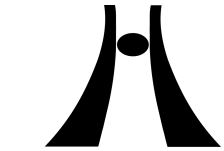
Beteckning:_____



**HÖGSKOLAN I GÄVLE**

**Akademin för teknik och miljö**

# Utveckling av tjänsteplaneringssystem

*Rickard Aurosell*
*Daniel Linde*
*juni 2012*

Examensarbete, 15 hp, B
Datavetenskap

**Dataingenjörsprogrammet**
**Examinator: Ann-Sofie Östberg**
**Handledare: Bengt Östberg**

# Utveckling av tjänsteplaneringssystem

av

Rickard Aurosell
Daniel Linde

Akademin för teknik och miljö
Högskolan i Gävle

801 76 Gävle, Sverige

Email:
*ntn08ral@student.hig.se*
*ndi09dle@student.hig.se*

**Abstrakt**

Högskolan i Gävle har inget gemensamt system för tjänsteplanering. Processen är ineffektivt och innefattar ifyllandet av Excel-ark. Detta arbete omfattar designen och skapandet av ett tjänsteplaneringssystem åt en tjänsteplanerare vid HiG. Det som efterfrågades var ett system som är smidigare och ger möjlighet att överblicka det som planerats. Systemet behövde bara stödja en användare. Ett system skapades som består av en databas och en användarapplikation som kommunicerar med databasen. Som databas användes MySQL och applikationen skrevs i Java. Befintliga datakällor på HiG med kurser och personal används för att minimera mängden manuell inmatning. Systemet är en fungerande lösning klart för användning som överträffade beställarens förhoppningar.

# Innehåll

# 1 Inledning

## 1.1 Problem

Högskolan i Gävle har för närvande ett ineffektivt system för tjänsteplanering. Avdelningarna har själva ansvaret för hur planeringen utformas. En lösning med ett excel-ark per personal och år används av avdelningen för industriell utveckling, IT och samhällsbyggnad vid akademin för teknik och miljö. Med den lösningen är det svårt att få en  översikt av personalens inplanerade dagar. Tjänsteplaneraren måste själv katalogisera alla filer. För ett excel-ark från det nuvarande systemet med mockup data se bilaga 1.

## 1.2 Syfte

Skapa ett tjänsteplaneringssystem åt tjänsteplaneraren för IT-avdelningen vid Akademin för teknik och miljö på Högskolan i Gävle. Systemet ska använda befintliga datakällor. Rapporter över utförd och planerad tid ska kunna skrivas ut. Översikt både i form av tabeller och diagram ska vara tillgängligt. Även personalens kompetenser ska kunna lagras och visas.

Systemet designas för endast en användare, men är tänkt att kunna vidareutvecklas till eller ge inspiration för ett framtida gemensamt system för tjänsteplanerarna vid Högskolan i Gävle.

## 1.3 Målsättning

Att utveckla ett användarvänligt system för tjänsteplanering som uppfyller de krav som specificerades i 1.2.

# 2 Bakgrund

Vid början av utvecklingen bestämdes en lösningen med en applikation som kommunicerar mot en databas. Handledaren hade som krav att en databas skulle användas.

## 2.1 Tjänsteplanering

Tjänsteplanering är fördelandet av arbetsuppgifter på anställda. I en organisation där anställda har flera och varierande arbetsuppgifter är tjänsteplanering en nödvändighet för att styra hur arbetstiden spenderas. Mängden tid som kommer att läggas på varje uppgift uppskattas i förväg. Målet är att fördela uppgifterna mellan de anställda så att alla anställda är fullbelagda, alla uppgifter blir utförda och alla de anställda har tid för kompetensutveckling. Efter tidsrapporteringen utvärderas arbetet under perioden genom att återkoppla och jämföra den tid som spenderats på varje uppgift mot den tid som var planerad.

# 3 Teori

## 3.1 Relationsdatabas

Det finns ett antal olika typer av databaser och den typ som här kommer att behandlas är relationsdatabasen.

### 3.1.1 Databasmodellering

Databasmodellering är framställningen av en databasmodell utifrån objekt och relationer mellan dessa[1]. Objekt måste ha en unik egenskap eller flera egenskaper som tillsammans är unikt. I relationsdatabaser använder man sig av relationsobjekt som binder samman två objekt för att skapa en relation genom att relationsobjektet delar en egenskap från båda objekten.

### 3.1.2 Databashanterare

Lagring av databaser sker på ett sekundärminne. För att kommunicera mot databasen krävs en databashanterare. Databashanteraren svarar på frågor som ställs mot databasen och utför ändringar av innehållet i databasen. MySQL är en av världens mest populära databashanterare[2].

### 3.1.3 Frågespråk

Språket SQL[1] används för att skicka frågor och kommandon till en databashanterare[3]. Genom SQL-satser kan poster i databasen hämtas, tas bort eller ändras. Det är även möjligt att genom SQL lägga till och ta bort använder, ändra användares rättigheter och även att skapa nya tabeller.

## 3.2 Grafiskt användargränssnitt i Java

Denna sektion behandlar skapandet av ett grafiskt användargränssnitt i Java med Swing och AWT[2].

### 3.2.1 Layout och komponenthierarkier

Swing är ett bibliotek med klasser för skapande av grafiska användargränssnitt. Swing bygger på och använder det äldre AWT. I Swing finns klasser för de vanligaste grafiska komponenter exempelvis. knappar, textfält och listor.

Alla Swing-komponenter är containers och kan innehålla andra komponenter, även om bara vissa specifika komponenter är avsedda att användas som containers[4]. Den vanligaste containern för andra komponenter är en panel, JPanel. Genom att komponenter skapas och läggs till i containers upprättas komponenthierarkier. Användandet av denna hierarkistruktur är en implementation av designmönstret Composite[5].

För att en komponenthierarki ska vara synlig för användaren måste den vara förankrad i en toppcontainer. Toppcontainers innefattar fönster, applets och dialogrutor.

För att styra hur komponenterna i en container placeras används en layouthanterare, en LayoutManager[6]. För att uppnå den önskade layouten är det ofta vanligt att

[1]Struktured Query Language - strukturerat frågespråk
[2]Abstract Window Toolkit - Abstrakt fönster bibliotek

kombinera layouthanterare genom att använda olika layouthanterare för containrarna i komponenthierarkin.

### 3.2.2 Händelsehantering

För att göra det grafiska användargränssnittet interaktivt är det nödvändigt att programmet reagerar på användarens handlingar. För att uppnå detta implementeras designmönstret Observer[7] genom användandet av händelser och lyssnarinterface.
Vid interaktion från användaren skapas händelseobjekt och skickas till alla lyssnare som är registrerade hos komponenten. På detta sätta är det möjligt att reagera på exempelvis klickningar på knappar, textinmatning till textfält eller att muspekaren flyttas.

### 3.2.3 Modell, vy och kontroller

I Swing är data och presentationen av data separerad enligt modell, vy och kontroller designmönstret[8]. Denna separation medför flera tekniska fördelar och gränssnittet försöks hållas enkelt genom att vyn har metoder som delegerar till modellen och kontrollern. Separationen medför även vissa komplikationer då kunskap om användningen av designmönstret är nödvändig för att implementera mer avancerade funktioner, som att bytta innehållet i en tabell genom att koppla den till en annan modell eller att lägga till en lyssnare som reagerar på inmatning till ett textfält.

## 3.3 Revisionshantering

Vid programmering är användandet av ett revisionshantingssystem en stor fördel och vid arbete i grupp är det en nödvändighet.
Speciellt användbar är dessa system när det kommer till större projekt för att hålla koll på versioner och kunna sammanfoga olika delar som skrivs av olika programmerare.

Några av de populäraste revionshanteringssystemen är Subversion[9], CVS[10], Git[11] och Mercurial[12]. Subversion och CVS använder en klient-server lösning, medan de modernare Git och Mercurial använder en distribuerad lösning utan centrallagring.

## 3.4 Utvecklingsmilö

Vid programmerande är det viktigt att använda en bra utvecklingsmiljö. En bra utvecklingsmiljö har funktioner som är nödvändiga och medför att användandet av en vanlig texteditor är otänkbart för allt utom det mest rudimentära arbete. Dessa funktioner innefattar automatisk kompilering av kod, förslag för automatisk ifyllning och debugger.

