Danmarks Tekniske Universitet

Eksaminations Projekt

Project Planner

Gruppemedlemmer:
Carsten Michael Rosenkilde Nielsen s123062
Christian Mathias Rohde Kiær s123812
Jonathan Binner Becktor s123094

13. maj 2013

Indledning

Til opgave har vi fået stillet, at designe en planlægningsapplikation. Denne applikation skal skabes for et softwarehus, der mangler et system til at planlægge deres projekter. Projekter skal kunne oprettes efter de informationer, der haves på oprettelses tidspunktet. Dette skal kunne inkludere navn, start/sluttidspunkt og projektleder. Alle disse oplysninger skal også kunne tilføjes eller ændres senere hen. Projekter skal kunne tildeles aktiviteter, som tilføjes med et navn, en beskrivelse,en start/slut dato, antal allokerede timer og det aktuelle timeforbrug. Udviklere skal kunne tildeles aktiviteter, og derfra registrere deres timeforbrug på den pågældende aktivitet. Til løsning af problemerne, har vi opstillet 6 Use Cases som vi vil implementere testdrevent. Følgende Use Cases har vi valgt at kigge på:

- Create projects
- Manage projects
- Define Activities
- Manage developers
- Hours worked on project
- Seek Assistance

Vi har implementeret alle Use Cases test drevent, og det er gjort i forbindelse og uden konflikt med opgavebeskrivelsen.

Indhold

1 Kravspecifikation

1.1 Væsentlige begreber

Under udviklingen har vi oprettet føglende liste af vigtige begreber:

- User: skal forstås som brugeren eller udviklerne
- Project: Projektet eller opgaven der er stillet softwarehuset. Projekter kan være tildelt navn og projektleder. Projekter kan også have deadlines.
- Activity: Aktiviteter er opgaverne tilknyttet projekterne. Udviklere skal kunne være tilknyttet en aktivitet og registrere deres arbejdstid. Aktiviteter kan være tildelt navn, beskrivelse og deadlines.
- Projectleader: Projektlederen er den ansvarlige udvikler for et projekt.
- Available developers: Det skal være muligt for både projektleder og udviklere, at finde udviklere der er tilgængelige til enten hjælp eller tildeling af aktivitet.
- Administrator: Superbrugeren af systemet. Administrator skal kunne registrere brugere og oprette projekter.
- Work: Udviklerne skal have en let måde at registrere det antal timer de har arbejdet på et givent projekt. Gennem work kan de nemt registrere dette, samt hvilken aktivitet de var tilknyttet.

Disse begreber er væsentlige for forståelsen af udviklingen af programmet, og de tanker der er gjort bag. Dette er de begreber, vi har defineret ud fra opgavebeskrivelsen og de use cases der er blevet opstillet.

1.2 Use cases

Alle vores Use cases ligger som bilag. Se bilaget for at finde de pågælende use cases.

Use case 1

Vores første use case, beskriver hvordan projekter skal oprettes. Se bilag Use case 1.

Use case 2

Vores anden use case, beskæftiger sig med modifikationen af projekter. Se bilag Use case 2.

Use case 3

Se bilag Use case 3.

Use case 4

Se bilag Use case 4.

