

IT-DIPLOMUDDANNELSEN

OBJEKTORIENTERED METODER

EKSAMENSPROJEKT
AF
JAN SCHRØDER HANSEN

FORÅR 2011

INDHOLD

1.	Indledning.....	3
2.	Opgavebeskrivelse/Krav.....	3
3.	Analyse	4
4.	Design.....	8
4.1.	Webside design	11
5.	Idriftsættelse	11
6.	UP Iterationer	12
7.	Konklusion	13
8.	Bilag	14
8.1.	Krav	14
8.2.	Danske/engelske termer	17
8.3.	Administrative use cases.....	18
8.4.	Use case beskrivelser.....	19
8.5.	Krav/use case matrix.....	30
8.6.	Analyse pakkediagram.....	31
8.7.	Solskins scenarie aktivitetsdiagram	32
8.8.	State Machine diagram for en sag	33
8.9.	Design use case realisering.....	34
8.10.	Webside design	36
8.11.	Indhold på vedlagte CD	42

1. INDLEDNING

Dette eksamensprojekt er lavet i forbindelse med faget ”Objektorienterede metoder” på IT-Diplomuddannelsen, Ingeniørhøjskolen i København.

Faget har taget udgangspunkt i bogen ”UML 2 and the Unified Process” med undertitlen ”Practical Object-Oriented Analysis and Design” af Jim Arlow og Ila Neustadt. Samt materiale udarbejdet af fagets underviser Jacob Nordfalk.

2. OPGAVEBESKRIVELSE / KRAV

For at komme igennem så meget af materialet i faget som muligt, har jeg valgt at lave en webløsning, til håndtering af ønsker og fejlreporter, til et eller flere softwareprodukter.

Der skal være mulighed for at følge status på fejl og ønsker. Er fejlen eller ønsket godkendt, afvist, er det under udvikling, under test etc. Derudover skal der være mulighed for at vedhæfte forskellige filer såsom skærmdumps af fejl, eller prototyper på ny skærmlayouts og andre filer, som kan hjælpe til at belyse en sag.

Programmets navn er BaRI, som står for ”Bugs and Request Interceptor”.

Det er en løsning, jeg startede på under faget ”Web og Serverprogrammering” efteråret 2009 og som jeg videreudviklede i faget ”Databasesystemer og Web” efteråret 2010. Begge rapporter kan findes på den vedlagte cd.

Under faget ”Web og Serverprogrammering” var mit fokus på et web framework kaldet Wicket¹. En lille del af opgaven skulle også demonstrere, at løsningen kunne persistere data i en database, så der blev lavet to tabeller. Under faget ”Databasesystemer og Web” var mit fokus på database delen, hvor antallet af tabeller kom op på 11. Begge løsninger kan godt betragtes som prototyper.

I denne opgave vil jeg starte forfra med krav, analyse og design af ovenstående system. Selve UP processen vil ikke fremgå af denne rapport, da det ikke svært at beskrive denne proces i rapportform. Så rapporten vil følge den gamle vandfaldsmodel, med krav, analyse, design, etc.

Funktionelle og ikke funktionelle krav til ovenstående opgavebeskrivelse, kan ses under bilag afsnit 8.1. Disse krav skal ses som kravspecifikationer, sammen med ovenstående opgavebeskrivelse.

Da jeg altid skriver på engelsk i min kode, har jeg valgt at mine UML diagrammer også er på engelsk. Men da rapporten her er på dansk, har jeg vedlagt en dansk/engelsk ordliste under bilag. Se afsnit 8.2. Dette gælder dog ikke for use case diagrammer og use cases. Som jo er det UML værktøj, som kan bruges over for mennesker, som ikke arbejder med it udvikling til dagligt.

Alle diagrammer er udarbejdet vha. af programmer MagicDraw².

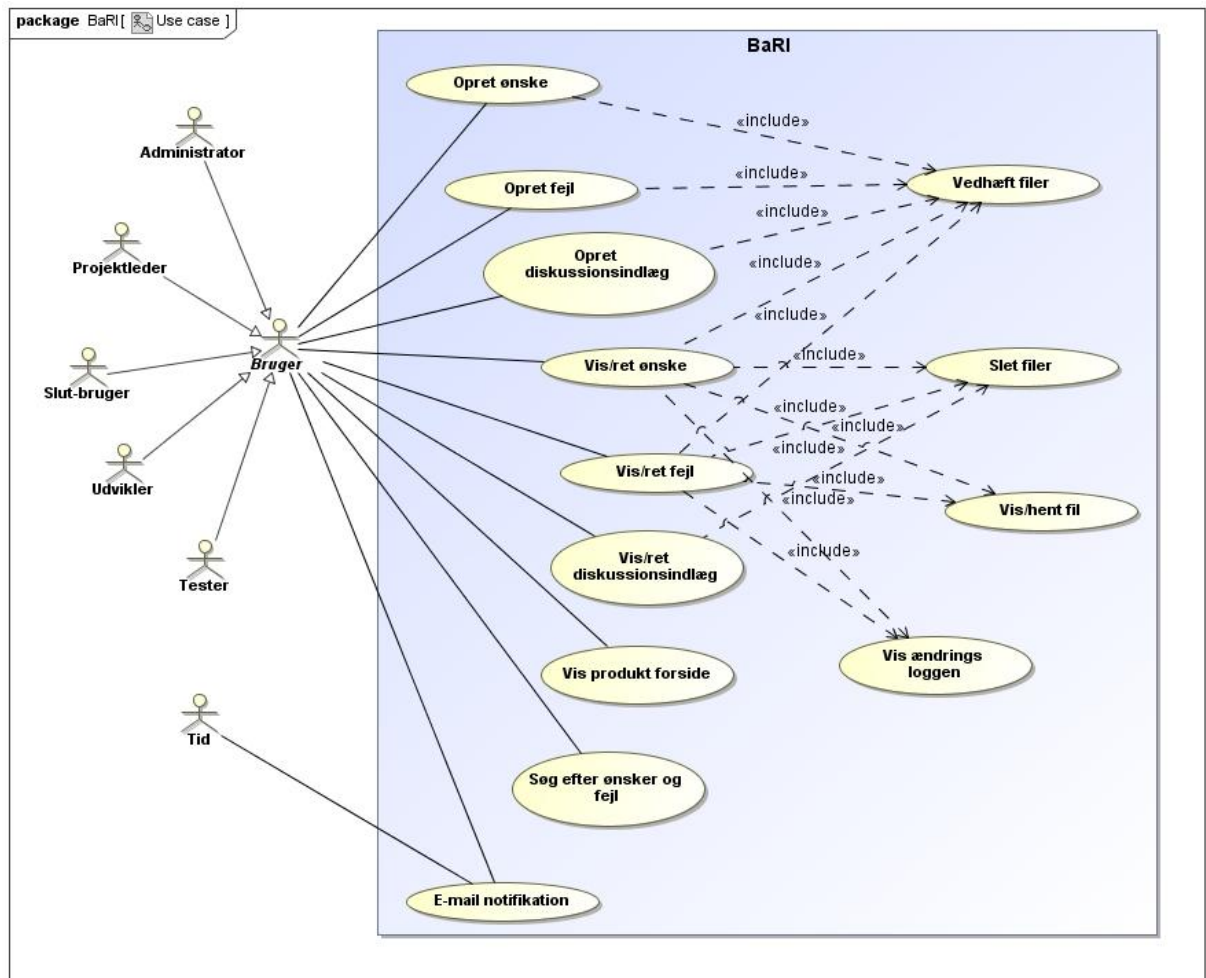
I de følgende afsnit gennemgås de forskellige udviklingsfaser, startende med analysen.

¹ Wicket – se <http://wicket.apache.org/>

² MagicDraw – se www.magicdraw.com

3. ANALYSE

Jeg har valgt at lave to use case diagrammer, selvom der kun er et system. Diagrammerne er opdelt efter administrative use cases og use cases som fortæller hvad systemets primære opgave bliver. De administrative use cases kan ses under bilag. Disse use cases vil jeg ikke komme ind på, da de er ret trivielle CRUD³ use cases, som vedligeholder basisdata i systemet, som bruger og produkt.



Figur 1- BaRI Use Cases

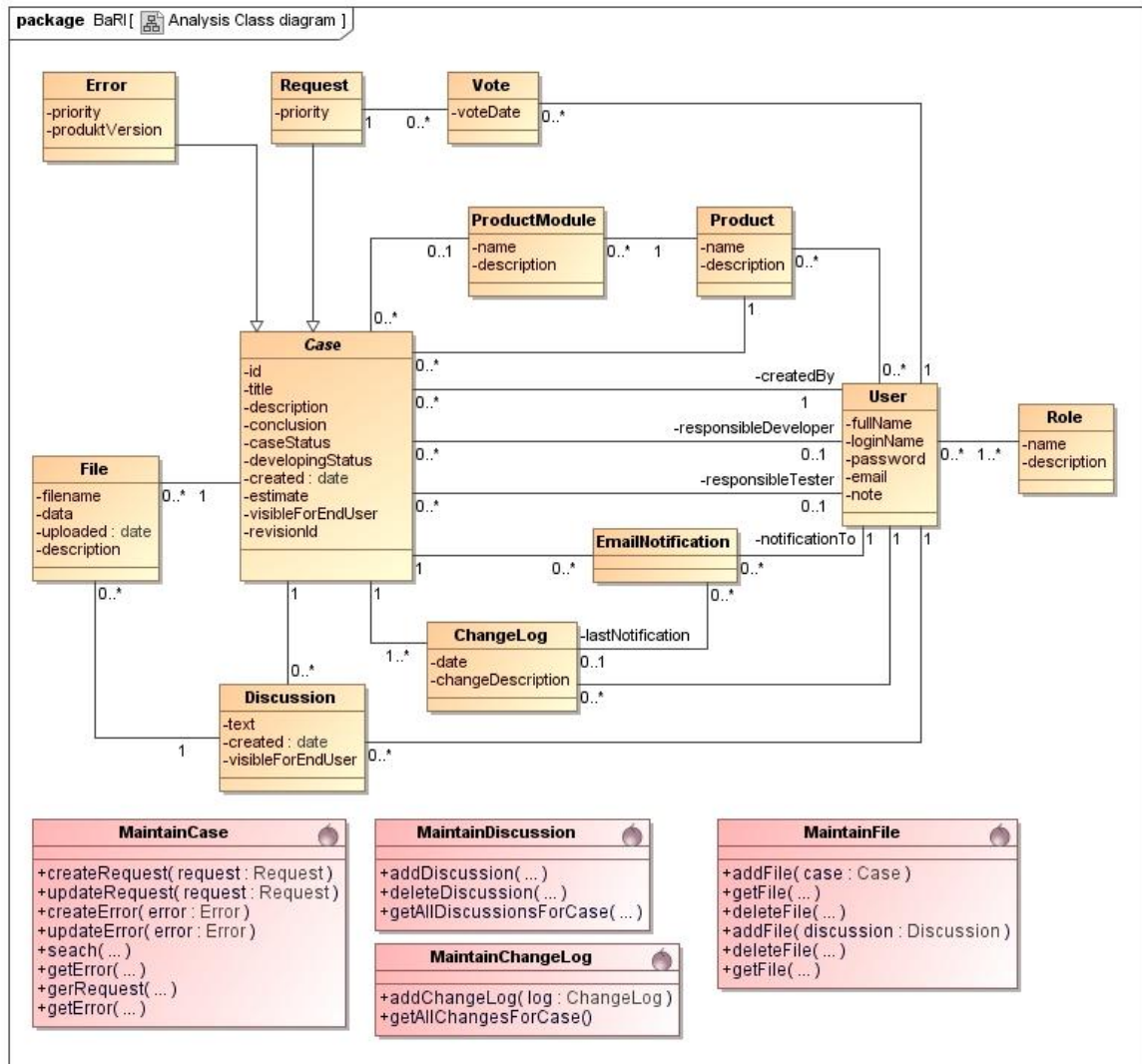
Som det fremgår af overstående, så er alle brugertyper, som også fremgår af kravene med på diagrammet. Der er fordi de enkelte brugerroller er vigtige for valgmulighederne i flere af ovenstående use cases. En Use case som login til systemet og logud, har jeg valgt helt at se bort fra, da jeg ikke mener de giver værdi i analysefasen. Det samme gælder for en administrators mulighed for at slette ønsker og fejl. Til slut har jeg lavet en krav/use case matrix, for at sikre at alle funktionelle krav er dækket ind af overstående use cases. Se afsnit 8.5.