Två av de populäraste utvecklingsmiljöerna för Java-programmering är Eclipse[13] och NetBeans[14]. Båda utvecklingsmiljöerna använder ett system där tilläggsmoduler kan installeras för att få tillgång till funktioner som inte ingår i standardinstallationen. Tilläggen omfattar verktyg till Eclipse för att genera grafiska användargränssnitt[15][16], något som är en standardfunktion i NetBeans[17]. Tilläggsmoduler finns för de vanligaste revisionshanteringssystemen, däribland Subversive[18] som tillför Subversion-stöd till Eclipse.

# 4 Genomförande

Använde ett arbetssätt där ett möte med beställaren hölls varje vecka. Vid mötet redovisades arbetet under föregående veckan, de nya funktionerna demonstrerades och beställaren framförde vilka ytterligare funktioner som önskades.

Första steget under utvecklandet av systemet var att systemera. Systemeringen följdes av implementation av databasen och slutligen skapande av användarapplikationen.

Till databashanterare valdes MySQL (Community Edition version 5.5.22). MySQL valdes för att det är en FOSS[3]-lösning som kan tillgodose systemets nuvarande och framtida behov. En bidragande anledning var även förekomsten av drivrutiner för att enkelt kunna kommunicera med databasen. För skapande och administrering av databasen användes MySQL Workbench (5.2.38 CE).

Valde att använda Java för att skriva applikationen. Detta val gjordes för att Java är plattformsoberoende och gott stöd för kommunikation mot databaser finns. En bidragande faktor var även författarnas tidigare erfarenheter av språket och dess bibliotek. Genom att välja ett språk som är välbekant kunde fokus läggas på att utveckla systemet istället för att sätta sig in i ett nytt språk.
Applikationen skrevs i Eclipse med Subversive för revisionshantering.

Möjlighet finns att kryptera all trafik mellan databasen och applikationen med SSL[4]. Ingen av informationen bedöms vara känslig nog för att detta ska vara nödvändigt.

För att uppfylla licenserna för de bibliotek som används distribueras licenserna för biblioteken tillsammans med applikationen.

## 4.1 Designfilosofi

Vid designen av systemet gjordes en mängd beslut för hur systemet ska fungera och systemets delar interagera. Dessa lösningar syftar till att effektivisera systemet och göra det öppet för förändringar.

### 4.1.1 Modifierbarhet

Ett mål under utvecklingen var att systemet ska vara så anpassningsbart som möjligt utan att applikationens källkod ska behöva ändras. Detta uppnås genom användandet av txt och properties-filer. En properties-fil är en textfil innehåller som en uppsättning med nyckel-värde par.

Sökvägen till databasen och till de datakällor som används är angivet i en properties-fil vilket medför att databasen kan flyttas och enda uppdateringen av applikationen som måste göras är att med en texteditor uppdatera sökvägen i properties-filen. Inloggningsuppgifterna till databasen är dock hårdkodade, då det bedömdes som osäkert att ha dem tillgängliga i klartext.

Vid tjänsteplaneringen anges finansiell information i form av olika koder. Ett krav var att det inte skulle vara möjligt att ange en ej existerande kod. Hårdkodning av de finansiella koderna bedömdes olämpligt, ifall någon ny kod skulle tillkomma. Detta

[3]Free and open source software - Gratis programvara med öppen källkod.
[4]Secure Socket Layer - Protokoll för säker kommunikation över internet[19].

löses genom att för varje finansiell post ha en txt-fil med de giltiga koderna för den posten.

All förutbestämd text på rapporten, det som inte anges av användaren eller genereras från data, hämtas från en properties-fil. Genom denna lösning kan innehållet i rapporterna som genereras enkelt uppdateras.

### 4.1.2 Minimerande av kommunikation mot databasen

En av fördelarna med att använda en databas för att lagra informationen är att det möjliggör en klient-server lösning där applikationen kan köras på vilken dator som helst med tillgång till internet och kommunicera med servern som hanterar databasen. Kommunikation över nätverk kan dock vara kostsam i den bemärkelse att det medför fördröjningar när applikationen väntar på svar. Därför har en målsättning under utvecklingen varit att minimera mängden kommunikation mellan databasen och applikationen. För att uppnå detta lagras all information från databasen lokalt och hämtas vid starten av applikationen.

### 4.1.3 Endast nödvändig data i databasen

Ett möjligt sätt att utnyttja de befintliga datakällorna vore att lägga in allt det önskade innehållet i den egna databasen. Denna lösning medför extremt mycket redundant information, då systemet kommer att användas för att tjänsteplanera endast en liten del av de anställda och endast en mycket liten del av alla de kurser som hålls.

Har valt att hämta all data från datakällorna till applikationen och endast tillföra något till databasen när det används. Detta medför att exempelvis en person kommer att läggas till i databasen först när något planeras för personens tjänst eller en kompetensnivå sätts för ett kompetensområde. Undantaget till detta är om användaren ändrar på eller lägger till något manuellt, vilket alltid lagras i databasen.

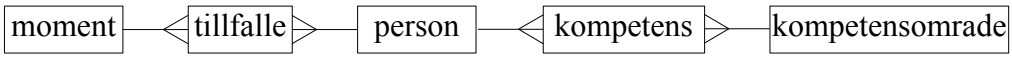
## 4.2 Databas

Designen av databasen utgick från det nuvarande systemet med Excel-ark. I samråd med uppdragsgivaren identifierades vilken data från Excel-arken som ska lagras i databasen. Från denna information identifierades de entiteter som är nödvändiga. Systemet utökades till att även omfatta kompetenser och kompetensområden, vilket inte ingår i det nuvarande systemet. Se tabell 1 för vad entiteterna i databasen representerar.

**Tabell 1. Entiter i databasen.**

| **Entitet** | **Lagrar** |
|---|---|
| moment | Aktiviteter att utföra bl.a. undervisning, kompetensutveckling, möten, semester |
| tillfalle | Utförandet av ett moment av en person under en bestämd tidsperiod. |
| person | Personalen som ska tjänsteplaneras. |
| kompetens | En persons kompetens inom ett kompetensområde. |
| kompetensomrade | De olika kompetensområdena. |

Relationerna mellan de olika entiteterna identifierades. Entiteterna och deras relationer illustreras med kråkfots notation i figur 1. För fullständig betäckning över tabellerna i databasen se bilaga 2.




*Figur 1. Relationsdiagram över entiteterna i databasen.*

Vid designen av databasen är det problematiskt att avgöra hur stor längden för vissa poster ska vara. Denna svårighet gäller främst poster som lagrar text. Vid insättning av värden som överskrider gränsen kommer de tecken som överskrider gränsen att trunkeras eller insättningen att misslyckas, beroende på hur databasen är konfigurerad. Undersökning av minnesanvändning för datatypen VARCHAR i MySQL[20] visade att minnesanvändningen endast är beroende av längden på texten som lagras. Detta medför att det inte finns någon nackdel med att sätta den maximala längden för posterna med datatypen så långa att problem aldrig kommer att uppstå.

## 4.3 Befintliga datakällor

Ett av målen var att systemet ska använda befintliga datakällor där det var möjligt för att inte användaren ska behöva ange information som redan finns, men i ett annat system. Den information som var av intresse var de kurser som hålls på Högskolan i Gävle och den personal som är anställd där. Tillgång till databaserna där informationen är lagrad var omöjlig på grund av säkerhetsskäl. Detta hinder kunde kringgås genom att IT-avdelningen tillhandahåller CSV-filer[5] med den önskade datan. CSV-filerna kan hämtas från en webserver där de uppdateras två gånger dagligen.

## 4.4 Applikation

### 4.4.1 Lagring av data

För varje tabell i databasen finns en motsvarande Java-klass. Varje klass har attribut av lämplig datatyp som motsvarar vad som lagras på en rad i tabellen. Objekt av dessa klasser används för att lagra informationen från databasen och datakällorna. Relationer i databasen bevaras genom komposition[6].

### 4.4.2 Kontroller

Lagrande av data och kommunikation med databasen hanteras av applikationens kontroller. Vid uppstart av applikationen upprättas anslutningen till databasen och en serie frågor ställs. Varje fråga hämtar en av databasens tabeller och skapar ett objekt innehållande datan på varje rad i tabellen. De tabeller som inte har främmande nycklar hämtas först och objekten som skapas lagras i hashtabeller med värdet från primärnyckeln som nyckel. Hashtabellerna används för att hämta objekten motsvarande de främmande nycklarna när tabeller innehållande främmande nycklar hämtas. Då hashfunktionen för objekten är väl vald medför användandet av hashtabeller att söktid för varje objekt motsvarande den främmande nyckeln blir konstant[21]. Resultatet är att tiden att kopiera databasens innehåll ökar linjärt med innehållet i databasen.

