VAASAN YLIOPISTO
TEKNILLINEN TIEDEKUNTA
TIETOTEKNIIKAN LAITOS

Kenneth Norrgård

OBJEKTORIENTERAD ANALYS MED UML: ETT PRAKTIKFALL

KTM, tietotekniikka
Pro Gradu -tutkielma

VAASA 2008

INNEHÅLLSFÖRTECKNING Sid

Begrepp och förkortningar

Begrepp	**Definition**
Aktör	En aktör är oftast en fysisk person men kan även vara ett angränsande system som kommunicerar med datasystemet, d.v.s. ger eller får information ur systemet.
Användnings-fall	Ett användningsfall är en beskrivning av det som en aktör gör i verksamheten. Det består av en samling aktiviteter som bildar en funktionell helhet som logiskt hör ihop.
Användnings-område	Den organisation som använder datasystemet och administrerar, övervakar eller styr ett problemområde (Mathiassen m.fl. 2001).
Arkitektur	Den struktur, bestående av olika logiska komponenter, som datasystemet är uppbyggd av. En vanlig arkitekturlösning består av tre komponenter: en modell-, en funktions- och en gränssnittskomponent. Även den tekniska plattformen kan ses som en del av systemarkitekturen.
Artefakt	Information som används eller skapas som en del av mjukvaruutvecklingen. Det kan vara ett dokument, en modell eller fråga om programkod. Man skulle generellt kunna säga att allt som bör ligga under versionshantering är en artefakt (Fägnell Urban 2003).
Attribut	Beskriver en egenskap som ett objekt har. Objektet ”Anders” som tillhör klassen ”Anställd” kan ha attribut som anställningsnummer, efternamn, förnamn, avdelningsnummer och löneklass.
Beskrivnings-teknik	Beskrivningsteknik är ett regelverk för hur man gör en beskrivning och hur man visualiserar och dokumenterar en del av verkligheten i en beskrivning. En beskrivningsteknik kan användas i flera olika metoder.
Beteende	Med beteende avses de aktiviteter eller arbetsuppgifter som en aktör eller ett objekt kan utföra i datasystemet. En aktörs beteende kan ses som det som personen gör i sitt arbete medan ett objekts beteende realiseras i klassens metoder.
Datasystem	En samling komponenter som realiserar krav på modell, funktioner och gränssnitt (Mathiassen m.fl. 2001).
Dialog	Med dialog avses den interaktion som användaren för med datasystemet då han/ hon vill ha något uträttat. Dialogen ligger till underlag för planeringen av gränssnittet.

(forts.)

ER -schema	ER står för Entity Relationship. ER-schema är en grafisk presentation av väsentliga självständiga begrepp och visar hur de är relaterade till varandra. Schemat ligger till underlag för en relationsdatabas där tabeller sammankopplas via deras nyckelfält.
Funktion	Med funktion avses här ett antal händelser som knyts ihop till ett händelseförlopp som bildar en logisk helhet. En funktion kan vara organisatorisk och finnas i verksamheten eller programteknisk och finnas i källkoden.
Gränssnitt	Med gränssnitt avses den delkomponent i ett system som ansvarar för att en användare eller ett annat system kan kommunicera med datasystemet (ge in data, bearbeta data och/ eller ta ut data)
Händelse	En händelse definieras som en momentan (”just nu, just här”) tilldragelse som involverar ett eller flera objekt. Tilldragelsen/ händelsen borde vara odelbar och man bör kunna konstatera att den skett. Händelsen kan vara organisatorisk och hända i verksamheten eller programteknisk och finnas i källkoden. En händelse dokumenteras verbalt i förfluten form, t.ex. ”produkt beställd”, faktura betald”, ”kund uppdaterad”, ”datum uppdaterat” och ”rapport utskriven”. Man kan se en händelse som en impuls som ges till systemet och som systemet förutsätts agera på. En händelse (event) är en form av signal till systemet på att någonting har hänt, eller en begäran som systemet skall reagera på. Det kan vara fråga om alltifrån att en användare klickat på musen, till att en tung bakgrundbearbetning har avslutats. Händelser kan genereras externt ifrån en användare via ett specificerat gränssnitt, eller internt inom ett system.
Klass	I en klass beskrivs vad dess objekt skall lagra för information (via sina attribut), vilka andra objekt de skall känna till (via sina relationer) och hur de skall fungera (via sina operationer). Varje klass har ett väldefinierat ansvar i ett datasystem. En klass samlar en mängd relaterade attribut och operationer. Ett viktigt kännetecken för en klass är att dess inre struktur inte är tillgänglig utanför klassen utan kan enbart manipuleras genom ett specifierat gränssnitt. Objekt är förekomster (instanser) av klasser.
Metod	En detaljerad beskrivning av sättet att lösa ett problem. En metod är mycket mer detaljerad än en modell och kan användas flera gånger i en modell.