³ CRUD – Create, Read, Update og Delete operationer.

Herunder er use casen ”Opret ønske” beskrevet. Alle use case beskrivelser, kan ses under bilag. Se bilag afsnit 8.4.

Use case: Opret ønske
ID: 1
Kort beskrivelse: En bruger opretter et nyt ønske.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger ”Opret nyt ønske” linket. 2. Systemet viser en tom side, med følgende felter: <ol style="list-style-type: none"> 2.1. Produkt skal vælges, og hvis tilgængeligt kan et modul også vælges. 2.2. Overskrift. 2.3. Beskrivelse. 2.4. Prioritet (”Skal udvikles”, ”Bør udvikles”, ”Kunne være rart at have”, ”Kan vente”). 2.5. E-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere. 2.6. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet opretter et nyt ønske, tildeler dette et entydigt ID, sætter sagsstatus til ”Oprettet”, udviklingsstatus til ”Ikke påbegyndt”, sætter en oprettet dato/tid på ønsket, samt gemmer hvilken bruger der har oprettet sagen. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data. 6. include (Vedhæft filer).
Slut betingelser: Et nyt ønske er oprettet.
Alternative forløb: Brugeren fortryder.

For at komme videre i analysefasen, har jeg udarbejdet følgende klassediagram. Se næste side.



Figur 2 - Analyse klassediagram

Alle de gule klasser er entitetsklasser. Her har jeg valgt ikke at have nogle get og set metoder, da disse ikke giver nogen værdi for diagrammet. Jeg har valgt at opdele data og forretningslogik i hver sit sæt klasser. Bl.a. fordi jeg har valgt at bruge Hibernate⁴, til at mappe mellem den objekt orienteret verden og den relationelle database verden. Mere om det under design.

Entitetsklasserne består af Error og Request, som begge arver fra den abstrakte Case. Case indeholder alle de attributter som er fælles for Error og Request. Der kan knyttes diskussionsindlæg til fejl og ønsker vha. klassen Diskussion, og filer på samme måde vha. klassen File. File bruges også til at knytte filer til diskussionsindlæg. Brugere kan knyttes til fejl og ønsker vha. associationerne createdBy, responsibleDeveloper og responsibleTester. Brugeren knyttes også til fejl og ønsker indirekte gennem Produkt, som bruges til hvilke Produkter de enkelte brugere har adgang til. En bruger kan have en eller flere roller vha. Role. Vote bruges til håndtering af stemmer til et ønske.

⁴ Hibernate – Se www.hibernate.org

De lilla business klasser (også kaldet kontrolklasser, cirklen er en stereotype) er delt op i følgende klasser:

- MaintainCase – til håndtering af fejl og ønsker. Oprettelse, rettelse og søgning.
- MaintainDiscussion – til håndtering af diskussionsindlæg
- MaintainFile – til håndtering af filer som knyttes enten til en sag (fejl eller ønske) eller til et diskussionsindlæg
- MaintainChangelog – til håndtering af ændringer til fejl og ønsker.

Som det fremgår af overstående diagram, har jeg to typer klasser, de gule entitetsklasser og de røde kontrolklasser. Disse to typer har jeg valgt at placere i hver sin pakke. Se pakkediagrammet under bilag afsnit 8.6.

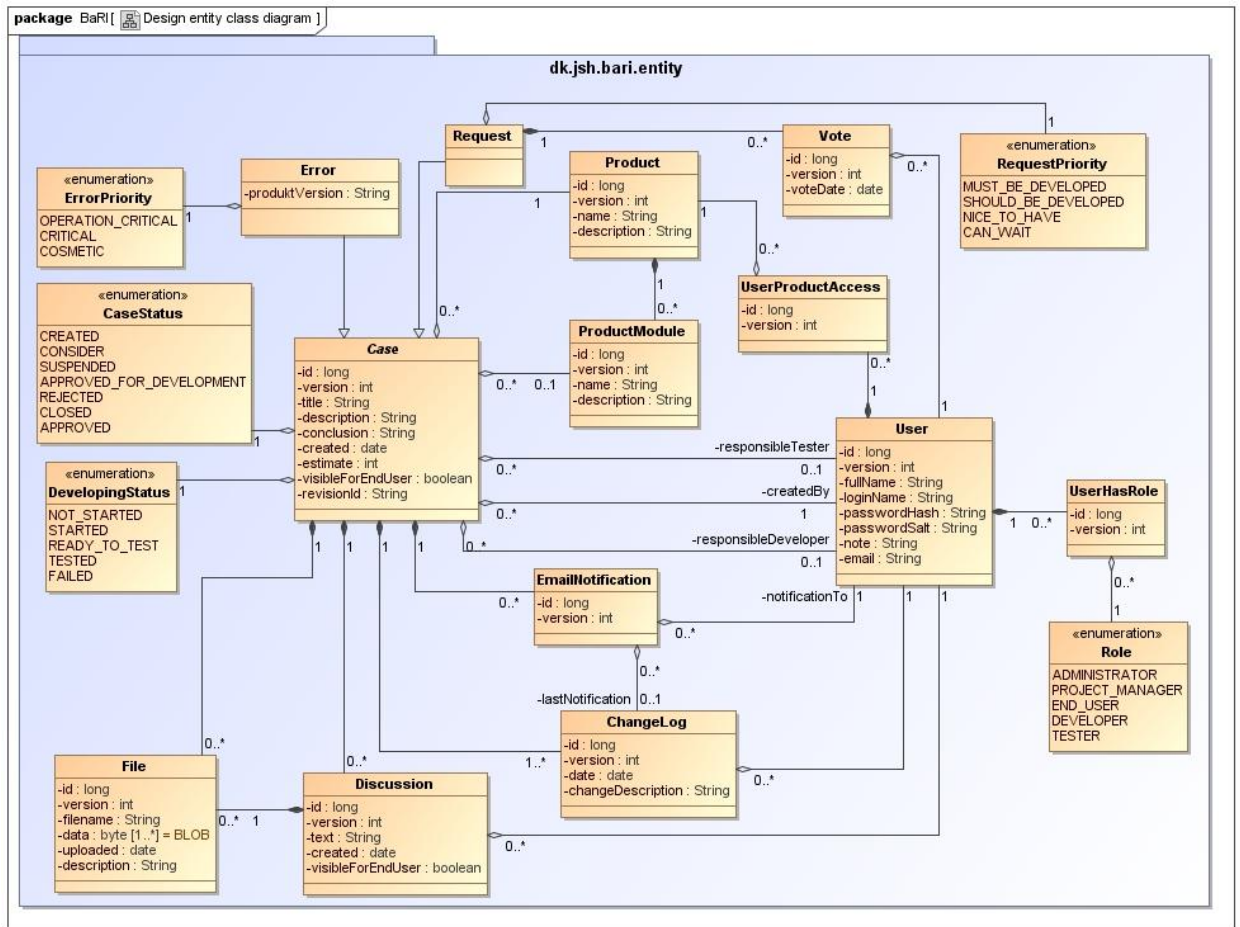
For at illustrere de forskellige tilstande en sag kan være i, har jeg udarbejdet et solskins scenarie aktivitetsdiagram. Dette diagram viser hvilke brugerroller, som er inde over en sag samt sagens statusser igennem forløbet. Se under bilag afsnit 8.7.

Jeg har valgt at ”låse” analysen, det skal forstås på den måde, at hvis jeg bliver ”klogere” senere i processen, f.eks. under design, så går jeg ikke tilbage og tilretter analysemodellerne, men indarbejder rettelserne i designmodellerne.

For at afgrænse opgaven har jeg ikke medtaget nogen analyse use case realiseringer.

4. DESIGN

Jeg er kommet frem til følgende klassediagram for mine entitetsklasser.



Figur 3 - Design entitetsklassediagram

Som det fremgår af diagrammet ligger alle mine entitetsklasser nu i en pakke kaldet `dk.jsh.bari.entity`, som er måden man navngiver pakker på i Java. Jeg har erstattet alle sagsbehandlings- og udviklingsprocesstatusser, samt prioritet attributter med enumerations klasser. Der er en form for Java konstanter.

De var to "mange til mange" relationer i analysediagrammet. Mellem Produkt og User, har jeg indsat `UserProductAccess` og mellem User og Role, har jeg indsat `UserHasRole`.

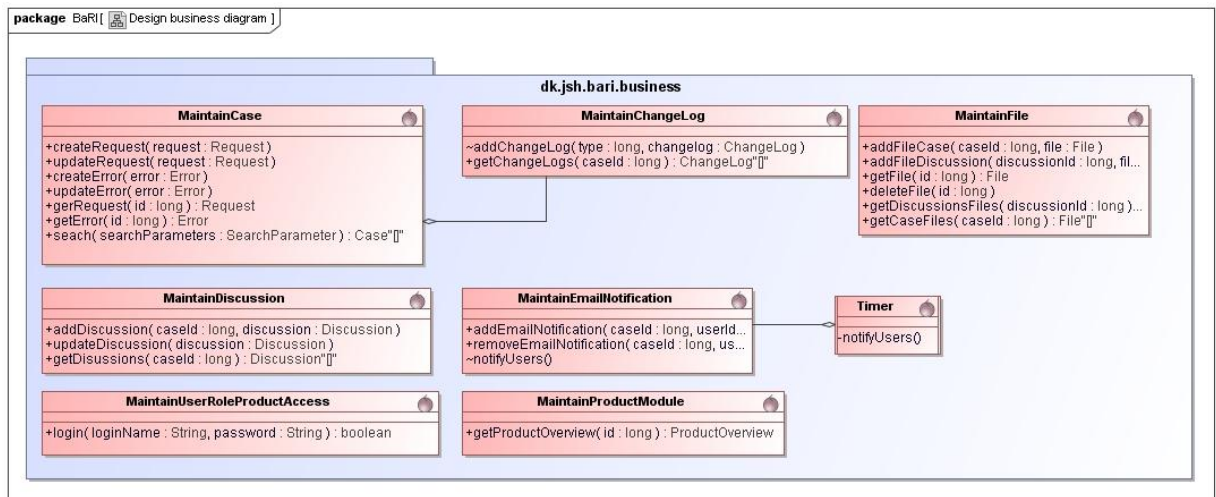
Derudover er alle associationer fra analysediagrammer rettet til enten aggregeringer eller kompositioner.

