DTU Informatik
Informatik og Matematiks Modellering

Spring 2020
Hubert Baumeister

02161 Software Engineering 1

Obligatorisk Opgave

Version 1, 8. marts 2020

1 Opgaven

I afsnit 5 findes projektbeskrivelsen, der vedrører udvikling af et planlægningsværktøj til brug i forbindelse med styring og planlægning af softwareudviklingsprojekter, herunder registrering af arbejdstidsforbrug på de forskellige projekter.

På baggrund af dette oplæg skal der udarbejdes

1. kravspecifikation og programdesign (rapport 1)
2. Feedback til en anden gruppes rapport 1 (på CampusNet)
3. det endelige program og test, (rapport 2 plus kildekode som Eclipse eller IntelliJ projekt)

således at de væsentlige begreber fra oplægget kan spores i det udviklede program.

Det er ikke alle aspekter, der omtales i oplægget, der skal medtages i det system der specificeres og udvikles. En del af denne øvelse består i at afgrænse opgaven, således at det er muligt at udvikle et sammenhængende produkt (rapport og tilhørende program) af høj kvalitet indenfor den afsatte tid. Men det er også vigtigt at der tages stilling til, hvordan systemet kan udvides med de aspekter, man så bort fra i forbindelse med afgrænsningen.

Programmets applikationslag skal udvikles test-dreven. Til hver funktion i applikationslag skal der findes en eller fler svarende Cucumber scenarier eller JUnit tests som sikrer at funktionen gøre det den skal (ligesom vi har gjort i øvelsenser.) Brugergrænsefladen behøves ikke udvikles test-dreven. Ved projekt demonstration skal I vise testene.

Ved bedømmelse af rapporterne vil der blive lagt vægt på

- fuldstændighed, dvs. at de valgte funktioner bliver komplet behandlet, og
- sporbarhed, herunder konsistens igennem alle abstraktionsniveauer.

Det er bedre at sikre disse kvalitetsegenskaber igennem en bevidst afgrænsning af problemet, end at brede sig for meget ud over problemstillingen.

2 Opgave Del 1: Kravspecifikation og Programdesign

Rapport 1 skal indeholde følgende afsnit.

1. *Kravspecifikation*: Afsnittet har følgende underafsnit:

   (a) *Indledning*: Dette afsnit skal indeholde en kort indledning og opsummering af besvarelsen.

   (b) *Væsentlige begreber (ordliste, glossar)*: Dette afsnit skal indeholde en opremsning af de væsentlige begreber fra problem domænet. Hvert begreb gives en kort definition på nogle få liner.

   (c) *Use case diagram*: Dette afsnit skal give en beskrivelse af de væsentlige operationer i form af et use case diagram. Hvis et use case diagram bliver uoverskueligt, kan der også bruges flere use case diagrammer. Husk et formålet af use case diagrammer er et skabe en oversigt over programmernes funktionalitet.

   (d) *Detaljeret use cases* Der skal også være detaljeret use cases med hoved– og alternative/fejl scenarier som Cucumber features / Cucumber scenarier. Der skal være mindst 8 detaljeret use cases for firmandsgrupper.

   Afsnittet afsluttes med en kort diskussion, hvor, for eksempel, uklarheder i oplægget diskuteres, og hvor der gøres rede for valg og afgrænsninger.

2. *Programdesign*: Afsnittet har følgende underafsnit:

   (a) *Klassediagram*: Dette klassediagram skal være bindeledet mellem kravspecifikation og program. Klasserne skal indeholde væsentlige attributter og operationer med typer, således at det fremgår hvordan de væsentlige begreber modelleres og samtidigt skal diagrammet give et overblik over programdesign.

   (b) *Sekvensdiagrammer*: Dette afsnit indeholder sekvensdiagrammer for de use case scenarier fra afsnit 1d, således at læseren få et overblik over hvordan programmet bør udføre disse user stories. Der skal være 8 for firmandsgrupper og 6 for tremandsgrupper.

Afsnittet afsluttes med en kort diskussion hvor der, for eksempel, gøres rede for de valg der er truffet. Afsnittet kan også omtale valg af algoritmer og datastrukturer.

Slidserne i uge 6 anbefaler en process som kan bruges til at lave programdesign.

3 Opgave Del 2: Peer Review af Kravspecifikation og Programdesign

Inden start af uge 8 får I rapport 1 fra en anden gruppe. Ved brug af CampusNet opgave modul skal I give feedback til denne rapport. Feedback skal fokusere på konstruktive feedback vedrørende:

1. Er notationen brugt korrekte? For eksempel, bliver use caserne i use case diagram vist som oval? I kan bruge `http://uml-diagrams.org` og "UML User Guide" hvis I er i tvivle om notationerne og I kan spørge mig. Til "UML User Guide" findes der et link til online version på kursets Web side.