(forts.)

Modell	En modell är en översikt över utvecklingsarbetet och beskriver i grova drag vilket arbete som måste utföras och vem som bör utföra det.
Multiplicitet	Multiplicitet uttrycker hur många instanser av en klass som kan relatera till en enstaka instans av en associerad klass. Multiplicitet uttrycks som en mängd icke-negativa tal, exempelvis: noll eller en, två till och med fyra, fem, eller många. Ofta uttrycks multipliciteten i bägge riktningarna: en-till-en, en-till-många, eller många-till-många. När multiplicitet anges så begränsas och förtydligas relationerna i en objektorienterad modell. Enligt samma principer kan man uttrycka relationen mellan tabellrader i tabellerna i ett ER-schema.
Objekt	Inom objektorientering är objekt det centrala. Ett objekt är en företeelse med egen identitet, egna egenskaper och eget beteende. Ett objekts egenskaper (attribut) och metoder (operationer) beskrivs i dess klassbeskrivning.
Operation	En operation är en funktion som är möjlig att utföra på objekt av en klass. Operationerna kan liknas vid funktioner. Operationerna har tillgång till attributen i klassen, och kan därmed både läsa och förändra dem. När en klass definieras/ deklareras specificeras vilka attribut och operationer som ingår. Varje operation definieras med ett protokoll (returtyp och parameterlista) och en implementation (exekveringsbar kod).
Problem-område	Den del av omgivningen som administreras, övervakas eller styrs med hjälp av ett datasystem (Mathiassen m.fl. 2001).
Stereotyp	Stereotyper används för att förändra eller specificera inne-börden av ett befintligt UML -element. Dubbelhakar anger att det är fråga om en stereotyp, t.ex. «boundary». Alternativt kan en grafisk symbol användas för att visualisera stereotypen.
System-definition	En kort och koncis sammanfattning av de kriterier som man ställer på det planerade datasystemet
Treskikts-arkitektur	Datasystemet indelas i tre skikt som bildar egna logiska komponenter i den slutgiltiga systemlösningen. Gränssnittskomponenten ansvarar för att användare (och andra angränsande system) skall ha tillgång till systemets funktioner och data. I funktionskomponenten finns systemets regelverk/ funktionalitet och där sker all bearbetning och läsning och/ eller uppdatering av data. Modellkomponenten ansvarar för att all data finns lagrat och är tillgängligt.

(forts.)