Use case 5

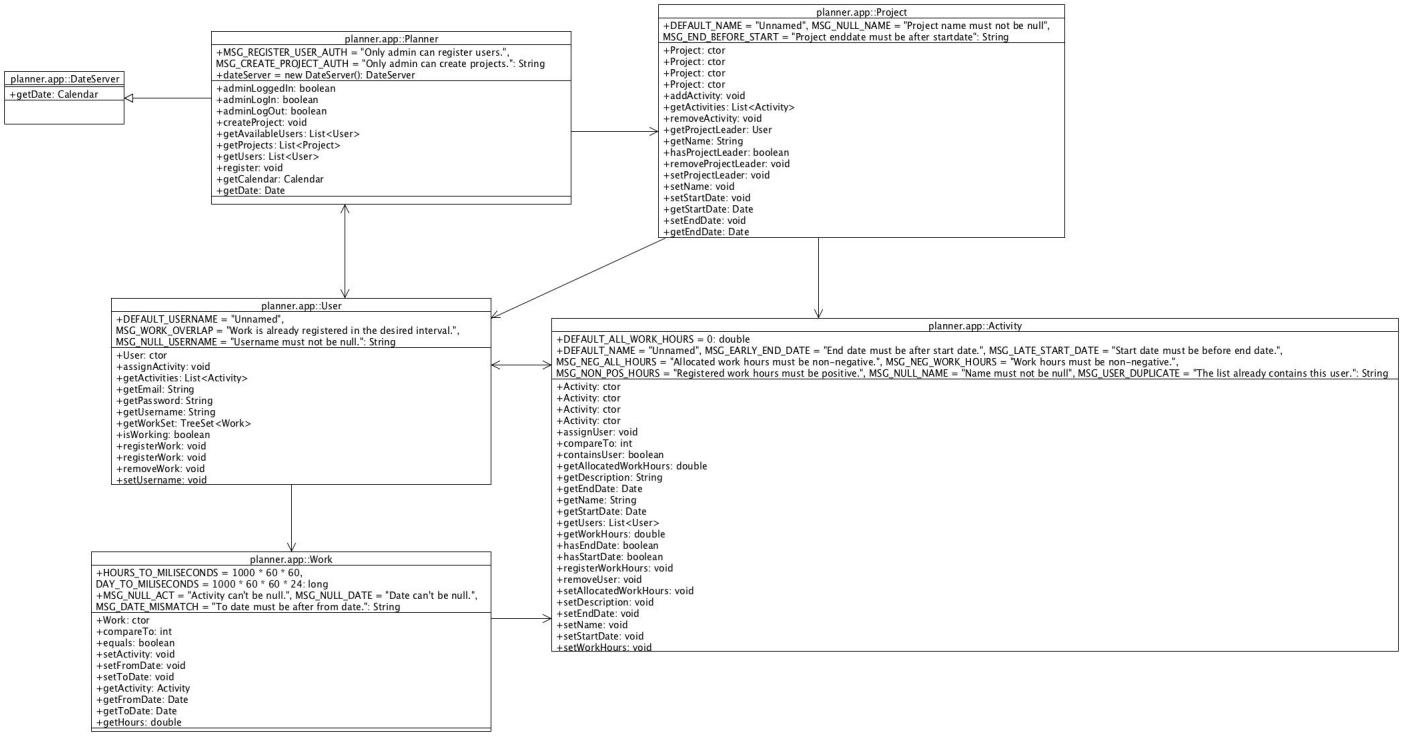
Se bilag Use case 5.

Use case 6

Se bilag Use case 6.

