2002:DS11

EXAMENSARBETE

Utveckling av webbapplikation för projektet Odin
Med hjälp av objektorienterad systemmodellering

Anna-Karin Carlsson
Anna Johansson

2002-05-23

Högskolan Trollhättan/Uddevalla
Institutionen för informatik & matematik
Box 957, 461 29 Trollhättan
Tele: 0520-47 53 30 Fax: 0520-47 53 99

EXAMENSARBETE

Utveckling av webbapplikation för projektet Odin
Med hjälp av objektorienterad systemmodellering

Sammanfattning

Odin är ett projekt inom radio- och rymdastronomi som startade år 1994. Projektet är ett samarbete mellan Sverige, Finland, Frankrike och Kanada och bedrivs i Sverige bl.a. vid Onsala Rymdobservatorium utanför Göteborg. Forskarna inom Odin har problem med att mycket redundant arbete förekommer och att diskussioner som framkommer blir långdragna och ineffektiva på grund av att de sitter i olika delar av världen och kommunikationen sker via e-post. Ytterligare ett problem inom projektet är att fel som upptäckts vid en mätning sällan kommer till andras kännedom än den som gör upptäckten. Vi fick med denna bakgrundsfakta i uppdrag att tillverka en webbapplikation som löste dessa problem inom projektet Odin.

För att komma fram till en lösning av problematiken har vi genomfört en objektorienterad systemmodellering med hjälp av The Unified Modeling Language (UML) notationen och sedan utifrån modelleringen konstruerat en webbapplikation med hjälp av skriptspråket PHP och en PostgreSQL databas.

Vår webbapplikation är en delapplikation till projektets redan befintliga hemsida och delades upp i tre olika delar. Innehållet i delarna är: ett diskussionsforum som underlättar diskussionerna inom projektet, en samordningslista där aktörerna registrerar sig på de källor som de arbetar med samt en felrapporteringslista där man kan skriva in de fel som framkommer under mätningens gång.

Nyckelord: Systemmodellering, UML, PHP, PostgreSQL

Utgivare: Högskolan Trollhättan/Uddevalla, Institutionen för Informatik & Matematik
Box 957, 461 29 Trollhättan
Tele: 0520-47 53 30 Fax: 0520-47 53 99

Författare: Anna-Karin Carlsson & Anna Johansson

Examinator: Docent och universitetslektor Stefan Mankefors

Handledare: Ingemar Hjorth, HTU & Henrik Olofsson, Rymdobservatoriet

Poäng: 10 **Nivå:** C

Huvudämne: Datavetenskap **Inriktning:** Programmering

Språk: Svenska **Nummer**: 2002:DS11 **Datum**: 2002-05-23

THESIS

Web Application Development for the Odin Project
With Object Oriented System Modelling

Summary

The year of 1994 the project of Odin was grounded. This is a project within radio- and space astronomy and collaboration between Sweden, Finland, France and Canada. In Sweden you can find the project at the Onsala space observatory right outside Gothenburg. The Odin scientists have problems with redundant work, that the discussions become tedious and ineffective and also that occurring errors rarely comes other scientists to knowledge. With these facts as background we got the assignment to make a web application that could solve these problems within the Odin project.

To come up with a solution to these problems we have accomplished an object-oriented system modelling with The Unified Modeling Language (UML) notation. After this modelling we have constructed a web application with the script language PHP and a PostgreSQL database.

Our application is a part of the projects already existing application and it is divided into three parts. The contents in the parts are: a discussion forum that makes the discussion within the project easier, a coordinate list where the actors can register on a source that they are working with ant last but not least a error list where the actors can rapport errors that have occurred under a measuring.

Keywords: System modelling, UML, PHP, PostgreSQL

Publisher: University of Trollhättan/Uddevalla, Department of informatics and mathematic
Box 957, S-461 29 Trollhättan
Phone: 0520-47 53 30 Fax: 0520-47 53 99

Author: Anna-Karin Carlsson & Anna Johansson

Examiner: Docent and university lecturer Stefan Mankefors

Advisor: Ingemar Hjorth, HTU & Henrik Olofsson, Observatory

Subject: Computer science

Language: Swedish **Number**: 2002:DS11 **Date**: May 23, 2002

Förord

I kapitel 1 och 2 har planeringsrapporten som vi gjorde i kursen system- och vetenskapsteori, varit utgångspunkten som vi har vidareutvecklat.

Det arbete som projektet har medfört har utförts av bägge gruppmedlemmarna tillsammans men med olika ansvarsområden. Anna har ansvarat för rapporten och Anna-Karin för programmeringen.

Vid vårt examensarbete har vi kommit i kontakt med personer som vi vill tillägna ett stort tack för den hjälp och det stöd vi fått av dem. Det är Mathias Fredrixon, systemansvarig vid Chalmers som ordnade med kontakten mellan Onsala Rymdobservatorium och oss. Vår handledare på observatoriet Henrik Olofsson (radioastronom) som har tagit sig tid att utforma applikationen tillsammans med oss samt Richard Torkar, doktorand på HTU som har hjälpt oss med den tekniska utrustningen och programmeringen.

Innehållsförteckning

Symbolförteckning

Source	Astronomiska objekt i solsystemet, vintergatan eller i andra galaxer som avger någon form av elektromagnetisk strålning. Satelliten Odin studerar m.h.a. strålning på våglängder 1000 ggr längre än vanligt ljus, i huvudsak källor som innehåller molekylgas t.ex. interstellära molekylmoln/stjärnbildningsområden. Ofta hör källorna ihop med och har namn efter objekt som först observerats på andra våglängder t.ex. stjärnor i det optiska området (Olofsson, 2002).
Line	Spektrallinjestrålning hos molekylerna åstadkommes genom att de strålar på en viss bestämd våglängd, denna lyckosamma omständighet gör att man kan bestämma vilka linjer som hör till vilken molekyl. Att det är så beror på att energinivåerna i molekylerna bara får anta vissa bestämda värden, dessa kan förstås i en kvantmekanisk beskrivning av materien. Eftersom samma molekyl kan stråla på flera våglängder anger man ibland vilken linje man menar genom att annotera de två energitillstånd som är inblandade vid avgivning av strålning, t.ex. CO J=2-1, utläses "kolmonoxid J-är-lika-med-två-till-ett" (Olofsson, 2002).
Type of error	Vad som är problemet med data, t.ex. - data saknas - data verkar opålitligt pga. ... - data är i princip bra, men har fått felaktig etikett, etc.
Concerning	Vad som är drabbat av felet ovan. Eftersom man simultant tar emot data från upp till fem olika instrument på en gång, är det viktigt att man anger vilken eller vilka som är drabbade. Man bör också ange under vilken tid detta var aktuellt, helst med hjälp av banvarvsnummer (Olofsson, 2002).
Artefakt	Konstgjort föremål

1 Inledning

År 1949 startade Onsala Rymdobservatorium sin verksamhet och blev år 1990 en nationell forskningsanläggning för radioastronomi. På Rymdobservatoriet bedrivs forskning både för den nationella och för den internationella världen (URL C). Det som observeras är radiostrålning från t.ex. molekyler i stora gasmoln ute i rymden. De kan med hjälp av dessa observationer bestämma vilka molekyltyper som finns, dess täthet samt avstånd och hastighet. Det är dock inte bara täthet och avstånd som observeras utan även gasmolnens temperatur och magnetfält. Allt det som undersöks är delar av ett pussel som ska hjälpa till med förståelsen av strukturen och utvecklingen av universum som t.ex. stjärnbildning och många andra av universums gåtor (URL E).

Odin som är en förhållandevis liten satellit är den sjätte svenska högteknologiska forskningssatelliten som finns i rymden. Satelliten har instrument som kan riktas både mot rymden och mot vår egen planet. Projektet Odin startades år 1994 och är ett samarbete mellan Sverige, Finland, Frankrike och Kanada. Grunden för samarbetet är det gemensamma intresset för forskningsområdena atmosfärforskning och astronomi. Namnet på projektet kommer efter satelliten Odin som har två uppdrag – att studera atmosfären och att undersöka rymden. De mätinstrument som används ska hjälpa forskarna att identifiera de molekyltyper som finns, samt svara på hur mycket av olika ämnen som finns inom de områden Odin riktas mot. Odinprojektet är inte bara framstående inom forskning utan också en viktig brygga till framtida internationella samarbeten för både industri och forskare (Rymdstyrelsen, 2001).

1.1 Bakgrund

Forskarna inom projektet Odin befinner sig i olika delar av världen och träffas bara ett fåtal gånger per år vid konferenser. Detta medför att all kommunikation och alla diskussioner inom projektet sker via e-post. Detta är ett dilemma för forskarna då diskussionerna tar lång tid och blir osystematiska då man får söka reda på vad som tidigare sagts i ”inkorgen” i e-postprogrammet. Felskrivningar och att information inte når fram till alla berörda på grund av den mänskliga faktorn medför konsekvenser för det fortsatta arbetet.

Inom Odin där forskarna är fokuserade på att arbete med radioastronomi är det vanligt att flera arbetar med samma källa. Detta har till följd att redundant arbete förekommer i stor utsträckning. Det finns ingen nuvarande rutin för att stämma av vad man arbetar med så att man kan ta hjälp av varandra, utan många utför själva den analys som krävs för forskningen. Detta trots att någon annan redan kan ha utfört liknande bearbetning.

I dagsläget så är all dokumentation av observationer och felupptäckter frivilliga. Om en dokumentering görs så lagras den på en privat hårddisk eller i pappersformat i en pärm.

Dessa ovanstående återkommande problem som ständigt dyker upp i forskarnas dagliga arbete har länge varit ett hinder för effektiv forskning. Man har vetat om problemen men varken haft tid, kunskap eller resurser att lösa dem. Funderingar som funnits är att man borde

utnyttja den redan befintliga hemsidan till Odin och utöka dess funktionalitet så att den kan lösa problematiken. Det var här vi kom in i bilden. Henrik Olofsson, radioastronom och forskarstuderande inom projektet Odin, bad oss utforma en webbapplikation som kunde kopplas till deras redan befintliga webbsida.

1.2 Problemformulering

Hur kan vi skapa en webbapplikation som underlättar och effektiviserar det dagliga arbetet hos forskarna som ingår i projektet Odin, dvs. minska det redundanta arbete och underlätta aktörernas kommunikation med varandra?

1.3 Syfte och mål

Syftet med arbetet är att göra en webbapplikation som ska lösa kommunikationsproblematiken och ge samordningsrutiner för att undvika redundant arbete samt införa rutiner för felrapportering. Vi kommer att koncentrera oss på att göra en användarvänlig webbapplikation då datorvanan hos en del av de berörda aktörerna är relativt låg.

För att lösa kommunikationsproblemet kommer vi att skapa ett diskussionsforum där aktörerna ska kunna skapa egna diskussioner som andra aktörer ska kunna göra inlägg på. Inläggen kommer att följa diskussionen som en röd tråd allteftersom de inkommer. För att man ska slippa gå in och kolla på forumet varje gång man vill se om ett nytt inlägg har gjorts, kan man registrera sig på att prenumerera på en viss diskussion. Forumet kommer då att generera ett e-postmeddelande till prenumeranterna en gång per dygn om nya inlägg har gjorts på den angivna diskussionen.

Det redundanta arbetet kommer att lösas med en samordningsfunktion där man kan registrera sig på vilken källa man arbetar med just nu. Källorna kommer att vara fördefinierade, men aktörerna ska själva kunna lägga till en källa när den uppstår. Registreringen innebär inte att man har ensamrätt på källan utan flera forskare kan vara registrerade på samma källa. Funktionen är till för att få en överblick över vem som arbetar med vad och för att få reda på vem man ska vända sig till om man vill ta del av en mätning.

En felrapporteringsfunktion kommer att utformas för att aktörerna ska kunna lägga in sina felaktiga upptäckter på de observationer som har utförts. Felen kommer att skrivas in av aktörerna efter en mall som är fördefinierad. Alla aktörer ska kunna ta del av sina egna och alla andras felupptäckter men man ska endast kunna radera sina egna inrapporteringar. En sökfunktion som gör att man kan söka reda på ett specifikt fel kommer att finnas för att underlätta för aktörerna.

1.4 Avgränsningar

Vår applikation kommer att bli en delapplikation av Odins befintliga webbapplikation. Vi behöver därmed inte tillhandahålla någon inloggningsfunktion till vår del eftersom det redan finns en tillgänglig på den befintliga applikationen.

Fokusering sker enbart på hur problemområdet ser ut idag och hur det ska lösas på bästa sätt för de berörda aktörerna. Vi tar alltså inte hänsyn till någon annan del i organisationen än den som problematiken berör.

Vi kommer inte heller att ha någon fortsatt support efter implementeringen av applikationen eftersom vårt arbete då är slutfört.

2 Metodik

För att kunna utföra ett vetenskapligt arbete på ett seriöst sätt är det viktigt att man använder sig av metoder som gör att man systematiskt och planmässigt kan lösa ett problem och komma fram till en lösning. Vi har valt följande för att uppnå ett optimalt resultat till vår webbapplikation.

2.1 Metodval

Den kvalitativa metoden har som mål att komma nära aktörerna och få en förståelse över deras tankar som kan bidra till att man kommer fram till en lösning på ett givet problem. Metoden kräver att man är öppen och flexibel gentemot data och försöker se till helheten. Insamlad data är inte konstant utan kan ändras under tidens gång (URL D). Informationen kan dock inte omvandlas till siffror och det är forskarens uppfattning och tolkning av problematiken som står i förgrunden. Representativitet är inte viktigt vid denna metod och man väljer som regel alltid att fokusera sig på några få enheter. Observationer utförs osystematiskt och ostrukturerat när man försöker sätta sig in i aktörernas roller. Rapporten består mestadels av deskriptiva redogörelser (Wallén, 2000).

Vid användning av den kvantitativa metoden har man redan en uppfattning om vilka resultat man kommer att nå genom statistiska generaliseringar. Statistiska analyser utförs för att informationen skall kunna omvandlas till mängder och siffror. Målet är att bevisa varför man kommit fram till den specifika lösningen på problemet. Oftast har man en given forskarplan där man redan har avlägsnat alla eventuella felkällor som kan uppstå och inriktar sig på en företeelse istället för att se helheten (URL D).

Vi valde bort den kvantitativa metoden på grund av att man där tar avstånd från aktörerna och har en jag-det-relation till undersökningsområdet. Observationerna sker här utifrån, vilket inte skulle ge oss någon fördel då vår uppgift kräver ett nära samarbete med aktörerna för att kunna utforma en webbapplikation som löser deras vardagsproblematik (Wallén, 2000).

För att kunna utföra uppgiften på ett tillfredsställande sätt har vi arbetat efter den kvalitativa metoden. Metoden har inneburit att vi har haft ett nära samarbete med vår handledare Henrik

Olofsson på observatoriet. Problemområdet har iakttagits inifrån organisationen för att få en djupare kunskap och förståelse gentemot problematiken.

2.2 Metodansats

Vi har använt oss av den induktiva ansatsen i vårt arbete när vi förutsättningslöst har samlat in information om användarvänlighet, skriptspråket och databasen som har varit nödvändig för att kunna utföra arbetet. Materialet har vi använt oss av för att få en djupare kunskap och förståelse inom det problemområde som är relevant för vår uppgift. Vi har även tagit hjälp av den induktiva ansatsen när vi observerat verkligheten i problemområdet för att kunna dra generella och teoretiska slutsatser om vad uppgiften som har utförts har omfattat (Wallén, 2000).

2.3 Metodstudie

Innan vi påbörjade studien valde vi mellan normativ och deskriptiv studie, men efter närmare granskning valde vi den deskriptiva på grund av att vårt arbete kommer att gestalta sig i form av en webbapplikation som kräver systemmodellering som förarbete. Förarbetet kommer att ligga som underlag för att underlätta programmeringen av applikationen. Den normativa studien innebär att forskarens uppgift är att visa på olika ståndpunkter och att ta fram handlingsförslag samt dess konsekvenser. Den passade inte vårat syfte då vi har givna riktlinjer att följa och slutprodukten kommer bara att finnas i en version.

2.4 Utvecklingsmodeller

Det finns ett antal utvecklingsmodeller som man använder vid utveckling av system. Det är vattenfallsmodellen, evolutionär utveckling, formell systemutveckling, återanvändorienterad utveckling, inkrementell utveckling och spiralmodellen (Sommerville, 2001).

Vattenfallsmodellen (*The waterfall model*) innebär att man gör ett steg färdigt innan man går vidare till det nästa. Stegen är kravspecifikation, system- och mjukvarudesign, implementation och testning, integration och systemtestning samt drift och underhåll.

Den *evolutionära* utvecklingen (*Evolutionary development*) baseras på idéen av att utveckla ett första utförande. Detta utförande ska sedan visas för användarna ett flertal gånger för att man som utvecklare ska få kommentarer så att det är det rätta systemet som utvecklas.

Formell utveckling (*Formal system development*) närmar sig mjukvaruutveckling och har en del gemensamt med vattenfallsmodellen. Men här är utvecklingsprocessen baserad på formella matematiska förändringar av systemspecifikationen till ett exekverbart program.

Vid många projekt använder man sig av *återanvändorienterad* utveckling (*Reuse-oriented development*). Detta händer informellt oftast när projektdeltagarna redan har gjort design eller kod som liknar det som nu ska utvecklas. De tittar på det som tidigare gjorts och modifierar det innan det integreras i det nya systemet.

Den *inkrementella utvecklingen* (*Incremental development*) innebär att kunderna till ett system kan försena beslut om deras krav tills de har lite erfarenhet av det kommande systemet. De kan då efter sin egna lilla erfarenhet bestämma vilka funktioner som är mer eller mindre viktiga.

Den sista utvecklingsmodellen är spiralmodellen. Processen är i denna modell representerad som en spiral, där varje loop i spiralen representerar en fas i utvecklingen. Varje loop är indelad i fyra sektioner och de är: *objective setting*, *risk assessment and reduction*, *development and validation* och *planning*. Den största skillnaden mellan spiralmodellen och de andra är att man i den här tar hänsyn till vilka risker som kan uppstå (Sommerville, 2001).

Vi valde att använda oss av vattenfallsmodellen vid utvecklandet av webbapplikationen. Faktorerna som gjorde att vi valde just Vattenfallsmodellen är att den är anpassad att användas i sammanhang där det finns tydliga och välformulerade krav som systemet ska uppfylla. Vårt system är relativt litet och det fanns redan från början tydliga riktlinjer av vad applikationen ska innehålla.

Modellen omfattar ett antal faser där varje fas resulterar i ett eller flera dokument som måste godkännas och beprövas av uppdragsgivaren innan man kan påbörja nästa fas. En fas kan inte påbörjas förrän föregående fas har avslutats men i praktiken så överlappar faserna varandra och ger därmed information som är relevant för systemet från föregående fast till nästföljande fas (Sommerville, 2001). Arbetssättet som modellen bygger på passar vårt arbete pga. att vi kommer att utföra arbetet stegvis och hela tiden ha ett nära samarbete med uppdragsgivarna så att systemet i slutändan kommer att tillgodose deras behov.

Problem med vattenfallsmodellen är att den ger en oföränderlig uppdelning av projektet genom dess tydliga faser. Åtaganden måste göras i ett tidigt skeende i processen och det innebär att det är svårt att vara flexibel och svara för ändrade förhållanden och behov som framkommer under utvecklingsprocessen (Sommerville, 2001). Dessa problem undgår oss eftersom vår uppdragsgivare tydligt har deklarerat vad de vill ha och en eventuell vidareutveckling av systemet ligger inte i vårt åtagande. Webbapplikationen som ska framställas är en utökning av Rymdobservatoriets redan befintliga applikation.

2.5 Tillvägagångssätt

Vår uppgift kommer att utmynna i en webbapplikation som kommer att implementeras på Odins webbserver. Uppgiften påbörjades efter första mötet med vår handledare Henrik Olofsson och andra inblandade aktörer inom projektet. Efter att vi observerat och samlat in tillräckligt mycket bakgrundsinformation om problematiken tog vi hjälp av The Unified Modeling Language (UML) för att göra en objektorienterad systemmodellering. Modelleringen har använts för att analysera och designa de problem som måste lösas inom Odin för att få en fungerande kommunikation. Ett nära samarbete med vår handledare har varit nödvändig under denna fas då vi kontrollerat att vi tolkat deras situation och målsättning på rätt sätt. När förarbetet är klart påbörjade vi programmeringen av applikationen. Applikationen

kommer tillsammans med Rymdobservatoriets systemansvarige och med vår hjälp att implementeras på projektet Odins webbserver.

2.6 Litteraturstudie

Litteratur- och referensökning har tillämpats för att få en djup och objektiv bakgrund till hur man gör UML-analyser, hur skriptspråket är uppbyggt, databasens utformning och hur man utformar en användarvänlig webbapplikation.

3 Systemmodellering

För att få en överblick över hur webbapplikationen ska utformas har vi använt oss av objektorienterad systemmodellering med hjälp av notationen UML. Som främsta underlag till UML-notationen har vi använt oss av Mathiassen, Munk-Madsen, Nielsen och Stages bok Objektorienterad analys och design (2001).

3.1 UML

The Unified Modeling Language (UML) är ett grafiskt språk för att visualisera, specificera, konstruera, och dokumentera artefakter som är nödvändiga i ett programvaruintensivt system. UML är sedan år 1997 ett standardiserat objektorienterat modelleringsspråk som ger en standard för systemets ritningar och behandlar verksamhetsprocesser och systemfunktioner. Språket behandlar även konkreta saker som klasser i programspråket, databasscheman och återanvändbara komponenter (Booch, Rumbaugh & Jacobson, 1998).

UML används i programmeringssammanhang för att skapa en överblick över systemet och öka förståelsen över realiseringsvillkoren. Man utför en analys och design av ett kommande system för att få en insikt över hur systemet som slutprodukt kommer att gestalta sig. För att komma ihåg alla olika resultat som framkommer så använder man sig av en notation (Mathiassen, Munk-Madsen, Nielsen & Stage, 2001).