Teknisk plattform	Med en teknisk plattform avses den IT-infrastruktur som ett datasystem skall implementeras i. Infrastrukturens komponenter kan delas in i hårdvara, mjukvara, databaser och olika kommunikationsteknologier.
UML	Unified Modeling Language är en beskrivningsteknik för att definiera, visualisera och dokumentera analys och design av ett datasystem.
VATOFA-kriteriet	VATOFA står för: Villkor; Användningsområde; Teknologi; Objekt; Funktionalitet; Ansvar. Det är frågan om en första systemdefinition där man lägger upp kriterier för det planerade datasystemet (Mathiassen m.fl. 2001). Systemdefinitionen kan sedan diskuteras i projektet med olika intressenter för att se om den stämmer överens med vad som förväntas av det kommande datasystemet.
Winha	System för studerandeadministration vid Vasa yrkeshögskola

VASA UNIVERSITET
Tekniska fakulteten
Gjord av: Kenneth Norrgård
Titel: Objektorienterad analys med UML: ett praktikfall
Handledare: Matti Linna
Examen: Ekonomiemagister
Institution: Institutionen för datavetenskap
Ämne: Datavetenskap
Studierna börjat år: 1998
Färdigställd år: 2008 **Antal sidor:** 61

SAMMANFATTNING:

Denna avhandling handlar om objektorienterad analys med syfte att testa användbarheten av beskrivningstekniken UML. Arbetet tar inte upp de mest grundläggande aspekterna utan utgår ifrån att läsaren redan är förtrogen med objektorienterade metoder och har erfarenheter av programutvecklingsprojekt. Genom ett praktikfall visas hur olika UML-diagram används för att modellera ett datasystem. Föremål för analysen är ett system för att planera arbetstidsfördelningen för lärare vid Vasa yrkeshögskola. Arbetet visar analysen steg för steg och resultatet är ett analysdokument som modellerar den tänkta lösningen.

I den senare delen av arbetet får analysen fungera som utgångspunkt för diskussionen om användbarheten av UML. Arbetet kan via sina konkreta exempel fungera som mall för andra liknande projekt. Arbetet utgår från antaganden att UML är lättanvänt och att beskrivningstekniken ger bra underlag för att modellera en systemlösning. Diskussionen visar att UML är ett flexibelt och användbart beskrivningsspråk.

NYCKELORD:
Objektorientering, beskrivningsspråk, modellering, programutveckling

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Kenneth Norrgård
Tutkielman nimi: Oliopohjainen analyysi ja UML: käytännön esimerkki
Ohjaajan nimi: Matti Linna
Tutkinto: Kauppatieteiden maisteri
Laitos: Tietotekniikan laitos
Oppiaine: Tietotekniikka
Opintojen aloitusvuosi: 1998
Valmistusvuosi: 2008 **Sivumäärä:** 61

TIIVISTELMÄ:

Tutkielma käsittelee oliopohjaista analyysiä ja tavoitteena on käytännössä testata UML-kuvauskielen käytettävyyttä. Työssä ei käsitellä oliopohjaisien menetelmien alkeita, vaan oletuksena on, että lukijalla on jo kokemusta sekä oliopohjaisista menetelmistä että ohjelmistoprojekteista. Käytännön esimerkeillä näytetään miten kuvauskieltä Unified Modeling Language käytetään tietosysteemin mallinnuksessa. Analyysin tavoite on mallintaa tietosysteemiä opettajien työaikasuunnittelulle Vaasan ammattikorkeakoulussa. Analyysi viedään läpi askel askeleelta ja tulos on systeemin määrittelydokumentti.

Analyysin tulokset toimivat työn jälkimmäisessä osassa lähtökohtina keskustelulle UML-kuvauskielen käytettävyydestä. Työtä voidaan käytännön esimerkkien ansiosta käyttää mallina vastaaviin projekteihin. Tutkielmaa aloittaessa olettamuksena oli, että UML on kuvauskielenä helppokäyttöinen ja antaa hyvää pohjaa tietosysteemin mallintamiselle. Yhteenveto osoittaa, että UML on hyvin käyttökelpoinen kuvauskieli.

