Schemaläggningsalgoritm för arbetstidsschema

Scheduling algorithm for work schedules

Mikael Eriksson

EXAMENSARBETE
Datateknik
2006
Nr: **E3426D**

HÖGSKOLAN
Dalarna

EXAMENSARBETE, C-nivå
Datateknik

Program	Reg nr	Omfattning
Datateknik, 120 p	E3426D	10 p
Namn Mikael Eriksson	Datum 2006-05-29	
Handledare Hans Jones	Examinator Ernst Nordström	
Företag/Institution Etex AB	Kontaktperson vid företaget/institutionen Lars Forslöf	
Titel Schemaläggningsalgoritm för arbetstidsschema		
Nyckelord VB.NET, Programmering		

Sammanfattning

Programmet schemaläggaren har funnits sedan 1994, och används på arbetsplatser för att på ett strukturerat sätt lägga scheman för de anställda. Programmet har sedan starten varit sådant att man själv ska skapa alla scheman och även koppla dessa till de anställda, men det finns en önskan att programmet ska vara mer användarvänligt och lättanvänt. Ett steg i utvecklingen är att skapa en automatisk schemaläggningsguide som gör det mesta av jobbet åt användaren.

Etex AB, som detta examensarbete är gjort åt, avser att med detta examensarbete skapa en prototyp på en sådan schemaläggningsguide som ska implementeras i det befintliga programmet. Denna guide ska alltså skapa scheman utifrån given data och sedan koppla dessa scheman till de anställda som valts. Allt detta ska ske automatiskt. Efter detta har man sedan möjlighet att antingen spara de scheman och kopplingar som gjorts, eller avvisa dem.

Grundläggande tanken med denna modul är att den ska passa in i det befintliga programmet utan att vara bundet till det, så man ska kunna överföra den till en annan applikation om nödvändigt. Trots vissa komplikationer efter vägen har arbetet ändå slutförts och med fortsatt arbete kan modulen bli en bra del av programmet Schemaläggaren.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

DALARNA
University College

DEGREE PROJECT
Computer Engineering

Programme	Reg number	Extent
Computer Engineering	E3426D	15 ECTS
Name of student	Year-Month-Day	
Mikael Eriksson	2006-05-29	
Supervisor	Examiner	
Hans Jones	Ernst Nordström	
Company/Department	Supervisor at the Company/Department	
Etex AB	Lars Forslöf	
Title		
Scheduling algorithm for work schedules		
Keywords		
VB.NET, Programming		

Summary

Schemaläggaren was created in 1994 and since that schedulers have been using the program to supervise schedules for employees at the workplace. In the program it's only possible to manually create schedules and then place them on employees, but there's a wish that the program should be easier to use. One thing that would make it easier would be to create an automatic scheduler that would do most of the job instead of the user.

The purpose with this work is to create a prototype of an automatic scheduler that could be implemented into the original program. This scheduler should create schedules from given data, and then place the schedules on those employees that has been chosen. This should all be done automatically. When this has been done, the user either chooses to save the schedules that has been created, or cancel and nothing will be saved.

The basic thought with this module is that it could be used in the original program, but it should also be possible to implement it into other applications if necessary. Even if some complications were met along the way, the goal was reached in time and with some work on this module, it could be a good part of the program Schemaläggaren.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Innehållsförteckning

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

BILAGOR

Bilaga 1 Schemaläggaren
Bilaga 2 Beskrivning av databasen

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

1 Inledning

Detta arbete har utförts åt Etex AB som har verksamhet i Ljusdal och Borlänge och ägs av Lars Forslöf. Etex AB ingår i ett konsultföretag som heter Balanz AB. De har kontor i Borlänge och där har även den stora delen av detta arbete gjorts.

1.1 Bakgrund

Programmet Schemaläggaren gavs ut 1994, och har funnits på marknaden sedan dess. Det utvecklades då i ett företag som hette Ramsjö Tryck och Data, och ägdes av Nils Forslöf. År 2005 startades ett samarbete med Etex AB, där utveckling skett sedan dess. Programmet är ett datorbaserat hjälpmedel för schemaläggning av arbetstider för anställda med Svensk Handels arbetstidsavtal med Handels. Programmet beräknar arbetstider med hänsyn till OB-tider (Obekväma arbetstider, speciell ersättning utgår dessa tider).

1.2 Syfte

Den besvärligaste delen i programvaran Schemaläggaren är den delen då användaren ska skapa scheman, mata in tider för scheman samt koppla dessa till anställda. Syftet med examensarbetet var att undersöka om det var möjligt att hitta en eller flera algoritmer som på ett generellt sätt kan lägga arbetstidsscheman automatiskt utifrån olika inparametrar. En sådan funktion skulle kunna eliminera många arbetstimmar för den som planerar tjänstgöringen på en arbetsplats.

1.3 Mål

Målet var att efter avslutat arbete ha en fungerande demonstration av modulen för programvaran Schemaläggaren. Modulen skulle alltså automatiskt lägga ett eller flera scheman för att passa verksamheten. I modulen skulle det också vara möjligt att välja att lägga scheman för helt år, eller endast en period av året. Det skulle vara möjligt att styra schemaläggningen utefter vissa önskemål, för att få ett mer exakt schema. Modulen skulle lägga scheman efter de då gällande regler för hur mycket personal får jobba etc. Mer information om dessa regler hittas i Arbetstid – detaljhandeln [1].

1.4 Programmeringsspråk

Programmeringsspråket som användes var VB.NET. Det fanns några alternativ till VB.NET, såsom tex. C#, men valet hamnade på VB.NET då Schemaläggaren är skriven i det språket och implementeringen förväntades bli lättare om man använde samma språk.

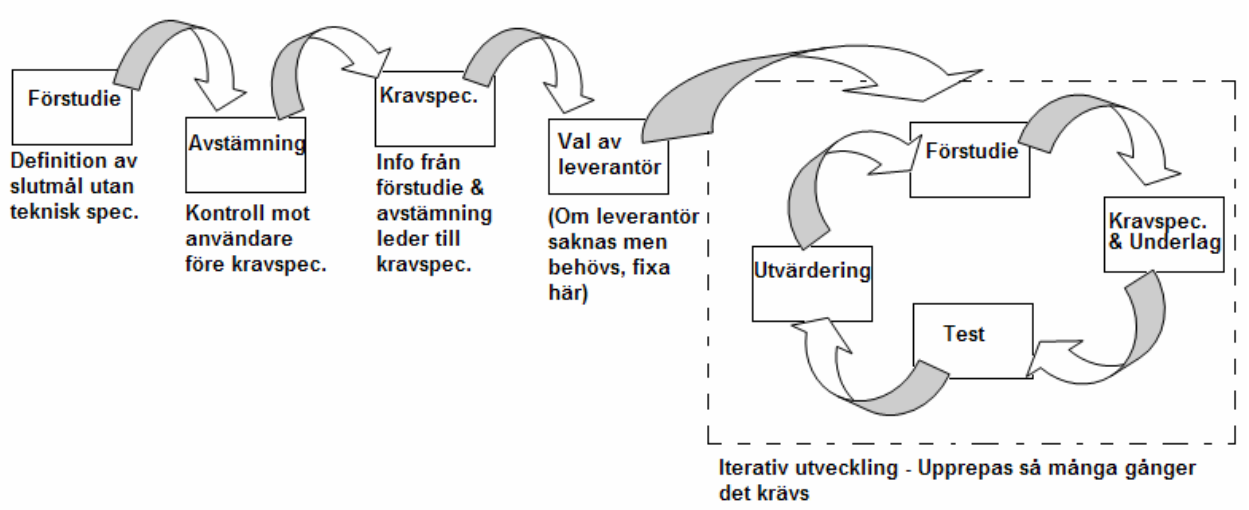
1.5 Databashantering

Schemaläggaren arbetar mot en databas av filtypen mdb, som är en Microsoft databas. För att öppna och läsa i databasen, används programmet Microsoft Access. I databasen kan man hitta all information om arbetsplatsen, dvs. utan en databas kan man inte göra mycket i programmet.

1.6 Metod

Metoden som användes under utvecklingen av denna modul brukar kallas ”Iterativ utveckling” [4] och dess grundtanke är att man tar en liten del av utvecklingsprocessen och gör den så klar att den kan testas i verkligheten. Sedan påbörjas nästa del av utvecklingsprocessen och gör den klar. Det är i princip så man ska hålla på tills hela utvecklingsprocessen är genomförd, se figur 5. Den iterativa utvecklingsmetoden fungerar bra, då man hela tiden tar små steg, istället för att gå direkt från kravspecifikation till färdig produkt. Samtidigt hänger inte allt på det som konstaterats vid den första förstudien, utan det korrigeras allt eftersom.

I detta arbete har ingen avstämning mot befintliga användare skett men däremot har kontakt funnits med tillverkaren av Schemaläggaren som vet hur användarna vill ha det. Det har under arbetet inte heller producerats några dokument från de faser som finns specificerade i figur 5, däremot har alla faser planerats i någon form.



Figur 1. Iterativ utveckling

1.7 Frågeställningar

Följande frågeställningar har kommit fram vid planering och genomförande av arbetet.
1) **Databas** – Lämpligaste databasen att använda, den ursprungliga eller en modifierad databas?
2) **Veckor och helgdagar** – Vad är det lättaste och bästa sättet att få fram veckor och helgdagar?
3) **Antal scheman** – Antalet scheman kan lätt rusa iväg. Hur begränsar man det enklast?
4) **Sparning** – Hur spara poster temporärt, för att kunna avbryta modulen utan att spara?

2 Avgränsningar

Vid början av arbetet fanns inga egentliga avgränsningar, då det var oklart hur mycket arbete som låg bakom uppgiften. När det blev mer känt vad som skulle göras, så växte även vissa avgränsningar fram.

2.1 Avgränsningar på rapporten

Med hänsyn till att detta program säljs av Etex AB, kommer ingen källkod från programmet redovisas i denna rapport. Istället ligger tyngdpunkten på hur systemet fungerar och ser ut.

2.2 Avgränsningar på programmet

De scheman som lagts behövde inte på något vis vara optimala, men ändå tillräckligt bra för att kunna användas (med eventuella ändringar). Modulen skulle endast behöva lägga scheman för arbetsplatser med 6-dagarsöppet, då det är olika regler för arbetsplatser med 6-dagarsöppet och 7-dagarsöppet.

3 Utredning

Under detta kapitel beskrivs de hörnstenar i projektet, utan vilka det skulle ha varit svårt eller omöjligt att genomfört detta arbete.

3.1 Schemaläggaren

Schemaläggaren var ursprungligen skrivet i en tidig version av VB men var vid arbetets start uppgraderat till VB.NET. I programmet fanns flera inbyggda funktioner för användaren, såsom beräkning av OB-tider, årsarbetstid etc. Från programmet kunde man även göra ett flertal utskrifter, för att t.ex. få ut arbetsscheman från programmet. Se bilaga 1 för utförligare beskrivning av programmet och dess funktionalitet.

3.2 Visual Basic .NET

.NET är ett koncept som Microsoft tagit fram och som funnits några år. Tanken är att det skulle användas för att sammankoppla information, människor, system och diverse tekniska hjälpmedel. .NET-anslutna lösningar ska, enligt Microsoft [2], vara snabbare och enklare att integrera med befintliga system än andra lösningar, samtidigt som man ska kunna nå information lättare. .NET är en del av Microsofts utbud och som namnet antyder så kretsar mycket runt Internet men .NET innebär ändå inte bara nätlösningar, utan även klientlösningar.