En notation innehåller symboler som används i modelleringen och ett antal regler som reglerar språket. Det finns tre olika typer av regler och de är: *syntaxisk*, *semantisk* och *pragmatisk*. De syntaxiska reglerna bestämmer hur symbolerna ska se ut och hur de kan kombineras med varandra. De semantiska reglerna berättar vad varje symbol betyder och hur de ska tolkas var och en för sig själva eller i kombination med en eller flera andra symboler. De pragmatiska reglerna förklarar hur vi ska använda språket (Eriksson & Penker, 2000).

Enhetlighet och generalitet är viktiga faktorer i UML. Tankesättet man använder vid objektorientering omfattar många viktiga begrepp och varje begrepp måste representeras av en specifik symbol som går att särskilja från andra begrepp i ett grafiskt språk. Symboler som representerar ett begrepp är därmed ett viktigt inslag i UML-notationen. Man behöver flera symboler för att skilja exempelvis ett objekt och en klass och för att särskilja dem har man infört några generella begrepp som inte är objektorienterade i sig själva men är knutna till notationen. Dessa begrepp är *element*, *stereotyp* och *paket*. Element är en beskrivning av en

odelbar enhet. Element används för att representera en viss egenskap hos det beskrivna fenomenet. Beskrivningen behöver ej vara fullständig varje gång den förekommer. Stereotyp används för att utvidga UML och gör det möjligt att addera ytterligare innebörder till ett UML-element. Dessa innebörder kan användas av en läsare eller av ett datorbaserat utvecklingsverktyg. Ett paket är ett element som kan bestå av flera andra paket och kan innehålla både beskrivningselement och diagram. Ett paket utgör en självständig och avgränsad helhet. Elementet inom ett paket måste därför ha olika namn. Det betyder att element som definieras i ett paket inte är direkt synliga i andra paket. För att de ska bli synliga måste en referens finnas (Mathiassen et al., 2001).

Att använda UML ger framförallt två fördelar, det är distinktionen mellan processer och notation dvs. inga inbyggda processlinjer att ta hänsyn till och tillgången till en stor marknad av UML-kompatibla utvecklingsverktyg tack vare deras breda användarbas. UML täcker även notationsbehoven i en objektorienterad utvecklingsprocess, från preliminär analys till en detaljerad designbeskrivning som kan utgöra grundval för automatisk generering av delar av programkoden. UML omfattar också ett stort antal objektorienterade konstruktioner av vilka många bara är relevanta i vissa situationer (Kruchten, 2002)

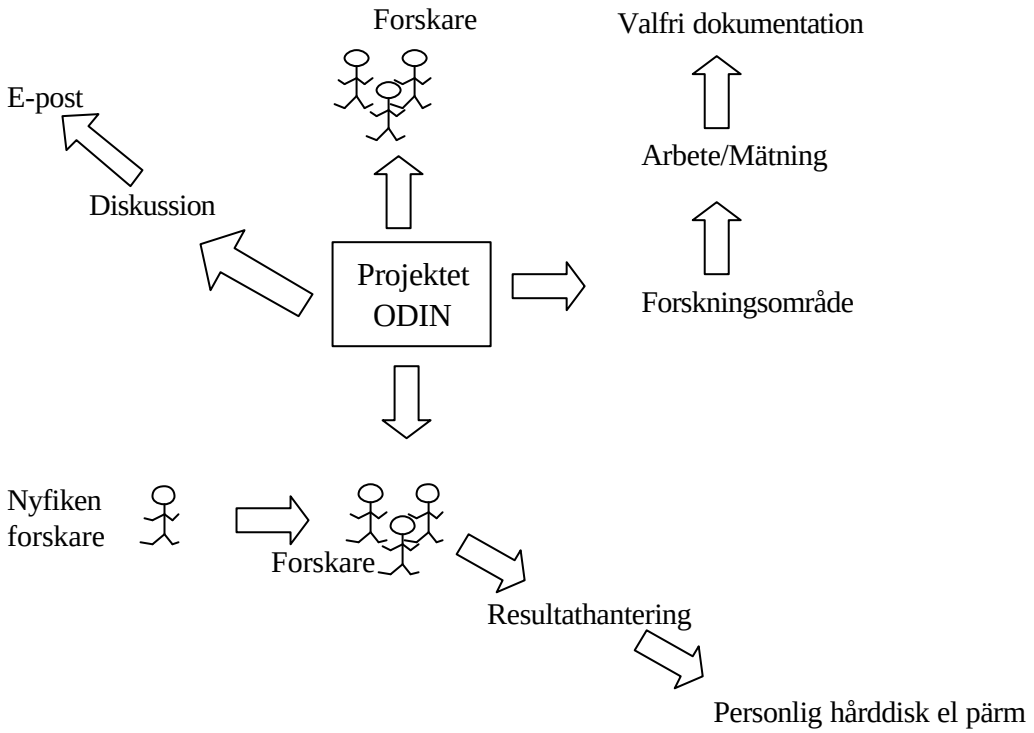
3.2 Systemval

Forskarna inom projektet Odin hade egna förslag på hur man kunde lösa deras kommunikations- och samordningsproblem. Vår uppgift blev då att vidareutveckla dessa idéer så att det resulterar i en webbapplikation.

3.2.1 Rik bild

En rik bild är en teckning utan formella symboler eller regler som beskriver helheten av en situation i en verksamhet. Bilden fokuserar på problemsituationens väsentliga omständigheter (Mathiassen et al., 2001).

För att få en förståelse över deras nuvarande situation och kunna skapa en rik bild, ”brainstormade” vi tillsammans med Michael Olberg och vår handledare Henrik Olofsson (båda forskare inom Odin). Resultatet blev följande:



Figur 1 – Rik bild

Den nuvarande situationen inom Odin är idag att alla diskussioner sker via e-post. Forskarna befinner sig i olika delar av världen och har endast ett fåtal konferenser där de möts i realtid. Detta medför att missförstånd och felskrivningar kan orsaka förvirringar som tar lång tid att reda ut.

Något som också är tidskrävande är det redundanta arbete som förekommer. Forskarna vet i stora drag vad de andra håller på med men inte i detalj. Detta kan orsaka att en och samma källa undersöks av många forskare, vilket kanske inte är nödvändigt alla gånger.

Dokumentering av felupptäckter är valfri i dagsläget och om dokumentering sker så hamnar den endast på forskarens personliga hårddisk eller på annan plats som endast den enskilde forskaren har tillgång till.

3.2.2 VATOFA-kriteriet

VATOFA-kriterierna används för att få en välformulerad systemdefinition. Kriterierna innebär att man definierar sex olika delar, med tonvikt på vad som är relevant för arbetet som ska utföras. Under kriteriet *villkor* ska man definiera de villkor som kommer att ligga till grund för systemets utveckling och användning. *Användningsområdet* är den del av organisationen som berörs av problemområdet och *teknologi* är den teknik som kommer att användas vid systemets utveckling samt den teknologi som används vid exekveringen. De *objektsystem* som omfattas av problemområdet tas upp under objekt och de funktioner som ska stödja användningsområdet preciseras under *funktionalitet*. Systemets generella *ansvar* mot

omgivningen anges under kriteriet ansvar. Kriterierna som vår webbapplikation ska uppfylla är följande (Mathiassen et al., 2001).

Villkor – Webbapplikationen ska vara till för att underlätta diskussionerna som uppkommer. Detta sker via ett forum. Redundant arbete undviks genom en samordningslista och fel som uppstår vid en mättning dokumenteras på en felrapporteringssida.

Användningsområde – Webbapplikationen kommer att användas av de medlemmar som ingår i projektet Odin.

Teknologi – Applikationen utvecklas för en webbserver som har Linux som operativsystem.

Objektsystem – Forskarna behöver ett diskussionsforum, en felrapporteringslista samt en samordningslista för att underlätta deras dagliga arbete.

Funktionalitet – Applikationen ska stödja kommunikationen mellan medlemmarna, få ett smidigt samarbete mellan användarna för att undvika redundant arbete och för att samla alla felrapporteringar som framkommer på ett ställe.

Ansvar – Kommunikationsmedium

3.3 Analys av problemområdet

Analysen ger en sammanhängande modell av systemets problemområde. Analysen ger en överblick och beskriver hur användaren kommer att se verkligheten.

3.3.1 Händelsetabell

Tabellen visar vilka klasser med tillhörande händelser som kommer att ingå i systemet. Klasserna fixerar de generella kategorier som definierar systemets element och selektionen tjänar som en första definition och avgränsning av systemet (Mathiassen et al., 2001).

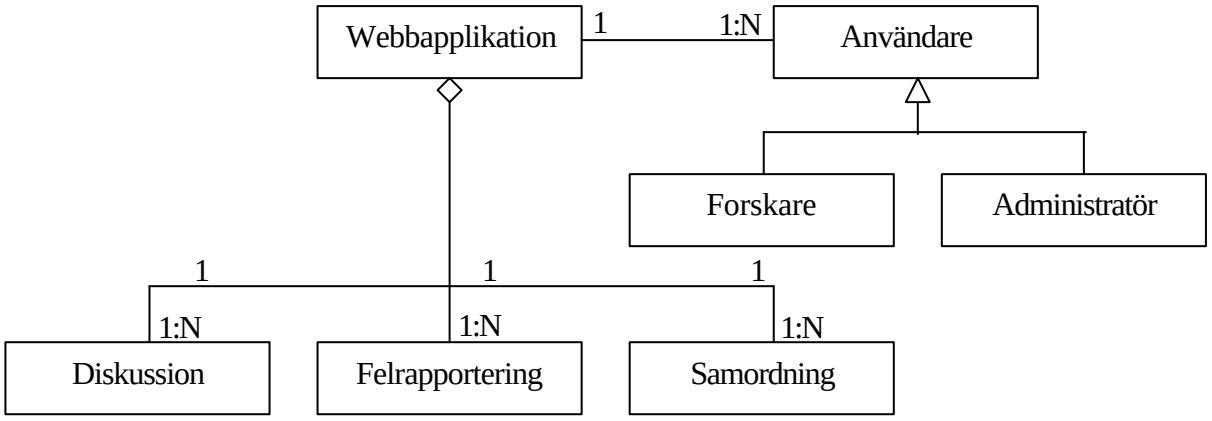
Tabell 1 - Händelsetabell

	KLASSER						
HÄNDELSER	Webb-applikation	Användare	Forskare	Administratör	Diskussion	Fel-rapport	Sam-ordning
Skapa diskussion	x	x	x		x		
Nytt inlägg	x	x	x		x		
Prenumerera på inlägg	x	x	x		x		
Lägga till källa	x	x	x				x
Boka källa	x	x	x				x
Avboka källa	x	x	x				x
Lägga till fel	x	x	x			x	
Ta bort fel	x	x	x			x	
Söka fel	x	x	x			x	
Underhåll		x		x			

Klassen *Användare* har tillgång till alla händelserna. Det finns sedan underklasser till denna och det är *Forskare* och *Administratör*. Klassen *Forskare* är kopplad till alla händelser som webbapplikationen kommer att kunna utföra förutom underhåll som klassen *Administratör* är ansvarig över. Den andra övergripande klassen är *Webbapplikation* som har tillgång till alla händelser utom underhåll. Klassen *Diskussion* är till för att lösa de händelser som uppkommer i samband med en diskussion. Man skapar ny diskussion, gör inlägg samt att man kan prenumerera på en eller flera diskussioner. Klassen *Felrapportering* tillhandahåller händelser som är relaterade till felrapporteringen och även till en sökning som effektiviserar framtagandet av ett specifikt fel. Klassen *Samordning* omfattar händelser som gör att man kan lägga till källor allteftersom de uppkommer. För att undvika redundant arbete finns boka och avboka källa.

3.3.2 Klassdiagram

Ett klassdiagram används för att visa de strukturella relationer som man återfinner mellan systemets klasser och objekt (Mathiassen et al., 2001).

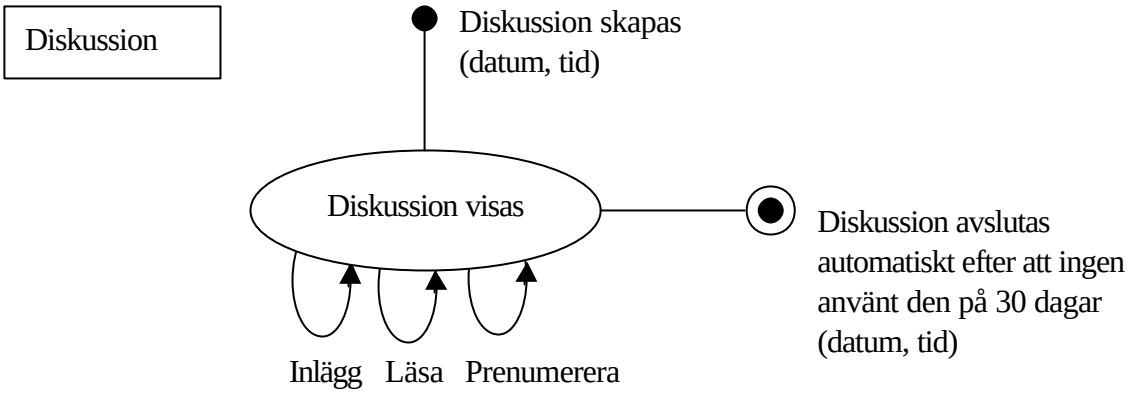


Figur 2 – Klassdiagram

Mellan de större klasserna *Webbapplikation* och *Användare* så har vi en associationsstruktur med multipliciteten 1:N (en-till-många). Klasserna *Diskussion*, *Felrapportering* och *Samordning* har en aggregatstruktur till klassen *Webbapplikation*. Detta innebär att *Webbapplikation* består av de underliggande klasserna. Klasserna *Forskare* och *Administratör* har en generaliseringsstruktur till klassen *Användare*. Detta medför att dessa klasser ärver egenskaper från den ovanliggande.

3.3.3 Tillståndsdiagram

Dynamiken i objektens beteende beskrivs genom ett diagram som förklarar varje klass beteendemönster (Mathiassen et al., 2001). Vi fokuserade på problemområdet när vi analyserade fram objektets beteende och resultatet för klassen *Diskussion* blev följande. Webbapplikationens övriga tillståndsdiagram finns i bilaga A.



Figur 3 – Tillståndsdiagram

Klassen *Diskussion* påbörjar sitt tillstånd genom att en forskare påbörjar en diskussion. Diskussionen blir iakttagbar av alla som har behörighet att komma in på sidan. Iterationerna som kan utföras av klassen är att man kan göra inlägg på befintlig diskussion, man kan läsa alla inlägg som görs samt att man kan prenumerera på den diskussion man är intresserad av. Om

ingen har utfört någon av iterationerna på trettio dagar kommer tillståndet automatiskt att avslutas och diskussionen kommer att tas bort från sidan.

3.4 Analys av användningsområdet

Analysen bestämmer systemets användningskrav och ger en fullständig lista över kraven. Man måste ta hänsyn till vad systemet kommer att användas till och hur designen kommer att utformas.

3.4.1 Aktörtabell

En aktörstabell ger en ökad förståelse över systemets användning. Tabellen ger en överblick över de mönster och aktörer som omfattas av systemet och hur de är relaterade till varandra (Mathiassen et al., 2001).

Tabell 2 – Aktörtabell

	AKTÖRER	
ANVÄNDNINGSMÖNSTER	Forskare	Administratör
Diskussions forum	x	
Felrapportering	x	
Samordning	x	
Underhåll		x

Webbapplikationen har två typer av användare, det är *forskarna* som ingår i projektet och *administratörer*. *Forskarna* är de som använder applikationen i det dagliga arbetet och de användningsmönster som utformas till systemet är tre delar som underlättar och effektiviserar arbetet. *Administratörens* roll är att sköta servicen med underhållet dvs. uppgradering och uppdatering.

3.4.2 Funktionslista

Listan ger en fullständig specifikation över funktionerna som ingår i systemet samt vilken komplexitet varje funktion erhåller. Funktionstypen utgår från samverkan mellan systemets komponenter och dess omgivning och utgör en kategorisering. Funktionerna ligger till grund för vad systemet kommer att utföra för att underlätta det dagliga arbetet (Mathiassen et al., 2001).

Tabell 3 - Funktionslista

FUNKTION	**KOMPLEXITET**	**TYP**
Visa diskussion	Enkel	Avläsning
Skapa diskussion	Enkel	Uppdatering
Nya inlägg	Komplext	Uppdatering
Prenumerera	Ytterst komplext	Beräkning
Automatisk borttagning av diskussion	Medel	Uppdatering

Visa fel	Enkel	Avläsning
Lägga till fel	Medel	Uppdatering
Ta bort fel	Medel	Uppdatering
Sök på fel	Enkel	Avläsning
Visa källor	Enkel	Avläsning
Lägg till ny källa	Enkel	Uppdatering
Boka källa	Enkel	Uppdatering
Avboka källa	Medel	Uppdatering

Funktioner som är av typen avläsning aktiveras av det informationsbehov som användaren har i sitt dagliga arbete. Användaren kommer genom funktionen att få ta del av modellens tillstånd. Forskarna inom Odin får genom systemets avläsningsfunktioner ta del av de diskussioner och källor som är tillgängliga samt de fel som uppstått vid observationer.

Beräkningstypen består av en beräkning som omfattar information om modellens tillstånd och som aktiveras av användarnas informationsbehov i samband med en arbetsuppgift. Funktionen kommer att utföra en beräkning som kommer att presenteras för användaren. Webbapplikationen har en funktion som beräknar till vilken användare som systemet ska skicka ut information till, ifall ett nytt inlägg har tillverkats på en specifik diskussion. De som får informationen är de användare som registrerat sig på att de vill prenumerera på diskussionen.

Uppdateringstypen aktiveras av en företeelse som har ett samband med problemområdet och ger en förändring i modellens tillstånd. De funktioner som är förändringsbara är de som omfattar en händelse och som i största utsträckning har en inmatning kopplad till sig (se tabell 3).

3.4.3 Gränssnitt

Forskarna inom Odin har liten erfarenhet av datorer och systemet är utformat för att ha en anpassad användarvänlighet efter användarnas förkunskaper.

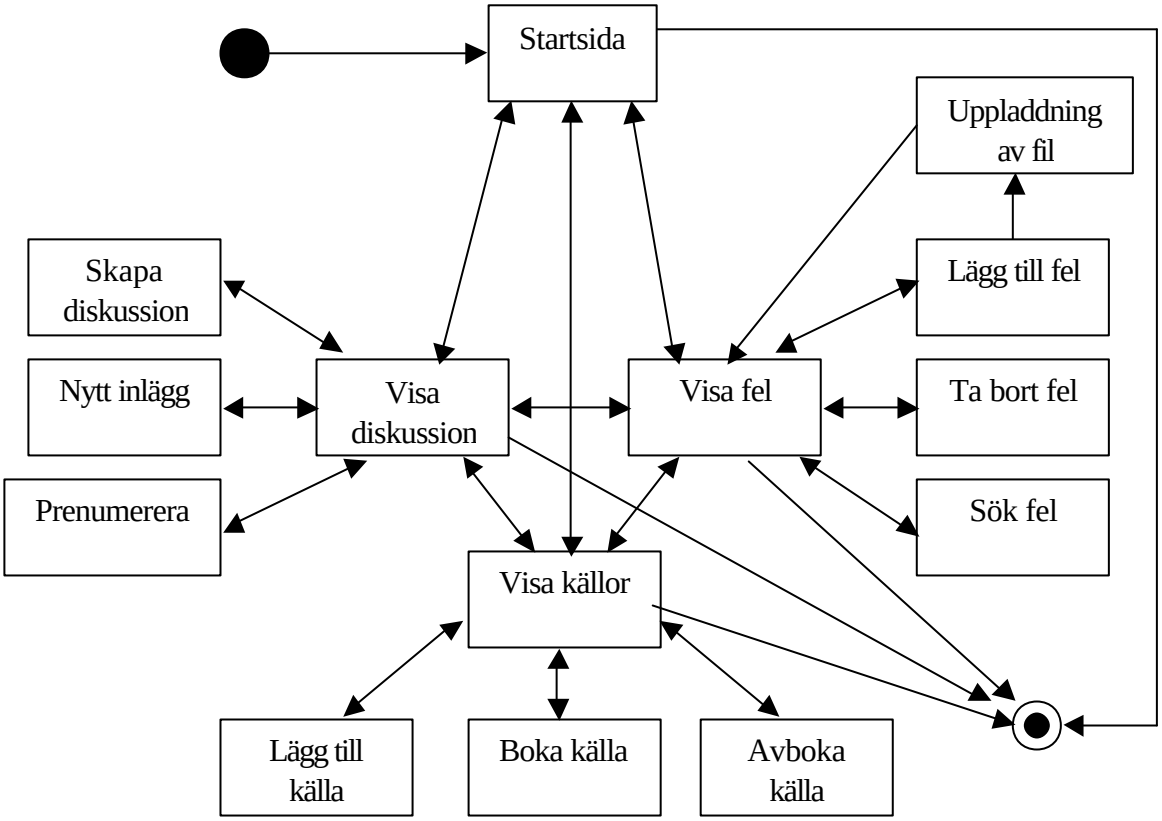
Systemet är uppbyggt kring ett menyvalsfönster vilket innebär att man förkortar inlärningstiden för användarna. Antalet knapptryckningar minskar eftersom man använder sig av länkar men det kräver effektiv skärmuppdatering och man måste tänka på att inte använda sig av alltför många menyer då det kan vara svårt att veta var i applikationen man befinner sig. De skärmbilder som webbapplikationen består av är följande:

Skärmbilder
- Start sida
- Visa diskussion
- Skapa diskussion
- Nytt inlägg
- Prenumerera
- Visa fel

- Lägga till fel
- Uppladdning av fil
- Ta bort fel
- Sök fel
- Visa källor
- Lägg till källa
- Boka källa
- Avboka källa

3.4.4 Navigeringsdiagram

Ett navigeringsdiagram gör man för att se hur sidorna i webbapplikationen är kopplade till varandra dvs. hur man navigerar i applikationen.



