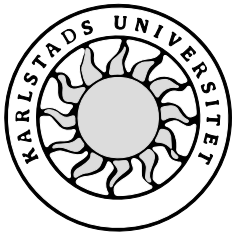


Datavetenskap

Marika Uhlander och Jimmy Byström

Automatisk schemaplanerare

Generering av scheman vid avdelningen för datavetenskap

Examensarbete, C-nivå

2002:13

Automatisk schemaplanerare

Generering av scheman vid avdelningen för datavetenskap

Marika Uhlander och Jimmy Byström

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Marika Uhlander

Jimmy Byström

Godkänd, 2002-06-04

Handledare: Mari Göransson

Examinator: Donald F. Ross

Tack

Vi vill tacka alla som jobbar på Marquée för deras ändlösa tålamod med att förklara olika saker för oss samt att lyssna på våra problem. Vi vill även tacka lärarna på avdelningen för datavetenskap på Karlstads universitet för deras hjälp med att sammanställa hur kurser är upplagda samt för deras synpunkter på vad ett schemaplaneringssystem bör innehålla.

Sammanfattning

Denna uppsats är en dokumentation över ett samarbete mellan företaget Marquée AB och avdelningen för datavetenskap vid Karlstads universitet. Målet med arbetet var att ta fram ett system som kunde per automatik generera scheman som tog hänsyn till personalens önskemål i bästa möjliga mån. Systemet skulle arbeta genom att ha tillgång till två typer av information, statisk och dynamisk. Enbart den dynamiska skulle behöva anges av användaren när ett schema skulle genereras. I denna uppsats beskrivs de olika delarna som ingått i arbetet för att ta fram ett sådant system. Då något val har gjorts beskrivs och diskuteras detta. Fördelar och nackdelar i sådana fall vägs mot varandra i en diskussion.

I uppsatsens början återfinns den kravspecifikation vi tog fram i början av projektet. Det är efter den vi har lagt upp utvecklingen av vårt system. I uppsatsen beskrivs vad en resursplaneringsmotor är samt hur Marquées resursplaneringsmotor använts i systemet. En viktig komponent i systemet är den databas som utvecklats i MySQL. I denna lagras all information som system genererat.

Automatic Schedule Planner

Schedule Generation for the Department of Computer Science

Abstract

This dissertion is collaboration between the company Marquée AB and the department of Computer Science at Karlstad University. The purpose of the project was to develop a system that could generate schedules automatically while taking into account, as much as possible, the wishes of the staff. The system would have access to two types of information, static and dynamic. The user should only have to specify the dynamic part of the information when a schedule is to be generated. The different parts of the system are described within this thesis. The advantages and disadvantages of the design choices made are describe and discussed.

The requirement specification that was produced in the beginning of this project can be found at the beginning of this thesis. The system was developed according to this requirement specification. We describe what a resource planning engine is and how the system uses Marquée`s resource planning engine. An important component in the system is the database, developed using MySQL. All the information that the system generates is stored in this database.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

Denna uppsats är del i ett examensarbete på C-nivå inom datavetenskap. Nedan följer en mer detaljerad beskrivning av målet med arbetet. I slutet av uppsatsen, se bilaga A, återfinns en lista över förkortningar och termer som används i arbetet.

Företaget Marquée har utvecklat en resursplaneringsmotor som behövde testas i en verklig applikation. Resursplaneringsmotorn kommer hädanefter i denna dokumentation att benämnas med förkortningen RPE (Resource Planing Engine). RPE:n är framtagen på ett sådant sätt att den kan användas i vitt skilda sammanhang, exempelvis inom sjukvård, skolor eller större företag för aktivitet - eller resursplanering. RPE:n är uppbyggd för att arbeta med resurser och aktiviteter av olika typer vilka används för att modellera en verksamhet. Dessa resurser och aktiviteter tas fram vid en designfas för en specifik verksamhet och registreras därefter i RPE:n så att de är tillgängliga vid ett senare tillfälle.

RPE:n har till uppgift att hålla reda på olika resurser och aktiviteter samt hur dessa hänger ihop i systemet för att modellera en verksamhet. För att få RPE:n att kunna vara mer intelligent finns det möjlighet att skriva regler som uttrycker vad som får utföras och vad som inte är tillåtet.

Avdelningen för datavetenskap på Karlstad universitet behövde ett system för att underlätta vid framtagning av nya scheman för kurser. Detta på grund av att deras nuvarande system för att ta fram scheman kändes onödigt arbetskrävande. Bland annat tyckte många lärare att det gick åt för mycket tid för att få fram något som egentligen inte utgjorde så mycket av deras egentliga arbete. Vid manuell schemaläggning är det även risk att vissa problem uppkommer, exempelvis finns det alltid risk att någon lärare råkar bli dubbelbokad på en viss tid. Detta på grund av att flera olika personer är involverade i samma scheman.

Ett annat problem som har uppkommit på senare tid är samarbetet mellan olika avdelningar på universitet, detta kräver tillgång till tjänsteplanering från flera olika avdelningar samtidigt. Det system som här diskuteras har inget direkt inbyggt stöd för att lösa detta problem men det borde inget vara något stort problem att implementera även detta.

Dessa andledningar ledde fram till ett samarbete som utformades till ett examensarbete. Uppgiften var att använda Marquées RPE för att modellera schemaläggningsprocessen vid avdelningen för datavetenskap vid Karlstads universitet och skapa ett program som automatiskt kan generera scheman.

1.1 Uppsatsens upplägg

För att läsa om de problem som låg bakom beslutet att genomföra detta arbete, samt vilket syfte arbetet hade, se kapitel 2 Problemformulering. Den kravspecifikation som följde av beslutet att genomföra detta arbete presenteras i kapitel 3 Kravanalys, där även information om vilka avgränsningar som gjorts finns. En mer ingående beskrivning av resursmotorer och hur de fungerar återfinns i kapitel 4 Resursmotorer och deras funktionalitet. I kapitel 4.3 Beskrivning av resurser och aktiviteter återfinns även en beskrivning av begreppen resurser, aktiviteter och regler. I kapitel 5 Programmets upplägg återfinns en beskrivning av de olika komponenter som ingår i systemet. Då vi har arbetat i ett flertal olika programmerings språk finns det en diskussion om varför vi valde just dessa språk i kapitel 6 Utvecklingsmiljö. De tester som har utförts på systemet samt deras utfall finns beskrivna i kapitel 7 Testning och verifiering. I slutet av uppsatsen återfinns de erfarenheter vi har fått i kapitel 8 Erfarenheter. Vårt sista kapitel handlar om de slutsater vi har dragit av vårt arbete, se kapitel 9 Slutsatser.

2 Problemformulering

När detta projekt startades var inte Marquées RPE helt färdigutvecklad utan vissa delar av RPE:n höll fortfarande på att förbättras samt nyutvecklas. Några av de delar som inte var helt färdigutvecklad hade direkt att göra med schemagenerering som vi skulle använda oss av.

Framtagning av ett schema är i dagsläget ett tidskrävande arbete då många olika lärare ibland måste samarbeta för att få fram ett slutgiltigt fungerande schema. Vid bokning av en sal i det nuvarande systemet måste det skickas en förfrågan till lokalbokningen där kraven på salarna är angivna och när de önskas vara tillgängliga. Problemet med detta är att önskade lokaler redan kan vara bokade. Detta leder till att schemat i värsta fall helt måste göras om.

Problemet med ovanstående lokalbokning kan tyckas enkelt. Det borde bara vara att implementera lokalbokningen i systemet, men det är väldigt svårt då hela universitetet använder sig av samma lokalbokning. En annan anledning till att denna lösning skulle bli krånglig att genomföra är att bokning av datasalar inte sköts på samma plats som bokningen av vanliga föreläsningssalar.

2.1 Syfte och mål

Huvudsyftet med vårt system är att möjliggöra ett enklare sätt att konstruera scheman för avdelningen för datavetenskap vid Karlstads universitet genom att skapa ett system som kan automatisera schemaläggninsprocessen så mycket som möjligt. Detta borde medföra en vinst i tidsåtgång genom att utnyttja personalens tid mer effektivt. Systemet ska även själv hålla reda på vilka personer som är upptagna vid olika tillfällen så att exempelvis dubbelbokningar kan förebyggas. Personalen ska själv kunna ange tider när det är önskvärt att de inte vill jobba, exempelvis tidiga måndagsmorgnar eller sena fredagskvällar. Utöver detta ska administrativ personal kunna blockera vissa tider för samtliga anställda. Detta för att exempelvis en anställd inte ska missa ett inplanerat möte eller jobba på oönskade tider under dygnet. Vid ett senare tillfälle kan systemet utökas så att personal kan få reda på deras lediga tid.

Systemet ska även testa hur Marquées RPE fungerar i en verklig verksamhet. Detta kan då leda till att de får information om regler som kanske kan lagras som generella regler i RPE:n direkt, detta kan förbättra användarvänligheten för RPE:n som helhet. Även idéer som uppkommit under arbetet kan påverka utseendet på den färdiga RPE:n, då viss funktionalitet kanske saknades eller behövde ändras för att på ett bättre sätt utnyttja RPE:ns kapacitet. Det

ska även testas hur enkelt det är att kommunicera med RPE:n samt hur lång svarstid som uppkommer vid olika typer anrop. Målet med arbetet är därmed att utvärdera dessa genom att skapa ett system för schemaläggning.

3 Kravanalys

Det fanns olika krav på vad systemet skulle klara av. I detta kapitel beskriver de förutsättningar som låg till grund för vårt arbete. Vi presenterar också den kravspecifikation som ska ligga till grunden för arbete. I slutet av kapitlet återfinns de olika avgränssningar som vi gjorde och en kortfattat diskussion om varför vi har gjort dessa avgränssningar.

3.1 Förutsättningar

I systemet ska Marquées RPE användas för att med hjälp av olika verksamhetsregler kunna ta fram ett schema där personal och andra resurser utnyttjas på ett bra och effektivt sätt. RPE:n ska själv hålla reda på när en resurs är upptagen och kunna ge ett korrekt och informativt felmeddelande när en felaktig allokering av en resurs utförs.

3.2 Kravspecifikation

Efter diskussion med vår handledare på universitetet samt med vår handledare på Marquée har vi tagit fram en kravspecifikation samt dokumenterat vissa avgränsningar.

Vårt program ska med hjälp av Marquées RPE kunna generera schema för avdelningen för datavetenskap vid Karlstad universitet. Detta ska utföras på ett enkelt sätt där användaren inte behöver ange en mängd information som är statisk. All statisk information lagras automatiskt när systemets tas i bruk. Om statisk information därefter behöver läggas till, exempelvis om en ny kurs startar, så ska det finnas ett gränssnitt som har detta som sin uppgift.

Programmet ska klara följande:

- Blockera tider som inte är lämpliga.
- Vara interaktivt.
- Ange antal studenter per kurs.
- Ange antal laborations grupper och räknestugor som ska finnas på kursen.
- Ange vilka lärare som är berörda och har vilken lektion/laboration.
- Ange antal timmar per kurs.
- Lagring av data kommer ska ske i en databas.
- Ange längden på varje lektionstillfälle.

- Programmet ska ha ett grafiskt gränssnitt.

Programmet bör klara följande:

- Grafisk visning av scheman.
- Val av olika schematyper.
- Rotation av laborationstider för de olika grupperna
- Möjlighet att ändra scheman vid ett senare tillfälle.

Programmet bör kunna byggas ut med följande funktioner:

- Skicka scheman till lärarnas kalendrar.
- Borttagning av gamla scheman.
- Automatisk lokalbokning.
- Framtagning av personalens lediga tid.

3.3 Avgränsningar

Den RPE som Marquée har tagit fram är mycket generell och kan användas inom många olika verksamheter och områden. Detta gjorde att vi var tvungna att göra vissa avgränsningar. Vi enades om att frågan om lokalbokning fick kvarstå tills vidare då detta inte är lokalt för varje institution och skulle medföra allt för stor komplexitet i systemet.

Vi beslutade att i första hand definiera en kurs i ett sekventiellt flöde så att alla lektioner och föreläsningar kommer i en förutbestämd ordning. Detta är en avgränsning eftersom vissa kurser har flera olika spår det vill säga kurser kan i vissa fall köra delar eller hela kursen parallellt i olika spår.

Under kursupplägg ryms en mängd information och alla olika avdelningar och institutioner vid universitetet har sina egna sätt att representera dessa. Då vi gör detta arbete för avdelningen för datavetenskap vid Karlstads universitet har vi enbart inriktat oss på det systemet som för närvarande används på avdelningen för datavetenskap vid Karlstads universitet.

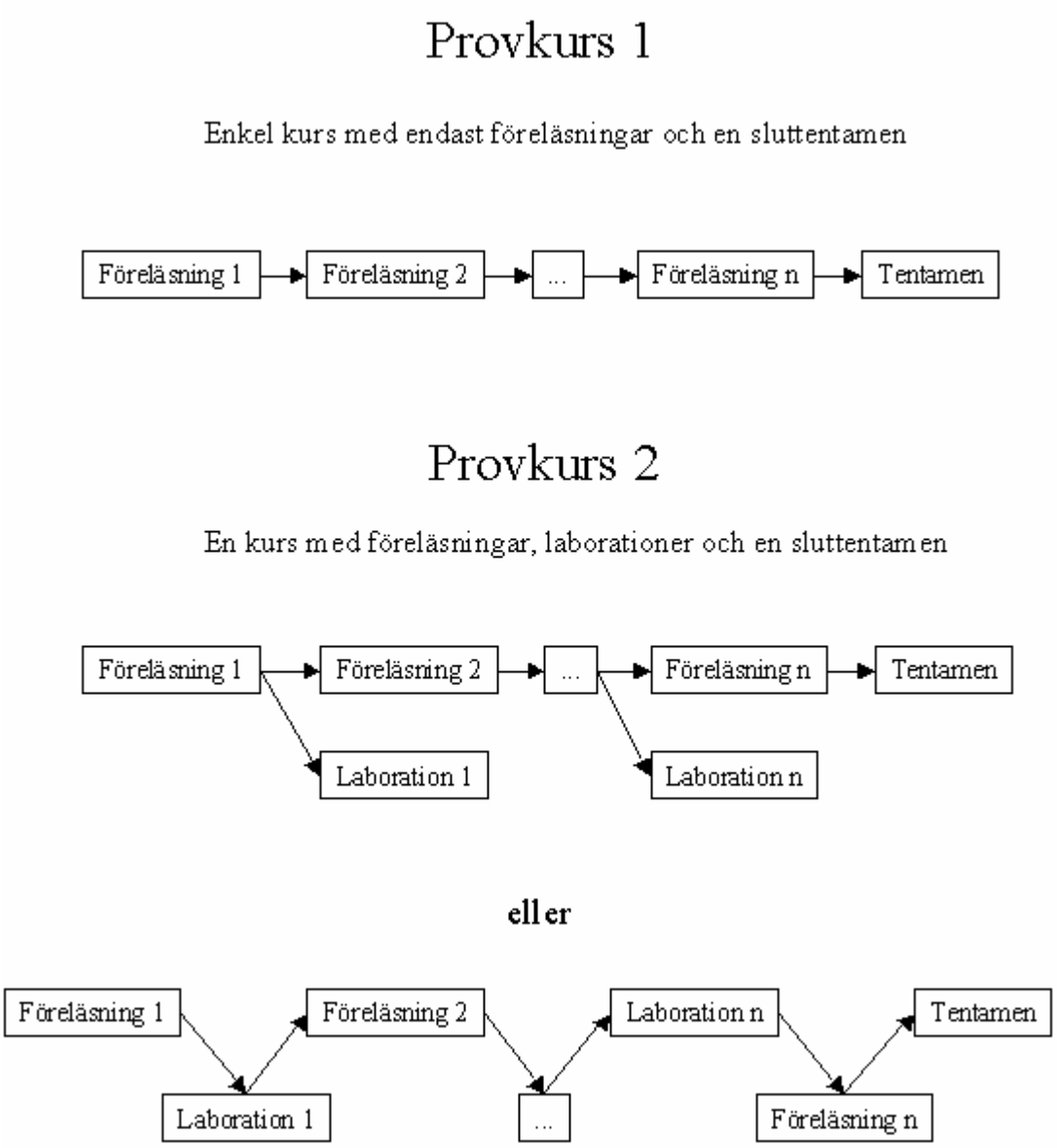
Vi har i vårt system inte heller tagit någon hänsyn till säkerhet och åtkomsträttigheter till databasen utan detta kräver vidare specifikation. När vi genererar scheman kontrollerar inte vår applikation hur många timmar som en lärare har uppnått. Därmed har ingen hänsyn tagits till tjänsteplanering.

För närvarande finns det enbart ett sätt att lägg till nya regler i vårt system. Enda sättet att lägga in nya regler är att skriva in de nya reglerna i initiatorn och därefter köra initiatorn igen. Detta är dock en dålig lösning då all information som hade lagts upp innan då försvinner och måste genereras igen. För att avhjälpa detta bör en funktion läggas upp där det finns möjlighet att lägga in nya regler och knyta dem till de resurser och aktiviteter som regeln ska gälla för. Denna förbättring av systemet har inte utförts på grund av brist på tid.

I diskussioner om det nuvarande systemet har det framkommit att lärarna har olika åsikter om vad som krävs och önskas för att ett system som automatiskt tar fram ett schema ska fungera på ett bra sätt. Vissa av dessa synpunkter har vi tagit hänsyn till i vår tänkta lösning medan andra plockades bort på grund av omfattningen och komplexiteten då blivit för stor för en C-uppsats.

Då varje kursupplägg måste specificeras innan ett schema kan genereras måste denna information finnas lagrad i RPE:n som regler, resurser och aktiviteter. Vi har inte har haft tillgång till avdelningens kursupplägg eftersom de fortfarande är under utveckling. Därför bestämde vi att avgränsa oss till att demonstrera hur systemet kan användas genom att definiera några exempelkurser som täcker upp de flest situationer som kan uppkomma.

I Figur 3-1 visas två exempel på hur kursers upplägg kan moduleras. I provkurs 1 finns enbart ett sekventiellt flöde av föreläsningar följt av en avslutande tentamen. Provkurs 2 visar på en kurs där det utöver ett sekventiellt flöde även finns vissa bispår. I den första av bilderna som beskriver provkurs 2 visas hur kursen verkligen ser ut. Medan den andra av bilderna visar hur kursen skulle kunna moduleras i vårt system.



Figur 3-1: Exempel på två olika kursflöden

4 Resursmotorer och deras funktionalitet

I detta delkapitel tas information upp om resursmotorer och om vad dessa kan användas till i olika verksamheter. Det beskrivs vad resurser och aktiviteter används för i vårt system. Senare i kapitlet finns en mer ingående beskrivning av Marquées resursmotor samt ett kortare exempel på hur deras resursmotor kan används.

4.1 Bakgrund om resursmotorer

Att planera och utnyttja resurser på ett bra och effektivt sätt är alltid av intresse för en större organisation. Dels arbetar människor bättre om de inte blir överbelastade med arbetsuppgifter. Organisationer tjänar även pengar på att få ett effektivt användande av sina tillgängliga resurser. Exempel på vad resurser kan beteckna är anställda på ett företag eller en lokal som kan användas för lagring av material. En mer ingående diskussion om resurser återfinns i kapitel 4.3.2 Resurser. Att på ett manuellt sätt försöka hålla reda på alla olika resurser inom organisationen samt att hålla reda på vilka resurser som interagerar med andra resurser i organisationen blir allt mer tidskrävande då organisationen växer och resurserna ökar i komplexitet.