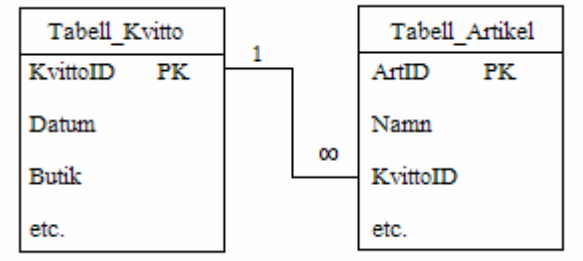
VB.NET kan användas för att programmera klientlösningar, vilket innebär att man skapar program som körs på den lokala datorn. Programmet har ingen anslutning till Internet, utan allt som programmet exekverar och visar, görs på den dator man sitter vid. Ett av argumenten för Visual Basic har enligt Microsoft [3] alltid varit att man lätt kan skapa program med bra användargränssnitt, som kan vara lite svårare i andra programmeringsspråk.

3.3 Databashantering

Schemaläggaren arbetar sedan tidigare mot en databas. Databasen består av ett antal tabeller, som är kopplade till varandra genom ett antal relationer av typen en till många. En relation av typen en till många betyder att en post i en tabell kan referera till flera poster i en annan tabell men inte vice versa. Hade man istället haft en relation av typen en till en så hade det inneburit att en post i en tabell skulle kunna referera till endast en post i den andra tabellen och vice versa. Hur databasen ser ut går att läsa mer om i Bilaga 2.

3.3.1 Primärnyckel

Varje tabell innehåller en primärnyckel. Den användes för att identifiera posterna i tabellen. Primärnyckeln är unik i den tabellen, vilket innebär att det i en tabell inte får finnas två poster med samma primärnyckel. I figur 1 nedan, finns två tabeller med en koppling mellan. KvittoID i Tabell_Kvitto och ArtID i Tabell_Artikel är primärnycklar.



Figur 2. Exempel på koppling mellan tabeller

3.3.2 Referensintegritet

Referensintegritet är en funktion för att skydda databasen mot data som gör att förbindelsen mellan olika poster inte kan bevaras. I det tabellexempel som finns ovan i figur 1, med kvitton och sålda artiklar, innebär referensintegritet att man inte kan registrera en ny såld artikel utan att först ha registrerat ett kvitto med det aktuella KvittoID. KvittoID i Tabell_Artikel måste med andra ord innehålla ett tal som redan finns i KvittoID i Tabell_Kvitto. För att detta ska fungera, måste kopplingen mellan tabellerna vara av typen 1-∞ (en till många), som det är i figuren.

Samma skydd gäller vid borttagning, vilket innebär att man inte kan ta bort en post från Tabell_Kvitto utan att först ha tagit bort de poster från Tabell_Artikel som har samma KvittoID som det man ska ta bort i Tabell_Kvitto.

3.3.3 Hämta data

Data kan hämtas från en databas genom att ställa en fråga skriven i SQL (Structured Query Language), sedan returneras de poster som passar in på frågan. Det går, förutom att hämta data med SQL, även att uppdatera en post, lägga till en ny post samt ta bort en post i en databas med hjälp av SQL.

Beroende på vad man vill göra så ser SQL-frågan annorlunda ut, men nedan i figur 2 visas hur det fungerar att hämta data med hjälp av SQL. SELECT menar att man vill hämta data, stjärnan betyder att alla fält från tabellen ska hämtas, FROM Tabell1 innebär att data ska hämtas från Tabell1 och WHERE antal=3 är villkoret som ska vara uppfyllt, alltså endast poster där fältet antal är lika med 3 ska hämtas. Fältet ID är i tabellerna nedan räknare, vilket innebär att man inte behöver lägga in något i dessa fält själv utan de räknas automatiskt upp med 1.

Databas

Tabell1

ID	namn	antal
1	aa1	2
2	bb2	4
3	aa2	3
4	aa1	3
5	bb1	2
6	bb2	3
7	cc2	6

Tabell2

ID	namn	antal
1	aa1	2
2	bb2	4
3	aa2	3
4	aa1	3
5	bb1	2
6	bb2	3
7	cc2	6

SQL-fråga: SELECT * FROM Tabell1 WHERE antal=3

Returnerar:

ID	namn	antal
3	aa2	3
4	aa1	3
6	bb2	3

Figur 3. Exempel SQL-fråga SELECT

Om man skulle vilja ta bort posten med id=2 i Tabell2, så skulle en sådan SQL-fråga se ut såhär: DELETE * FROM Tabell2 WHERE id=2.

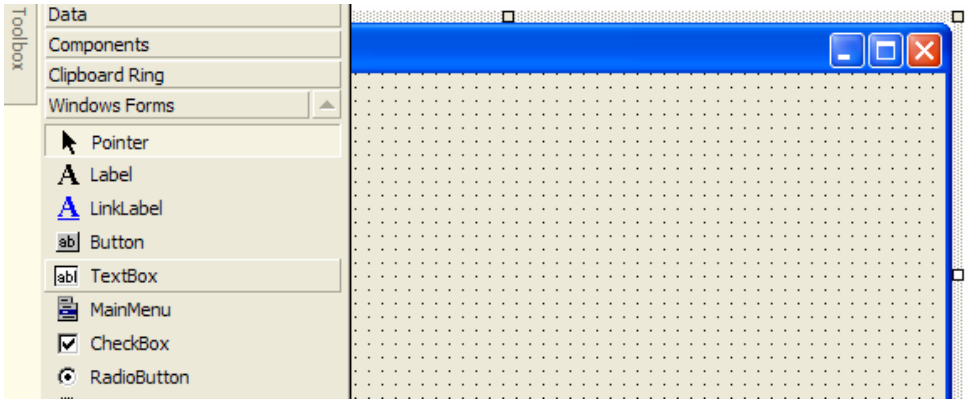
Skulle man istället vilja ändra värdet på namn för posten med id=2, så ser SQL-frågan ut såhär: UPDATE Tabell2 SET namn = "aa2" WHERE id=2.

Och om man till sist istället skulle vilja lägga till en post i Tabell2, så ser SQL-frågan ut såhär: INSERT INTO Tabell2 (namn, antal) VALUES (’bb3’, 3).

3.4 Utvecklingsmiljö

Den utvecklingsmiljö som använts under arbetet är Microsoft Visual Studio .NET 2003. Visual Studio .NET 2003 har utvecklats av Microsoft och det är ett välutvecklat program, med många fördelar. Den erbjuder både bra funktionalitet och lättanvändbarhet, vilket innebär att man oavsett liten eller stor programmeringserfarenhet snabbt kan komma ingång med programmering i programmet. Man får lätt översikt över det program som man ska skapa, och när något fel uppstår så säger Visual Studio till, och man får ändra koden för att ta bort felet. I programmet finns en verktygslåda med alla olika verktyg som man kan använda i ett windowsformulär. Där är det bara att ta tag i det man vill ha med, dra ut den i sitt formulär och släppa den. I figur 3 kan man se hur detta ser ut.

En nackdel med denna utvecklingsmiljö är att den inte är gratis, men som alternativ finns ett flertal gratisprogram med liknande funktionalitet som man kan använda istället. De programmen brukar sällan vara lika användarvänliga, men de har istället fördelen att de är gratis. Ett exempel på ett sådant program är #develop, eller SharpDevelop som det även kallas.



Figur 4. Toolbox (verktygslåda) i Visual Studio .NET 2003

3.5 Resurser

Under arbetets gång har mycket tankar och inspiration hämtats från det ursprungliga programmet. Utöver detta har två programmeringsböcker använts som stöd, och det är ”Visual basic .NET : how to programme”[DH02] och ”Lär dig Visual Basic .NET på 3 veckor”[MD05].

4 Lösningsförslag

Tanken med schemaläggningsguiden var som sagt att det skulle förenkla det ursprungliga programmet, genom att schemaläggningen genomförs automatiskt efter de val som gjorts. Grundidén var att schemaläggningsguiden skulle vara en löstagbar modul som inte skulle vara bunden till det ursprungliga programmet, vilket i så fall skulle innebära att det skulle gå att ”plocka loss” den från programmet och exportera den till annat program om det skulle behövas. Detta har påverkat de val som gjordes under utvecklingen.

4.1 Frågeställningar

I början av rapporten (se 1.7) listades ett antal frågeställningar. Detta avsnitt avser svara på dessa frågeställningar. Här ges svar på hur detta löstes i detta arbete, vilka alternativa lösningar som framkommit samt vilket som vore det mest lönsamma. Vid funderingar kring detta, har hänsyn också tagits till det ursprungliga programmets funktionalitet, samt hur väl integrerad modulen skulle bli med systemet.

4.1.1 Databas

Den lösning som användes i detta arbete var att arbeta med den som det ursprungliga programmet använde, helt utan modifikation. Detta är en enkel lösning, då ingen modifiering av databasen krävdes.

En annan lösning vore att använda samma databas men något modifierad för att få in speciell information om de scheman som skapats. Detta skulle kunna ge mer kraft till modulen, då det t.ex. skulle ge möjlighet att i databasen ange vilka scheman som tillsammans skapar ett helårsschema. Nackdelen med denna lösning vore att programmets nya version inte skulle bli kompatibel med databaser från tidigare versioner. Detta skulle dock kunna lösas genom att databasen kompletteras med de nya tabellerna, om de saknas, när modulen startas. Ännu en lösning vore att istället ha en extra databas bredvid den ursprungliga, men då vore det smidigare att ha all information i samma databas.

Den bästa och mest genomtänkta lösningen av dessa vore att använda en modifierad databas med extra information om de scheman som skapats. Detta skulle kunna ge mer bredd till programmet.

4.1.2 Veckor och helgdagar

I detta arbete löstes detta genom att kopiera de redan skrivna algoritmer från det ursprungliga programmet, och använda dem för att få fram alla veckor och helgdagar för ett år.

Utöver denna lösning fanns tankar på att istället skriva denna algoritm själv, vilket skulle vara något avancerat att få fram en generell algoritm för detta. Ännu en lösning hade varit att direkt göra anrop till de funktioner som fanns, utan att kopiera dessa. Det hade dock blivit lite krångligare, då de funktionerna krävde fler inparametrar och gjorde lite mer än nödvändigt.

Den bästa lösningen av dem som uppkommit i detta fall var den lösning som valdes.

4.1.3 Antal scheman

Då modulen skulle lägga ett flertal scheman för ett flertal anställda så vore det trevligt om antalet scheman kunde ligga så lågt som möjligt. I denna modul skapas först fem scheman (en ledig vardag varje schema, samt lördag ledig) sedan fem scheman till (en ledig vardag, jobba lördag). Dessa scheman var tänkta att kunna täcka i princip hela året. När sedan scheman för veckor med röda dagar skapas, så kontrolleras först om något av de skapade schemana stämmer in och om det inte gör det så skapas istället ett eget schema för den veckan.

En annan lösning var att t.ex. lägga ett schema som man jobbar på hela året, vilket skulle ha inneburit att varje anställd endast skulle ha ett schema kopplat till sig. Sedan skulle programmet själv få kontrollera vilka dagar den anställde borde vara ledig och vid utskrifter och beräkningar ange detta. Om anställd inte skulle jobba någon dag under en vecka, skulle ett speciellt schema skapas för just den veckan. Antal scheman skulle då reduceras, men det skulle behöva en ombyggnad i hela det ursprungliga programmet. Lösningsförslag utöver detta och liknande har dock inte kommit fram under arbetet.

