evt. Cloud deployment - Christian
- 12. Evt. konvertering af C# til service - Runi

Vi nåede i mål med alle vigtige delmål. Ved projektslut er det muligt at opsamle data fra en mikrometerskrue hhv. fra en lækstrømsmåling i DasyLab. Data gemmes instantant i en SQL database og kan godkendes af en bruger. Data kan udhentes via en web applikation og det er muligt at generere en rapport, der kan vises som html eller downloades som en csv - fil til import i Noliacs eksisterende regnearksløsning. Desuden kan man administrere brugerne gennem samme web applikation.

Vi valgte at nedprioritere cloudfunktionalitet, da Noliac ikke var interesserede i løsningen, men hellere ville have en lokal løsning og vi havde andre ting med større betydning for Noliac at arbejde med.

Vi brugte således ekstra tid på exceptionhandling, således at data ikke går tabt og brugeren præsenteres for meningsfyldte fejlmeddelelser eks. ved manglende databaseforbindelse. Vi brugte desuden tid på hot folder, da det gør løsningen betydeligt mere anvendelig for slutbrugeren og på integration af vores C# komponent - således at den opstarter og afslutter i baggrunden uden behov for brugerens interaktion. Vi konstruerede desuden to værktøjer til opsætning af programmet - et til konfiguration af servertilslutningen og til kalibrering af mikrometerskruen - også for at gøre programmet mere anvendeligt for slutbrugeren.

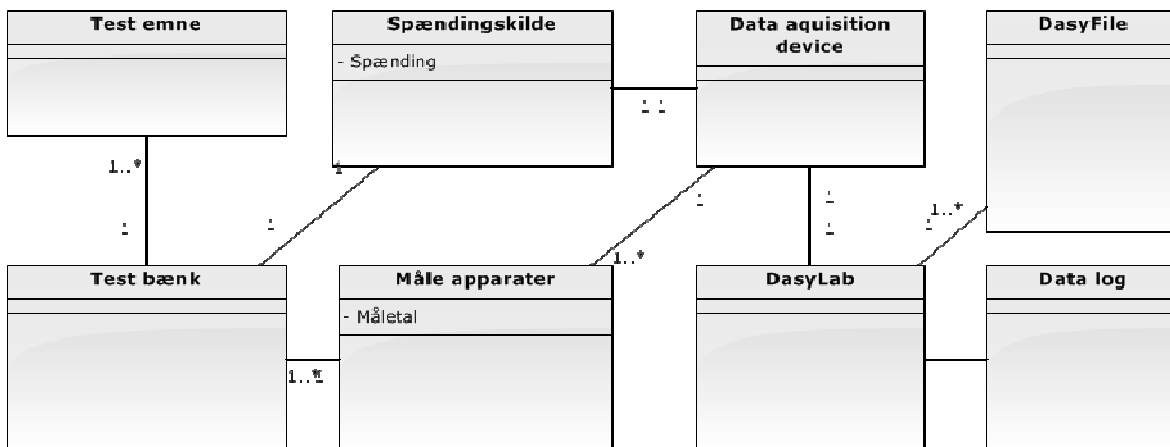
Kravspecifikation

Da vi ikke arbejdede med standardprojektet, har vi analyseret og specificeret projektet løbende

i samarbejde med Noliac.

Den del af kvalitetskontrollen, som vi fokuserede på bestod af en arbejdsgang, hvor et piezokeramisk element blev testet i to forskellige forsøgsopstillinger. Emnet fastgøres i en testbænk og udmåles med en mikrometerskrue - først uden spænding og dernæst med spænding på elementet - forskellen er elementets 'Stroke' (udbøjning).

Herefter påføres elementet en ladning ved at elementet påføres en vekslende strømstyrke - elementet 'masseres' - og ved maksimum afbrydes strømmen hvorefter strømstyrken som elementet afgiver registreres over tid. Efter en specifik tid i forløbet, registreres strømstyrken som 'Leak Current'. Modelleret som en domænemodel, kan det betragtes som i figur 10.



Figur 10. Domænemodel. Måleapparater dækker i vores setting over mikrometerskrue og Amperemeter.

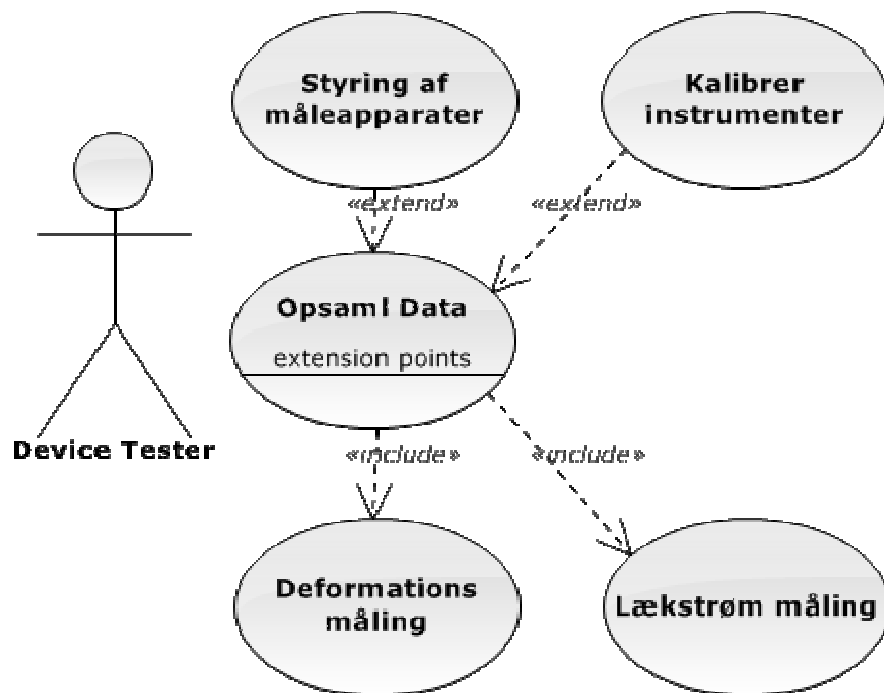
Vi fik frie tøjler fra Noliac, og da vi ikke fik stillet dem de rigtige/og præcise spørgsmål, til at lave vores kravspecifikation, tog nogle implementeringer unødvendig lang tid, og nogle endda unødvendige. Kravspecifikation blev derfor ændret fra et program som skulle lave nogle målinger på en bestemt måde, til at være et proof of concept, hvor vi kunne sætte nogle blackboxes op, der hvor vi ikke kunne lave en fornuftig løsning med hensyn til automatisering, på grund af hardware komponenterne simpelthen ikke eksisterede. Kravspecifikation er derfor præget af, at man fx bruger et eksternt program at lave den komplekse måling (leak), og stroke målingen ikke er optimal, da der både skal trykkes på en knap for at påføre en spænding, og samtidig bede programmet om at hente en måling.

Use Cases

Vi identificerede og forfinede 3 main use cases - Selve udførelsen af målingen, udsøgning af data og administration af brugere.

Preconditions:

Der er fremfundet et batch af piezoelektriske elementer, der skal testes. Testemnet er monteret i testbænken. Måleinstrumenter (mikrometerskrue) tilsluttet korrekte porte på USB- Data Acquisition Device'et. Måleinstrumenter er tændt. Alle nødvendige apparatindstillinger er udført.

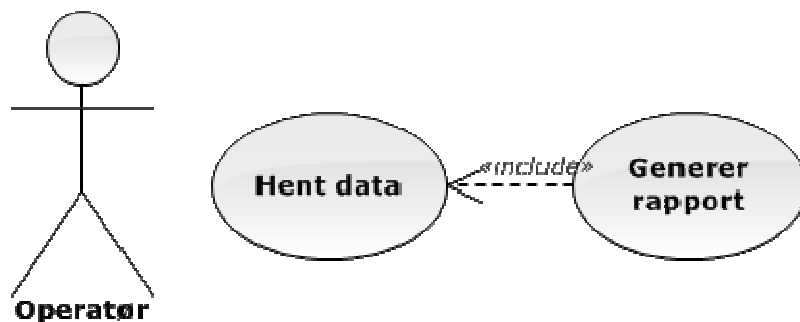


Figur 11

Aktør	System
<ol style="list-style-type: none"> 1. Opstarter program 2. Logger ind. 3. Opretter Batch 4. Opretter en profil med måleindstillinger. 5. Sætter hot folder til DasyLab 6. Starter måling af Stroke. 7. DasyLab bruges til at generere filer til indlæsning 8. evt sidste stroke/leak måling slettes 9. Gentager 6-8 indtil alle emner er målt 10. Afslutter Batch (approve) og logger ud 	<ol style="list-style-type: none"> 1. Modtager nødvendige brugerindstillinger. C# opstartes automatisk C# server opretter kontakt til USB -DAQ. 2. Modtager Brugerdata 3. Opretter Batch i database 4. Gemmer ny måleprofil. 5. Starter ny tråd hvor nye dasyLab filer med leak måling fanges. 6. Sender besked om måleserie til C# server, der igen henter måledata fra mikrometerskrue. Gemmer data i database 7. genererede fil behandles og gemmes i database og vises for bruger 8. Sletter evt. tidligere måling i database. 9. ds. 10. Logger bruger af

Extensions:

- 1a: Apparater kalibreres.
- 6a. Evt. styring af måleapparater



Figur 12.

Hent data

Preconditions: Data er opsamlet og lagret.