2. Er notationen brugt konsistent? For eksempel har use caseren i use case diagram samme navn som de detaljerede use cases? Svarer use-case scenarier til sekvensdiagram?

3. Er notationen fuldstændig? For eksempel forklares der alle domænekoncepter som bruges i use case beskrivelsen?

4. Er kravspecifikation og programdesign nemt at forestår? Ville det være muligt for jer at implementere program ud fra design?

4 Opgave Del 3: Implementering og Test

Der skal afleveres jeres endelige version af kildekoden, testene (Cucumber og JUnit), det kørende program og en instruktion hvordan man kører programmet som Eclipse eller IntelliJ projekt. Projektet skal indpakkes i en ZIP fil. Eclipse projekt skal kunne importeres som Eclipse projekt i Eclipse. Når man udpakker InjelliJ projektet så skal det være muligt at åbne projekt i IntelliJ. Det er vigtig at i sikrer jer at jeg kan åbne jeres projekter med enten Eclipse eller IntelliJ og at jeg kan køre program og alle testerne og der mangler ikke nogen filer eller jar-filer.

Desuden skal rapport 2 indeholde følgende afsnit.

1. *Systematiske test (white box test)*: Dette afsnit skal indeholde planer for systematiske white box tests til udvalgte metoder. Der skal vælges 4 metoder for firmandsgrupper. Prøve at vælge metoder som gør noget. Der skal ikke vælges getter og setter metoder. Vis metoderne i rapporten. Benyt samme skabelon til systematiske tests som er brugt i forbindelse med forelæsninger (dvs. to tabeller; den ene med input egenskaber og den anden med konkrete værdier). De planlagte systematiske test skal implementeres og udføres ved brug af JUnit eller Cucumber og skal afleveres som en del af Eclipse/IntelliJ projektet.

2. Ydeligere skal rapport 2 have et diagram som viser code coverage af de enkelte Java filer efter alle testene er kørt, sammen med en forklaring af code coverages resultater.

3. *Design by contract*: Dette afsnit skal indeholde kontrakterne til de 4 metoder I har valgt i afsnit om systematiske white box tests. Kontrakterne, dvs. pre– og post–betingelserne af de metoder, skal vises som matematiske formler. Desuden skal der tilføjes assert statements, som implementerer pre– og post–betingelserne i jeres kode, og som er også beskrevet og forklaret i rapporten. Der kan være en eller flere assert statements til pre– og post–betingelserne af metoderne. Hvis der er ingen særligt pre–condition så skrives `assert true;`.

4. Design mønstre: I dette afsnit skal der redegøres af de brugte design mønstre i jeres design/kode. Hvilke design mønstre har I brugt og hvorfor? Hvis I har ikke brugt nogen design mønstre så skal der skrives hvorfor I synes det giver ikke mening at bruge design mønstre. Bemærk, der er ingen krav til at bruge design mønstre i jeres design og kode. Design mønstre er mulige løsninger til nogle særlige design problemer. Hvis I har igen af disse design problemer, eller der findes bedre løsninger, så skal der heller ikke bruges design mønstre.

5. *Konklusion*: Der skal gives en kort status og vurdering af det afleverede produkt. Desuden skal der være et kort afsnit, der reflekterer omkring projektets forløb, og sammenligner det aktuelle programdesign efter implementeringen med programdesign fra rapport 1.

Programmets applikationslag skal udvikles test-dreven. Til hver funktion i applikationslag skal der findes en eller fler svarende Cucumber eller JUnit tests som sikrer at funktionen gøre det den skal (ligesom vi har gjort i øvelsenser.) Brugergrænsefladen behøves ikke udvikles test-dreven. Ved projekt demonstration skal I vise testene.

Der skal bruges Git som versions kontrol system. Og ved afleveringen skal der vedlægges URL til read-only adgang til Git repository i rapport 2. Git har muligheden for at tilknytte fler remote repsositories til jeres projekt. Det nemmeste er at I har også et remote repository hos GBar (`https://gitlab.gbar.dtu.dk`) hvor I giver mig adgang til repository. Være opmærksom på at jeg skal have ret til at se indholdet af jeres repository.

Ved bedømmelse af rapporterne vil der blive lagt vægt på

- fuldstændighed, dvs. at de valgte funktioner bliver komplet behandlet, og
- sporbarhed, herunder konsistens igennem alle abstraktionsniveauer.