Alle ovenstående klasser, minus enumeration klasser, skal mærkes med Hibernate/Java annotations⁵. Så Hibernate ved hvordan de enkelte klasser, skal mappes til databasetabeller. Disse annotationer kan også benyttes til at generere et DDL⁶ database skema.

Mine kontrolklasser kan ses af følgende diagram, som ligger i pakken `dk.jsh.bari.business`.

⁵ Java Annotations – se mere en.wikipedia.org/wiki/Java_annotation

⁶ DDL – Data Definition Language – se mere en.wikipedia.org/wiki/Data_Definition_Language



Figur 4 – Design kontrolklassediagram

Her har jeg tilføjet nogle ekstra klasser, bl.a. en Timer, som er en singleton⁷ klasse, der vha. MaintainEmailNotification klassen, starter e-mail notifikationen med faste intervaller. Dette skal foregå i en separat programtråd⁸.

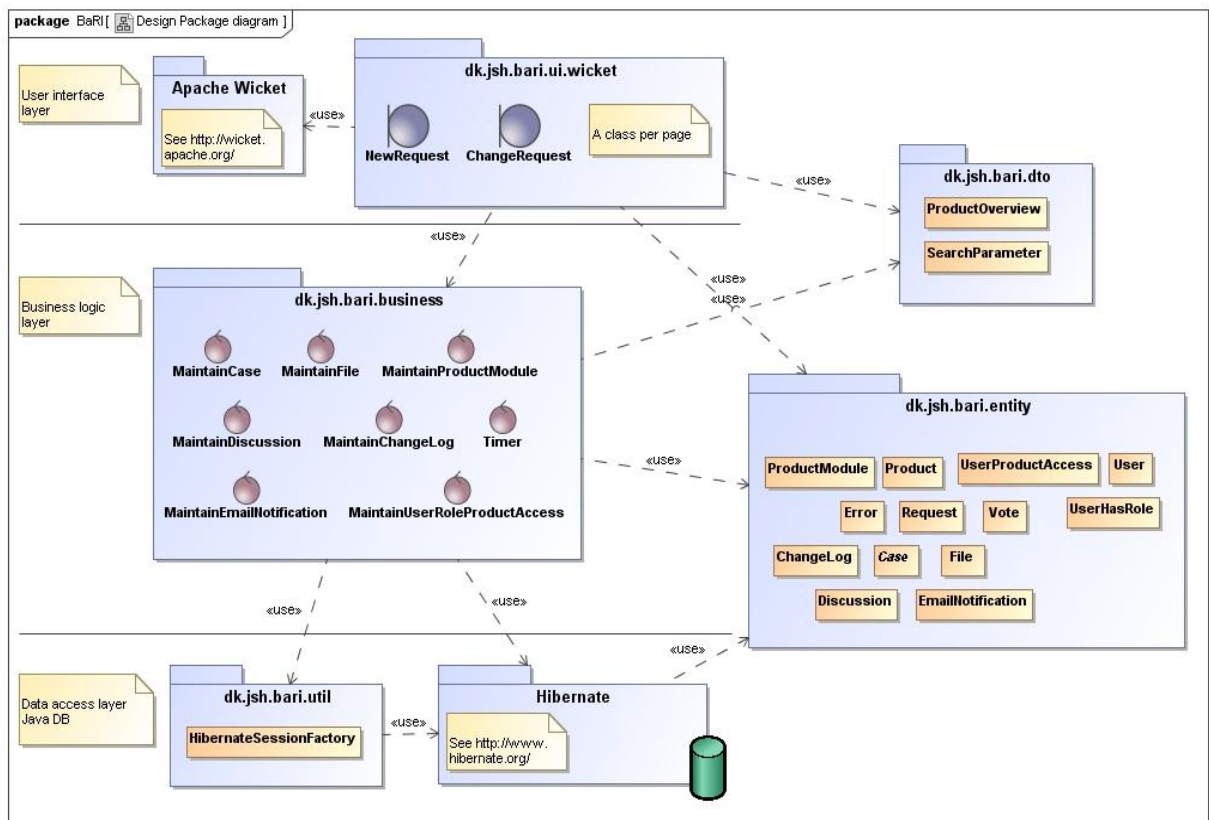
Derudover er der tilføjet klasserne MaintainProductModule og MaintainUserRoleProductAccess, som begge mangler en del metoder til at vedligeholde produkter/moduler, brugere og brugeradgang. Disse blev fravalgt i forbindelse med afgrænsning af opgaven under analysefasen. I MaintainProductModule har jeg en enkelt metode kaldet getProductOverview, som bruges til at fremvise en produktforside.

Under bilag, se afsnit 8.8, har jeg vedlagt et state machine diagram, der beskriver hvilke brugerroller, som må ændre fejl- og ønskestatusserne, sagsbehandlingsstatus og udviklingsprocesstatus.

⁷ Singleton – se mere en.wikipedia.org/wiki/Singleton_pattern

⁸ Programtråd – se mere [en.wikipedia.org/wiki/Thread_\(computer_science\)](http://en.wikipedia.org/wiki/Thread_(computer_science))

Det samlede design kommer til at se således ud:



Figur 5 - Design pakke diagram

For at afgrænse opgaven, har jeg valgt ikke at gå i dybden med pakkerne: dk.jsh.bari.wicket, dk.jsh.bari.util og dk.jsh.bari.dto. Wicket pakken er til bruger interface klasser. Og der vil komme en klasse pr. web side. Dvs. ca. en klasse pr. use case. Pakken dto er til entitetsklasser, som ikke skal gemmes i databasen. F. eks. klassen SearchParameter, der bruges i forbindelse med use casen ”Søg efter fejl og ønsker”, samt klassen ProductOverview som bruges i forbindelse med use casen ”Vis produkt forside”. Og sidst Util klassen der bruges til initiering af Hibernate.

Som det ses af pakke diagrammet, er det opdelt i følgende 3 lag:

- User interface – til håndtering af websiderne.
- Business logic – som er til håndtering af forretningslogik.
- Data Access – som håndterer database delen. Vha. Hibernate.

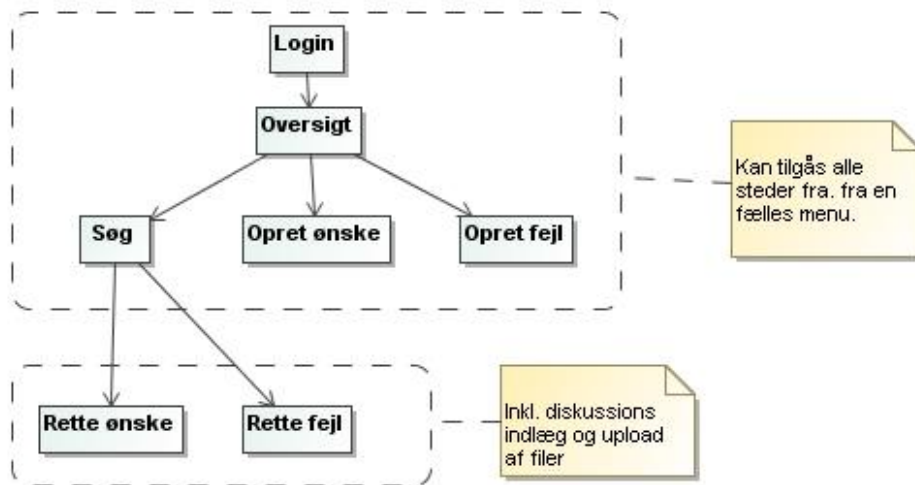
Pakkerne dk.jsh.bari.dto og dk.jsh.bari.entity, er til transport klasser mellem lagene. Derudover skal klasserne i entity pakkes have tilføjet Hibernate annotations, så Hibernate kan finde ud af at mappe disse klasser til tabeller i databasen. Hibernate kan også ud fra disse annotations danne database DDL skemafilere.

Pakkerne Apache Wicket og Hibernate, er med for at illustrere at disse to framework benyttes.

For at kvalitetssikre designet har jeg lavet nogle use case realiseringsdiagrammer i form af nogle sekvensdiagrammer. Se eksempler på disse under bilag afsnit 8.9.

4.1. WEBSITE DESIGN

Jeg er kommet frem til følge webside flow i BaRI.



Figur 6 - Side flow

Siderne "Oversigt", "Søg", "Opret ønske" og "Opret Fejl" kan tilgås fra alle sider i BaRI, dog ikke Login, via en fælles menulinje som går igen på alle sider.

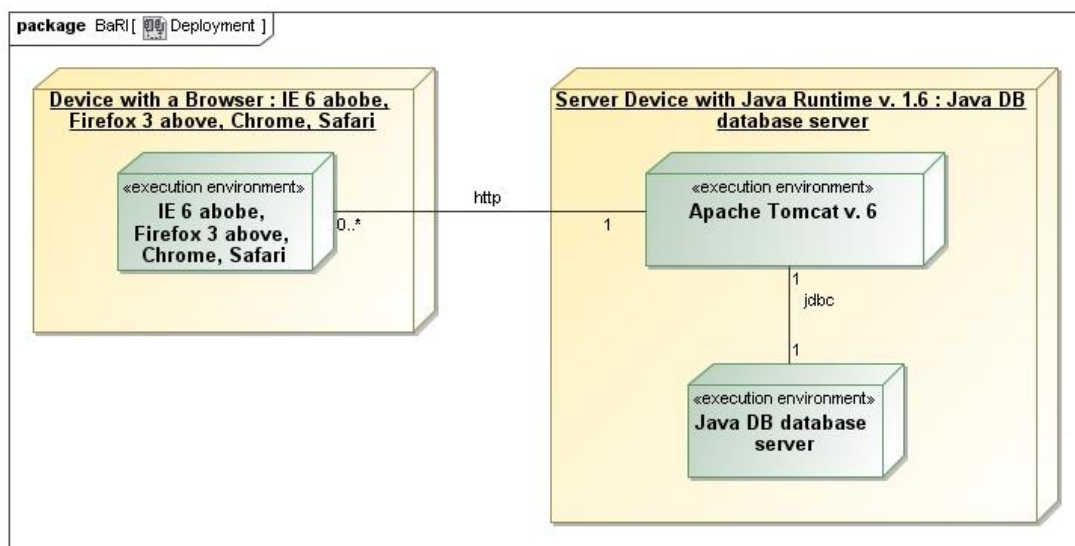
Under bilag afsnit 8.10 kan de enkelte webside layout ses.