2 Programdesign

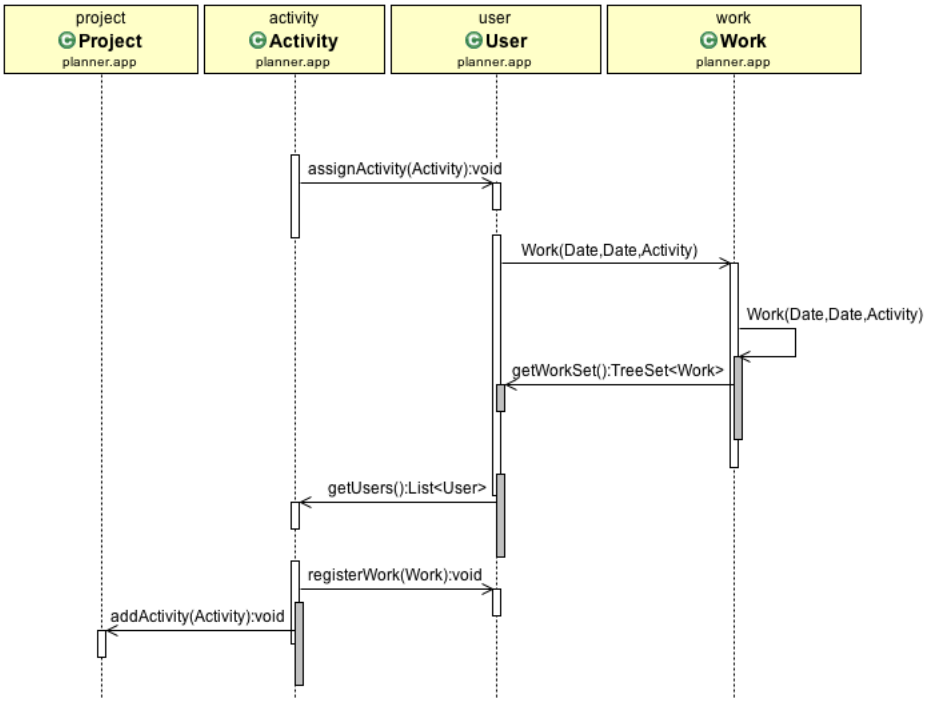
2.1 Klasser



Figur 1: Klassediagram

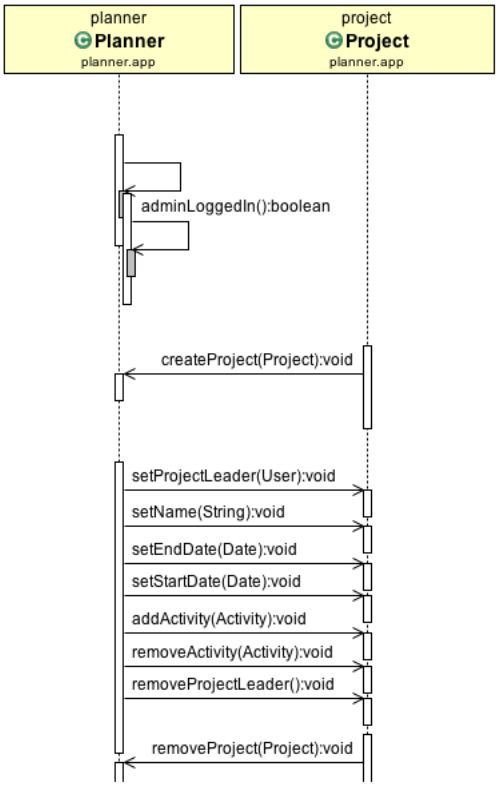
Ovenfor er vores klassediagram opstillet. Vi har valgt kun at vise de public metoder, for alle vores klasser. Derudover viser diagrammerne hvilke funktioner, der kan hente oplysninger fra hvad.

2.2 Sekvensdiagrammer



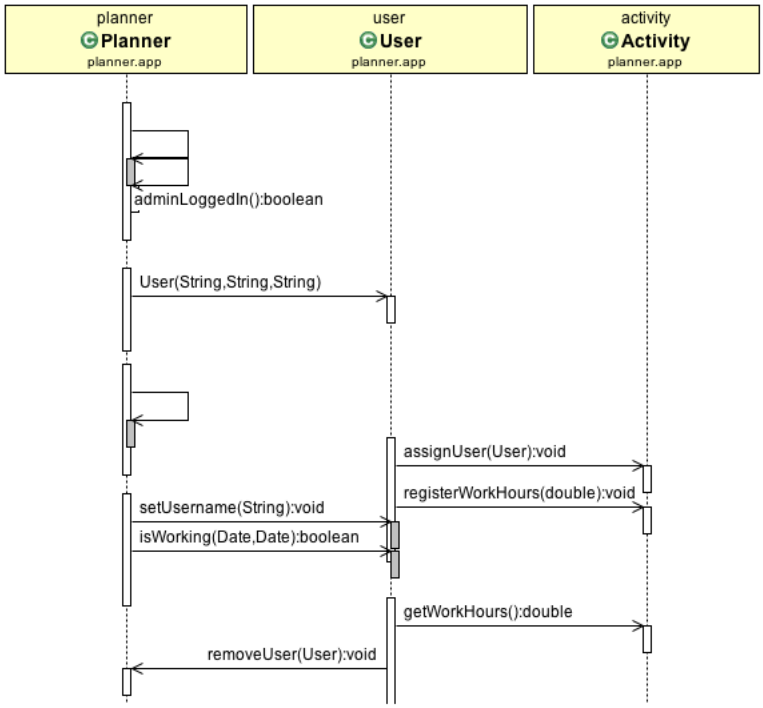
Figur 2: Workhours

asdasdasd asdasdasd
asdads
asdasd
asdasdasd asdasdasd
asdads
asdasd



Figur 3: Project creation and modification

asdasdasd asdasdasd
asdads
asdasd
asdasdasd asdasdasd
asdads
asdasdasdasdasd asdasdasd
asdads
asdasdasdasdasd asdasdasd
asdads
asdasdasdasdasd asdasdasd
asdads
asdasdasdasdasd asdasdasd
asdads