Aktør 1. Starter browser og tilgår web-side 2. Vælger batch til rapport	System 1. Modtager login 2. Præsenterer rapport.
--------------------------------------------------------------------------------------	---------------------------------------------------------------

Administrer brugere

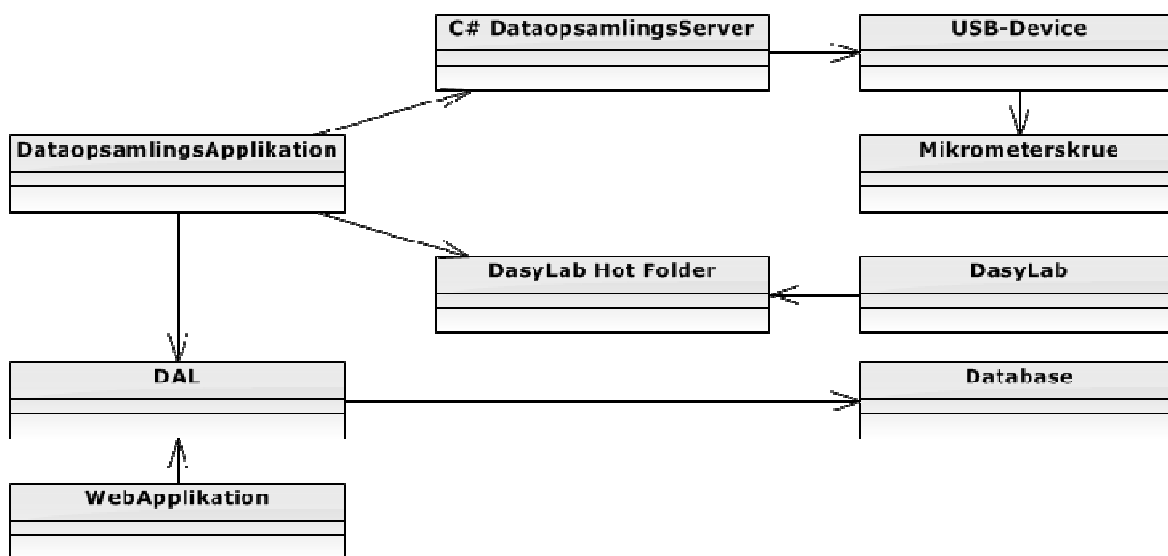
Preconditions: Minimum en administrator i databasen.

Aktør 1. Starter browser og tilgår web-side 2. Vælger bruger 3. Opdaterer bruger	System 1. Modtager login. afgør om brugeren er administrator. præsenterer brugeradministrations side. 2. Præsenterer brugerdata 3. Gemmer ændrede bruger data
------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Design

Vi kunne ud fra kundens ønsker forstå at de ønskede en form for proof-of-concept applikation, som viste om det var muligt at automatisere dataopsamlingsprocessen. Noliac havde ikke yderligere specificeret opgaven, så det var vores opgave at spørge ind til hvordan opgaven kunne tænkes løst.

Ud fra vores forståelse af kravene kunne vi nedbryde projektet i delkomponenter, som ses på figur 13:



Figur 13. Overblik over delkomponenter i applikationsframeworket.

Nødvendigheden af at kunne kommunikere med Noliacs USB-DAQ gjorde, at vi var nødt til at udvikle en del af koden i C#, da der kun fandtes API til C#. Da vores undervisning og viden ellers fortrinsvis omhandler Java, valgte vi at lave en minimal C#-løsning og ellers interface med den med en veldefineret protokol, nemlig TCP/IP.

Da vi ydermere skulle kunne hente data fra DasyLab - et proprietært laboratorieprogram, som ikke umiddelbart har en nem måde at interface med - løste vi det ved at designe en 'hot folder' løsning, hvor vores java-applikation holder øje med nye filer og parser disse for relevante data - således at det forekommer så problemfrit og simpelt som muligt for slutbrugeren.

En SQL-database har de egenskaber som vi har behov for til opbevaring af vores data. Vi designede et Data Access Layer med Data Access Objects til at håndtere CRUD operationer på vores Data Transfer Objects. Af praktiske årsager konstruerede vi en DatabaseController til at håndtere DAO'er og SQL forbindelsen. DataBaseControlleren kunne så bruges i både java og web applikationen.

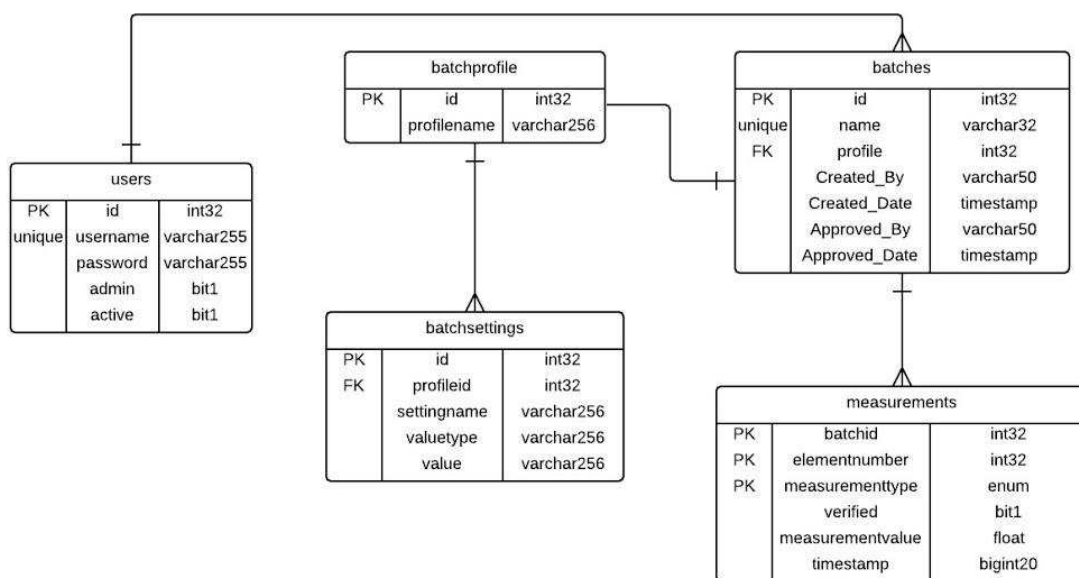
Designovervejelser- og afgrænsning

Undervejs i projektet har vi overvejet forskellige løsningsmodeller og anvendelse af hvilke teknologier vi skulle implementere. Det har blandt andet været overvejet om løsningen skulle laves som en cloud-løsning, men det blev droppet, dels på grund af tidspres, men også fordi det ikke gav en bedre løsning og Noliac havde en streng data politik. Af andre ting kan nævnes at selve stroke-målingen sker ved at brugeren trykker på en fysisk knap (se figur 9), hvorefter målingen aflæses på mikrometerskruen. Det var oprindeligt tænkt at vi også skulle styre strømforsyningen, så det manuelle tryk kunne undværes, men det viste sig at være en kæmpe opgave at lave denne styring, i hvertfald i forhold til vores kompetencer og viden (Her er måske en opgave for en elektro gruppe?). Adskillige andre trin i kvalitetskontrollen kunne indgå i automatiseringen, men vi valgte to umiddelbart tilgængelige problemstillinger, der kunne løses med den teknologi, vi fik undervisning i på semesteret (og et par andre teknologier).

Design processen i projektet

Projektet har båret præg af at det har været en iterativ proces, og nogle ting har vi løbende måtte justere og helt lave om igen. Vi har forsøgt at matche de specifikationer og ønsker, som Noliac har givet os. Vi har dog måttet sande at det har været svært at spørge præcist nok ind til de specifikke krav, hvorfor vi har brugt unødvendig tid på implementeringer, der viste sig at være mere eller mindre ligegyldige. Et eksempel herpå er implementeringen af hvordan man tilføjer målinger synkront - altså en måling af hver slags (stroke/leak), når det viste sig at det ikke betød noget om man først lavede en masse målinger af den ene type for derefter at lave den anden type.

Database diagram



Figur 14. Overblik over databasen.

Database Model

Vi har fem tabeller i databasen, som er vist i figur 14. En bruger har et id som primær nøgle og et brugernavn der er unikt.

En bruger kan lave flere batches, og brugerens navn bliver gemt i batchen, så der er sporbarhed i systemet. Det havde været mere attraktivt med en foreign key til user.id - så man altid er garanteret sporbarhed, men kravet om sporbarhed var ikke åbenlyst fra starten og refaktoreringen havde lavere prioritet end at sikre et pålideligt og anvendeligt program. Sporbarheden er dog sikret ved at alle brugernavne er unikke og det er programmatisk ikke muligt at føre et ikke-registreret brugernavn på et batch.

De measurements man udfører får et tilknyttet batchid, der hører til batchet som foreign key, og kombineret med measurementet's elementnumber og measurementtype danner de en primær nøgle - således at det kun er muligt at udføre én af hver måling på hvert element i en batch.

Før et batch bliver oprettet skal der være oprettet en batchprofil, da profilens id er lagt i

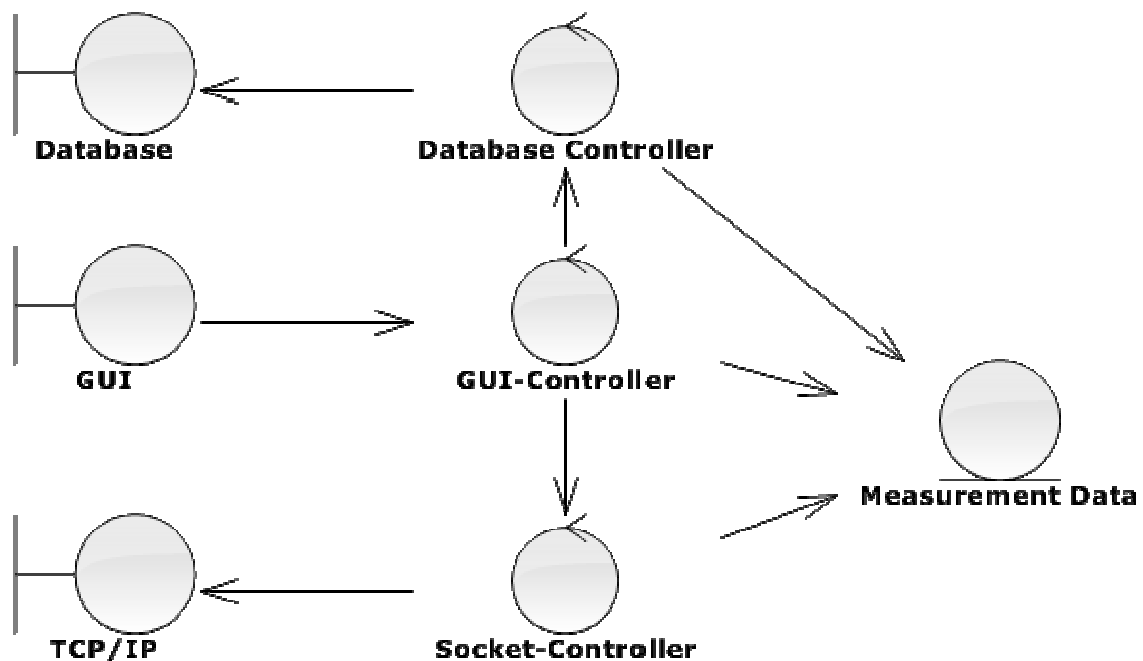
batchet som foreign key. På den måde sikrer vi at alle batches har en tilknyttet profil.