5. IDRIFTSÆTTELSE

Da BaRI er en Java webapplikation, uden afhængigheder til andre systemer. Så er selve idriftsættelsen enkel. Det kræver en server med installeret Java og en Apache Tomcat Web Server⁹, samt en Java DB. Java DB er en del af Java. Dvs. at der ikke er nogen specielle krav til operativsystem, da Tomcat og Java DB kan køre på bl.a. Linux, Unix og Windows. Andre webservere kunne også benyttes.

Følgende er et deployment diagram.

⁹ Apache Tomcat – se mere tomcat.apache.org



Figur 7 - Deployment diagram

For at deploye en webapplikation i en Tomcat server, bygges der en war¹⁰ fil, vha. et udviklingssystem. Og denne war fil kopieres over i et specielt Tomcat applikationskatalog. Første gang systemet installeres skal databasen klargøres. Dette gøres med et SQL-script, som opretter alle tabellerne, samt indsætter en administratorbruger. Så denne bruger kan begynde at oprette brugere og produkter i systemet.

Som det også fremgår af ovenstående diagram, skal BaRI virke sammen med gængse browsere på markedet. Dvs. Internet Explore, Firefox, Opera, Chrome og Safari. Det vil kræve noget test. Men da antallet af antal websider er begrænset, er det en overkommelig opgave.

6. UP ITERATIONER

Indtil nu har denne rapport fuldt den meget udskældte vandfaldsmodel. Dvs. først indsamles der krav, disse analyseres, hvorefter systemet designes. Dette er også en logisk form, hvis et system skal beskrives i rapportformat. Men for at komme tilbage til UP, har jeg valgt at beskrive hvilke indledende iterationer, som BaRI kunne starte sin udvikling efter.

Iteration	Beskrivelse
1	Helt basal forretnings-logik til at hente brugere og produkt/modul oplysninger fra DB. Data til disse kan indsættes manuelt i databasen til at starte med.
2	Oprette et ønske, samt simpel mulighed for at fremsøge denne igen. Herfra kan de enkelte iterationer demonstreres.
3	Som 2 bare med en fejl.
4	Tilføje diskussionsindlæg til fejl og ønsker. Og vise disse igen.
5	Tilføje filer til fejl og ønsker. Og vise disse igen.
6	Tilføje filer til diskussionsindlæg. Og vise disse igen.
7	Tilføje e-mail notifikation.
	etc.

¹⁰ WAR – se mere [en.wikipedia.org/wiki/WAR_file_format_\(Sun\)](http://en.wikipedia.org/wiki/WAR_file_format_(Sun))

En produktejer kan løbende justere disse iterationer, efter hvad der giver mest værdi. Derfor bør alle iterationer ikke planlægges på forhånd, for at give mulighed for at justere undervejs. For hver af disse iterationer skulle BaRI løbende analyseres, designes, implementeres og testes.

7. KONKLUSION

Denne opgave har mere været en øvelse i den gamle vandfaldsmodel. For som jeg har skrevet tidligere, at få opgaven til passe ind i rapportformen. Der nødvendigvis må starte med krav og analyse og fortsætte med design og implementering. En anden grund til at UP ikke er fulgt, er at der ikke har været noget krav om udvikling. Og noget kørende kode, må være en forudsætning for iterative udviklingsprocesser. Man bør kunne vise et kørende program, til interessenterne, for at blive klogere på, om man er på rette vej.

Jeg er heller ikke sikker på, at UP er den rette proces for alle projekter. I dette tilfælde kunne XP¹¹ eller SCRUM¹² godt være brugt. UP har efter min mening mere sin berettigelse, hvis der er høje dokumentationskrav til projektet/produktet. Fordi der ligges stor vægt på UML diagrammer og tegninger i alle faser, samt hvilke dokumenter der er en forudsætning for at komme videre til næste fase i processen. Men til forskel for vandfaldsmodellen, så er UP dokumenterne noget der tilrettes for hver iteration.

Processer som XP og SCRUM ligger mere sin vægt på tests, simpelt design, refactoring¹³ og kørende kode, som kan bruges her og nu ellers i det mindste demonstreres.

Til slut har jeg et enkelt kritikpunkt til bogen UML and the Unified Process. Da denne helt forbigår brugerfladedesign, som jeg mener, er en vigtigt fase. Da det er her man, som designer, opdager mange fejl og mangler ved sit design.

¹¹ XP – eXtreme Programming se mere her en.wikipedia.org/wiki/Extreme_Programming og www.extremeprogramming.org/

¹² SCRUM – se mere her [en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development)) og www.scrumalliance.org/learn_about_scrum

¹³ Refactoring – se mere en.wikipedia.org/wiki/Code_refactoring

8. BILAG

8.1. KRAV

Følgende er en liste af krav til systemet, opdelt efter funktionelle og ikke funktionelle krav. De enkelte krav prioriteres efter MoSCoW (Must have, Should Have, Could Have, Wants to have).

Funktionelle krav:

ID	Krav	Prioritet
1	BaRI skal kunne oprette og vedligeholde ønsker til andre systemer.	M
2	BaRI skal kunne oprette og vedligeholde fejl som er fundet i andre systemer.	M
3	BaRI skal kunne knytte diskussionsindlæg til både ønsker og fejl. Et indlæg kan rettes af den bruger som har oprettet dette, så længe der ikke er nyere indlæg.	M
4	BaRI skal give mulighed for at vedhæfte forskellige filer, som knyttes til enten en fejl, et ønske eller et diskussionsindlæg.	M
5	BaRI skal kunne håndtere følgende sagsbehandlingsstatusser for både ønske og fejl: "Oprettet", "Behandles", "I bero", "Godkendt til udv.", "Afvist", "Lukket" og "Godkendt".	M
6	BaRI skal for godkendte fejl og ønsker kunne håndtere følgende udviklingsprocesstatusser: "Ikke påbegyndt", "Begyndt", "Klar til test", "Testet" og "Fejl fundet".	M
7	BaRI skal kunne prioritere ønsker med følgende prioriteter: "Skal udvikles", "Bør udvikles", "Kunne være rart at have" og "Kan vente".	M
8	BaRI skal kunne prioritere fejl med følgende prioriteter: "Driftkritisk", "Kritisk" og "Kosmetisk".	M
9	BaRI skal logge, alle ændringer på fejl og ønsker. Disse ændringer skal kunne vises i brugerfladen.	C
10	BaRI skal give mulighed for, at slutbrugere kan stemme om, hvilke ønsker der skal prioriteres.	C
11	BaRI skal kunne vise en grafisk overside pr. system, som viser antallet af ønsker med en given status, og på samme måde for fejl.	M
12	BaRI skal kunne operere med følgende brugerroller: Administrator, Slutbruger, Udvikler, Tester og Projektleder.	M
13	En bruger kan have en eller flere roller.	M
14	En BaRI administrator kan som den eneste oprette og rette systemer, nye brugere og bruger/system grupper.	M
15	BaRI skal kunne sende E-mail notifikationer, når der er ændringer på en fejl eller et ønske. De enkelte brugere skal for hvert ønske eller fejl, selv tilslutte sig notifikation.	C
16	BaRI skal kunne henvise til produktversioner for fejl. Og koderevisioner for fejl og ønsker.	C
17	BaRI skal tildele alle fejl og ønsker et entydigt ID, som kan bruges til at fremsøge sagen, samt skal kunne bruges som henvisning/link fra andre sager/diskussionsindlæg.	M
18	BaRI skal give mulighed for at opdele systemer i moduler, så fejl og ønsker kan henvise til et modul inden for et system. Dette skal dog være valgfrit.	C
19	BaRI skal give mulighed for at afgive estimater på fejl og ønsker.	M
20	BaRI skal give alle ikke slut-brugere mulighed for at mærke fejl, ønsker og diskussionsindlæg som "ikke synlige for slut-brugere".	M
21	BaRI skal give en projektleder mulighed for at tildele en udvikler og en tester til fejl og ønsker.	M

Følgende er funktionelle krav, som er opstillet i rollematrixer. Rollematrix (C=Create, R=Read, U=Update og D=Delete)

Krav ID 100	Fejl	Ønske	Diskussionsindlæg	Uploads	Bruger	Gruppe	System
Administrator	CRUD	CRUD	CRUD	CRUD	CRUD	CRUD	CRUD
Projektleder	CRU	CRU	CRU	CRU			
Udvikler	CRU	CRU	CRU	CRU			
Tester	CRU	CRU	CRU	CRU			
Slut-bruger	CRU	CRU	CRU				

De enkelte rollers mulighed for at rette fejl- og ønskestatusser.

Sagsbehandlingsstatusser, brugere kan ændre status til følgende:

Krav ID 101	Oprettet	Overvejes	I bero	Godkendt til udv.	Godkendt	Afvist	Lukket
Administrator	J	J	J	J	J	J	J
Projektleder	J	J	J	J	J	J	J
Udvikler	J	N	N	N	N	N	N
Tester	J	N	N	N	N	N	N
Slut-bruger	J	N	N	N	J	N	N

Udviklingsprocesstatus:

Krav ID 102	Ikke pågyndt	Begyndt	Klar til test	Testet	Fejl fundet
Administrator	J	J	J	J	J
Projektleder	J	J	J	J	J
Udvikler	J	J	J	N	N
Tester	N	N	N	J	J
Slut-bruger	N	N	N	J	J

Sagsbehandlingsstatus og udviklingsprocesstatus, er også beskrevet vha. af et aktivitetsdiagram, se afsnit 8.7 og et state machine diagram, se afsnit 8.8.

