CDIO opgave forår 2014 - Del 4

Alternativt projekt

Dataopsamling fra tests af piezoelektriske krystaller.

Morten Hesselbjerg, s017704
Christian Budtz, s134000
Rúni Egholm Vørmadal, s134004
Martin Nielsen, s123064
Eirik Oterholm Nielsen, s123006

51_02324_F14_D4

Dato	Navn	Design	Impl.	Test	Dok.	Andet	I alt	Bemærk
04/05/2014	Martin		5				5	
05/05/2014	Martin		5				5	
10/05/2014	Christian		1				1	
10/05/2014	Rúni		7	1		3	11	
	Morten				1		1	
	Eirik		4				4	
10/05/2014	Martin		7				7	
11/05/2014	Martin		2		2		4	
11/05/2014	Christian		2	1	2		5	
11/05/2014	Rúni		2	1	3		6	
11/05/2014	Morten				5		5	
11/05/2014	Eirik		2		2		4	
	Total						48	
	Christian	0	3	1	2	0	6	
	Eirik	0	6	0	2	0	8	
	Martin	0	9	0	2	0	11	
	Morten	0	0	0	6	0	6	
	Rúni	0	9	2	3	3	17	
							48	

Indledning

På grund af tidspres har vi nedprioriteret nogle af implementeringerne og lagt kræfterne i html/JSP delen. Vi har lavet en side, hvor en bruger kan logge ind med det brugernavn som er gemt i databasen. Derefter kan han hente en slutrapport ved indtastning af batch id. Mulighederne er begrænset, da vi mangler noget data, som skal indtastes i javaprogrammet, og derfor ikke ved helt hvordan det skal behandles. Vi havde et møde med Noliac 30/4 hvor vi præsenterede produktet og fik feedback.

Rapporten indeholder kun elementer, hvor der er sket noget siden D3-afleveringen, og det kan være nødvendigt at læse D3 for at forstå alle sammenhænge.

Status

Det grundlæggende programflow i java-progammet er ikke ændret væsentligt - Vi har præsenteret det for Noliac og de er tilfredse med GUI'en og måden man navigerer i programmet. Data kan opsamles fra to forskellige kilder, hhv. fra et USB-device tilsluttet computeren og fra DasyLab filer genereret af kundens egen software. USB-modulet kan opsamle analoge data fra en tilsluttet mikrometerskrue. Selve tilslutningsprogrammet til USB devicet er udviklet i C#, da der kun findes drivere til dette. Når C# programmet kører, opstarter man et Java-program og kommunikation med C# programmet foregår så over en TCP/IP forbindelse via vores egen protokol. Al data gemmes for hver måling i DTU's mySQL database og det er muligt at genoptage en måleserie (batch), der er blevet afbrudt før tid.
Vi har præsenteret vores web-mockup for Noliac og har brugt deres feedback til at ændre lidt på udformningen af siderne samt udviklet en egentlig funktionel web-applikation. Man kan nu logge ind via hjemmesiden, udsøge et batch og se en rapport genereret ud fra data i databasen.
Vi har arbejdet lidt på robustheden og exception handling er udvidet en anelse til at give mere meningsfyldte exceptions og tilsvarende beskeder.