Figur 4 – Navigeringsdiagram

När användaren påbörjar en process i webbapplikationen utgår man från en startsida. Man kan därifrån välja fyra olika vägar att ta sig vidare – visa diskussion, fel, källor eller att avsluta. Från tre av dem kan man ta sig tillbaka till startsidan den fjärde avslutar applikationsprocessen. Visa diskussion, fel och källor har olika funktioner kopplad till sig som alla är kompatibla i bägge riktningarna samt att de kan navigera sinsemellan.

3.4.5 Fönsterbeskrivning

Fönsterbeskrivningen visar vilka skärmbilder som ingår i webbapplikationen och hur de är sammankopplade med varandra. När man har loggat in på Odinprojektets hemsida så kommer man till en sida där det kommer att finnas en länk till vår delapplikation. När man klickat på länken kommer man till vår applikations startsida, där man kan läsa lite om de tre olika delarna som ingår, dvs. vad som kan göras på respektive del. Applikationen kommer att ha en statiskt ram som innehåller en länkmeny till vänster och till höger kommer information att visas beroende på vilken del man befinner sig på. Våra fönsterbeskrivningar kan ses i bilaga B. Fönsterbeskrivningarna har även använts som prototyp och visats för vår handledare på Rymdobservatoriet för att stämma av att vi uppfyller de krav som ställts.

Figur 1 i bilaga B visar hur delen för diskussionsforumet fungerar. Om man klickar på länken till diskussionsforumet kommer alla diskussionstitlar som finns i databasen att synas i en lista och man kan klicka på en titel för att få läsa själva inlägget och för att se tråden under diskussionen. Vi har visat hur sidorna ser ut när man ska skapa en diskussion, svara på ett inlägg eller prenumerera på en diskussion.

Figur 2 i bilaga B visar hur delen för felrapporteringen fungerar. På denna sida kommer alla fel som upptäcks rapporteras in för att de andra berörda forskarna ska kunna se vilka fel som redan blivit upptäckta och om det finns någon lösning på dem. Forskarna kan här ladda upp rapporter för att man som läsare ska kunna få en mer utförlig beskrivning av felet än vad som står i formuläret på sidan. Det kommer att finnas en sökfunktion som underlättar framtagande av en specifik felrapportering.

Figur 3 i bilaga B visar hur delen för samordningen av arbeten fungerar. Sidan kommer att visa alla källor som forskarna använder sig av vid olika mätningar. Sidan är till för att minska det redundanta arbetet genom att man kan boka upp sig på en källa och på så vis tala om för de andra att man faktiskt arbetar med den. Detta innebär dock inte att man har ensamrätt till källan och att ingen annan får arbeta med den, utan är enbart till för att de ska kunna se om en mätning är gjord på en källa och utav vem i så fall. Sedan så får respektive forskare avgöra om de vill göra om arbetet med källan eller försöka få tag på mätningsresultaten från forskaren som har arbetat med källan.

3.5 Arkitekturdesign

Arkitekturdesignen utför man för att strukturera systemets komponenter och processer. Ju mer exakt man är här desto effektivare system får man i slutänden. Designen bygger på tre olika principer (Mathiassen et al., 2001).

- Definiera och prioritera kriterier, vilket innebär att man utgår från systemkraven som man kommit fram till i analysfasen samt de operationella kraven som ordnas efter prioritet.
- Ett brobygge mellan kriterier och den tekniska plattform där designkriterierna bottnar i en djup förståelse av systemets omgivning.

- Utvärdera designen tidigt för att fastställa implementeringen. Kriterierna måste förverkligas och flaskhalsar måste förebyggas. Man gör utvärderingen så tidigt som möjligt för att inte försvaga produktiviteten i arkitekturen.

3.5.1 Kvalitetskriterier

För att få en design som fungerar som en vägledning för systemet bör man välja ut ett fåtal kriterier som är väsentliga för webbapplikationen. Systemet ska vara välstrukturerat och lätt att modifiera samt att man gör en objektiv avvägning mellan flera kriterier för att få den optimala lösningen på applikationen. Inom objektorienterad analys och design finns det framförallt tre generella kriterier: användbarhet, flexibilitet och begriplighet som används vid designen (Mathiassen et al., 2001). Vi har valt att använda oss av dem samt ytterligare en del kriterium som vi anser att vårt system behöver.

Tabell 4 - Kvalitetskriterier

Kriterium	**Mått på**
Begripligt (användarvänligt)	Systemet ska vara användarvänligt vilket innebär att alla behöriga snabbt och enkelt ska kunna få en överblick över systemet
Användbarhet	Systemet ska vara anpassat och användbart i den organisation som den ska användas i.
Korrekt	Systemet ska uppfylla och utföra de krav som beställaren framfört.
Pålitlighet	Att systemet uppfyller den funktionalitet som behövs för att uppnå kraven.
Flexibelt	Kostnad för att modifiera implementerat system.

Begriplighet är en viktig faktor som man måste ta hänsyn till vid designarbetet. Man fokuserar sig på målgruppen dvs. de aktörer som kommer att använda sig av applikationen och utformar designen efter deras förkunskaper. Systemet måste även vara utformat efter den teknik som finns tillgänglig och man måste bryta isär komplexiteten hos funktionerna så att de blir överskådliga för aktörerna (Mathiassen et al., 2001).

Användbarheten ska ses som en helhet utan hänsyn till designens inre struktur. Designen ska vara utformad efter aktörernas behov och vara anpassad efter den tekniska plattformen (Mathiassen et al., 2001). Vår applikation kommer att inneha en hög användarvänlighet och vara lätt att navigera. Man ska utan större förkunskaper kunna hantera funktionerna på ett tillgodogörande sätt. Webbapplikationen utvecklas med hjälp av HTML och PHP som båda

stöds av Odins webbserver. Till lagring av data används en PostgreSQL-databas som även den stöds av webbservern.

Det är viktigt att man har gjort ett bra förarbete så att systemet lever upp till de förväntningar som aktörerna har. Ett nära samarbete med aktörerna och öppenhet mot problematiken är ett måste i vårt utförande av applikationen.

Systemet måste vara flexibelt då det är omöjligt att redan innan implementeringen ha förutspått alla konsekvenser som kan inträffa. Krav som inte varit relevanta för applikationen kan i ett senare skede visa sig vara av stor betydelse för den framtida användningen.

3.5.2 Prioriteringslista av kriterier

När man har valt ut vilka kriterier som ska ingå i systemet måste man prioritera dem efter de processer som ska dominera designaktiviteterna.

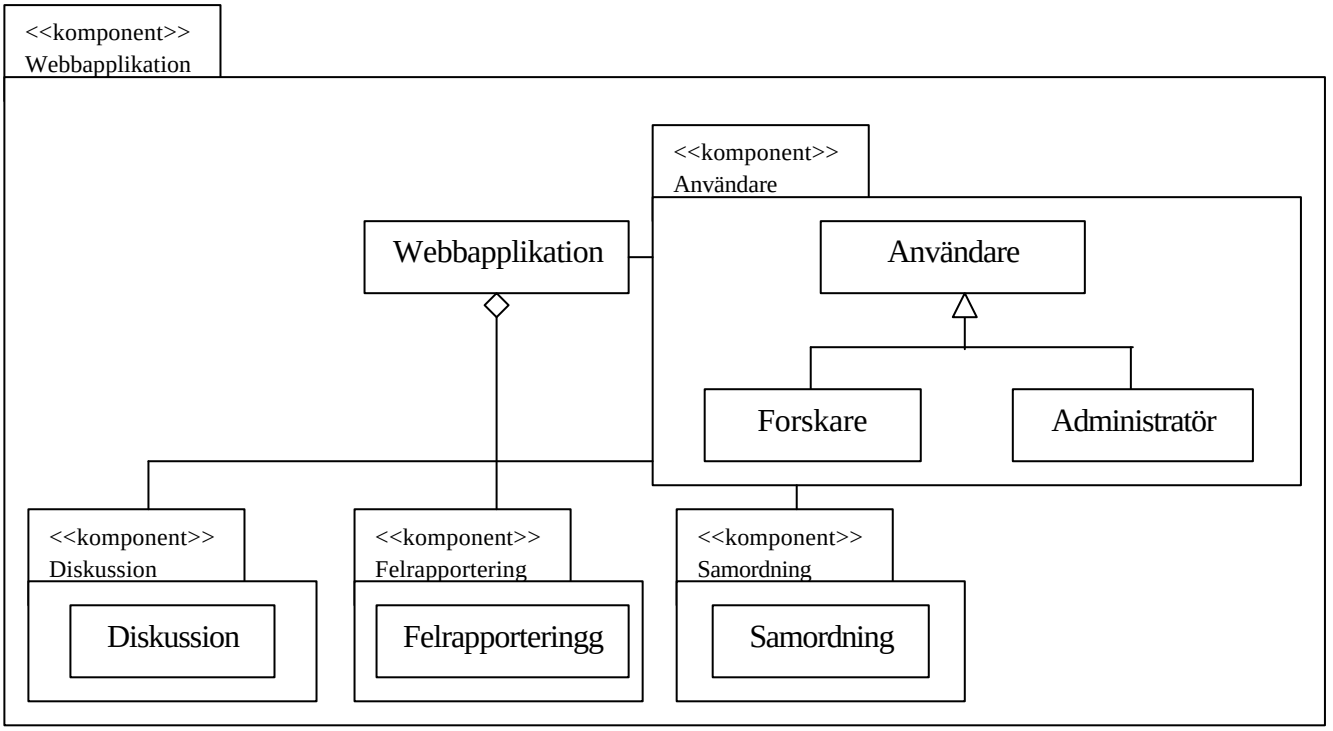
Tabell 5 - Prioriteringslista

Kriterium	Mycket viktigt	Viktigt	Mindre viktigt
Begripligt	x		
Användbarhet		x	
Korrekt	x		
Pålitlighet		x	
Flexibilitet			x

Webbapplikationens viktigaste kriterier är att den ska vara begriplig för aktörerna samt att den ska uppfylla de givna krav som angivits.

3.5.3 Komponentarkitektur

En komponent är de delar av systemet som tillsammans bildar en helhet och har ett gemensamt ansvarsområde. I komponentarkitekturen utgår man från klassdiagrammet och skapar komponenter av de klasser som har inbördes relationer. Detta gör man för att få en bättre överblick över systemets helhet (Mathiassen et al., 2001).

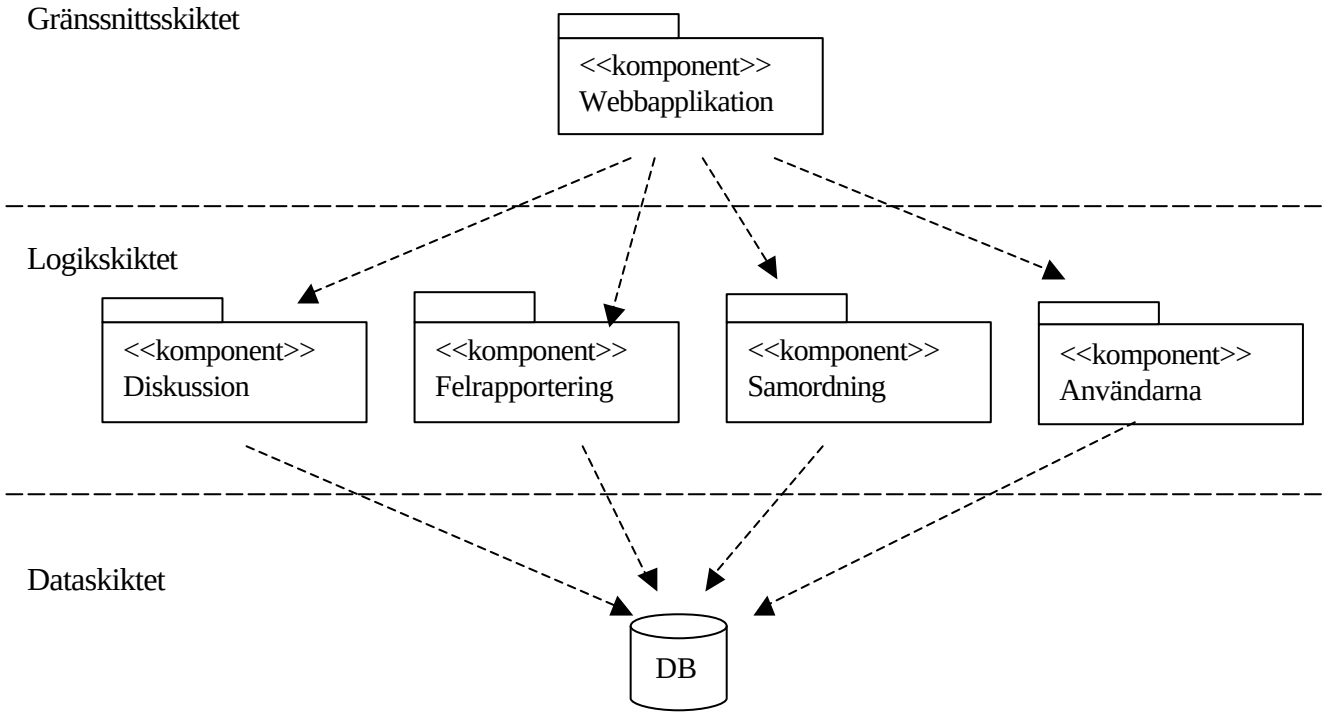


Figur 5 – Komponentarkitektur

Komponenten *Webbapplikation* är vår huvudkomponent som har relationer till alla klasser som ingår i systemet. Inuti webbapplikationskomponenten finns ytterligare fyra komponenter. Tre av dessa komponenter innehåller endast en klass vardera medan den fjärde, *Användare*, innehåller tre klasser. De tre klasserna i komponenten *Användare* har en generaliseringsstruktur mellan sig vilket medför att de har ett beroende till varandra och därför blir en större komponent.

3.5.4 Systemarkitektur

En systemarkitektur är en generell beskrivning över hur system är uppbyggda. Det finns olika sorters arkitekturer men den som lämpar sig bäst för en webbapplikation är treskiktsarkitekturen. Treskiktsarkitekturen består av komponenterna – gränssnitt, logik och data där varje komponent är en fristående del. Detta är en fördel då varje del i applikationen lätt kan förändras eller återanvändas. Arkitekturen grundas på att all data lagras i en databas och hanteras av ett Database Management System (DBMS). Med arkitekturen döljer man applikationens komplexitet för användaren men man ökar intrångsmöjligheterna genom att angriparna kan attackera från flera håll (Jonsson, 2001).



Figur 6 – Systemarkitektur

Arkitekturen visar här vilka komponenter som hör till vilket skikt. På gränssnittsskiktet har vi placerat komponenten *Webbapplikation* för att det är den som visar innehållet för användaren. De fyra komponenterna på logikskiktet är de komponenter som bearbetar data på ett eller annat sätt. På dataskiktet lagras informationen och i vårt fall är det i PostgreSQL-databasen.

3.6 Komponentdesign

När man utformar en komponentdesign gör man en beskrivning över de komponenter som ingår i systemet. Man måste även ge riktlinjer för hur komponenterna är sammankopplade. Designen bygger på två olika principer och det är att man ska respektera komponentarkitekturen och att anpassa designen efter de tekniska förutsättningar som föreligger (Mathiassen et al., 2001).

Vi har utformat ett klassdiagram med attribut och funktioner till vår webbapplikation för att visa vad varje klass innehåller (se bilaga C).

3.7 Normalisering

Normalisering är framförallt en databasdesignsteknik, men den används även vid alla steg i systemmodelleringen (Brown, 2002). Man använder sig av normalisering när man vill skapa en enkel och hanterbar databas. Detta görs genom att man ser till att relationerna i databasen uppfyller normalformerna (Apelkrans & Åbom, 2001). Enligt Brown (2002) finns det upp till 18 normalformer men de som vanligtvis används är normalformerna 1 till 3. Mindre än 5 % av

alla de gånger man normaliserar behöver man använda normalformerna 4 och uppåt (Brown, 2002).

Den första normalformen (1:a NF) handlar om att man ska se till att det endast finns ett värde i varje ruta i tabellen. Termerna ska vara odelbara och endast uppträda en gång i tabellen. Den andra normalformen (2:a NF) handlar om att alla attribut (egenskaper) i tabellen ska bero på hela primärnyckeln. Man ska inte kunna härleda egenskaper genom att använda en del av primärnyckeln. För att uppfylla 2:a NF måste 1:a NF vara uppfyllt först. Den sista normalformen av de som vi tar upp är den tredje (3:e NF) och den handlar om att man ska försöka undvika att man har attribut som skulle kunna fungera som nyckel. Det får heller inte finnas något transitivt beroende mellan ett attribut och primärnyckeln. Det innebär att om man kan identifiera ett attribut genom ett annat attribut som är identifierat av primärnyckeln så är det första attributet transitivt beroende av primärnyckeln. Även här måste 1:a NF och 2:a NF uppfyllas innan 3:e NF prövas (Apelkrans & Åbom, 2001).

Vår databas innehåller tabellerna *discussion*, *subscribe*, *error* och *coordinate*. De innehåller attributen som följer och primärnyckeln är det understrukna attributet. Tabellernas normaliseras enligt följande:

DISCUSSION: id, parent, title, date, text, fname, root

SUBSCRIBE: id_sub, disc_id, email

ERROR: id_err, source, obsdate, error, concern, comment, date, file, sign

COORDINATE: id_co, source, obsdate, line, fname, sname, sign

Våra tabeller strider inte mot 1:a NF pga. att det endast kommer att finnas ett värde i varje cell i tabellerna. Inte heller strider de mot 2:a NF pga. att alla attribut är beroende av hela primärnyckeln för att rätt information ska kunna tas fram. Däremot så bryter tabellen *discussion* mot den 3:e NF genom att man kan få fram texten om man vet om titeln. Men vi byter inte tabellen mot två nya eftersom den är ganska liten.

4 Tekniska verktyg

Den teknik som vi använder oss av är språket PHP och databasen PostgreSQL som webbservern har stöd för på Rymdobservatoriet samt att det är den teknik som används till den redan befintliga applikationen. Vi har valt att använda oss av samma verktyg som de redan använder sig av för att det kommer att underlätta implementeringen och underhållet av vår applikation. Detta var även ett önskemål från observatoriet.

Under utvecklingen av webbapplikationen kommer vi inte att jobba mot Rymdobservatoriets server utan mot en dator på högskolan i Trollhättan. Vi har på den datorn med hjälp av Rickard Torkar, doktorand på HTU, installerat Linux Red Hat 7.2. I den versionen av Red Hat så ingår bl.a. stöd för webbservern Apache, skriptspråket PHP och databasen PostgreSQL, dvs. de delar som vi behövde.

4.1 PHP

PHP är ett populärt skriptspråk som är speciellt anpassat för webbutveckling. Popularitet kommer av att språket har stöd för många av de vanligaste programmen och programmeringsbiblioteken. Det går lätt att koppla PHP med ett antal olika databaser som man sedan kan kommunicera med genom att använda olika funktioner. De stöd som PHP ska ha bestäms av något som kallas för PHP-tillägg. Vilka utav dessa tillägg som man vill använda bestämmer man vanligtvis vid kompileringen av PHP. De tillägg som man valt länkas då in och behöver inte konfigureras separat för att fungera efter installationen (Jonsson, 2001).

PHP behöver en webbserver för att kunna köras och kopplingen mellan PHP och webbservern görs med hjälp av Server Application Programmig Interface (SAPI). SAPI är ett gränssnitt som i sin tur har stöd för de vanligaste och populäraste webbservrar som finns, t.ex. Apache, Roxen och IIS. På grund av detta så kan PHP bli en del av webbservern istället för att vara ett externt program. PHP kan man nu säga är plattformsöverskridande efter införandet av SAPI version 4 och det betyder att det kan köras på de vanligaste webbservrarna och i de flesta operativsystem (Jonsson, 2001).

4.2 PostgreSQL

PostgreSQL är en objektrelationell databas baserad på POSTGRES version 4.2. Den utvecklades på University of California och är open-source. PostgreSQL erbjuder en ökad förmåga att lägga till koncept på ett sådant sätt att användarna lätt kan utöka systemets arv, datatyper och funktioner. Det finns andra kännetecken som tillhandahåller ökad förmåga och flexibilitet. Det är restriktioner, triggers, regler och transaktionell integritet. Dessa kännetecken placerar PostgreSQL i kategorin objektrelationella databaser. PostgreSQL stödjer nästan alla SQL-uttryck, inklusive subselects, transaktioner och användardefinierade typer och funktioner (URL G).

Med över ett decennium av utveckling bakom sig så är PostgreSQL den mest avancerade open-source databasen som är tillgänglig överallt och som stödjer nästan alla SQL-konstruktorer samtidigt som den har ett brett antal språkbindningar tillgängliga. År 1986 började man med implementeringen av POSTGRES DBMS som år 1994 blev Postgres95 efter det att Andrew Yu och Jolly Chen adderade en SQL interpretator till POSTGRES. År 1996 blev det uppenbart att namnet Postgres95 inte skulle klara av testet i tiden. PostgreSQL gruppen valde då ett nytt namn som blev just PostgreSQL. Det nya namnet skulle representera relationen mellan originalet POSTGRES och den senare versionen som hade SQL-stöd (URL H).

5 Människa-Dator-Interaktion

Aktörerna i projektet Odin är forskare inom radio- och rymdastronomi med begränsad datorvana. Ett krav från vår handledare och övriga berörda är att webbapplikationen måste bli

lätthanterlig och användarvänlig gentemot aktörerna. För att få ökad kunskap inom området har vi förutsättningslöst samlat in information om ämnet.