Det mest slagkraftiga lösningsförslaget av dessa vore att skapa ett eller några få scheman som kopplas till en anställd under ett helt år. Detta skulle kunna öka förståelsen hos användaren samtidigt som det skulle förenkla schemaläggningen. Detta skulle ju dock även omfatta en hel ombyggnad av programmet, vilket kan vara ganska omfattande.

4.1.4 Spara

För att kunna spara scheman temporärt, valdes att endast spara data i ett DataSet (en datatyp i .NET som beskriver en databas), och sedan spara all information till databasen om användaren accepterar.

En alternativ lösning vore att, i databasen, lägga till ett fält med information om att posten är lagrad temporärt. Denna information skulle i så fall sparas när användaren accepterar. Om användaren avbryter så tas alla dessa poster bort.

Lösningen som valdes här var nog också den bästa av dem som diskuterats.

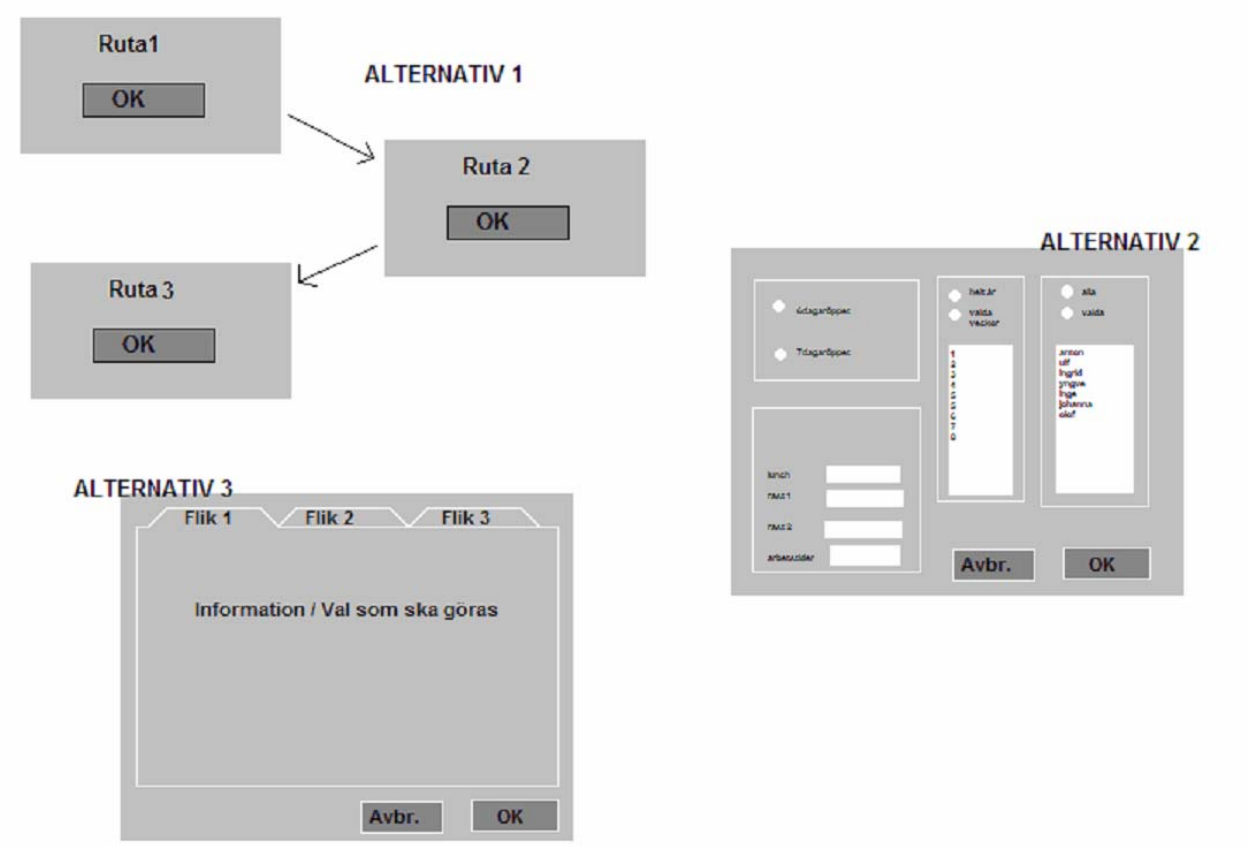
4.2 Utseende och Design

Inmatning
Det finns många sätt att planera designen på, men svårt att få till det bra och lättförstått. I början av utvecklingen jobbades det mest med meddelanderutor, inmatningsrutor och andra rutor som kom upp i tid och otid. Detta blev tidvis ganska jobbigt och irriterande. I allmänhet brukar det uppfattas som irriterande när man får upp en ruta med ett meddelande eller ett val som ska göras och när man stänger rutan så kommer nästa ruta upp.

För att undvika detta kunde man istället ha haft all information och alla val som skulle göras på en enda ruta. Risken med en sådan lösning var att den skulle kunna bli överfull och användaren ändå inte skulle förstå.

En tredje variant skulle ha varit att precis som förra lösningen endast ha en ruta, men uppdelad i flera delar med hjälp av flikar. Ett exempel på program som använder sig av flikar är Microsoft Excel som använder det mellan olika arbetsböcker. Det hjälper till att strukturera programmet, och det blir mer tydligt.

I början av detta arbete användes, som tidigare informerats om, alternativ 1. Men det blev som sagt aningen frustrerande, så en övergång till alternativ 2 blev då aktuell.



Figur 5. Alternativ på utseende och design för inmaning av värden

Visning
För att sedan redovisa de scheman och kopplingar som skapats kan man också ha många olika designer, då det ju ska redovisas på något vis. Även här fanns ett antal lösningar som tänkts igenom, där ett var att försöka visa resultatet i det ursprungliga programmets huvudfönster. Problemet om man skulle göra det, vore om användaren inte skulle vilja spara resultatet av schemaläggningen, då det redan blivit sparat för att kunna visa det i huvudfönstret. Det skulle också ha inneburit att implementeringen av modulen skulle ha blivit djupare, vilket i sin tur skulle innebära att det skulle bli svårare att ”få loss” den.

Istället skulle man ha skapat ett nytt fönster, som skulle se i princip likadant ut som huvudfönstret, och redovisa resultatet där. På så vis skulle modulen varit mindre inbakad i programmet och därmed också lättare att använda i ett annat program. Här valdes att använda det andra alternativet, som använde sig av ett till fönster för att redovisa resultatet.

5 Resultat

Resultatet av detta examensarbete är en modul till programmet Schemaläggaren. Denna modul är ett steg mot ett mer lättanvändligt program, där användare av programmet kan skapa scheman automatiskt istället för manuellt som tidigare. Även om denna modul inte är fulländad ännu så är ändå den på väg åt rätt håll och kan bli en användbar del av Schemaläggaren i framtiden.

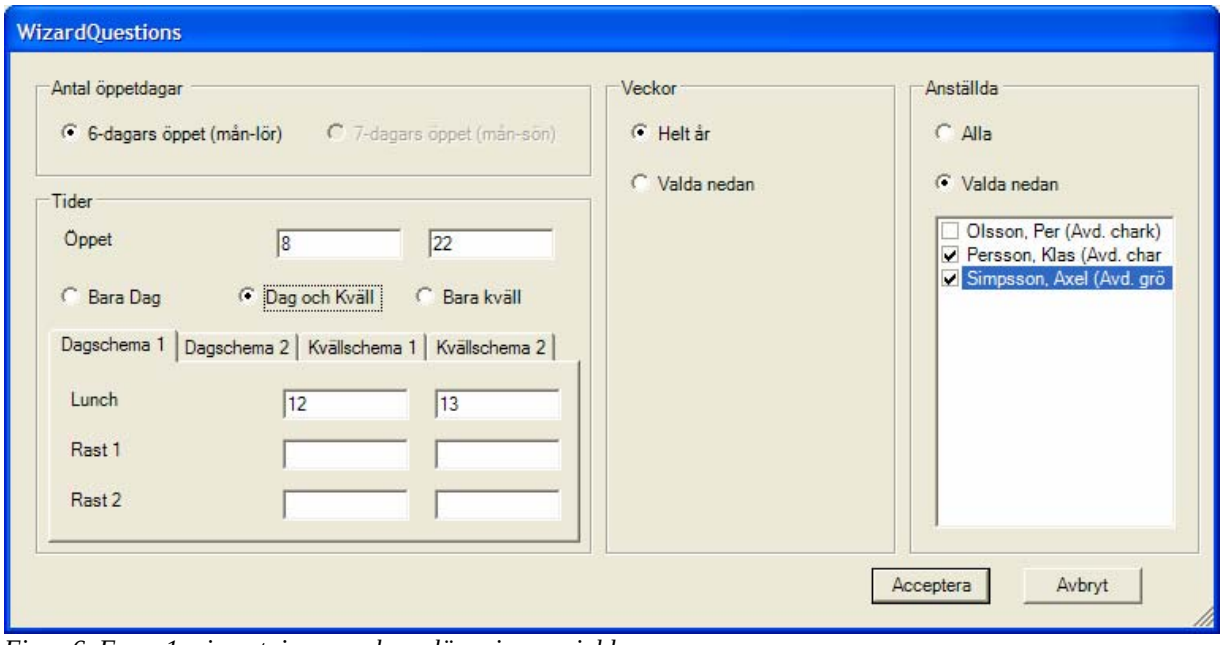
5.1 Utseende

Utseendet och användargränssnittet hos modulen ligger i närhet till hur det ser ut och fungerar i det ursprungliga programmet. Detta för att det inte ska bli någon stor omställning när man startar schemaläggningsguiden istället för att använda programmet som man tidigare gjort.

I VB.NET byggs programmets synliga del upp av en eller flera forms. Varje form är ett windowsfönster, där man kan lägga in textrutor, knappar mm. Man kan sedan koppla olika händelser till varje knapp, lista etc. Schemaläggningsguiden består av två forms, ett för att mata in de värden som behövs för att skapa scheman och ett för att visa de scheman som skapats och hur de kopplats till anställda.