AVAINSANAT:
Oliokeskeisyys, kuvauskielet, mallintaminen, ohjelmistokehitys

UNIVERSITY OF VAASA
Faculty of technology
Author: Kenneth Norrgård
Topic of Master´s Thesis: Object Oriented Analysis using UML: a Case Study
Instructor: Matti Linna
Degree: Master of Science in Economics and Business Administration
Department: Department of Computer Science
Major Subject: Computer Science
Year of Entering the University: 1998
Year of Completing the Thesis: 2008 **Pages:** 61

ABSTRACT:

This thesis is about object oriented analysis and the usability of the UML as a modeling language. The work does not discuss the most basic aspects of object oriented methods, on the contrary the presumption is that the reader is familiar with object oriented methods and has some experiences with software projects. Different UML diagrams are used in a case study to define and document a software system. The object system for the analysis is a software package for planning and administrating the teaching work time for teachers at Vaasa Polytechnic. The work shows how the analysis is realized step by step. The result is a document showing the specifications for the solution.

In the latter part the results of the analysis serve as a starting point for the discussion about the usability of the Unified Modeling Language. Thanks to the usage of simple examples the result can be used as a model for similar software projects. The presumptions stated in the beginning are that the UML is easy to use and that the UML diagrams can form the basis of modeling the software system. The summary shows that the Unified Modeling Language is very useful.

KEYWORDS:
Object Orientation, Modeling Language, Modeling, Software Development

1. INLEDNING

Då det blev aktuellt att välja ämne för min avhandling kändes det naturligt att den skulle handla om systemplanering och objektorienterade metoder. Jag undervisar själv i systemplanering och projektledning vid Vasa yrkeshögskola och har erfarenhet av olika systemplaneringsmetoder. När jag för ett antal år sedan ville förnya min undervisning och kom i kontakt med den objektorienterade filosofin blev jag övertygad om att detta tankesätt var överlägset de traditionella strukturerade planeringsmetoderna. Syftet med denna avhandling är att bl.a. peka på fördelarna med en objektorienterad metod och att diskutera användbarheten av de beskrivningstekniker som används vid analysen. Min förhoppning är att detta arbete via praktiska exempel skall visa på användbarheten av objektorienterade metoder och tekniker.

Denna avhandling handlar alltså om systemutveckling och objektorientering och diskuterar utifrån ett praktikfall användbarheten av beskrivningstekniken UML. Arbetet tar inte upp de mest grundläggande aspekterna utan utgår ifrån att läsaren känner till objektorientering och har erfarenhet av programutvecklingsprojekt. Det är viktigt att skilja på begreppen filosofi, metod och teknik och det är bra att här inledningsvis förklara innebörden av dessa tre begrepp.

Objektorientering ses som den filosofi, det tankesätt, som ligger till grund för själva metoden. Det finns alltså en mängd olika metoder som baserar sig på det objektorienterade synsättet. Analysen i detta arbete stöder sig på den metod som presenteras i boken ”Objektorienterad analys och design” (Mathiassen, Munk-Madsen, Nielsen & Stage 2001). En metod använder sig i sin tur av olika tekniker vilka kan ses som de praktiska tillvägagångssätt som behövs för att genomföra en analys och dokumentera resultatet. Syftet med denna avhandling är att genomföra en objektorienterad analys och diskutera användbarheten av beskrivningstekniken UML. I kapitel 2 finns en kort översikt över de UML - diagram som använts. Det antas att läsaren redan är bekant med beskrivningstekniken.

1.1. Syfte

Det finns två olika syften med denna avhandling. Det första syftet är att genomföra ett planeringsuppdrag åt Vasa yrkeshögskola, där uppdraget är att göra en objektorienterad analys av ett datasystem för planering av arbetsfördelningen för undervisningen för nästkommande läsår. Uppdraget finns närmare förklarat i kapitel 1.3. Beskrivningstekniken som används är UML. Analysen är redovisad i kapitel 3. Det andra syftet är att testa och utvärdera UML som beskrivningsteknik i projekt av denna storlek. Utvärderingen görs genom att resultaten från den praktiska analysen diskuteras utifrån de uppställda hypoteserna. Denna diskussion återfinns i kapitel 4.