Der bliver lavet 55 batchsettings for hver batchprofile, og de har profilens id som foreign key.

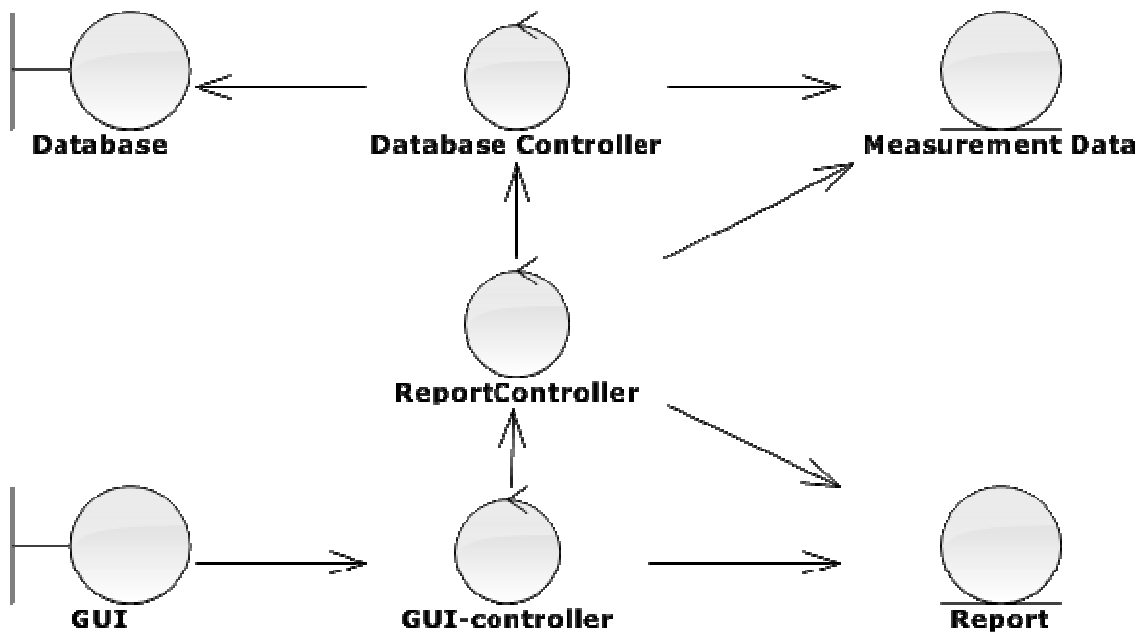
Med de 55 settings har vi en samlet gruppe af data, der beskriver batchet og målingerne kan begynde. Ønsker man nu et udtræk af data for en måleserie, kan man udhente et batch, med tilhørende settings og measurements.

Design patterns

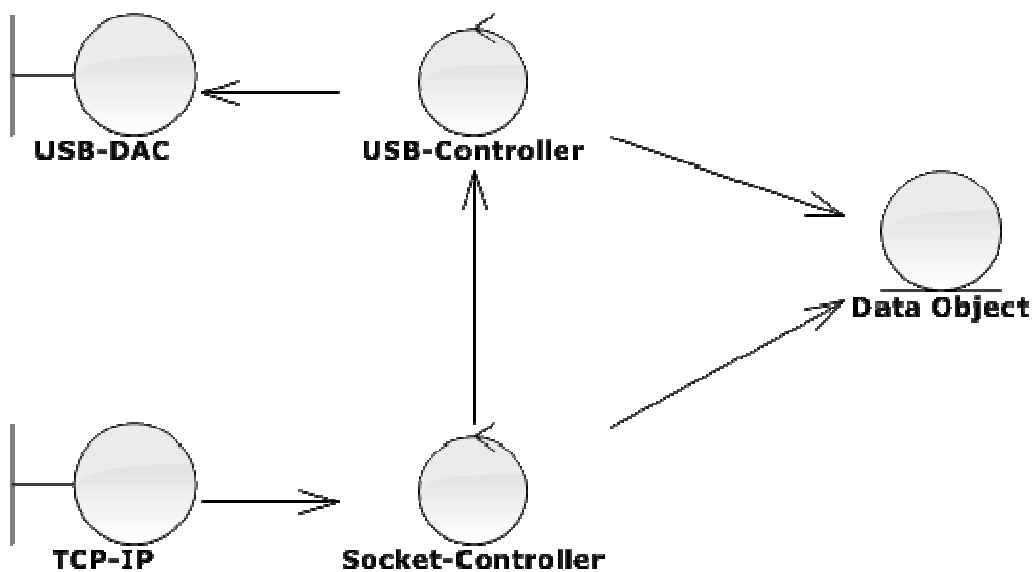
Som vi kan se i figur 15, 16 og 17 så har vi søgt at overholde BCE modellen. Vores projekt er tænkt som et 3 delt applikations framework. I nedenstående figurer (14abc), ses vores originale BCE diagrammer, der har dannet grundlag for designet af klassediagrammet. Diagrammerne har vist sig at kunne anvendes stort set uændret gennem hele projektet.



14a. Brugermodule til kontrol af dataopsamlingen. En GUI interagerer med GUI-controlleren, der igangsætter opsamlingen af data fra Dataopsamlingsmodulet over TCP/IP og lagrer data i databasen.



14b. *Rapportmodul*. GUI interagerer med GUI-controller, der beder report controlleren om at generere en report. Reportcontrolleren beder database controlleren om at hente relevante data.



14c. *Dataopsamlingsmodul*. Henter data fra USB-device og lagrer det i et data object, som socket controlleren videresender over socket forbindelsen til Java komponenten.

Kerneprogrammet er vores Java måleprogram, der interfacer med vores database, C#komponent og DasyLab via vores hotfolder. Hver Boundary har sin egen controller - hhv. DataBaseController, CConnector, DasyFileReader samt LoginController. Desuden har java

programmet 3 boundaries mod brugeren - ILoginGui, BatchMeasureGui og NewBatchGui, der ligeledes har hver deres Controller.

Anden del udgøres af vores Web applikation. Her er boundaryen mod databasen styret af den samme DataBaseController som i Java applikationen. Grænsefladerne mod brugeren - udgjort af jsp-siderne, styres af servlets - således at der er en servlet pr. side - svarende til et *pagecontroller* paradigme.

Tredje del er vores C# server - her er 2 grænseflader mod hhv USB device og TCP/IP - Her fungerer SocketAsync som TCP/IP socket controller og UsbConnection som controller for USB-interfacet til USB-DAQ'en. Data er på dette tidspunkt repræsenteret ved en float og en int og ikke indkapslet i en egentlig entitet.

DataBaseControlleren optræder som controller for grænsefladen mod SQL-Databasen. DataBaseControlleren håndterer alle metodekald fra andre klasser, delegerer ansvaret videre til DAO'erne, der håndterer SQL-kald mod databasen og gemmer entiteterne, samt henter dem op igen når de andre controllere beder DataBaseControlleren om information. Samlet fremtræder DataBaseControlleren som en simplificeret og robust grænseflade til persistent storage for vores DTO'er.