Ønskeprioriteter:

Krav ID 103	Skal udvikles	Bør udvikles	Kunne være rart	Kan vente
Administrator	J	J	J	J
Projektleder	J	J	J	J
Udvikler	N	N	N	N
Tester	N	N	N	N
Slut-bruger	J	J	J	J

Fejlprioriteter:

Krav id 104	Driftkritisk	Kritisk	Kosmetisk
Administrator	J	J	J
Projektleder	J	J	J
Udvikler	J	J	J
Tester	N	N	N
Slut-bruger	J	J	J

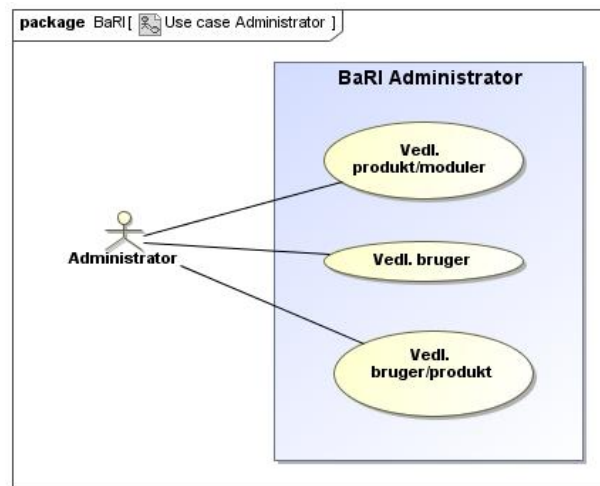
Ikke funktionelle krav:

ID	Krav	Prioritet (MoSCoW)
B1	BaRI skal være en WEB-Løsning.	M
B2	BaRI skal kunne køre på en Apache Tomcat version 6 eller nyere.	M
B3	BaRI skal benytte følgende 2 java frameworks skal benyttes: Apache Wicket and Hibernate.	M
B4	BaRI skal persistere data i en JavaDB.	M
B5	BaRI skal kunne benyttes sammen med HTTPS.	M
B6	BaRI skal benytte optimistisk låsning, vha. Hibernate.	M

8.2. DANSKE/ENGELSKE TERMER

Gruppe	Dansk term	Engelsk term	Beskrivelse
Brugerroller	Administrator	Administrator	En BaRI administrator. Som kan oprette, rette og slette produkter og brugere.
	Projektleder	Project Manager	
	Slut-bruger	End user	
	Udvikler	Developer	
	Tester	Tester	
Entiteter	Sag	Case	En abstrakt sag.
	Ønske	Request	
	Fejl	Error	
	Sagsbehandlingsstatus	Case status	
	Udviklingsprocesstatus	Developing status	
	Fejlprioritet	Error priority	
	Ønskeprioritet	Request priority	
	Fil	File	
	Diskussion	Discussion	
	Ændringslog	Change log	
	Produkt	Product	Kaldes og system bl.a. i krav.
	Produktmodul	Product module	
	E-mail notifikation	E-Mail notification	
	Stemme	Vote	
	Rolle	Role	
	Sagsbehandlings statusser	Oprettet	Created
		Behandles	Consider
		I bero	Suspended
		Godkendt til udvikling	Approved for development
		Afvist	Rejected
		Lukket	Closed
		Godkendt	Approved
	Udviklingsproces statusser	Ikke påbegyndt	Not started
		Begyndt	Started
		Klar til test	Ready to test
		Fejl fundet	Failed
		Testet	Tested
	Fejlprioriteter	Driftkritisk	Operation critical
		Kritisk	Critical
		Kosmetisk	Cosmetic
	Ønskeprioriteter	Skal udvikles	Must be developed
		Bør udvikles	Should be developed
		Kunne være rart at have	Can wait
		Kan vente	Nice to have

8.3. ADMINISTRATIVE USE CASES



Figur 8 - Administrative Use Cases

8.4. USE CASE BESKRIVELSER

Herunder er de enkelte use case beskrivelser. De fleste use cases er lig med en webside. Disse kan findes under afsnit 8.10.

Use case: Opret ønske
ID: 1
Kort beskrivelse: En bruger opretter et nyt ønske.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter.
Hovedforløb: <ol style="list-style-type: none">1. Brugeren vælger "Opret nyt ønske" linket.2. Systemet viser en tom side, med følgende felter:<ol style="list-style-type: none">2.1. Produkt skal vælges, og hvis tilgængeligt kan et modul også vælges.2.2. Overskrift.2.3. Beskrivelse.2.4. Prioritet ("Skal udvikles", "Bør udvikles", "Kunne være rart at have", "Kan vente").2.5. E-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere.2.6. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger).3. Brugeren udfylder felterne og vælger at gemme.4. Systemet opretter et nyt ønske, tildeler dette et entydigt ID, sætter sagsstatus til "Oprettet", udviklingsstatus til "Ikke påbegyndt", sætter en oprettet dato/tid på ønsket, samt gemmer hvilken bruger der har oprettet sagen.5. Systemet opretter en post i en ændrings loggen, med ovenstående data.6. include (Vedhæft filer)
Slut betingelser: Et nyt ønske er oprettet.
Alternative forløb: Brugeren fortryder.

Use case: Opret fejl
ID: 2
Kort beskrivelse: En bruger opretter en ny fejl.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger ”Opret ny fejl”. 2. Systemet viser en tom side, med følgende felter: <ol style="list-style-type: none"> 2.1. Produkt skal vælges, og hvis tilgængeligt kan et modul også vælges. 2.2. Overskrift. 2.3. Beskrivelse. 2.4. Prioritet (Driftkritisk”, ”Kritisk” eller ”Kosmetisk”). 2.5. E-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når denne fejl rettes af andre brugere. 2.6. Produktversion. 2.7. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 2.8. include (Vedhæft fil). 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet opretter et nyt ønske, og tildeler dette et entydigt ID, og sætter sagsstatus til ”Oprettet”, udviklingsstatus til ”Ikke påbegyndt” og sætter en oprettet dato/tid på ønsket, samt gemmer hvilken bruger der har oprettet sagen. 5. Systemet opretter en post i en ændrings loggen, med ovenstående data.
Slut betingelser: En ny fejl er oprettet.
Alternative forløb: Brugeren fortryder.

Use case: Opret diskussionsindlæg
ID: 3
Kort beskrivelse: En bruger opretter et nyt diskussionsindlæg til en fejl eller et ønske.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren ”står” på enten et ønske eller en fejl, som denne ønsker at oprette et diskussionsindlæg til.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger at ”Tilføj indlæg” linket. 2. Systemet giver mulighed for at indtaste følgende: <ol style="list-style-type: none"> 2.1. Selve indlægget. Her skal det være muligt at skrive #sagsid for at linke til en anden sag. 2.2. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 2.3. include (Vedhæft filer). 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet opretter et nyt diskussionsindlæg, som knyttes til den aktuelle sag (ønske eller fejl) og den aktuelle bruger. Tidspunkt for oprettelse af indlægges gemmes også. 6. Systemet opretter en post i ændringsloggen, med ovenstående data.
Slut betingelser: Et diskussionsindlæg er oprettet og knyttet til et ønske eller en fejl.
Alternative forløb: Brugeren fortryder.

Use case: Vis/ret ønske
ID: 4
Kort beskrivelse: En bruger retter oplysninger på et eksisterende ønske.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og det pågældende ønske er til et af disse produkter.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren kommer til ønsket enten via et link fra en anden sag, eller fra et link fra en søge side (Se use case "Søg efter ønsker og fejl"). 2. Systemet viser en side, med følgende felter: <ol style="list-style-type: none"> 2.1. Produkt (låst). 2.2. Evt. modul (låst). 2.3. Overskrift. 2.4. Beskrivelse. 2.5. Oprettelsesdag (låst). 2.6. Prioritet ("Skal udvikles", "Bør udvikles", "Kunne være rart at have", "Kan vente"). 2.7. E-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere. 2.8. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 2.9. Konklusion. 2.10. Revisions ID (Vises kun for slut-bruger og tester, andre brugere kan rette). 2.11. Estimat (Vises kun for slut-bruger, andre brugere kan rette). 2.12. Stemme, har den aktuelle bruger stemt på dette ønske (Kun for slut-brugere). 2.13. Antal stemmer dette ønske har i alt (låst). 2.14. Sagsbehandlingsstatus ("Oprettet", "Behandles", "I bero", "Godkendt", "Afvist", "Lukket"). Hvis et ønske er Afvist eller Lukket, kan ingen data rettes, medmindre en administrator eller en projektleder retter status tilbage til enten "Oprettet", "Behandles" eller "I bero". 2.15. Udviklingsstatus ("Ikke påbegyndt", "Begyndt", "Klar til test", "Testet", "I produktion"). 2.16. Hvis bruger er projektleder. <ol style="list-style-type: none"> 2.16.1. Mulighed for at tildele en ansvarlig udvikler og tester. 2.17. Derudover vises alle eksisterende diskussionsindlæg, med mulighed for at rette sidste, hvis dette er oprettet af samme bruger. 2.18. include (Vedhæft filer). 2.19. include (Slet filer). 2.20. include (Vis/hent fil). 2.21. include (Vis ændringsloggen). 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet retter et ønske. 5. Systemet opretter en post i en ændringsloggen, med ovenstående data.
Slut betingelser: Et ønske er rettet.
Alternative forløb: Brugeren fortryder.

Use case: Vis/ret fejl
ID: 5
Kort beskrivelse: En bruger retter oplysninger på en eksisterende fejl.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og den pågældende fejl er til et af disse produkter.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren kommer til fejlen enten via et link fra en anden sag, eller fra et link fra en søge side (Se use case Søg efter ønsker og fejl). 2. Systemet viser en side, med følgende felter: <ol style="list-style-type: none"> 2.1. Produkt (låst). 2.2. Evt. modul (låst). 2.3. Overskrift. 2.4. Beskrivelse. 2.5. Oprettelsesdag (låst). 2.6. Prioritet (Driftkritisk", "Kritisk", "Kosmetisk"). 2.7. E-mail notifikation – ønsker den aktuelle bruger e-mail notifikation, når dette ønske rettes af andre brugere. 2.8. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 2.9. Konklusion. 2.10. Revisions id (Vises kun for slut-bruger og tester, andre brugere kan rette). 2.11. Estimat (Vises kun for slut-bruger, andre brugere kan rette). 2.12. Stemme, har den aktuelle bruger stemt på dette ønske (Kun for slut-brugere). 2.13. Sagsbehandlingsstatus ("Oprettet", "Behandles", "I bero", "Godkendt", "Afvist", "Lukket") Hvis en fejl er Afvist eller Lukket, kan ingen data rettes, medmindre en administrator eller en projektleder retter status tilbage til enten "Oprettet", "Behandles" eller "I bero". 2.14. Udviklingsstatus ("Ikke påbegyndt", "Begyndt", "Klar til test", "Testet", "I produktion"). 2.15. Hvis bruger er projektleder: <ol style="list-style-type: none"> 2.15.1. Mulighed for at tildele en ansvarlig udvikler og tester. 2.16. Derudover vises alle eksisterende diskussionsindlæg. Med mulighed for at rette sidste, hvis dette er oprettet af samme bruger. 2.17. include (Vedhæft filer). 2.18. include (Slet filer). 2.19. include (Vis/hent fil). 2.20. include (Vis ændrings loggen). 3. Brugeren udfylder felterne og vælger at gemme. 4. Systemet retter et ønske. 5. Systemet opretter en post i en ændringsloggen, med ovenstående data.
Slut betingelser: En fejl er rettet.
Alternative forløb: Brugeren fortryder.

Use case: Vis/ret diskussionsindlæg
ID: 6
Kort beskrivelse: En bruger vises og får mulighed for at rette et diskussionsindlæg, for enten et ønske eller en fejl.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter. Og dette indlæg er til et ønske eller en fejl, til et af disse produkter. Det er kun det sidste indlæg, der kan rettes. Og det er kun den bruger som har oprettet indlægget, som kan rette dette.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger ”ret” knappen på det sidste diskussionsindlæg. Knappen vil kun være synlig hvis det er den samme bruger som har oprettet dette indlæg. 2. Systemet giver mulighed for at rette følgende: <ol style="list-style-type: none"> 2.1. Selve indlægget. Her skal det være muligt at skrive #sagsid for at linke til en anden sag. 5.1. Synlig for slut-bruger (vises ikke hvis aktøren en er slut-bruger). 2.2. include (Vedhæft filer). 2.3. include (Slet fil). 3. Brugeren retter felterne og vælger at gemme. 4. Systemet retter et nyt diskussionsindlæg. 5. Systemet opretter en post i en ændringsloggen, med ovenstående data.
Slut betingelser: Et diskussionsindlæg er rettet.
Alternative forløb: Brugeren fortryder.