1.2. Hypoteser

Utgångspunkten för diskussionen i kapitel 4 är två hypoteser. Hypoteserna är avsiktligen formulerade på ett sådant sätt att de lika gärna kunde tolkas som en projektledares önskemål på ett användbart verktyg.

Den första hypotesen lyder: "UML är en beskrivningsteknik som är lätt att förstå, lätt att använda och resulterar i användbara dokument." Speciellt intressant är alltså användbarheten av de diagram som produceras. Faktorer som inverkar på användbarheten är symbolerna begriplighet, diagrammens överskådlighet och modifierbarhet samt deras fungerbarhet som diskussionsunderlag för beslut vid utvecklingen av datasystemet.

Den andra hypotesen berör frågan om hur väl beskrivningstekniken kan integreras i ett programutvecklingsprojekt, d.v.s. hur väl tekniken kan anpassas till det praktiska projektarbetet. Diskussionen utgår ifrån kundorienterade och relativt små programutvecklingsprojekt där kraven på snabba resultat och kostnadseffektivitet är speciellt stora. Ett typiskt projekt, som här avses, har en tidsplan som understiger ett kalenderår och involverar 2-5 projektmedlemmar. Den valda tekniken borde alltså vara lätt att använda och ge snabba resultat. Den andra hypotesen lyder således: "En objektorienterad metod med UML som

beskrivningsteknik ger tillräckligt underlag för definition, dokumentation och presentation av en modell av datasystemet."

1.3. Planeringsuppdrag

Planeringsuppdraget som ligger till grund för avhandlingen kom från Vasa yrkeshögskola. Uppdraget gick ut på att planera och dokumentera ett system för planeringen av arbetsfördelningen för nästkommande läsår inom enheten för företagsekonomi och turism. Analysen skulle göras som en objektorienterad analys och resultera i en analys som kunde ligga till grund för förverkligandet av datasystemet. Utgångspunkterna för analysen finns i kapitel 3.1. Analysen finns i sin helhet i kapitel 3. Analysen kan läsas fristående från de övriga kapitlen. Det var från början givet att analysen skulle resultera i en relationsdatabas och detta faktum hade en viss betydelse för analysen av klassdiagrammet. Arbetsnamnet för det planerade datasystemet är ATP och är en förkortning för arbetstidsplanering.

1.4. Metodbeskrivning

Den objektorienterade analysen baserar sig, med vissa modifikationer, på den metod som finns beskriven i boken ”Objektorienterad analys och design” (Mathiassen m.fl. 2001). I boken används inte objektdiagram men ses här det som ett viktigt diagram vid skapandet av förståelse för den kommande arkitekturen. Analysen har kompletterats med en affärsprocesskarta (Fagerström, Bjurhager, Wallström & Jönsson 1998) som ger en bra översikt över problemområdet. Som tidigare framgått används UML som beskrivningsspråk. UML-diagrammen är ritade med modelleringsverktyget Rational Rose Enterprise och är direkt kopierade (copy/ paste) som bilder från programmet. Ett modelleringsverktyg var inte nödvändigt vid genomförandet av denna analys, men underlättar läsbarheten av analysresultaten och stöder genomförandet av den efterföljande designen. Arbetet är avgränsat till att gälla analysen och därmed behandlas inte själva programmeringen och förverkligandet.

Det finns vissa grundläggande återkommande metodologiska begrepp som är viktiga att klarlägga här i inledningen. Dessa är: (1) problemområde, (2) användningsområde (3) objektsystem och (4) datasystem.

(1) Problemområde definieras som den del av omgivningen som administreras med hjälp av datasystemet

(2) Användningsområde definieras som den organisation som administrerar problemområde

(3) Objektsystem uttrycker en användares uppfattning av det framtida datasystemet

(4) Datasystemet definieras som en samling av komponenter, som implementerar kraven på datamodell, funktioner och gränssnitt (Mathiassen m.fl. 2001: 20-24).