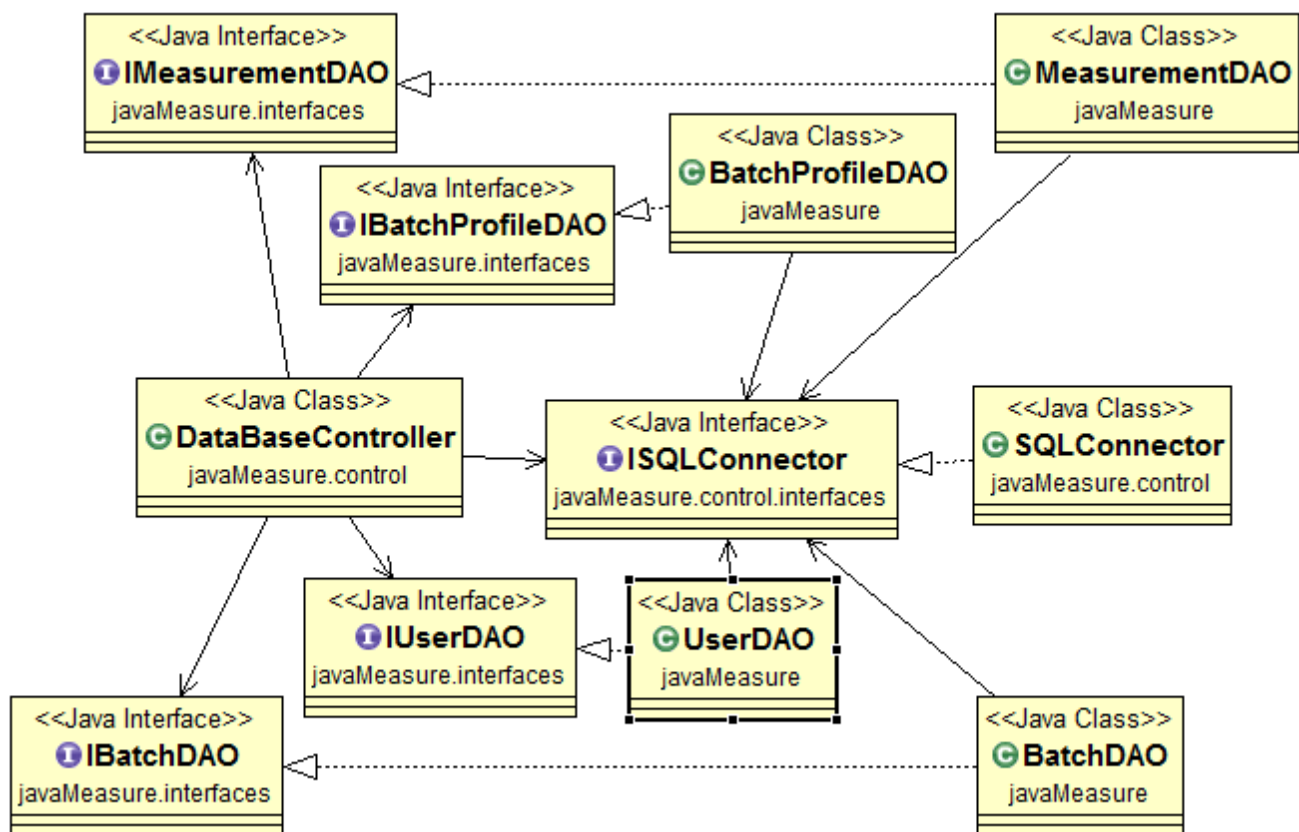
Vi har tilsvarende to andre grænseflader til andre delkomponenter - C# - dataopsamlingskomponenten og vores DasyLab/'hot folder', hvor DasyLab programmet lægger filer som vores program skal parse. CConnectoren optræder som controller for C# dataopsamlingen over TCP/IP boundaryen. Den modtager en forespørgsel om data opsamling og returnerer data i form af et DTO. CConnectorens interface udstiller hvilke forespørgsler data opsamlingsmodulet kan håndtere.

Overordnet set overholder programmerne MVC paradigmet, i det alle grænseflader mod brugeren - 'views' styres af controllers - uden direkte adgang til den bagvedliggende datamodel.

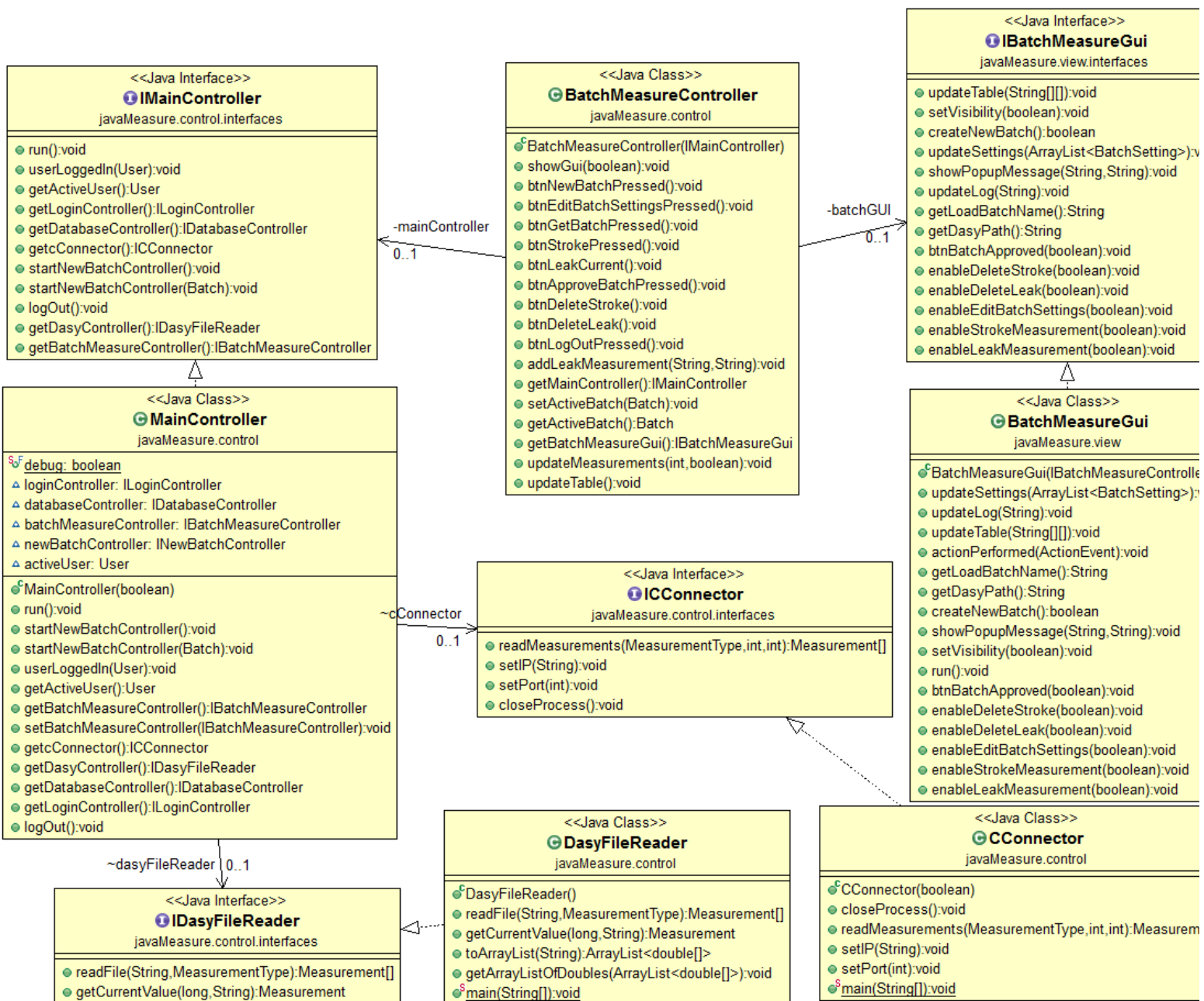
Vi har ligeledes søgt at overholde GRASP - principperne. Controller er grundigt beskrevet ovenfor. Alle klasser oprettes så vidt muligt af den klasse der skal bruge dem (Creator). Endvidere har vi sikret at metoder til at håndtere en type data og data'en er søgt holdt samlet og kun ét sted (Information Expert og High Cohesion) vi har også forsøgt at undgå referencer mellem subcontrollers (Low coupling). Desuden har vi indført 'samleklasser' så en enkelt klasse ikke får for mange koblinger til andre klasser - eks. DataBaseController - der samler funktionalitet for alle DAO'er (Indirection).

Praktisk taget alle klasser, frasat DTO'er er implementeret med interfaces, hvilket sikrer robust, men også fleksibel kode - det er let at udskifte delkomponenter i applikationerne.

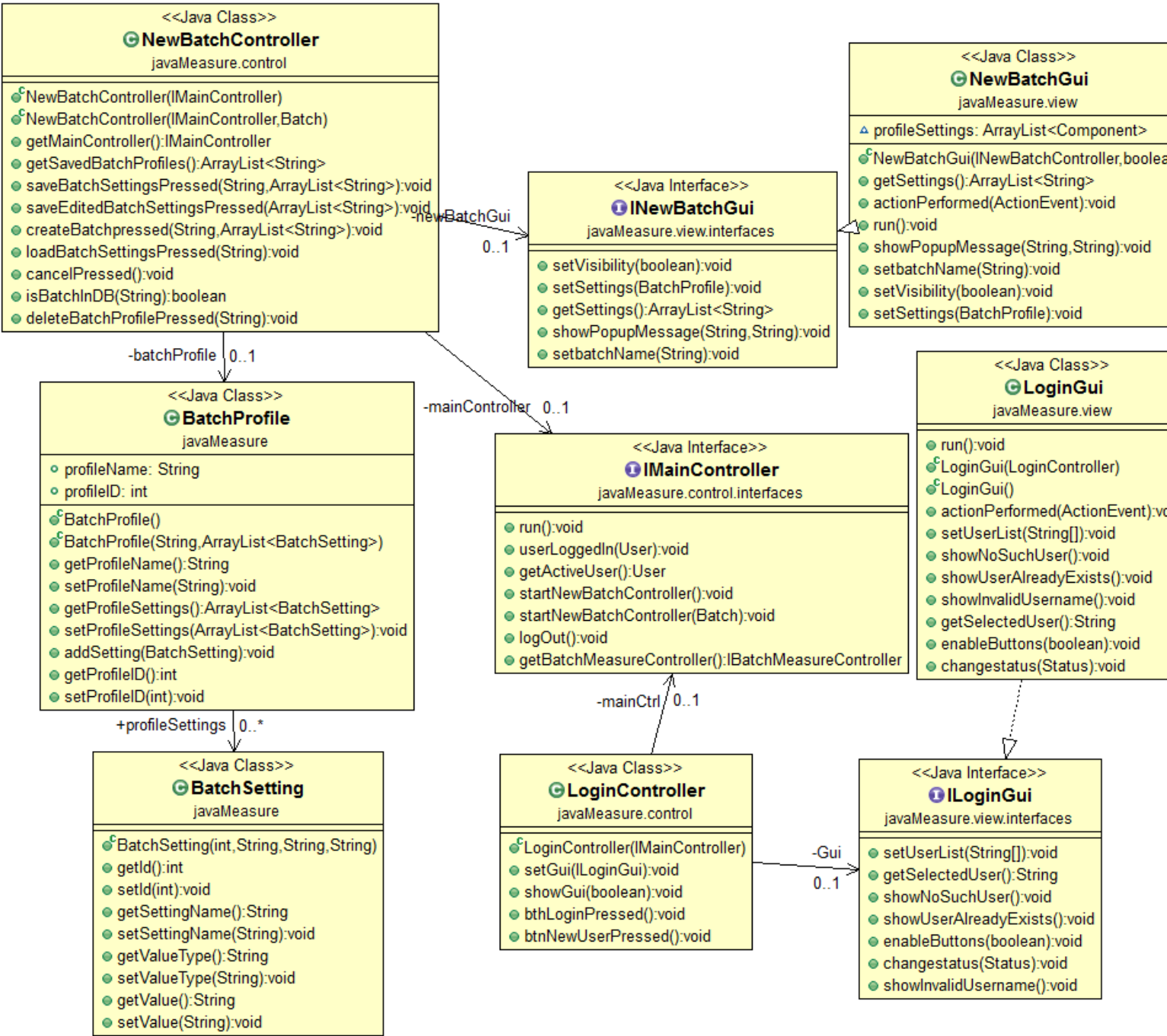
Det har alt sammen sikret et robust og overskueligt projekt som kan udvides modulært, når NOLIAC får behovet.