[5]Comma-separated values – Lista med värden där värden på samma rad separeras med ett skiljetecken, vanligen ett komma.
[6]Att ett objekt består av flera andra objekt, t.ex. kompetens som består av en person och ett kompetensområde.

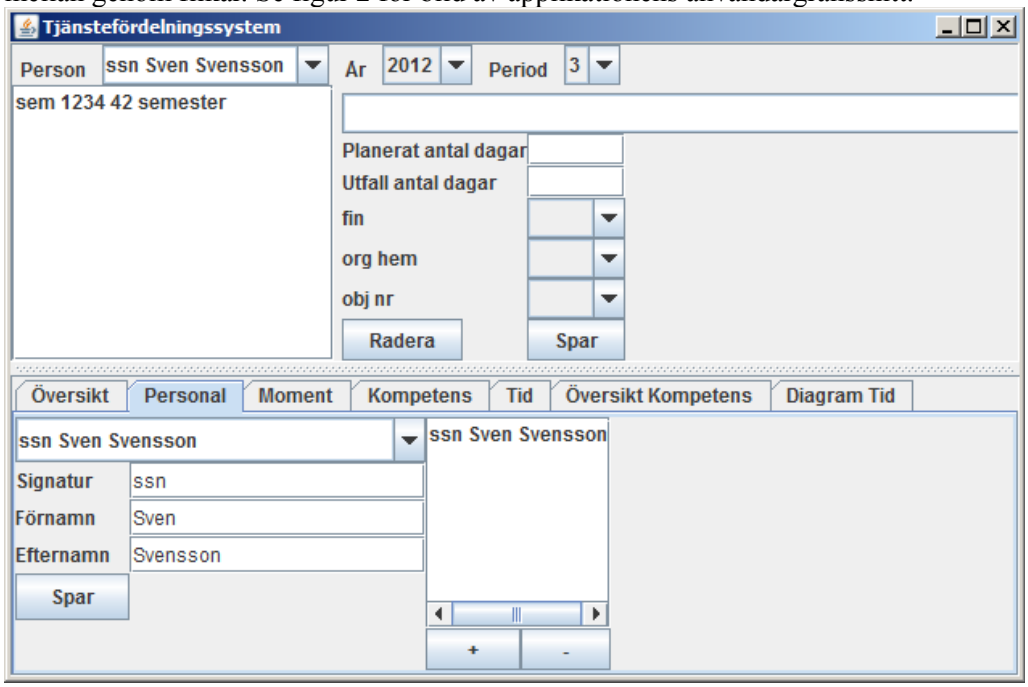
Objekten från tabellerna med främmande nycklar lagras i listor[7] och listor skapas med innehållet från hashtabellerna. Listorna med objekt där en naturlig sorteringsordning finns, t.ex. signaturen för personal, sorteras och sorteras om vid tillägg för att garantera att informationen som visas för användaren alltid är i rätt ordning.

När användaren påverkar ett objekt (lägger till, tar bort eller ändrar något av objektets attribut) utförs ett SQL-kommando mot databasen som upprepar förändringen i databasen. Om objektet består av andra objekt och dessa inte finns i databasen läggs de till först.

### 4.4.3 Grafiskt användargränssnitt

För att skapa det grafiska användargränssnittet användes grafikbiblioteken Swing och AWT. Ingen verktyg för generande av gränssnitt användes, hela gränssnittet skrevs för hand. Önskad funktionalitet som behövde tillföras var att en JcomboBox, komponenten för ett textfält med drop-down lista för val av alternativ, ska filtrera listan till att bara innehålla de alternativ som överensstämmer med den inmatade texten. Det var även nödvändigt att utöka komponenten för en tabell med möjligheten att anpassa kolumnbredden efter innehållet i kolumnerna. Dessa klasser bygger på exempelkod[22][23].

Gränssnittet består av två delar åtskilda av en horisontell separator. Separatorn kan flyttas för att anpassa hur stor yta vardera del upptar. Den övre delen i gränssnittet innefattar en panel för att skapa, uppdatera och ta bort personalens arbetsuppgifter för vald period och år. Den undre delen innehåller flera olika paneler som går att växla mellan genom flikar. Se figur 2 för bild av applikationens användargränssnitt.




*Figur 2 Applikationens användargränssnitt med tjänsteplanering uppe och fliken för hantering av personal vald nere.*

[7]Vector används för listorna då den kan användas direkt ihop med klasserna i Swing.

Flikarna i den undre delen är:

**Personal**
Då all personal på HiG laddas in från en CSV-fil är det nödvändigt att kunna avgränsa och välja vilken personal som ska tjänstefördelas. Detta sker genom en lista som visar den personal som är vald för hantering och möjligheten att lägga till eller ta bort personal från listan. Listan på vald personal sparas och laddas vid start av applikationen. Under denna flik finns även möjligheten att lägga till och ändra uppgifter för personalen.

**Översikt**
Visar en tabell med personalens inplanerade dagar för ett givet år. Visar alla planerade arbetsuppgifter eller en summering per kategori och person. Vid visning av alla arbetsuppgifter kan avgränsning göras till person och/eller period. När avgränsning är gjord till person och alla perioder kan utskrift av rapport göras. Vid utskrift öppnas ett nytt fönster med en förhandsvisning av rapporten. Rapporten är utformad för att till så hög grad som möjligt likna Excel-arket som används i det nuvarande systemet. Vissa fält i förhandsvisningen går att editera innan sidan skrivs ut. Dessa är initierade till standardvärden hämtade från en properties-fil.

**Tid**
Funktion för att finna den personal som för ett givet år eller år och period har en planerad tid som över eller understiger ett angivet värde. Resultatet redovisas i en tabell.

**Diagram Tid**
Visar ett stapeldiagram över arbetade dagar för ett givet år. Höjden på staplarna anger antalet arbetade dagar och färgerna visar kategorierna för de utförda arbetsuppgifterna. Genererande av diagram saknas i Javas standardbibliotek. Biblioteket jCharts[24] används för skapandet av diagram.

**Moment**
Hantering av de arbetsuppgifter som förekommer. Nya arbetsuppgifter kan läggas till och befintliga kan ändras.

**Kompetens**
Skapande, raderande och uppdaterande av kompetenser för personal. Möjlighet finns också att byta namn på kompetensområden.

**Översikt kompetens**
Visar en tabell med översikt över de kompetenser som är inlagda. Visning kan göras per person, per kompetensområde eller all personal grupperad per kompetensområde. Kompetensnivån kan uppdateras genom ändring i tabellen.

För bilder av alla användargränssnittets delar, beskrivning av funktionerna och instruktioner för hur det ska användas se användarhandledningen i bilaga 3.

För att alla paneler ska uppdateras när data i kontrollern ändrats meddelas huvudklassen för det grafiska gränssnittet, som meddelar alla paneler att en förändring har skett och vilken data som har förändrats. Att meddela panelerna har delegerats till huvudklassen för att minska kopplingarna mellan grafikklasserna och kontrollern.
För att panelerna ska ha ett gemensamt interface mot huvudklassen ärver de en abstrakt basklass där metoderna för att utföra uppdateringarna är deklarerade. Basklassen innehåller även hjälpmetoder som utnyttjas av de ärvande klasserna.

# 5 Resultat

Resultatet är ett system som efter inledande konfigurering medför effektivare ifyllande av informationen för tjänsteplanering. Denna effektivisering är huvudsakligen en följd av att kursnamn (vid undervisning) inte behöver skrivas in, då uppgifterna för alla kurser lagras i applikationen.

Nackdelen är att systemet måste konfigureras genom att de personer som ska tjänsteplaneras måste väljas, txt-filerna med de alternativen för de finansiella posterna måste fyllas i och de arbetsuppgifter som inte är undervisning måste läggas in. Tiden för att utföra denna konfigurering är en engångskostnad.

De främsta fördelarna med systemet är den överblick som tjänsteplaneraren får och som tidigare var omöjlig. Stora mängder tid kommer att kunna sparas genom att enkelt få information om vilken personal som inte är fullbelagd. Denna information är nödvändig för att besluta hur personalen ska tjänstefördelas.