Ett system som håller ordning på de olika resurserna samt hur dessa samverkar och i vilka olika tillfällen de är aktiva skulle kunna ge stora effektivitetsvinster. Ett sådant system benämns oftast som en resursplanerare eller resursmotor. Dessa system har ofta också möjlighet att visa vilka resurser som för tillfället är upptagna och när det är möjligt att ge en resurs en ny aktivitet utan att resursen får problem eller konflikter med sina andra aktiviteter.

4.2 Diskussion om existerande resursmotorer

På marknaden finns ett flertal resursmotorer att tillgå. De flesta av dessa är tyvärr allt för komplexa eller svåra att använda för mindre organisationer och även i vissa fall för större organisationer. I många fall återfaller organisationer till att använda ett manuellt system för att hålla reda på sina aktuella resurser, vilket leder till att samma arbete utförs parallellt. Vid en inspektion kan detta i värsta fall leda till att resultaten inte stämmer överens med varandra. Denna inkonsistens kan vid en omorganisation ge stora problem för effektiviteten i organisationen.

Vad som krävs av ett system är att det är kraftfullt och samtidigt enkelt att använda vid förändringar inom organisationer. Systemen bör klara av att ge information om när en resurs är upptagen och vid vilka tillfällen en resurs kan blir delaktig i nya aktiviteter. Systemen bör även vara lätta att underhålla, det vill säga att lägga till nya resurser samt att ta bort gamla resurser samt kopplingar mellan olika resurser eller aktiviteter.

En annan fördel som ett system kan ha är att det är litet och anpassningsbart. Detta gör att en organisation inte behöver ha ett stort generellt system där de enbart använder en bråkdel av vad systemet klarar av. Utöver detta kan de möjligen fortfarande arbeta manuellt för att hantera viss information om olika resurser.

4.3 Beskrivning av resurser och aktiviteter

Nedan följer en kort beskrivning av vad resurser och aktiviteter står för i RPE:n samt vad dessa kan representera.

4.3.1 Resurstyper

Resurstyper är bastyper som en resurs kan betecknas vara. Exempelvis betraktas professor som en resurstyp och sekreterare som en annan resurstyp. Dessa måste registreras först i RPE:n då alla resurser måste bestå av åtminstone en resurstyp. Till dessa kopplas även de generella regler som ska gälla för alla resurser av denna typ.

4.3.2 Resurser

En specifik lärare är ett exempel på en resurs. Dessa är speglingar mot verkligheten och är i vårt system fristående, även om RPE:n har möjlighet att låta resurser ha kopplingar mellan olika resurser.

Ett exempel på en sådan koppling skulle vara att betrakta hela Karlstad universitet som en resurs som utbildar personer. Denna resurs har i sig en mängd olika resurser som representerar de olika lärarna. Allt som universitetet står för står även lärarna implicit för. Det finns möjlighet att direkt koppla regler till en resurs. Detta om en specifik regel finns för resursen.

4.3.3 Aktivitetstyper

Aktivitetstyper är bastyper på samma sätt som resurstyper, dock är de bastyper för olika typer av aktiviteter som kan förekomma i verksamheten. En kurs eller en föreläsning är exempel av aktivitetstyper.

4.3.4 Aktiviteter

En specifik föreläsning eller en specifik kurs är exempel på aktiviteter. Det är lättare att förstå att en aktivitet kan ha underordnade aktiviteter än att resurser kan ha detta. I vårt system används detta för att modellera en kurs upplägg.

4.4 Marquées resursmotor

I detta delkapitel beskriver vi bakgrunden till varför Marquées RPE skapades. Vi beskriver även hur RPE:n är uppbyggd. Det återfinns även ett kort kapitel angående vad regler används till och hur dessa skrivs. Avslutningsvis ger vi ett kort exempelscenario där RPE:n kan användas.

4.4.1 Bakgrund

Idén om att skapa en liten men ändå kraftfull resursmotor uppkom för många år sedan. Det var Greger Ohlsson vid nuvarande Marquée som födde idén att skapa RPE:n. Hur denna resursmotor skulle fungera och vara uppbyggd har under årens gång förändrats flera gånger för att anpassas till nya idéer. Ett av kraven på resursmotorn har alltid varit att den ska vara liten och anpassningsbar till olika situationer med hjälp av regler som användaren kan skapa på ett lätt sätt.

4.4.2 Uppbyggnad

Marquées resursmotor som nu går under arbetsnamnet Callista®, modellerar organisationer genom att använda sig av två olika typer av objekt, nämligen resurser och aktiviteter. För att kunna simulera intelligens i motorn finns regler som kan kopplas till dessa objekt.

Resurser i motorn representerar objekt som på olika sätt kan utföra ett arbete i den verkliga organisationen, exempel på resurser är människor, maskiner men även lokaler. Aktiviteter är händelser som resurser kan vara kopplade mot, exempel på aktiviteter är möten, resor men även ledighet och semestrar kan representeras som aktiviteter. Alla resurser och aktiviteter tilldelas en unik identifierare så att RPE:n kan hålla reda på dessa.

Möjligheten att gruppera olika objekt som beter sig på liknande sätt finns också representerade i motorn. Varje objekt av typen resurs måste vara en instans av en övergripande typ som i motorn kallas resurstyp. Samma förhållande gäller även för aktiviteter som måste ha en aktivitetstyp som övergripande typ. Detta ger en möjlighet att koppla generella regler till bastyper istället för att koppla samma regel till alla resurser som ska bete sig på liknande sätt.

Anställda inom en organisation skulle kunna utgöra en resurstyp då alla har minst en sak gemensamt, alla är anställda på den organisationen. En undergrupp till detta skulle kunna vara avdelningschefer, vilket då också kan representeras som en resurstyp som i sig har en övergripande typ, nämligen resurstypen anställd. Detta gör att alla avdelningschefer som finns inom organisationen också betraktas som anställda, något vi också rent intuitivt känner är korrekt. Som övergripande aktivitetstyp skulle möten kunna representeras och en undergrupp till detta skulle kunna vara planeringsmöten som är en mer specifik typ av aktivitetstypen möte. Denna möjlighet av trädstrukturer inom resurser och aktiviteter ger en stor möjlighet till att modellera en organisation på ett korrekt sätt.

Reglerna i motorn är för närvarande representerade i språket JavaScript och används för att ställa krav på när och hur en resurs eller aktivitet får användas. Reglerna kan kopplas direkt till en resurs eller aktivitet men kan även kopplas till en resurstyp eller aktivitetstyp. De kommer då att gälla för samtliga resurser eller aktiviteter som beror på den övergripande typen. Detta ger en möjlighet att koppla generella regler som kommer att gälla exempelvis samtliga anställda på en organisation. Samtidigt finns det möjlighet att specifikt koppla en viss regel till en specifik resurs eller aktivitet.

Om en djupare kunskap önskas angående Marquées RPE kan förfrågan angående deras dokumentation ställas till dem. Detta eftersom RPE:n fortfarande var under utveckling och dokumentation över denna inte heller var färdigställd.

4.4.3 Regler

RPE:n har ingen intelligens i sig själv för att kunna göra optimala val, om RPE:n utan någon som helst kontroll skulle planera in en aktivitet skulle den hamna på första möjliga tillfälle. För att hjälpa RPE:n att dra smarta och intelligenta slutsatser om hur den ska lägga aktiviteterna använder vi oss av regler.

Det finns generella regler inlagda i RPE:n, till exempel att en resurs inte kan bli schemalagd på två aktiviteter vid samma tid punkt. Sedan finns det regler som vi själv har definierat, dessa är mer specifika för verksamheten vid avdelningen för datavetenskap. Ett exempel på en sådan regel är att det är möjligt att blockera tider då en lärare inte vill ha en aktivitet/föreläsning. Alla regler är skrivna i JavaScript och för ett exempel på hur en regel kan vara skriven se bilaga D Regelexempel.

4.4.4 Exempelscenario

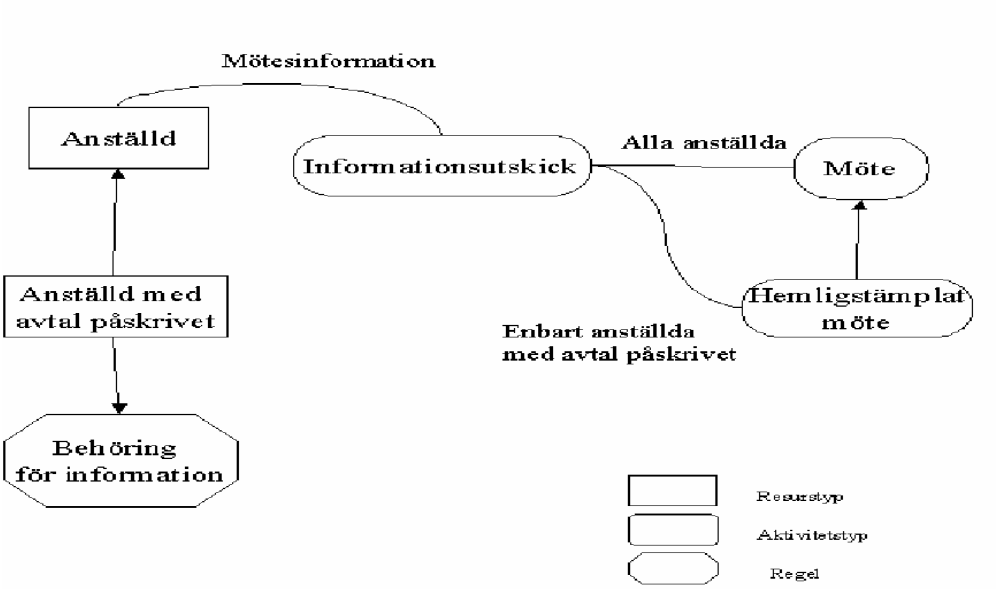
En organisation är uppdelad på flera olika avdelningar där en avdelning arbetar med utveckling av nya produkter. De dokument som rör framtagande av nya produkter är hemligstämplad. Därför finns en regel som säger att alla som ska arbeta med dessa dokument måste ha skrivit på ett avtal. I detta lovar den anställde att inte diskutera innehållet i dessa dokument med någon som inte har skrivit på ett sådant avtal.

Vid vissa tillfällen hålls möten där de nya produkterna diskuteras och när det bestäms att ett sådant möte ska hållas skickas information om detta ut till alla anställda som har skrivit på avtalet. Utöver dessa möten hålls årligen möten för samtliga anställda där organisationens arbete diskuteras, informationen om när dessa möten hålls skickas också ut till samtliga anställda i god tid så att alla vet när mötet ska hållas.

Med hjälp av denna information bestäms att alla anställda på organisationen ska representeras som en resurstyp och till denna kopplas regler som gäller för samtliga anställda. De som har skrivit på avtalet och därmed får jobba med de olika hemligstämplade dokumenten representeras som en undergrupp till ovanstående resurstyp.

Det finns två olika aktivitetstyper, en bastyp som representerar alla möten inom organisationen och en undertyp till denna som representerar de möten som hålls om de nya produkterna. Utöver detta finns även en annan aktivitetstyp som representerar utskickandet av information till berörda när ett möte ska hållas inom organisationen.

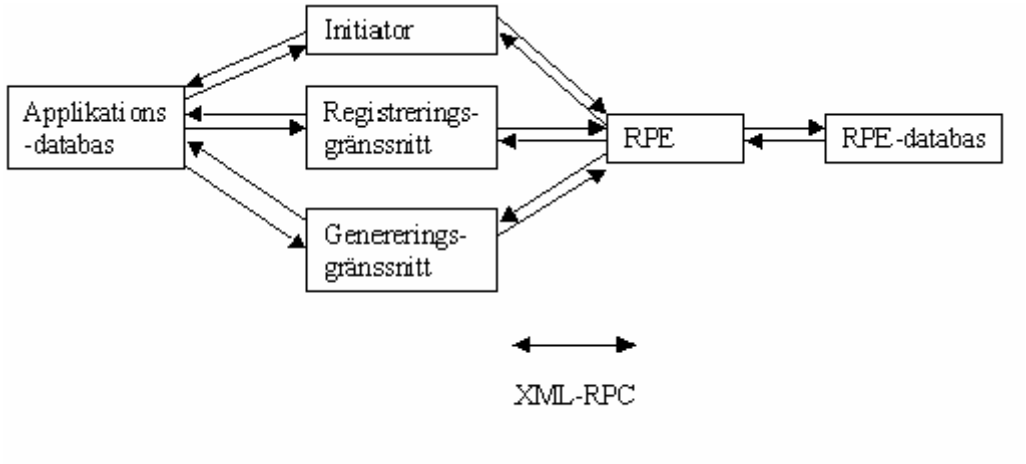
Åtminstone en regel behövs för att hålla reda på vilka som ska få informationen om de olika möten som ska hållas och denna regel kopplas med fördel till aktivitetstypen som har hand om att informationen blir utskickade. Diagram i Figur 4-1 visar hur de olika resurstyperna och aktivitetstyper kopplas samman.



Figur 4-1: Exempelscenario

5 Programmets upplägg

Vårt system består av flera olika delar som samverkar med varandra. De delar som applikationen är uppbyggd runt är en applikationsdatabas, Marquées RPE och två olika användargränssnitt, ett för registrering av nya resurser och aktiviteter samt ett för generering av scheman. Vid uppstart av systemet används även en initiator som registrerar generella regler, resurser och aktiviteter. Figur 5-1 visar hur de olika programdelarna kommunicerar med varandra. Ett exempel kan vara om en lärare ska generera ett schema startas genereringsgränssnittet. Från detta gränssnitt går ett anrop till applikationsdatabasen för att hämta upp statisk information. Därefter får läraren ange den dynamiska informationen som då resulterar i ett anrop till både RPE:ns databas, via RPE:n, och ett anrop till applikationsdatabasen. Anrop mellan vårt program och RPE:n sker med hjälp av XML-RPC se kapitel 5.4.1 XML-RPC för en beskrivning av detta.



Figur 5-1: Programmets uppbyggnad

5.1 Applikationsdatabasen

Databasen som används i systemet är skapad i MySQL. Databasen följer tredje normalformen. De regler som vår databas enligt denna normalform uppfyller är således:

- Alla attribut i databasen ska vara atomära, det vill säga bara bestå av ett värde.
- Alla attribut ska vara fullständigt funktionellt beroende av hela sin primärnyckel.
- Inget attribut får vara fullständigt funktionellt beroende av något icke-nyckel attribut

5.1.1 Syfte

Applikationsdatabasen är den databas där informationen från programmet lagras. Här finns även en koppling till RPE:n. Vid uppstarten av systemet finns en del information redan lagrad, nämligen den information som genom initieringen ska göras tillgänglig från början. Den information som redan från början finns lagrad i databasen utgörs av information om de anställda samt vilka kurser som ges på avdelningen.

Efterhand som systemet utnyttjas läggs mer specifik information till i databasen. Om någon nyanställning görs eller kursutbudet förändras reflekteras även detta i databasen.

5.1.2 Konstruktion

Databasen består av två typer av tabeller, kopplingstabeller och bastabeller. Kopplingstabellerna använder sig av främmande nycklar på ett sådant sätt som är vanligt för relationsdatabaser i allmänhet. I bastabellerna används också en främmandenyckel, dock på ett lite annorlunda sätt då dessa är nycklar i RPE:ns interna databas och används vid allokering och deallokering av resurser, aktiviteter och regler. Informationen i bastabellerna kan betraktas som statisk och här bör inte förändringar uppkomma så ofta. Kopplingstabellerna däremot förändras ofta då informationen som lagras är väldigt dynamisk och speglar den aktuella situationen i verksamheten.

Innan systemet startas första gången bör viss grundläggande information manuellt skrivas in i bastabellerna. Den information som ska skrivas in är den statiska informationen som redan finns tillgänglig. Kopplingstabellerna ska lämnas tomma då dessa automatiskt fylls i av systemet när behov uppkommer. Ingen av dessa tabeller får efter uppstart av systemet förändras manuellt utan ska enbart förändras genom användning av registreringsgränssnittet.

Det återfinns ett E/R diagram som visar hur de olika tabellerna i databasen kopplas samman i bilaga C E/R diagram över databasmodellen.

5.1.2.1 Bastabeller

I bastabellerna finns information som betraktas som statisk och som inte ändras så ofta. Här ryms bland annat information om de anställda, kurserna med mera. I dessa tabeller finns en koppling mot RPE:n genom det unika id som varje entitet blivit tilldelad från RPE:n. Dessa tabeller är i sig ganska statiska efter uppstarten av systemet och nästan samtliga förändringar i dessa tabeller resulterar i ett anrop till RPE:n. I Tabell 5-1 följer en kortfattad beskrivning av samtliga bastabeller i applikationsdatabasen. En mer ingående beskrivning av bastabellerna finns i bilaga B Databastabeller.

Tabellnamn	Beskrivning av tabell
Teacher	Denna tabell används för lagring av information om anställda vid avdelningen för datavetenskap vid Karlstads universitet. I tabellen lagras information som senare presenteras på de olika scheman som genereras.
Course	Används för lagring av institutionens olika aktuella kurser.
Activitytypes	I denna tabell lagras de olika typerna av aktiviteter som är intressanta för verksamheten. Denna tabell är i nuläget helt statisk och det finns ingen möjlighet att vare sig ta bort eller lägga till nya aktivitetstyper efter initieringen. Detta bör dock implementeras i en senare version av systemet.
Rules	Alla namn på regler som finns lagrade i RPE:n samt deras unika id lagras i denna tabell.
Resourcetypes	De olika typer av resurser som kan förekomma inom verksamheten representeras i denna tabell. Även denna tabell är helt statisk för tillfället och möjlighet att förändra lagrad information i denna tabell bör läggas till i en senare version av systemet.

Tabell 5-1: Beskrivning av bastabeller

5.1.2.2 Kopplingstabeller

I kopplingstabellerna finns information som är mer eller mindre varierande med tiden. Viss information i dessa tabeller skulle kunna betraktas som statisk men det finns även en annan anledning till att vi har särskrivit dessa tabeller gentemot de andra. Dessa tabeller används nämligen för att koppla samman flera olika bastabeller. I Tabell 5-2 följer en beskrivning av samtliga kopplingstabeller i applikationsdatabasen. En djupare beskrivning av kopplingstabellernas uppbyggnad kan ses i bilaga B Databastabeller.

Tabellnamn	Beskrivning av tabell
Teacher_resourcetype	Sammankopplar en anställd med en resurstyp vilket representeras av deras titel. I denna tabell lagras enbart den högsta varianten av titel som en lärare har. Det vill säga att en professor får titeln professor och inte titeln lärare.
Course_activity	I denna tabell lagras information om vilka aktiviteter en kurs har och hur många av varje. För tillfället lagras här ingen information om alla

	de olika aktivitetstillfällena även om detta varit möjligt har det bestämts att den informationen varit onödig att lagra.
Teacher_course	Denna tabell används för tillfället inte men finns redan definierad då behovet av en sådan tabell är uppenbar i en förlängning av projektet. Denna tabell håller reda på vilka lärare som är tillgängliga för vilka kurser, denna tabell beror på tjänsteplaneringen och bör uppdateras efter varje tjänsteplanering.

Tabell 5-2: Beskrivning av kopplingstabeller

5.2 Användargränssnitt

I systemet finns det två olika typer av gränssnitt, ett registreringsgränssnitt och ett genereringsgränssnitt. Dessa används vid olika tillfällen i systemet och bör användas av olika användare. Detta är dock inte implementerat utan för närvarande kan vem som helst komma in och använda dessa gränssnitt. Vid ett senare tillfälle bör dessa utrustas med någon säkerhetsmekanism för att skydda mot otillåten förändring av den information som systemet är uppbyggt runt.