Figur 15 Overblik over Data Access Layer. DataBaseController fungerer som frontcontroller, der videre delegerer ansvaret til DAO'er.



Figur 16. Klassediagram. Main controlleren styrer to boundary objekter - dasyfilereader og cconnector. Den styrer ellers logincontrolleren, newbatchcontrolleren og batchmeasurecontrolleren. Disse controllere styrer igen diverse gui'er, og dermed er bce modellen fulgt.



Figur 17. Klassediagram. Illustrer anvendelsen af ModelViewController. Al control af userinput og output håndteres af en controller, der håndterer sit eget view. MainController fungerer som facadecontroller, og delegerer ansvar til de andre subcontrollers.

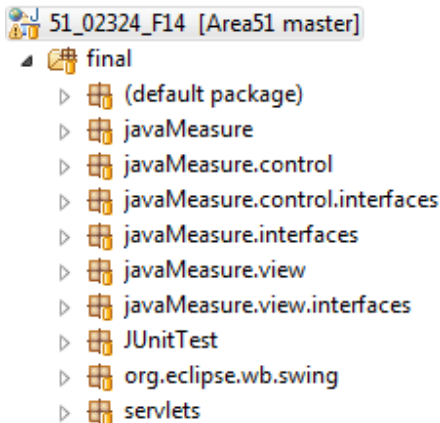
Implementering

Ud fra vores designbetragtninger har vi implementeret vores løsning i nærværende klasser.

Koden er segmenteret i pakker efter funktionalitet.

I default pakken har vi java klasser, der er beregnet til at køres - de indeholder Main metoder. JavaMeasure indeholder DTO'er, DAO'er og hjælperklasser. .control indeholder klasser med control-funktionalitet og .view klasser med gui-indhold.

Hver pakke har en .interfaces underpakke, hvor klassernes interfaces ligger.



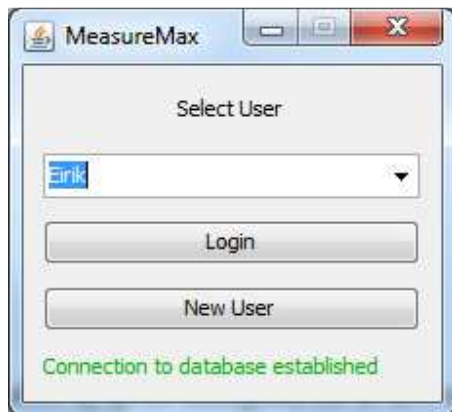
Figur 18: Oversigt over pakkerne i projektet - pakkernes navne bruges som overskrifter i det følgende.

.(default package)

- MainNoTestMode - Starter applikationen i produktionsmode - kræver installeret USB-DAQ software og tilsluttet USB-DAQ.
- MeasureMainTestMode - Starter applikationen i testmode - her returnerer vores C#-serverkomponent dummy data til demonstrationsbrug, og det kræves ikke at det specielle API er installeret.
- NoliacServlet - Starter man webapplikationen herfra, får man det naturlige flow gennem webapplikationen. Modtager alle GET requests som ikke har en modsvarende jsp/Servlet - redirecter til LoginServlet hvis brugeren ikke er logget ind og til MenuServlet, hvis brugeren allerede er logget ind - svarende til at Sessions-attributten 'user' er forskellig fra null.
- Configuration - Lille stand alone program til konfiguration af databasen.

.view

LoginGui

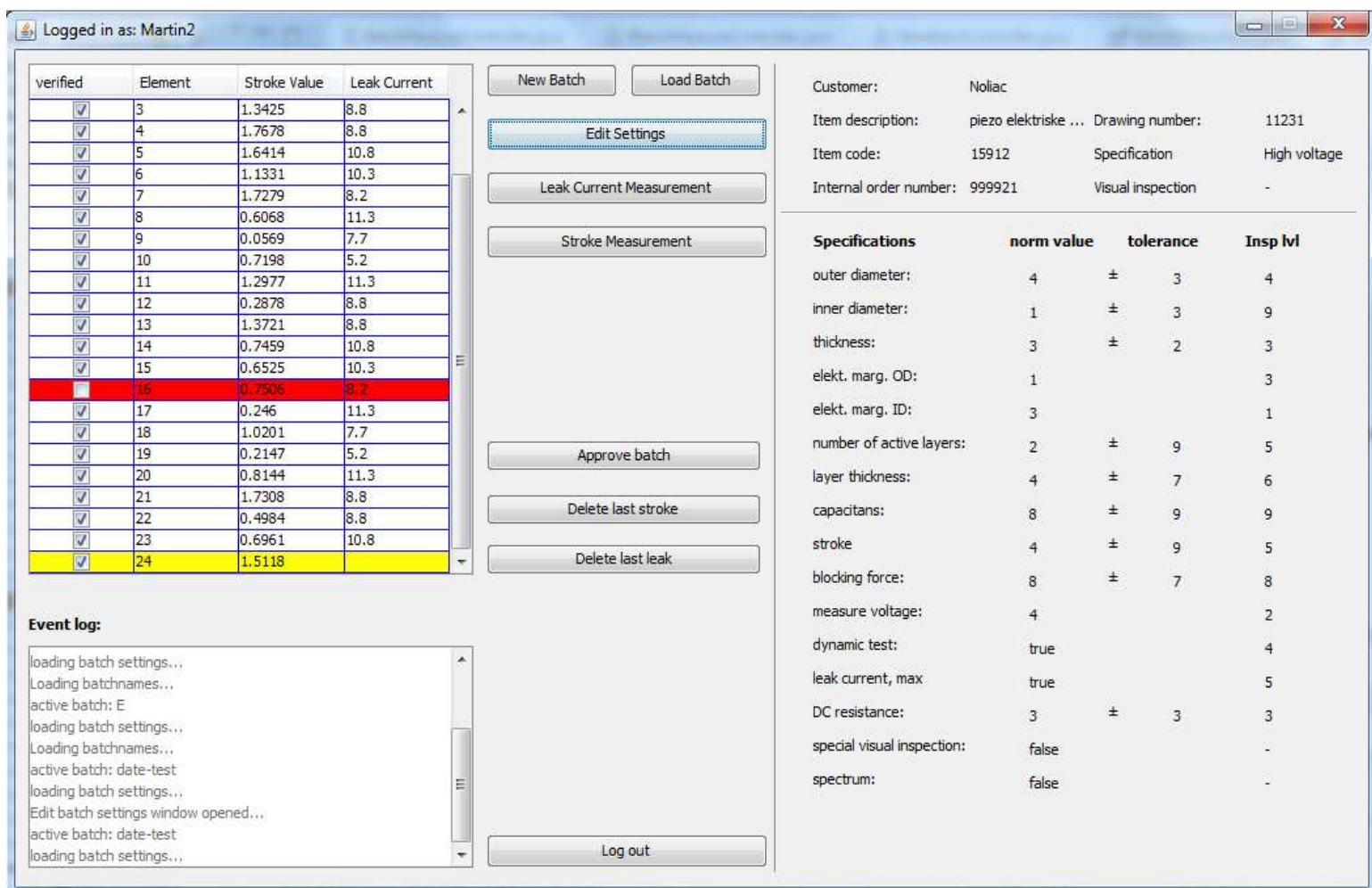


Figur 19. Login

Viser status om man har forbindelse med databasen eller ej, giver mulighed for at lave ny bruger, eller logge ind.

BatchMeasureGui

Viser 9 knapper til oprettelse/loading af batch samt ændring af et batch's settings, lækstrømsmåling, "stroke" måling, sletning af sidste lækstrøm/stokemåling, en approve batch knap og en logud knap. I figur 20 ses BatchMeasureGui'en. I venstre side ses et vindue med målingerne fra det aktive batch, et eventlog, og ude i højre siden vises indstillingerne for det aktive batch. Metoden showPopupMessage bruges til at vise popup beskeder eksempelvis, når brugeren prøver at loade et batch med et navn, der ikke eksisterer.



Figur 20. Her vises BatchMeasureGui, der er hovedgrænsefladen i programmet.

NewBatchGui

Viser fem forskellige knapper til oprettelse af batch, sletning/indlæsning af eksisterende batchprofil, oprettelse af batchprofil, og annuller, som det ses i figur 21. Hvis den er i editMode, viser den kun to knapper, nemlig "Save Batch Settings" og "cancel", hvor "Save Batch Settings" gemmer de ændrede batch indstillinger.

The screenshot shows a software window titled "New Batch". At the top, there are three buttons: "Delete Settings", "Save Batch Settings", and "Create Batch". Below these is a "Batch:" label followed by an empty text box, and two more buttons: "Load Batch Settings" and "Cancel".