Det er bedre at sikre disse kvalitetsegenskaber igennem en bevidst afgrænsning af problemet, end at brede sig for meget ud over problemstillingen.

Slidserne i uge 6 anbefaler en process som kan bruges til at implementere systemet ud fra programdesign i rapport 1.

5 Projekt Beskrivelsen: Timeregistrering og projektstyring

Softwarehuset A/S bruger i dag et ældre system til timeregistrering og projektstyring. Systemet er på flere måder tungt at arbejde med. Specielt klager mange medarbejdere over besværet med at registrere hvor mange timer man har arbejdet på forskellige aktiviteter. Projektlederne klager over at det er svært at bemande et projekt fordi man ikke kan se hvem der har tid hvornår.

Softwarehuset A/S har ca. 50 udviklingsmedarbejdere og 30 igangværende projekter. Mange projekter varer blot et par måneder, men nogle varer flere år. Hvert projekt har

en projektleder (valgt blandt udviklingsmedarbejderne). En af projektlederes opgaver er at opdele projektet i aktiviteter, f.eks. kravspecifikation, projektledelse, analyse, design, programmering, test med mere.

Et projekt har typisk 30 aktiviteter, men nogle har næsten 100. Et projekt oprettes, når der skal arbejdes på en opgave. Det kan være en mulig kunde, som ønsker en udvikling, eller det kan være et "internt"projekt, der går ud på at lave noget for Softwarehuset A/S selv. På oprettelsestidspunktet er information om aktiviteter og starttidspunkt ikke fuldstændige, men dog kendt i et vist omfang. Projektlederen er ikke nødvendigvis udpeget fra starten.

Projektlederen skal gradvis opdele projektet i aktiviteter. Aktiviteterne skal kunne planlægges længe inden arbejdet på dem begynder, men planerne skal kunne ændres hyppigt. For hver aktivitet skal projektlederen kunne anføre forventet antal arbejdstimer til løsning af opgaven (budgetteret tid). Endvidere skal der kunne anføres start- og sluttid på aktiviteten. Der ønskes en opløsning på ugenummer niveau. Projektlederne skal kunne bruge systemet til at bemande projekter i god tid. Der kan være behov for, at mere end en medarbejder deltager i at udføre en bestemt aktivitet. Systemet skal tillade, at en medarbejder kan søge assistance hos en kollega og at denne så kan registrere den forbrugte tid på aktiviteten, selv han ikke direkte er tilordnet den af projektlederen. Softwarehuset A/S ønsker et system der kan hjælpe projektlederne i forbindelse med bemanding af projekter. Det skal således give overblik over, hvilke medarbejdere, der burde være ledige på det tidspunkt, hvor aktiviteten skal udføres.

En medarbejder arbejder typisk på mindre end 10 aktiviteter ad gangen og varigheden af den enkelte aktivitet er typisk 2-3 uger. Systemet skal tillade enkelte medarbejdere at arbejde på 20 aktiviteter i løbet af en uge. Der skal være faste aktiviteter for registrering af ting som ferie, sygdom, kurser med videre, som ikke kan pålignes det enkelte projekt.

Medarbejderne skal dagligt registrere hvor meget tid de har brugt på de forskellige aktiviteter (1/2 times nøjagtighed er tilfredsstillende). Det skal være enkelt at foretage denne registrering, så medarbejderen ikke føler det er en byrde. Medarbejderen skal enkelt kunne se, om han/hun har registreret alle timer man har arbejdet i dag. Det skal være muligt at rette i allerede registrerede data ligesom det skal være muligt at registrere fremtidige aktiviteter, f.eks. ferie.

Projektlederen skal kunne bruge systemet til opfølgning. Det betyder, at projektlederen let skal kunne se, hvordan timeforbruget udvikler sig per aktivitet og for hele projektet. Der skal kunne laves rapporter til brug ved Softwarehuset A/s' ugentlige projektmøder. Ved disse møder skal projektlederen endvidere se det forventede restarbejde på projektet.

5.1 Lidt data

Medarbejdere identificeres med initialer på indtil 4 bogstaver. Projektnumre skal tildeles af systemet og have formen årstal og et løbenummer, f.eks. 030901. Desuden skal projektet kunne have et navn.

Systemet er til internt brug, så det er ikke nødvendigt med adgangskontroller.

5.2 Afgrænsning

Kunden prioriterer projekt planlægning før registrering af arbejdstidsforbrug. Til sidst kommer rapporter til brug ved projektmøder.