5.1 Vad är Människa-Dator-Interaktion

Människa-Dator-Interaktion (MDI) är en forskningsstudie över hur människor påverkas av datorer och i vilken omfattning systemet är utvecklat för att influera mänsklig användning. MDI är ett område som är under utredning och det finns inga standarder utan just nu bara rekommendationer om hur man bör utveckla användarvänliga system (Preece, J, Rogers, Y, Sharp, H, Benyon, D, Holland, S, Carey, T, 1999).

De processer som bearbetas i vår hjärna när vi tar emot, förvarar, överför och använder oss av kunskap kallas kognition och används framförallt vid samspelet mellan människa och dator. Forskare har påvisat och skapat lagar för hur individer minns vad de upplever och hur de lär sig av sina erfarenheter samt hur processerna bakom hur vi tolkar och lär oss är uppbyggda. Varseblivning av saker och ting handlar om hur vår hjärna tolkar det vi ser och upplever. Hjärnan kan inte se tvådimensionellt utan utgår från att allt är tredimensionellt och letar efter djup i allt den ser. Om ett föremål ändrar form väljer hjärnan att anta att föremålets avstånd har förändrats i första hand innan den motsträvigt kan medge att något av de som vanligtvis är konstant har omskapats (URL F).

Allt man uppfattar med ögonen blandas med de minnen man har och bidrar till att hjärnan skapar en bild som den visar för oss. Denna bild behöver nödvändigtvis inte vara en exakt kopia av verkligheten utan något som hjärnan representerar för oss med hjälp av förändring, förstärkning och förkastning av informationen (URL F).

En viktig faktor som man måste ta hänsyn till i utvecklingen är att olika aktörer uppfattar och behåller information på olika sätt beroende på om de använder höger eller vänster hjärnhalva. Yttre omständigheter som utbildning, kulturell och nationell bakgrund är också något man måste ta med i sina beräkningar över hur man ska designa system. Att utvecklingen går fort fram och att det hela tiden kommer nya teknologier som aktörerna måste anpassa sig efter gör att aktörerna är vana att förändringar sker men föredrar att få dem gradvis (Preece et al., 1999).

Det är viktigt att systemet utvecklas efter de förkunskaper som finns hos aktörerna och anpassar sig till den nivån. Om aktörerna har liten erfarenhet från datorer är det viktigt att designa funktionerna så att de blir enkla att förstå. Utformningen av ett komplext objekt tar minst dubbelt så lång tid som det vanligtvis tar att utveckla den på grund av att komplexiteten på objektet måste brytas ner. Objektet måste bli överblickbart även för en ovan aktör. Det är mycket lättare att känna igen ett moment än att komma ihåg det (Preece et al., 1999).

5.2 Utformning av gränssnitt

Innan man påbörjar utformningen av en webbapplikation måste man tänka på vad den kommer att användas till och vilka som kommer att vara aktörerna. Applikationens innehåll

ska styra strukturens form, och inte tvärtom. Aktörernas erfarenheter av datorer och internetanvändning ger riktlinjer på hur mycket man måste anpassa applikationen.

En viktig faktor att tänka på är att man läser långsammare och mindre på webben och man bör därmed undvika långa texter. Webben är inte en primär läsplats och sidor med mycket text bör kunna skrivas ut.

Innehållet på sidan bör vara välorganiserad och strukturerad så att aktörerna snabbt hittar vad de söker. Startsidan på webbapplikationen ska vara informativ och ge en första överblick över vad applikationen har att erbjuda. En övergripande länkmeny som visar huvudavsnitten samt en ingress som kortfattat beskriver vad varje del innehåller gör att man snabbt får en förståelse över vad man kan utföra på applikationen.

Länkning eller rullningslister kan ha inverkan på aktörerna. En djup länkhierarki kan medföra att aktörerna mister orienteringen över var man befinner sig inom applikationen och att man upplever avståndet mellan sektionerna som enorma. Länkning kan resultera i att nedladdningen av nya sidor kan vara tidskrävande. Mycket grafik bidrar till att nedladdningstiden blir förlängd och gör att läsrytmen rubbas (URL I).

De hjälpmedel som man använder sig av vid navigering är till för att indikera vart på sidan man hamnar. Abstrakta termer och svårtolkade symboler bör undvikas för att underlätta navigeringen för aktörerna. Att skriva ”Till föregående sida” eller använda sig av pilar gör att aktören måste komma ihåg vart han tidigare varit och kan skapa förvirring. En aktör ska aldrig behöva gissa sig till vilken sida han kommer att hamna på. Skriv klart och tydlig ”Till startsidan” eller namnet på den sidan som länken leder till. Länkar rekommenderas att vara understrukna och inneha en blå färg och efter att man har klickat på den bli lila. Detta är ingen standard men en regel som många tillämpar i praktiken. Använd inte blå färg för text som inte är länkar då det kan vilseleda användaren.

Rubriker och logotyper bör finnas på alla sidor så att aktören kommer ihåg vilken sida han befinner sig på. Logotyper brukar vanligtvis vara länkade så att man kommer tillbaka till startsidan. Det ska vara lätt att navigera mellan sidorna och att ta sig tillbaka till huvudmenyn. Ramar kan användas för att få navigeringen konstant och gör att man kan orientera sig både på global och på lokal nivå. Det är lätt för användaren att förflytta sig mellan avsnitten, få en god överblick över innehållet samt att det inte krävs så många steg i länkhierarkin. Nackdelarna är att de tar längre tid att ladda ner och att alla webbläsare inte har stöd för ramar (URL I).

Man bör utforma webbapplikationen konsekvent och enhetligt så att användarna känner igen sig. Två länkar till olika sidor bör inte ha samma namn och länkar till samma sida bör heta likadant för att få en enhetlighet. Länkar i texten bör undvikas för att inte störa läsrytmen och för att undvika att man inte kommer tillbaka till ursprungstexten. Vid utformandet av applikationen bör man anpassa sidlayouten efter skärmstorleken. I dagsläget rekommenderar man att anpassa sidutformningen till 14-15 ” bildskärmar.

Om man använder mörk text på ljus bakgrund får man den bästa kontrastverkan. Använd endast ett par färger och var konsekvent med färgerna på alla sidor. Vid val av teckensnitt ska

man tänka på att teckensnitt med seriffer är mer lättlästa på bildskärmen än teckensnitt utan. Använder man sig av kursivstil kan det medföra att texten blir mer svårläst. Man kan även använda sig av en större teckenstorlek på texten som visas på bildskärmen än vad som man vanligtvis har i pappersformat för att göra det mer läsvänligt för användaren (URL I).

6 Säkerhetsaspekter

För att säkerhetställa vår applikation frågade vi Michael Olberg, systemansvarige på Rymdobservatoriet vilka säkerhetsaspekter de ville att vi skulle vidta på vår applikation. Det visade sig att den säkerhet som de ansåg sig behöva redan fanns på den befintliga applikationen och eftersom vår applikation är en del av den så kommer även vår del att omfattas av dessa aspekter.

Projektet Odin använder .htacess och .htpasswd som säkerhetsåtgärd på sin nuvarande applikation. Funktionerna .htacess och .htpasswd finns inbyggda i webbservern Apache och är till för att skapa en inloggningsfunktion som ser till att endast behöriga kan komma åt kataloger och filer på applikationen.

För att skapa en inloggningsfunktion börjar man med att skapa filerna .htacess och .htpasswd. Filen .htpasswd behandlar användarnamnet med tillhörande lösenord i krypterad form. För att filen inte ska kunna kommas åt från exempelvis webbläsaren lägger man filen i en icke publik katalog alltså inte i public_html katalogen. När man har placerat filen i en icke publik katalog måste man modifiera sökvägarna AuthUserFile och AuthGroupFile så att de stämmer överens med sin servers (URL A). Filen .htacess placeras i varje katalog som ska skyddas. När en användare försöker ansluta sig till en katalog som har htacess kommer en inloggningsruta att visas där endast behöriga kommer att ha tillgång till lösenordet (URL B) .

7 Resultat

Vårt arbete har resulterat i en webbapplikation som är en utökning av projektet Odins redan befintliga hemsida. Webbapplikationen har vi utformat tillsammans med berörda aktörer på Rymdobservatoriet. Genom systemmodelleringen och våra tekniska förkunskaper kom vi fram till en lösning av problematiken.

I följande underkapitel kommer vi att förklara hur varje del är uppbyggd och hur de fungerar. Implementeringen av webbapplikationen kommer att ske vid ett senare tillfälle då Rymdobservatoriets systemansvarige inte kunde ta emot oss tidigare än juni. Under implementeringsrubriken kommer det i dagsläget bara att stå en kortfattad beskrivning över hur vi kommer att gå tillväga för att implementera applikationen på projektet Odins server.

7.1 Applikationslösning

När systemmodelleringen var avklarad påbörjades den tekniska lösningen av webbapplikationen. Med hjälp av modelleringen fick vi en uppfattning om systemets utformning.

Eftersom vår applikation är en del av Rymdobservatoriets redan befintliga hemsida, innebär detta att vi har en del restriktioner som vi måste följa. Restriktionerna berör delar som design och språkval. Detta medför att applikationen kommer att vara uppbyggt med ramar och att all text står på engelska som är aktörernas officiella språk. På varje sida finns det två ramar, den vänstra ramen innehåller en länkmeny och logotypen till Onsala Rymdobservatorium som är länkad till deras hemsida. Denna ram kommer inte att ändras någon gång utan är statisk. Den högra ramen innehåller information som är beroende på vilken länk man har aktiverat.

7.1.1 Databas

Vi hade inga tidigare erfarenheter av att arbeta med en PostgreSQL databas, så den första tiden gick åt till att lära sig hur databasen fungerade och hur den kunde användas för att tillgodose våra behov.

Vid normaliseringen som gjordes i systemmodelleringen har ett behov av fyra tabeller med tillhörande attribut i databasen framkommit. Det är tabellerna, *discussion*, *subscribe*, *error* och *coordinate*. Dessa tabeller är till för att lagra information som skrivs in i databasen på ett strukturerat sätt och ska efter givna kommandon kunna modifieras av aktörerna och därefter uppdateras via applikationen. För att visualisera uppbyggnaden av tabellerna se bilaga D.

Tabellen *discussion* innehåller sju olika attribut som har till uppgift att lagra information om de diskussioner som förekommer. Utav dessa attribut är det tre stycken som kanske inte är självförklarande var de får sina värden ifrån, så dem förklarar vi närmare här. Attributet *id* som är det första attributet i tabellen genereras automatiskt av databasen och fungerar som en räknare. Attributet *parent* får sitt värde med hjälp av PHP-koden. Detta attribut indikerar om ett inlägg har en ”förälder” eller inte. Vid alla nya diskussioner som skapas får *parent*-attributet värdet noll. Svarar man däremot på ett inlägg så får attributet *parent* värdet från det tidigare inläggets *id*-nummer. Till sist har vi attributet *root* som håller reda på alla inlägg inom en och samma diskussion. Detta görs genom att när en ny diskussion skapas så blir *root*-värdet satt till noll. På alla de kommande inläggen inom denna diskussion blir värdet på *root*-attributet satt till diskussionens *id*-nummer. De övriga attributen som ingår i den här tabellen är *title*, *text*, *date* och *fname*.

Förklaring till hur data lagras i tabellen *discussion*, den är tom som utgångspunkt (se tabell 6 för visualisering):

1. Ny diskussion med namnet ”Test” skapas. Attributet *id* får nu värdet 1 och attributen *parent* och *root* får båda värdet 0.
2. Ett inlägg görs under diskussionen ”Test”. Attributet *id* får nu värdet 2 och attributen *parent* och *root* får båda värdet 1.

3. Ett svar på ett tidigare inlägg sker under diskussionen ”Test”. Attributet *id* får nu värdet 3, *parent* får värdet 2 och *root* får värdet 1.
4. Nu skapas en ny diskussion med namnet ”Test 2”. Attributet *id* får då värdet 4 samt att *parent* och *root* båda får värdet 0.
5. Ett inlägg görs under diskussionen ”Test 2”. Attributet *id* får värdet 5, *parent* får värdet 4 och *root* får värdet 4.

Tabell 6 – Databastabellen *discussion*

id	parent	title	date	text	fname	root
1	0	Test	2002-05-20	En ny diskussion, Test, skapas.	Karin	0
2	1	Re: Test	2002-05-21	Detta är ett inlägg på diskussionen Test	Anna	1
3	2	Re: Re: Test	2002-05-22	Detta är ett inlägg på föregående svar på diskussionen Test	Karin	1
4	0	Test2	2002-05-22	En ny diskussion, Test2	Anna	0
5	4	Re: Test2	2002-05-23	Svar på diskussionen, Test2	Karin	4

Tabellen *subscribe* används när en aktör vill prenumerera på en diskussion. Den här tabellen innehåller endast 3 attribut och de är: *id_sub*, *disc_id* och *email*. Attributet *id_sub* är en räknare som automatiskt får sitt värde av databasen. Attributet *disc_id* däremot får sitt värde från PHP-koden och värdet kommer från en diskussions id-nummer och *email* innehåller precis som det låter, en e-postadress.

Tabellen *error* innehåller information om de fel som uppstått vid olika mätningar. Även den här tabellen har en räknare, *id_err*, som används för att identifiera varje post för sig i tabellen. I detta fall när någon aktör vill radera en post ur tabellen. De övriga attributen förutom *sign* innehåller information om felupptäckterna. På den PHP-sida där tabellen används har aktörerna möjlighet till att ladda upp filer till webbservern. Namnet på dessa filer lagras sedan i attributet *file*. Signaturen från den person som rapporterat in felet lagras av attributet *sign* och det används sedan när en post ska raderas ur tabellen.

Tabellen *coordinate* består bl.a. av attributen *source*, *obsdate* och *line* vilket innebär att man kan lägga till en post med de attribut som sedan kommer att användas för att underlätta det redundanta arbetet. När posten är registrerad kan aktörerna skriva upp sig på posterna vilket påvisar för de andra aktörerna att man arbetar med källan. Attributet *id_co* genereras automatiskt av databasen. Detta attribut används då en aktör vill registrera sig på en källa eller då man vill avregistrera sig från en källa. Vid registrering måste förnamn, efternamn och

signatur skrivas in för att registreringen ska gå igenom. När man sedan vill avregistrera sig räcker det att skriva in signaturen igen.

För att skapa en förbindelse till databasen så får man skriva: **$conn=pg_connect("host=localhost dbname=odin user=examen1 password=DYWRBK");** Samtidigt som man skapar förbindelsen, ger man en variabel värdet av förbindelsen. I detta fall skapade vi en variabel som heter *conn* och den får värdet av förbindelsen till sig, dvs. databasens värd, namn, användare och lösenord. För att man sedan ska kunna exekvera en sql-fråga så kan det vara bra att även där använda sig av en variabel som får frågan som värde. Det kan då se ut så här: **$sql="SELECT * FROM coordinate ORDER BY source";** När sedan frågan ska exekveras skrivs: **$result=pg_exec($conn, $sql);** Variablen result kan sedan användas vid andra tillfällen och den får värdet från exekveringen av databaskopplingen och sql-frågan.

7.1.2 Diskussionsforum

Diskussionssidan är en funktion i webbapplikationen som är till för att underlätta de diskussioner som uppkommer inom projektet Odin. När man kommer in på sidan visas rubrikerna till alla diskussioner samt de inlägg som finns tillgängliga. För att läsa ett inlägg eller vad diskussionen berör klickar man på knappen ”Read” och innehållet kommer att visas i ett popup-fönster. Om man vill läsa alla inlägg som finns på diskussionen går man in på länken ”View thread”. Efter att man har läst texten kan man välja att svara på diskussionen eller att stänga fönstret. Om man väljer det sistnämnda kommer man tillbaka till sidan som visar alla rubriker på befintliga diskussioner. Om man väljer att svara på diskussionen klickar man på ”Reply” och ett nytt fönster kommer att visas där man ser vilken diskussion man kommer att svara på och där man själv skriver in sitt inlägg. Här skriver man in sitt förnamn så att övriga aktörer ska veta vem som har gjort inlägget. När man svarar på en diskussion kommer inlägget att hamna en bit in på sidan så att aktörerna ska veta vad som är rubrik och vad som är svar på diskussionen.

När en aktör vill påbörja en ny diskussion går man in på diskussionsforumet och klickar på länken ”Create new discussion”. Ett popup-fönster kommer då att visas på skärmen och där skriver man in titeln på diskussionen som man vill starta samt den text som berör ämnet som ska diskuteras. Om inga inlägg har gjorts på diskussionen de senaste 30 dagarna kommer diskussionen med tillhörande inlägg att automatiskt tas bort av systemet.

Om en aktör vill prenumerera på en diskussion och på så sätt få ett meddelande genererat till sin e-postadress som talar om att nya inlägg har gjorts på diskussionen, kan aktören gå in på länken ”Subscribe to discussion” och registrera sig på den diskussion som man vill följa. När man klickat på länken kommer ett fönster visas med en rullgardinslista över befintliga diskussioner. Man väljer den diskussion som man är intresserad av från listan och skriver sedan till vilken e-postadress man vill att meddelandet ska skickas till.

Filerna *sendmail.php* och *autodelete.php* exekveras automatiskt. Filen *sendmail.php* kontrollerar om några nya inlägg har gjorts i de pågående diskussionerna. Om det har det och ifall en eller flera aktörer prenumererar på de diskussioner som fått nya inlägg så ska ett e-

postmeddelande skickas ut till dem. Filen *autodelete.php* kontrollerar om diskussionerna har inlägg som är nyare än en månad, om de inte har det så raderas hela de diskussionerna med tillhörande inlägg från databasen. För att få dessa filer att exekveras automatiskt använder man sig av kommandot *crontab* i Linux. Detta kommando möjliggör att en eller flera filer kan specificeras vid vilken tid de ska köras. För att kunna skriva in en eller flera filer så börjar man med att skriva *crontab –e* som gör att man kommer in i editorn VI. Där kan man sedan skriva in tiden och vilka dagar som man vill att respektive fil ska köras samt naturligtvis även vilken fil som ska köras. Man skriver in det i ordningen <minuter> <timmar> <dag> <månad> <veckodag> <kommando>. Vi har skrivit detta i vår crontab-fil:

59 23 * * * cd /home/examen1/public_html/ && /usr/bin/php ./sendmail.php
59 23 * * * cd /home/examen1/public_html/ && /usr/bin/php ./autodelete.php

Dessa rader betyder att båda våra filer ska köras klockan 23:59 varje dag och att i katalogen public_html finns de filer som ska exekveras, dvs. *sendmail.php* och *autodelete.php*.

Om man vill se programmeringskoden till diskussionssidan finns den att beskåda i bilaga E – diskussionsforum.

7.1.3 Felhantering

Felhanteringssidan är till för att aktörerna ska få kunskap om vilka fel som upptäckts på specifika källor. När man kommer till sidan visas alla nuvarande fel på skärmen sorterade i alfabetisk ordning efter källa och det andra sorteringsvillkoret är observationsdatumet.

På denna sida finns även en sökfunktion som kan hantera kriterier på attributen *source*, *type of error* och *concern*. Själva sökningen och sökresultatet visas i ett popup-fönster.

När man upptäcker nya fel kan de läggas till genom att aktören fyller i formuläret som finns tillgängligt via en länk på felhanteringssidan. Aktören ska nu fylla i data till de attribut som står. Det finns även här möjlighet att ladda upp en kompletterande fil för att på så sätt kunna rapportera mer än vad själva formuläret klarar av. För att formuläret ska kunna hantera en filuppladdning så måste man skriva form-taggen på ett speciellt sätt, dvs. så här: **<form enctype="multipart/form-data" name="add" action="<?=$PHP_SELF?>" method=POST>**. Det speciella här är att man skriver multipart/form-data istället för text/plain efter attributet enctype. Oftast använder man inte enctype-attributet om man bara ska ta hand om ren text men om det skrivs in på detta sätt så kan en fil laddas upp till webbservern.

Använder man sig a PHP för uppladdning av filer så är formulär med enctype standard. Genom att använda sig av ”MAX_FILE_SIZE” kan en maxstorlek anges på de filer som tillåts laddas upp. Om inget värde anges där så gäller defaultvärdet 2MB som är satt av PHP, detta gäller i vårt fall. För att servern ska kunna lagra filerna som laddas upp har vi använt oss av metoden POST som finns tillgänglig i PHP. $HTTP_POST_FILES är en global vektor som innehåller följande element som vi har använt oss av:

$HTTP_POST_FILES["file"]["name"]vilket innebär att vektorn får information om ursprungsnamnet på filen dvs. det namn som filen innehar innan uppladdningen.

$HTTP_POST_FILES["file"]["tmp_name"] ger ett temporärt namn till filen som blev uppladdad till servern.

move_uploaded_file("$tmpfile", "files/$file") innebär att filen kommer att flyttas till katalogen *files* och döpas om till sitt ursprungsnamn. Funktionen tar emot två argument och det är filnamnet och destinationen dit filen ska skickas. Filnamnet måste vara namnet på en uppladdad fil och destinationen måste innehålla hela sökvägen, inte bara ett katalognamn (Jonsson, 2001).

Om man vill ta bort en inrapportering av problem som uppstått är det möjligt genom den delete-knapp som finns på varje rad. Innan posten raderas ur databasen så får aktören en förfrågan om att fylla i sin signatur vilket han/hon måste göra för att raderingen ska gå igenom. Det finns nämligen ett villkor i SQL-frågan (**DELETE FROM error WHERE id_err='$id_err' AND sign='$sign';**) som möjliggör den kontrollen så att ingenting raderas av misstag. Tanken är att man bara ska kunna ta bort sina egna rapporteringar.

Om man vill se programmeringskoden till felhanteringssidan finns den att beskåda i bilaga E – felhantering.