Below the buttons, there are several input fields:

- Customer: NASA
- Item description: Secret item42
- Drawing number: 13125
- Item code: 231523
- Specification: (empty)
- Internal order number: (empty)
- Visual Inspection: (empty)

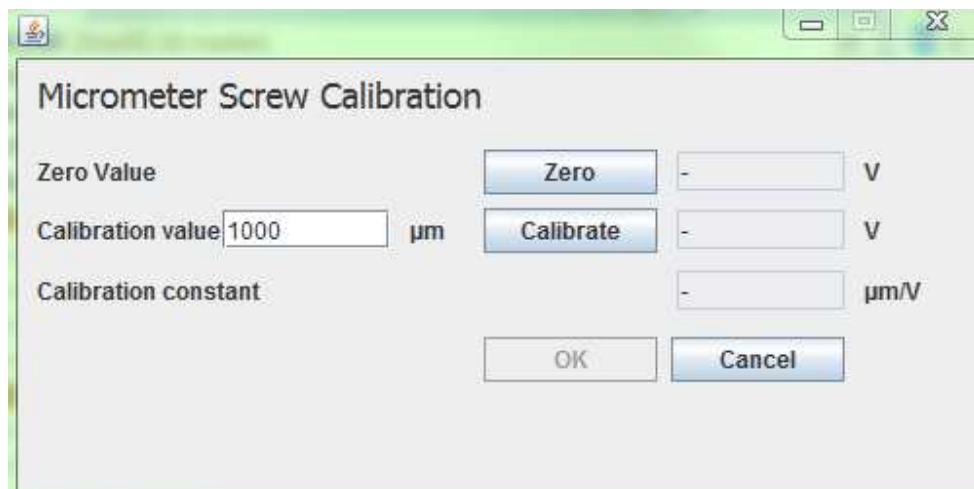
Below these fields is a table-like structure for specifications:

Specifications	norm value	tolerance	inspection level
outer diameter:	45 ±		
inner diameter:	21 ±		
thickness:	±		
elekt. marg. OD:			
elekt. marg. ID:			
number of active layers:	±		
layer thickness:	±		
capacitans:	±		
stroke:	±		
blocking force:	±		
measure voltage:			
dynamic test:	<input type="checkbox"/>		
leak current, max:	<input type="checkbox"/>		
DC resistance:	±		
special visual inspection:	<input type="checkbox"/>		
spectrum:	<input type="checkbox"/>		

Figur 21. NewBatchGui. Specifikationer for målingerne indtastes.

CalibrationGui

På figur 22 vises brugergrænsefladen til at kalibrere programmet til en micrometerskrue, som styres af CalibrationController.



Figur 22. *CalibrationsGui*. Bruges til at finde den kalibreringskonstant, der skal bruges for at omregne fra det analoge signal til μm

.control

LoginController

LoginControlleren sørger for at der bliver logget ind før man kan lave målinger og oprette batches. På denne måde kan vi nemt holde styr på hvilken bruger har oprettet batches, godkendt batches og lavet de forskellige målinger. LoginControlleren kan også oprette nye brugere. Vi tillader ikke dubletter blandt brugernavne i databasen, selv hvis den gamle bruger er sat til at være inaktiv.

BatchMeasureController

Instantierer en BatchMeasureGui og håndterer bruger inputs. `btnNewBatchPressed()` beder maincontroller om at instantiere en `newBatchController`. `btnGetBatchPressed()` beder om et brugerinput, i form af en streng, og henter en batch fra databasen som matcher denne streng, hvis denne eksisterer kan man fortsætte måleserien af batchet. Ellers bliver BatchMeasureGui bedt om at sende en fejl besked til brugeren. `btnStrokePressed()` henter en strokemåling fra CConnectoren og sender den til databaseControlleren, som sørger for at den bliver gemt. Desuden opdateres GUI'en. `btnLeakCurrent()` åbner et søgevindue, hvor den folder DasyLab filerne bliver gemt, skal vælges. Dette skal kun laves i starten, før man starter måleserien, derefter sørger DirectoryListener klassen for resten i en ny tråd. Som en ekstra feature til denne metode gemmer vi den sidste folder brugeren valgte, og bliver denne loadet næste gang, så man kan slippe fra at starte i dokumenter hver gang.

Hvis der er et aktivt batch starter `btnEditBatchSettingsPressed()` en "editmode" version af `newBatchController`, hvis ikke, kommer en besked i loggen som siger at der ikke er nogen aktiv batch. `btnLogOutPressed()` beder maincontrolleren om at logge brugeren ud og derefter lukkes vinduet.

NewBatchController

Håndterer oprettelsen af nye batches og måleprofiler samt ændringer af eksisterende batchSettings. Controlleren har to konstruktører som bliver kaldt alt efter om man skal lave nyt batch eller ændre et allerede eksisterende batch. `saveBatchSettingsPressed()` gemmer de værdier, der er indtastet i tekstfelterne i NewBatchGui'en. Der oprettes først en batchProfile, og derefter

tilføjes batchSettings en af gangen, hvorefter batchProfilen gemmes. createBatchPressed() gemmer først de batchSettings, der er valgt i en unavngiven BatchProfile. Databasen returnerer profilens ID og en Batch oprettes med det tilsvarende profil id. Herefter gemmes batchen i databasen. loadBatchSettingsPressed() henter settings fra databasen svarende til det profilnavn der er valgt i gui'en og beder gui'en om at opdatere settings. deleteBatchSettingsPressed åbner et vindue, hvor brugeren kan vælge en af de eksisterende settings profiler og slette dem. Når man så trykker OK bliver den pågældende Batchprofile hentet og metoden deleteBatchProfile bliver kaldt med pågældende Batchprofile som argument, og den og dens settings bliver slettet.

Når editMode bliver true bliver der kun vist to knapper. saveEditedBatchSettingsPressed() læser først de indtastede settings fra NewBatchGui'en. Derefter kalder den MainControlleren, der henter DatabaseControlleren og opdaterer batchens settings i databasen. Fra MainControlleren får den også BatchMeasureControlleren, der sætter det pågældende batch til activeBatch. cancelPressed() er den samme som når editMode er false.

CalibrationController

CalibrationController er skrevet til at indhente målinger fra mikrometerskruen, således at vi kan kalibrere hvor mange μm en given ændring i spænding på vores USB-DAQ svarer til. CalibrationController har sin egen main() og kan køres som standalone applikation. Da kalibreringen kun burde være nødvendig en gang pr. mikrometerskrue, har vi ikke tilføjet funktionaliteten i selv programmet.

DirectoryListener

Er del af en af de seje ekstra features. Bliver oprettet i ny tråd og holder derefter øje med ændringer i en valgt folder. Alle typer af ændringer bliver fanget, men vi har dog kun haft behov for at behandle de filer der bliver tilføjet mappen. Så hvis man vil teste klassen, kan man copy/paste en af dasyLab filerne i den valgte folder efter folderen er valgt i javaprogrammet.

DasyFileReader

Læser målinger fra en DASYlab fil. Først scanner den filen, spiller hver linje med målinger op i en array af doubles, hvor hver array indeholder tiden til en måling og målingen, og sætter derefter denne array ind i en ArrayList af double arrays. Ved hjælp af metoden getCurrentValue, kan man finde målingen til et givent tidspunkt i millisekunder.

DataBaseController

DataBaseController (DBC) er mellemlidende for databaseoperationer. Vi har så vidt muligt forsøgt at kode controlleren således at den modtager eller leverer et data transfer object (DTO) eller en liste over tilgængelige objekter. Når et DTO overgives til DBC serialiseres det til et eller flere SQL statements, der kan eksekveres på en SQL server. På samme vis kan et objekt deserialiseres ved SQL-statements. Desuden er det muligt at bede om en liste over tilgængelige objekter af en given slags fra databasen.

Databasen indeholder brugere, måleindstillinger, og målinger. Forespørgsler eksekveres via SQLConnector klassen.

CConnector

Denne socket controller styrer kommunikationen med C# komponenten, instancierer den, og lukker den igen. Kommunikationen følger den vedlagte protokol 1. Hvis ingen forbindelse findes, når den kaldes, prøver den at genetablere forbindelsen. Hvis det ikke virker, bliver en fejl sendt

tilbage. Der er også mulighed for C# at returnere andre fejlmeddelelser, som fx. der ingen usb komponent tilkoblet. Da C#Komponenten kræver at en pakke, med drivers og programmer, installeres for at styre USB-DAQ'en, kan denne controller være i et testmode, hvor en anden version af C# komponenten bliver startet, som ikke bruger USB-DAQ'en og derfor heller ikke de ellers nødvendige programmer.