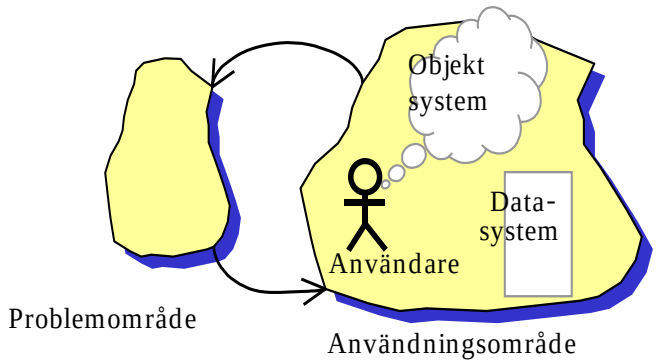


Bild 1. Samband mellan problemområde, användningsområde, objektsystem och datasystem (Mathiassen m.fl. 2001: 21).

Förenklat kan man säga att arbetet går ut på att analysera problem- och användningsområde och att skapa ett datasystem som uppfyller uppställda kriterier. Vad som menas med problem- resp. användningsområde i ATP-systemet samt vilka kriterier som ställts beskrivs i kapitel 3.1.

2. Kort om UML

UML är en beskrivningsteknik som framgångsrikt utvecklats ur de objektorienterade metoder som var rådande under slutet av 80-talet och i början av 90-talet. I UML sammanförs de tankar och teorier som utvecklats av Grady Booch, James Rumbaugh och Ivar Jacobson. Utvecklingen tog fart i mitten av 90-talet då dessa tre, som skämtsamt brukar kallas för de tre amigos, blev anställda av det amerikanska bolaget Rational. Efter en ganska lång utvecklingsprocess och ett flertal olika versioner blev UML godkänt av OMG [1] som UML 1.0 i oktober 1997 och blev industristandard (Fowler Martin & Scott Kendall 1997: 169). UML 2.0 utkom år 2005 och UML har sedermera etablerat sig som det mest använda visuella modelleringsspråket i världen (UML Resource Page).

Unified Modeling Language (UML) är ett visuellt beskrivningsspråk som ger planerare och designers möjlighet att visualisera, specificera, konstruera och dokumentera strukturer och beteende i ett system samt möjlighet att modellera den verksamhet som använder systemet (Bennett Simon, Skelton John, Lunn Ken, 2001:6). Tanken är att med olika diagram beskriva datasystemet från olika perspektiv och att genom en kombination av dessa få till stånd en helhetsbild. Symbolerna som används kallas för element och varje element har sin givna betydelse och kan jämföras med syntaxen i ett programmeringsspråk. Diagrammen kan ses som modeller av det system som håller på att konstrueras och genom att välja ett perspektiv i taget är det lättare att förstå och överblicka komplexiteten i ett system. Bild 2 presenterar uppdelningen i olika typer av diagram. Diagrammen visar systemet ur olika perspektiv.

[1] OMG = Object Management Group. Standardiseringsorganisation som har som mål att utveckla och främja objektorienterad paradigm. De viktigaste standards som OMG definierat är CORBA och UML.