Use case: Vis produkt forside
ID: 7
Kort beskrivelse: En bruger vises en produktforside, med en statusoversigt over antallet af fejl og ønsker.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet, så denne har adgang til et eller flere produkter.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger et af de produkter denne har adgang til. Siden er forudfyldt med data for det første produkt i listen. (Sorteret alfabetisk). 2. Systemet viser 4 lagkagediagrammer: (med antal i hvert stykke): <ol style="list-style-type: none"> 2.1. Et for alle ønsker, opdelt efter sagsbehandlingsstatus. 2.2. Et for alle ønsker, opdelt efter udviklingsstatus. 2.3. Et for alle fejl, opdelt efter sagsbehandlingsstatus. 2.4. Et for alle fejl, opdelt efter udviklingsstatus. 3. Under ønske diagrammerne skal det totale antal ønsker stå. 4. Under fejl diagrammerne skal det totale antal fejl stå. 5. Samt en liste med de 20 fejl/ønsker til dette produkt, som seneste er oprettet eller rettet, med direkte link til disse.
Slut betingelser: Ingen.
Alternative forløb: Brugeren fortryder.

Use case: Søg efter ønsker og fejl
ID: 8
Kort beskrivelse: En bruger har mulighed for at fremsøge fejl og ønsker.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren er oprettet i systemet.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger et af de produkter denne har adgang til. 2. Hvis det valgte produkt, har tilknyttet moduler: <ol style="list-style-type: none"> 2.1. Så kan brugeren også vælge imellem disse, eller "alle" moduler som er valgt på forhånd. 3. Brugeren vælger mellem fejl, ønske eller begge. Begge er valgt på forhånd. 4. Brugeren vælger sagsbehandlingsstatus, her er alle også valgt på forhånd. 5. Brugeren vælger udviklingsstatus, her er alle også valgt på forhånd. 6. Brugeren har mulighed for et at udfylde et fritext søgefelt, som søger i alle tekstfelter for fejl, ønsker og diskussionsindlæg. Hvis et ID for selve en fejl eller et ønske kendes i forvejen, så kan dette også indtastes her, for at finde den aktuelle sag samt sager som henviser til denne. 7. Brugeren trykker på søg knappen. 8. Systemet finder alle de sager, som opfylder ovenstående søgekriterier, og viser disse på en liste med følgende kolonner: <ol style="list-style-type: none"> 8.1. Id, type (fejl eller ønske), overskrift, oprettet dato, sidste rettelsesdato, sagsbehandlingsstatus, udviklingsstatus, prioritet og antal stemmer. Listen kan sorteres ved at trykke på de enkelte kolonneoverskrifter. 9. Brugeren kan vælge at trykke på en linje i listen for at få denne vist (Se use cases: Vis/ret ønske og Vis/ret fejl).
Slut betingelser: Ingen.
Alternative forløb: <ul style="list-style-type: none"> - Der skal være en mulighed for at indtaste et sags ID, så brugeren har mulighed for at gå direkte til en sag vha. dette id. - Brugeren fortryder.

Use case: E-mail notification
ID: 9
Kort beskrivelse: Systemet har mulighed for at sende e-mail notifikationer til en bruger.
Primære aktører: Systemet (Tiden).
Sekundære aktører: Alle brugere af systemet
Start betingelser: Brugernes e-mail adresser findes og er valide.
Hovedforløb: <ol style="list-style-type: none"> En gang i timen mellem kl. 7 og 22. <ol style="list-style-type: none"> For hver fejl og ønske en bruger har ønsket at blive notificeret om vil: <ol style="list-style-type: none"> Systemet undersøger om der er sket ændringer på sagen, vha. ændringsloggen: Hvis dette er tilfældet: <ol style="list-style-type: none"> Vil systemet sende en e-mail til denne bruger, med følgende oplysninger: <ol style="list-style-type: none"> Fejl/eller ønske ID. Overskrift på sagen. Hvis det sidste er et nyt diskussionsindlæg skrives dette. Link til sagen.
Slut betingelser: En eller flere brugere er blevet notificeret.
Alternative forløb: Ingen.

Use case: Vedhæft filer
ID: 10
Kort beskrivelse: En bruger vedhæfter en eller flere filer til et ønske, en fejl eller et diskussionsindlæg.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Et ønske, en fejl eller et diskussionsindlæg findes som filerne kan tilknyttes.
Hovedforløb: <ol style="list-style-type: none"> Brugeren vælger ”vedhæft filer” linket. For hver fil som en bruger ønsker at vedhæfte gælder at: <ol style="list-style-type: none"> Aktøren udfylder en kort beskrivelse af filens indhold, samt vælger den fil der skal vedhæftes. Systemet gemmer: beskrivelse, filnavn, dato/tid for vedhæftningen, samt fil data og knytter filen til enten et ønske, fejl eller et diskussionsindlæg. Systemet opretter en post i ændringsloggen med ovenstående data.
Slut betingelser: En eller flere filer er vedhæftet.
Alternative forløb: Ingen.

Use case: Vis/hent fil
ID: 11
Kort beskrivelse: Mulighed for at vise eller hente en fil (Så den kan gemmes lokalt).
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren har adgang til produktet hvortil filen er knyttet. Enten til en fejl, et ønske eller et diskussionsindlæg.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger ”vis fil” linket, på enten en fejl, ønske eller diskussionsindlæg. Der kan være 0 eller mange. 2. Systemet streamer (sender) den aktuelle fil til brugers browser. 3. Browseren giver brugeren mulighed for at vise filen, hvis denne er i et format, den kan arbejde med (f.eks. billede formater, html, pdf etc.). 4. Ellers giver browseren mulighed for at gemme filen lokalt.
Slut betingelser: En fil er vist eller hentet og gemt lokalt.
Alternative forløb: Brugeren fortryder.

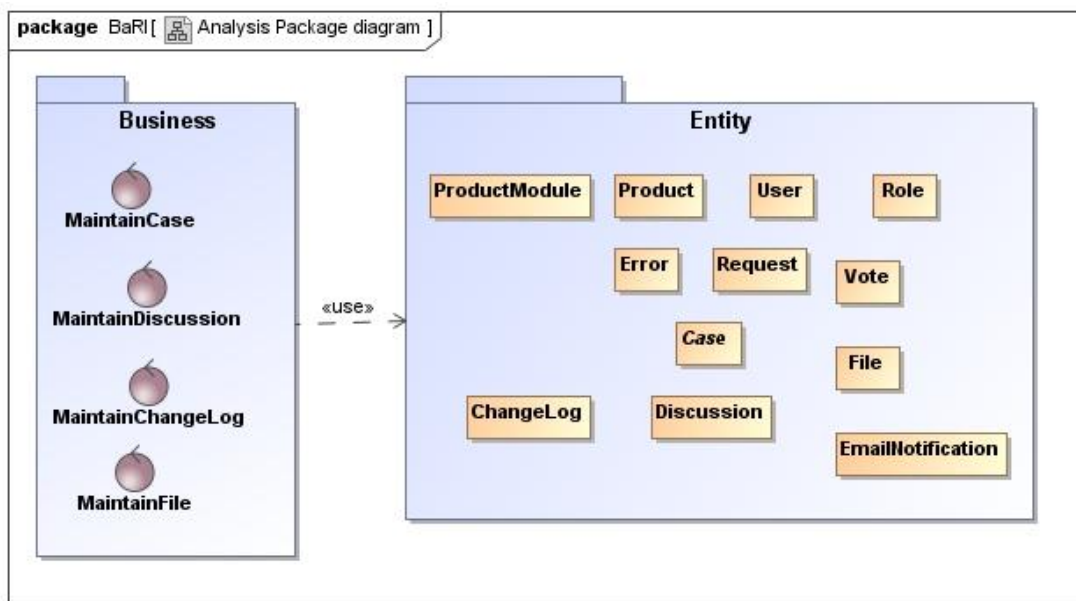
Use case: Slet filer
ID: 12
Kort beskrivelse: Mulighed for at slette en fil.
Primære aktører: Alle aktører har denne mulighed.
Sekundære aktører: Ingen.
Start betingelser: Brugeren har adgang til produktet, hvortil filen er knyttet enten via en fejl, et ønske eller et diskussionsindlæg.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren vælger ”slet fil” linket, på enten en fejl, ønske eller diskussionsindlæg. Der kan være 0 eller mange. På diskussionsindlæg er det kun det sidste indlæg der kan slettes filer fra. Og det er kun af samme bruger som har lavet indlægget. 2. Systemet sletter filen.
Slut betingelser: En fil er slettet.
Alternative forløb: Brugeren fortryder.

Use case: Vis ændringsloggen
ID: 13
Kort beskrivelse: Se ændringsloggen for en fejl eller et ønske.
Primære aktører: Alle aktører har denne mulighed. Dog ikke slut-brugere, da der i loggen kan være oplysninger som slut-bruger ikke må se. Se Feltet ”synlig for slut-bruger” i tidligere use cases.
Sekundære aktører: Ingen.
Start betingelser: Brugeren har adgang til produktet, hvortil en fejl eller et ønske er tilknyttet.
Hovedforløb: <ol style="list-style-type: none"> 1. Brugeren er på et ønske eller en fejl. (Se use cases ”Vis/ret fejl” og ”Vis/ret ønske”). 2. Brugeren vælger ”Vis ændringsloglinket”. 3. Systemet viser en side med følgende oplysninger (Listeform, nyeste først): <ol style="list-style-type: none"> 3.1. Dato/tid for ændringen. 3.2. Bruger. 3.3. Brugers rolle. 3.4. Om det er en oprettelse, rettelse eller en sletning. 3.5. Type: fejl, ønske eller diskussionsindlæg. 3.6. Overskrift. 4. Brugeren kan trykke på en af ovenstående ændringer. 5. Systemet viser i fri tekst hvad ændringerne er. (F.eks. før og efter værdien på forskellige felter på en fejl eller et ønske, eller teksten fra et diskussionsindlæg som er tilføjet etc.)
Slut betingelser: Brugeren har set ændringsloggen for en fejl eller et ønske.
Alternative forløb: Ingen.