7.1.4 Samordning

Samordning är en sida där aktörerna kan koordinera sig med varandra och på så sätt minimera redundant arbete.

Alla kända källor med observationsdatum och linje finns fördefinierade och visas sorterade i bokstavsordning på sidan. När en ny källa upptäcks registrerar man den genom att skriva in källans namn och datum samt väljer linje ur den rullgardinslista som finns. Alternativen i rullgardinslistan måste läggas in av administratören i programmeringskoden.

När en aktör vill registrera sig på en källa går han in på ”Sign up” knappen på den rad där källan står. Man fyller i sitt för- och efternamn och avslutar med sin signatur. Fönstret stängs automatiskt när man trycker på ”ok” knappen. Om man inte fyller i de tre fälten kommer en alertruta upp, som påminner om att alla fälten måste vara ifyllda för att man ska få registrera sig på en källa.

När man vill avregistrera sig på en källa som man tidigare har registrerat sig på så trycker man på ”Remove” knappen. Popup-fönstret visar en textruta där man måste fylla i sin signatur. En SQL-fråga kontrollerar att id-numret som raden har i databasen stämmer överens med det nummer som skickas med till fönstret samt att signaturen på raden ska stämma med den signatur som aktören uppger. Om allt stämmer kommer applikationen att ta bort registreringen på den specifika raden.

Om man vill se programmeringskoden till samordningssidan finns den att beskåda i bilaga E – samordning.

7.2 Användarvänligt gränssnitt

För att resultatet av webbapplikationen ska bli användarvänligt för aktörerna har vi använt oss av en informativ startsida med strukturerat innehåll. Den informativa texten på startsidan talar om vad man kan utföra på applikationens delar och är kortfattad och koncis.

Applikationen är anpassad efter 15” bildskärmar och alla sidorna är utformade konsekvent med en enhetlig layout för att skapa igenkänning hos aktörerna. Teckensnittet som används är samma på alla sidor men varierar i storlek och format beroende på om det är en rubrik, underrubrik eller brödtext. Färgerna är begränsade för att skapa ett homogent utseende. All vanlig text är svart och länkad text är blå och understruken för att användarna ska veta vad som är ”klickbart”. Applikationen är uppbyggd med hjälp av ramar och dessa ramar har en gul och en grön bakgrundsfärg. Den gula ramen är statisk och visas hela tiden. Rymdobservatoriets logo finns där och är länkad till deras hemsida. Logotypen är dock den enda bild som används i applikationen och medför att sidorna har kort nedladdningstid. Ramen innehåller förutom logotypen den länkmeny som applikationen har att erbjuda och beroende på vad man klickar på kommer information visas i den ram som innehar den gröna bakgrundsfärgen. Länkar och knappar som förekommer i applikationen har informativa namn som gör det lätt för aktörerna att navigera mellan sidorna och förstå vad som händer när man använder sig av dem.

7.3 Implementering

Vi har utfört vårt arbete med att utforma en webbapplikation åt projektet Odin här på HTU. Implementeringen på Rymdobservatoriet och projektet Odins server kommer att ske genom att vi sparar ner koden på disketter och via disketterna implementerar detta på deras server. Tabellerna i databasen kommer att skapas manuellt direkt i Odins databas.

8 Slutsatser

När vi kom i kontakt med Rymdobservatoriet och projektet Odin hade organisationen problem med redundant arbete och att diskussioner som berörde forskarna blev långdragna och ineffektiva. Fel som upptäcktes vid mätningar kom aldrig till andras kännedom än till den som gjorde upptäckten. Vår uppgift var att lösa dessa problem genom att tillverka en webbapplikation som effektivt kunde hantera problematiken inom projektet och underlätta arbetssituationen för berörda aktörer.

8.1 Analys av resultat

Webbapplikationen som tillverkats har ett enkelt gränssnitt som gör det okomplicerat för aktörerna att navigera mellan sidorna. Varje sida innehåller information som gör det lätt att komma tillbaka till ursprungssidan. Koden som vi har programmerat har vi försökt att kommentera och göra så lättförstålig som möjligt för att underlätta det fortsatta arbetet för Rymdobservatoriets systemansvarige.

Webbapplikationen har till största delen blivit som vi och berörda aktörer från Rymdobservatoriet analyserat fram genom systemmodelleringen. Applikationen innehåller de restriktioner som var dess syfte.

Diskussionsforumet var den del som inte blev riktigt som vi trott från början då vi genom UML-notationen utförde en fönsterbeskrivning. Tanken var att varje diskussion skulle ha en rubrik som var synlig i fönstret från första början och när man klickade på den skulle den visa innehållet av diskussionen i samma fönster. Varje inlägg på diskussionen skulle visas på samma sätt och efter varje rubrik och inlägg skulle en ”svara” knapp ligga. Nu blev resultatet att varje diskussion med tillhörande inlägg kommer att visa endast sin rubrik och om man vill läsa innehållet får man trycka på länken ”Read” och innehållet kommer då att visas i ett popup-fönster. Forumet har dock den funktionalitet som projektet Odin är i behov av men innehar inte den layout som vi analyserat fram i det tidigare stadiet.

8.2 Rekommendation till fortsatt arbete

Vid en eventuell vidareutveckling av applikationen så föreslår vi att tid läggs ner på att skapa en administratörssida. Denna sida ska då underlätta underhållet av applikationen för administratören.

Eftersom systemet har en begränsning på att endast en aktör kan vara registrerad på varje källa medför det att om ytterligare en aktör vill registrera sig så måste han/hon lägga till källan på nytt och sedan registrera sig på den. Detta kan innebära att en och samma källa står med flera gånger och att det när avregistreringar har skett finns mer än en likadan post tom. Det borde alltså utvecklas en funktion som innebär att mer än en aktör kan registrera sig på en post utan att posten behöver läggas till på nytt igen. De källor som blivit inaktuella bör också kunna tas bort av administratören via en funktion för att slippa gå in i databasen och radera de berörda källorna.

Vid samordningslistan finns det även ”linjer” att välja mellan i en rullgardinslista. Dessa linjer är nu inskrivna i programkoden vilket innebär att om en ny linje tillkommer måsta administratören gå in i koden och lägga till den. Detta skulle kunna lösas genom att alla linjer sparas i en databastabell och att man väljer i rullgardinslistan från de värden som finns där. Då kan administratören lättare lägga till nya linjer samt ta bort de som inte kommer att användas mer på ett smidigt sätt.

9 Referensförteckning

9.1 Böcker/tidskrifter

Apelkrans, M & Åbom, C. (2001). OOS/UML – *En objektorienterad systemutvecklingsmodell för processorienterad affärsutveckling*. Lund: Studentlitteratur.

Booch, G, Rumbaugh, J & Jacobson, I. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley.

Brown, D (2002). *An introduction to object-oriented analysis.* New York:Wiley.

Eriksson, H-E, & Penker, M. (2000). *Business Modeling with UML*: business patterns at work. New York: Wiley.

Jonsson, V. (2001). *Webbprogrammering med PHP*. Lund: Studentlitteratur.

Kruchten, P. (2002). *The rational unified process –En introduktion.* Addison-Wesley.

Mathiassen, L, Munk-Madsen, A, Nielsen, P A & Stage, J. (2001). *Objektorienterad analys och design*. Lund: Studentlitteratur.

Olofsson, H. (2002-05-17). Radioastronom vid Onsala Rymdobservatorium.

Preece, J, Rogers, Y, Sharp, H, Benyon, D, Holland, S, Carey, T. (1999). *Human-Computer Interaction. Addison-Wesley.*

Rymdstyrelsen. (2001). *Satelliten Odin – skarpa ögon i rymden*. Nynäshamn: Jetlag Kommunikation AB.

Sommerville, I. (2001). *Software engineering.* Addison-Wesley.

Wallén, G. (2000). *Vetenskapsteori och forskningsmetodik*. Lund: Studentlitteratur.

9.2 Elektronisk information

URL A: The Apache Software Foundation. (1999-2002). [www dokument]. URL http://httpd.apache.org/docs-2.0/programs/htpasswd.html

URL B: The Apache Software Foundation. (1999-2002). [www dokument]. URL http://httpd.apache.org/docs-2.0/mod/core.html#accessfilename

URL C: Chalmers. (1999, sep 17). Onsala rymdobservatorium [www dokument]. URL http://www.chalmers.se/forskningskompetens/onsala.html

URL D: Gunnarsson, R. (2002-04-13). Forskningsmetodik [www dokument] URL http://www.infovoice.se/fou/index.htm

URL E: Hammargren, R. (1998, sep 7). *Onsala Rymdobservatorium* [www dokument]. URL http://www.oso.chalmers.se/popular/faq/faq.html

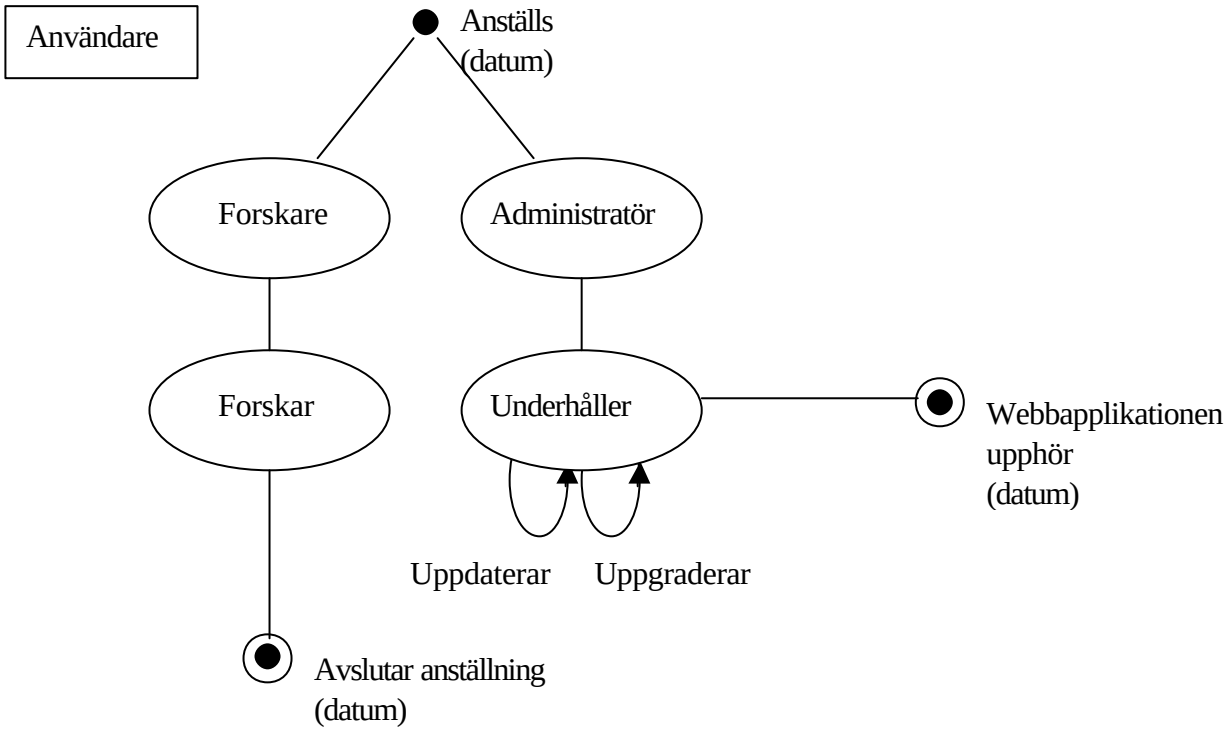
URL F: Häggstrand, P. (2000, okt 13). *Kognition -dator och människa i samarbete* [www dokument]. URL http://www.masda.vxu.se/multimedia/km/kogn_vt00/kog-del1.html

URL G: PostgreSQL Group [www dokument]. URL http://www2.se.postgresql.org/

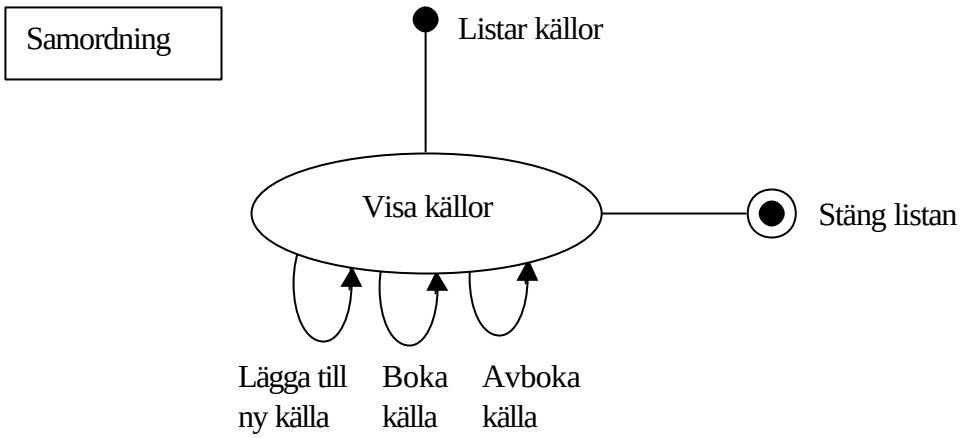
URL H: PostgreSQL Global Development Group [www dokument]. URL http://www2.se.postgresql.org/users-lounge/docs/7.2/tutorial-7.2-A4.pdf

URL I: Ullman, Ortman, Lenman, Torgny. (2000, sep 14). *CID'97-Riktlinjer för utformning av webbplatser* [www dokument]. URL http://www.nada.kth.se/cid/projekt/cid97/ANVANDARGRANSSNITT/Akorthet.html