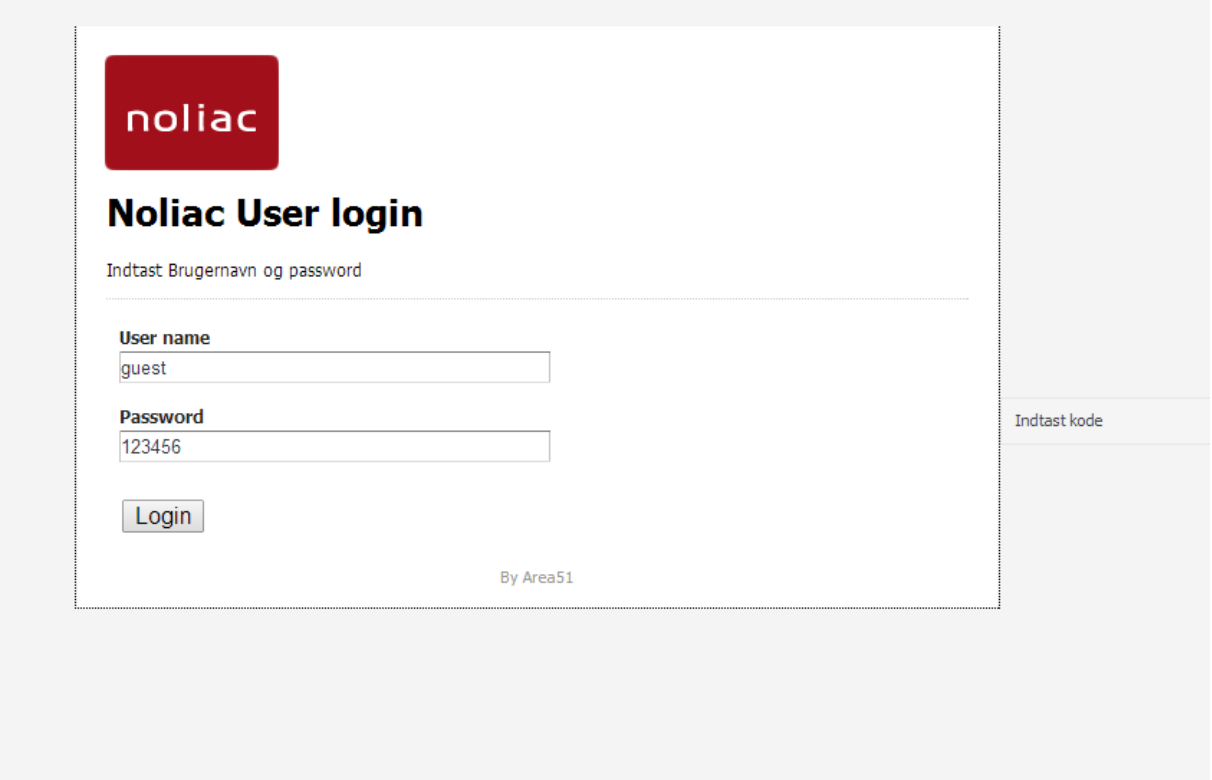


Fig.1 Login side -userlogin.jsp (welcome file). Felterne er forudfyldte til testbrug - forsøger man at ændre password nægtes adgang. Der anvendes stadig en textbox til password (med synligt password), da det er lettere at teste (bliver en password box senere).