5.2.1 Registreringsgränssnitt

I denna del av systemet bestämde vi att följande operationer skulle kunna utföras.

- Registering av anställd
- Modifiering av anställd
- Avregistering av anställd
- Registering av kurs
- Registering av kursupplägg
- Avregistering av kurs
- Visning av aktuella tabeller

En mer detaljerad beskrivning av dessa operationer följer i nedanstående delkapitel. Detta gränssnitt används när något har förändrats i verksamheten exempelvis när en ny lärare har anställts eller en ny kurs har tagits fram. All förändring i verksamheten bör föras in i systemet så fort som möjligt, detta för att systemet ska fungera optimalt. Ifrån detta gränssnitt går kopplingar mot applikationsdatabasen och bastabellerna däri. Detta gränssnitt bör enbart få användas av administrativ personal. I nedanstående delkapitel har tabellnamn i

applikationsdatabasen angivits med kursiv stil. Mer information om hur dessa tabeller är uppbyggda återfinns i bilaga B Databastabeller.

5.2.1.1 Registrering av anställd

Vid en nyanställning på avdelningen ska relevant information föras in om den nya personen i databasen. Informationen lagras i bastabellen *teacher* så att personen finns tillgänglig i systemet vid ett senare tillfälle. Eftersom en anställd har olika titlar lagras även en koppling mellan detta i databasen, då i kopplingstabellen *teacher_resourcetype*. När registreringen görs anropas även RPE:n som ger den nya resursen dess interna identifierare.

5.2.1.2 Modifiering av anställd

Om en anställd förändrar något/några av attributen som registrerades från början ska detta ändras så att databasen hålls uppdaterad. Däremot bibehålls den interna identifieraren även efter förändringen. Detta för att RPE:n använder sig av en resurs, inte den anställdes attribut.

5.2.1.3 Avregistering av anställd

Då en anställd av någon anledning försvinner från avdelningen går det lätt att avregistrera denna genom att ange dess signatur. Detta ska även medföra att RPE:n kontrollerar om resursen är bunden till någon aktivitet. Om någon sådan koppling uppkommer får användaren ett meddelande om detta och får därefter ta ställning till om den anställda ska tas bort direkt eller om den aktiviteten ska kopplas om först. Därefter avregistreras även den interna identifieraren för att förhindra nedlusning av systemet, det vill säga data som ligger kvar i systemet utan att fylla någon funktion.

5.2.1.4 Registrering av ny kurs

Då en ny kurs upprättas krävs registrering mot databasen innan ett schema kan genereras för kursen. Registreringen innebär att kursen får en intern identifierare som RPE:n använder sig av senare.

5.2.1.5 Registrering av kursupplägg

Innan ett schema kan genereras krävs att kursens upplägg har angivits. Detta innebär att kursens attribut har bundits i RPE:n. Attributen som krävs för kursen är antalet för olika aktiviteter och hur långt varje tillfälle ska vara. Dessa attribut lagras sedan i tabellen *course_activity* i applikationsdatabasen.

5.2.1.6 Avregistering av kurs

När en kurs slutar att ges ska RPE:n kontrollera att den inte är kopplad till någon resurs eller aktivitet. Om så ej är fallet tas den bort och dess interna identifiera deallokeras. Om däremot kursen har en koppling till någon annan aktivitet eller resurs ska ett meddelande ges om detta.

5.2.1.7 Visning av aktuella tabeller

När användaren vill få en överblick över databasen och den information som lagras i den finns det även funktionalitet som visar dessa. Det finns för närvarande två olika tabeller som på detta sätt kan presenteras, nämligen tabellerna *teacher* och *course*.

5.2.2 Genereringsgränssnitt

I följande delkapitel beskrivs nedanstående punkter som uppkommer vid generering.

- Inmatning av kursattribut
- Den genererade informationen

Detta gränssnitt används då ett schema för en kurs ska genereras. För att kunna generera ett schema behövs viss information från användaren. Denna information består bland annat i att användaren anger datum för kursstart och tentamen. Även information om vilka lärare och vilket/vilka kurstillfällen de är involverade i fastställs. Denna information ska sedan ligga till grund för att RPE:n ska kunna planera ett schema utifrån önskningar och krav som de olika involverade personerna har. När en användare önskar generera ett schema möts han alltid av ett antal skärmbilder som ser väldigt lika ut varje gång. Skärmbilder hur genereringen kan se ut återfinns i kapitel 7.1.3.1 Testfall för genereringsgränssnittet.

5.2.2.1 Inmatning

Användaren presenteras med en skärmbild där olika val får tas. Dessa val består av vilken kurs ett schema önskas genereras för, antal antagna studenter, start- respektive tentamensdatum, kursens fart samt vilka dagar i veckan den ska ges. Dessa val används sedan som grund för genereringen av schemat för kursen.

Under inmatningsfasen av genereringen får användaren ange vilka lärare som ska vara involverade i kursen samt vilka undervisningstillfällen de ska vara ansvariga för. Även information angående hur långt varje undervisningstillfälle är bestäms. Därefter bearbetas informationen och ett schema genereras utifrån alla regler och informationen som angivits.

5.2.2.2 Den genererade informationen

Informationen från genereringen presenteras som ett textdokument där även möjlighet till att spara schemat ska ges. Detta har av tidsbrist ej kunnat genomföras men borde dock inte vara något större problem.

5.3 Initiator

Vid installation av systemet används initiatorn för att registrera resurs- och aktivitetstyperna som är angivna i applikationsdatabasen, samt för att skapa en koppling mellan applikationsdatabasen och RPE:ns interna databas. Även regler kopplas samman mot RPE:n och databasen. Regler som ska vara generella kopplas ihop med korrekta resurs- eller aktivitetstyper. En närmare beskrivning av vad resurer och aktiviteter återfinns i kapitel 4.3 Beskrivning av resurser och aktiviteter. För en mer ingående beskrivning av regler se kapitel 4.4.3 Regler

5.4 Kommunikation med resursmotorn

För att lägga till ny information och hämta hem information från RPE:n måste det finnas en möjlighet att kommunicera med den. Kommunikationen bör fungera på ett lätt sätt samt vara enkel att använda. Sättet som valdes för kommunikation med RPE:n är XML-RPC vilket beskrivs nedan.

5.4.1 XML-RPC

XML-RPC är en specifikation som beskriver hur olika komplexa datastrukturer ska skickas över internet. RPC handlar om att det är ett distribuerat funktionsanrop och XML bestämmer hur de olika datastrukturerna ska vara beskrivna. Om en djupare förståelse för hur XML-RPC fungerar samt vad som krävs av användare för att använda det så finns en mer ingående beskrivning av detta på XML-RPC’s hemsida [5].

XML-RPC har valts som kommunikationsmedium då det inte hindras av brandväggar och inte kräver speciella inställningar innan det kan användas. XML-RPC fungerar även väldigt bra tillsammans med Java och är enkelt att använda och lära sig. Den XML-RPC som har använts i detta system är utvecklad av Marquée och om en mer ingående beskrivning av den önskas finns en länk om detta att tillgå från deras hemsida [4].

6 Utvecklingsmiljö

Under implementationsfasen av vårt arbete har vi använt oss av programmet J-Builder för att utveckla vårt system. J-Builder stödjer utveckling av kompletta system där alla aspekter kan modelleras. I J-Builder har vi arbetat med JSP, JavaScript och delvis med SQL-anrop. Med hjälp av J-Builder's inbyggda Tomcat Javamotor "kompileras" sidorna som sedan kan exekveras och visas på en vanlig webapplikation som Explorer eller Netscape.

Vi har skapat två olika paket i J-Builder där det ena är initiatorn och det andra används för registrering och generering. Dessa paket består av ett flertal .jsp filer och .java filer där funktionaliteten har delats upp. En mer ingående beskrivning av vad dessa paket används till finns i kapitel 5.2 Användargränssnitt samt kapitel 5.3 Initiator. I bilaga E Källkod för systemet återfinns även den källkod som ingår i de båda paketen.

6.1 Kortfattad beskrivning av använda programspråk

Under arbetets gång har vi växlat mellan ett flertal olika språk då delar av arbetet är lättare att genomföra i ett visst språk. Nedan följer en kort beskrivning av de olika språken som använts under arbetets gång.

6.1.1 JSP

JSP är ett språk som är en blandning av Html och Java. Grunden i JSP är Html-koden med möjlighet till att använda sig av Java kod genom en viss sorts kommentering i koden. Detta ger en möjlighet att skapa dynamiska html-sidor. Dessa kan förändra utseendet beroende på vad användaren väljer och vad som finns representerat exempelvis i applikationsdatabasen.

6.1.1.1 Html

Html används för att skapa hemsidor. Dock blir dessa sidor till största delen statiska vilket inte lämpar sig så bra i ett system som förändras. Detta språk används och stöds av de flesta webbrowsrar vilket är anledningen till att vi valt att presentera vårt system genom Html.

6.1.1.2 Java

Java är ett rent objektorienterat språk som är väldigt enkelt att arbeta med när sidor ska presenteras på Internet. Det är även enkelt att göra kraftfulla operationer i Java samt att representera verkligheten genom klasser och objekt. Java är grundspråket i vårt system.

6.1.2 JavaScript

JavaScript används ofta för att skapa dynamiska hemsidor. Språket är uppbyggt som Java men har ingen som helst typning utan alla variabler ses i första hand som strängar. JavaScript används ofta för att kontrollera att formulär är korrekt ifyllda på en sida samt för att kommunicera med användaren.

Vi har använt JavaScript för att skriva våra regler, detta var det enda språket som RPE:n förstod när den skulle exekvera reglerna. Vi har även använt JavaScript till en del kontroller samt vid val som användaren gör.

6.1.3 SQL

SQL är ett språk som används vid hantering av databaser och är för närvarnade det i särklass största språket för databashantering. SQL använder vi när vi vill kommunicera med applikationsdatabasen.

7 Testning och verifiering

Vi har utfört tester på vårt system och en närmare beskrivning på vilka tester vi har utfört följer nedan.

7.1 Beskrivning av testning

I följande delkapitel beskriver vi vilka tester som utförts samt vilka resultat som dessa gett. Vi har delat upp testerna efter systemets olika delar. De skärmbilder vi har tagit med i detta kapitel är till för att förtydliga för användaren. Utöver detta bör även program köras när testfallen uförs.

7.1.1 Initiatorn

För att testa initiatorn förde vi in all relevant information om de anställda och om kurserna i applikationsdatabasen. För att inte bryta mot applikationsdatabasens regel angående att ett resursid eller ett aktivitetsid inte får vara odefinierat, skrivs slumpvärden in i dessa positioner. När sedan initiatorn registrerar upp dessa resurser och aktiviteter tilldelas det verkliga id-numret.

Även information om vilka resurstyper och vilka aktivitetstyper som ska finns i systemet skrevs in i applikationsdatabasen. Vi definierade även upp en del regler direkt i initiatorn, vilka därefter registrerades upp mot RPE:n samt att deras namn fördes in i tabellen *rules* i databasen.

För att testa att rätt information lagrades tillsammans med korrekt id i RPE:n och databasen ställde vi frågor mot RPE:n. Svar på dessa frågor visade att det fungerade som det skulle.

7.1.2 Registeringsgränssnittet

Vi har utfört likartade tester på alla operationer i detta gränssnitt förutom operationen ”visa tabeller”. Testerna vi har utfört är:

- Inläggning av ny information
- Inläggning av redan existerande information
- Borttagning av information
- Definiering av en kurs
- Förändring av lagrad information, endast för lärare ej kurser

Inläggning av en lärare eller kurs har testats både när informationen existerar i databasen och när informationen är ny. Vid inläggning av information som redan existerar ger systemet tillbaka ett felmeddelande att posten redan finns. Då inte posten finns i databasen läggs den in samt registreras upp mot RPE:n.

Borttagning går endast att utföra på existerande information, detta på grund av att systemet visar endast de poster som finns i databasen.

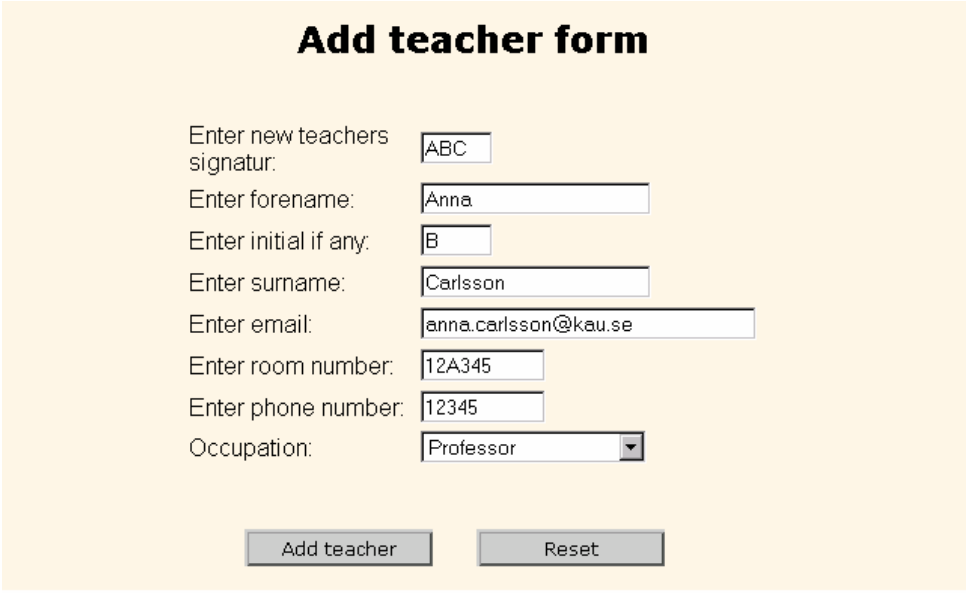
Det går enbart att förändra en lärares attribut då denna redan finns i systemet. En sådan förändring kan vara att efternamn eller telefonnummer förändras. Vi har testat alla möjliga förändringar som en lärare kan göra.

7.1.2.1 Testfall för registreringsgränssnittet

I detta delkapitel visas testfall då en ny anställd, kurs och kursupplägg registreras upp, även testning av att lägga till en redan inlagd anställd utförs. Testfall för borttagning görs på en kurs. Om programmet redan är startat bortses första punkten på alla testfallen nedan.

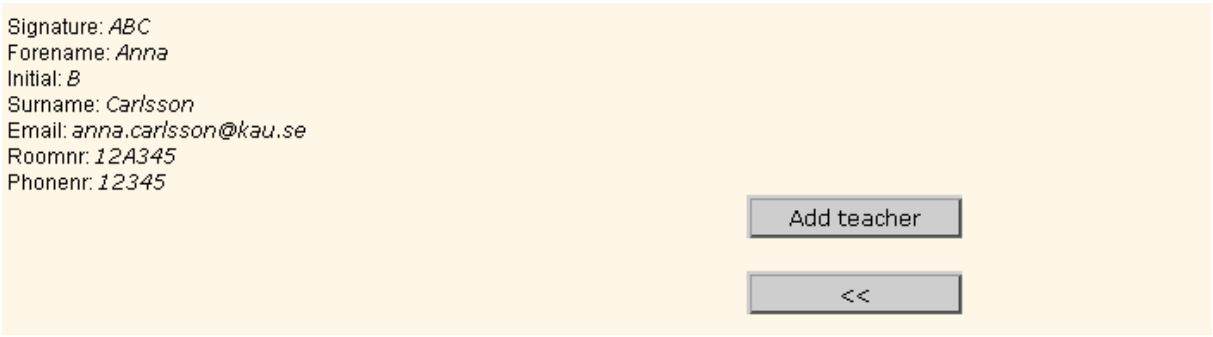
Testfall 1-Lägg till anställd.

1. Starta programmet
2. Tryck på knappen ”Add Teacher”
3. Fyll i attribut som i Figur 7-1 nedan
4. Tryck på knappen ”Add teacher”



Figur 7-1: Skärmbild av in-data vid registrering av ny lärare.

Förväntat resultat av Testfall 1 ses i Figur 7-2.

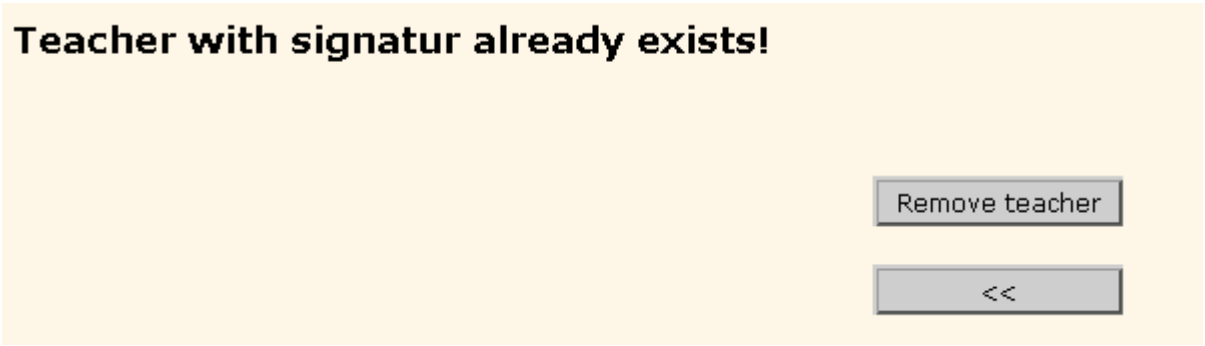


Figur 7-2: Skärmbild av förväntat resultat vid Testfall 1

Testfall 2- Lägg till redan inlagd anställd.

1. Följ steg ett till fyra i Testfall 1.(Detta kräver att Testfall 1 har utförts)

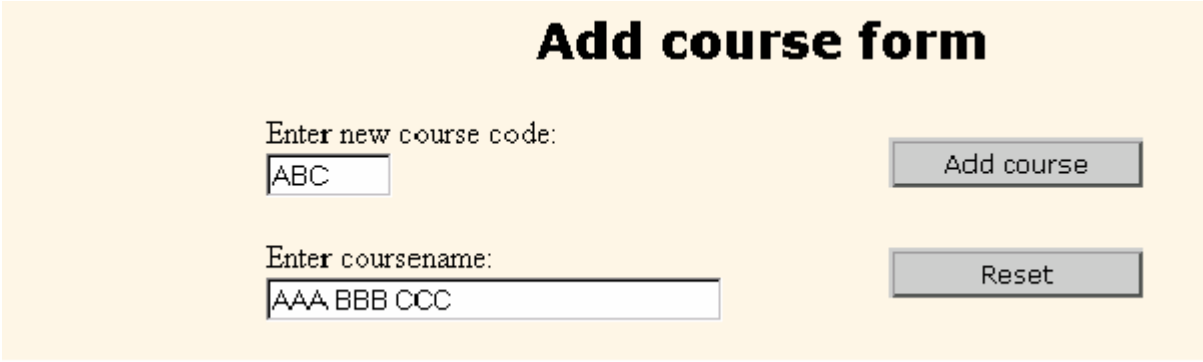
Förväntat resultat av Testfall 2 ses i Figur 7-3



Figur 7-3: Förväntad skärmbild vid Testfall 2.

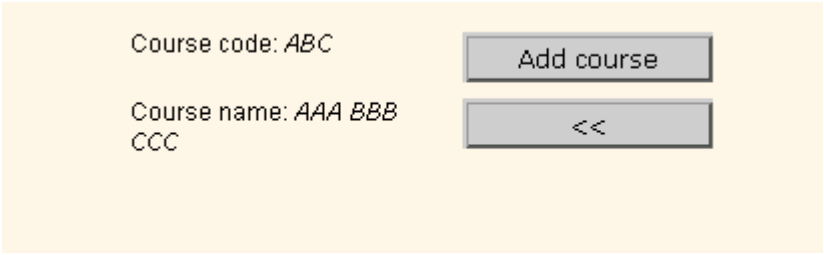
Testfall 3- Lägga till ny kurs.