Figur 4: User creation and modification

asdasdasdasdasd asdasdasd
asdads
asdasd

2.3 Diskussion

Vi som gruppe diskuterede vi hvad der skulle lægges fokus på. Til at starte med var vi enige om at der skulle være GUI, persistency og god funktionalitet. Dog da vi var et godt stykke inde valgte vi at fokusere på programmets funktionalitet og de tilhørende tests for at få så god "code coverage"som muligt. Da alle vores use cases kun beskæftiger sig med funktionalitet af programmet, kunne vi derfor opnå at færdiggøre dem alle. Dette medfører også, at vores use cases har tests for de fleste scenarioer.
Oplægget til opgaven skaber en del uklarheder rundt omkring, der dog let kan ændres på. Specielt de forskellige brugers rettigheder i programmet, var vi i tvivl om. Derfor er programmet meget åbent, og kun nogle få metoder kræver brugerrettigheder.

3 Systematiske test

Alle vores tests er i bund og grund lavet for at kunne få så høj "code coverage"som muligt. Med dette som bagrund har vi lavet tests til mere eller mindre alt i programmet. Vi har taget brug af EclEmma for at finde hvor vi manglede test og hvad der ikke var covered. Nogle af de oftest funde test er tests for getters & setters hvilket var forholdsvist let at få testet man laver en værdi som sættes i en setter hvorefter vi bruger assertEquals for at se om den værdi vi har sat i setteren er den samme vi får fra getteren når vi spørger efter den. En anden test der ofte skulle testes var hvis fx. en setter ikke kunne være null og hvis den var skulle kaste en fejl. Her bruger vi try/catch i testsne for at fange de fejl som programmet giver os og sammenligner dem med de ønskede fejlmeddelelser.

Hvis vi kigger på usecase1 først tester vi om admin er logget ind. Dette gør vi ved at have en boolean value som sættes til true når det rigtige password er blevet skrevet adminLogIn() metoden. Derefter kan administratoren lave projekter. Hvis administratoren laver et projekt uden navn vil programmet tildele den et. Dette testes ved at lave et program uden navn og derefter teste ved getName metoden om navnet er blevet til "default name". Denne slags test bliver brugt flere gange i denne usecase hvis der ikke skrives start dato bliver den af default sat til dags dato. dette tjækkes på samme måde. Hvis projektet bliver lavet med end date før start date kaster programmet en fejl. Denne fanger vi med en try/catch som tjekker om fejlen og fejlmeddelelsen er det samme.

4 Konklusion

Vi har fået skabt et program, der lever op til de kravspecifikationer vi har fået stillet. Programmet er gennemtestet med mange scenarioer, og derfor har vi prøvet at sikre en høj kvalitet. Der er plads til udvidelse, hvis dette skulle være at ønske. Af egenskaber vi ikke har nået, som dog ikke er en del af kravspecifikationerne, kan nævnes et persistency layer og et UI. Dette er selvfølgelig påkrævet for et fuldt funktionelt program. Vi har dog opnået at få skabt det grundlæggende, for en planlægningsapplikation.
Projektet har forløbet fint. Dog har der grundet andre kurser været funktioner vi har måtte udelade. Derfor har vores fokus været på de grundlæggende funktioner, og ting funktioner vi havde tiltænkt, som GUI og persistency, blev udeladt. Dette er dog noget vi i sidste ende har været glade for, da det har givet os bedre tid til at løse vores use cases. I forhold til vores oprindelige projektplan, kom vi forsent igang allerede fra uge 1. Dette medførte de fornævnte fravalg. Dette var dog noget vi blev enige om forholdsvist hurtigt. Ellers vil vi mene at vi har forsøgt at følge planen, og har prøvet at have så få overskridelser af tiden som muligt.
Alt i alt har dette givet os en mulighed, for at prøve at arbejde mod en deadline. Vi har fået indblik i hvilke valg der skal tages, og har indset at ikke alle de ideer man har gjort sig nødvændigvis kan nås.