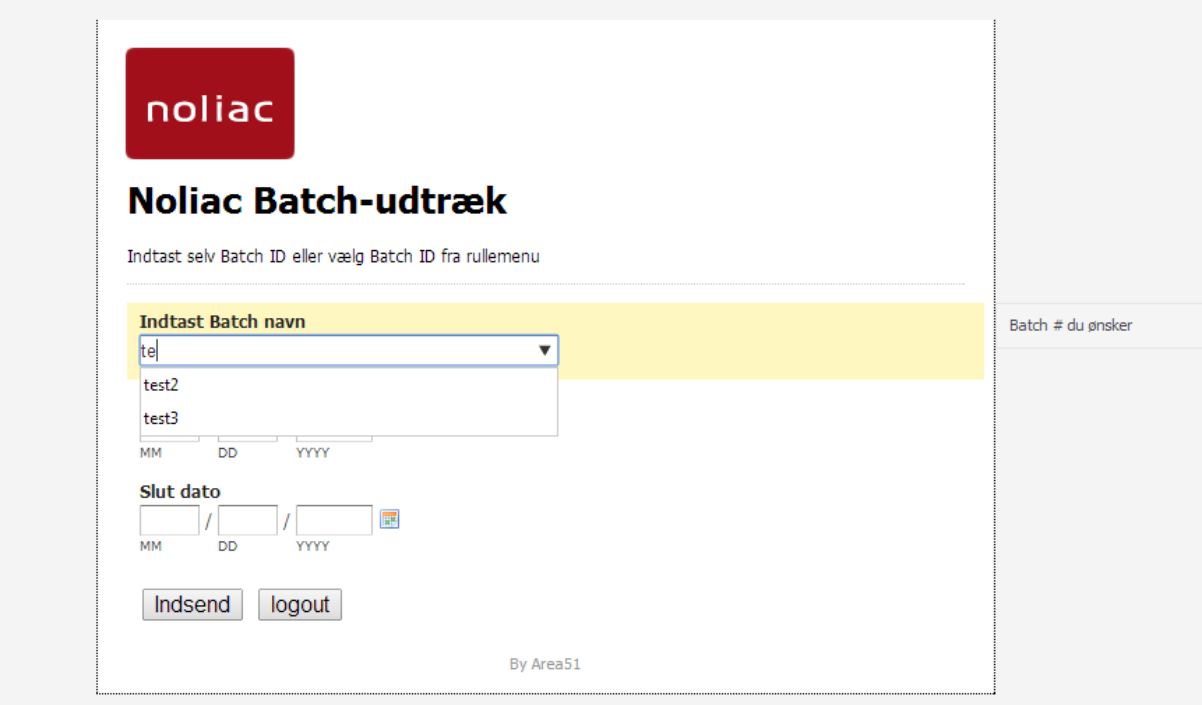


Fig.2 Udsøgning af batches - form.jsp. Når indtastning påbegyndes vises forslag fra databasen i drop down box. Det vil være muligt at filtrere på dato og muligvis andre kriterier på et senere

tidspunkt.