1. Starta programmet
2. Tryck på knappen ”Add Course”
3. Fyll i attributen enligt Figur 7-4
4. Tryck på knappen ”Add course”



Figur 7-4: In-data till testfall 3.

Förväntat resultat av testfall 3 kan ses i Figur 7-5.



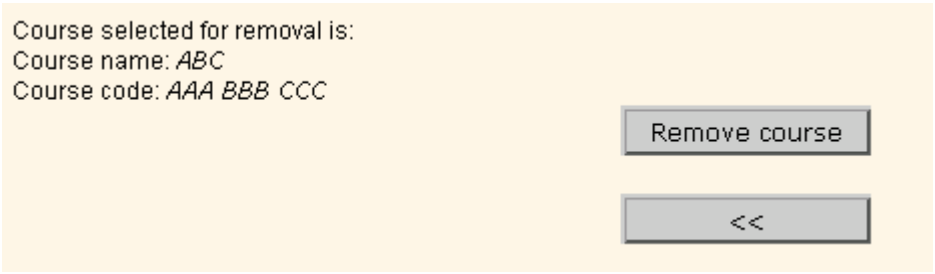
Figur 7-5: Förväntat resultat av Testfall 3.

Testfall 4- Borttagning av kurs

1. Starta programmet
2. Tryck på knappen ”Remove Course”
3. Fyll i listan som i Figur 7-6
4. Tryck på knappen “Remove course”

Figur 7-6: Skärmbild på hur listan ska fyllas i.

Förväntat resultat av Testfall 4 kan ses i Figur 7-7.

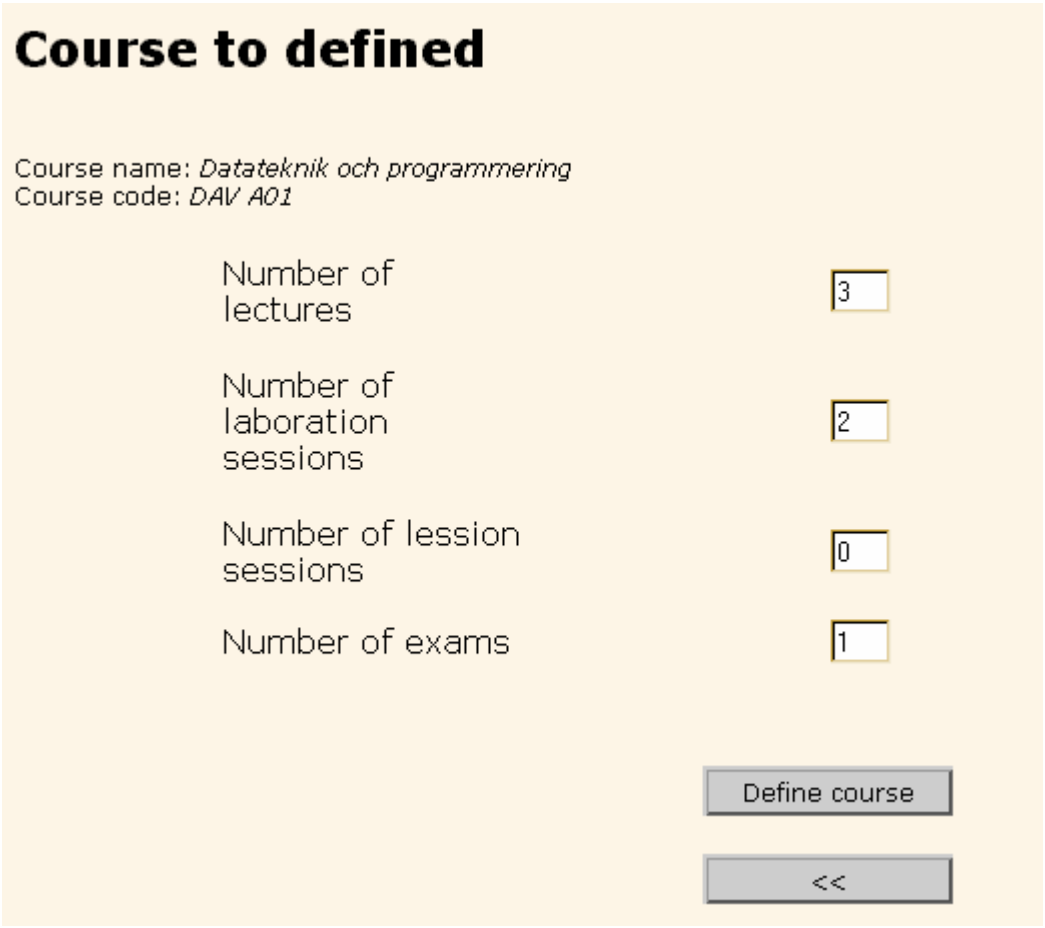


Figur 7-7: Förväntat resultat av Testfall 4.

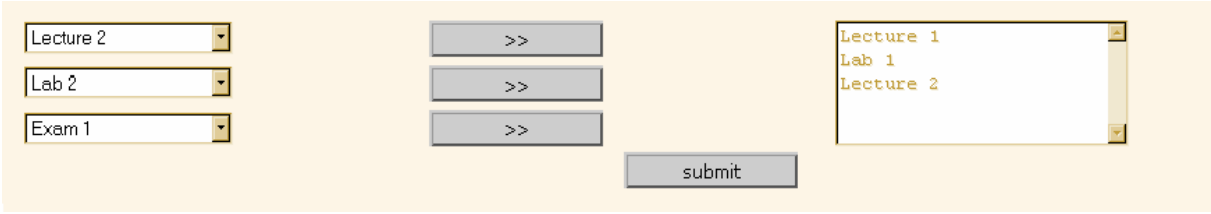
Testfall 5- definiera en kurs

1. Starta programmet
2. Tryck på knappen ”Define course”
3. Välj kursen ”DAV A01 i listan
4. Tryck på knappen ”Define course”
5. Fyll i sidan enligt Figur 7-8
6. Tryck på knappen “ Define course”
7. Markera Lecture 1
8. Tryck på knappen “ >>”
9. Markera Lab 1
10. Tryck på knappen “ >>”
11. Markera Lecture 2
12. Tryck på knappen “ >>”

Förväntat resultat av Testfall 5 ses i Figur 7-9



Figur 7-8: Andra skärmbilden under upplägg av kursen Datateknik & programmering



Figur 7-9: Avslutande skärmbild vid registrering av ett kursupplägg

7.1.3 Genereringgränssnittet

Då detta gränssnitt ej till fullo fungerar har inga fullständiga tester kunnat utföras. Däremot har tester utförts på delar av gränssnittet. Dessa tester har gått ut på att se att lärare och datum bundits upp korrekt till den specificerade aktiviteten eller undervisningstillfället. Testerna har även utförts för att se att regler blir bundna på ett korrekt sätt.

7.1.3.1 Testfall för genereringsgränssnittet

Innan Testfall 6 utförs måste testfall 5 ha gjorts innan.

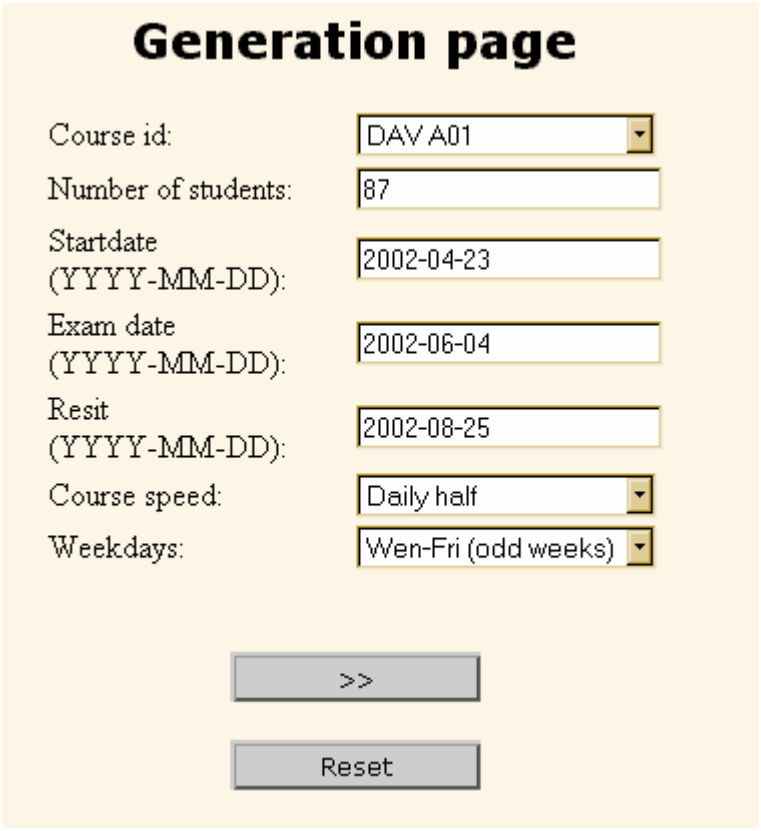
Testfall 6– Generering av ett schema

1. Starta programmet
2. Tryck på knappen ”Generate”
3. Fyll i fälten enligt Figur 7-10
4. Tryck på knappen “>>”
5. Välj ”Martin Blom” i listan
6. Tryck på knappen ”>>”
7. Välj ”Anna Brunnström” i listan
8. Tryck på knappen ”>>”
9. Tryck på knappen ”submit”
10. Markera både Martin Blom och Anna Brunnström i lärarfältet och markera Exam
11. Tryck på knappen ”>>”
12. Markera Martin Blom och Lecture1 och Lecture2
13. Tryck på knappen ”>>”
14. Markera Anna Brunnström och Lecture3
15. Tryck på knappen ”>>”
16. Markera Martin Blom och Lab1
17. Tryck på knappen ”>>”

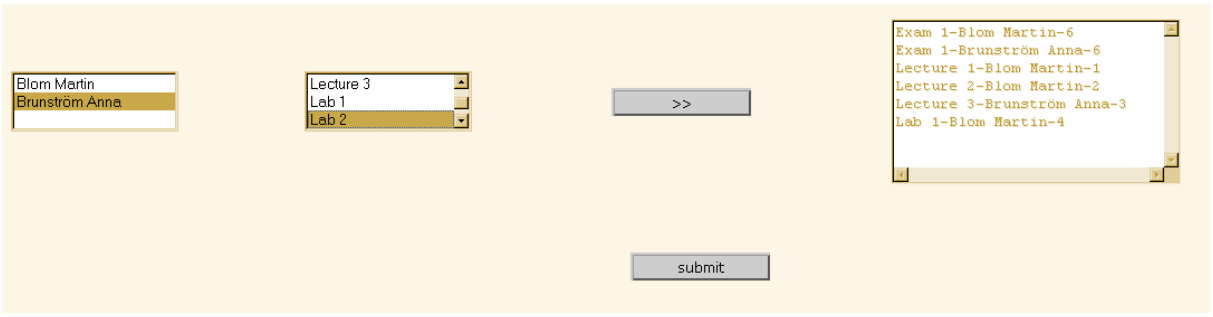
Förväntat resultat av kan ses i Figur 7-11.

18. Tryck på knappen ”submit”
19. Markera nummer ”2” och Lecture 1-3
20. Tryck på knappen ”>>”
21. Markera nummer ”3” och Lab 1
22. Tryck på knappen ”>>”
23. Tryck på knappen ”submit”

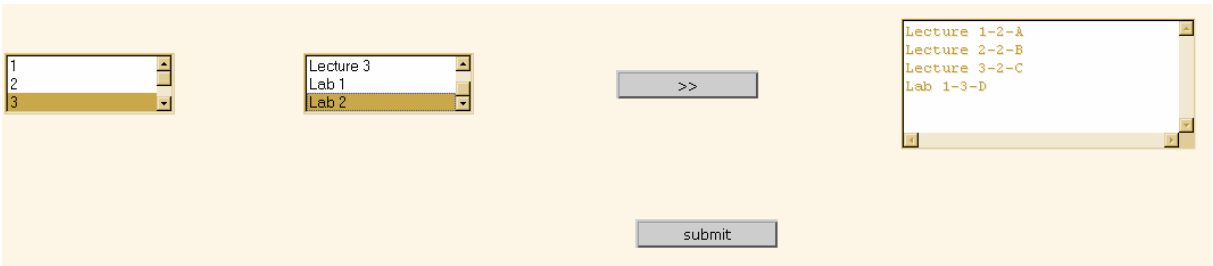
Förväntat resultat återfinns i Figur 7-12



Figur 7-10: Startskärmbild för generering av ett schema.



Figur 7-11: Den tredje skärmbilden vid generering av ett schema.



Figur 7-12: Den avslutande skärmbilden vid schemagenerering.

7.2 Utvärdering

Genom de tester som utförts har vi kunnat verifiera att systemet fungerar utifrån kravspecifikationen. Vi har tyvärr inte haft möjlighet att testa hela vårt system då det har varit vissa problem som har kvarstått under hela projektets gång. Vi har testat alla delar i vårt program men genereringsgränssnittet har inte testats fullt ut.

8 Erfarenheter

Under arbetets gång har vi märkt att inför ett sådant här arbete bör en stund avsättas för att sätta sig ned och planera hur arbetets ska läggas upp. Då vi inte gjorde det blev det ett flertal gånger som vi kände oss tvungna att ändra på en del saker som vi redan gjort för att det inte passade in i vår nya syn på systemet.

Om vi skulle utveckla ett liknande system i framtiden finns det ett flertal olika val som vi antagligen skulle göra på ett annat sätt. Bland annat skulle vi troligtvis bara skapa en databas manuellt först och sedan direkt gå på genereringsdelen av systemet. Detta då den delen av systemet är den krångligaste och därmed tar mest tid. Det är dock väldigt enkelt att fastna och ta fram ett gränssnitt för registrering av nya tupler till databasen. Detta tar också en hel del tid men är inte lika svårt att utveckla i efterhand.

Ett annat val som vi troligtvis skulle fatta på ett annorlunda sätt är det att vi skulle först skapa ett väldigt litet system där vi lär oss att arbete med RPE:n och vilka regler som krävs för att schemagenereringen skulle fungera på ett tillfredsställande sätt. När ett system skapas och blir väldigt komplext är det svårt att genomföra felsökningar vilket leder till att mer tid krävs för att finna vart fel har uppkommit. Detta kan undvikas till viss del om det finns möjlighet att testa olika delar av systemet oberoende av andra. En sådan uppdelning gick i genereringsfasen av vårt system inte att utföra vilket gjorde att det blev svårt att finna vart fel uppkommit när väl ett schema skulle genereras.

Att försöka sätta sig in i hur två olika system, RPE:n och avdelningens schemaläggning fungerar på samma gång har också visat sig varit svårt. Detta på grund av att det inte fanns några dokumenterade testfall för RPE:n samt att lärare inte har samma syn på hur ett schema ska läggas upp. Under arbetets gång har vi även fått större förståelse för hur svårt det kan vara att planera ett schema så att det passar alla parter på ett bra sätt.

Att med hjälp av enkla regler beskriva hur en schemaläggningsprocess fungerar är inte svårt. Att presentera information och ta emot information på ett vettigt sätt från användarna är svårare. Till stor del beror detta på att vi innan detta projekt påbörjades inte hade sysslat med programspråken JavaScript eller JSP.

9 Slutsatser

Den slutsats som vi har dragit efter att ha skrivit denna uppsats är hur svårt det kan vara att använda ett redan befintligt system för att utföra ett arbete. Att använda ett system som till fullo inte är helt färdigutvecklat eller där en klar dokumentation inte existerar är i många situationer krångligt. Vi skulle antingen behövt att Marquées RPE vart helt färdigutvecklad angående schemaläggning eller att upplägg av kurser funnits standardiserat på avdelningen för datavetenskap.

En annan slutsats vi dragit är att det finns en risk att för mycket tid spenderas på vissa delar av systemet utan kontroll om detta egentligen är nödvändigt gentemot den tidsåtgång som krävs. Det behövs en bra planerad och uppföljd tidsplan för att kontrollera detta, men även en sådan är svår att ta fram innan arbetet helt har kommit igång.

Marquées RPE har visat sig att var enkel att använda även om det vid vissa tillfällen blivit problem eftersom den var under utveckling. Under arbetets gång visade det sig att viss funktionalitet saknades i RPE:n som i efterhand tillkommit. Marquée har även fått ytterligare ett sätt hur resurser och aktiviteter kan kopplas samman för att modulera en verksamhet.

Trots att vi inte hann få schemagenereringen att fungera tillfredsställande under projekttiden rekomendera vi ett sådant system för att spara tid vid schemaläggning. Vårt system kan fungera som en utgångspunkt för att skapa ett mer avancerat system.

Referenser

[1] Cascading Style Sheets, level 1, http://www.w3.org/TR/REC-CSS1, 2002-05-16

[2] Java ™2 Platform, Standard Edition, v1.2.2 API Specification, http://java.sun.com/products/jdk/1.2/docs/api/, 2002-05-16

[3] Petter Åström. Databoken JavaScript, 91 7882 484 2, 1999

[4] Marquées XML-RPC, http://xmlrpc.sourceforge.net/, 2002-05-16

[5] XML-RPC Homepage, http://www.xml-rpc.com, 2002-05-16

A Beskrivning av förkortningar och uttryck

Nedan följer en kortfattat beskrivning av förkortningar och uttryck som förekommer i uppsatsen. Det finns även en hänvisning till i vilket kapitel det finns en mer ingående beskrivning av dessa uttryck.

Förkortning/uttryck	**Beskrivning**	**Hänvisning**
RPE	Resource Planing Engine	4 och 4.4
Html	Hyper text markup language	6.1.1.1
SQL	Structured Query Language	6.1.3
JSP	Java Server Pages	6.1.1
XML	eXtended Markup Language	5.4.1
RPC	Remote Procedure Call	5.4.1
Resurs	En resurs kan enklast beskrivs som ett objekt i den verkliga världen som har möjlighet att utföra eller inrymma en händelse.	4.3.2
Aktivitet	En aktivitet är en matchning mot en händelse som en resurs kan utföra.	4.3.4
Regel	Det är reglerna i RPE:n som ger den dess intelligens. Utan dessa skulle RPE:n godtyckligt placera ut aktiviteterna där plats finns.	4.4.3
Resursplaneringsmotor	En resursplaneringsmotor används oftast till att modellera upp en verksamhet med hjälp av olika resurser, aktiviteter samt i de flesta fall några organisationsregler.	4 och 4.4

Tabell A-1: Kortfattad beskrivning av uttryck

B Databastabeller

De tabellerna som finn i vår applikationsdatabas kan ses i tabellen nedan. De understrukna och kursiverade fälten i kolumnen Name är tabellernas namn.