JavaMeasure

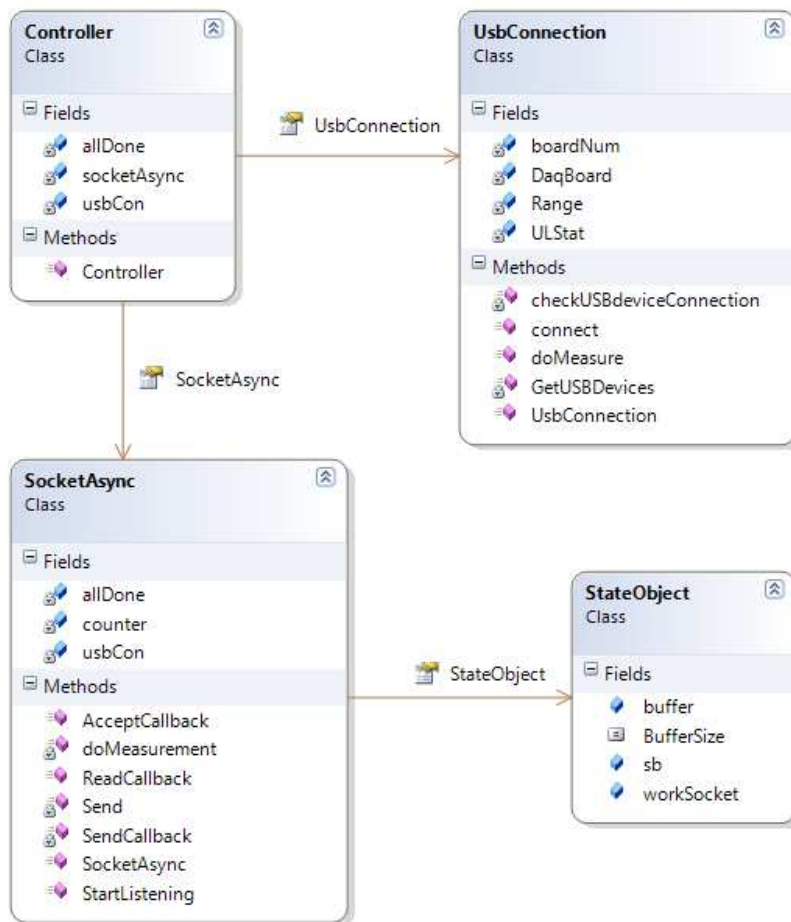
Data Transfer Objects

Vi har 5 DTO'er - Batch, BatchProfile, BatchSetting, Measurement og User. Batch har et ID, et navn og en ArrayList af Measurements samt metoder til at tilgå disse. På samme måde har BatchProfile et ID og en liste af tilhørende BatchSettings. BatchSettings har et ID, navn, valueType og value. MeasureElement repræsenterer det reelle målelement og har 2 tilhørende Measurement objekter samt et element nummer. Measurement udgør de reelle måledata. De kan enten have den enumererede type LEAK eller STROKE. De har et BatchID og et elementNo, der identificerer hvilket Batch og hvilket element de er udført på og desuden en measureValue, der repræsenterer den reelle måleværdi og et timeStamp, der angiver hvornår målingen er udført.

Data Access Objects

Udover regulære CRUD operationer, indeholder DAO'erne nogle convenience metoder. Eksempelvis kan BatchDAO'en rapportere alle navne på batches, hvilket er praktisk i web-søgefunktionens autocomplete feature. Ligeledes har UserDAO en validateUser() funktion, til at kontrollere en brugers login-information, og en canWeRemoveAnotherAdmin() tjekker om det er mere end en aktiv administrator i databasen.

C# Komponent



Figur 23: Klassediagram over C# komponenten. Controller klassen instantierer SocketAsync- og UsbConnection-klasserne. Stateobject-klassen bruges som DTO i den event der genereres, når der modtages data på socket.

C# komponenten står for opsamling af data fra USB-DAQ'en. Vi har udviklet en rudimentær protokol til:

1. at forespørge om en serie af dataopsamlinger med et givent interval fra en given port på USB-DAQ'en
2. modtage en serie af data med identifikation af port, timestamp og måleværdi. I protokollen er også specificeret format for fejlmeddelser.

Den præcise specifikation for protokollen findes i bilag 1.

I figur 23 ses klassediagrammet over C# komponenten, som forklares i de nedenstående afsnit.

Udvikling af C# komponenten var forholdsvis vanskelig, da det viste sig at være lidt af en udfordring at afprøve/debugge to forskellige udviklingsmiljøer (Visual Studio og Eclipse) ved siden af hinanden. Heldigvis fandt vi programmet Netcat, der på en meget simpel måde kunne hjælpe med

at teste om begge komponenter sendte de korrekte svar. Netcat kan forbindes til en socket, og kommandoer kan indtastes i ren tekst.

C# komponent - Controller klasse

Denne klasse sørger for at instantiere USBConnection- og SocketASync-klasserne, samt sørge for at der lyttes på den socket der oprettes.

C# komponent - Asynkron socket listener (SocketASync-klasse)

Socket-klassen i C# komponenten er kodet som en asynkron socket listener. Dette betyder at hovedtråden fortsætter med eksekvering, mens en eventlistener afventer at der kommer noget data fra Javakomponenten. Når data modtages, checkes der om de modtagne data overholder protokollen, hvorefter den ønskede måling foretages.

Når målingen er foretaget, genoptages aflytningen på socketen.

Stateobject-klassen bruges som et DTO for socket eventen.

C# komponent - USBConnection klasse

Denne klasse står for at teste forbindelsen til USB-DAQ'en samt udføre den konkrete analoge måling. Inden målingen udføres, testes det om der er forbindelse til USB-DAQ'en. Forbindelsen testes ved at aflæse de forbundne USB devices der er på maskinen og søge efter det specifikke device ID for USB DAQ'en.

Web-delen

Vi har lavet en hjemmeside hvor Noliacs medarbejdere kan hente rapporter på de forskellige batches, samt ændre brugerne i deres system. Stylesheetet er Noliacs eget - tilpasset vores behov, således at vores webapplikation bedst muligt ligner Noliacs egen web-side www.noliac.com.

En administrator er i vores webinterface en bruger med rettigheder til at ændre i andre brugere, navn, status, osv... vi gør kun forskel på brugere og administratorer hvor det er relevant.

Endvidere er der tjek i alle servlets på om man er logget ind, og hvis man ikke er det, bliver man sendt til LoginServlet. Der er også admin checks, hvor det er relevant.

Noliac Servlet

Sat op til at modtage alle GET-requests der modsvarer en tilgængelig fil på serveren. Redirecter til LoginServlet, hvis brugeren ikke er logget in og til MenuServlet, hvis brugeren er logget ind.

userlogin.jsp

Er siden hvor man logger ind, der er to felter hvor man kan taste brugernavn og password ind.

LoginServlet

Styrer userlogin.jsp, siden hvor brugerne logger ind. Når login knappen bliver trykket tjekkes der om det indtastede brugernavn er i databasen, om brugeren er aktiv, og om det er det rigtige password, hvis alle er rigtige, så bliver man sendt til MenuServleten, hvis knappen ikke er trykket

på, så bliver man sendt til userlogin.jsp.

menu.jsp

Der er muligt at søge efter batchnavne og datoer, og man får matchende batches løbende i tabellen. "I dag", "denne uge", "i går", og "sidste uge" knapperne sætter allesammen de tilhørende datoer i start og slutdatoerne, og udfører søgningen.

Tabellen genereres via et AJAX GET kald til en anden JSP side, hvilket gør at tabellen kan opdateres uden at siden skal genindlæses.

Når brugeren klikker på en række i tabellen, laves et submit af siden via javascript.

MenuServlet

Hvis ingen knap er trykket, så loades menu.jsp. Hvis man har søgt efter en batch, med navn eller dato, og vælger den i menu'en, så bliver den batch gemt i sessionen og man bliver sendt videre til ReportServlet. Edit user knappen sender administratorer til UserChooseServlet, og normale brugere til UserEditServlet. Logout knappen logger brugeren ud, og sender han videre til loginservletten.

report.jsp

Er siden hvor brugere kan se den valgte rapport fra menuen. Der er mulighed for at downloade rapporten i Excel format.

ReportServlet

Styrer report.jsp, hvis ingen knap er blevet trykket, så loader den siden, hvis "ny rapport" er blevet trykket, bliver brugeren sendt til MenuServlet, når "gem som Excel" bliver trykket, gemmer servletten informationen i en fil og henter den ned.

chooseuser.jsp

Er siden hvor administratorer vælger hvilken bruger han vil ændre i. Brugernavn feltet har en dropdown menu der viser alle aktive brugernavne i databasen, og når useren begynder at indtaste brugernavne, så viser dropdown menuen kun de brugernavne der begynder med det der er indtastet.

UserChooseServlet

Styrer chooseuser.jsp. Hvis en normal bruger er logget ind, bliver han sendt tilbage til MenuController. Hvis det ikke er blevet trykket på nogen knap, så bliver chooseuser.jsp loadet. Hvis det er choose, så tjekker vi om brugernavnet der er indskrevet er i databasen, hvis det er det, så bliver den bruger gemt i sessionen.

useredit.jsp

Hvis det er en administrator der er logget ind, så vises der to indtastningsfelter, brugernavn, password, to checkboxe, admin og active. Alle er fyldt ud til at vise den information der hører til den valgte bruger, fx. om de er aktive eller ej. Hvis det er en normal bruger der er logget ind, så kommer kun indtastning feltet password, igen, udfyldt med brugerens kodeord.

UserEditServlet

Styrer useredit.jsp, siden hvor administratoren kan ændre i den valgte bruger (brugernavn, password, admin, active), og hvor brugere kan ændre deres eget password.

Hvis der ikke er submittet form data, bliver useredit.jsp åbnet.

Hvis en normal bruger trykker på save, så bliver brugerens kodeord opdateret i databasen med det der er skrevet i dataindtastningsfeltet, og servleten sender en videre til menu servlet.

Hvis en administrator trykker på save, så bliver den valgte bruger opdateret i databasen med informationen fra dataindtastningsfelterne.

En administrator får lov til at sætte sig selv til at være inaktiv, eller normal bruger, men han må ikke gøre dette, hvis der ikke er andre aktive administratorer i databasen.

Cancel knappen vil altid sende brugeren til MenuServlet.

returnDataAjax.jsp

Denne side genererer den tabel, hvor brugeren kan se de batches, som søgningen i form.jsp returnerede. Siden kaldes fra form.jsp via et AJAX GET kald (Asynkron JavaScript og XML). Der er ikke lavet servlet til siden, grundet tidspres.

Test

En stor del af koden er allerede testet i 13-ugers perioden - Data Acces Layeret er testet konceptuelt som helhed med en Junit test, der indsætter en brugerDTO og validerer korrektheden af dette. De øvrige DAO'er er herefter opbygget på samme måde og funktionaliteten er gennemprøvet ved brugertest og efterfølgende validering af indholdet i databasen. På denne måde har vi kunnet fremprovokere problemer med tegnsætkonvertering, uhensigtsmæssige brugerinput, tab af databaseforbindelse og andre hyppige fejlkilder i et normalt programflow. Vi har således valgt en restriktiv inputvalidering, der kun anvender få tegn ud over alfanumeriske tegn, således at vi undgår tegntabelproblemer - hvilket vi vurderede vigtigere end muligheden for specialtegn. Resten af applikations frameworket er testet intensivt ved brugertest og validering af dataintegritet. Mange null pointer exceptions er nu håndteret med rigtige exceptions og programmet kan håndtere fejlene uden at gå ned. De fleste fejl outputtes til brugerens vinduer - således at fremtidig pilotdrift kan afdække systematiske fejl.