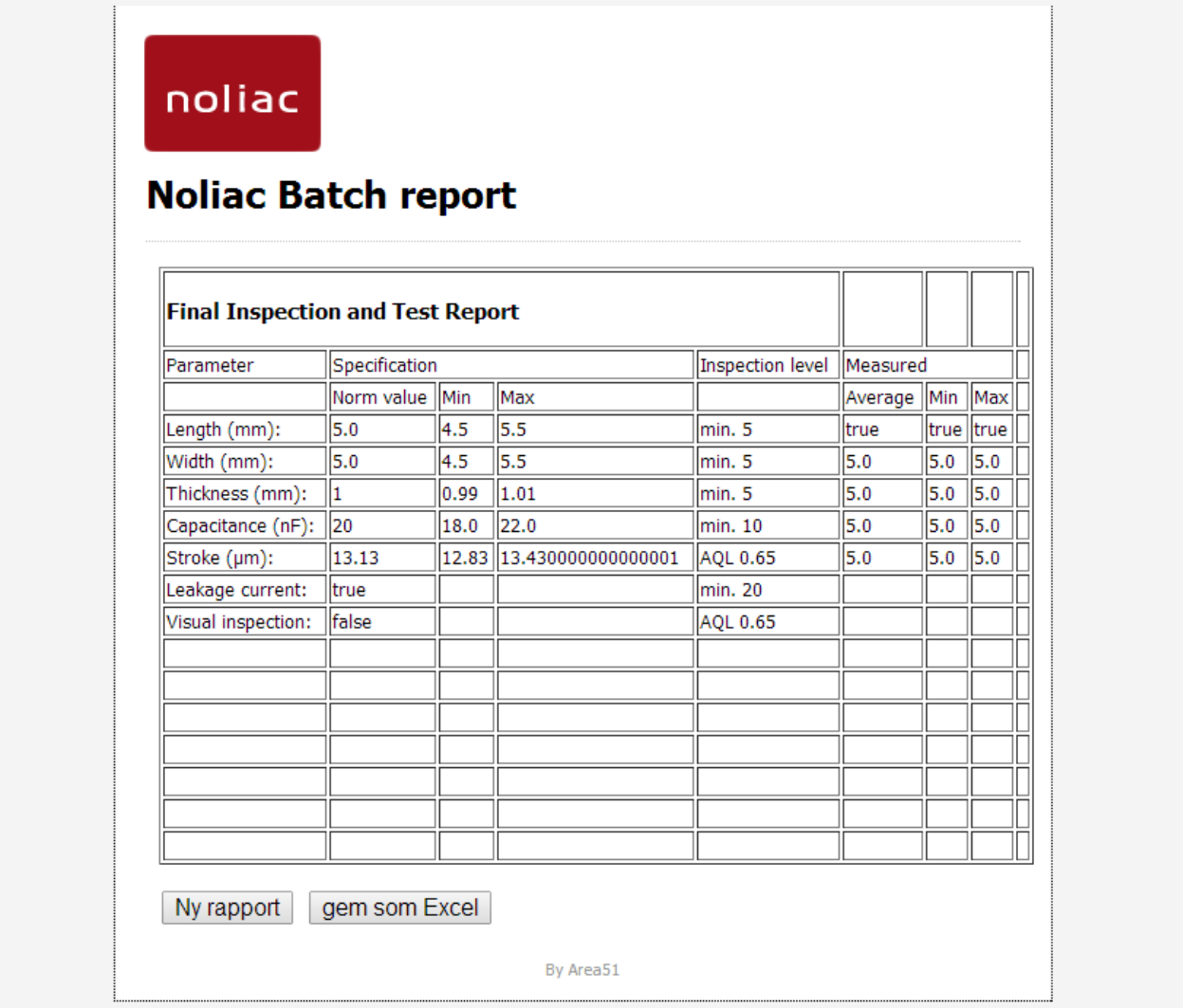


Fig.3. Selve rapportsiden report.jsp. Rapporten genereres ud fra data hentet i databasen. Det er ikke muligt at generere et excel ark endnu.

Kravspecifikation

Vi har ingen væsentlige ændringer i kravene til vores projekt. Vores måleproces består fortsat af to separate målinger - hhv. en *stroke* måling og en *leak current* måling. Stroke målingen forudsætter at vi opsamler data over USB porten via et data aquisition device. Leak current målingen foretages i kundens software og genererer en .asc-fil, som vi så indlæser i vores program og trækker relevante data ud af.

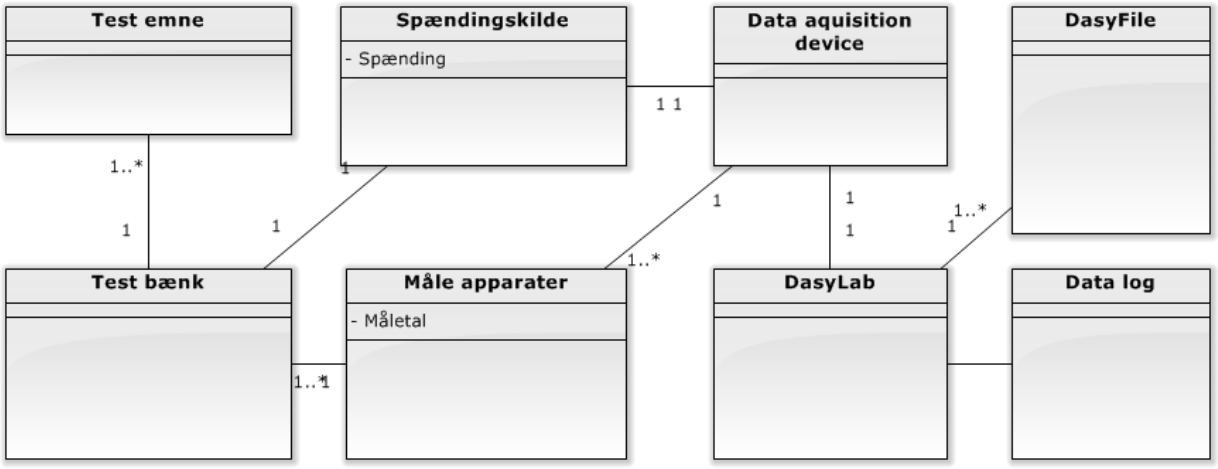


Fig 4. Domæne model. Ingen væsentlige ændringer.

Use Cases

Main use cases er i væsentlige træk uændret, men er opdateret til at reflektere den løbende udvikling i implementeringen.