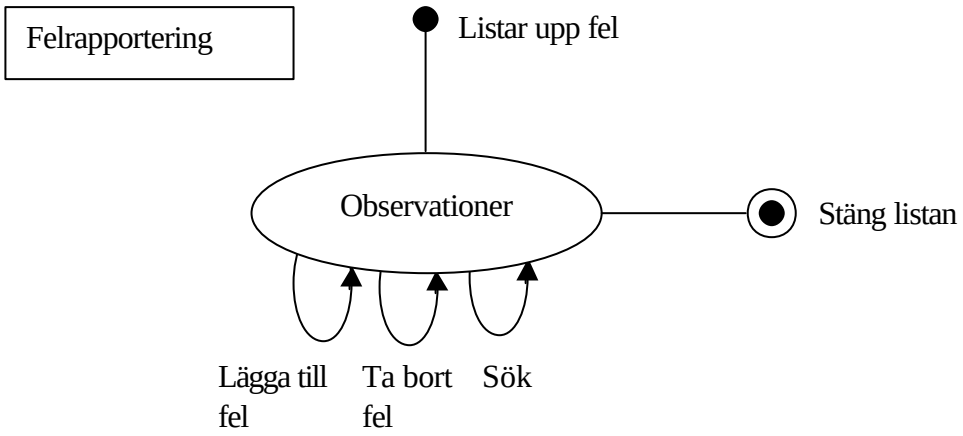
Bilaga A – Tillståndsdiagram



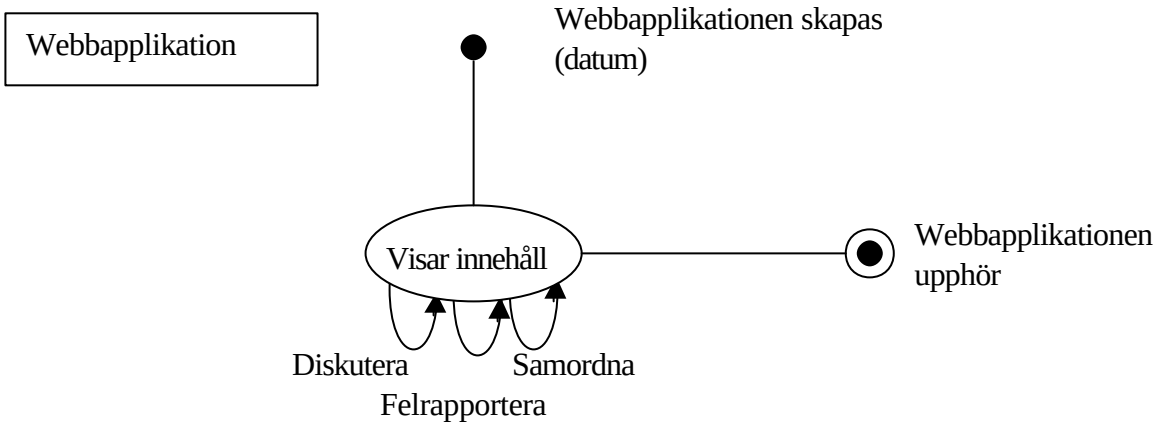
Figur 1 – Tillståndsdiagram, användare



Figur 2 – Tillståndsdiagram, samordning

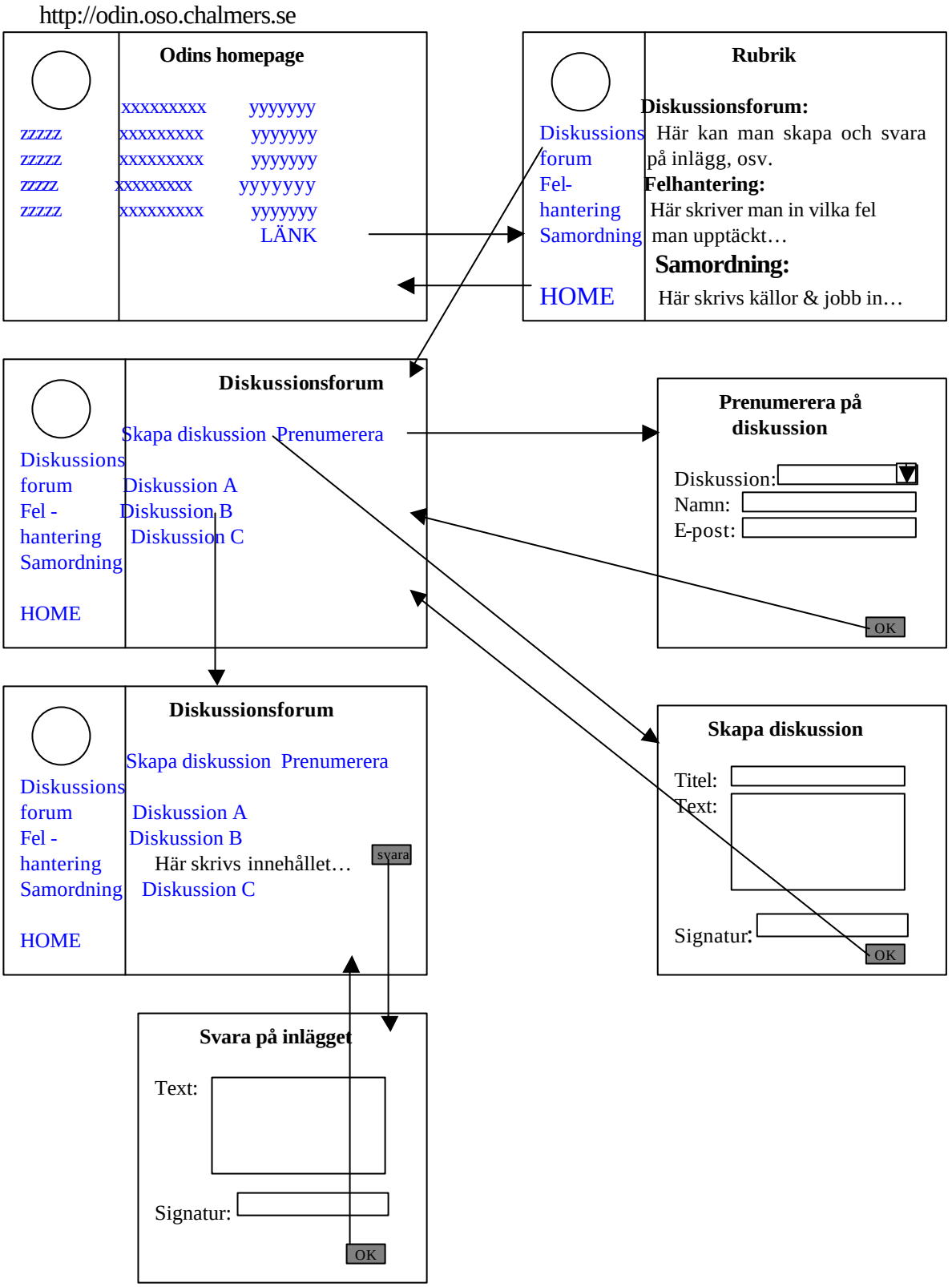


Figur 3 – Tillståndsdiagram, felrapportering

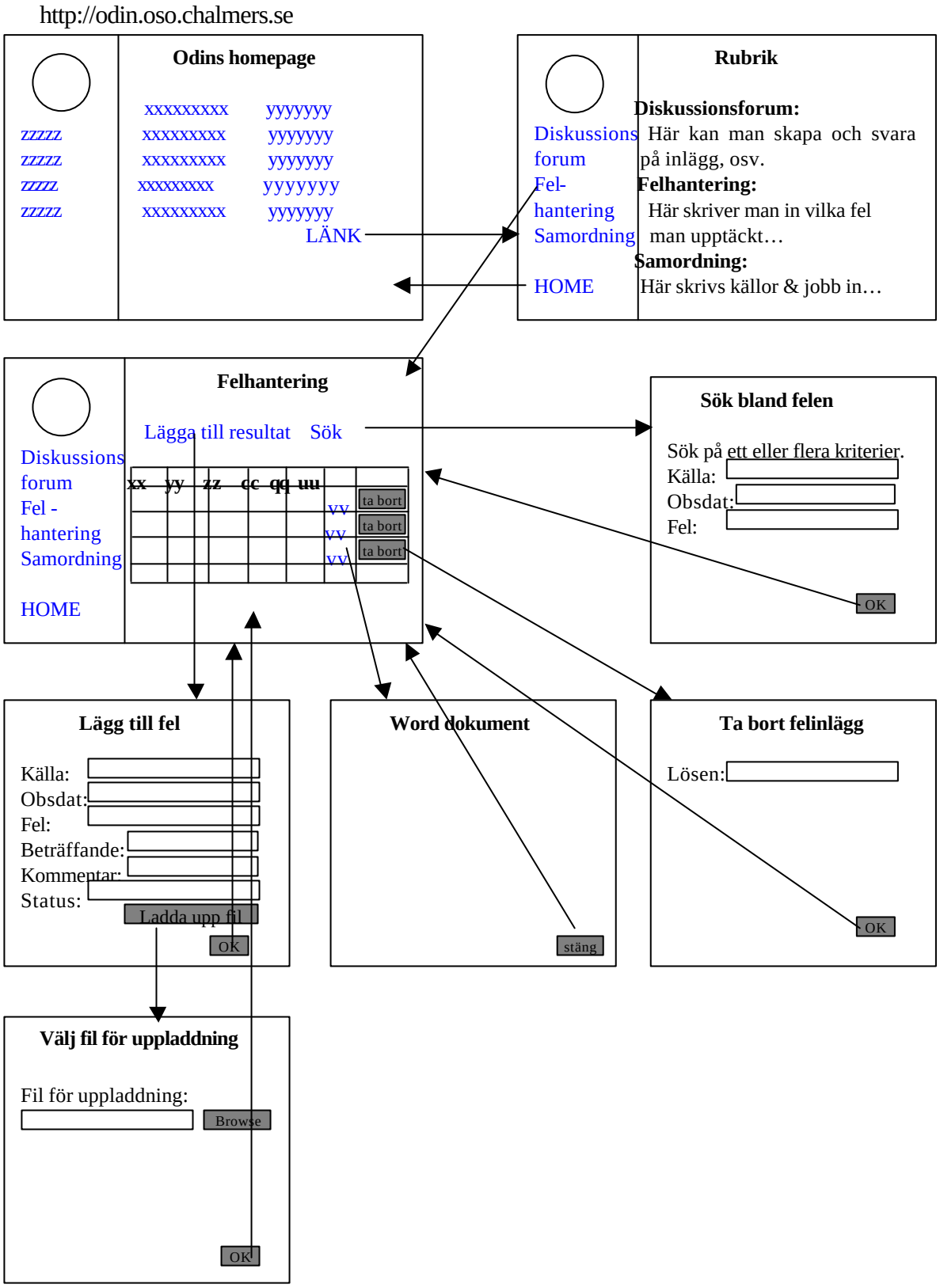


Figur 4 – Tillståndsdiagram, webbapplikation

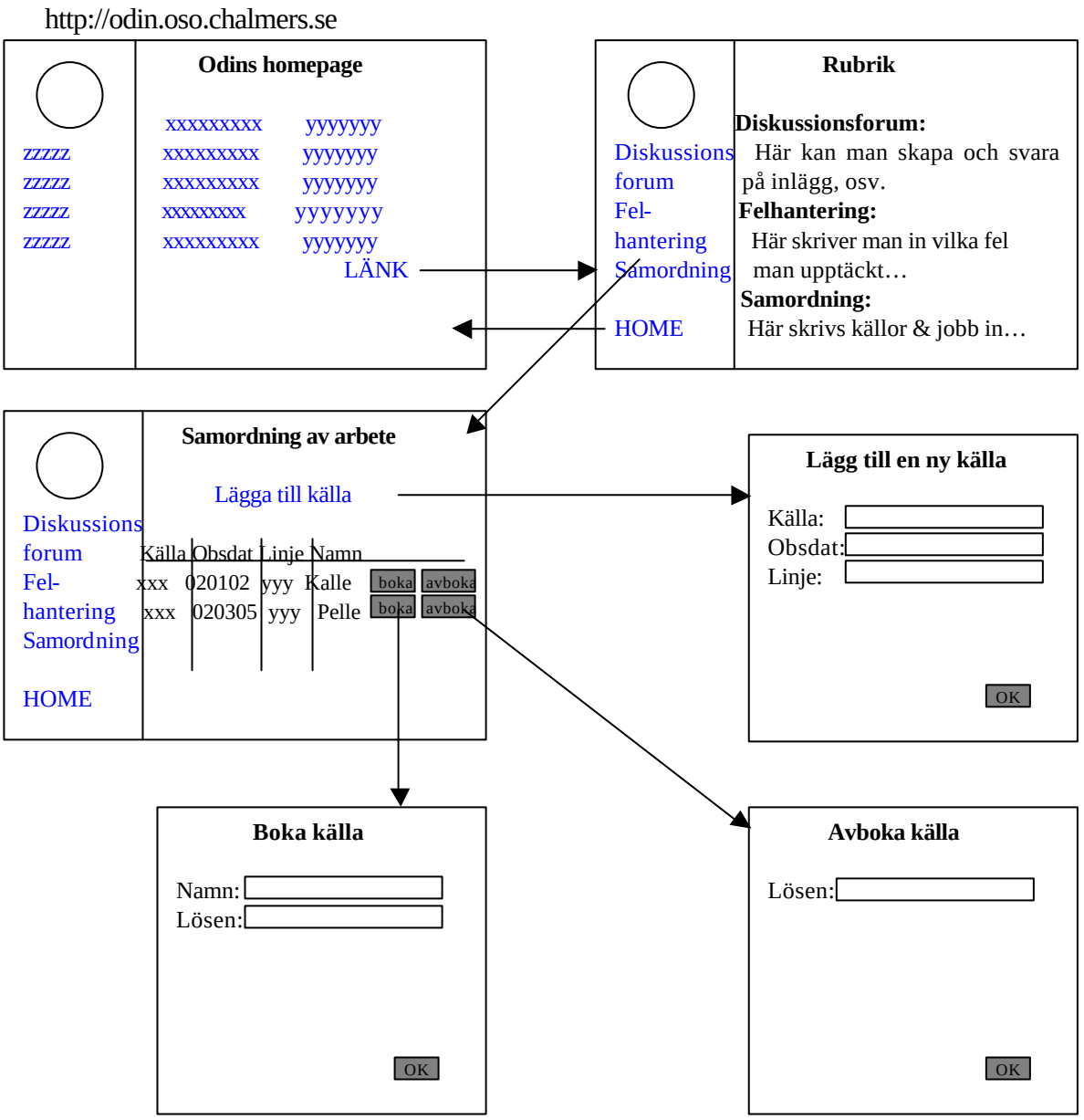
Bilaga B – Fönsterbeskrivningar



Figur 1 – Fönsterbeskrivning, diskussionsforum

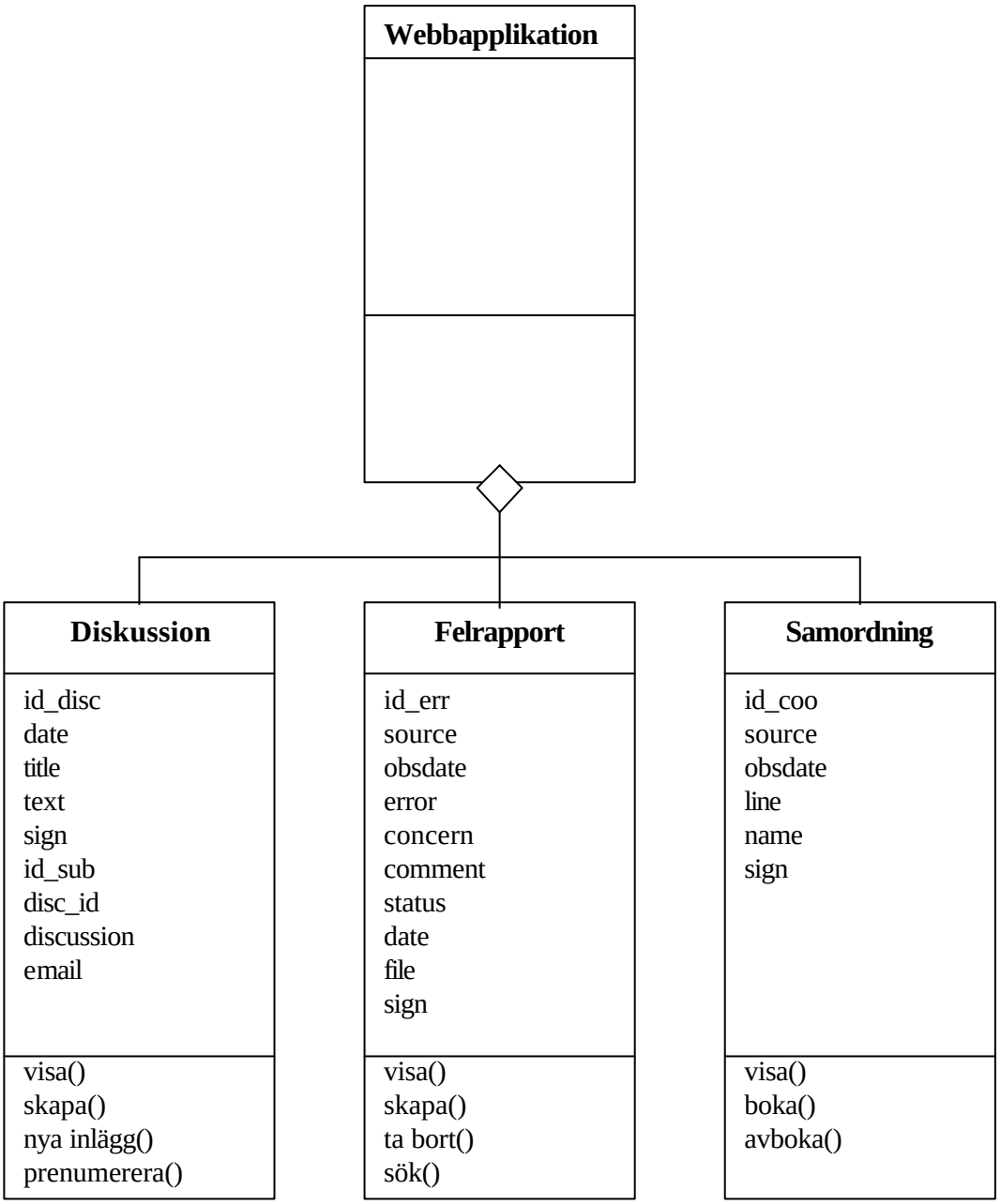


Figur 2 – Fönsterbeskrivning, felrapportering



Figur 3 – Fönsterbeskrivning, samordning

Bilaga C – Klassdiagram med attribut och funktioner



Bilaga D – Databastabeller

Table "discussion"

Attribute	Type	Modifier
id	integer	not null default nextval('"discussion_id_seq"'::text)
parent	integer	
title	character varying(255)	
date	date	
text	text	
fname	character varying(15)	
root	integer	

Index: discussion_pkey

Table "subscribe"

Attribute	Type	Modifier
id_sub	integer	not null default nextval('"subscribe_id_sub_seq"'::text)
disc_id	integer	
email	character varying(60)	

Index: subscribe_pkey

Table "error"

Attribute	Type	Modifier
id_err	integer	not null default nextval('"error_id_err_seq"'::text)
source	character varying(15)	
obsdate	character varying(10)	
error	character varying(20)	
concern	character varying(20)	
comment	character varying(50)	
date	date	
file	character varying(40)	
sign	character varying(5)	

Index: error_pkey

Table "coordinate"

Attribute	Type	Modifier
id_co	integer	not null default nextval('"coordinate_id_co_seq"'::text)
source	character varying(15)	
obsdate	character varying(10)	
line	character varying(10)	
fname	character varying(15)	
sname	character varying(20)	
sign	character varying(5)	

Index: coordinate_pkey

Bilaga E – Programmeringskod

Style sheet

style.css

```
th {font-family: Verdana; font-size: 10pt; font-style: bold}
td {font-family: Verdana; font-size: 10pt}
.big {font-family: Verdana; font-size: 12pt; font-weight: bold}
dt {font-family: Verdana; font-size: 10pt}
dd {font-family: Verdana; font-size: 10pt}
h1 {font-family: Verdana; font-size: 14pt; font-style: bold}
h2 {font-family: Verdana; font-size: 12pt; font-style: bold}
a:link { text-decoration: underline; color: #0066CC; font-size: 11pt}
a:visited { text-decoration: underline; color: #0066CC; font-size: 11pt}
a:active { text-decoration: underline; color: #0066CC; font-size: 11pt}
a:hover { text-decoration: underline; color: #000000; font-size: 11pt}
```

Ramar

index.php

```
<html>
   <head>
      <title>ODIN</title>
   </head>
   <frameset cols="200,*" frameborder="no" border="1" framespacing="0">
      <frame name="meny" frameborder="no" border="1" scrolling="no" noresize
src="left.php" marginwidth="2">
      <frame name="main" src="main.php" scrolling="auto">
   </frameset>
   <noframes>
      <body bgcolor="#FFFFFF">
      </body>
   </noframes>
```

```
</html>
```

left.php

```
<html>
   <head>
      <title>Meny</title>
      <link rel="stylesheet" type="text/css" href="style.css">
   </head>
   <body bgcolor="#FFFFCC">
      <table>
         <tr><td align="center"><a href="http://www.oso.chalmers.se" target="_parent"><img
src="./pics/Blue.gif" border="0"></a></td></tr>
         <tr><td height="10"> </td></tr>
         <tr><td align="center"><a href="./main.php" target="main">What can you do on this
site?</a></td></tr>
         <tr><td height="10"> </td></tr>
         <tr><td align="center"><a href="./discussion.php" target="main">Discussion
Forum</a></td></tr>
         <tr><td align="center"><a href="./error.php" target="main">Known data
problems</a></td></tr>
         <tr><td align="center"><a href="./coordinate.php" target="main">Data processing
coordination</a></td></tr>
         <tr><td height="10"> </td></tr>
         <tr><td align="center"><a href="http://odin.oso.chalmers.se"
target="_parent">ODIN</a></td></tr>
      </table>
   </body>
</html>
```

main.php

```
<html>
   <head>
      <title>ODIN</title>
      <link rel="stylesheet" type="text/css" href="style.css">
   </head>
   <body bgcolor="#CCCC99">
      <table width="500" border="0" cellpadding="0" cellspacing="0">
         <tr><td height="20"> </td></tr>
         <tr><td><h1>What can you do on this site?</h1></td></tr>
         <tr><td class="big">Discussion Forum</td></tr>
         <tr><td>On the discussion forum can you create new discussions, read contributions,
write an answer to a discussion and also sign up to subscribe on a discussion.</td></tr>
         <tr><td height="5"> </td></tr>
         <tr><td class="big">Known data problems</td></tr>
         <tr><td>On this side you can report errors that have occured in connection with a
measuring and also read about errors that your colleauges have discovered.</td></tr>
         <tr><td height="5"> </td></tr>
         <tr><td class="big">Data processing coordination</td></tr>
         <tr><td>Here you will find all the sources and you can sign up to a source which
means that you work with the source but not that the source is reserved. This means that you
can see if anyone are working with a source at the moment that leads to that you might choose
to ask for a result from a colleauge instead of doing the measuring yourself.</td></tr>
      </table>
   </body>
</html>
```

Diskussionsforum

discussion.php

```
<html>
  <head>
    <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
    <title>Discussion Forum</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="JavaScript">
      //All of these functions opens a popup-window. Three of them also sends a variable-value to the next window.
      function popUp_create()
      {
        window.open("create.php", "poppage", "toolbars=0, scrollbars=0, location=0, statusbars=0, menubars=0, resizable=0, width=500, height=300 left = 100, top = 100");
      }
      function popUp_subscribe()
      {
        window.open("subscribe.php", "poppage", "toolbars=0, scrollbars=0, location=0, statusbars=0, menubars=0, resizable=0, width=350, height=250 left = 100, top = 100");
      }
      function popUp_read(id)
      {
        var url="showpost.php?id="+id;
        window.open(url, "poppage", "toolbars=0, scrollbars=1, location=0, statusbars=0, menubars=0, resizable=0, width=500, height=300 left = 100, top = 100");
      }
      function popUp_reply(id)
      {
        var url="reply.php?id="+id;
        window.open(url, "poppage", "toolbars=0, scrollbars=0, location=0, statusbars=0, menubars=0, resizable=0, width=500, height=300 left = 100, top = 100");
      }
```

```
        function popUp_thread(id)
        {
            var url="viewthread.php?id="+id;
            window.open(url, "poppage", "toolbars=0, scrollbars=1, location=0, statusbars=0, menubars=0, resizable=0, width=500, height=350 left = 100, top = 100");
        }
    </script>
  </head>
  <body bgcolor="#CCCC99">
    <table width="600">
        <tr><td><h1>Discussion Forum</h1></td></tr>
        <tr><td><hr></td></tr>
        <tr><td><a href="javascript:popUp_create()">Create new discussion</a>        <a href="javascript:popUp_subscribe()">Subscribe to discussion</a></td></tr>
        <tr><td><hr></td></tr>
    </table>
    <?
        //This function view all the posts that are the first one in a discussion.
        function show()
        {
            global $a;
            $a=1;
            //This first row here opens a database connection.
            $conn=pg_connect("host=localhost dbname=odin user=examen1 password=DYWRBK");
            $sql="SELECT * FROM discussion WHERE parent=0";
            $result=pg_exec($conn, $sql);
            $num=pg_numrows($result);
            echo "<dl>";
            for($i=0; $i<$num; $i++)
            {
```

```
            $rad=pg_fetch_row($result, $i);
            echo "<dt><input type='hidden' name='id'
value='".$rad[0]."'><b>".$rad[2]."</b>  ".$rad[3]."  <i>".$rad[5].
"</i>  <a href='javascript:popUp_read($rad[0])'>Read</a>  <a
href='javascript:popUp_reply($rad[0])'>Reply</a>  <a
href='javascript:popUp_thread($rad[0])'>View thread</a>";
            showreply($rad[0], $a);
        }
        echo "</dl>";
    }
    //This function will view all posts that are an answer to another post.
    function showreply($id, $a)
    {
        //This first row here opens a database connection.
        $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
        $sql2="SELECT * FROM discussion WHERE parent=$id";
        $result2=pg_exec($conn2, $sql2);
        if($a==1)
        {
            $num=pg_numrows($result2);
            for($i=0; $i<$num; $i++)
            {
                $rad=pg_fetch_row($result2, $i);
                echo "<dd><input type='hidden' name='id'
value='".$rad[0]."'>".$rad[2]."  ".$rad[3]."  <i>".$rad[5]."</i>&nb
sp; <a href='javascript:popUp_read($rad[0])'>Read</a>  <a
href='javascript:popUp_reply($rad[0])'>Reply</a>";
                $a++;
                showreply($rad[0], $a);
            }
        }
        else
```

```
            {
               $num=pg_numrows($result2);
               for($i=0; $i<$num; $i++)
               {
                  $rad=pg_fetch_row($result2, $i);
                  echo "<dl>";
                  echo "<dt><dd><input type='hidden' name='id'
value='".$rad[0]."'>".$rad[2]."  ".$rad[3]."  <i>".$rad[5]."</i>&nb
sp; <a href='javascript:popUp_read($rad[0])'>Read</a>  <a
href='javascript:popUp_reply($rad[0])'>Reply</a>";
                  $a++;
                  showreply($rad[0], $a);
               }
               echo "</dl>";
            }
         }
         show();
      ?>
   </body>
</html>
```

create.php

```
<html>
   <head>
      <title>Create discussion</title>
      <link rel="stylesheet" type="text/css" href="style.css">
      <script language="javascript">
         //This function puts the first textfield onfocus.
         function focusa()
         {
            document.create.title.focus();
         }
```

```
        //This function refreshes the parent-page in the main-frame.
        function stop()
        {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/discussion.php
";
        }
      </script>
   </head>
   <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
      <table width="300">
         <tr><td align="center"><h2>Create a new discussion here</h2></td></tr>
         <tr><td><hr></td></tr>
         <tr>
            <td>
               <?
                  function start()
                  {
                     ?>
                     <!-- This form takes care of the data that the user fill in -->
                     <form method="post" name="create" action="<?=$PHP_SELF?>">
                        <table align="center">
                           <tr><td>Titel</td><td><input type="text" name="title"></td></tr>
                           <tr><td valign="top">Text</td><td><textarea name="text" rows="5"
cols="40"></textarea></td></tr>
                           <tr><td>First name</td><td><input type="text"
name="fname"></td></tr>
                           <tr><td heighet="10" colspan="2"> </td></tr>
                           <tr><td>  </td><td><input type="submit" name="ok"
value="OK"></td></tr>
                        </table>
                     </form>
                     <?
```

```
                }
                //If the submit-button (called ok) are pressed down then the create-function
are called.
                if($ok)
                {
                   create();
                }
                //This function takes care of the data that the user have written and put it in to
the database.
                function create()
                {
                   global $title;
                   global $text;
                   global $fname;
                   $date=date("Y-m-d");
                   $parent=0;
                   $root=0;
                   //This first row here opens a database connection.
                   $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
                   $sql="INSERT INTO discussion (parent, title, date, text, fname, root)
VALUES ('$parent', '$title', '$date', '$text', '$fname', '$root')";
                   $result=pg_exec($conn, $sql);
                   echo "<script language='javascript'> window.close();</script>";
                }
                start();
             ?>
           </td>
         </tr>
      </table>
   </body>
</html>
```