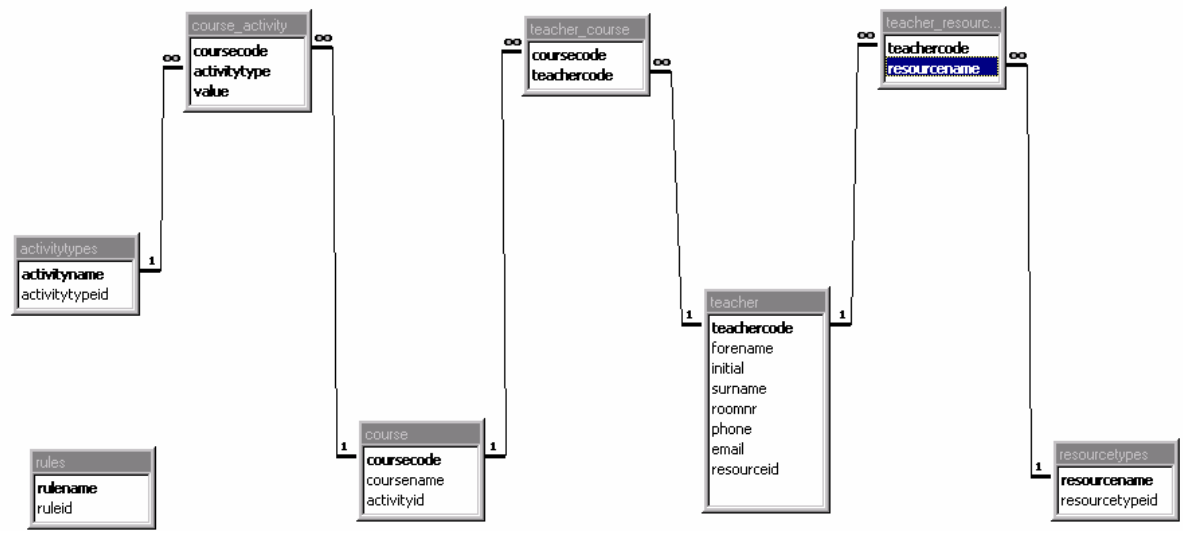
Name	**Type**	**Restrictions**	**Default value**	**Primary key**	**Unique**
activitytypes					
activityname	varchar(30)	NOT NULL	0	x	x
activitytypeid	varchar(16)	NOT NULL	0		x
course					
coursename	varchar(50)	NOT NULL	0		
coursecode	varchar(20)	NOT NULL	0	x	x
activityid	varchar(16)	-	0		x
course_activity					
coursecode	varchar(10)	NOT NULL	0	x	
activitytype	varchar(16)	NOT NULL	0	x	
value	tinyint(3)	NOT NULL	0	x	
resourcetypes					
resourcename	varchar(16)	NOT NULL	0		x
resourcetypeid	varchar(30)	NOT NULL	0	x	x

rules					
rulename	varchar(20)	NOT NULL	""	x	x
ruleid	varchar(16)	NOT NULL	""		x
teacher					
teachercode	varchar(5)	NOT NULL	0	x	x
forename	varchar(25)	NOT NULL	0		
initial	varchar(1)				
surname	varchar(30)	NOT NULL	0		
roomnr	varchar(15)				
phone	varchar(15)				
email	varchar(30)				
resourceid	varchar(16)	NOT NULL	0		x
teacher_course					
coursecode	varchar(15)	NOT NULL	0	x	
teachercode	varchar(5)	NOT NULL	0	x	
teacher_ resourrcetype					
teachercode	varchar(5)	NOT NULL	0	x	
resourcename	varchar(5)	NOT NULL	0	x	

Tabell B-1: Uppbyggnaden av applikationsdatabasen

C E/R diagram över databasmodellen

Nedan visas med hjälp av ett E/R diagram hur de olika tabellerna kopplas samman i vår databas. Det finns även en del kopplingar som inte syns i detta diagram då dessa kopplinga går mot RPE:ns interna databas.



Figur C-1: E/R diagram över databasmodellen

D Regelexempel

Nedan presenteras ett exempel på hur en regel kan implementeras samt hur ett anrop till RPE:n för att skapa den regeln kan se ut. Nedanstående regel är endast ett exempel och det är inte säkert att regeln är möjlig att utnyttja i ett färdigt system. Alla parametrar som krävs för ett sådant anrop till RPE:n beskrivs och diskuteras.

Reglen nedan används för att specificera när en aktivitet inte får förekomma. I detta fall ges straffpoäng om aktiviten lägg mellan kl 18 och kl 8.

```
result = 0;
iter = activity.occurrences.iterator();
while(iter.hasNext()) {
        occ = iter.next();
        if(occ.start.hour < 8 || occ.end.hour > 18)
                result += 1;
}
result;
```

Om ovanstående regel lagras i strängen script kan ett anrop till RPE:n för att registrera upp denna regel se ut som följer.

```
client.invoke(”RuleHome.createule”, new Object[]
        {session,
        ”ordinaryWorkDay”,
        ”This rule prohibits that activities can be placed”
        + ”during a time when you normaly do  not work”,
        ”ACTIVTIES”,
        ”\”Activity \” + activity.name + \” can not have”
        + “a start hour before 8 AM or end after 6 PM\”;”,
        script,
        new Boolean(true),
        new Integer(100) });
```

Detta anrop visar även hur vi kommer åt RPE:ns olika operationer genom XML-RPC anrop. Den funktion som vi här har anropat är funktionen createRule i klassen RuleHome. Alla parametrar inkluderas därefter i ett objekt som skickas till RPE:n som själv plockar ut de olika parametrarna som krävs ur objektet. Om ett felaktigt antal parametrar skickats med skickar RPE:n tillbaka ett felmeddelande om detta. Parametrarna till funktionen som används i detta fall är beskrivna i nedanstående tabell.

Parametertyp	Parameternamn	Beskrivning
String	Session	En strängidentifierare som anger vilken session som har initierat anropet
String	Name	Namnet på den nya regeln
String	Description	En beskrivning av den nya regeln
String	AppliesTo	Denna sträng talar om vad regel kan knytas till. Antigen har strängen värdet ”ACTIVITY” eller så har den värdet ”RESOURCE”
String	Violation	En sträng innehållande det felmeddelande som skickas tillbaka om en aktivitet eller resurs bryter mot regln.
String	Script	I denna sträng ligger själva scriptdelen av regeln. Den del som ska exekvera när kontroll av regeln utförs.
Boolean	TimeConstraint	Ett booleskt värde som anger som regeln är ett organisations regel (False) eller en tidsregel (True).
Int	Weight	Regelns vikt, används när flera regelbrott har uppkommit och RPE:n vill bestämma vilken av dessa regler som gör minst skada av att brytas.

Tabell D-1: Beskrivning av parametrarna i ett funktionsanrop till RPE:n

E Källkod för systemet

I denna bilaga återfinns den källkod som systemet bygger på. Bilagan är uppdelad i tre olika delar där varje del bearbetar olika delar av systemet. Den första delen av bilagan berör de olika filerna som ingår i initaitorn. Därefter bearbetar bilagan registeringsgränssnittet för att slutligen avsluta med genereringsgränssnittet. Alla filnamn är skrivna i fet stil.

E.1 Initiatorn

Nedan följer källkoden som används av initiatorn.

E.1.1 Init.jsp

```
<%@page import="java.util.Vector" %>
            <%@page import="initiator2.Servletinit" %>

<html>
<head>
   <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>Initator</title>
</head>
<body>
<%
   Vector parameters = new Vector();
   Servletinit initiator = new Servletinit();
   initiator.init();
   String mainRID = initiator.addMainResourceType("employee");
   String mainAID = initiator.addMainActivityType("course");
   initiator.addResourceTypes(mainRID);
   initiator.addActivityTypes(mainAID);
   initiator.addResource();
   initiator.addActivity();
   Vector params = new Vector();
   params.addElement("startHour");
   params.addElement("endHour");

   String rule =                "result = 0;"
                              + "i = activity.occurrences.iterator();"
                              + "while ( i.hasNext() ) {"
                              + "o = i.next();"
                              + "if ( o.start.hourOfDay <= startHour”
                              + ”|| o.start.hourOfDay >= endHour ) {"
                              + "result += 1.0; } }"
                              + "result;";

   initiator.addRules("NotBeforeOrAfter",
                              "The current activity can not start before or after decided times",
                              "ACTIVITY",
```

```
                              rule,
                              "\"Activity \" + activity.name + \" is not
                              allowed to start before \" + startHour + \" or
                              to end after \" + endHour;",
                              true,
                              10,
                              params);

    params = new Vector();
    params.addElement("preceding_activity");
    String rule2 =                 "result = 1;"
                              + "parents = activity.parents.iterator();"
                              + "while ( parents.hasNext() ) {"
                              + "act = parents.next();"
                              + "if ( act.id == preceding_activity ) {"
                              + "occs = activity.occurrences.iterator();"
                              + "while ( occs.hasNext() ) {"
                              + "occ = occs.next();"
                              + "occurrencesList = act.parents.occurrences;"
                              + "occurrencesList.add( act.occurrences );"
                              + "occurrences = occurrencesList.iterator();"
                              + "while ( occurrences.hasNext() ) {"
                              + "o = occurrences.next();"
                              + "if ( occ.start.before( o.start ) ) {"
                              + "result += 1;"
                              + "} } } } }"
                              + "result;";

    initiator.addRules("activityOrder",
                              "The current activity must be preceded by
                              specified activity",
                              "ACTIVITY",
                              rule2,
                              "\"Activity \" + activity.name + \" is not
                              allowed to start before specified activity\";",
                              true,
                              100,
                              params); %>
</body>
</html>
```

E.1.2 Servletinit.java

```
    package initiator;

    import javax.servlet.*;
    import javax.servlet.http.*;
    import java.io.*;
    import java.util.*;
    import java.lang.*;
    import java.sql.ResultSet;
    import marquee.util.database.*;
    import initiator.Operations;
    import java.util.Vector;

    public class Servletinit extends HttpServlet {
      private static final String CONTENT_TYPE = "text/html";
      private static String empty = "";
```

```
private static String session = "";
private static String s = "\"";
private static String k = ";";
private static String id = "";
private static String complete = "";

public static String addMainResourceType(String main) {
 try {
          String resourceupdate = "UPDATE resourcetypes SET resourcetypeid=";
          String resourceend = "WHERE resourcename=";
          String rname = "resourcename";
          String select = "SELECT * FROM resourcetypes where resourcename=\"" + main + "\"";
          Operations op = new Operations();
          ResultSet rt = op.query(select);
          rt.next();

          id = op.runOp("ResourceTypeHome.createResourceType", new Object[]
          {new String(session), new String(rt.getString(rname)), new String(empty) } ).toString();
          complete = resourceupdate + s + id + s + resourceend + s + rt.getString(rname) + s + k;
          op.query(complete);
          op.closeTransaction();
 }//End of tryblock
 catch ( Exception e ) {
          e.printStackTrace();
 }
 return id;
}//End of addMainResourceType

public static String addMainActivityType(String main) {
  try {
          String activityupdate = "UPDATE activitytypes SET activitytypeid=";
          String activityend = "WHERE activityname=";
          String aname = "activityname";
          String select = "SELECT * FROM activitytypes where activityname=\"" + main + "\"";
          Operations op = new Operations();
          ResultSet at = op.query(select);
          at.next();

          id = op.runOp("ActivityTypeHome.createActivityType", new Object[]
          {new String(session), new String(at.getString(aname)), new String(empty) } ).toString();
          complete = activityupdate + s + id + s + activityend + s + at.getString(aname) + s + k;
          op.query(complete);
          op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
  return id;
}//End of addMainActivityType

public static void addResourceTypes(String mainRID) {
  try {
   String resourceupdate = "UPDATE resourcetypes SET
                           resourcetypeid=";
    String resourceend = "WHERE resourcename=";
    String rname = "resourcename";
    String select = "SELECT * FROM resourcetypes";
    Operations op = new Operations();
    ResultSet rt = op.query(select);
```

```
    while(rt.next()) {
          id = op.runOp("ResourceTypeHome.createResourceType", new Object[]
          {new  String(session),  new  String(rt.getString(rname)),  new  String(empty),  mainRID  }
          ).toString();
          complete = resourceupdate + s + id + s + resourceend + s + rt.getString(rname) + s + k;
      op.query(complete);
    }
    rt.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
} //End of addResourceTypes()

public static void addActivityTypes(String mainAID)
{
  try {
   String activityupdate = "UPDATE activitytypes SET
                                          activitytypeid=";
   String activityend = "WHERE activityname=";
    String aname = "activityname";
    String select = "SELECT * FROM activitytypes";
    Operations op = new Operations();

    ResultSet at = op.query(select);
    while(at.next()) {
          id = op.runOp("ActivityTypeHome.createActivityType", new Object[]
          {new String(session), new String(at.getString(aname)), new String(empty) } ).toString();
          complete = activityupdate + s + id + s + activityend + s + at.getString(aname) + s + k;
      op.query(complete);
    }
    at.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
}//End of addActivityTypes()

public static void addResource(){
  try {
    String resourceupdate = "UPDATE teacher SET resourceid=";
    String resourceend = "WHERE teachercode=";
    String teachername = "teachername";
    String surname = "surname";
    String code = "teachercode";
    String resourcename = "resourcename";
    String select = "SELECT * FROM teacher";
    String query2 = "SELECT * FROM teacher_resourcetype WHERE
                             teachercode=";
    Operations op = new Operations();

    ResultSet r = op.query(select);
    r.next();
    ResultSet rt = op.query(query2 +s+ r.getString(code) +s+ k);
    r.beforeFirst();
    while(r.next()) {
```

```
      rt = op.query(query2 + s + r.getString(code) + s + k);
      rt.next();
          id = op.runOp("ResourceHome.createResource", new
          Object[]
          {new              String(session),              new              String(r.getString(code)),
          r.getString(surname),rt.getString(resourcename)}
          ).toString();
          complete = resourceupdate + s + id + s + resourceend + s +
                          r.getString(code) + s + k;
      op.query(complete);
    }
    r.close();
    rt.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
}//End of addResource()

public static void addActivity(){
  try {
    String activityupdate = "UPDATE course SET activityid=";
    String activityend = "WHERE coursecode=";
    String coursename = "coursename";
    String code = "coursecode";
    String activitytypeid = "activitytypeid";
    String select = "SELECT * FROM course";
    String query2 = "SELECT * FROM activitytypes WHERE
                            activityname=" + s + "course" + s + k;

    Operations op = new Operations();
    ResultSet a = op.query(select);
    ResultSet at = op.query(query2);
    at.next();
    String aid = at.getString(activitytypeid);
    while(a.next()) {
      id = op.runOp("ActivityHome.createActivity", new Object[]
      {session, a.getString(code), a.getString(coursename),
           aid}).toString();
      complete = activityupdate + s + id + s + activityend + s +
                          a.getString(code) + s + k;
      op.query(complete);
    }
    a.close();
    at.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
}//End of addActivity()


public static void removeResourceTypes()
{
  try {
    String rid = "resourcetypeid";
    String select = "SELECT * FROM resourcetypes";
```

```
    Operations op = new Operations();

    ResultSet rt = op.query(select);
    while(rt.next()) {
      op.runOp("ResourceTypeHome.removeResourceType", new
           Object[]{session, rid});
    }
    rt.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
} //End of removeResourceTypes()

public static void removeActivityTypes()
{
  try {
    String aid = "activitytypeid";
    String select = "SELECT * FROM activitytypes";
    Operations op = new Operations();

    ResultSet at = op.query(select);
    while(at.next()) {
      op.runOp("ActivityTypeHome.removeActivityType", new
           Object[]
      {session, aid});
    }
    at.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
}//End of removeActivityTypes()

public static void removeResource()
{
  try {
    String resourceid = "resourceid";
    String select = "SELECT * FROM teacher";
    Operations op = new Operations();

    ResultSet r = op.query(select);
    while(r.next()) {
      op.runOp("ResourceHome.removeResource", new Object[]
      {session, r.getString(resourceid)});
    }
    r.close();
    op.closeTransaction();
  }//End of tryblock
  catch ( Exception e ) {
    e.printStackTrace();
  }
} //End of removeResource()

public static void removeActivity() {
  try {
    String activityid = "activityid";
```

```
      String select = "SELECT * FROM course";
      Operations op = new Operations();

      ResultSet r = op.query(select);
      while(r.next()) {
        op.runOp("ActivityHome.removeActivity", new Object[]
        {session, r.getString(activityid)});
      }
      r.close();
      op.closeTransaction();
    }//End of tryblock
    catch ( Exception e ) {
      e.printStackTrace();
    }
  } //End of removeActivity()

  public static String addRules(String name,
                                String desc,
                                String type,
                                String script,
                                String violation,
                                boolean time,
                                int weight,
                                Vector params) {
    try {
      Operations op = new Operations();
      id = op.runOp("RuleHome.createRule", new Object[]
           {session, name, desc, type, script, violation,
                new Boolean(time), new Integer(weight),
                params}).toString();
      op.query("insert into rules values(" + s + name + "\",\""
                   + id + "\");");
      op.closeTransaction();
    }//End of tryblock
    catch ( Exception e ) {
      e.printStackTrace();
    }
    return id;
  } //End of addRules

  public void init() {
    Operations op = new Operations();
    op.serialize();
  } //End of init
}
```

E.1.3 Operations.java (för initiatorn)

```
package initiator;

import marquee.xmlrpc.*;
import java.sql.ResultSet;
import marquee.util.database.*;
import marquee.xmlrpc.serializers.VectorSerializer;
import marquee.xmlrpc.serializers.HashtableSerializer;
import marquee.xmlrpc.serializers.CollectionSerializer;
import marquee.xmlrpc.serializers.MapSerializer;
```

```
   import marquee.xmlrpc.serializers.ObjectArraySerializer;

   public class Operations {
    private XmlRpcClient client;
    client = new XmlRpcClient("127.0.0.1", 9999, "");
    private Transaction t = new Transaction( "kauscheduler" );

     public Object runOp(String operation, Object[] param)
       throws Exception {
         return client.invoke(operation,param);
     }

     public ResultSet query(String q)
       throws Exception {
         return t.doQuery(q);
     }

     public void closeTransaction()
       throws Exception {
        t.commit();
        t.getConnection().close();
     }

     public void bindRule(String ruleID,
                                  String id,
                                  String method,
                                  String [] params)
       throws Exception{
         runOp(method,new Object[]
                   {new String(""), id, ruleID, params});
     }

     public void serialize() {
       XmlRpcSerializer.registerCustomSerializer(
               new MapSerializer());
       XmlRpcSerializer.registerCustomSerializer(
               new CollectionSerializer());
       XmlRpcSerializer.registerCustomSerializer(
               new VectorSerializer());
       XmlRpcSerializer.registerCustomSerializer(
               new HashtableSerializer());
       XmlRpcSerializer.registerCustomSerializer(
               new ObjectArraySerializer());
     }
}
```

E.2 Registereringsgränssnittet

Nedan återfinns källkoden som används under registeringsgränssnittet.