Design

Vi har holdt fast i opdelingen af vores kodebase i Data transfer objects (DTO'er), Controllers og GUI kode. Vi har 5 DTO klasser - Batch, BatchProfile, BatchSetting, Measurement og User. Desuden har vi en utility-klasse - PropertyHelper, der anvendes til at hente programindstillinger fra en properties fil.
Siden sidste iteration har vi valgt at segmentere database-delen yderligere, således at alle DTO'er har deres eget data transfer object (DAO). Vi havde placeret alle databasekald i DatabaseControlleren og med flere metoder og exceptionhandling var den efterhånden svær at overskue. Vi har bevaret DataBaseControlleren, men den håndterer nu kun instantiering af DAO'er og sender alle metodekald videre til relevante DAO'er (se fig. 5).
Det overordnede programflow håndteres af MainController, der fungerer som facadecontroller og delegerer ansvar videre til relevante sub controllers. Hvert View har en egen controller klasse - BatchMeasureController, LoginController, MainMenuController og NewBatchController.
Indhentning af måledata sker i CConnector og DasyFileReader. Når data skal gemmes/hentes i databasen håndteres det af DataBaseController, relevante DAO'er og SQLConnector.
For yderligere udspecificering af design og klassediagrammer henvises til tidligere rapport.
For at sikre robusthed i tilfælde af nedbrud eller præmatur afslutning af måleserie gemmes data efter hver enkelt måling. Vi bruger aktuelt en forbindelse til den mySQL database vi har fået stillet til rådighed i Diskret matematik og databaser.