Systemet kommer även att vara kompatibelt med de administrativa processer på Högskolan i Gävle då de rapporter som generas är snarlika de gamla, se bilagor 1 och 4.

Beställaren var mycket nöjd med slutresultatet. Resultatet överträffade de förhoppningar beställaren hade vid arbetets början.

För källkoden till applikationen kontakta någon av författarna.

# 6 Diskussion

En nackdel med minimerandet av vad som lagras i databasen, beskrivet i 4.1.3, är att om datakällorna inte är tillgängliga kan användaren vara tvungen att manuellt lägga till ny information som annars hade varit tillgänglig från datakällorna. Risken för detta bedöms vara liten då upptiden på datakällorna hittills varit utmärkt och om databasen som detta system använder körs på samma servrar skulle även detta system vara ur funktion.

Ytterligare testning av applikationen och förbättrad felhantering behövs. Fel som aldrig förutsågs under utvecklingen kan enkelt förekomma när användaren ska bruka applikationen. Ett av dessa fel är användandet av en inaktuell version av en properties-fil där problemet inte meddelas användaren.

För att göra koden till applikationen mer lämpad för vidare utveckling, lättare att underhålla och enklare att läsa kan arbete med att omstrukturera koden vara önskvärt. Lämplig omstrukturering omfattar att identifiera metoder som förekommer i flera klasser och kan flyttas till en gemensam huvudklass, hitta kod som upprepas flera gånger i samma klass och bryta ut det till en metod. Det kan även vara önskvärt att införa interface för att minska beroenden mellan klasser. Kan vara bra att se över namngivningskonventionen, databasen använder svenska namn och applikationen är skriven med engelska.

## 6.1 Möjlighet för vidareutveckling

Det är möjligt att skapa ett enklare och mer estetiskt tilltalande gränssnitt som också tillför ytterligare funktionalitet. För att designa ett bättre användargränssnitt vore det bra att införa en utvecklingscykel där applikationen används och användarens intryck ligger till grund för applikationens fortsatta utveckling.

Under utvecklingen har ett flertal möjliga förbättringar och mer heltäckande lösningar identifierats. Arbetet behövde avgränsas för att passa tidsramen och därför kunde dessa inte implementeras. Förbättring som identifierats är:

Att om kontakten mellan applikationen och databasen bryts ska applikationen försöka återupprätta kontakten. För närvarande blir användaren informerad om felet och måste starta om applikationen för att återupprätta kontakten.

Om systemet ska utökas till att omfatta flera användare är det nödvändigt att ändra hur applikationen kommunicerar med databasen. En server som hanterar kommunikationen mellan klienterna och databasen är nödvändig. Om en klient ska ändra på innehåll i databasen meddelar den servern, servern uppdaterar databasen och meddelar alla klienterna. För en lösning med flera klienter kan det även vara nödvändigt att i databasen lagra användare, inloggningsuppgifter och användarnas befogenheter.

Rapporten kan komma att behöva ha olika utseenden beroende på tjänsteplanerarens avdelning. Vid lösningen med flera klienter kan olika mallar användas beroende på vilken avdelning den inloggade tillhör. Ytterligare typer av rapporter kan vara önskvärt att generera.

Det är möjligt att kryptera all trafik mellan databas och applikation med SSL. Denna möjlighet kan komma att vara önskvärd att utnyttja. Det är även möjligt att använda SSL för att kryptera trafiken mellan klienterna och servern i klient-server lösningen som diskuterades ovan.

Även skapandet av verktyg för hantering och ändring av inställningar och ändring av fontstorleken i applikationen kan vara önskvärt.

# 7 Slutsats

Det finns klara fördelar med att använda ett tjänsteplaneringssystem som underlättar för tjänsteplaneraren under planeringsprocessen. Systemet har skapats för att ge detta stöd, men ytterligare funktioner kan vara nödvändiga för ett ännu bättre stöd. Att med användare testa systemet är den naturliga fortsättningen för om systemet ska vidareutvecklas.

Systemet är designat för att användas av endast en användare. Under utvecklingen har det varit tydligt att det även fungerar med flera samtidiga användare. Då systemet är designat för endast en användare saknas funktioner som skulle vara en självklar del om systemet hade designats för flera samtidiga användare. Den tydligaste av dessa brister vid flera samtidiga användare är att klienten inte hålls uppdaterad med vad de övriga klienterna utför. Denna brist är helt irrelevant om varje användare tjänsteplanerar för olika grupper personal. Systemet är komplett nog för att användas skarpt av en användare, men rekommenderas inte för användning av flera användare om det förekommer överlappning mellan de grupper som ska tjänsteplaneras.

## Referenser


[1] L. Axelsson, M. Hidefjäll *Praktisk datamodellering – ta greppet om begreppen* Studentlitteratur 1993, sid 19 – 44
[2] MySQL http://www.mysql.com/why-mysql/marketshare/
[3] J. Groff, P. Weinberg, A. Oppel *SQL The Complete Reference Third Edition* McGraw Hill 2010, sid. 3-20.
[4] J. Skansholm, ”Utplacering av barnkomponenter”, *Java direkt med Swing,* Studentlitteratur, 2005, sid. 180-181
[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides, ”Composite”, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison Wesley, 1994, sid. 183 – 195
[6] A Visual Guide to Layout Managers http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides, ”Observer”, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison Wesley, 1994, sid. 326 – 337
[8] A Swing Architecture Overview http://java.sun.com/products/jfc/tsc/articles/architecture/
[9] Apache Subversion http://subversion.apache.org/
[10] CVS - Concurrent Versions System http://www.nongnu.org/cvs/
[11] Git http://git-scm.com/
[12] Mercurial http://mercurial.selenic.com/
[13] Eclipse http://www.eclipse.org/
[14] NetBeans http://netbeans.org/index.html
[15] Jigloo SWT/Swing GUI Builder for Eclipse and WebSphere http://www.cloudgarden.com/jigloo/
[16] WindowBuilder Pro https://developers.google.com/java-dev-tools/wbpro/
[17] NetBeans GUI Builder http://netbeans.org/features/java/swing.html
[18] Subversive http://www.eclipse.org/subversive/
[19] RFC 6101 – The Secure Socket Layer (SSL) Protocol Version 3.0 http://tools.ietf.org/html/rfc6101
[20] MySQL 5.0 Reference Manual : 11.4.1 The CHAR and VARCHAR Types http://dev.mysql.com/doc/refman/5.0/en/char.html
[21] D.E. Knuth ”Hashing”*The Art Of Computer Programming, Volume 3, Sorting and Searching* (2nd ed.), Addison Wesley, 1998, sid. 513 – 549
[22] Presenting: Auto Adjust Combo Box http://www.coderanch.com/t/561143/GUI/java/Presenting-Auto-Adjust-Combo-Box
[23] Packing a Column of a JTable Component http://www.exampledepot.com/egs/javax.swing.table/PackCol.html
[24] JCharts http://jcharts.sourceforge.net/

# Bilaga 1: Excel-ark från nuvarande system

HÖGSKOLAN I GÄVLE
Institutionen för teknik och byggd miljö

**2011:årsplan**

**PLANERINGS- OCH BOKFÖRINGSUNDERLAG TJÄNSTEFÖRDELNING**

| | |
|---|---|
| AKADEMI | Akademin för teknik och miljö |
| AVDELNING | Avdelningen för industriell utveckling, IT och |
| NAMN | Test Test |
| Användarnamn | Test |
| Anställningstid | Test |
| Tjänsteomfattning | Test |

| | |
|---|---|
| Tjänstgöringsskyldighet, antal dagar | 214,0 |
| Tillkommer: Årets semester | 35,0 |
| Tillkommer: Tj ledigt | 0,0 |
| ATT REDOVISA: | 249,0 |
| BOKFÖRDA DAGAR: | 57,0 |
| Övertid (+) / undertid (-) vid årets slut | -192,0 |

| PER | ORG HEM | OBJ NR | FIN | KURS/PROJEKT | AKT | DAGAR I TJÄNST S:A Prel | S:A Utfall | Diff |
|---|---|---|---|---|---|---|---|---|
| 3 | 111 | 111 | 1 | Mockup1 | utb | 8,0 | 8,0 | 0,0 |
| 3 | 111 | 111 | 1 | Mockup2 | utb | 5,0 | 5,0 | 0,0 |
| 3 | 111 | 111 | 1 | Mockup3 | övr | 5,0 | 5,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| 3 | | | | | | 0,0 | 0,0 | 0,0 |
| SUMMA PERIOD 3 | | | | | | 18,0 | 18,0 | 0,0 |
| 4 | 111 | 111 | 1 | Mockup1 | utb | 10,0 | 10,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| 4 | | | | | | 0,0 | 0,0 | 0,0 |
| SUMMA PERIOD 4 | | | | | | 10,0 | 10,0 | 0,0 |
| 1 | 111 | 111 | 1 | Mockup1 | utb | 17,0 | 17,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| 1 | | | | | | 0,0 | 0,0 | 0,0 |
| SUMMA PERIOD 1 | | | | | | 17,0 | 17,0 | 0,0 |
| 2 | 111 | 111 | 1 | Mockup1 | utb | 11,0 | 11,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| 2 | | | | | | 0,0 | 1,0 | 1,0 |
| 2 | | | | | | 0,0 | 0,0 | 0,0 |
| SUMMA PERIOD 2 | | | | | | 11,0 | 12,0 | 1,0 |
| SUMMA TOTALT | | | | | | 56,0 | 57,0 | 1,0 |