E.2.1 Index.html

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

```
   <title>
   index
   </title>
   </head>
   <frameset rows="20%,70%,10%">
               <frame src="Topframe.html" name="topframe"
                              noresize="yes">
               <frame src="Mainpage.html" name="actionframe"
                              noresize="yes">
               <frame src="Bottomframe.html" name="bottomframe"
                              noresize="yes">
   </frameset>
</html>
```

E.2.2 Topframe.html

```
   <html>
   <head>
       <link rel="stylesheet" href="kau_style.css" type="text/css">
   <title>
   Topframe
   </title>
   </head>
   <body>
     <p>
     <table>
       <tr align="left">
         <td align="center" width=20%>
           <img src="C:\sandbox_gohlson\KauScheduler\kausol.gif"
                              width="90" height="90">
         </td>
         <td align="left" width=80%>
           <span class="f3">K</span><span class = "f2">arlstad</span>
           <span class="f3">U</span>
               <span class="f2">niversity </span>
           <br>
           <span class="f3">D</span>
               <span class="f2">epartment of </span>
           <span class="f3">C</span>
               <span class="f2">omputer      </span>
           <span class="f3">S</span>
               <span class="f2">cience       </span>
         </td>
       </tr>
     </table>
     </p>
   </body>
</html>
```

E.2.3 Bottomframe.html

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

```
   </title>
   </head>
   <body>
   <script src="Scripts.js">
   </script>
   <form name="myForm" method="post">
     <table align="right" width="100%">
       <tr>
         <td align="right">
           <input type="button" value="Mainpage"
                          class="orientbutton" onClick="go('Main');">
         </td>
       </tr>
     </table>
   </form>
   </body>
</html>
```

E.2.4 Mainpage.html

```
   <html>
   <head>
       <link rel="stylesheet" href="kau_style.css" type="text/css">
   <title>
   Mainframe
   </title>
   </head>
   <body>

   <table width="60%" height="60%" align="center">
     <tr align="left">
       <td align="center" width=50% valign="bottom">
         <form action="AddTeacher.jsp" method="post">
           <input type="submit" name="Start"
                          value="Add teacher" class="button">
           <br><br><br>
         </form>
       </td>
       <td align="center" width=50%>
         <form action="AddCourse.html" method="post">
           <input type="submit" name="Start"
                          value="Add course" class="button">
           <br><br><br>
         </form>
       </td>
     </tr>
     <tr align="left">
       <td align="center" width=50%>
         <form action="RemoveTeacher.jsp" method="post">
           <input type="submit" name="Start"
                          value="Remove teacher" class="button">
           <br><br><br>
         </form>
       </td>
       <td align="center" width=50%>
         <form action="RemoveCourse.jsp" method="post">
           <input type="submit" name="Start"
```

```
                            value="Remove course" class="button">
          <br><br><br>
        </form>
      </td>
    </tr>
    <tr align="left">
      <td align="center" width=50%>
        <form action="UpdateTeacher.jsp" method="post">
          <input type="submit" name="Start"
                            value="Update teacher" class="button">
          <br><br><br>
        </form>
      </td>
      <td align="center" width=50%>
        <form action="CourseDescription.jsp" method="post">
          <input type="submit" name="Start"
                            value="Define course" class="button">
          <br><br><br>
        </form>
      </td>
    </tr>
    <tr>
      <td align="center" width=50%>
        <form action="ShowTables.jsp" method="post">
          <input type="submit" name="Show"
                            value= "Show teachers" class="button">
          <br><br><br>
        </form>
      </td>
      <td align="center" width=50%>
        <form action="ShowTables.jsp" method="post">
          <input type="submit" name="Show"
                            value="Show courses" class="button">
          <br><br><br>
        </form>
      </td>
    </tr>
  </table>
  </body>
</html>
```

E.2.5 kau_style.css

```
  body
  {
            background-color: #FDF5E6;
            link-color: #000000;
            alink-color: #FFCC00;
            vlink-color: #000000;
            text-color: #000000;
            text-size: 12px;
  }

  select
  {
          width: 150px;
  }
```

```
td
{
        width: 150px;
}

text
{
        width: 150px;
}

.orientbutton
{
            color: #000000;
            background-color: #CDCDCD;
            font-family: verdana, arial;
            font-size: 12px;
        font-style: normal;
            border-top: #CCCCCC 3px outset;
            border-left: #CCCCCC 3px outset;
            border-bottom: #666666 2px inset;
            border-right: #666666 2px inset;
            width: 125px;
            height: 25px;
}

.f3
{
            font-size: 30px;
            font-style: normal;
}

.f2
{
            font-size: 26px;
            font-style: normal;
}

.digitinput
{
        width: 30px;
}

table
{
        cellspacing="0";
        cellpadding="0";
}

h3
{
            COLOR: #808080;
            FONT-FAMILY: Verdana, Arial, Helvetica;
            FONT-SIZE: 16px;
            font-style: bold;
}

h2
{
```

```
            COLOR: #000000;
            font-family: Verdana, Arial, Helvetica;
            font-size: 18px;
            font-style: bold;
}

h1
{
            COLOR: #000000;
            font-family: Verdana, Arial, Helvetica;
            font-size: 24px;
            font-style: bold;
}

.infotext
{
            color: #000000;
            font-family: Verdana, Arial, Helvetica;
            font-size: 12px;
}

.statictext
{
      color: #000000;
      font-family: Arial;
      font-size: 12px;
}

.variabletext
{
      color: #000000;
      font-family: Verdana, Arial, Helvetica;
      font-size: 12px;
      font-style: italic;
}
```

E.2.6 Scripts.js

```
function go(command) {
  if(command == 'Main') {
    parent.frames[1].document.location = "Mainpage.jsp";
  }
  else {
    document.myForm.comm.value = document.myForm.Update.value;
    document.myForm.action = command;
    document.myForm.submit();
  }
  return true;
}

function check(variable, command) {
  if(variable.value.length < 1) {
    alert("Some of the data isn't filled in correctly");
    variable.focus();
    return false;
  }
  go(command);
}
```

```
function number(s) {
  var i = 0;
  while(!s.options[i].selected) {
    i++;
  }
  return i;
}

function sendToFrom(to, from) {
  var tt = to.value;
  var ff = from.options[from.selectedIndex].text;
  var reg1 = new RegExp("\n");
  if(!reg1.test(tt)) {
    to.value = from.options[from.selectedIndex].text + "\n";
    to.rows++;
  }
  else if(tt.indexOf(ff) == -1) {
    to.value =to.value+from.options[from.selectedIndex].text+"\n";
    to.rows++;
  }
  return true;
}

function concatToFrom(to, from) {
  var i = j = 0;
  var temp;
  while(i < to.options.length) {
    j = 0;
    if(to.options[i].selected) {
      temp = to.options[i].text;
      while(j < from.options.length) {
        if(from.options[j].selected) {
          if(to.options[i].text.indexOf(from.options[j].text)==-1)
              {
            temp = temp + " " + from.options[j].text;
          }
        }
        j++;
      }
      to.options[i].text = temp;
    }
    i++;
  }
  return true;
}

function moveToFrom(to, from1, from2) {
  var seperator = "-";
  var check, id;
  var reg1 = new RegExp("\n");
  var i = 0, j;
  while(i < from1.options.length) {
    if(from1.options[i].selected) {
      j = 0;
      while(j < from2.options.length) {
        if(from2.options[j].selected) {
          check = from1.options[i].text + seperator +
                           from2.options[j].text;
```

```
            id = from1.options[i].value;
            if(!reg1.test(to.value)) {
              to.value = check + seperator + id + "\n";
              to.rows++;
            }
            else if(to.value.indexOf(check) == -1) {
              to.value = to.value + check + seperator + id + "\n";
              to.rows++;
            }
          }
          j++;
        }
      }
      i++;
    }
    return true;
  }

  function moveToFromHours(to, from1, from2) {
    var seperator = "-";
    var check, id;
    var reg1 = new RegExp("\n");
    var i = 0, j;
    while(i < from1.options.length) {
      if(from1.options[i].selected) {
        j = 0;
        while(j < from2.options.length) {
          if(from2.options[j].selected) {
            check = from1.options[i].text + seperator +
                            from2.options[j].text;
            id = from1.options[i].value;
            if(!reg1.test(to.value)) {
              to.value = check + seperator + id + "\n";
              to.rows++;
            }
            else if(to.value.indexOf(from1.options[i].text) == -1) {
              to.value = to.value + check + seperator + id + "\n";
              to.rows++;
            }
          }
          j++;
        }
      }
      i++;
    }
    return true;
}
```

E.2.7 AddCourse.html

```
  <html>
  <head>
      <link rel="stylesheet" href="kau_style.css" type="text/css">
  <title>
  Add Course
  </title>
  </head>
```

```
    <body>
    <script src="Scripts.js">
    </script>
    <center><h1>Add course form</h1></center>
    <form name="myForm" method="post">
      <input type="hidden" name="comm" value="">
      <table width="50%" align="center">
        <tr>
          <td align="left" width=50% rowspan=3>
            Enter new course code:
            <br>
            <input size=7 name="coursecode">
            <br><br>
            Enter coursename:
            <br>
            <input size=30 name="cname">
          </td>
          <td align="center" width=50% valign="middle" height=25>
            <input type="button" name="Update" value="Add course"
                                 class="orientbutton" OnClick="check(document.myForm.coursecode,
                                 'AddCourseform.jsp');">
          </td>
        </tr>
        <tr>
          <td align="center" width=50% valign="middle" height=25>
            <input type="reset" name="Reset" value="Reset"
                                 class="orientbutton">
          </td>
        </tr>
      </table>
    </form>
    </body>
</html>
```

E.2.8 AddCourseform.jsp

```
    <%@page import="java.sql.ResultSet" %>
    <%@page import="kauscheduler.Operations"%>

    <html>
    <head>
        <link rel="stylesheet" href="kau_style.css" type="text/css">
    <title>
    Add Course form
    </title>
    </head>
    <body>
    <script src="Scripts.js">
    </script>

    <form name="myForm" method="post">
      <input type="hidden" name="comm" value="">
      <%
        Operations op = new Operations();
        String coursecode = request.getParameter("coursecode");
        String coursename = request.getParameter("cname");
        String s ="\"";
        String complete = "select * from course where coursecode=" + s
```

```
                                        + coursecode + s + ";";
      ResultSet rs = op.query(complete);
      rs.last();
      op.closeTransaction();

      if(rs.getRow() != 0) { %>
        <center>
          <h2>Course with coursecode
                          <%=coursecode%> already exists!</h2>
        </center>
        <br><br>
        <center>
          <input type="button" name="Remove existing course"
                          value="Remove course"
                          class="orientbutton" onClick="go('RemoveCourse.jsp');">
          <br><br>
          <input type="button" name="Error" value="<<"
                           class="orientbutton"
                          onClick="go('AddCourse.html');">
          <br><br>
        </center>
    <%
      rs.close();
      }
      else { %>
        <table align="center">
          <tr>
            <td align="left" rowspan=2 width="80%">
              <span class="statictext">
                Course code:
              </span>
              <span class="variabletext">
                <%= coursecode %>
              </span><br><br>
              <span class="statictext">
                Course name:
              </span>
              <span class="variabletext">
                <%= coursename %>
              </span>
            </td>
            <td align="center" valign="middle" width="20%">
              <input type="hidden" name="coursecode" value=
                          <%= s + coursecode + s%>>
              <input type="hidden" name="cname" value=
                          <%=s + coursename + s %>>
              <input type="button" name="Update" value="Add course"
                          class="orientbutton"
                          onClick="go('Update.jsp');">
            </td>
          </tr>
          <tr>
            <td align="center" valign="middle" width="20%">
              <input        type="submit"        name="back"        value="<<"        class="orientbutton"
onClick="go('AddCourse.html');">
            </td>
          </tr>
        </table>
      <%
```

```
        rs.close();
        } %>
    </form>
    </body>
</html>
```

E.2.9 AddTeacher.jsp

```
    <%@page import="java.sql.ResultSet" %>
    <%@page import="kauscheduler.Operations" %>

    <html>
    <head>
        <link rel="stylesheet" href="kau_style.css" type="text/css">
    <title>
    Add Teacher
    </title>
    </head>
    <body>
    <script src="Scripts.js">
    </script>
    <form name="myForm" method="post">
      <input type="hidden" name="comm" value="">
      <center><h1>Add teacher form</h1></center><br>
      <span class="statictext">
        <table align="center" >
          <tr><td align="left" width="50%">
              Enter new teachers signatur:
            </td><td align="left" width="50%">
              <input size=5 name="sig">
            </td></tr>
          <tr><td align="left" width="50%">
              Enter forename:
            </td><td align="left" width="50%">
              <input size=20 name="fname">
            </td></tr>
          <tr><td align="left" width="50%">
              Enter initial if any:
            </td><td align="left" width="50%">
              <input size=5 name="initial">
            </td></tr>
          <tr><td align="left" width="50%">
              Enter surname:
            </td><td align="left" width="50%">
              <input size=20 name="sname">
            </td></tr>
          <tr><td align="left" width="50%">
              Enter email:
            </td><td align="left" width="50%">
              <input size=30 name="email">
            </td></tr>
          <tr><td align="left" width="50%">
              Enter room number:
            </td><td align="left" width="50%">
              <input size=10 name="roomnr">
            </td></tr>
          <tr><td align="left" width="50%">
```

```
          Enter phone number:
        </td><td align="left" width="50%">
          <input size=10 name="phonenr">
        </td></tr>
      <tr><td align="left" width="50%">
          Occupation:
        </td><td align="left" width="50%">
          <%
            Operations op =  new Operations();
            ResultSet rs;
                rs = op.query("select * from resourcetypes");
            rs.next();
            String s = "\""; %>
            <select name="resourcename">
              <%
              do { %>
                <option value=
                            <%= s + rs.getString("resourcename") + s %> >
                  <%=rs.getString("resourcename")%>
                </option>
              <%
                }while(rs.next()); %>
            </select>
        </td></tr>
    </table>
  </span>
  <%
    op.closeTransaction();
    rs.close(); %>
  <br><br>
  <table align="center">
    <tr><td align="left" valign="top" width="33%">
        <input type="button" name="Update" value="Add teacher"
                            class="orientbutton"
                            onClick="check(document.myForm.sig,
                            'AddTeacherform.jsp');">
      </td><td align="left" valign="top" width="33%">
        <input type="reset" name="Reset" value="Reset"
                            class="orientbutton">
      </td></tr>
  </table>
</form>
</body>
</html>
```

E.2.10 AddTeacherform.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Add Teacherform
</title>
</head>
<body>
```

```
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <%
    Operations op = new Operations();
    String s = "\"";
    String teachercode = request.getParameter("sig");
    String forename = request.getParameter("fname");
    String surname = request.getParameter("sname");
    String initial = request.getParameter("initial");
    String email = request.getParameter("email");
    String roomnr = request.getParameter("roomnr");
    String phonenr = request.getParameter("phonenr");
    String resourcename = request.getParameter("resourcename");
    String complete = "select * from teacher where teachercode=" +
                                          s + teachercode + s + ";";
    ResultSet rs = op.query(complete);
    rs.last();

    if(rs.getRow() != 0) { %>
      <h2>Teacher with signatur
            <%= teachercode %> already exists!</h2>
      <br><br>
      <center>
        <input type="button" name="Remove existing teacher"
                             value="Remove teacher"
                             class="orientbutton"
                             onClick="go('RemoveTeacher.jsp');">
        <br><br>
        <input type="button" name="Error" value="<<"
                             class="orientbutton"
                             onClick="go('AddTeacher.jsp');">
        <br><br>
      </center>
      <%
        rs.close();
        op.closeTransaction();
    }
    else { %>
      <span class="statictext">
        Signature: <span class="variabletext">
                             <%= teachercode %></span><br>
        Forename: <span class="variabletext">
                             <%= forename %></span><br>
        Initial: <span class="variabletext">
                             <%= initial %></span><br>
        Surname: <span class="variabletext">
                             <%= surname %></span><br>
        Email: <span class="variabletext">
                             <%= email %></span><br>
        Roomnr: <span class="variabletext">
                             <%= roomnr %></span><br>
        Phonenr: <span class="variabletext">
                             <%= phonenr %></span><br>
        Occupation: <span class="variabletext">
                             <%=resourcename %></span><br>
      </span>
```

```
      <input type="hidden" name="sig"
                           value=<%= s + teachercode + s %>>
      <input type="hidden" name="fname"
                           value=<%= s + forename + s %>>
      <input type="hidden" name="sname"
                           value=<%= s + surname + s %>>
      <input type="hidden" name="initial"
                           value=<%= s + initial + s %>>
      <input type="hidden" name="email"
                           value=<%= s + email + s %>>
      <input type="hidden" name="roomnr"
                           value=<%= s + roomnr + s %>>
      <input type="hidden" name="phonenr"
                           value=<%= s + phonenr + s %>>
      <input type="hidden" name="resourcename"
                           value=<%= s + resourcename + s %>>
      <center>
        <input type="button" name="Update" value="Add teacher"
                           class="orientbutton"
                           onClick="go('Update.jsp');"><br>
      </center>
      <br>
      <center>
        <input type="button" name="back" value="<<"
                           class="orientbutton"
                           onClick="go('AddTeacher.jsp');">
      </center>
      <%
        rs.close();
        op.closeTransaction();
    } %>
</form>
</body>
</html>
```

E.2.11 RemoveCourse.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Remove Course
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form  name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <center><h1>Remove course</h1></center><br>
  <%
    Operations op = new Operations();
    ResultSet rs = op.query("select * from course");
    rs.next();
```

```
    String s = "\""; %>

  <table width="50%" align="center">
    <tr align="center">
      <td align="left" width="75%" rowspan="2">
        Course code: <br>
        <select name="coursecode">
          <%
            do { %>
              <option value=<%=s+rs.getString("coursecode")+s %> >
                <%=rs.getString("coursecode")%>
              </option>
            <%
              }while(rs.next()); %>
        </select>
      </td>
      <%
        rs.close();
        op.closeTransaction(); %>
      <td align="left" width="25%">
        <input type="button" name="Update" value="Remove course"
                             class="orientbutton"
                             onClick="go('RemoveCourseform.jsp');">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

E.2.12 RemoveCourseform.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
RemoveCourseform
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <%
    Operations op = new Operations();
    String coursecode = request.getParameter("coursecode");
    String query = "select * from course where coursecode=" + "\""
                            + coursecode + "\";";
    ResultSet rs = op.query(query);
    rs.next();
    String s = "\"";%>

  <span class="statictext">
```

```
    Course selected for removal is:<br>
    Course name: <span class="variabletext">
                          <%=rs.getString("coursename")%></span><br>
    Course code: <span class="variabletext">
                          <%=rs.getString("coursecode")%></span><br>
  </span>
  <%
    rs.close();
    op.closeTransaction(); %>

  <input type = "hidden" name = "coursecode" value =
                          <%=s +coursecode +s%>>
  <center>
    <input type="button" name="Update" value="Remove course"
           class="orientbutton"
                          onClick="go('Update.jsp');">
    <br><br>
  </center>
  <center>
    <input type="submit" name="back" value="<<"
                          class="orientbutton"
                          onClick="go('RemoveCourse.jsp');">
  </center>
</form>
</body>
</html>
```

E.2.13 RemoveTeacher.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
RemoveTeacher
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
<%
  Operations op = new Operations();
  ResultSet rs = op.query("select * from teacher");
  rs.next();
  String s = "\""; %>

  <center><h1>Remove teacher</h1></center>
  <br>
  <table width="50%" align="center">
    <tr align="center">
      <td align="left" width="75%" rowspan=2>
        Teacher signature:
        <select name="sig">
```

```
          <%
            do { %>
             <option value=<%=s+rs.getString("teachercode")+s %> >
                <%=rs.getString("teachercode")%>
              </option>
            <%
            }while(rs.next());%>
        </select>
      </td>
    <%
      rs.close();
      op.closeTransaction(); %>
      <td align="center" width=25% valign="middle" height=25>
        <input type="submit" name="Update" value="Remove teacher"
                            class="orientbutton"
                            onClick="go('RemoveTeacherform.jsp');">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

E.2.14 RemoveTeacherform.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
RemoveTeacherform
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <span class="statictext">
  <%
    Operations op = new Operations();
    String teachercode = request.getParameter("sig");
    String query = "SELECT * FROM teacher WHERE teachercode=" +
                            "\"" + teachercode + "\";";
    ResultSet rs = op.query(query);
    rs.next();
    String s = "\""; %>
    <table width="50%" align="center">
      <tr align="center">
        <td align="left" width="75%" rowspan=2>
          Teacher selected for removal is:<br>
          Name: <span class="variabletext"><%=rs.getString("forename")%></span>
        <%if(rs.getString("initial") != null) {%>
          <span class="variabletext"><%=rs.getString("initial")%></span>
        <%} %>
        <span class="variabletext"><%=rs.getString("surname")%></span><br>
```