5 Bilag

5.1 Use case 1

Use Case 1	**Create Project**
Scope:	System-wide
Primary Actor:	Administrator
Stakeholders and Interests:	• Administrator: Needs to be able to create and modify projects • Project Leader: Needs to modify projects and add activities to projects • Developer: Needs to be able to see projects
Description:	Makes it possible to create a project

Main Scenario:

1. Gives the administrator the possibility to create projects
2. The administrator will be able to create with a given name and projectleader.
3. The administrator can assign a start and end date to the project

Alternative Scenarios:

1.a Invalid login data:

1. System shows failure message
2. System returns to step 1

2.a No projectleader assigned:

1. If projectleader is not assigned, it will create the proejct with no projectleader.
2. Administrator will be able to set the projectleader by modifying the project

2.b No projectname assigned:

1. If projectname is not assigned, a default name will be assigned
2. Administrator or proejctleader will be able to change the name, by modifying the project.

3.a If start date not assigned:

1. If start are not assigned the project will be created with the current date as start date.

3.b If end date not assigned:

1. If end date is not assigned, the project will be created with no value as end date.

3.c End date is before start date

1 The system automaticly changes the end date to be after the start date.

5.2 Use case 2

Use Case 2	**Manage Project**
Scope:	System-wide
Primary Actor:	Project Leader
Administrator, Project Leader:	• Administrator: Assign project leader. • Project Leader: Assign activities to project and manage them.
Description:	Makes it possible to assign a project leader, change project leader and assign and manage activities to project

Main Success Scenario:

1. Admin assigns project leader.
2. Project leader assigns activities to project.
3. Project now managed.

Alternative Scenarioes:

1.a Admin assigns wrong project leader:

1. Admin changes project leader.
2. Project Leader continues on step 2.

1.b Admin assigns invalid project leader:

1. System shows failure message
2. Admin assigns correct project leader

2.a Wrong activity data:

1. Project leader corrects activity data
2. Project leader continues with step 3

2.b Invalid activity data:

1. System shows failure message
2. Project leader returns to step 2 and corrects the errors

2.c Wrong activity:

1. Project leader removes activity
2. Project leader returns to step 2

5.3 Use case 3

Use Case 3	**Define activities**
Scope:	System-wide
Primary Actor:	Project leader and Developer
Stakeholders and Interests:	• Project Leader: Needs add activities to projects and modify them. • Developer: Needs add activities to projects and modify them. Also needs to register hours worked on given activity.
Description:	Makes it possible to create an activity connected to a project, add developers to the activity and assign workhours to the activity

Main Scenario:

1. Lets the users create activities.
2. The users can create activities, defined by name, description and allocated workhours.
3. Lets the user connect developers to an activity.
4. Lets the users define the expected start and end dates of the activity. It's possible to assign an endDate without startDate and vice versa.
5. Lets the developers define the actual hours of work put into the activity. It's possible to set the final workhours and add work hours while working.

Alternative Scenarios:

2.a No name assigned:

1. The activity will be assigned a default name.
2. The activitys name can be modified at a later time

2.b Description not assigned:

1. The activity will have no description.
2. The description of the activity can be set at a later time.

3.a Adding same user twice

1. System throws an error
2. System returns to 3.

4.a If start date not assigned:

1. If start date is not assigned the activity will have no start date.
2. The start date can be modified later.

4.b If end date not assigned:

1. If end date is not assigned, the activity will have no end date.
2. The end date can be modified later.

4.c End date is before start date

1. The system will show an error.
2. The system will go back to 4.

5.a Define negative hours of work

1. System returns an error
2. System returns user to 5.

5.4 Use case 4

Use Case 4	**Manage Developers**
Scope:	System-wide
Primary Actor:	Admin
Administrator:	• Administrator: Create Developers and manage them.
Description:	Makes it so admin can manage all information in developers, add and remove new developers.