Lärarens underskrift

Avdelningschef / Prefekts underskrift

2012-05-29

Kapningen av datumet nere till höger uppstod vid utskrift till pdf. Detta antas vara en följd av brister i kompatibiliteten mellan läsaren som användes och xls-filformatet.

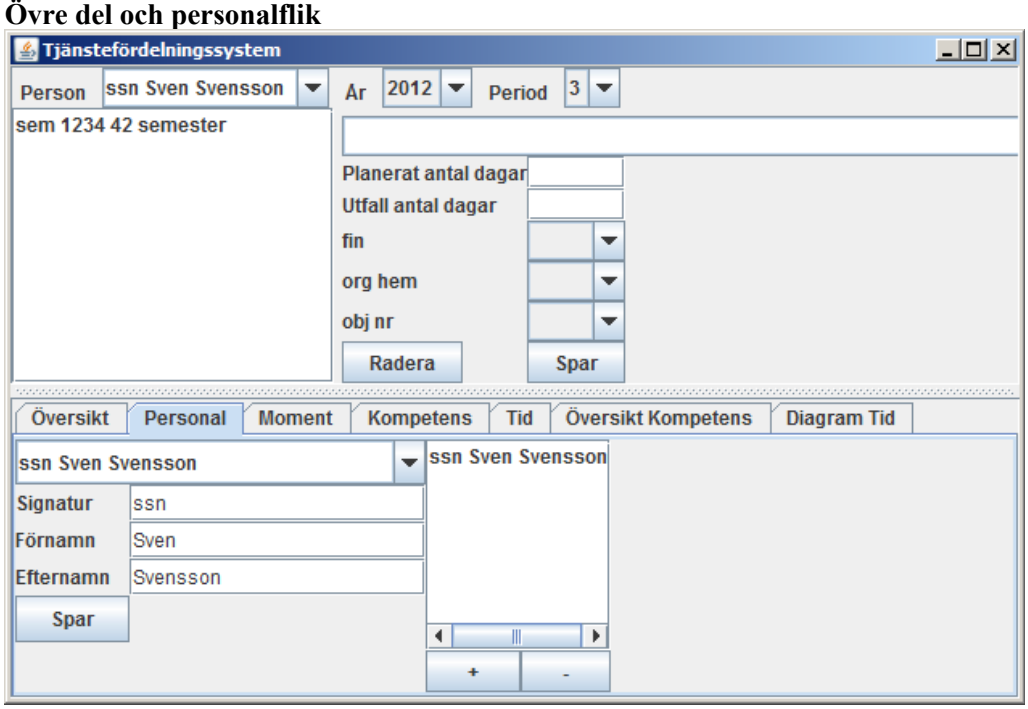
# Bilaga 2: Termkatalog över databasen

| **tabellnamn** | | | |
|---|---|---|---|
| kolumn | datatyp | egenskaper* | beskrivning |
| | | | |
| **kompetens** | | | |
| person_signatur | VARCHAR(10) | PK, FK, NN | Personens signatur |
| kompetensomrade_id_ kompetensomrade | INT | PK, FK, NN | Id för kompetensområdet |
| niva | INT | NN | Personens kompetens inom området (1 till 10) |
| | | | |
| **kompetensomrade** | | | |
| id_kompetensomrade | TINY INT | PK, NN, AI | Id för kompetensområdet |
| beskrivning | VARCHAR(1000) | | namn/beskrivning för kompetensområdet |
| | | | |
| **moment** | | | |
| id_moment | INT | PK, NN | Id för momentet, ansökningsnummer för kurser |
| namn | VARCHAR(1000) | | Namnet på kursen/aktiviteten |
| akt | VARCHAR(10) | | kategori |
| kod | VARCHAR(10) | | kurskoden |
| | | | |
| **person** | | | |
| signatur | VARCHAR(10) | PK, NN | Personens signatur |
| for_namn | VARCHAR(200) | | förnamnet |
| efter_namn | VARCHAR(200) | | efternamnet |
| | | | |
| **tillfalle** | | | |
| id_tillfalle | INT | PK, NN, AI | Id för tillfället, löpnummer |
| period | INT | | Perioden, 1 till 4 |
| ar | INT | | Året |
| org_hem | INT | | ekonomi information |
| obj_nr | INT | | ekonomi information |
| fin | INT | | ekonomi information |
| utfall_dagar | FLOAT | | Antalet dagar spenderade på momentet |
| planerade_dagar | FLOAT | | Planerade antalet dagar att spendera på momentet |
| person_signatur | VARCHAR(10) | FK | Personens signatur |
| moment_id_moment | INT | FK | |

* egenskaper: PK primärnyckel. FK främmandenyckel. NN inte null. AI automatisk uppräkning.

# Bilaga 3: Användarhandledning för applikationen

**Övre del och personalflik**




- **Undre delen av programmet**

Den i bilden valda fliken Personal används för hantering av personal.
Högst upp till vänster är en drop-down lista med all personal på högskolan.
Till Höger är en lista över vald personal. Det är endast de personer som finns i listan som kommer att vara alternativ när en person ska hanteras i någon av programmets övriga delar.

Personal kan hanteras på följande sätt:

**Lägga till en person**
Ange signatur, förnamn och efternamn för personen i respektive textfält. Klicka på sparknappen.
Notera att om en person redan finns med den angivna signaturen kommer den befintliga personen att uppdateras.

**Uppdatera en person**
Välj den person som ska uppdateras från drop-down listan. Genomför ändringar i textfälten. Klicka på sparknappen.
En person kan även uppdateras genom att de nya uppgifterna anges direkt i textfälten.
Om signaturen ändras kommer den valda personen inte att uppdateras, personen med den angivna signaturen kommer att uppdateras eller så kommer en ny person med den signaturen att läggas till om ingen person existerar med den signaturen.
**Lägga till en person till vald personal**
Välj en person i drop-down listan. Klicka på knappen med ett plus.

**Ta bort en eller flera personer från vald personal**
Välj en eller flera personer i vald personal listan. Klicka på knappen med ett minus.
Notera att personen/personerna fortfarande finns kvar i drop-down menyn.

- **Övre delen av programmet**

Används för att tjänstefördela personalen. Tillfällen (utförandet av ett moment under en period) kan skapas, uppdateras och raderas. När person, år och period är vald listas alla tillfällen för den personen, året och perioden i listan till vänster. Alternativen för de finansiella posterna (fin, obj nr och org hem) är lagrade i filerna fin.txt, obj.txt och org.txt. Genom att editera filerna kan alternativ läggas till eller tas bort.

**Skapa ett tillfälle**

Välj personen, år och period från respektive drop-down meny. Välj det moment personen ska utföra från drop-down menyn högst upp till höger. Ange planerade dagar, utfall dagar, fin, obj nr och org hem. Klicka på spar.