```
        <%
          rs.close();
          op.closeTransaction(); %>
      </td>
      <input type = "hidden" name = "sig" value = <%= s + teachercode + s %>>

      <td align="left" width="25%">
        <input type="button" name="Update" value="Remove teacher"
               class="orientbutton" onClick="go('Update.jsp');">
        <br><br>
      </td>
    </tr>
    <tr align="center">
      <td align="left" width="25%">
          <input type="submit" name="back" value="<<" class="orientbutton"
                 onClick="go('RemoveTeacher.jsp');">
      </td>
    </tr>
  </table>
</span>
</form>
</body>
</html>
```

E.2.15 UpdateTeacher.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
UpdateTeacher
</title>
</head>
<body>
<script src="Scripts.jsp">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <center><h1>Update teacher</h1></center><br>
  <%
    Operations op = new Operations();
    ResultSet rs = op.query("SELECT * FROM teacher");
    rs.next();
    String s = "\""; %>

  <table width="50%" align="center">
    <tr>
      <td align="left" width=50%>
        Teacher signature:
        <br>
        <select name="sig">
          <%
            do { %>
              <option value= <%= s + rs.getString("teachercode") + s %> >
                <%=rs.getString("teachercode")%>
              </option>
```

```
            <%
              }while(rs.next()); %>
        </select>
        <%
          rs.close();
          op.closeTransaction(); %>
      </td>
      <td align="center" width="50%">
        <input type="button" name="Update" value="Update teacher"
               class="orientbutton" onClick="go('UpdateTeacherform.jsp');">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

E.2.16 UpdateTeacherform.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
UpdateTeacher
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
<%
  Operations op = new Operations();
  String s = "\"";
  String end = "\";";
  String teachercode = request.getParameter("sig");
  String query = "select * from teacher where teachercode=" + s +
                           teachercode + end;
  ResultSet rs = op.query(query);
  rs.next(); %>

  <center><h1>Update the fields needed</h1></center><br><br>
  <span class="infotext">
    <table align="center" >
    <tr>
      <td align="left" width="50%">
        Teachers signatur:
      </td>
      <td align="left" width="50%">
        <input size=5 name="sig" value=
                        <%=rs.getString("teachercode")%>>
      </td>
    </tr>
    <tr>
      <td align="left" width="50%">
        Forename:
```

```
  </td>
  <td align="left" width="50%">
    <input size=20 name="fname" value=
                        <%=rs.getString("forename")%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    Initial if any:
  </td>
  <td align="left" width="50%">
    <input size=5 name="initial" value=
      <%if(rs.getString("initial") != null){%>
          <%=rs.getString("initial")%>
      <%}
      else {%>
        <%=""%>
      <%}%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    Surname:
  </td>
  <td align="left" width="50%">
    <input size=20 name="sname" value=
                        <%=rs.getString("surname")%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    Email:
  </td>
  <td align="left" width="50%">
    <input size=30 name="email" value=
                        <%=rs.getString("email")%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    Room number:
  </td>
  <td align="left" width="50%">
    <input size=10 name="roomnr" value=
                        <%=rs.getString("roomnr")%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    Phone number:
  </td>
  <td align="left" width="50%">
    <input size=10 name="phonenr" value=
                        <%=rs.getString("phone")%>>
  </td>
</tr>
<tr>
  <td align="left" width="50%">
    <%
```

```
          ResultSet rs1 = op.query("select * from resourcetypes");
          rs1.next();
          ResultSet rs2;
                rs2 = op.query("select * from teacher_resourcetype where
                                        teachercode ="
                                        +s + rs.getString("teachercode")+ end);
          rs2.next(); %>
        Occupation:
      </td>
      <td align="left" width="50%">
        <select name="resourcename">
          <option value=<%=s+rs2.getString("resourcename")+s%> >
            <%= rs2.getString("resourcename")%>
          </option>
        <%
          do {
            if(!rs2.getString("resourcename").equalsIgnoreCase(
                          rs1.getString("resourcename")) ){ %>
             <option value=<%=s+rs1.getString("resourcename")+s%>>
                <%=rs1.getString("resourcename")%>
              </option>
            <%}
          }while(rs1.next());
        rs1.close();
        rs2.close(); %>
        </select>
      </td>
    </tr>
  </table>
  <input type="hidden" name="osig" value =
                          <%= s + rs.getString("teachercode") + s%>>
</span>
<%
  rs.close();
  op.closeTransaction(); %>
  <center>
    <input type="button" name="Update" value="Update teacher"
                          class="orientbutton"
                          onClick="go('Update.jsp');">
    <br><br>
  </center>
  <center>
    <input type="submit" name="back" value="<<"
                          class="orientbutton"
                          onClick="go('UpdateTeacher.jsp');">
  </center>
</form>
</body>
</html>
```

E.2.17 Update.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="kauscheduler.Operations"%>

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

```
    Update
</title>
</head>
<body>

<%
  Operations op = new Operations();

//Delimiters
  String comma = "\",\"";
  String comma2 ="\",";
  String parentes = "\");";
  String semicolon = "\";";
  String s = "\"";

//Common strings
  String first_query = "";     //First query
  String second_query = "";    //Second query
  String sess = "";            //session name currently not in use
  String id = "";              //id holder

  String insert = "insert into ";
  String select = "select * from ";
  String delet = "delete from ";
  String update = "UPDATE ";
  String values = " values(";
  String where = " where ";
  String set = "set ";
  String tables[] = {"teacher ", "teacher_resourcetype ",
                     "course ", "course_activity ",
                     "resourcetypes ", "activitytypes "};
  String attribut[] = {"teachercode=", "coursecode=",
                       "resourcename=", "activityname="};

  if(request.getParameter("comm").equals("Add teacher")) {
    String teachercode = request.getParameter("sig");
    String forename = request.getParameter("fname");
    String surname = request.getParameter("sname");
    String initial = request.getParameter("initial");
    String email = request.getParameter("email");
    String roomnr = request.getParameter("roomnr");
    String phone = request.getParameter("phonenr");
    String resourcename = request.getParameter("resourcename");
    String start = insert + tables[0] + values;
    ResultSet rt = op.query(select + tables[4] + where +
                            attribut[2] +s+ resourcename + semicolon);
    rt.next();
    id = op.runOp("ResourceHome.createResource",new Object[]
         {sess, teachercode, surname,
              rt.getString("resourcetypeid")}).toString();
    first_query = start + s + teachercode + comma + forename +
                            comma + initial + comma +surname
                            + comma + roomnr + comma + phone + comma + email
                            + comma + id + parentes;
    second_query = insert + tables[1] + values + s + teachercode +
                            comma + resourcename + parentes;
    rt.close();
  }
```

```
else if(request.getParameter("comm").equals("Remove teacher")) {
  String teachercode = request.getParameter("sig");
  String start = delet + tables[0] + where + attribut[0];
  ResultSet rt = op.query(select + tables[0] + where +
                          attribut[0] + s + teachercode + semicolon);
  rt.next();
  op.runOp("ResourceHome.removeResource",new Object[]
          {sess, rt.getString("resourceid") });
  first_query = start + s + teachercode + semicolon;
  second_query = delet + tables[1] + where + attribut[0] + s +
                          teachercode + semicolon;
  rt.close();
}

else if(request.getParameter("comm").equals("Add course")) {
  String coursecode = request.getParameter("coursecode");
  String coursename = request.getParameter("cname");
  ResultSet rt = op.query(select + tables[5] + where +
                          attribut[3] + s + "course" + semicolon);
  rt.next();
  id = op.runOp("ActivityHome.createActivity",new Object[]
      {sess, coursecode, coursename,
           rt.getString("activitytypeid") }).toString();
  String start = insert + tables[2] + values;
  first_query = start + s + coursename + comma + coursecode +
                          comma + id + parentes;
  rt.close();
}

else if(request.getParameter("comm").equals("Remove course")) {
  String coursecode = request.getParameter("coursecode");
  String start = delet + tables[2] + where + attribut[1];
  ResultSet rt = op.query(select + tables[2] + where +
                          attribut[1] + s + coursecode + semicolon);
  rt.next();
  op.runOp("ActivityHome.removeActivity",new Object[]
          {sess, rt.getString("activityid") });
  first_query = start + s + coursecode + semicolon;
  rt.close();
}

else if(request.getParameter("comm").equals("Update teacher")) {
  String teachercode = request.getParameter("sig");
  String forename = request.getParameter("fname");
  String surname = request.getParameter("sname");
  String initial = request.getParameter("initial");
  String email = request.getParameter("email");
  String roomnr = request.getParameter("roomnr");
  String phone = request.getParameter("phonenr");
  String oteachercode = request.getParameter("osig");
  String resourcename = request.getParameter("resourcename");
  String start = update + tables[0] + set + attribut[0];
  String sentence[] = {"forename=\"", "surname=\"",
                                       "initial=\"", "email=\"",
                                       "roomnr=\"", "phone=\"",
                                       "resourceid=\""};
  String ends = where + attribut[0];

  ResultSet rt = op.query(select + tables[0] + where +
```

```
                          attribut[0] + s + oteachercode + semicolon);
    rt.next();
    first_query = start + s + teachercode + comma2 + sentence[0] +
                          forename + comma2 + sentence[1] + surname +
                          comma2 + sentence[2] + initial + comma2 +
                          sentence[3] + email + comma2 + sentence[4] +
                          roomnr + comma2 + sentence[5] + phone + comma2 +
                          sentence[6] + rt.getString("resourceid") + s +
                          ends + s + oteachercode + semicolon;
    second_query = update + tables[1] + set + attribut[0] + s +
                          teachercode + comma2 + attribut[2] + s +
                          resourcename + s + ends + s + oteachercode +
                          semicolon;
    rt.close();
  }

  op.query(first_query);
  if(second_query.length() != 0) {
    System.out.println(second_query);
    op.query(second_query);
  }
  op.closeTransaction(); %>
<script>
  window.location = "Mainpage.html";
</script>
</body>
</html>
```

E.2.18 ShowTables.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="kauscheduler.Operations"%>
<%@page import="java.sql.ResultSetMetaData"%>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Show Tables
</title>
</head>
<body>

<% Operations op = new Operations();
   String q = "";
   if(request.getParameter("Show").equalsIgnoreCase(
                          "Show teachers")) {
     q = "Select * from teacher";
   }
   else {
     q = "Select * from course";
   }

   ResultSet t = op.query(q);
   ResultSetMetaData rm = t.getMetaData();
   int j = rm.getColumnCount();
   int i = 1;
   int size = 100 / j;
   String wd = "\"" + size + "%\""; %>
```

```
<table align="left" border="1">
  <tr align="left">
<% while(i < j) { %>
    <td align="center" width= <%=wd%> >
      <%=rm.getColumnLabel(i++)%>
    </td>
  <%}
  i = 1; %>
  </tr>
<% while(t.next()) { %>
  <tr align="left">
  <% while(i < j) { %>
      <td width= <%=wd%> >
        <%= t.getString(i++) %>
      </td>
  <%}
  i = 1;%>
  </tr>
<%}
  t.close();
  op.closeTransaction(); %>
</table>
</body>
</html>
```

E.2.19 CourseDescription.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Course Descriptions
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <center><h1>Course Description</h1></center>
  <%
    Operations op = new Operations();
    ResultSet rs = op.query("select * from course");
    rs.next();
    String s = "\""; %>

  <table width="50%" align="center">
    <tr align="center">
      <td align="left" width="75%">
        Course code: <br>
        <select name="coursecode">
        <%
          do { %>
            <option value=<%=s+rs.getString("coursecode")+s%> >
              <%= rs.getString("coursecode")%>
```

```
            </option>
            <%
          }while(rs.next()); %>
        </select>
        <%
          rs.close();
          op.closeTransaction(); %>
      </td>
      <td align="left" width="75%">
        <input type="button" name="Update" value="Define course"
                          class="orientbutton"
                          onClick="go('CourseDescriptionform.jsp');">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

E.2.20 CourseDescriptionform.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
   <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
CourseDescriptionform
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <%
    Operations op = new Operations();
    String coursecode = request.getParameter("coursecode");
    String query = "select * from course where coursecode=" + "\""
                            + coursecode + "\";";
    ResultSet rs = op.query(query);
    rs.next();
    String s = "\"";%>

  <h1>Course to defined</h1><br>
  <span class="infotext">
    Course name: <span class="variabletext">
                          <%=rs.getString("coursename")%></span><br>
    Course code: <span class="variabletext">
                          <%=rs.getString("coursecode")%></span><br><br>

  <table align="center" width="75%" height="50%">
    <tr align="left">
      <td align="left" width="25%">
        Number of lectures
      </td>
      <td align="left" width="25%">
        <input type="text" name="nrlectures" value="1"
```

```
                          class="digitinput">
      </td>
    </tr>
    <tr align="left">
      <td align="left" width="25%">
        Number of laboration sessions
      </td>
      <td align="left" width="25%">
        <input type="text" name="nrlabs" value=”2”
                          class="digitinput”>
      </td>
    </tr>
    <tr align="left">
      <td align="left" width="25%">
        Number of lession sessions
      </td>
      <td align="left" width="25%">
        <input type="text" name="nrlessions" value=”0”
                          class="digitinput">
      </td>
    </tr>
    <tr align="left">
      <td align="left" width="25%">
        Number of exams
      </td>
      <td align="left" width="25%">
        <input type="text" name="nrexams" value="1"
                           class="digitinput">
      </td>
    </tr>
  </table>
  <br><br><br>
  </span>
  <%
    rs.close();
    op.closeTransaction();%>

  <input type = "hidden" name = "coursecode" value =
                          <%= s + coursecode + s %> >
  <center>
    <input type="button" name="Update" value="Define course"
                          class="orientbutton"
                          onClick="go('CoursePlan.jsp');">
    <br><br>
    <input type="button" name="back" value="<<"
                          class="orientbutton"
                          onClick="go('CourseDescription.jsp');">
  </center>
</form>
</body>
</html>
```

E.2.21 Courseplan.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
```

```
   <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
CoursePlan
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm" value="">
  <%
    Operations op = new Operations();
    String coursecode = request.getParameter("coursecode");
    int nrlectures = op.StringToInt(request.getParameter(
                           "nrlectures"));
    int nrlabs = op.StringToInt(request.getParameter(
                           "nrlabs"));
    int nrlessions = op.StringToInt(request.getParameter(
                           "nrlessions"));
    int nrexams = op.StringToInt(request.getParameter(
                           "nrexams"));
    String s = "\"";
    op.closeTransaction();%>
  <%!
    public int big(int t) {
      return (t>6 ? 6 : t);
    } %>

  <input type="hidden" name="nrlabs" value=<%= nrlabs %>>
  <input type="hidden" name="nrlectures" value=
            <%= nrlectures %>>
  <input type="hidden" name="coursecode" value=
            <%= s + coursecode + s%>>
  <input type="hidden" name="nrlessions" value=
            <%= nrlessions %>>
  <input type="hidden" name="nrexams" value=
            <%= nrexams %>>

  <table width="80%" align="center">
    <tr>
      <td align="right" width=40%>
        <select name="lectures" size=<%=big(nrlectures)%>>
          <%
            for(int i = 1; i <= nrlectures; i++) { %>
              <option value=<%=i%>>Lecture<%=i%></option>
            <%}%>
        </select>
      </td>
      <td align="left" width="20%">
        <input type="button" value=">>"
                           class="orientbutton"
                           onClick="sendToFrom(document.myForm.Update,
                           document.myForm.lectures);">

      </td>
      <td align="right" width="40%" rowspan=4>
        <textarea rows=0 name="Update" DISABLED cols=24>
        </textarea>
      </td>
```

```
  </tr>
  <tr>
    <%
      if(nrlabs != 0) { %>
        <td align="right" width="40%">
          <select name="labs" size=<%=big(nrlabs)%>>
            <%
              for(int i = 1; i <= nrlabs; i++) { %>
                <option value=<%=i%>>Lab<%=i%></option>
              <%}%>
          </select>
        </td>
        <td align="left" width="20%">
          <input type="button" value=">>"
                         class="orientbutton"
                         onClick="sendToFrom(document.myForm.Update,
                         document.myForm.labs);">
        </td>
      <%}%>
  </tr>
  <tr>
    <%
      if(nrlessions != 0){%>
        <td align="right" width="40%">
          <select name="lessions" size=<%=big(nrlessions)%>>
            <%
              for(int i = 1; i <= nrlessions; i++) { %>
                <option value=<%=i%>>Lession<%=i%></option>
             <%}%>
          </select>
        </td>
        <td align="left" width="20%">
          <input type="button" value=">>"
                         class="orientbutton"
                         onClick="sendToFrom(document.myForm.Update,
                         document.myForm.lessions);">
        </td>
      <%}%>
    </tr>
  <tr>
    <%
      if(nrexams != 0){%>
        <td align="right" width="40%">
          <select name="exams" size=<%=big(nrexams)%>>
            <%
              for(int i = 1; i <= nrexams; i++) { %>
                <option value=<%=i%>>Exam<%=i%></option>
             <%}%>
          </select>
        </td>
        <td align="left" width="20%">
          <input type="button" value=">>"
                         class="orientbutton"
                         onClick="sendToFrom(document.myForm.Update,
                         document.myForm.exams);">
        </td>
      <%}%>
    </tr>
</table>
```

```
  <center>
    <input type="button" value="submit"
                           class="orientbutton"
                           onClick="go('SetUpCourse.jsp');">
  </center>
</form>
</body>
</html>
```

E.2.22 SetUpCourse.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>
<%@page import="java.util.StringTokenizer"%>

<html>
<head>
<title>
SetUpCourse
</title>
</head>
<body>
<%
  Operations op = new Operations();
  op.serialize();
  String sequence = request.getParameter("comm");
  String labsessions = request.getParameter("nrlabs");
  String lessions = request.getParameter("nrlessions");
  String lectures = request.getParameter("nrlectures");
  String coursecode = request.getParameter("coursecode");
  String comma = "\",\"";
  String rparent  = "\");";
  StringTokenizer st = new StringTokenizer(sequence,"\n");
  ResultSet r = op.query("select * from course where
                                          coursecode=\"" + coursecode + "\";");
  r.next();
  String parentid = r.getString("activityid");
  ResultSet at;
  String temp = "";
  String typeid = "";

  op.query("insert into course_activity values(\"" + coursecode +
           comma + "lecture" + comma + lectures + rparent);
  op.query("insert into course_activity values(\"" + coursecode +
           comma + "labsession" + comma + labsessions + rparent);
  op.query("insert into course_activity values(\"" + coursecode +
           comma + "lession" + comma + lessions + rparent);
  ResultSet rules = op.query("select * from rules where
           rulename=\"activityOrder\"");
  rules.next();

  String params;
  while(st.hasMoreTokens()) {
    temp = st.nextToken();
    if(temp.substring(0,3).equalsIgnoreCase("lec")) {
      at = op.query("select * from activitytypes where
                           activityname=\"lecture\"");
      at.next();
      typeid = at.getString("activitytypeid");
```

```
    }
    else if(temp.substring(0,3).equalsIgnoreCase("lab")) {
      at = op.query("select * from activitytypes where
                          activityname=\"labsession\"");
      at.next();
      typeid = at.getString("activitytypeid");
    }
    else if(temp.substring(0,3).equalsIgnoreCase("les")) {
      at = op.query("select * from activitytypes where
                          activityname=\"lession\"");
      at.next();
      typeid = at.getString("activitytypeid");
    }
    else if(temp.substring(0,3).equalsIgnoreCase("exa")) {
      at = op.query("select * from activitytypes where
                          activityname=\"exam\"");
      at.next();
      typeid = at.getString("activitytypeid");
   }
    params = parentid;
    parentid = op.addActivityChild(parentid, typeid, temp);
    op.addRuleBinding(parentid, rules.getString("ruleid"), new
            Object[] {params});
  }
  op.closeTransaction();
  rules.close();
  r.close();%>
<script>
  window.location = "Mainpage.html";
</script>
</body>
</html>
```