Det vil være meningsfyldt at køre flere strukturerede test af applikationens robusthed, men da applikationen er tænkt som et proof of concept og fortsat vil være udsat for udvikling og omstrukturering, har vi valgt at prioritere at fremvise mulige løsninger på de problemstillinger som Noliac har præsenteret, fremfor et gennemtestet og skudsikkert applikationsframework. Applikationen er derfor primært testet ved at indsætte midlertidige Main-metoder ind, fremfor at udvikle JUnit-tests for hver klasse og metode.

Konklusion

Ikke uventet har vores valg af alternativt projekt givet os nogle ekstra udfordringer. Vi har selv måttet udvikle en kravspecifikation i samarbejde med Noliac og det løbende samarbejde om udformningen af programmet har været utrolig lærerigt, men også ressourcekrævende. Vi har

heldigvis kunnet holde en tæt dialog, men har ikke overraskende været nødt til at ændre specifikationerne efterhånden som projektet skred frem og vi - og Noliac blev klogere på mulighederne i opgaven.

Da vores applikationer skal fungere sammen med eksisterende hardware og software har det været nødvendigt for os at tillære og anvende teknologier ud over pensum. Vi har måttet anvende C# for at kunne interface med vores USB-DAQ, da der ikke er udviklet et Java-API til den. Da den anden af vores målinger foretages med et proprietært program, DasyLab, har vi måttet udvikle en 'DirectoryListener', der holder øje med en fil-mappe og sender evt. nye data til vores applikation, når DasyLab har genereret datafilen. For at brugeren kan få muligheden for at downloade en rapport i csv format, har vi måttet arbejde med at generere rapporten serverside og lade klienten downloade den.

Overordnet set er det lykkedes os at udvikle et produkt der modsvarer forventningerne hos vores kunde, Noliac, samtidig med at vi har fået brugt både de teknologier der er blevet undervist i på semesteret og flere ekstra teknologier ud over pensum. Det har været tidskrævende, men også tilfredsstillende at kunne føre projektet til det niveau vi er på nu - og det er vores indtryk at Noliac er interesserede i at fortsætte automatiseringen i resten af kvalitetskontrollen.

Muligt fremtidigt projekt

Nærværende projekt har løst opgaven omkring et proof-of-concept, der viser at det kan lade sig gøre at foretage målingerne maskinelt og reducere antallet af manuelle handlinger. Der kan ligge et muligt projekt for en gruppe elektro-studerende, hvor integration med Noliacs eksisterende hardware kan være i fokus - eks med automation af to eller flere måle procedurer i et step - styret af relæer.

Litteraturliste

alle tidligere CDIO rapporter gruppen har afleveret.

Bilag

Bilag 1 - Java/C# protokol

The C# component should be set to listen, and java will connect when the program starts. By default, java will try to connect to localhost port 4567, but this can be overloaded.

When the connection has been established, the following requests and responses are allowed.

Request 1:

Java sends a string to C# in the following format: port+";"+number+";"+period+";<EOF>" where port is the desired port on the DAQ to read from, number is the desired amount of readings, and period is the time between readings in milliseconds. port, number and period are ints. <EOF> marks the end of communication.

Response 1:

C# then returns a string in the format: data+";"+timeStamp+";" this is repeated *number* times.

Data is the reading, a float value, and timestamp is the time since the previous reading(long).

If an error has occurred, C# returns a string in the format: ERROR+";"+number+";" error is a string that reads error, and number is an int, that describes what error has occurred.

List of errors

errors from java

- 01: No connection between java and C#
- 02: Return from C# does not follow protocol
- 04: Connection timed out
- 05: Can not connect to C# part
- 06: Connection is already closed.
- 07: Could not properly set Socket timeout
- 08: Could not create either socket writer or reader.
- 09: Could not read what was sent from C#

errors from C#

- 01: DAQ not connected or malfunctioning
- 02: Illegal port
- 03: Illegal number of measurements
- 04: Illegal period
- 05: Message does not conform to protocol

Bilag 2 Bidragydere til projekt

Fil/klasse:	Kodesprog	Bidragyder(e)
Configuration.java	Java	Christian
MainNoTestMode.java	Java	Rúni
MeasureMainTestMode.java	Java	Rúni
NoliacServlet.java	Java	Christian
Batch.java	Java	Morten, Martin
BatchDAO.java	Java	Eirik, Martin, Morten, Rúni
BatchProfile.java	Java	Martin
BatchProfileDAO.java	Java	Eirik, Martin, Morten, Rúni
BatchSetting.java	Java	Rúni
MeasureElement.java	Java	Rúni
Measurement.java	Java	Eirik, Christian
MeasurementDAO.java	Java	Eirik, Rúni
PropertyHelper.java	Java	Rúni
User.java	Java	Eirik
UserDAO.java	Java	Eirik, Christian, Rúni
Validator.java	Java	Rúni, Christian
BatchMeasureController.java	Java	Rúni, Martin
CalibrationController.java	Java	Christian
CConnector.java	Java	Eirik, Rúni, Christian
ConnectionException.java	Java	Eirik
DasyFileReader.java	Java	Martin, Rúni
DataBaseController.java	Java	Eirik, Martin, Christian
DirectoryListener.java	Java	Rúni
LoginController.java	Java	Eirik, Christian
MainController.java	Java	Christian, Rúni
NewBatchController.java	Java	Martin, Rúni
Fil/klasse:	Kodesprog	Bidragyder(e)

SQLConnector.java	Java	Christian
TestDBCtrl.java	Java	Christian
TestMySQLConnection.java	Java	Christian
BatchMeasureController.java	Java	Martin, Rúni
CalibrationController.java	Java	Christian
CCConnector.java	Java	Eirik, Christian
DasyFileReader.java	Java	Martin
DatabaseController.java	Java	Eirik, Martin
LoginController.java	Java	Eirik
MainController.java	Java	Christian
NewBatchController.java	Java	Martin, Rúni
SQLConnector.java	Java	Christian
BatchDAO.java	Java	Eirik, Martin
BatchProfileDAO.java	Java	Eirik, Martin
MeasurementDAO.java	Java	Eirik
UserDAO.java	Java	Eirik
BatchMeasureGui.java	Java	Martin, Rúni
CalibrationGui.java	Java	Christian
LoginGui.java	Java	Eirik, Christian
NewBatchGui.java	Java	Martin, Rúni
BatchMeasureGui.java	Java	Martin, Rúni, Christian
CalibrationGui.java	Java	Christian
LoginGui.java	Java	Christian
NewBatchGui.java	Java	Martin, Rúni
testCCConnector.java	Java	Eirik
TestDatabaseConnector.java	Java	Morten
LoginServlet.java	Java	Eirik, Christian
MenuServlet.java	Java	Eirik, Christian
ReportServlet.java	Java	Eirik, Morten, Rúni, Christian
UserChooseServlet.java	Java	Eirik
UserEditServlet.java	Java	Eirik
returnDataAjax.jsp	JSP + javascript + CSS	Morten
Fil/klasse:	Kodesprog	Bidragyder(e)
menu.jsp	JSP + javascript + CSS	Morten, Eirik, Christian

report.jsp	JSP + javascript + CSS	Morten,Eirik, Christian
userchoose.jsp	JSP + javascript	Eirik
useredit.jsp	JSP + javascript	Eirik
userlogin.js	Javascript	Eirik
Controller.cs	C#	Morten
JavaConverter.cs	C#	Morten, Martin
SocketAsync.cs	C#	Morten
UsbConnection.cs	C#	Morten

directoryListener

```
public class DirectoryListener extends Thread
{
    private Thread thread;
    private String path;
    private Path dir;
    private WatchService watcher;
    private IBatchMeasureController batchMeasureController;

    public DirectoryListener(String path, IBatchMeasureController batchMeasureController)
    {
        System.out.println("initialize: " + System.nanoTime());
        this.path = path;
        this.batchMeasureController = batchMeasureController;

        try {
            this.watcher = FileSystems.getDefault().newWatchService();
        } catch (IOException e1) {
            System.err.println("something went wrong when newWatchService() was called");
            System.err.println(e1.getMessage());
        }
        this.dir = Paths.get(path);
    }

    public void run()
    {
        synchronized(this){
            boolean deletedFile, createdFile, modifiedFile;
            try
            {
                // wait for key to be signaled
                this.watcher = this.dir.getFileSystem().newWatchService();
                this.dir.register(this.watcher, StandardWatchEventKinds.ENTRY_CREATE,
```

```

        StandardWatchEventKinds.ENTRY_DELETE,
StandardWatchEventKinds.ENTRY_MODIFY);

        WatchKey watchKey = null;
        watchKey = this.watcher.take(); // waits until any changes occur
        System.out.println(path + " is being watched");
        while(!this.isInterrupted()){
            String filename = null;
            createdFile = false;
            modifiedFile = false;
            Thread.sleep(1000);

            List<WatchEvent<?>> events = watchKey.pollEvents();
            // one change can trigger up to 3 events but only 2 of them are needed at
the moment
            for (WatchEvent<?> event : events)
            {
                filename = event.context().toString(); // sets filename of file to be
read by dasyFileReader

                if (event.kind() == StandardWatchEventKinds.ENTRY_CREATE)
                    createdFile = true;

                if (event.kind() == StandardWatchEventKinds.ENTRY_MODIFY)
                    modifiedFile = true;
            }

            if(createdFile && modifiedFile) // if file is pasted these events are
triggered
                batchMeasureController.addLeakMeasurement(path, filename);
        }
    } catch (Exception e){
        System.out.println("Error: " + e.toString());
    }
}
}
}

```