5.1.1 Inmatningsfönster

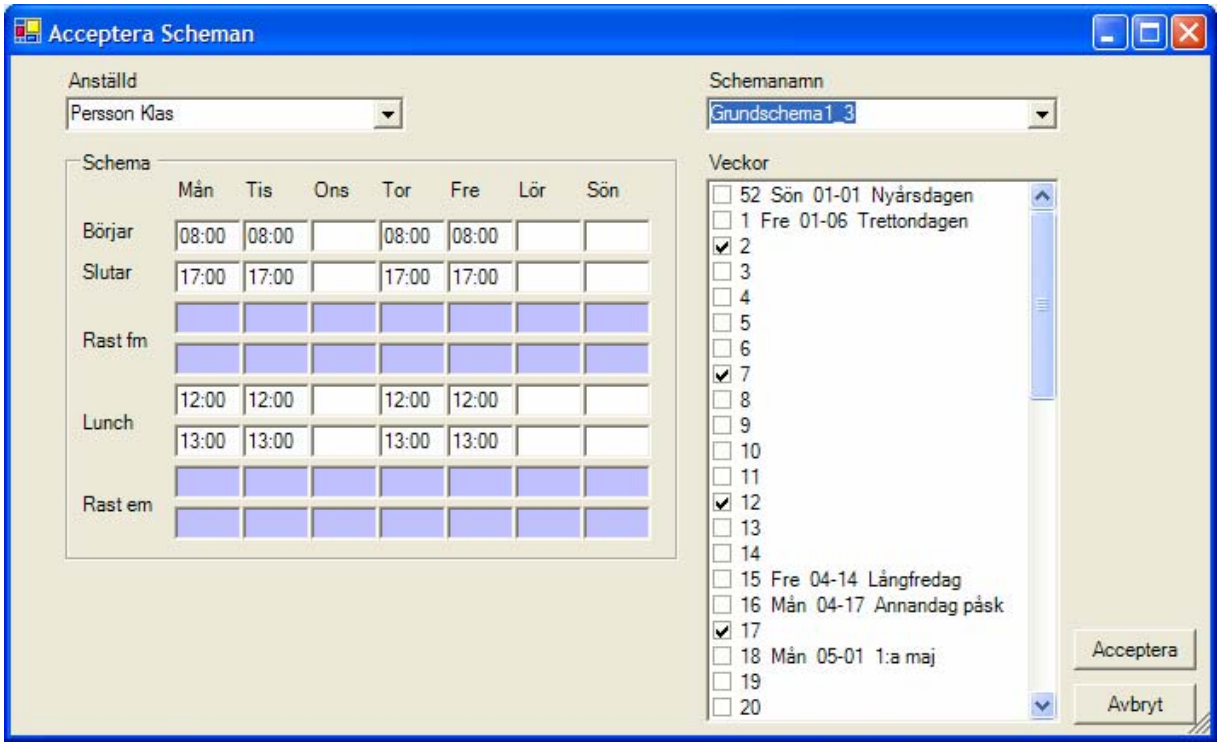
Detta är det första som möter användaren när guiden startas. Guiden startas från huvudprogrammet, som också skickar med de nödvändiga variabler som behövs av guiden. I detta fönster väljer man mellan vilka tider scheman ska läggas, samt hur många scheman som ska skapas. Sedan väljs de veckor som ska schemaläggas (antingen ett helt år, eller de veckor som väljs) samt vilka anställda som ska schemaläggas. Alla val här är dock inte klara, vilket innebär att schemaläggningen kan bli felaktig om man gör något fel. Viss säkerhet finns inbyggd, för att minska risken att felaktiga val görs, men denna säkerhet är dock inte helt färdigutvecklad.



Figur 6. Form 1 – inmatning av schemaläggningsvariabler

5.1.2 Kontrolleringsfönster

I detta fönster visas alla scheman som skapats av guiden. Här visas även de schemakopplingar som skapats. Högst upp till vänster visas en lista med alla anställda, till höger en lista med alla scheman som skapats och under den finns en lista med alla veckor på det aktuella året. I den listan visas de kopplingar som skapats mellan vald anställd och valt schema. De veckor som föregås av en bock är kopplade till det schema som visas för den anställd som är vald.



Figur 7. Form 2 – visning av skapade scheman och kopplingar

5.2 Funktionalitet

Bakom det visuella programmet finns de funktioner som gör allt som ska göras. Då det inte finns möjlighet att redovisa källkoden i denna rapport, kommer istället en beskrivning av systemets funktioner redovisas, samt kommer dess funktionalitet gås igenom. För mer information om programmets ursprungliga funktionalitet, se Bilaga 1.

I tabellen nedan anges de funktioner som skapats under projektet. De funktioner som visas nedan är endast de funktioner som skapats under detta arbete. Vissa av funktionerna gör anrop till funktioner som fanns innan detta arbete startade, men dessa kommer inte redovisas här.

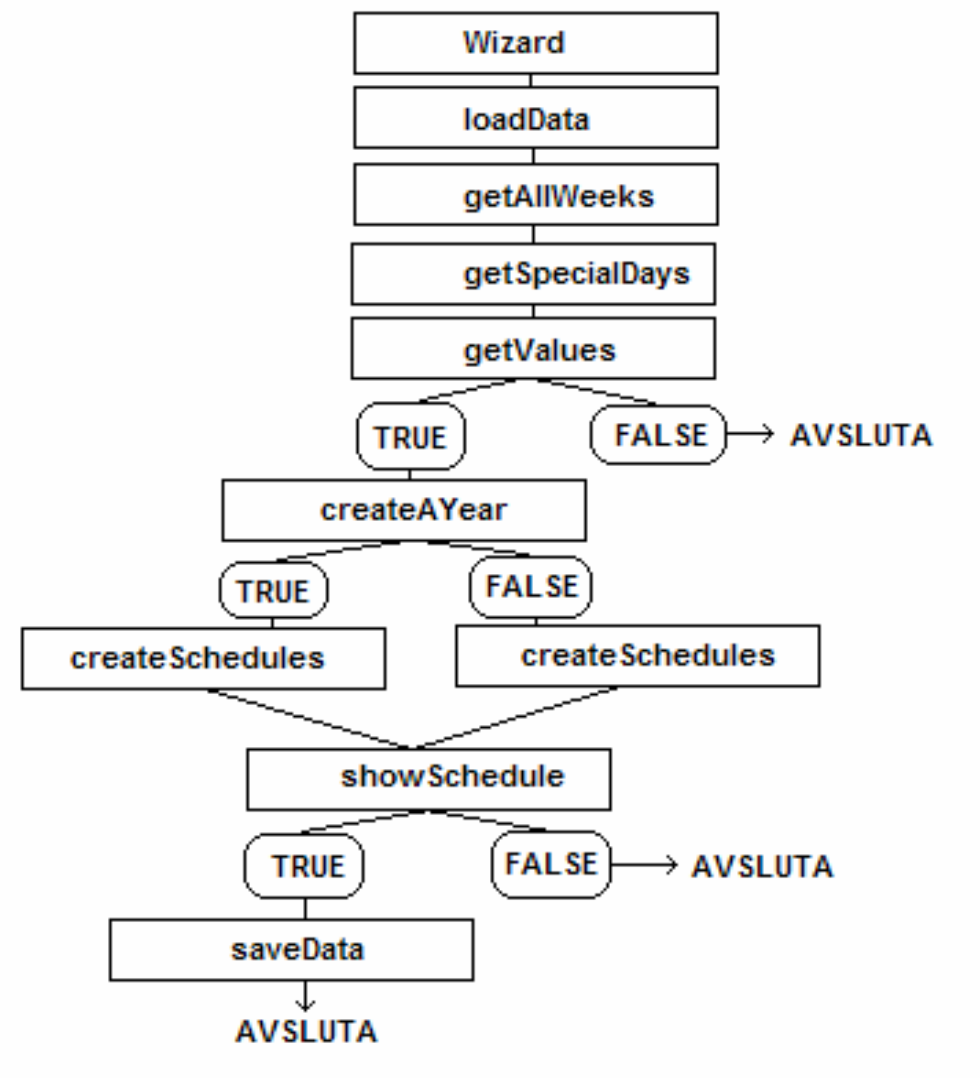
Modulen startas genom att funktionen Wizard anropas. Där anropas funktionen loadData, som laddar all data från databasen till ett DataSet. Efter det anropas getAllWeeks, som hämtar alla veckor på ett år till en array. Därefter anropas getSpecialDays, som hämtar de röda dagar som finns under ett år och sparar denna information i arrayer. Sedan anropas getValues, som i sin tur öppnar en form (Se 5.1.1) där användaren får mata in värden.

Om värdena är korrekt inmatade och användaren accepterar, så returnerar funktionen TRUE, annars FALSE och modulen avslutas. Om TRUE returneras så kontrolleras variabeln

createAYear, vars värde beror på val gjorda vid inmatningen. Variabeln avgör om schema ska läggas för helt år eller för veckor som blivit valda. Därefter anropas createSchedules, som skapar de scheman som ska skapas (Se 5.3).

Om något blir fel under skapande av scheman (något inmatat värde som inte stämmer) så returnerar funktionen FALSE och modulen avslutas. Om inget fel stötes på, så returneras TRUE och showSchedules anropas. Där visas alla scheman i en form (Se 5.1.2). Funktionen returnerar FALSE om användaren väljer att avbryta i detta läge och modulen avslutas då.

Om användaren istället accepterar schemana, så returneras TRUE och funktionen saveData anropas. Den funktionen sparar all data som lagts till under körning. Därefter avslutas modulen. I figur 8 finns detta flödesdiagram illustrerat.



Figur 8. Flödesdiagram för genomförande av guiden

Tabell 1. Funktionsbeskrvning

Funktionsnamn	Invariabler och dess datatyp, beskrivning och returvärde
loadData	Invariabler: fileToRead (String) Funktion: Laddar all information som ligger i databasen till ett DataSet. Returvärde: DataSet
showSchedule	Invariabler: year (String), ds (DataSet) Funktion: Visar de scheman som ligger sparade i ds för aktuellt år (year). Scheman visas en ett eget fönster. Returvärde: Boolean
computeSchedule	Invariabler: ds (DataSet), startAt (String), stopAt (String), lunchStart (String), lunchStop (String), days (String), name (String), startFirstBreak (String), stopFirstBreak (String), startSecondBreak (String), stopSecondBreak (String) Funktion: Skapar en post i tabellen ”Grundschema” där alla kolumner har värden från de invariabler som skickades med vid funktionsanrop. Returvärde: Boolean
getValues	Invariabler: weeks (Integer Array), employees (Integer Array), allWeeks (Integer Array), specialWeeks (Integer Array), specialWeekdayName (String Array), ds (DataSet), startLunch (String Array), stopLunch (String Array), startBreak1 (String Array), stopBreak1 (String Array), startBreak2 (String Array), stopBreak2 (String Array), startWork (String Array), stopWork (String Array), createAYear (Boolean) Funktion: En ny Form visas, där användaren får göra val för att styra schemaläggningsguiden. Användaren får också mata in de tider som är aktuella för schemaläggning. Returvärde: Boolean
getNewStartTime	Invariabler: time (String), diff (Integer) Funktion: time avser den tid då det schema slutar, som behöver den nya starttiden. diff är hur många timmar det ska vara mellan start- och stopptid. Returvärde: String
getNewStopTime	Invariabler: time (String), diff (Integer) Funktion: time avser den tid då det schema börjar, som behöver den nya stopptiden. diff är hur många timmar det ska vara mellan start- och stopptid. Returvärde: String
Wizard	Invariabler: year (String), file (String) Funktion: Startar schemaläggningsguiden. year anger det år som ska schemaläggas, och file avser sökvägen till den databas som ska hämtas och sparas till ett DataSet.