showpost.php

```
<html>
   <head>
      <title>Show post</title>
      <link rel="stylesheet" type="text/css" href="style.css">
   </head>
   <body bgcolor="#CCCC99">
      <table width="450">
         <tr><td align="center"><h2>Read a contribution here</h2></td></tr>
         <tr><td><hr></td></tr>
         <tr>
            <td>
               <?
                  //This function views the post that has been choosen at the discussion page.
                  function showpost()
                  {
                     global $id;
                     //This first row here opens a database connection
                     $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
                     $sql="SELECT * FROM discussion WHERE id=$id";
                     $result=pg_exec($conn, $sql);
                     $num=pg_numrows($result);
                     $rad=pg_fetch_row($result, $i);
                     echo "<table><tr>";
                     echo "<td>Title:</td><td><input type='hidden' name='id'
value='".$rad[0]."'>".$rad[2]."</td></tr>";
                     echo "<tr><td>Date:</td><td>".$rad[3]."</td></tr>";
                     echo "<tr><td>Text:</td><td>".$rad[4]."</td></tr>";
                     echo "<tr><td>First name:</td><td>".$rad[5]."</td></tr>";
```

```
                    echo "<tr><td colspan='2'><hr width='450'></td></tr>";
                    echo "<tr><td><a href='reply.php?id=$rad[0]'>Reply</a></td><td><a
href='javascript:window.close()'>Close</a></td></tr></table>";
                }
                showpost();
            ?>
          </td>
        </tr>
      </table>
  </body>
</html>
```

viewthread.php

```
<html>
  <head>
    <title>View a whole thread</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body bgcolor="#CCCC99">
    <table width="450">
      <tr><td colspan="2" align="center"><h2>Read a whole thread here</h2></td></tr>
      <tr><td colspan="2"><hr></td></tr>
      <tr>
        <td>
    <?
      //This function view the first post in the discussion.
      function show()
      {
        global $id;
        //This first row here opens a database connection.
```

```
        $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
        $sql="SELECT * FROM discussion WHERE id=$id";
        $result=pg_exec($conn, $sql);
        $num=pg_numrows($result);
        echo "<table border='0'>";
        for($i=0; $i<$num; $i++)
        {
            $rad=pg_fetch_row($result, $i);
            echo "<tr><td colspan='2'><input type='hidden' name='id'
value='".$rad[0]."'><b>".$rad[2]."</b></td></tr>";
            echo "<tr><td colspan='2'>".$rad[3]."</td></tr>";
            echo "<tr><td colspan='2'>".$rad[4]."</td></tr>";
            echo "<tr><td width='3'> </td><td>".$rad[5]."</td></tr>";
            echo "<tr><td colspan='2'><a
href='reply.php?id=$rad[0]'>Reply</a></td></tr>";
            echo "<tr><td colspan='2'><hr width='450'></td></tr>";
            showreply($rad[0]);
        }
        echo "</table>";
    }
    //This function views all the other posts in the discussion.
    function showreply($id)
    {
        //This first row here opens a database connection.
        $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
        $sql2="SELECT * FROM discussion WHERE parent=$id";
        $result2=pg_exec($conn2, $sql2);
        $num=pg_numrows($result2);
        for($i=0; $i<$num; $i++)
        {
```

```
                $rad=pg_fetch_row($result2, $i);
                echo "<tr><td colspan='2'><input type='hidden' name='id'
value='".$rad[0]."'><b>".$rad[2]."</b></td></tr>";
                echo "<tr><td colspan='2'>".$rad[3]."</td></tr>";
                echo "<tr><td colspan='2'>".$rad[4]."</td></tr>";
                echo "<tr><td width='3'> </td><td>".$rad[5]."</td></tr>";
                echo "<tr><td colspan='2'><a
href='reply.php?id=$rad[0]'>Reply</a></td></tr>";
                echo "<tr><td colspan='2'><hr width='450'></td></tr>";
                showreply($rad[0]);
            }
        }
        show();
    ?>
            </td>
        </tr>
        <tr><td><a href='javascript:window.close()'>Close</a></td></tr>
    </table>
  </body>
</html>
```

reply.php

```
<html>
  <head>
    <title>Reply</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="javascript">
      //This function puts the first textfield onfocus.
      function focusa()
      {
        document.reply.text.focus();
```

```
        }
        //This function refreshes the parent-page in the main-frame.
        function stop()
        {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/discussion.php
";
        }
      </script>
   </head>
   <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
      <table width="490">
        <tr><td align="center"><h2>Reply to a contribution here</h2></td></tr>
        <tr><td><hr></td></tr>
        <tr>
          <td>
            <?
              //This function is called upon when this page are executed.
              function reply()
              {
                global $id;
                global $title;
                global $fname;
                //This first row here opens a database connection.
                $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
                $sql="SELECT * FROM discussion WHERE id=$id";
                $result=pg_exec($conn, $sql);
                $rad=pg_fetch_row($result, $sql);
                ?>
                <form action="<?=$PHP_SELF?>" name="reply" method="POST">
                  <table border=0>
```

```
                    <tr><td align="right">Title</td><td><input type="text" name="title"
value="Re: <? echo $rad[2] ?>"></tr>
                    <tr><td valign="top" align="right">Text</td><td><textarea
name="text" rows="6" cols="40"></textarea>
                    <tr><td valign="top" align="right">First name</td><td><input
type="text" name="fname"></td></tr>
                    <tr><td> </td><td><input type="submit" name="ok2" value="
OK  "></tr>
                  </table>
                </form>
                <?
              }
              //If the submit-button (called ok2) are pressed down then the create_reply-
function are called.
              if($ok2)
              {
                create_reply();
              }
              //This function takes care of the data that the user have written and put it in to
the database.
              function create_reply()
              {
                global $id;
                global $title;
                global $text;
                global $fname;
                global $parent;
                global $root;
                $date=date("Y-m-d");
                $parent=$id;
                parent($id);
                //This first row here opens a database connection.
```

```
                $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
                $sql="INSERT INTO discussion (parent, title, date, text, fname, root)
VALUES ('$parent', '$title', '$date', '$text', '$fname', '$root')";
                $result=pg_exec($conn, $sql);
                echo "<script language='javascript'> window.close();</script>";
            }
            //This function determinate who the parent is for the contribution.
            function parent($parent)
            {
                global $root;
                if($parent!=0)
                {
                  //This first row here opens a database connection.
                  $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
                  $sql="SELECT * FROM discussion WHERE id=$parent";
                  $result=pg_exec($conn, $sql);
                  $rad=pg_fetch_row($result, $sql);
                  $root=$rad[0];
                  parent($rad[1]);
                }
            }
            reply();
          ?>
        </td>
      </tr>
    </table>
  </body>
</html>
```

subscribe.php

```
<html>
  <head>
    <title>Subscribe to a discussion</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body bgcolor="#CCCC99">
    <table width="300">
      <tr><td align="center"><h2>Subscribe to a discussion here</h2></td></tr>
      <tr><td><hr></td></tr>
      <tr>
        <td>
          <!-- This form is for letting the user choose a discussion to subscribe on. -->
          <form method="post" action="<?=$PHP_SELF?>">
            <table align="center">
              <tr><td>Discussion</td><td><select name="discussion">
                  <option selected>-- choose here --</option>
                    <?
                      //This first row here opens a database connection
                      $conn=pg_connect("host=localhost dbname=odin
user=examen1 password=DYWRBK");
                      $sql="SELECT * FROM discussion WHERE parent=0";
                      $result=pg_exec($conn, $sql);
                      $num=pg_numrows($result);
                      for($i=0; $i<$num; $i++)
                      {
                        $rad=pg_fetch_row($result, $i);
                        echo "<option>".$rad[2]."</option>";
                      }
                    ?>
                  </select>
                </td>
```

```
                    </tr>
                    <tr><td>E-mail</td><td><input type="text" name="email"></td></tr>
                    <tr><td heighet="10" colspan="2"> </td></tr>
                    <tr><td>  </td><td><input type="submit" name="subscribe"
value="Subscribe"></td></tr>
                </table>
            </form>
        </td>
    </tr>
</table>
<?
    //If the submit-button (called subscribe) are pressed down, then the addpren-function
will be called upon.
    if($subscribe)
    {
        addpren();
    }
    //This function creates a new post in the database table subscribe.
    function addpren()
    {
        global $discussion;
        global $disc_id;
        global $email;
        if($discussion=='-- choose here --' || $email=='')
        {
            echo "<script language='javascript'> window.alert('You must choose a discussion
and fill in the email field!')</script>";
        }
        else
        {
            //This first row here opens a database connection.
```

```
            $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
            $sql2="SELECT id FROM discussion WHERE title='$discussion'";
            $result2=pg_exec($conn2, $sql2);
            $num=pg_numrows($result2);
            $rad=pg_fetch_row($result2, $i);
            $disc_id=$rad[0];
            //This first row here opens a database connection.
            $conn3=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
            $sql3="INSERT INTO subscribe (disc_id, email) VALUES ('$disc_id',
'$email')";
            $result3=pg_exec($conn3, $sql3);
            echo "<script language='javascript'> window.close(); </script>";
        }
      }
    ?>
  </body>
</html>
```

sendmail.php

```
<?
  //This function are called when this file executes.
  function start()
  {
     //This first row here opens a database connection.
    $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
    $sql="SELECT * FROM discussion WHERE parent=0";
    $result=pg_exec($conn, $sql);
    $num=pg_numrows($result);
    echo "<table border='0'>";
```

```
    for($i=0; $i<$num; $i++)
    {
      $rad=pg_fetch_row($result, $i);
      echo "<tr><td><input type='hidden' name='id' value='".$rad[0]."'><input
type='hidden' name='parent' value='".$rad[1]."'><input type='hidden' name='title'
value='".$rad[2]."'><input type='hidden' name='date' value='".$rad[3]."'><input type='hidden'
name='text' value='".$rad[4]."'><input type='hidden' name='fname' value='".$rad[5]."'><input
type='hidden' name='root' value='".$rad[6]."'></td></tr>";
      show($rad[0]);
    }
    echo "</table>";
  }
  //This function will get all posts that are an answer to another post.
  function show($id)
  {
    //This first row here opens a database connection.
    $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
    $sql2="SELECT * FROM discussion WHERE root=$id";
    $result2=pg_exec($conn2, $sql2);
    $num=pg_numrows($result2);
    for($i=0; $i<$num; $i++)
    {
      $rad=pg_fetch_row($result2, $i);
      echo "<tr><td><input type='hidden' name='id' value='".$rad[0]."'><input
type='hidden' name='parent' value='".$rad[1]."'><input type='hidden' name='title'
value='".$rad[2]."'><input type='hidden' name='date' value='".$rad[3]."'><input type='hidden'
name='text' value='".$rad[4]."'><input type='hidden' name='fname' value='".$rad[5]."'><input
type='hidden' name='root' value='".$rad[6]."'></td></tr>";
      show($rad[0]);
    }
    $today=date("Y-m-d");
```

```
//If tha date in the table matches todays date and the root-number or id-number matches the number on the sixth place in the row at the table then this if-part will executes, that is send a mail to the actors that a new contribution has been made.
    if($today==$rad[3] && ($root==$rad[6] || $id==$rad[6]))
    {
      //This first row here opens a database connection.
      $conn2=pg_connect("host=localhost dbname=odin user=examen1 password=DYWRBK");
      $sql2="SELECT * FROM subscribe WHERE disc_id=$rad[6]";
      $result2=pg_exec($conn2, $sql2);
      $num=pg_numrows($result2);
      for($i=0; $i<$num; $i++)
      {
        $rad2=pg_fetch_row($result2, $i);
        echo "<td><input type='hidden' name='id' value='".$rad2[0]."'><input type='hidden' name='id' value='".$rad2[2]."'></td>";
        mail($rad2[2], "New contribution", "A new contribution has been made in the discussion (".$rad[2].").", "From: Discussion Forum");
      }
    }
  }
  start();
?>
```

autodelete.php

```
<?
  //This function are called when this file executes.
  function start()
  {
    //This first row here opens a database connection.
    $conn=pg_connect("host=localhost dbname=odin user=examen1 password=DYWRBK");
```

```
    $sql="SELECT * FROM discussion WHERE parent=0";
    $result=pg_exec($conn, $sql);
    $num=pg_numrows($result);
    echo "<table border='0'>";
    for($i=0; $i<$num; $i++)
    {
      $rad=pg_fetch_row($result, $i);
      echo "<tr><td><input type='hidden' name='id' value='".$rad[0]."'><input
type='hidden' name='parent' value='".$rad[1]."'><input type='hidden' name='title'
value='".$rad[2]."'><input type='hidden' name='date' value='".$rad[3]."'><input type='hidden'
name='text' value='".$rad[4]."'><input type='hidden' name='fname' value='".$rad[5]."'><input
type='hidden' name='root' value='".$rad[6]."'></td></tr>";
      show($rad[0]);
    }
    echo "</table>";
  }
  //This function will get all posts that are an answer to another post.
  function show($id)
  {
    $year=date("Y");
    $month=date("m")-1;
    $day=date("d");
    $lastdate=$year."-".$month."-".$day;
    //This first row here opens a database connection.
    $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
    $sql2="SELECT * FROM discussion WHERE root=$id";
    $result2=pg_exec($conn2, $sql2);
    $num=pg_numrows($result2);
    for($i=0; $i<$num; $i++)
    {
      $rad=pg_fetch_row($result2, $i);
```

```
        echo "<tr><td><input type='hidden' name='id' value='".$rad[0]."'><input
type='hidden' name='parent' value='".$rad[1]."'><input type='hidden' name='title'
value='".$rad[2]."'><input type='hidden' name='date' value='".$rad[3]."'><input type='hidden'
name='text' value='".$rad[4]."'><input type='hidden' name='fname' value='".$rad[5]."'><input
type='hidden' name='root' value='".$rad[6]."'></td></tr>";
        //If the date in the database table is from last month that entire discussion will be
removed from the table.
        if($lastdate==$rad[3])
        {
          $conn3=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
          $sql3="DELETE FROM discussion WHERE id=$rad[6] OR root=$rad[6]";
          $result3=pg_exec($conn3, $sql3);
        }
        show($rad[0]);
      }
   }
   start();
?>
```

Felhantering

error.php

```
<html>
   <head>
      <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
      <title>Known data problems</title>
      <link rel="stylesheet" type="text/css" href="style.css">
      <script language="JavaScript">
        //All of these functions opens a popup-window. One of them also sends a variable-
value to the next window.
        function popUp_addproblem()
        {
```

```
            window.open("addproblem.php", "poppage", "toolbars=0, scrollbars=0,
location=0, statusbars=0, menubars=0, resizable=0, width=450, height=400 left = 100, top =
100");
        }
        function popUp_search()
        {
            window.open("search.php", "poppage", "toolbars=0, scrollbars=1, location=0,
statusbars=0, menubars=0, resizable=0, height=400 left = 100, top = 100");
        }
        function popUp_delete(id_err)
        {
            var url="deleteerror.php?id_err="+id_err;
            window.open(url, "poppage", "toolbars=0, scrollbars=0, location=0, statusbars=0,
menubars=0, resizable=0, width=360, height=260 left = 100, top = 100");
        }
    </script>
</head>
<body bgcolor="#CCCC99">
    <table width="1030" border="0">
        <tr><td colspan="9"><h1>Known data problems</h1></td></tr>
        <tr><td colspan="9"><hr></td></tr>
        <tr><td colspan="9"><a href="javascript:popUp_addproblem()">Add a
problem</a>        <a
href="javascript:popUp_search()">Search</a></td></tr>
        <tr><td colspan="9"><hr></td></tr>
        <tr>
            <th align="left" valign="top">Source</th>
            <th align="left" valign="top">Observation date</th>
            <th align="left" valign="top">Type of error</th>
            <th align="left" valign="top">Concerning</th>
            <th align="left" valign="top">Misc. comment</th>
            <th align="left" valign="top">More information</th>
```

```
        <th align="left" valign="top">Entry-date</th>
        <th align="left" valign="top">Delete</th>
    </tr>
    <?
        //This function view all the posts from the error-table and oreder them first by source and second by observationdate.
        function show()
        {
            //This first row here opens a database connection.
            $conn=pg_connect("host=localhost dbname=odin user=examen1 password=DYWRBK");
            $sql="SELECT * FROM error ORDER BY source, obsdate DESC";
            $result=pg_exec($conn, $sql);
            $num=pg_numrows($result);
            for($i=0; $i<$num; $i++)
            {
                $rad=pg_fetch_row($result, $i);
                echo "<form method='post' action='$PHP_SELF'>";
                echo "<tr>";
                echo "<td><input type='hidden' name='id_err' value='".$rad[0]."'>".$rad[1]."</td>";
                echo "<td>".$rad[2]."</td>";
                echo "<td>".$rad[3]."</td>";
                echo "<td>".$rad[4]."</td>";
                echo "<td>".$rad[5]."</td>";
                echo "<td><a href='http://193.10.236.220/~examen1/files/".$rad[7]."' target='new'>".$rad[7]."</a></td>";
                echo "<td>".$rad[6]."</td>";
                echo "<td><input type='submit' onClick='javascript:popUp_delete($rad[0])' name='delete' value='Delete'></td></tr></form>";
            }
        }
```

```
            show();
        ?>
    </table>
  </body>
</html>
```

addproblem.php

```
<html>
  <head>
    <title>Add result</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="javascript">
      //This function puts the first textfield onfocus.
      function focusa()
      {
        document.add.source.focus();
      }
      //This function refreshes the parent-page in the main-frame.
      function stop()
      {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/error.php";
      }
    </script>
  </head>
  <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
    <table width="420">
      <tr><td align="center"><h2>Add your discovered problem here</h2></td></tr>
      <tr><td><hr></td></tr>
      <tr>
        <td>
```

```
<!-- Here is a form for the user to fill in and it also makes it possible to upload a file -->
<form enctype="multipart/form-data" name="add" action="<?=$PHP_SELF?>" method=POST>
<table align="center">
<tr><td>Source</td><td><input type="text" name="source"></td></tr>
<tr><td>Observation date</td><td><input type="text" name="obsdate"></td></tr>
<tr><td>Type of error</td><td><input type="text" name="error"></td></tr>
<tr><td>Concerning</td><td><input type="text" name="concern"></td></tr>
<tr><td>Misc. comment</td><td><input type="text" name="comment"></td></tr>
<tr><td>Signature (Max 5 letters and don't forget the signature)</td><td><input type="text" name="sign"></td></tr>
<tr><td>Upload file</td><td><input type="hidden" name="MAX_FILE_SIZE"><input type="file" name="file">  <input type="submit" name="submit" value=" OK "></td></tr>
</table>
</form>
</td>
</tr>
<?
//This function is for write all the form-data to the database
function insert()
{
global $id_err;
global $source;
global $obsdate;
global $error;
global $concern;
global $comment;
```

```
            global $date;
            global $file;
            global $sign;
            $date=date("Y-m-d");
            //This first row here opens a database connection
            $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
            $sql="INSERT INTO error (source, obsdate, error, concern, comment, date,
file, sign) VALUES ('$source', '$obsdate', '$error', '$concern', '$comment', '$date', '$file',
'$sign')";
            $result=pg_exec($conn, $sql);
            echo "<script language='javascript'> window.close();</script>";
        }
        //If the submit-button are pressed down the file will be uploaded before the insert-
function are called upon.
        if($submit)
        {
            $tmpfile=$HTTP_POST_FILES["file"]["tmp_name"];
            $file=$HTTP_POST_FILES["file"]["name"];
            move_uploaded_file("$tmpfile", "files/$file");
            insert();
        }
      ?>
    </table>
  </body>
</html>
```

deleteerror.php

```
<html>
  <head>
    <title>Delete</title>
    <link rel="stylesheet" type="text/css" href="style.css">
```

```
    <script language="javascript">
      //This function puts the first textfield onfocus.
      function focusa()
      {
        document.error.sign.focus();
      }
      //This function refreshes the parent-page in the main-frame.
      function stop()
      {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/error.php";
      }
    </script>
  </head>
  <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
    <table width="300">
      <tr><td colspan="2" align="center"><h2>Delete a known problem
here</h2></td></tr><tr><td colspan="2"><hr></td></tr>
      <tr>
        <td>
          <?
            //This function view th form that the user should fill in.
            function show()
            {
              echo "<form method='post' name='error' action='$PHP_SELF'>";
              echo "<tr><td>Signature</td><td><input type='text'
name='sign'></td></tr>";
              echo "<tr><td> </td><td><input type='submit' name='submit'
value=' OK '></td></tr></form>";
            }
          ?>
        </td>
      </tr>
```

```
        <?
            //If the submit-button (called submit) are pressed down then the delete-function are
called.
            if($submit)
            {
               delete();
            }
            //This function remowes a post in the database.
            function delete()
            {
               global $id_err;
               global $source;
               global $obsdate;
               global $error;
               global $concern;
               global $comment;
               global $date;
               global $file;
               global $sign;
               //This first row here opens a database connection.
               $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
               $sql2="DELETE FROM error WHERE id_err='$id_err' AND sign='$sign'";
               pg_exec($conn2, $sql2);
               echo "<script language='javascript'> window.close();</script>";
            }
            show();
        ?>
      </table>
   </body>
</html>
```

search.php

```
<html>
   <head>
      <title>Search</title>
      <link rel="stylesheet" type="text/css" href="style.css">
      <script language="javascript">
         //This function puts the first textfield onfocus.
         function focusa()
         {
            document.search.source.focus();
         }
      </script>
   </head>
   <body bgcolor="#CCCC99" onLoad=focusa()>
      <table width="300" border="0">
         <tr><td colspan="2" align="center"><h2>Search for a problem here</h2></td></tr>
        <tr><td colspan="2"><hr></td></tr>
         <?
            //If the submit-button (called serach) are pressed down the search-function are
called upon.
            if($search)
            {
               search();
            }
            //Otherwise this form will be viewed.
            else
            {
               ?>
               <form method="post" name="search" action="<?=$PHP_SELF?>">
                  <tr><td>Source</td><td><input type="text" name="source"></td></tr>
```

```
            <tr><td>Type of error</td><td><input type="text" name="error"></td></tr>
            <tr><td>Concern</td><td><input type="text" name="concern"></td></tr>
            <tr><td> </td><td><input type="submit" name="search"
value="Search"></td></tr>
        </form>
        <?
    }
    //This function search in the database for posts that matches with the users
requriments.
    function search()
    {
        global $source;
        global $error;
        global $concern;
        //This first row here opens a database connection
        $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
        $sql="SELECT * FROM error WHERE source LIKE '$source%' AND error
LIKE '$error%' AND concern LIKE '$concern%'";
        $result=pg_exec($conn, $sql);
        $num=pg_numrows($result);
        if($num!=0)
        {
            ?>
            <table border="0">
                <tr>
                    <th align="left" valign="top">Source</th>
                    <th align="left" valign="top">Observation date</th>
                    <th align="left" valign="top">Type of error</th>
                    <th align="left" valign="top">Concerning</th>
                    <th align="left" valign="top">Misc. comment</th>
                    <th align="left" valign="top">More information</th>
```

```
                    <th align="left" valign="top">Entry-date</th>
                </tr>
                <?
                //$num=pg_numrows($result);
                for($i=0; $i<$num; $i++)
                {
                    $rad=pg_fetch_row($result, $i);
                    echo "<tr>";
                    echo "<td>".$rad[1]."</td>";
                    echo "<td>".$rad[2]."</td>";
                    echo "<td>".$rad[3]."</td>";
                    echo "<td>".$rad[4]."</td>";
                    echo "<td>".$rad[5]."</td>";
                    echo "<td><a href='http://193.10.236.220/~examen1/files/".$rad[7]."'
target='new'>".$rad[7]."</a></td>";
                    echo "<td>".$rad[6]."</td>";
                    echo "</tr>";
                }
                echo "<tr><td height='10'> </td></tr>";
                echo "<tr><td colspan='7'><a href='search.php'>New
search</a></td></tr>";
                echo "<tr><td colspan='7'><a
href='javascript:window.close()'>Close</a></td></tr>";
                echo "</table>";
            }
        else
        {
            echo "<tr><td>No matches were found!</td></tr>";
            echo "<tr><td height='10'> </td></tr>";
            echo "<tr><td colspan='7'><a href='search.php'>New
search</a></td></tr>";
```

```
                    echo "<tr><td colspan='7'><a
href='javascript:window.close()'>Close</a></td></tr>";
                }
            }
        ?>
    </table>
  </body>
</html>
```