E.2.23 Operations.java (för registergränssnittet)

```
package kauscheduler;

import marquee.xmlrpc.*;
import java.sql.ResultSet;
import marquee.util.database.*;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.lang.Math;
import java.util.Collection;
import java.util.Hashtable;
import java.util.Vector;
import marquee.xmlrpc.serializers.VectorSerializer;
import marquee.xmlrpc.serializers.HashtableSerializer;
import marquee.xmlrpc.serializers.CollectionSerializer;
import marquee.xmlrpc.serializers.MapSerializer;
import marquee.xmlrpc.serializers.ObjectArraySerializer;
import java.util.Iterator;

public class Operations {
  private XmlRpcClient client =
                          new XmlRpcClient("127.0.0.1",9999,"");
  private Transaction t = new Transaction( "kauscheduler" );
  private final String session = "";
```

```
  public Object runOp(String operation, Object[] param)
           throws Exception {
                        return client.invoke(operation,param);
  }

  public ResultSet query(String q)
           throws Exception {
                        return t.doQuery(q);
  }

  public void closeTransaction()
           throws Exception {
                        t.commit();
                        t.getConnection().close();
  }

  public String addActivityChild(String parentid,
                                              String typeid,
                                              String name)
           throws Exception {
    String childid = client.invoke("ActivityHome.createActivity",
           new Object[]
        {session, name, name, typeid}).toString();
    client.invoke("Activity.addChild", new Object[]
        {session, parentid, childid});
    return childid;
  }

  public void addRuleBinding(String actid,
                                          String ruleid,
                                          Object[] params)
           throws Exception {
    client.invoke("Activity.addRuleBinding", new Object[]
            {session, actid, ruleid, params});
  }

  public Collection getChildren(String parentid, boolean all)
           throws Exception {
return (Collection)client.invoke("ActivityHome.getActivities",
           new Object[]
        {session, (client.invoke("Activity.getChildren",
           new Object[]
        {session, parentid, new Boolean(all)}))});
  }

  public int CharToInt(char c)
           throws Exception {
                        return (c - '0');
  }

  public int StringToInt(String s)
    throws Exception {
           int i = 0;
           int ans = 0;
           while(i < s.length()) {
             ans+=(s.charAt(i)-'0')*Math.pow(10,(s.length()-(i+1)));
      i++;
    }
    return ans;
```

```
}

public int maxLength(ResultSet c, String col0, String col2)
 throws Exception {
  int max = 0, t = 0;
  while(c.next()) {
   t = c.getString(col0).length()+c.getString(col2).length()+1;
   if(max < t) {
     max = t;
   }
  }
  return max;
}

public void serialize() {
  XmlRpcSerializer.registerCustomSerializer(
          new MapSerializer() );
  XmlRpcSerializer.registerCustomSerializer(
          new CollectionSerializer() );
  XmlRpcSerializer.registerCustomSerializer(
          new VectorSerializer() );
  XmlRpcSerializer.registerCustomSerializer(
          new HashtableSerializer() );
  XmlRpcSerializer.registerCustomSerializer(
          new ObjectArraySerializer() );
}

public Vector getNameAndID(Collection v)
 throws Exception {
  Vector r = new Vector();
  Iterator iter = v.iterator();
  String description;
  String id;
  while(iter.hasNext()) {
    String t = iter.next().toString();
    t = t.substring(t.indexOf(
          "rpengine/business-object/description=") +
          "rpengine/business-object/description=".length());
    description = t.substring(0,t.indexOf(","));
    t = t.substring(t.indexOf("rpengine/business-object/id=") +
          "rpengine/business-object/id=".length());
    id = t.substring(0,16);
    r.addElement(id);
    r.addElement(description);
  }
  return r;
}

public Vector sortVector(Vector v) {
  Object temp = new Object();
  boolean sorted = true;

  do {
    sorted = true;
    for(int i = 1; i < v.size() - 1; i += 2) {
      for(int j = i + 2; j < v.size(); j += 2) {
        if(v.elementAt(i).toString().compareTo(
                          v.elementAt(j).toString()) > 0) {
          temp = v.elementAt(i);
```

```
            v.setElementAt(v.elementAt(j), i);
            v.setElementAt(temp, j);
            temp = v.elementAt(i - 1);
            v.setElementAt(v.elementAt(j - 1), i - 1);
            v.setElementAt(temp, j - 1);
            sorted = false;
          }
        }
      }
    }while(!sorted);
    return v;
  }

  public void connect(String blob)
   throws Exception {
    String teacher_id = "";
    String occasion_id = "";
    ResultSet rs;
    int i =1;
    while(blob.length() > 0) {
      blob = blob.substring(blob.indexOf("-") + 1);
      rs = query("select resourceid from teacher where
            forename=\"" + blob.substring(0, blob.indexOf(" ")) + "\"
            and surname=\"" + blob.substring(blob.indexOf(" ") + 1,
            blob.indexOf("-")) + "\";");
      rs.next();
      teacher_id = rs.getString("resourceid");
      blob = blob.substring(blob.indexOf("-") + 1);
      occasion_id = blob.substring(0, blob.indexOf("\n"));
      blob = blob.substring(blob.indexOf("\n") + 1);

      runOp("Activity.addResource", new Object[]
            {session, occasion_id, teacher_id});
    }
  }
  public Vector getRes(String blob)
  throws Exception {
    Vector id = new Vector();
    String fname, ename;
    ResultSet rs;
    blob = blob.substring(blob.indexOf("-") + 1);
    while(blob.length() > 20) {
      System.out.println("in while blob=" + blob);
      fname = blob.substring(0, blob.indexOf(" "));
      ename = blob.substring(blob.indexOf(" ")
                  + 1,blob.indexOf("-"));
      rs = query("select resourceid from teacher where
            forename=\"" + fname + "\" and surname=\"" + ename +
            "\";");
      rs.next();
      id.addElement(rs.getString("resourceid"));
      blob = blob.substring(blob.indexOf("-") + 1);
      blob = blob.substring(blob.indexOf("-") + 1);
    }
    return id;
  }
}
```

E.3 Genereringsgränssnittet

Nedan följer källkoden som genereringsgränssnittet är uppbyggt runt.

E.3.1 Startpage.jsp

```
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>
<%@page import="java.util.Date" %>
<%@page import="java.text.SimpleDateFormat" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Startpage for generation of schedule
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post" onSubmit="check();">
  <input type="hidden" name="comm" value="">
  <center><h1>Generation page</h1></center>
  <%
    Operations op = new Operations();
    ResultSet rs = op.query("select * from course");
    rs.next();
    ResultSet rt = op.query("select * from teacher");
    rt.next();
    SimpleDateFormat formatter;
    formatter = new SimpleDateFormat ("yyyy-MM-dd");
    Date currentTime = new Date();
    String dateString = formatter.format(currentTime);
    String s = "\"";
    int nrOfChoices = 31;%>

  <table align="center">
    <tr><td align="left" width="50%">
        Course id:
      </td><td align="left" width="50%">
        <select name="cid">
          <%
            do { %>
              <option value=<%=s+ rs.getString("coursecode")+s%> >
                <%=rs.getString("coursecode")%> </option>
              <%
            }while(rs.next()); %>
        </select>
      </td></tr>
    <tr><td align="left" width="50%">
        Number of students:
      </td><td align="left">
        <input name="nrstud" value = 0>
      </td></tr>
    <tr><td align="left">
```

```
            Startdate<br>(YYYY-MM-DD):
          </td><td align="left" width="50%">
            <input name="startdate" value = <%= dateString %>>
          </td></tr>
        <tr><td align="left">
            Exam date<br>(YYYY-MM-DD):
          </td><td align="left">
            <input name="examdate" value = <%= dateString %>>
          </td></tr>
        <tr><td align="left">
            Resit<br>(YYYY-MM-DD):
          </td><td align="left">
            <input names="resitdate" value= <%= dateString %>>
          </td></tr>
        <tr><td align="left">
            Course speed:
          </td><td align="left">
            <select name="spd">
              <option value="Df">Daily full</option>
              <option value="Dh">Daily half</option>
              <option value="Dq">Daily quarter</option>
              <option value="Ef">Evening full</option>
              <option value="Eh">Evening half</option>
              <option value="Eq">Evening quarter</option>
            </select>
          </td></tr>
        <tr><td align="left">
            Weekdays:
          </td><td align="left">
            <select name="days">
              <option value="MF">Mon-Fri</option>
              <option value="MWE">Mon-Wen (even weeks)</option>
              <option value="MWO">Mon-Wen (odd weeks)</option>
              <option value="WFE">Wen-Fri (even weeks)</option>
              <option value="WFO">Wen-Fri (odd weeks)</option>
            </select>
          </td></tr>
      </table>
      <br><br>
      <%
        rt.close();
        rs.close();
        op.closeTransaction(); %>

      <center>
        <input type="button" name="Update" value=">>"
              class="orientbutton" OnClick="go('TeacherChoice.jsp');">
        <br><br>
        <input type="reset" name="Reset" value="Reset"
                  class="orientbutton">
      </center>
    </form>
    </body>
</html>
```

E.3.2 Teacherchoice.jsp

```
<%@page import="java.util.Calendar" %>
<%@page import="java.util.Date" %>
<%@page import="java.text.DateFormat" %>
<%@page import="java.sql.ResultSet" %>
<%@page import="kauscheduler.Operations" %>

<html>
<head>
    <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
TeacherChoice
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
  <input type="hidden" name="comm">
  <%
    Operations op = new Operations();
    ResultSet rt = op.query("SELECT * FROM teacher");
    rt.next();
    String s = "\""; %>
  <table width="50%" align="center">
    <tr><td align="center" valign="top" width="40%">
        <select name="teacher" size=5>
            <%
              do { %>
                <option value= <%=rt.getString("teachercode")%> >
                  <%=rt.getString("forename")+" "
                              + rt.getString("surname")%>
                </option>
              <%
              }while(rt.next());
              rt.beforeFirst();%>
          </select>
      </td><td align="center" valign="bottom" width="10%">
        <input type="button" value=">>"
                            class="orientbutton"
                            onClick="sendToFrom(document.myForm.Update, document.myForm.teacher);">
      </td><td align ="center" valign="top" width="40%">
        <textarea rows=0 name="Update" DISABLED cols=
                             <%=op.maxLength(rt,"forename","surname") + 5%>>
        </textarea>
      </td></tr>
  </table>
  <center>
    <input type="button" value="Submit"
            class="orientbutton" onClick="go('Connect.jsp');">
  </center>
  <%
    rt.close();
    op.closeTransaction(); %>
  <input type="hidden" name="cid" value=
            <%= s + request.getParameter("cid") + s%>>
  <input type="hidden" name="sdate" value=
```

```
            <%=request.getParameter("startdate")%>>
     <input type="hidden" name="edate" value=
            <%=request.getParameter("examdate")%>>
     <input type="hidden" name="nrstud" value=
            <%=request.getParameter("nrstud")%>>
     <input type="hidden" name="days" value=
            <%=request.getParameter("days")%>>
   </form>
   </body>
</html>
```

E.3.3 Connect.jsp

```
   <%@page import="java.sql.ResultSet"%>
   <%@page import="kauscheduler.Operations" %>
   <%@page import="java.util.StringTokenizer"%>
   <%@page import="java.util.Collection" %>
   <%@page import="java.util.Iterator" %>

   <html>
   <head>
      <link rel="stylesheet" href="kau_style.css" type="text/css">
   <title>
   Connect
   </title>
   </head>
   <body>
   <script src="Scripts.js">
   </script>
   <form name="myForm" method="post">
   <%
     String s = "\"";
     Operations op = new Operations();
     op.serialize();
     String teachers = request.getParameter("comm");
     String coursecode = request.getParameter("cid");
     ResultSet rt = op.query("select * from course_activity where
                              coursecode = \"" + coursecode + "\";");
     ResultSet rs = op.query("select * from course where coursecode =
                              \"" + coursecode + "\";");
     rs.next();
     rt.next();

     if(rt.getRow() != 0) {
       String cid = rs.getString("activityid");
       StringTokenizer st = new StringTokenizer(teachers, "\n");
       String cur = "";
       Collection children = op.getChildren(cid, true);
       Vector vec = op.sortVector(op.getNameAndID(children));
       int i = 0; %>

       <table align="center" width="100%">
         <tr><td align="left" width="25%">
             <select name="teachers" size=3 multiple>
               <%
                 while(st.hasMoreTokens()) {
                   cur = st.nextToken();%>
                   <option value=<%=cur%>><%=cur%></option>
                 <%}%>
```

```
            </select>
          </td><td align="right" width="25%">
            <select name="education" size=3 multiple>
              <%
                while(i < vec.size()) {%>
                  <option value=<%=vec.elementAt(i++).toString()%> >
                              <%=vec.elementAt(i++).toString()%></option>
                <%}%>
            </select>
          </td><td align="center" width="25%">
            <input type="button" value=">>" class="orientbutton"
                              onClick="moveToFrom(document.myForm.Update,        document.myForm.education,
                              document.myForm.teachers);">
          </td><td align="right" width="25%">
            <textarea rows=0 name="Update" DISABLED cols=30
                              wrap="off">
            </textarea>
          </td></tr>
        <tr><td align="center" width="100%" colspan=3>
            <input type="button" value="submit"
                   class="orientbutton" onclick="go('Hours.jsp');">
          </td></tr>
        </table>
        <%
        }
        else { %>
          <center>
            <h1>The choosen course isn't defined!</h1>
            <input type="button" value="Define course"
                              class="orientbutton"
                              onClick="go('CourseDescription.jsp');"><br>
            <input type="button" value="<<"
                              class="orientbutton" onClick="go('Startpage.jsp');">
          </center>
        <%
        }
        rt.close();
        rs.close();
        op.closeTransaction(); %>

      <input type="hidden" name="comm">
      <input type="hidden" name="cid" value=
                <%= s + request.getParameter("cid") + s%>>
      <input type="hidden" name="sdate" value=
                <%=request.getParameter("sdate")%>>
      <input type="hidden" name="edate" value=
                <%=request.getParameter("edate")%>>
      <input type="hidden" name="nrstud" value=
                <%=request.getParameter("nrstud")%>>
      <input type="hidden" name="days" value=
                <%=request.getParameter("days")%>>
    </form>
    </body>
</html>
```

E.3.4 Hours.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="kauscheduler.Operations" %>
<%@page import="java.util.Iterator"%>
<%@page import="java.util.Collection"%>

<html>
<head>
   <link rel="stylesheet" href="kau_style.css" type="text/css">
<title>
Hours
</title>
</head>
<body>
<script src="Scripts.js">
</script>
<form name="myForm" method="post">
<%
  Operations op = new Operations();
  op.serialize();
  String s = "\"";
  op.connect(request.getParameter("comm"));
  String coursecode = request.getParameter("cid");
  ResultSet rt = op.query("select * from course_activity where
                                          coursecode = \"" + coursecode + "\";");
  ResultSet rs = op.query("select * from course where coursecode =
                                          \"" + coursecode + "\";");
  rs.next();
  rt.next();

  String cid = rs.getString("activityid");
  Vector vec;
  vec = op.sortVector(op.getNameAndID(op.getChildren(cid, true)));
  int i = 0, j = 5; %>

  <table align="center" width="100%">
    <tr><td align="right" width="25%">
        <select name="duration" size=3>
          <option value="1">1</option>
          <option value="2">2</option>
          <option value="3">3</option>
          <option value="4">4</option>
          <option value="5">5</option>
        </select>
      </td><td align="right" width="25%">
        <select name="education" size=3 multiple>
          <%
            while(i < vec.size() - 1) {%>
              <option value=<%=vec.elementAt(i++).toString()%> >
                          <%=vec.elementAt(i++).toString()%></option>
            <%}%>
         </select>
      </td><td align="center" width="25%">
        <input type="button" value=">>" class="orientbutton"
                          onClick="moveToFromHours(document.myForm.Update,   document.myForm.education,
                          document.myForm.duration);">
      </td><td align="right" width="25%">
```

```
          <textarea rows=0 name="Update" DISABLED cols=30
                              wrap="off">
          </textarea>
        </td></tr>
      <tr><td align="center" width="100%" colspan=3>
          <input type="button" value="submit"
                 class="orientbutton" onclick="go('Generate.jsp');">
        </td></tr>
    </table>
    <%
      op.closeTransaction();
      rs.close();
      rt.close(); %>

    <input type="hidden" name="connection" value=
              <%= s + request.getParameter("comm") + s%>>
    <input type="hidden" name="cid" value=
              <%=s + request.getParameter("cid") + s%>>
    <input type="hidden" name="sdate" value=
              <%=request.getParameter("sdate")%>>
    <input type="hidden" name="edate" value=
              <%=request.getParameter("edate")%>>
    <input type="hidden" name="nrstud" value=
              <%=request.getParameter("nrstud")%>>
    <input type="hidden" name="days" value=
              <%=request.getParameter("days")%>>
    <input type="hidden" name="comm">
  </form>
  </body>
</html>
```

E.3.5 Generate.jsp

```
  <%@page import="java.sql.ResultSet"%>
  <%@page import="kauscheduler.Operations" %>
  <%@page import="java.text.SimpleDateFormat" %>
  <%@page import="java.lang.Integer" %>
  <%@page import="java.util.Vector" %>
  <%@page import="java.util.Hashtable" %>

  <html>
  <head>
  <title>
  Generate
  </title>
  </head>
    <input type="hidden" name="Update" value="dummy_value">
    <input type="hidden" name="sdate" value=
              <%=request.getParameter("sdate")%>>
    <input type="hidden" name="edate" value=
              <%=request.getParameter("edate")%>>
    <input type="hidden" name="nrstud" value=
              <%=request.getParameter("nrstud")%>>
    <input type="hidden" name="days" value=
              <%=request.getParameter("days")%>>
    <input type="hidden" name="cid" value=
              <%=request.getParameter("cid")%>>
  <%
    Operations op = new Operations();
```

```
      op.serialize();
      int students = op.StringToInt(request.getParameter("nrstud"));
      String cid = request.getParameter("cid");
      String time = request.getParameter("comm");
      String sdate = request.getParameter("sdate");
      String edate = request.getParameter("edate");
      SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
      ResultSet rs = op.query("select * from course where
                                              coursecode=\"" + cid + "\";");
      rs.next();

      Vector vec = new Vector();
      Vector v;
      v = op.getNameAndID(op.getChildren(rs.getString("activityid"),
                               true));

      int occ = 1;
      for(int i = 0; i < v.size() - 1; i++) {
        occ = 1;
        Hashtable h = new Hashtable();
        h.put("activity", v.elementAt(i));
        h.put("duration", new Integer(2));
        h.put("durationUnit", "HOURS");

        if(v.elementAt(i+1).toString().substring(0,3).
           equalsIgnoreCase("Lab") ||
           v.elementAt(i+1).toString().substring(0,3).
           equalsIgnoreCase("Les"))
             occ = 1 + students / 40;
        h.put("occurrences", new Integer(occ));
        vec.addElement(h);
        i++;
      }

      op.runOp("Scheduler.addOccurencces", new Object[] {
                op.runOp("Scheduler.suggestStartTimes", new Object[]
                {"", vec, formatter.parse(sdate),
                 formatter.parse(edate), new Integer(60), "MINUTES"}
                                  )
                  }); %>
    <script>
      window.location = "Startpage.jsp";
    </script>
    </body>
</html>
```