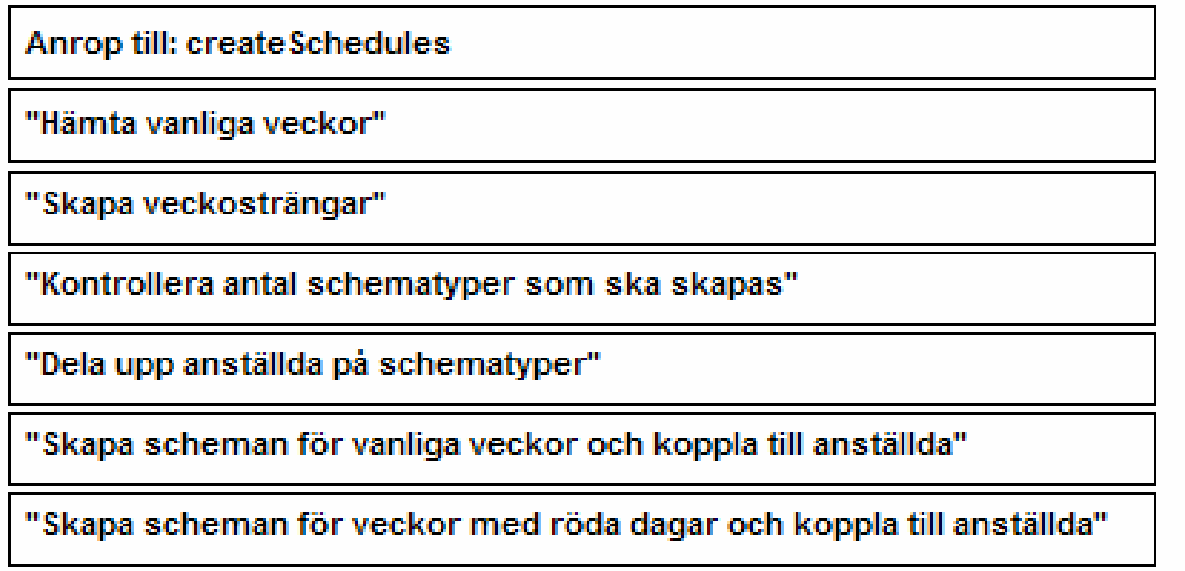
getAllWeeks	Invariabler: year (String), allWeeks (Integer Array) Funktion: Hämtar alla veckor som finns det år som anges (year) och sparar dessa till arrayen allWeeks.
getSpecialDays	Invariabler: year (String), Vnr (Integer Array), Datum (String Array), HNamn (String Array), DagNamn (String Array) Funktion: Alla röda dagar hämtas och dess veckonummer sparas i arrayen Vnr. Helgdagarnas datum sparas i Datum, helgdagens namn i HNamn och i DagNamn sparas veckodagens namn.
createSchedules	Invariabler: year (String), ds (DataSet), weeks (Integer Array), specialDays (String Array), startingTime (String Array), endingTime (String Array), startingLunch (String Array), endingLunch (String Array), startingBreak1 (String Array), endingBreak1 (String Array), startingBreak2 (String Array), endingBreak2 (String Array), employees (Integer Array) Funktion: Skapar scheman för de anställda som finns i arrayen employees med de tider som finns angivna i invariablerna. Scheman sparas i ds.
putScheduleOnEmployee	Invariabler: ds (DataSet), year (String), weeks (String), schedule (String), employee (Integer Array) Funktion: Sparar schema på valda anställda och med valda veckor under valt år i ds.
saveData	Invariabler: ds (DataSet), saveFile (String) Funktion: Alla ändringar som gjorts till ds under modulens körning sparas till filen saveFile.

5.3 Schemaläggningsalgoritm

Flödet för den algoritm som skapades för att generera scheman beskrivs nedan i figur 9. Det första som sker är att funktionen createSchedules anropas. Då kontrolleras först alla veckor på året, och markerar de veckor utan röd dag. Sedan skapas veckosträngar bestående av veckor som tidigare markerats som helgfria. Dessa veckosträngar används sedan vid schemaläggningen.

Efter detta kontrolleras hur många schematyper som ska skapas, maximalt kan detta bli 4 (2 dagsscheman och 2 kvällsscheman). Anställda delas därefter upp på det antal schematyper som tidigare tagits fram. Scheman skapas sedan för de vanliga veckorna, och de anställda som valts i steget innan kopplas till dessa scheman.

Efter detta skapas scheman för de veckor som innehåller röda dagar, vilka också kopplas till de anställda som valts till denna schematyp. De två sista stegen genomförs lika många gånger som antalet schematyper.

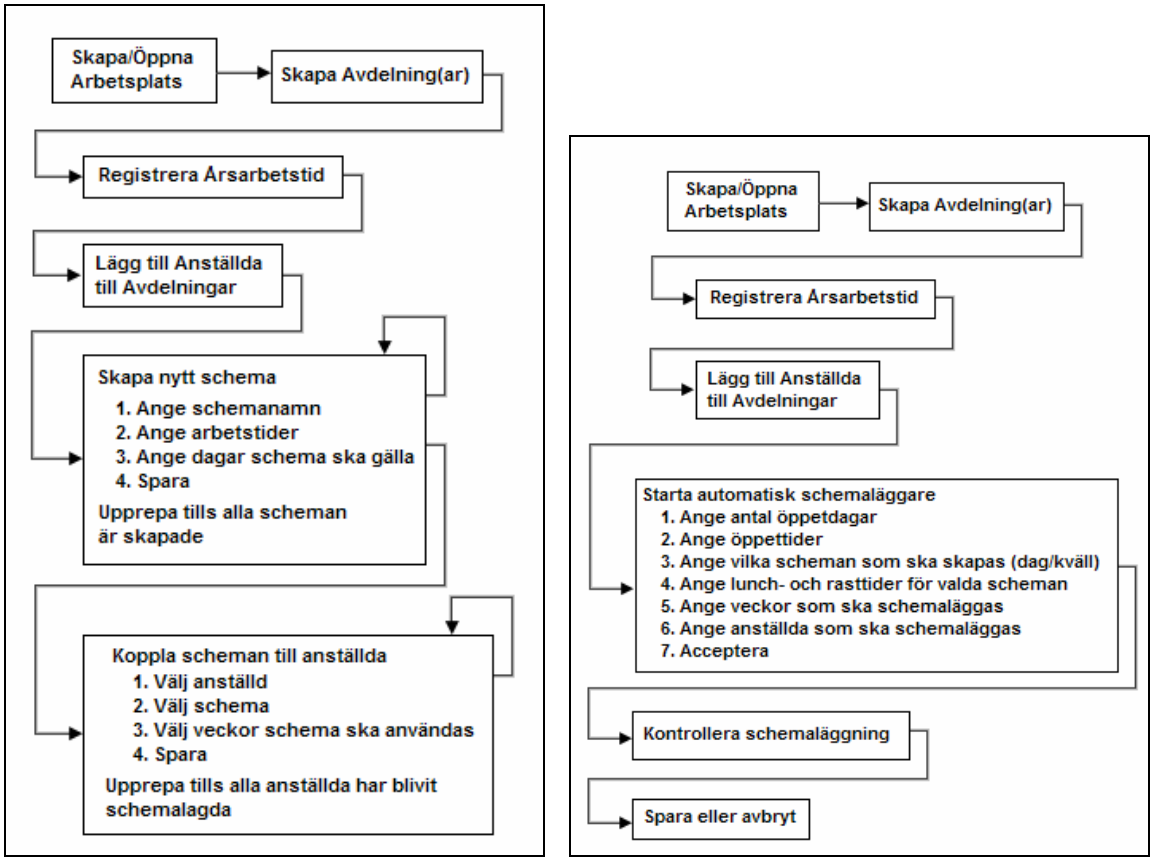


Figur 9. Pseudokod för schemaläggningsalgoritmen

5.4 Programflöde

Då avsikten med detta arbete var att förenkla ett befintligt program, så kommer även flödet i programmet ändras om man använder guiden. I figur 10 visas programflödet för programmet utan schemaläggningsguiden resp. med schemaläggningsguide.

Som synes så är de 4 första stegen lika, och det är steg som måste göras för att kunna skapa scheman. De två sista stegen till vänster är steg som upprepas flera gånger om, vilket gör att det kan bli ett ganska omfattande arbete att göra för hand. Detta undviks vid användning av guiden.



Figur 10. Programflöde utan resp. med guide

6 Slutsatser

I detta kapitel reflekteras över de val som gjorts under arbetets gång, vilka saker man skulle kunna göra bättre samt mina egna tankar om arbetet.

6.1 Syfte och mål

Syftet var att undersöka om det är möjligt att hitta en eller flera algoritmer för att på ett generellt sätt lägga ett arbetsschema. Detta för att förenkla arbetet för användare av programmet Schemaläggaren, och minska det antal timmar ansvarig schemaläggare behöver sitta och arbeta med detta.

Kravet var att det skulle finnas en demonstrationsfärdig modul för programvaran Schemaläggaren. Detta anser jag vara nästan uppfyllt, då jag har en modul som lägger schema och sedan visar schemat för användaren. Modulen tar dock inte hänsyn till alla de regler som finns att tillgå, då alla anställda får liknande scheman, oavsett om de jobbar heltid, deltid eller något annat.

Hänsyn tas inte heller till den årsarbetstid som finns registrerad i programmet, vilket den borde göra för att rätt antal timmar ska registreras på ett helår. Den har heller inte många parametrar att ta hänsyn till, vilket innebär att här finns mer att jobba på.

6.2 Avgränsningar

Det fanns vid arbetets start inga riktiga avgränsningar klara, då det var något oklart hur mycket arbete som skulle ligga bakom funktionen. Desto mer kontroll man fick på vad och hur mycket som skulle göra, så växte också avgränsningarna fram. Det är svårt att redan från början veta hur mycket det är som ska göras, när man aldrig gjort det förut.

6.3 Framtida utveckling

Enligt mig finns stor anledning att utveckla denna prototyp i framtiden. Med hjälp av denna modul, finns chanser att nå nya målgrupper, som inte känner sig så bekväma framför datorn och skulle uppskatta en funktion som denna då programmet blir mer lättanvändbart. Det finns ännu en del saker som saknas eller kan behöva en förbättring, dessa listas nedan.

Deltidsscheman	Modulen tar inte hänsyn till anställdas arbetsgrad, vilket innebär att en deltidsanställd ändå får ett likadant schema kopplat till sig som andra anställda.
Hänsyn till Årsarbetstid	Modulen tar inte hänsyn till årsarbetstid, vilket innebär att de anställda inte får jobba så mycket som de borde.
Speciella omständigheter	Borde finnas möjlighet att välja att en eller flera anställda önskar jobba endast dag eller dylikt. Inget sådant stöd finns för tillfället.
Ny tabell i databas	Man skulle kunna ha en tabell i databasen, som håller koll på vilka scheman som hör ihop. Detta ifall man i ett läge efter att man kört schemaläggningsguiden skapar ny personal som ska använda scheman som skapats tidigare. Detta skulle vara ett alternativ till att man manuellt kopplar scheman till anställd.
Använda guiden flera gånger	Det finns i skrivande stund ingen möjlighet att köra igenom programmet två gånger på rad, då man i det läget skulle få flera scheman med samma namn. Hade man en databas där genererade namn sparades, så behövde man inte få detta problem.
Tydligare instruktioner	I nuläget finns ingen hjälp för användaren som ska använda modulen. Det skulle behövas tydligare instruktioner på alla formulär för att användaren inte ska vara osäker på vad som ska göras. För ökad förståelse kan även krävas viss designändring.
Bättre inmatningssäkerhet	Det finns i nuläget endast liten inmatningssäkerhet, denna borde utökas. Detta för att det inte ska kunna uppstå några problem vid körning av schemaläggningsguiden.
Bara visa kopplade scheman	I nuläget visas alla scheman i visningsformuläret, oavsett vilken anställd som valts. Det innebär att många scheman finns i listan, men bara ett antal som har betydelse för den anställde som valts i listan. Här borde endast de scheman som har koppling till anställd visas.

6.4 Egna reflektioner

Det har varit ett intressant projekt, och man har fått mer insikt i hur det fungerar att arbeta med detta. Då jag tidigare inte programmerat mycket i VB.NET, gick det ganska mycket tid åt i början att introducera sig i språket. Men introduceringen gick ändå ganska snabbt, då man hade tillgång till källkoden till det ursprungliga programmet.

Jag känner att jag personligen borde ha ägnat mig åt rapportskrivning tidigare i processen, då det är en krävande och viktig del av arbetet. Känner ändå att jag haft tillräckligt med tid för att få det gjort, men det hade kanske varit enklare om man jobbat lite parallellt med programmering och rapportskrivning.

En sak som försvårat arbetet något, är det faktum att man inte vet alla regler som finns för arbetstidsscheman. Hade jag kunnat alla dessa regler och riktlinjer, så hade arbetet fortgått smidigare och utan större komplikationer.

I övrigt tycker jag att det har gått bra, och det har varit skönt att ha möjlighet att sitta tillsammans med en annan student som gjorde ett examensarbete mot samma programvara som man kunde prata och diskutera med.