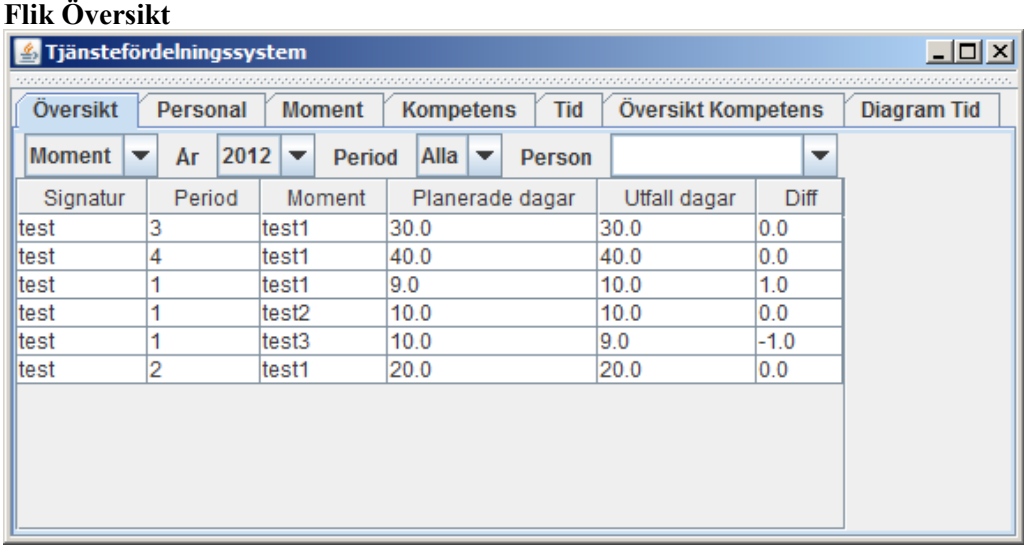
**Uppdatera ett tillfälle**

Välj tillfälle från listan. Uppdatera värdena. Klicka på spar.

**Radera ett tillfälle**

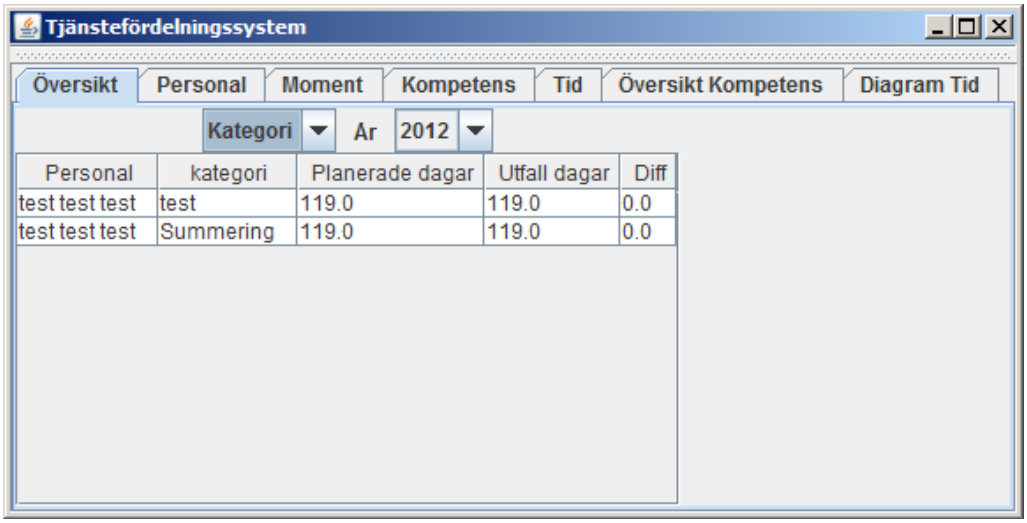
Välj tillfälle från listan. Klicka på radera.

**Flik Översikt**



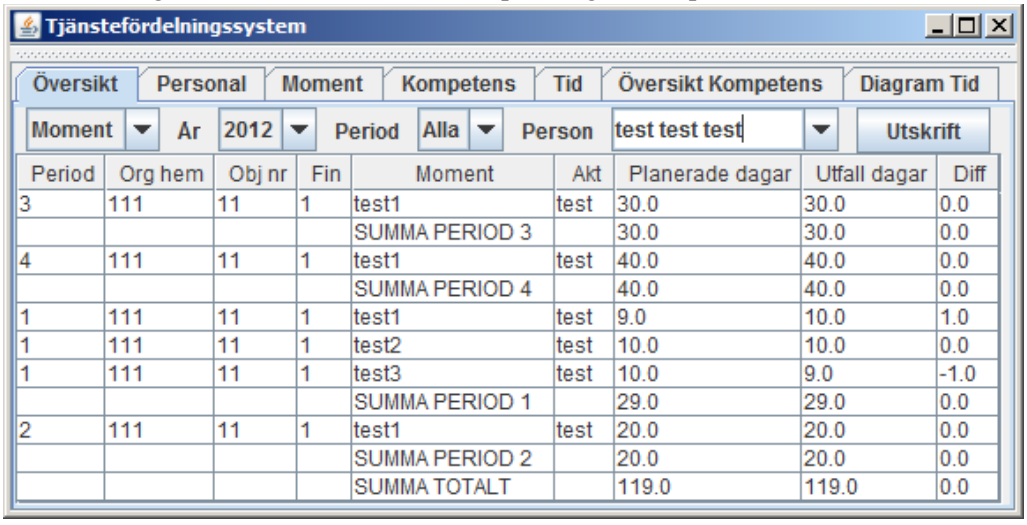

| Signatur | Period | Moment | Planerade dagar | Utfall dagar | Diff |
|---|---|---|---|---|---|
| test | 3 | test1 | 30.0 | 30.0 | 0.0 |
| test | 4 | test1 | 40.0 | 40.0 | 0.0 |
| test | 1 | test1 | 9.0 | 10.0 | 1.0 |
| test | 1 | test2 | 10.0 | 10.0 | 0.0 |
| test | 1 | test3 | 10.0 | 9.0 | -1.0 |
| test | 2 | test1 | 20.0 | 20.0 | 0.0 |

Vid val Moment visas personalens inplanerade tillfällen. Filtrering kan göras på år, period och person.




| Personal | kategori | Planerade dagar | Utfall dagar | Diff |
|---|---|---|---|---|
| test test test | test | 119.0 | 119.0 | 0.0 |
| test test test | Summering | 119.0 | 119.0 | 0.0 |

Vid val Kategori summeras alla tillfällena på kategori och person.



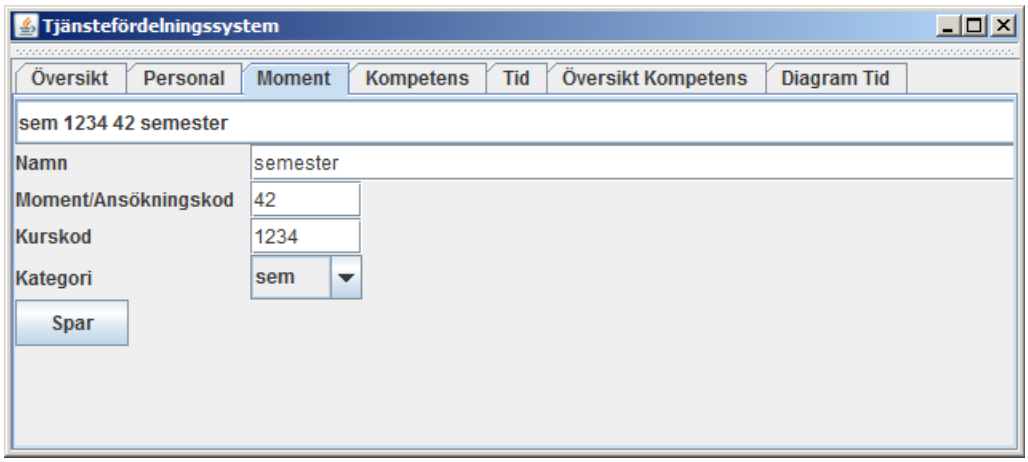

| Period | Org hem | Obj nr | Fin | Moment | Akt | Planerade dagar | Utfall dagar | Diff |
|---|---|---|---|---|---|---|---|---|
| 3 | 111 | 11 | 1 | test1 | test | 30.0 | 30.0 | 0.0 |
| | | | | SUMMA PERIOD 3 | | 30.0 | 30.0 | 0.0 |
| 4 | 111 | 11 | 1 | test1 | test | 40.0 | 40.0 | 0.0 |
| | | | | SUMMA PERIOD 4 | | 40.0 | 40.0 | 0.0 |
| 1 | 111 | 11 | 1 | test1 | test | 9.0 | 10.0 | 1.0 |
| 1 | 111 | 11 | 1 | test2 | test | 10.0 | 10.0 | 0.0 |
| 1 | 111 | 11 | 1 | test3 | test | 10.0 | 9.0 | -1.0 |
| | | | | SUMMA PERIOD 1 | | 29.0 | 29.0 | 0.0 |
| 2 | 111 | 11 | 1 | test1 | test | 20.0 | 20.0 | 0.0 |
| | | | | SUMMA PERIOD 2 | | 20.0 | 20.0 | 0.0 |
| | | | | SUMMA TOTALT | | 119.0 | 119.0 | 0.0 |

Filtrering på person med alla perioder ger möjlighet till utskrift av rapport.