8.5. KRAV/USE CASE MATRIX

Krav	1. Opret ønske	2. Opret fejl	3. Opret disk.	4. Vis/ret ønske	5. Vis/ret fejl	6. Vis/ret disk.	7. Vis produktforside	8. Søg efter ønsker og fejl	9. E-mail notifikation	10. Vedhæft filer	11. Slet fil	12. Vis/hent fil	13. Vis ændringsloggen	14. Adm. Use cases
1	✓			✓										
2		✓			✓									
3			✓			✓								
4	✓	✓		✓	✓					✓				
5	✓	✓		✓	✓									
6	✓	✓		✓	✓									
7	✓			✓										
8		✓			✓									
9	✓	✓	✓	✓	✓	✓				✓	✓		✓	
10				✓										
11						✓								
12		✓		✓	✓									✓
13														✓
14														✓
15				✓	✓				✓					
16		✓		✓	✓									
17	✓	✓	✓			✓								
18	✓	✓												✓
19				✓	✓									
20			✓	✓	✓	✓								
21				✓	✓									

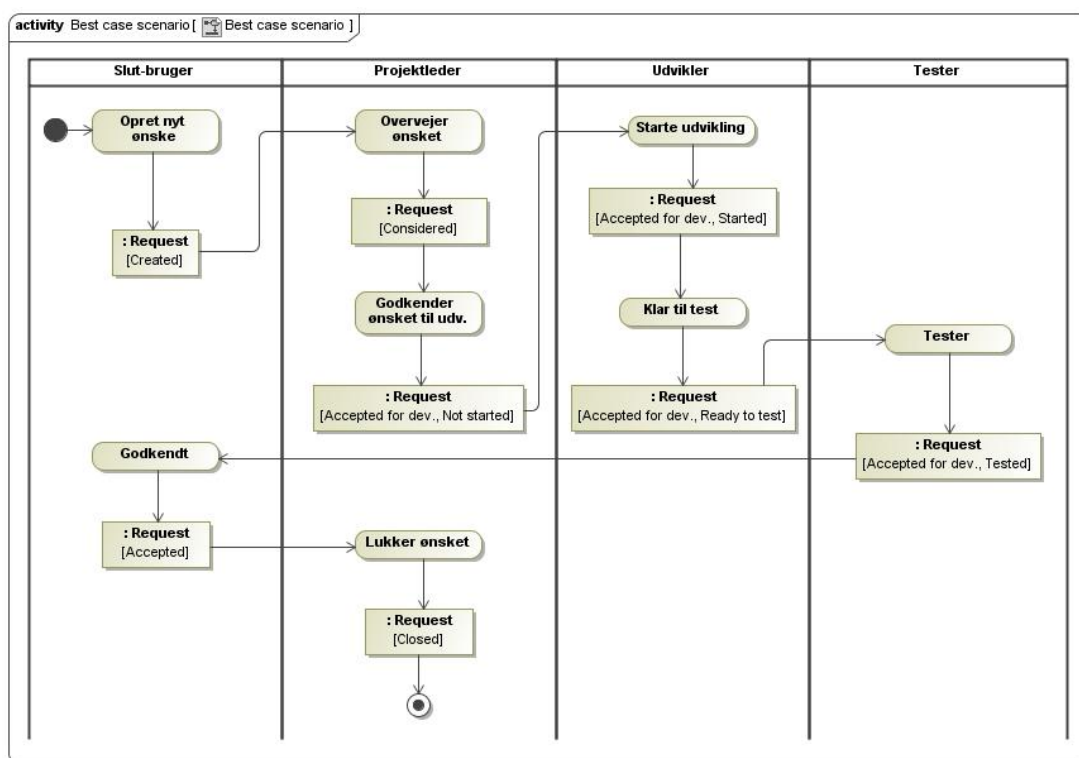
8.6. ANALYSE PAKKEDIAGRAM



Figur 9 - Analyse pakkediagram

Som det fremgår af overstående diagram, har jeg lagt mine klasser i to pakker. Business og Entity pakkerne. Business håndterer kald fra brugerfladen til databasen. Og Entity klasserne er databærende klasser.

8.7. SOLSKINS SCENARIOE AKTIVITETSDIAGRAM

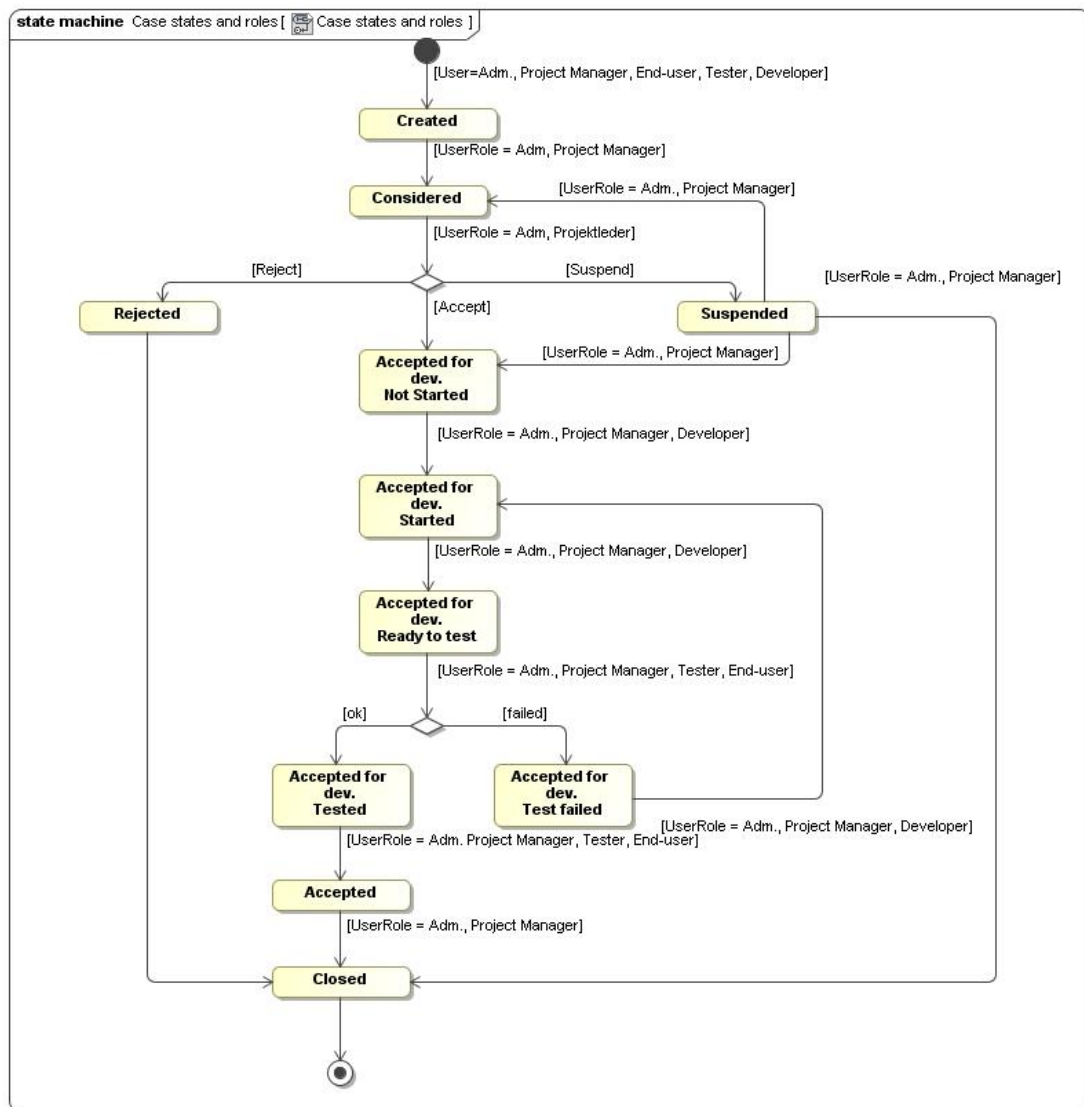


Figur 10 – Solskins scenarie aktivitetsdiagram

Ovenstående aktivitetsdiagram beskriver et solskins scenarie for et ønske. På request objektet kan ses sagsbehandlingsstatus og udviklingsprocesstatus (I de kantede parenteser).

8.8. STATE MACHINE DIAGRAM FOR EN SAG

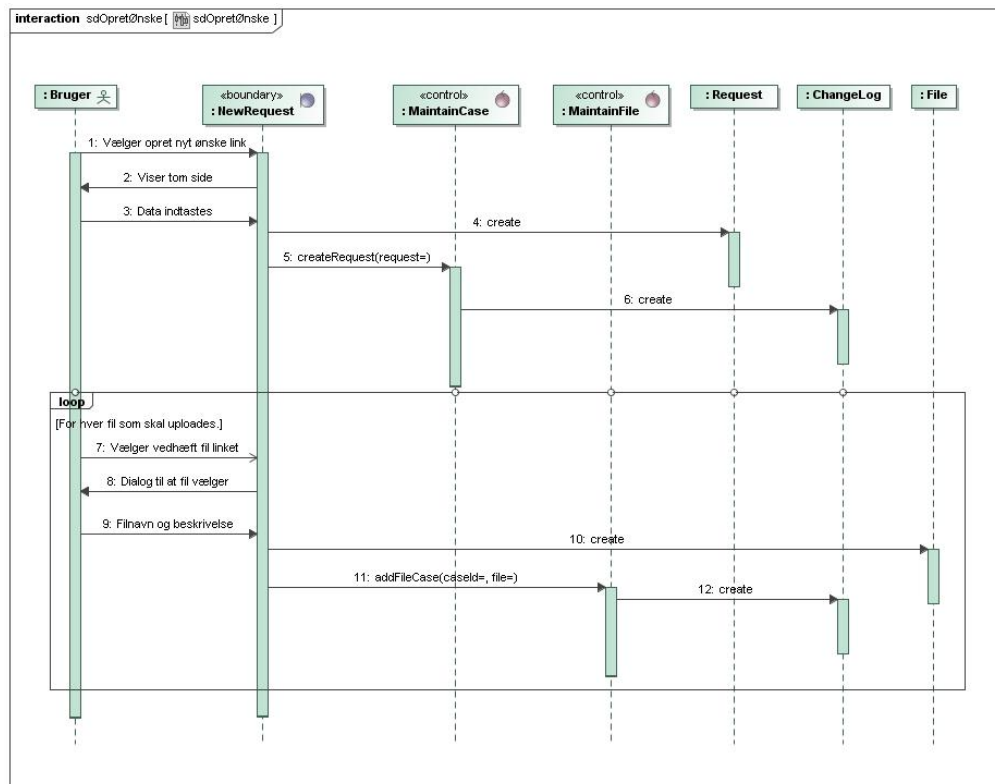
Diagrammet herunder viser hvilke brugerroller, som kan ændre de to sagsstatusser, der er fælles for fejl og ønsker. Alle brugere kan oprette fejl og ønsker.