7 Källförteckning

7.1 Elektroniska referenser

[1] Gunnar Ekbrant (2003) Arbetstid – detaljhandeln. Svensk Handel.

[2] (2005) What Is .NET? Microsoft.
< http://www.microsoft.com/net/basics.mspx > 2006-05-21

[3] (2005) .NET Framework Developer Center. Microsoft
< http://msdn.microsoft.com/netframework/ > 2006-05-21

[4] (2003) Iterativ utveckling – om utvecklingsmetoder. Hillar Loor.
< http://www.imcms.net/1126 > 2006-05-27

7.2 Litteratur

[DH02] Deitel, Harvey M. (2002)
Visual basic .NET : how to programme. 2 uppl.
Upper Saddle River, N.J. : Prentice Hall. ISBN - 0-13-029363-6

[MD05] Mackenzie, Duncan (2002)
Lär dig Visual Basic .NET på 3 veckor.
Sundbyberg : Pagina. ISBN - 91-636-0700-X

8 Figurförteckning

9 Ordlista

.NET	.NET är Microsofts senaste synsätt för att bygga program. Det bör poängteras att .NET i sig själv utgör en plattform som egentligen endast finns för Microsoft Windows.
Balanz AB	Ett konsultföretag med kontor i Borlänge. www.balanz.se.
C#	Ett programmeringsspråk lanserat av Microsoft. Bygger vidare på andra C-språk.
DataSet	En datatyp som kan spara en hel databas. I denna kan man spara tabeller från databaser och sedan använda den för att komma åt data utan att hela tiden ha koppling till databasen.
Datatyp	All data som skickas och sparas inom programmet har en viss datatyp. Valet av datatyp avgör hur stor plats data tar i minnet, samt vilken typ av data som kan sparas.
Etex AB	Etex AB är ett företag som ingår i Balanz AB. www.etex.se
Form	En form är den visuella delen av programmeringen. Den kan beskrivas som ett windowsfönster, där man kan lägga in knappar, menyer, textfält och mycket mer.
Handels	Ett fackförbund för handelsanställda. www.handels.se
Klient	Med klient avses en vanlig dator.
Mdb	Filändelse på databasfiler skapade i Microsoft Access.
Microsoft	Microsoft är ett mycket framgångsrikt programvaruföretag. www.microsoft.se
Microsoft Access	Ett program för att bl.a. hantera databasfiler av typen mdb. I programmet kan man skapa databaser, ställa frågor och mycket mer. Säljs av Microsoft.
Microsoft Visual Studio .NET 2003	Ännu ett program som säljs av Microsoft. Detta används för programmering och man kan programmera bland annat VB.NET i denna programvara.
OB-tid	Obekväm arbetstid. Arbetare som jobbar på sådana tider får i allmänhet mer lön.
SQL	Står för Structured Query Language och är ett standardiserat språk för att ställa frågor till databaser och på så vis få fram data ur dem.
VB	Visual Basic är en familj av programspråk definierat av Microsoft.
VB.NET	Ett av de viktigaste språken i Microsofts .NET-plattform. En efterföljare till Visual Basic.
Årsarbetstid	Det antal timmar man ska arbeta på ett helt år. En siffra som ändras varje år.

BILAGA 1

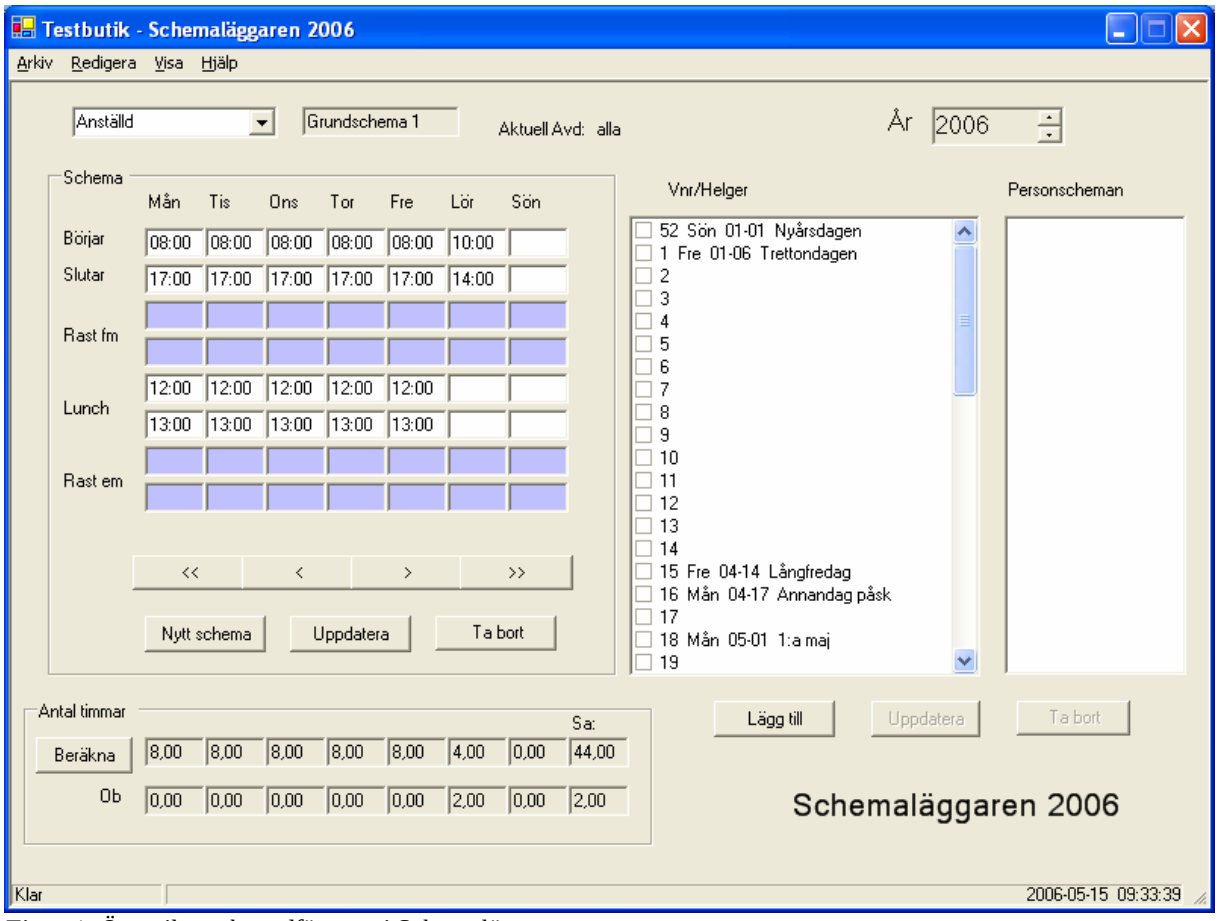
Schemaläggaren

Innehållsförteckning

1 Bakgrund

Schemaläggaren är ett kraftfullt stöd vid schemaläggning, där man kan planera öppettider, raster och med automatik se hur OB-tider etc. påverkar arbetskostnaden. Schemaläggaren är en produkt grundad på Handelsarbetsgivarnas arbetstidsavtal med Handels. Programmet kan användas för att planera veckoarbetstider för individuella personer eller hela arbetsavdelningar.

Schemaläggaren är skriven i Visual Basic .NET kod, och databasen är av typ Microsoft Access.



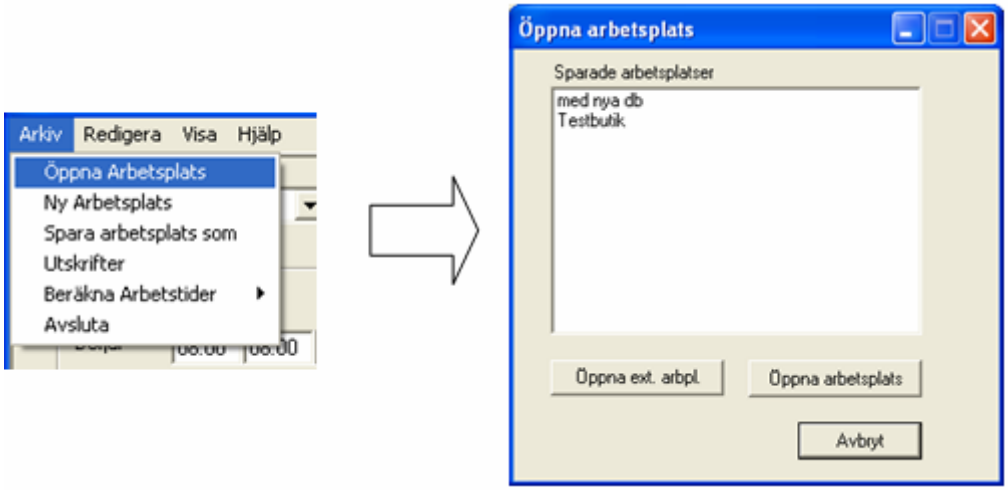
Figur 1. Översikt av huvudfönstret i Schemaläggaren

2 Arbeta med schemaläggaren

För att börja arbeta med schemaläggaren, antingen välj att skapa en ny arbetsplats eller öppna en befintlig.

2.1 Öppna en befintlig arbetsplats

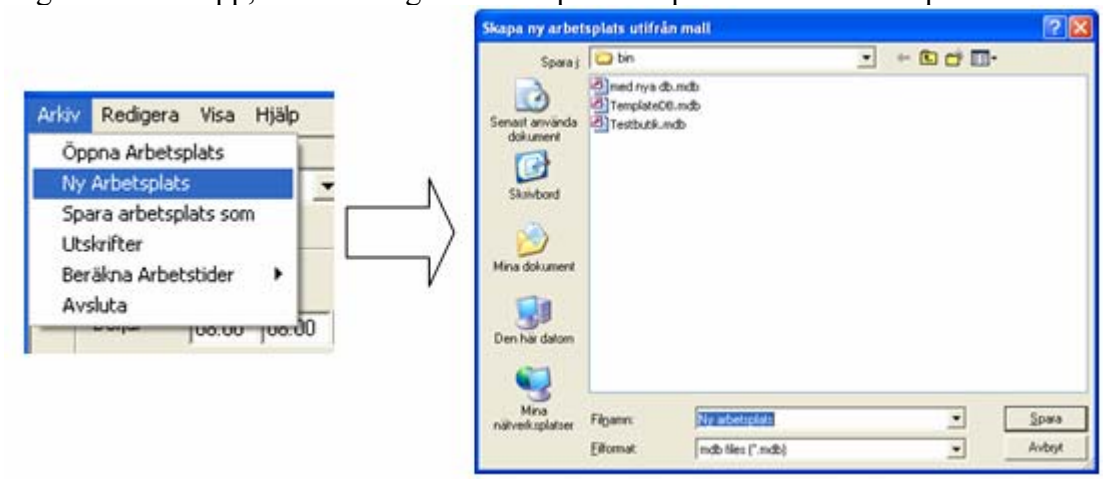
För att öppna en arbetsplats, välj ’Öppna Arbetsplats’ i menyvalet ’Arkiv’. Fönstret till höger kommer då visas, som listar de arbetsplatser som finns sparade i samma katalog som programmet. I listan väljs den arbetsplats man ska arbeta med, och trycker på ’Öppna arbetsplats’. Vill man istället öppna en arbetsplats som är sparad i en annan katalog så trycker man på ’Öppna ext. arbpl.’.