Möjlighet finns att ändra vissa av fälten ovanför tabellen. Bokförda dagar hämtas från tabellen. Övertid/undertid beräknas som differensen mellan bokförda dagar och de dagar som ska redovisas. Rubriker och standardvärdena för de fält som går att ändra på finns i printing.properties och kan ändras. Tryck på skriv ut för utskrift.

Notera att bilden ovan har anpassats genom att övre och undre halvan har klippts ihop och tomt utrymme där emellan har utelämnats.

**Flik Moment**




Flik för att lägga till eller ändra moment.

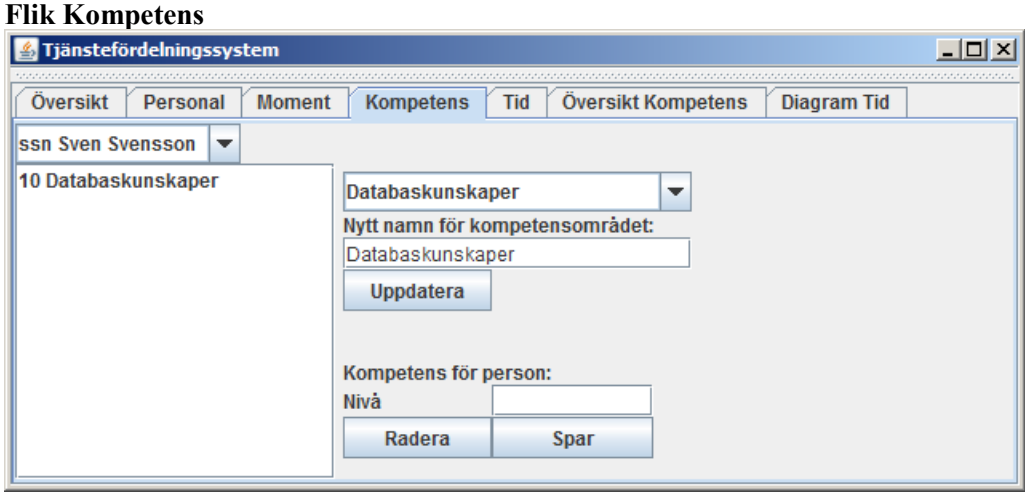
**Lägga till ett moment**

Ange namn, moment/ansökningskod, kurskod och kategori. Klicka på spar.
Om ett moment med den angivna moment/kurskoden redan existerar kommer det momentet att uppdateras.

**Uppdatera ett moment**

Välj moment från listan. Uppdatera värdena. Klicka på spar.

**Flik Kompetens**




För tillägg av kompetenser för personal och hantering av kompetensområden. När en person väljs i drop-down listan uppe till vänster visas alla personens kompetenser i listan.

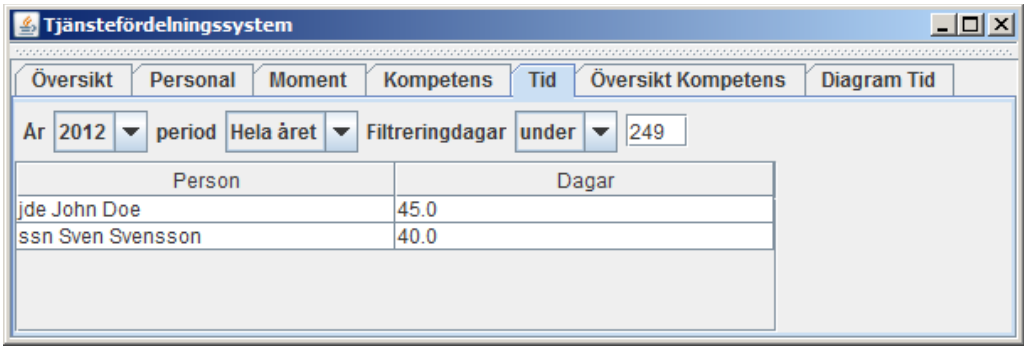
**Lägg till kompetens**
Välj ett befintligt eller ange ett nytt kompetensområde i drop-down listan uppe till höger. Ange kompetensgrad på skalan 1-10. Klicka spar.
Om personen har en kompetens i det valda kompetensområdet kommer den befintliga kompetensen att uppdateras.

**Uppdatering av kompetens**
Välj kompetens från listan. Ange kompetensgrad. Klicka spar.

**Radera kompetens**
Välj kompetens från listan. Klicka radera.

**Uppdatera kompetensområde**
Välj ett kompetensområde. Ange nytt namn. Klicka på uppdatera.

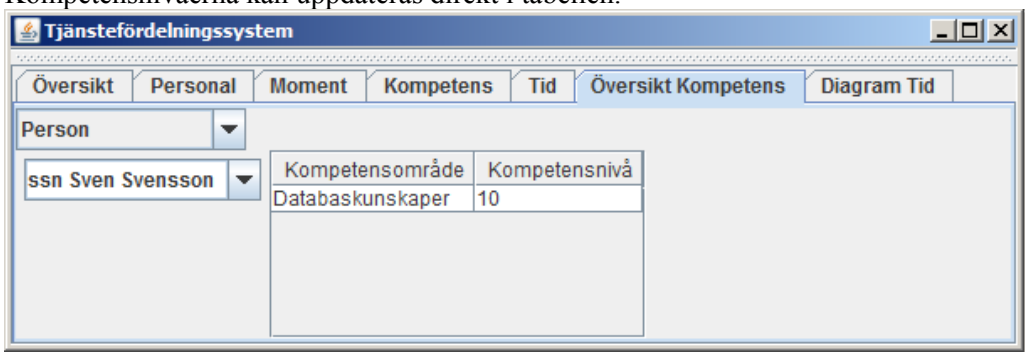
**Flik Tid**



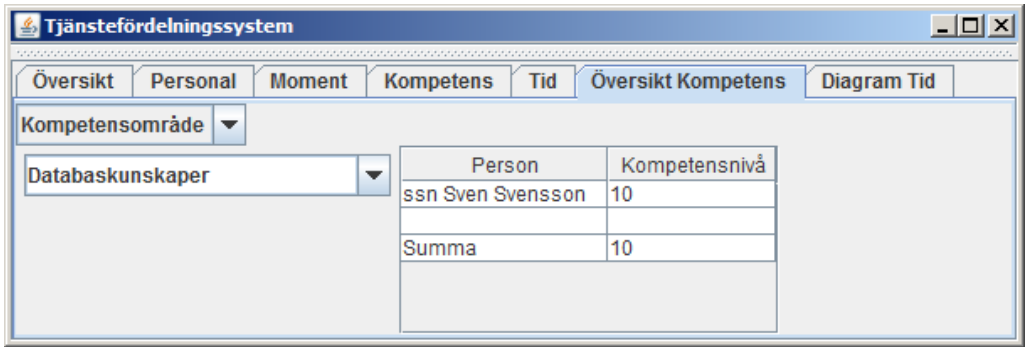

Fliken visar personal och inplanerade dagar där summan planerade dagar för vald tidsperiod under eller överstiger ett specificerat antal dagar.

### Flik Översikt Kompetens

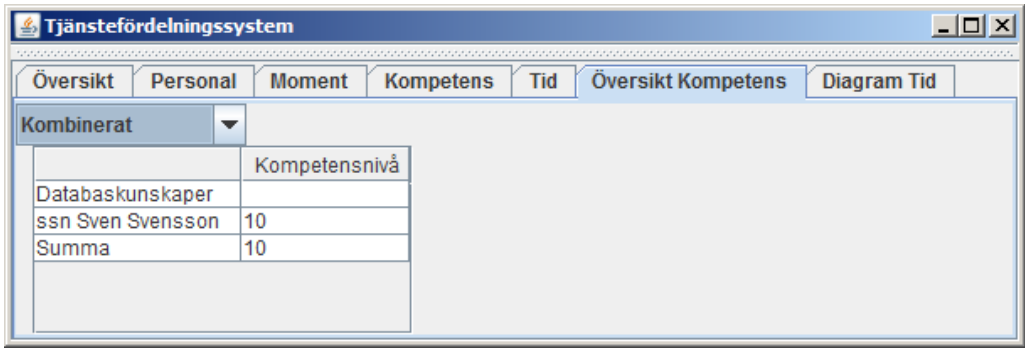
Kompetensnivåerna kan uppdateras direkt i tabellen.