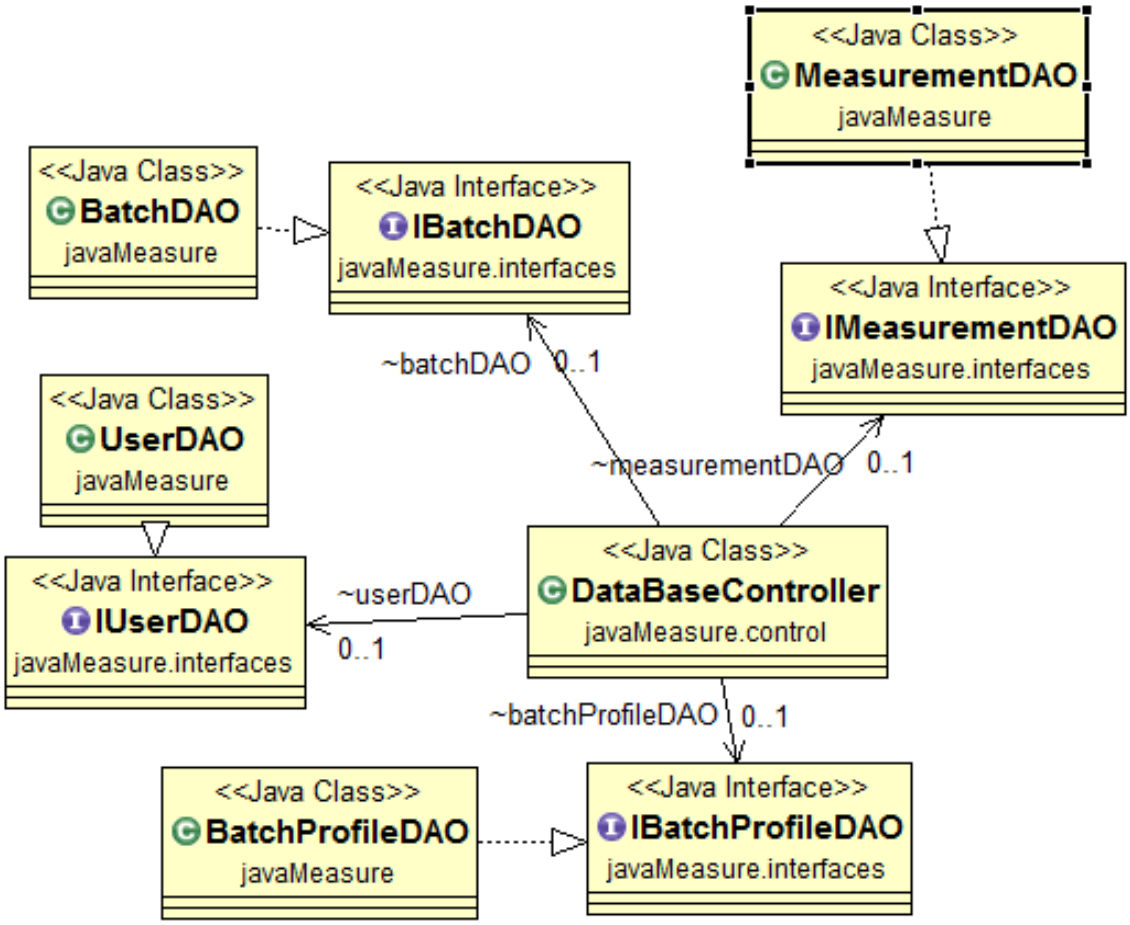


Fig. 5. Refaktorering af databasedel. DTO'er har fået egne DAO'er. DataBaseController er bevaret til at håndtere instantiering af og delegering af metodekald til DTO'er. Se bilag 1 for klassediagram med metoder inkluderet.

Implementering

JSP/Web

Der er tilføjet JSP kode til at håndtere login, udsøgning af målebatches og generering af rapporter. Indtil videre er det muligt at logge ind og generere en standard rapport.
Det er planen at udvide funktionaliteten i 3-ugers forløbet, så der kan genereres rapporter til Excel.

userlogin.jsp

I userlogin.jsp anvendes den samme databasecontroller som vores Java-program anvender. Brugen af databasecontrolleren er implenteret som en JavaBean. På den måde genbruges let de DAO'er vi har skrevet til at tilgå data fra SQL databasen. Når man submitter formen

valideres indtastede data mod SQL,databasen. Der er anvendt prepared statements, så SQL-injection er ikke muligt.

form.jsp

Når siden loades, checkes først om brugeren er logget ind. Hvis ikke, redirectes brugeren til login-siden. Formen er pt en POST form, men det er planen at ændre den til en GET senere, så det er muligt for brugere at gemme et link til en specifik rapport/rapportkriterier.

report.jsp

Ligesom med form.jsp, checker report.jsp også om brugeren er logget ind.
De hentede data skrives direkte i en statisk tabel. Det er planen at der skal laves kode, der kan generere tabellen (i html) dynamisk. Derudover mangler der at blive implementeret kode til at gemme rapporten som en Excel eller CSV-fil.

Java

Data Transfer Objects

Som nævnt har vi 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 måleelement og har 2 tilhørende Measurement objekter samt et element nummer. Measurement udgør de relle måledata. de kan enten have den enummererede 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 er 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.

BatchMeasureController

Instantierer en BatchMeasureGui og håndterer bruger inputs. btnNewBatchPressed() beder maincontroller om at instantiere en newBatchController. btnGetBatchPressed() henter en batch fra databasen, således at man kan fortsætte fra en tidligere måleserie. btnStrokePressed() henter en strokemåling fra CConnectoren og sender den til databaseControlleren. Desuden opdateres GUI'en. btnLeakCurrent() beder dasyController om at parse den fil der er valgt aktuelt og udhenter værdien til et særligt tidspunkt, hvorefter den gemmes i databasen. btnLogOutPressed() beder maincontrolleren og at logge brugeren ud.

NewBatchController

Håndterer oprettelsen af nye batches og måleprofiler. 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.

Vi har desuden følgende 4 controller klasser, der sørger for data input/output:

DataBaseController

DataBaseController (DBC) er mellemled for databaseopereationer. 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 objecter. 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 tilgængeligt objekter af en given slags fra databasen.
Databasen indeholder brugere, måleindstillinger,og målinger. Queries eksekveres via SQLConnector klassen.

NewBatchGui

Viser 5 knapper til at oprette et batch, load/save/delete af settings og annuller. setSettings() opdaterer de viste settings i gui'en. NewBatchGui har elementer af control, der skal flyttes til controller klassen når tiden tillader. I actionPerformed() håndteres oprettelsen af et nyt batch - createBatch, load af settings fra databasen - loadSettings og save af settings - saveBatchSettings. Der arbejdes på en edit funktion, så det vil være muligt at ændre settings profiler.