Figur 2. Öppna befintlig arbetsplats

2.2 Skapa en ny arbetsplats

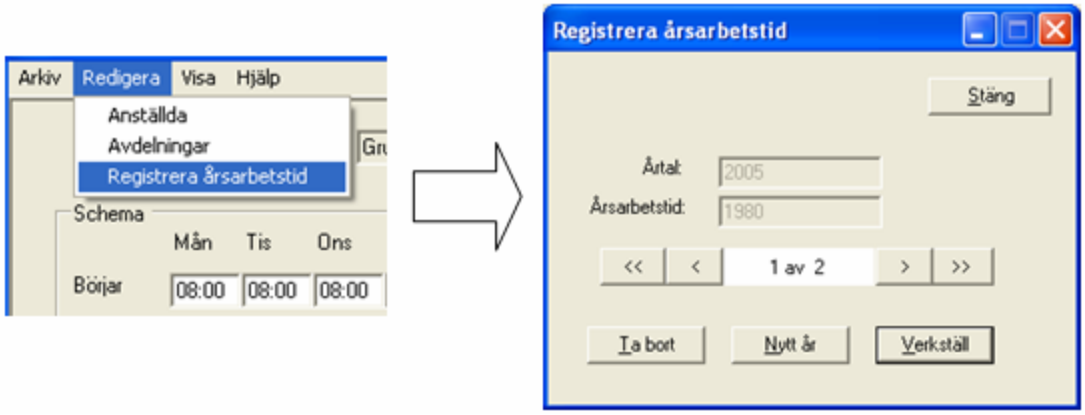
Vill man skapa en arbetsplats så väljer man ’Ny Arbetsplats’ i menyn ’Arkiv’. Rutan till höger kommer upp, där man anger namnet på arbetsplatsen som ska skapas.



Figur 3. Skapa en ny arbetsplats

2.3 Registrera årsarbetstid

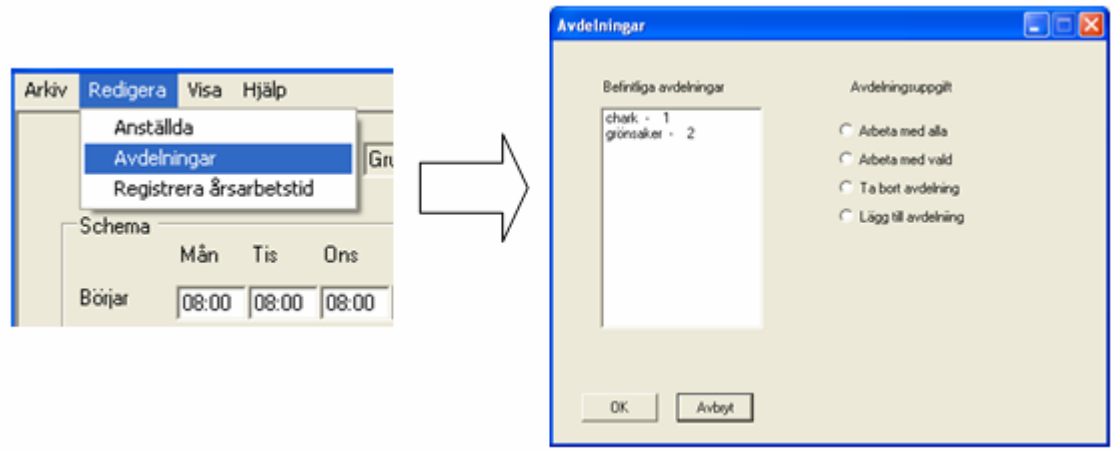
Vill man registrera en årsarbetstid, så väljer man ’Registrera årsarbetstid’ i menyn ’Redigera’. Fönstret till höger kommer upp, där man kan bestämma årsarbetstid för ett nytt år eller ta bort befintliga årsarbetstider.



Figur 4. Registrera årsarbetstid

2.4 Avdelningar

I fönstret avdelningar kan man lägga till eller ta bort avdelningar. För en större arbetsplats kan avdelningar användas för att få en bättre översikt och lättare arbeta med en mindre grupp av anställda. ”Arbeta med alla” betyder att alla anställda finns att arbeta med i huvudfönstret. ”Arbeta med vald” betyder att endast anställda som tillhör vald avdelning kommer att finnas med i huvudfönstret. För att öppna fönstret väljer man ’Avdelningar’ i menyn ’Redigera’.



Figur 5. Skapa, ändra eller ta bort avdelning(ar)

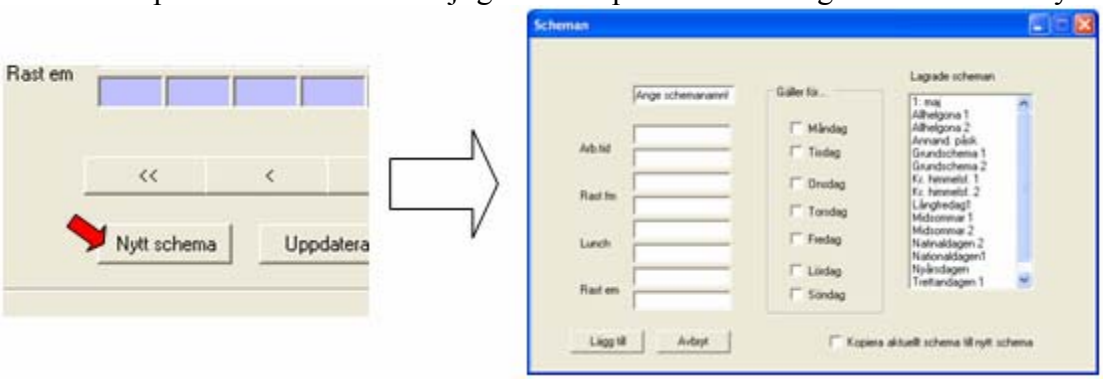
2.5 Anställda

I anställda fönstret kan uppgift om anställda läggas till, ta bort eller ändras. Här kan personuppgifter om anställda ändras, samt deras tjänst i procent, vilken avdelning de tillhör samt OB regler. För att öppna fönstret, välj ’Anställda’ i menyn ’Redigera’.

Figur 6. Ändra, lägg till eller ta bort information om anställda

2.6 Skapa nytt schema

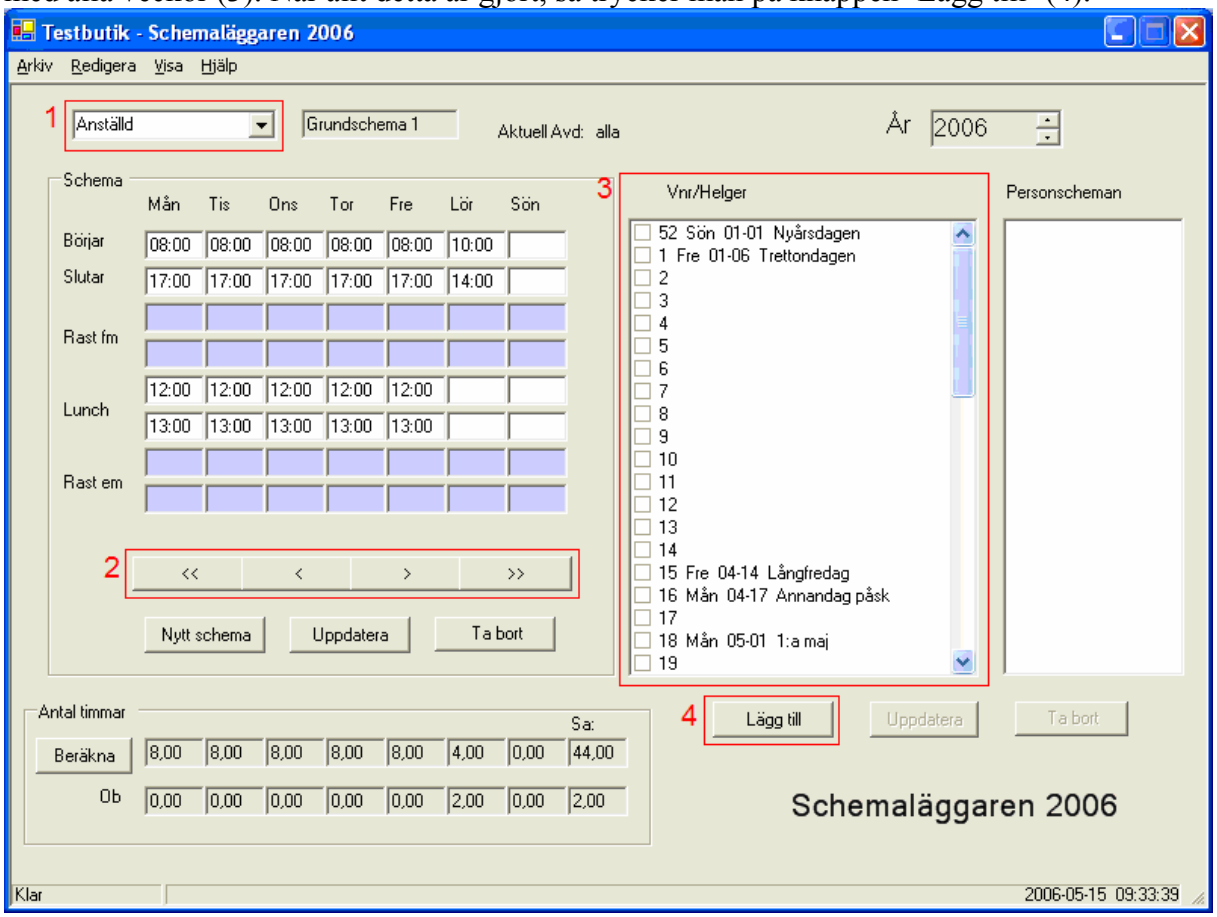
För att skapa ett nytt veckoschema, tryck på ’Nytt schema’ knappen i huvudfönstret av Schemaläggaren. Fönstret till höger visas, där man matar in tider och väljer dagar för schemat som ska skapas. Här finns även möjlighet att kopiera ett befintligt schema till ett nytt schema.