Filtrering på person för översikt av vald persons kompetenser.




Filtrering på kompetensområde. Visar all personal som besitter kompetens inom valt kompetensområde, kompetensnivå och summa.




Kombinerad vy. Visar alla kompetenser grupperat på kompetensområde med summa för varje kompetensområde.

**Flik Diagram Tid**

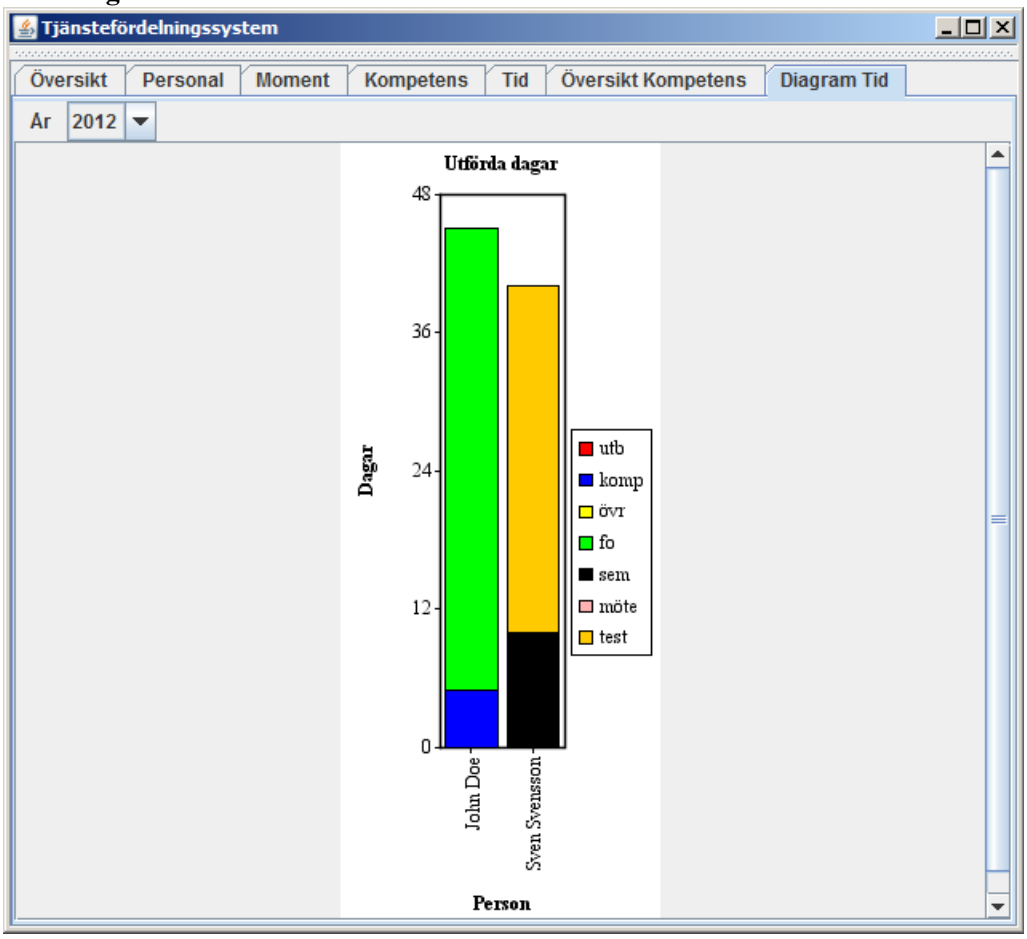



Diagram över personalens antal arbetade dagar för valt år. Färgerna motsvarar de olika kategorierna för momenten.

# Bilaga 4: Utskrift av rapport

**HÖGSKOLAN I GÄVLE**
**Institutionen för teknik och byggd miljö**

**2012:årsplan**

## PLANERINGS- OCH BOKFÖRINGSUNDERLAG TJÄNSTEFÖRDELNING

| | | | |
|---|---|---|---|
| **AKADEMI** | **Akademin för teknik och miljö** | **Tjänstgöringsskyldighet, antal dagar** | **214** |
| **AVDELNING** | **Avdelningen för industriell utveckling, IT och samhällsbyggnad** | **Tillkommer: Årets semester** | **35** |
| **NAMN** | **Sven Svensson** | **Tillkommer: Tj ledigt** | **0** |
| **Användarnamn** | | **ATTREDOVISA:** | **249** |
| **Anställningstid** | **tillsvidare** | **BOKFÖRDA DAGAR:** | **260.0** |
| **Tjänsteomfattning** | **100%** | **Övertid(+)/undertid (-) vid årets slut** | **11.0** |

| Period | Org hem | Obj nr | Fin | Moment | Akt | Planerade dagar | Utfall dagar | Diff |
|---|---|---|---|---|---|---|---|---|
| 3 | 111 | 11 | 1 | ATM-möte | möte | 7.0 | 7.0 | 0.0 |
| 3 | 111 | 11 | 1 | Ekologisk trädgårdsodling 15.00hp | utb | 7.0 | 7.0 | 0.0 |
| 3 | 111 | 11 | 1 | Kompetensutveckling | komp | 15.0 | 15.0 | 0.0 |
| 3 | 111 | 11 | 1 | semester | sem | 10.0 | 10.0 | 0.0 |
| 3 | 111 | 11 | 1 | test1 | test | 30.0 | 30.0 | 0.0 |
| | | | | SUMMA PERIOD 3 | | 69.0 | 69.0 | 0.0 |
| 4 | 111 | 11 | 1 | ATM-möte | möte | 5.0 | 5.0 | 0.0 |
| 4 | 111 | 11 | 1 | Ekologisk trädgårdsodling 15.00hp | utb | 20.0 | 20.0 | 0.0 |
| 4 | 111 | 11 | 1 | Forskning | fo | 20.0 | 20.0 | 0.0 |
| 4 | 111 | 11 | 1 | Människor och trädgårdar 15.00hp | utb | 20.0 | 20.0 | 0.0 |
| 4 | 111 | 11 | 1 | semester | sem | 2.0 | 2.0 | 0.0 |
| | | | | SUMMA PERIOD 4 | | 67.0 | 67.0 | 0.0 |
| 1 | 111 | 11 | 1 | ATM-möte | möte | 12.0 | 12.0 | 0.0 |
| 1 | 111 | 11 | 1 | Ekologisk trädgårdsodling 15.00hp | utb | 10.0 | 10.0 | 0.0 |
| 1 | 111 | 11 | 1 | Kompetensutveckling | komp | 10.0 | 10.0 | 0.0 |
| 1 | 111 | 11 | 1 | Landskap, trädgårdar och parker 7.50hp | utb | 10.0 | 10.0 | 0.0 |
| 1 | 111 | 11 | 1 | semester | sem | 3.0 | 3.0 | 0.0 |
| | | | | SUMMA PERIOD 1 | | 45.0 | 45.0 | 0.0 |
| 2 | 111 | 11 | 1 | Ekologisk trädgårdsodling 15.00hp | utb | 15.0 | 15.0 | 0.0 |
| 2 | 111 | 11 | 1 | Examensarbete inom Trädgårdsmästarprogrammet 15.00hp | utb | 15.0 | 18.0 | 3.0 |
| 2 | 111 | 11 | 1 | Forskning | fo | 15.0 | 11.0 | -4.0 |
| 2 | 123 | 11 | 1 | Kompetensutveckling | komp | 15.0 | 15.0 | 0.0 |
| 2 | 111 | 11 | 1 | semester | sem | 20.0 | 20.0 | 0.0 |
| | | | | SUMMA PERIOD 2 | | 80.0 | 79.0 | -1.0 |
| | | | | SUMMA TOTALT | | 261.0 | 260.0 | -1.0 |

.....
**Lärarens underskrift**

.....
**Avdelningschef / Prefekts underskrift**

**2012-05-29**