Begrænsninger i nuværende implementering

Der er stadig de samme begrænsninger som i D3, som kan ses nedenfor.
C# programmet skal aktuelt startes manuelt - for brugervenlighedens skyld planlægger vi at det skal startes automatisk. Log-vinduet er ikke up to date, og der kommer intermitterende meddelelser, som ikke stemmer overens med fejltilstanden. Målingerne skal foretages konsekutivt og det er aktuelt ikke muligt at kassere enkeltmålinger. Det er ikke muligt at bruge programmet uden en aktiv SQL-forbindelse. C# serveren skal eksekveres på samme maskine som Java-programmet.

Test

Vi har udført både brugertest af programmet i sin helhed og desuden unittestet de komplicerede datakommunikationsklasser.
Da vi har concurrency på flere niveauer i vores program, har vi også mange muligheder for at få fejl, når forskellige tråde ikke er klar til at håndtere interaktion med andre tråde. Således opstår en null-pointer exception, hvis det lykkes brugeren at trykke på login-knappen, før der er hentet data fra SQL-databasen. Arbejder man på DTU, ses denne fejl ikke, da data hentes for praktiske formål instant, men her er mulighed for at forbedre robustheden. Samme type af problemer kan provokeres andre steder i programmet, men giver ikke anledning til nedbrud - det er muligt at fortsætte med at anvende programmet uden at data korrumperes.
De enkelte metoder til vedligehold og oprettelse af SQL forbindelsen i SQL-connectoren er testet ved test driver metoder i klassen TestMySQLConnection.

Vi har oprettet en JUnit test der tester DatabaseControlleren, ved at tilføje en ny bruger til databasen, søge efter den for at sikre at den er indsat korrekt, og til sidst slette test-brugeren igen.

Udviklingsplan

Vi er kommet bagud på nogle punkter, som vi selvfølgelig kommer til at lave i 3 ugers perioden. Grund elementerne i JSP/html er lavet, og de resterende funktioner skal implementeres. Noget mere data skal kunne tilføjes, og det skal vi finde passende pladser til. Funktionalitet, som vi i dialog med Noliac har tilføjet projektbeskrivelsen, er indsat i planen.

3-ugers plan

Bugfixing, optimering - videreudvikling af web-brugergrænseflade med funktioner og udvidelse af implementering

1. Kravspecificering
    a. Løbende tæt dialog med Noliac, mhp. validering af løsningsforslag og prioritering af udviklingsopgaver.
2. Design
    a. Nuværende design opdateres efter behov.
3. Implementering
    a. Optimering af kodebase/GUI
        i. Test af grænsetilfælde
        ii. Inputvalidering
        iii. Bedre exception handling
        iv. Bedre/mere robust håndtering af SQL-connection
    b. Manglende funktionalitet i Javaprogram
        i. Superbruger funktionalitet
        ii. Måleprofil redigering
        iii. Verifikation af måledata - mulighed for at kassere fejlagtige måledata