Figur 7. Skapa ett nytt veckoschema/kopiera existerande schema till ett nytt schema

2.7 Koppla schema till anställd

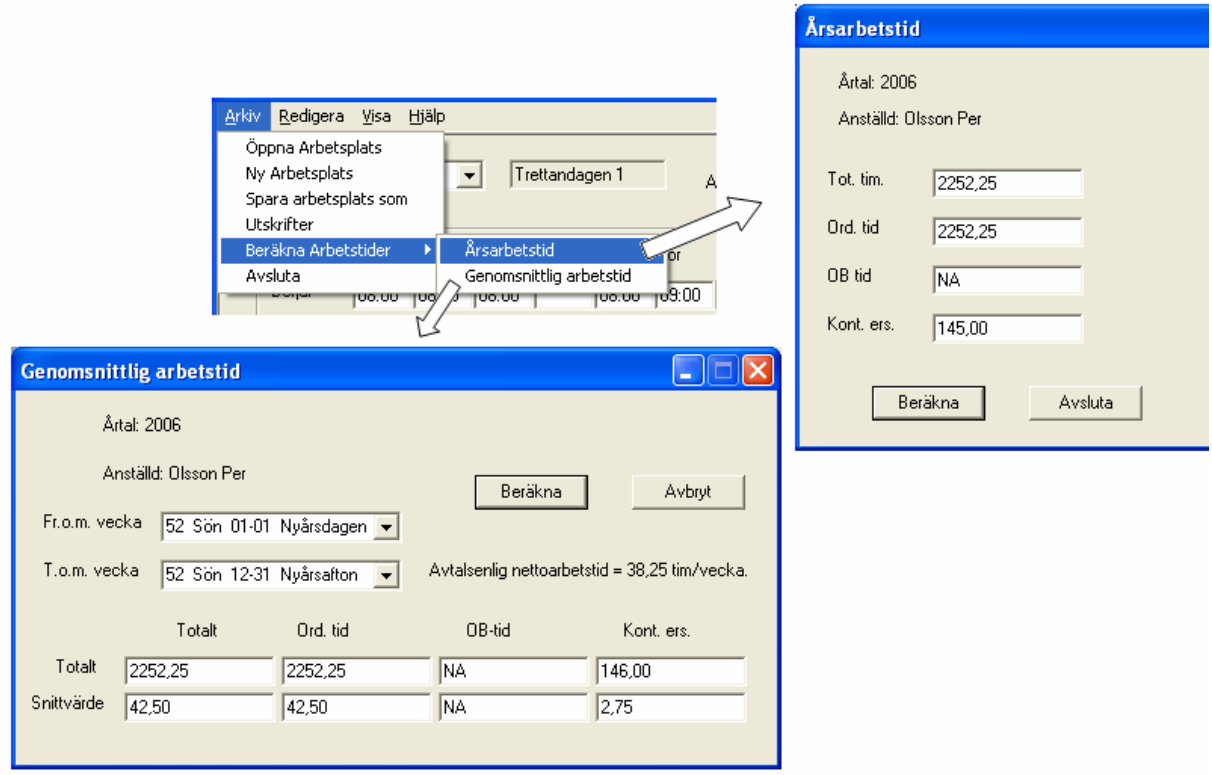
När ett schema ska kopplas till en anställd, så ska detta göras i följande steg. I figur 8 visas vart man ska arbeta. Först ska man välja den anställd man ska jobba med i den listan (1 i figur 8). Sedan bläddrar man fram det schema som ska kopplas till anställd via knapparna under schematiderna (2). De veckor som den anställde ska jobba på detta schema, kryssas för i listan med alla veckor (3). När allt detta är gjort, så trycker man på knappen ’Lägg till’ (4).



Figur 8. Koppla schema till anställd

2.8 Beräkna årsarbetstid/genomsnittlig arbetstid

I programmet finns även möjlighet att beräkna årsarbetstid för en anställd samt se den genomsnittliga arbetstiden för en anställd. Detta görs genom att i menyn ’Arkiv’ öppna undermenyn ’Beräkna Årsarbetstider’ och sedan välja antingen ’Årsarbetstid’ eller ’Genomsnittlig arbetstid’. I fönstret årsarbetstid trycker man på knappen ’Beräkna’ för att beräkna årsarbetstiden och i fönstret genomsnittlig arbetstid väljer man mellan vilka veckor man ska räkna och sedan trycker man på knappen ’Beräkna’.



Figur 9. Menyvalen Årsarbetstid och Genomsnittlig arbetstid

2.9 Utskrifter

Programmet kan generera ett flertal olika utskrifter med all information som finns i programmet. I figur 10 finns ett exempel på en av de utskrifter som kan göras.

Arbetstider

Avdelning: alla **År: 2006**

	Måndag	Tisdag	Onsdag	Torsdag	Fredag	Lördag	Söndag
Vecka:1							
Klas Persson	**Schema:** Trettandagen 1						
	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	-	08:00 - 17:00	09:00 - 14:00	-
Per Olsson	**Schema:** Trettondagen						
	09:00 - 18:00	09:00 - 18:00	09:00 - 18:00	-	09:00 - 18:00	09:00 - 14:00	08:00 - 12:00
Vecka:2							
Per Olsson	**Schema:** Trettandagen 1						
	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	-	08:00 - 17:00	09:00 - 14:00	-
Axel Simpsson	**Schema:** Trettondagen						
	09:00 - 18:00	09:00 - 18:00	09:00 - 18:00	-	09:00 - 18:00	09:00 - 14:00	08:00 - 12:00
Vecka:3							
Klas Persson	**Schema:** Grundschema 1						
	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	10:00 - 14:00	-
Per Olsson	**Schema:** Grundschema 2						
	09:00 - 18:00	09:00 - 18:00	09:00 - 18:00	09:00 - 18:00	09:00 - 18:00	09:00 - 14:00	-
Axel Simpsson	**Schema:** Grundschema 1						
	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	08:00 - 17:00	10:00 - 14:00	-

Figur 10. Exempel på utskrift av arbetstider

3 Figurförteckning

BILAGA 2

Beskrivning av databasen

Innehållsförteckning

1 Inledning

Då databasen spelar en ganska central roll i programmet (ingen möjlighet att spara utan användning av databasen), kan det vara ganska nyttigt att ha insikt i hur databasen är uppbyggd med tabeller och diverse kopplingar. Noterbart är också att inga ändringar har gjorts i databasen från dess ursprungliga utseende, utan den ser precis likadan ut som innan.

2 Databasen

Databasen är uppbyggd av 6 tabeller som alla har med programmet att göra. Varje skapad databas ska illustrera en arbetsplats. Nedan finns alla tabellerna beskrivna samt en liten figur som visar vad de innehåller för fält.

2.3 Beskrivning av databas

Nedan finns alla tabeller i databasen angivna. Där anges namn på tabell, tabellens användningsområde samt fältens namn och datatyp.

2.3.1 Anställda

Tabellen Anställda innehåller information om anställda på arbetsplatsen som skapats. Dessa är valbara i programmet, och kan schemaläggas.

Fältnamn	Datatyp
AnstallningsID	Räknare
EfterNamn	Text
ForNamn	Text
Adress	Text
PostNummer	Text
Ort	Text
HemTelefon	Text
MobilNummer	Text
Arsarbetstid	Text
Anmarkning	Text
AvdelningId	Tal
Erskod	Tal

Figur 1. Tabell anställda

2.3.2 Arsarbetstid

För varje år finns ett visst antal timmar man ska jobba. Detta kallas årsarbetstid, och i denna tabell lägger man via programmet in året och årsarbetstiden för det aktuella året.

Fältnamn	Datatyp
Ar	Text
Arsarbetstid	Text

Figur 2. Tabell arsarbetstid

2.3.3 Avdelning

I tabellen Avdelning finns alla avdelningar som skapats i programmet.

Fältnamn	Datatyp
AvdelningId	Räknare
Avdelning	Text

Figur 3. Tabell avdelning

2.3.4 Grundschema

Tabellen Grundschema innehåller alla scheman som skapats i programmet. Här finns information om schemats namn samt tider för alla dagar som är aktuella.

Fältnamn	Datatyp
GrundSchemaID	Räknare
Schemanamn	Text
BM	Text
SM	Text
BRFM	Text
SRFM	Text
BLM	Text
SLM	Text
BREM	Text
SREM	Text
BTi	Text
STi	Text
BRFTi	Text
SRFTi	Text
BLTi	Text
SLTi	Text
BRETi	Text
SRETi	Text
BO	Text
SO	Text
BRFO	Text
SRFO	Text
BLO	Text
SLO	Text
BREO	Text
SREO	Text
BTo	Text
STo	Text
BRFTo	Text
SRFTo	Text
BLTo	Text
SLTo	Text
BRETo	Text
SRETo	Text

Fältnamn	Datatyp
BF	Text
SF	Text
BRFF	Text
SRFF	Text
BLF	Text
SLF	Text
BREF	Text
SREF	Text
BL	Text
SL	Text
BRFL	Text
SRFL	Text
BLL	Text
SLL	Text
BREL	Text
SREL	Text
BS	Text
SS	Text
BRFS	Text
SRFS	Text
BLS	Text
SLS	Text
BRES	Text
SRES	Text

Figur 4. Tabell grundschema

2.3.5 PersonArsArbetstid

I tabellen finns information om anställdas antal timmar, OB-tid etc.

Fältnamn	Datatyp
PersonArsArbetstidID	Räknare
AnstallningsID	Tal
Ar	Text
TotTim	Text
OrdTid	Text
OBTid	Text
KontErs	Text

Figur 5. Tabell personarsarbetstid

2.3.6 Personschema

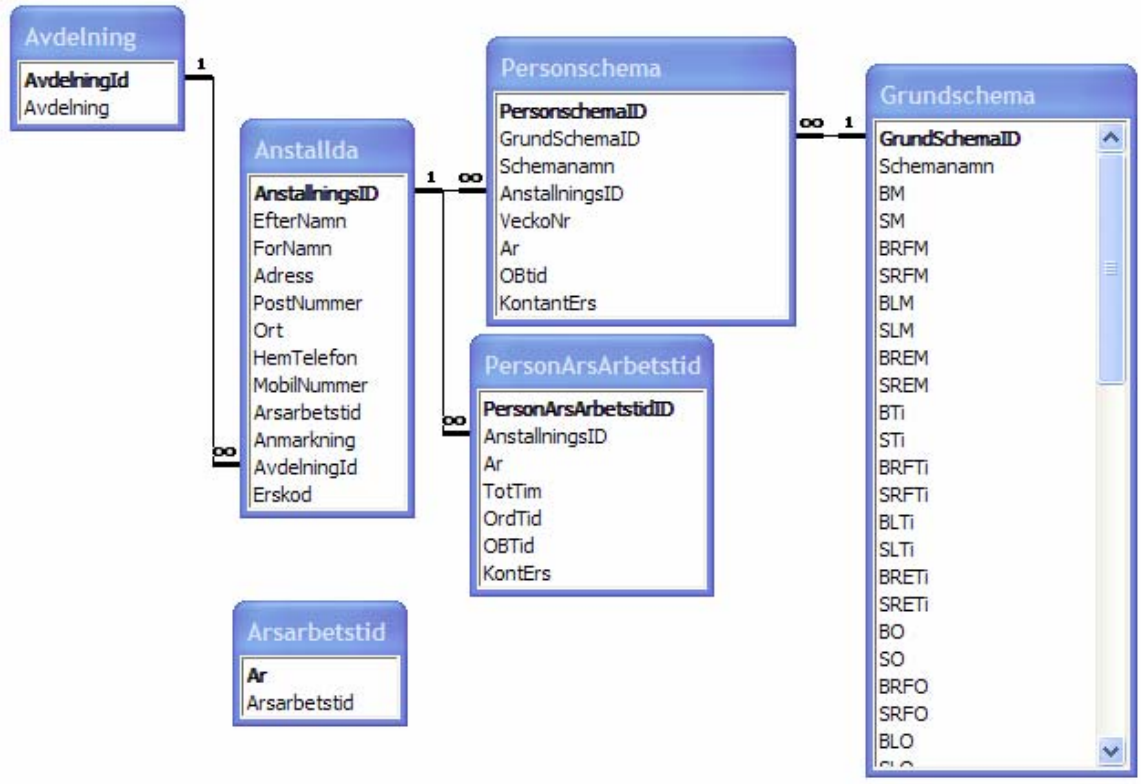
Denna tabell innehåller alla personscheman. Här finns alla scheman som är kopplade till anställda.

Fältnamn	Datatyp
PersonschemaID	Räknare
GrundSchemaID	Tal
Schemanamn	Text
AnstallningsID	Tal
VeckoNr	Text
Ar	Text
OBtid	Tal
KontantErs	Tal

Figur 6. Tabell personschema

2.3.7 Kopplingar i databasen

Databasen är uppbyggd av ett flertal kopplingar för att ha kontroll på hur allt hänger ihop. I figuren nedan visas hur dessa kopplingar ser ut. Alla kopplingar är av typen ”1-många”, vilket innebär att en post i en databas kan peka på många poster i den andra, men inte tvärtom.



Figur 7. Kopplingar mellan alla tabeller

3 Figurförteckning