Figur 11 - State machine diagram for Request og Error

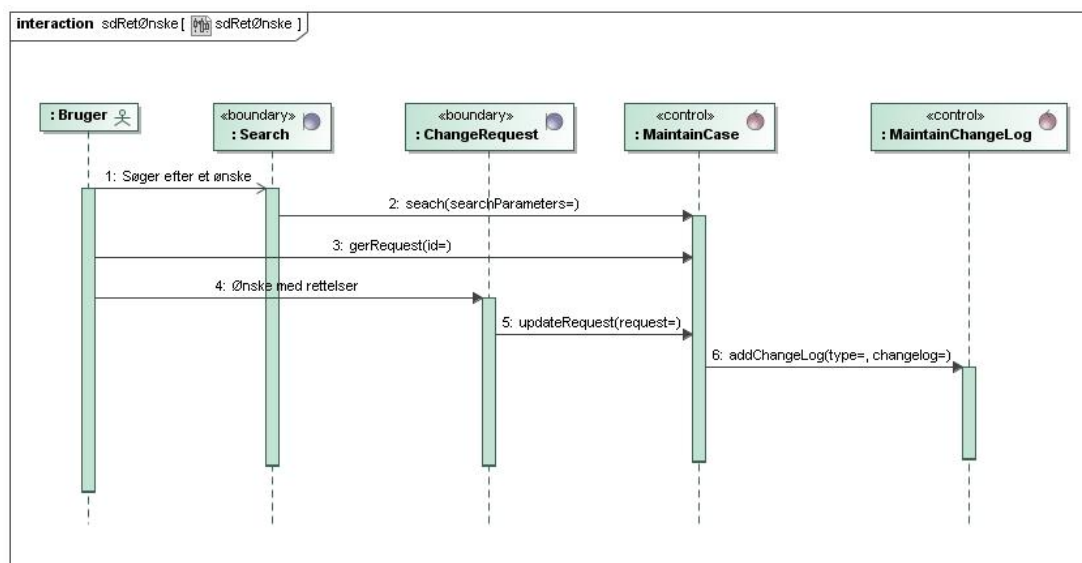
8.9. DESIGN USE CASE REALISERING

Følgende sekvensdiagram er en use case realisering af use casen ”Opret ønske”.



Figur 12 - Use case realisering af ”Opret Ønske”

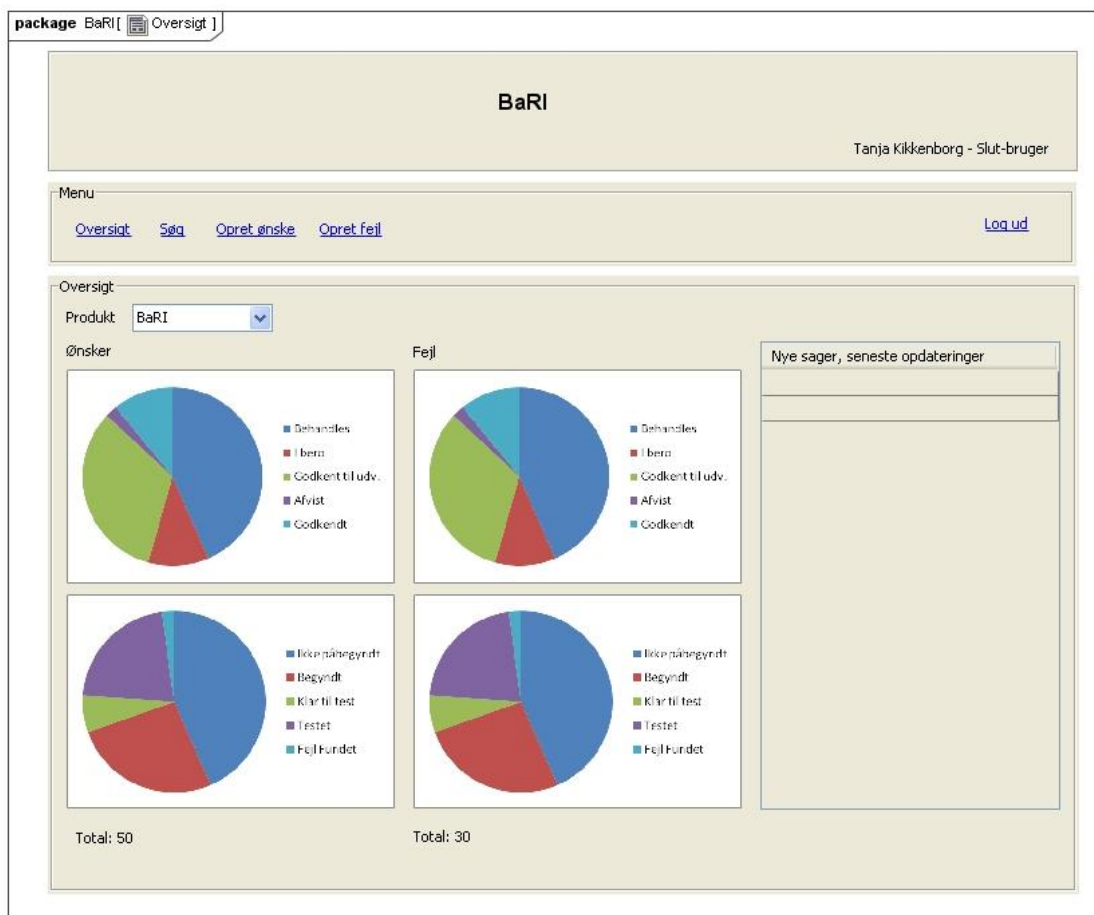
Følgende sekvensdiagram berører ”Søg efter ønsker og fejl” og ”Ret ønske”. Jeg har ikke medtaget håndtering af filer, da det vil ligne det fra Figur 12.



Figur 13 – Use case realisering af Søg og Ret ønske

8.10. WEBSITE DESIGN

Følgende webside er den første som vises efter login.



Figur 14 - Oversigt

Overskriftsboksen og menuboksen er fælles for alle sider (dog ikke login siden), og vises derfor kun på ovenstående figur.

package BaRI [Søg]

Søg

Produkt
BaRI
Oversigt

Type
☒ Ønsker
☒ Fejl

Sags status
☒ Oprettet
☒ Behandlse
☒ Godkendt til udv.
☒ Afvist
☒ Godkendt
☒ Lukket

Udviklings status
☒ Ikke påbegyndt
☒ Begyndt
☒ Klar til test
☒ Testet
☒ Fejl fundet

Søg efter tekst eller ID

Søg


ID	Type	Overskrift	Oprettet	Rettet	Sagsbehan...	Udviklings...	Stemmer	Prioritet

Side 1/2 Antal: 23

Forrige Næste

Figur 15 - Søg

Søge siden bruges til at fremsøge ønsker og fejl. De enkelte kolonner kan sorteres ved at trykke på kolonne overskriften. På den måde kan man finde sager med flest stemmer. Feltet "Søg efter tekst eller ID" bruges til af finde sager med, som har den en given tekst i de forskellige tekstfelter, som er i ønsker, fejl og diskussionsindlæg. Her kan man også skrive et sags ID, så findes sagen med det givne sags ID, samt sager som henviser til dette.

package BaRI [ Opret ønske]

Opret ønske/fejl

Produkt

Modul

Overskrift

Prioritet

☐ E-mail notification

☐ Synlig for slutbruger

Beskrivelse

[Vedhæft filer](#)

Filnavn	Beskrivelse

Figur 16 - Opret ønske eller fejl

Opret ønske- eller fejlsiden adskiller sig, kun i hvilke prioriter der kan vælges, samt på fejl kan det også indtastes et produktversionsnummer. Linket "Vedhæft filer" er først synlig efter at sagen er gemt første gang. Dvs. at tryk på gem knappen bliver på siden.

package BaRI [Ret ønske fane 1]

Ret ønske/fejl

Ønske Diskussionsindlæg Ændringsloggen

ID: 120

Produkt: BaRI

Modul: Oversigt

Overskrift: Ny total. For alle ønsker og fejl tilsammen.

Beskrivelse: Beskrivelse ..

Oprettet: 20/5-2011 af Tanja Kikkenborg Sidst rettet: 21/5-2011 af Tanja Kikkenborg

Stemmer: 20 ☐ Min stemme

Estimat: 3

Prioritet: Skal udvikles

Sagsbehandlerstatus: Godkendt

Udviklingsstatus: Klar til Test

Konklusion:

☐ Synlig for slutbruger

☐ E-mail notifikation

Ansvarlig udvikler: Jan Schrøder Hansen Tester: Jan Schrøder Hansen

[Vedhæft filer](#)

Filnavn	Beskrivelse	Gemt den	_Slet_	_Vis_

Figur 17 - Ret ønske eller fejl – første fane

For hver fil der er vedhæftet, er der to links. Et slet link, som viser en "er du sikker dialog" før der slettes. Og et vis link, som henter filen. Browseren giver mulighed for at vise filen hvis den kender formatet, eller at gemme filen lokalt. Hvis brugeren har rollen administrator, så ville der også være en "Slet" knap på denne side. Hvis det er en fejl så kan der også indtastes et produktversionsnummer.

package BaRI [Ret ønske fane 2]

Ret ønske/fejl

Diskussionsindlæg

20/1-2011 kl 12:00 af Tanja Kikkenborg
 Jeg mener at der bør være en ny knap.
 _filnavn_beskrivelse_
 _filnavn_beskrivelse_
 21/1-2011 kl. 12:00 af Jan Schrøder Hansen
 Enig
 Ret

Nyt indlæg

Text

☐ Synlig for slutbruger Gem

[Vedhæft filer](#)

Filnavn	Beskrivelse	Gemt den	_Slet_	_Vis_

Figur 18 - Ret ønske eller fejl – anden fane

Den store tekstboks i toppen indeholder alle tidligere indlæg, i dato/tid rækkefølge. Hvert indlæg skrives med dato/tid og bruger. Selve indlægget kommer på en ny indrykket linje. Hvis der er vedhæftet filer, vises disse som links med filnavn og beskrivelse. På det sidste indlæg er der et "Ret" link, hvis den aktuelle bruger er den samme, som har lavet indlægget. Hvis "Ret" linket vælges kommer teksten og evt. filer ned i "Nyt indlæg" delen. Når et nyt indlæg gemmes, kan teksten ses med det samme i den store tekstboks i toppen.

package BaRI [Ret ønske fane 3]

Ret ønske/fejl

Ændringsloggen

Dato/tid	Bruger	Brugerrolle	Handling	Type	Overskrift

Side 1/2 Antal: 23

Forrige Næste

Beskrivelse

Beskrivelse af ændringen.

Figur 19 - Ret ønske eller fejl – tredje fane

I toppen af ændringsloggen er der en tabel med alle ændringer til en fejl eller et ønske. Hver linje i tabellen er et link. Hvis der trykkes på linket vises selve rettelsen i "Beskrivelse" tekstboksen. F. eks. hvis en status er rettet, skrives både førværdien og efterværdien.

F. eks. "Udviklingsstatus er rettet fra "Klar til test" til "Testet"

8.11. INDHOLD PÅ VEDLAGTE CD

Indholdet på den vedlagte CD er inddelt i følgende 4 kataloger:

- Rapport – Indeholder denne rapport i Word 2007 og PDF format.
- Databasesystemer og WEB – Indeholder rapporten til faget Databasesystemer og WEB i PDF format.
- WEB og Serverprogrammering – Indeholder rapporten til faget WEB og Serverprogrammering i PDF format.
- MagicDraw – Indeholder 3 MagicDraw projekter. Et for analyse, et for Design og implementering samt et for websider.