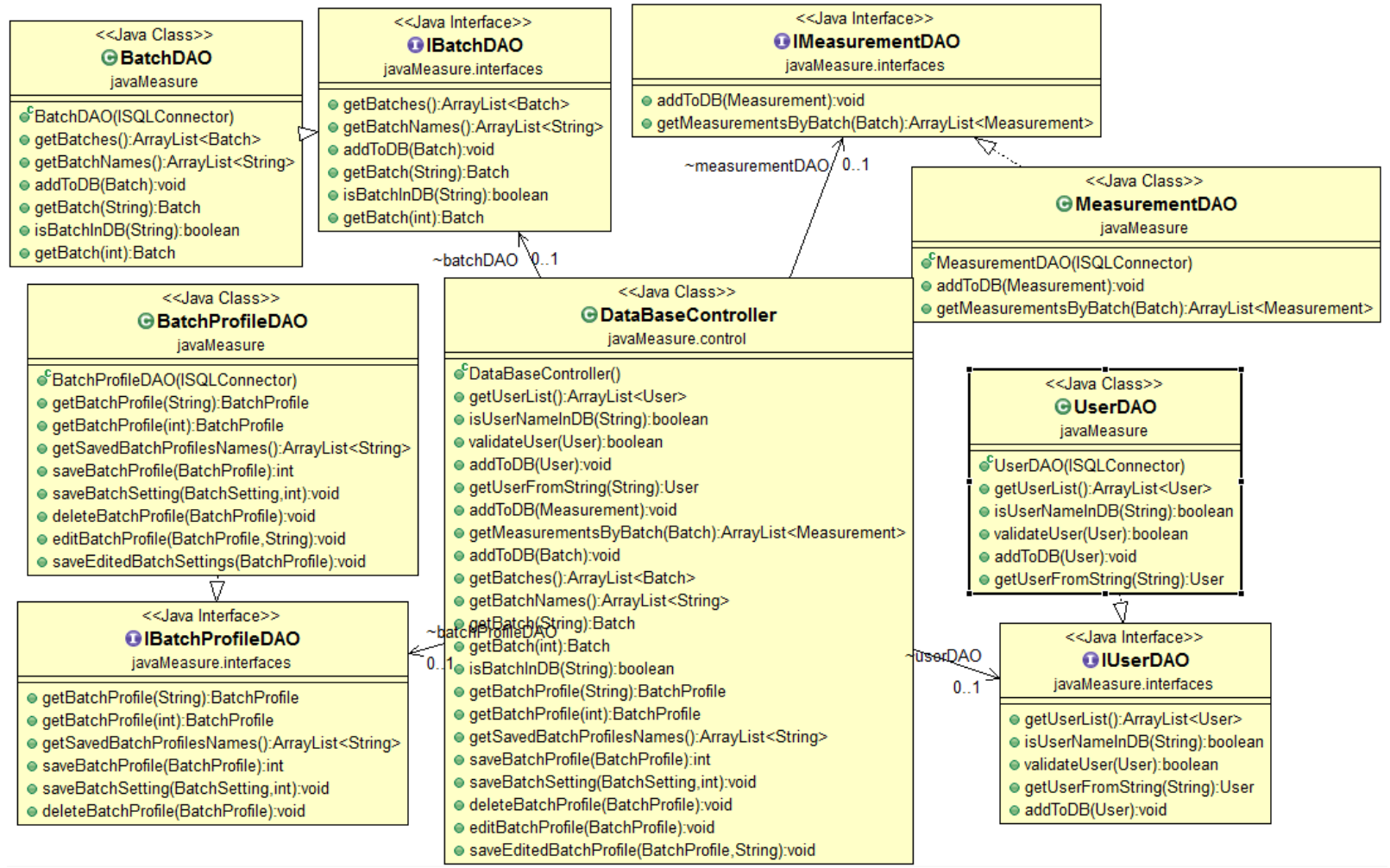
iv. Fleksibilitet af måleflow - så man evt. kan bestemme hvilket element, der bliver målt på
v. Program setup - mulighed for at configurere eks IP-adresser, måleporte, standard profiler.
vi. Gemt data, samles i én property fil (nu gemmes data i flere properties)

c. Implementering af SQL/JSP i HTML/CSS grænseflade
i. Grundlæggende funktionalitet til at udhente de indsamlede måledata.

4. Test.
   a. Videre Unit testing af essentielle databehandlingsklasser
   b. Gerne brugertest med Noliac
5. Evt:
   a. Implementering af 'custom' rapportgenerering.
   b. Måske rapportopslag for kundens kunder efter batchnr?
   c. Statistisk udregning af sample size for at garantere en vis tolerance med en given sandsynlighed.
   d. Brugerstyring
   e. Større robusthed af database-interface
      i. evt. offline mode og synkroniserings funktionalitet (lav prioritet).

Mål: Optimering af løsning, evt. implementering af flere use cases. Kunden skal kunne bruge programmet i drift i en basal udgave.
Slut: 11/5

Bilag



Bilag 1: Klassediagram med vigtige metoder.