Kunden ønsker at det er muligt at oprette et projekt og tilføje en medarbejder til dette projekt. Desuden skal det være muligt at tilknytte en projektleder til projektet og opretter en aktivitet. Medarbejderen skal være i stand til at registrere tid til den aktivitet.

Angående programmet, så er det ikke et krav, at data kan gemmes på en fil eller en database. Systemet skal have en brugergrænseflade, men det er heller ikke et krav, at systemet kan benyttes via en grafisk brugergrænseflade.

Med tak til Michael R. Hansen og Søren Lauesen for lån af ideen til system og for store dele af beskrivelsen.

6 Generelle Regler

- Opgaven skal løses i grupper på 4 studerende.
- Rapporterne skal skrives på dansk eller engelsk og afleveres som PDF fil ved brug af opgave modul på CampusNet. Rapporterne skal hedde `rapport_xx_1.pdf` og `rapport_xx_2.pdf` hvor `xx` er jeres gruppe nummer skrevet med 2 cifre, dvs. 01, 02, 03, ..., 10, 11, ... Være opmærksom på at det skal være `rapport_xx_1.pdf` og ikke `Rapport_xx_1.pdf` eller `report_xx_1.pdf` osv. Navne er meget vigtig fordi det hjælper mig at se med det samme om jeg har alle rapporter og Eclipse/IntelliJ projekter.
- Program skal afleveres på CampusNet som Eclipse/IntelliJ projekt som indeholder kildekoden, testene, det kørende program og en instruktion hvordan man kører programmet og testene. Projektet skal være i en ZIP fil som hvis den er et Eclipse projekt kan importeres i Eclipse. Hvis den er et IntelliJ projekt, så skal ZIP filen indholder alle filerne for at åbne og køre projektet (og testene) i IntelliJ. Det er en god ide at tjekke om jeres ZIP fil kan importeres i Eclipse eller åbnes i IntelliJ (efter udpakning) og et programm kører i Eclipse eller IntelliJ inden aflevering. Navnet skal være `projekt_xx.zip` hvor `xx` igen er jeres grupper nummer skrevet med 2 cifre. **Program må ikke være afhængig af nogle særlige pakker eller jar-filer som er ikke nævnet i kurset. Hvis I har behov for at bruge særlige pakker eller jar-filer, så skal I spørge mig først.**
- Hver rapport skal have en forside med titel, fx. "Rapport 1", gruppe nummer, dato og studienummer og navn for alle gruppens medlemmer.
- Hvert afsnit eller use case, diagram o.s.v. skal have navnet på fra personen, som har lavet afsnittet, use case, diagrammet o.s.v. Der kan være kun et navn på hvert afsnit, use case, diagram, o.s.v. Sikre jer at hver medlem af gruppen har bidraget lige til alle rapporter og dele i rapporterne.

- Hver klasse og method skal have navnet på fra personen, som har skrevet klasse/metode. Der kan være kun et navn på hvert klasse/metode. Sikre jer at hver medlem af gruppen har bidraget lige til kildekoden.

- *Diskussion af den obligatoriske opgave er tilladt også med andre gruppers medlemmer og kursusdeltager fra tidligere år, men ingen form for anvendelse af andres program eller rapportdele er tilladt. Hver besvarelse skal være selvstændig.* **Afskrift uden kildeangivelse betragtes som snyd.**

- **Mandag den 11 maj** giver hver gruppe en kort demo (10 min) af deres program i E-databaren til en hjælpelærer. Demoen skal (som minimum) vise testresultaterne fra afsnit 4 i rapporten.

- Tidsfristerne til de enkelte delopgaver findes i opgave modul på CampusNet. Til hver delopgave findes der en egen opgave på CampusNet.

- Yderligere informationer vil blive oplyst via kursets hjemmeside.

7 Evalueringskriterier

Karakteren i kurset gives som helheds vurdering af rapport 1, feedback til andres rapport 1, rapport 2 og kildekoden (inkl. test). Fokus er på den korrekte brug af teknikerne der bliver vist i kurset, fx. brug af den korrekte notation ved use case–, sekvens– og klassediagrammer, og om teknikerne er forstået eller ej.

I programkoden bliver der lagt vagt på en god arkitektur, og at kildekoden er velstruktureret og let at forstår og læse, fx. ved brug af korte metoder, de retter abstraktioner osv. Det er også vigtig at jeg kan se i rapporten og kildekoden at hver projektmedlemmer har brugt alle teknikerne og har bidraget til koden.

Det drejer sig ikke om hvor meget af den funktionalitet i problem beskrivelsen er implementeret, men om kvaliteten af implementeringen og rapporterne. Det er også vigtig at I er i stand til at afgrænse problemet.