Samordning

coordinate.php

```
<html>
  <head>
    <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
    <title>Data processing coordination</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="JavaScript">
      //All these functions are used for opening popup-windows.
      function popUp_addsource()
      {
        window.open("addsource.php", "poppage", "toolbars=0, scrollbars=0, location=0,
statusbars=0, menubars=0, resizable=0, width=370, height=300 left = 100, top = 100");
      }
      function popUp_signUp(id_co)
      {
        //This variable is used to make it possible to send a variable-value to the popup-
window.
        var url="signup.php?id_co="+id_co;
        var signup = window.open(url, "poppage", "toolbars=0, scrollbars=0, location=0,
statusbars=0, menubars=0, resizable=0, width=360, height=260 left = 100, top = 100");
      }
      function popUp_offSign(id_co)
```

```
        {
            //This variable is used to make it possible to send a variable-value to the popup-window.
            var url="offsign.php?id_co="+id_co;
            window.open(url, "poppage", "toolbars=0, scrollbars=0, location=0, statusbars=0, menubars=0, resizable=0, width=360, height=260 left = 100, top = 100");
        }
    </script>
  </head>
  <body bgcolor="#CCCC99">
    <table width="600" border="0">
      <tr><td colspan="7"><h1>Data processing coordination</h1></td></tr>
      <tr><td colspan="7"><hr></td></tr>
      <tr><td colspan="7"><a href="javascript:popUp_addsource()">Add a source</a></td></tr>
      <tr><td colspan="7"><hr></td></tr>
      <tr>
        <th align="left">Source</th>
        <th align="left">Obsdate</th>
        <th align="left">Line</th>
        <th align="left">Name</th>
        <th> </th>
        <th align="left">Sign up</th>
        <th align="left">Remove</th>
      </tr>
      <?
        //This function view all of the posts in the coordinate-table and order them by the source.
        function show()
        {
          //This first row here opens a database connection.
```

```
            $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
            $sql="SELECT * FROM coordinate ORDER BY source";
            $result=pg_exec($conn, $sql);
            $num=pg_numrows($result);
            for($i=0; $i<$num; $i++)
            {
               $rad=pg_fetch_row($result, $i);
               echo "<form method='post' action='$PHP_SELF'>";
               echo "<tr>";
               echo "<td><input type='hidden' name='id_co'
value='".$rad[0]."'>".$rad[1]."</td>";
               echo "<td>".$rad[2]."</td>";
               echo "<td>".$rad[3]."</td>";
               echo "<td>".$rad[4]."</td>";
               echo "<td>".$rad[5]."</td>";
               echo "<td><input type='submit' onClick='javascript:popUp_signUp($rad[0])'
name='signup' value='Sign up'></td>";
               echo "<td><input type='submit' onClick='javascript:popUp_offSign($rad[0])'
name='offsign' value='Remove'></td></tr></form>";
            }
         }
         show();
      ?>
   </table>
</body>
</html>
```

addsource.php

```
<html>
   <head>
      <title>Add source</title>
```

```
        <link rel="stylesheet" type="text/css" href="style.css">
        <script language="javascript">
            //This function puts the first textfield onfocus.
            function focusa()
            {
                document.add.source.focus();
            }
            //This function refreshes the parent-page in the main-frame.
            function stop()
            {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/coordinate.php
";
            }
        </script>
    </head>
    <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
        <table width="320">
            <tr><td align="center"><h2>Add a new source here</h2></td></tr>
            <tr><td><hr></td></tr>
            <tr>
                <td>
                    <!-- This form takes care of the data that the user fill in -->
                    <form method="post" name="add" action="<?=$PHP_SELF?>">
                        <table align="center">
                            <tr><td>Source</td><td><input type="text" name="source"></td></tr>
                            <tr><td>Observation date (YYYY-MM-DD or write
"All")</td><td><input type="text" name="obsdate"></td></tr>
                            <tr><td>Line</td><td><select name="line">
                                    <option selected>-- choose here --</option>
                                    <option>H2O</option>
                                    <option>H2O-18</option>
                                    <option>C-13-O</option>
```

```
                        <option>C18O</option>
                        <option>CI</option>
                        <option>O2</option>
                        <option>NH3</option>
                        <option>PH</option>
                    </select>
                </td></tr>
                <tr><td heighet="10" colspan="2"> </td></tr>
                <tr><td> </td><td><input type="submit" name="submit"
value="OK">  <input type="reset" name="reset" value="Clear"></td></tr>
            </table>
        </form>
    </td>
</tr>
<?
    //This function is for write all the form-data to the database
    function insert()
    {
        global $source;
        global $obsdate;
        global $line;
        //This first row here opens a database connection
        $conn=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
        $sql="INSERT INTO coordinate (source, obsdate, line) VALUES ('$source',
'$obsdate', '$line')";
        $result=pg_exec($conn, $sql);
        echo "<script language='javascript'> window.close();</script>";
    }
    //If the submit-button are pressed down the insert-function are called upon
    if($submit)
    {
```

```
                insert();
            }
        ?>
    </table>
  </body>
</html>
```

signup.php

```
<html>
  <head>
    <title>Sign up</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="javascript">
      //This function puts the first textfield onfocus.
      function focusa()
      {
        document.signup.fname.focus();
      }
      //This function refreshes the parent-page in the main-frame.
      function stop()
      {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/coordinate.php
";
      }
    </script>
  </head>
  <body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
    <table width="300">
      <tr><td colspan="2" align="center"><h2>Sign up here</h2></td></tr>
      <tr><td colspan="2"><hr></td></tr>
      <tr>
```

```
<td>
  <?
    //This function view the form that the user should fill in.
    function show()
    {
      echo "<form method='post' name='signup' action='$PHP_SELF'>";
      echo "<tr><td>First name</td><td><input type='text'
name='fname'></td></tr>";
      echo "<tr><td>Sir name</td><td><input type='text'
name='sname'></td></tr>";
      echo "<tr><td>Signature (Don't forget the signature and max 5
letters)</td><td><input type='text' name='sign'></td></tr>";
      echo "<tr><td> </td><td><input type='submit' name='submit'
value='  OK  '></td></tr></form>";
    }
  ?>
</td>
</tr>
<?
  //If the submit-button (called submit) are pressed down the update-function are
called upon.
  if($submit)
  {
    update();
  }
  //This function updates a post in the database table.
  function update()
  {
    global $id_co;
    global $source;
    global $obsdate;
    global $line;
```

```
            global $fname;
            global $sname;
            global $sign;

            //If one of the textfileds aren't filled in this if-part will executes.
            if($fname=='' || $sname=='' || $sign=='')
            {
               echo "<script language='javascript'> window.alert('You must fill in all three
fields!')</script>";
            }
            else
            {
               //This first row here opens a database connection.
               $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
               $sql2="UPDATE coordinate SET fname='$fname', sname='$sname',
sign='$sign' WHERE id_co='$id_co'";
               pg_exec($conn2, $sql2);
               echo "<script language='javascript'> window.close(); </script>";
            }
         }
         show();
      ?>
   </table>
</body>
</html>
```

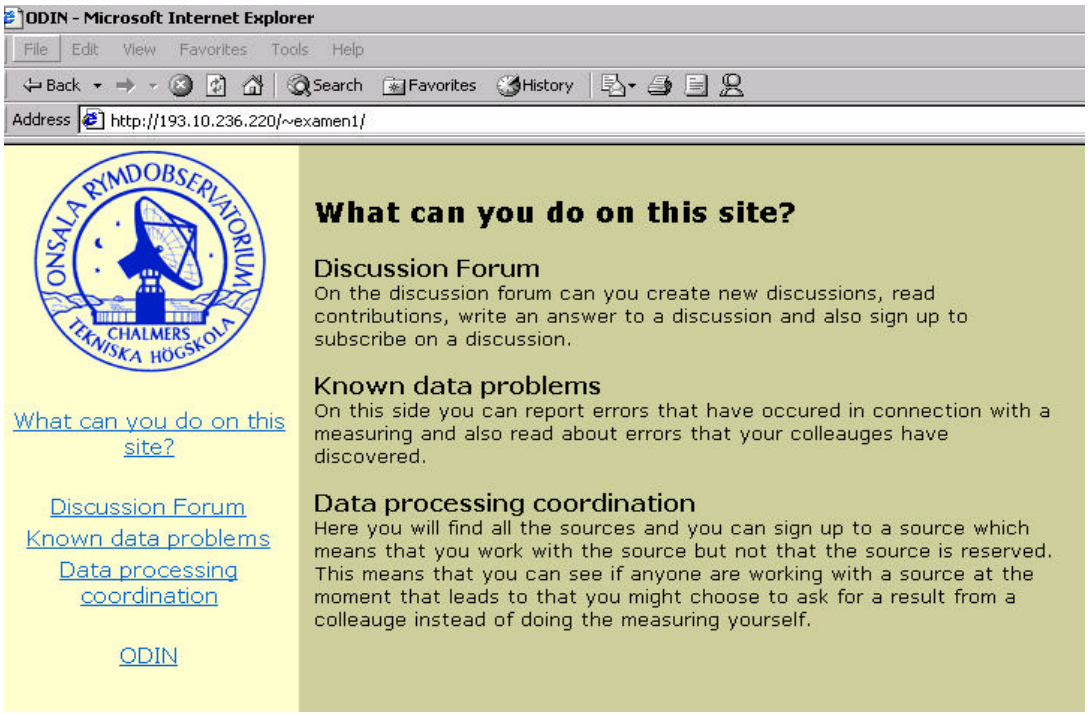
offsign.php

```
<html>
   <head>
      <title>Remove</title>
      <link rel="stylesheet" type="text/css" href="style.css">
```

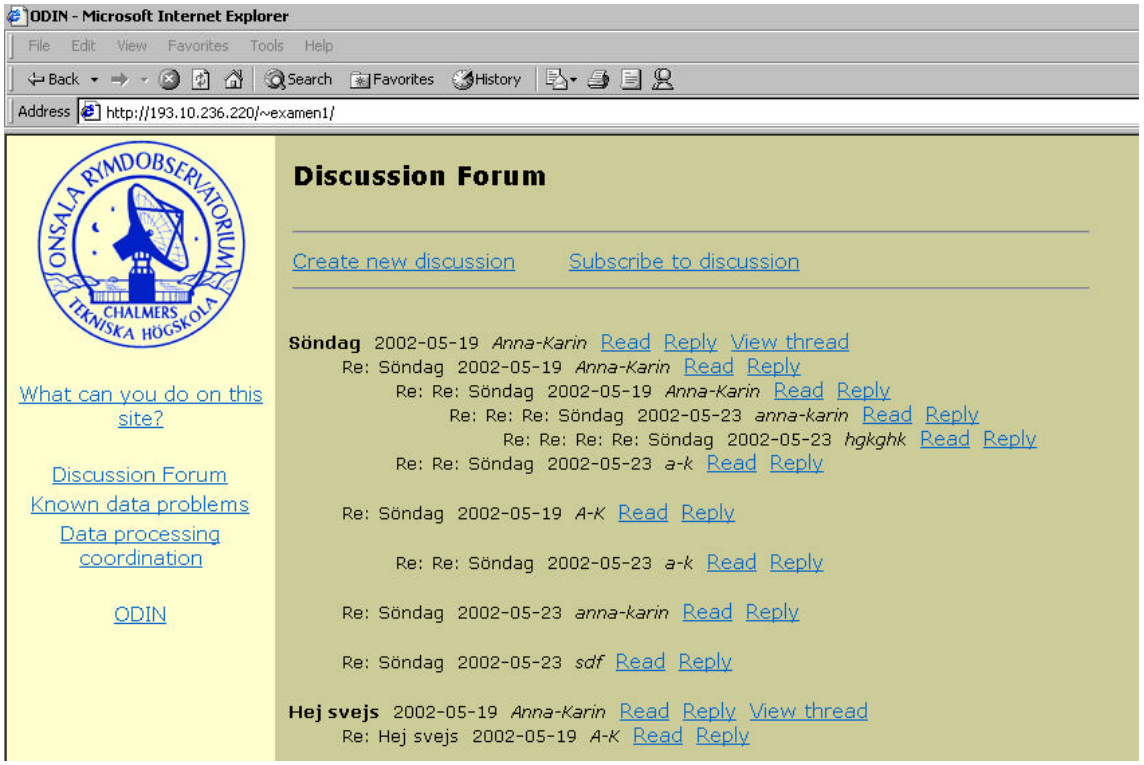
```
<script language="javascript">
    //This function puts the first textfield onfocus.
    function focusa()
    {
        document.offsign.sign.focus();
    }
    //This function refreshes the parent-page in the main-frame.
    function stop()
    {
opener.parent.main.document.location.href="http://193.10.236.220/~examen1/coordinate.php
";
    }
</script>
</head>
<body bgcolor="#CCCC99" onLoad=focusa() onUnLoad=stop()>
<table width="300">
    <tr><td colspan="2" align="center"><h2>Delete yourself from a source
here</h2></td></tr>
    <tr><td colspan="2"><hr></td></tr>
    <tr>
        <td>
            <?
                //This functions view the form that the user should fill in.
                function show()
                {
                    echo "<form method='post' name='offsign' action='$PHP_SELF'>";
                    echo "<tr><td colspan='2'>If wrong signature are written the post will not
be deleted.</td></tr>";
                    echo "<tr><td>Signature</td><td><input type='text'
name='sign'></td></tr>";
                    echo "<tr><td> </td><td><input type='submit' name='submit'
value='  OK  '></td></tr></form>";
```

```
                }
            ?>
        </td>
    </tr>
    <?
        //If the submit-button (called submit) are pressed down then the delete-function are
called upon.
        if($submit)
        {
            delete();
        }
        //This functions remove at person from the list with sources.
        function delete()
        {
            global $id_co;
            global $fname;
            global $sname;
            global $sign;
            //This first row here opens a database connection.
            $conn2=pg_connect("host=localhost dbname=odin user=examen1
password=DYWRBK");
            $sql2="UPDATE coordinate SET fname=", sname=", sign=" WHERE
id_co='$id_co' AND sign='$sign"';
            pg_exec($conn2, $sql2);
            echo "<script language='javascript'> window.close();</script>";
        }
        show();
    ?>
  </table>
 </body>
</html>
```

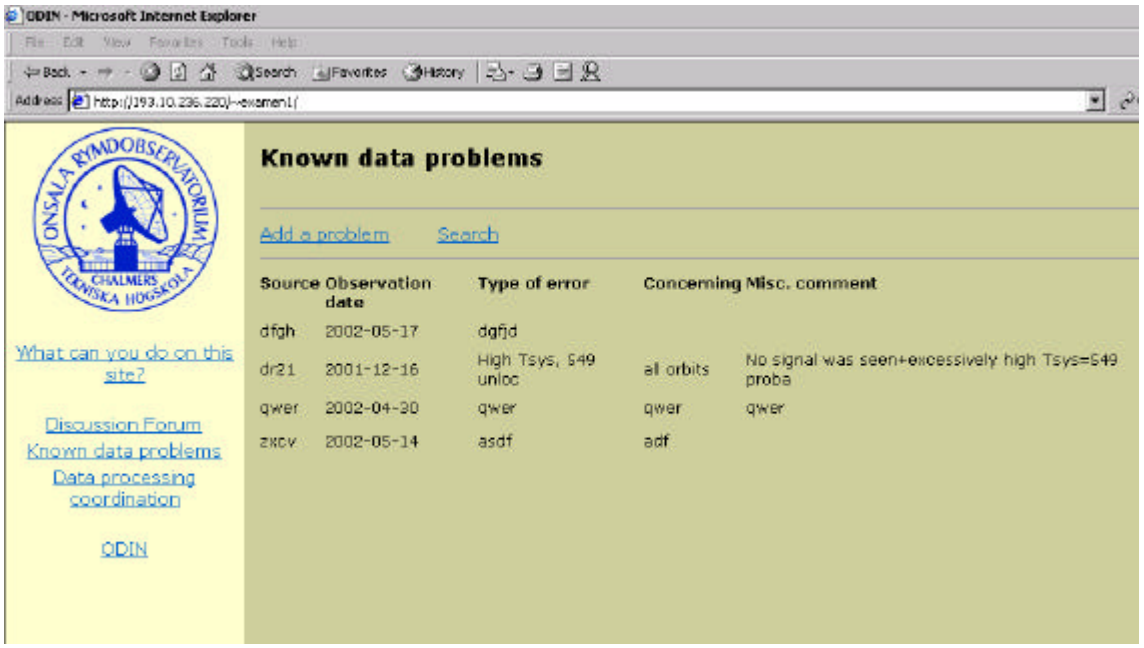
Bilaga F – Skärmdumpar



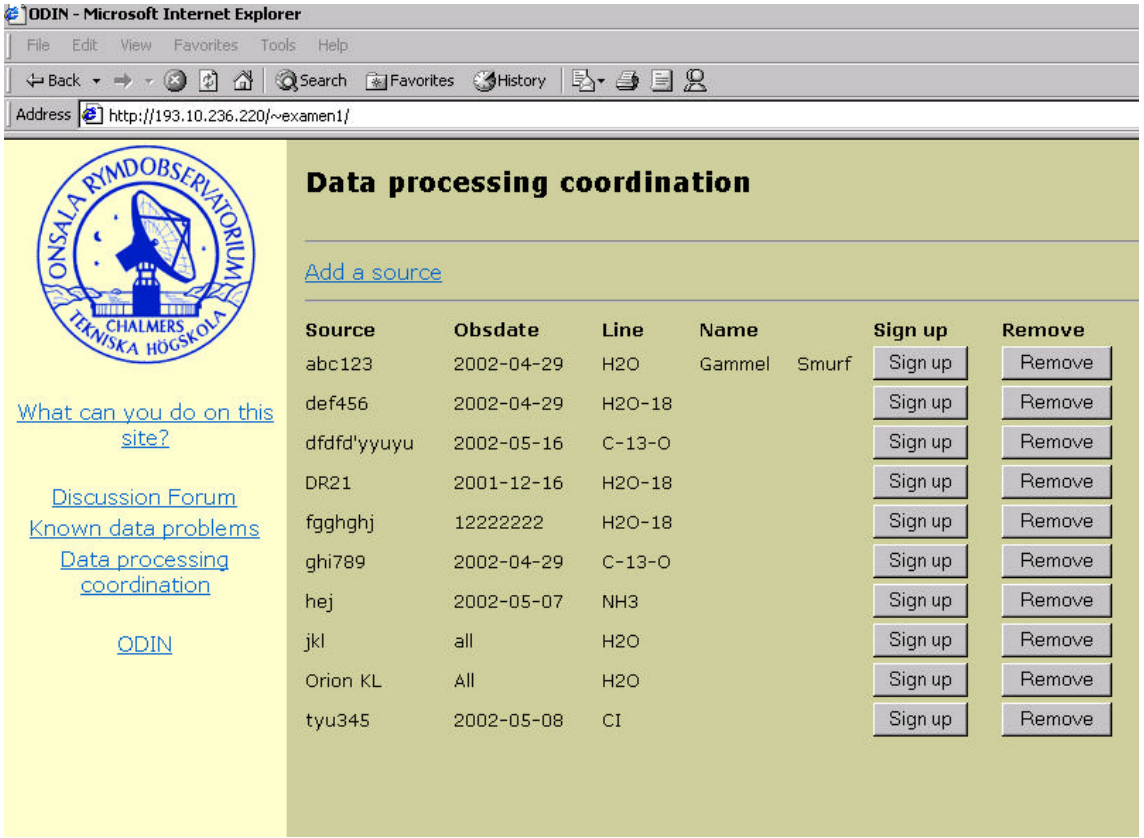
Figur 1 – Startsida



Figur 2 –Diskussionsforum



Figur 3 – Felhantering



Figur 4 - Samordning