Main Success Scenario:

1. Admin creates developer
2. Admin changes developer username
3. Admin changes developer email
4. Admin changes developer password
5. Developer now managed

Alternative Scenarioes:

1.a Admin creates wrong developer

1. Admin removes developer
3. Admin returns to step 1.

2.a Admin sets null username:

1. System shows failure message
2. Admin changes developer username properly
3. Admin continues with step 2

3.a Admin sets null email:

1. System shows failure message
2. Admin changes developer email properly
3. Admin continues with step 3

4.a Admin sets null password:

1. System shows failure message
2. Admin changes developer password properly
3. Admin continues with step 4

5.5 Use case 5

Use Case 5	**Hours worked on project**
Scope:	System-wide
Primary Actor:	Developer
Stakeholders and Interests:	• Developer: The developers needs to be able to set themselves as working, and register the hours they are working
Description:	Makes it possible for a developer to get set himself as working, and register the ammount of hours he is working

Main Scenario:

1. User finds activity
2. User enters hours spent on activity
3. Activity registers hours Developer has spent.

Alternative Scenarios:

1.a User set to work:

1. Sets himself to working

1.b Non existent activity:

1. System casts errormessage.
2. Returns to step 1.

1.c null activity:

1. System casts errormessage.
2. Returns to step 1.

2.a allocated hours more than actual time on project:

1. System casts errormessage.
2. returns to step 2.

2.b negative work hours:

1. System casts errormessage.
2. returns to step 2.

3.a allocated hours more than actual time on project:

1. System casts errormessage.
2. returns to step 2.

5.6 Use case 6

Use Case 6	**Seek assistence**
Scope:	System-wide
Primary Actor:	Developer
Stakeholders and Interests:	• Developer: Needs to be able to see which developers are available, at the given time.
Description:	Makes it possible for developers to see which developers is not currently working on an activity

Main Scenario:

1. Lets the user get a list of developers.
2. The user can see which developers is not working on an activity.

Alternative Scenarios:

2.a Requires work to not overlap:

1. If work overlaps, system cast error.
2. Returns to 1.

5.7 Original tidsplan

Figur 5: Oprindeligt planlægningsark

Numbers of persons	Hours a week per person	hours a week	no. of weeks	hours for whole project				Christian Mathias Rohde Kiær, s123812
3	8	24	5	120				Jonathan Binner Becktor, s123094
								Carsten Nielsen s123062
	Week	User Story/Tasks	Ideal man hour	Man hour with load factor	total hours in week	total hours		Use cases:
	week 1	Base structure of report	2	3	3	3		Create projects
		user register	2	3	6	6		Manage projects
		user group	2	3	9	9		Define activities
		Create project	2	3	12	12		Manage developers
		Create activities from projects	2	3	15	15		Hours worked on project
		Assign developers to activities	2	3	18	18		Seek assistance
		Assign project leader	2	3	21	21		
		Syst. test for the use case in this iteration	2	3	24	24		
	week 2	Writing introduction and design	2	3	3	27		
		Login and logoff	2	3	6	30		
		Create hour logger	2	3	9	33		
		Assign projects activities	2	3	12	36		
		Assign projects estimated time	2	3	15	39		
		Developer availability check	2	3	18	42		
		Report	2	3	21	45		
		Syst. test for the use case in this iteration	2	3	24	48		
	week 3	Assistance from developers outside project	2	3	3	51		
		Developer time management	2	3	6	54		
		Read data from file	2	3	9	57		
		Output data to file	2	3	12	60		
		Database management	2	3	15	63		
		Midways review	2	3	18	66		
		Report	2	3	21	69		
		Syst. test for the use case in this iteration	2	3	24	72		
	week 4	GUI	8	12	12	84		
		Weekly workhours display	2	3	15	87		
		Project, activity and user display	2	3	18	90		
		Report	2	3	21	93		
		Syst. test for the use case in this iteration	2	3	24	96		
	week5	Bugfixes	12	12	12	108		
		Report	12	12	24	120		