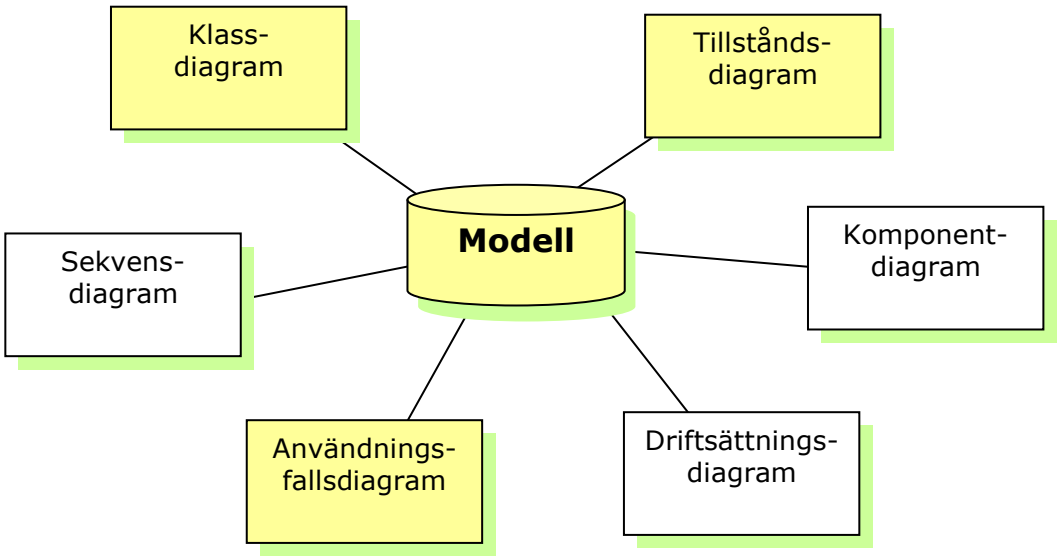


Bild 2. Olika UML-diagram visar ett system ur olika perspektiv (Kruchten 2002: 12).

Diagrammen brukar även delas in i två grupper, statiska och dynamiska diagram. Statiska diagram visar strukturer och relationer mellan olika element medan dynamiska diagram visar hur systemets komponenter samarbetar för att realisera sina funktioner. Av de diagram som behandlas i detta arbete är objektdiagram, användningsfallsdiagram och klassdiagram statiska medan tillståndsdiagrammet är dynamiskt. Denna analys behandlar inte sekvens-diagram, komponentdiagram och driftssättningsdiagram emedan dessa hör till förverkligandet i designfasen. En analys fokuserar vanligtvis antingen på verksamheten eller på datasystemet. Analysen i detta arbete och alla UML-diagram tar fasta på att beskriva själva datasystemet. Av detta följer att objekten som analyseras är datasystemets objekt. Samma diagram och beskrivnings-teknik kunde även användas i en verksamhetsanalys med den skillnaden att man då istället skulle fokusera på verksamhetsobjekt.

I kapitel 2.1 – 2.4 finns redogjort för de diagram som använts i analysen i kapitel 3 samt beskrivningar till hur de element som ingår i respektive diagram har använts.

2.1. Objektdiagram

Målet med objektdiagrammets är att visa vilka objekt som är väsentliga för berört problemområde i datasystemet som håller på att utvecklas. De UML-element som ingår finns visualiserade i bild 3. I objektdiagrammet strävar man till att använda de benämningar på ting/ företeelser/ objekt som används i verksamheten och man borde kunna utläsa hur utvecklaren ser på problemområdet. Objektdiagrammet har här även använts för presentation av ett utkast på planerad systemarkitektur. Objekten har tilldelats stereotyper av vilka följande har använts: (1) «boundary» gränssnitt, (2) «control» funktion och (3) «entity» data (Kroll 1999: 340). Varje stereotyp har sin egen grafiska presentation. Målet är att skapa en treskiktsarkitektur bestående av en gränsnitts-, en funktions- och en modellkomponent.

Objektdiagrammet ligger till grund för den efterföljande diskussionen om arkitekturen och komponenternas utformning. Gränsnittskomponenten ansvarar för att användaren skall kunna interaktera med systemet, funktionskomponeneten innehåller behandlingsregler för inamatad och lagrad information (läs: programlogik) och modellkomponenten innehåller den data som systemet skall lagra. Objekten sammanbinds med ett väl valt verb som visar planerarens syn på relationen mellan objekten. Utifrån detta kan man sluta sig till vilket ansvar respektive objekt har getts. Ett gemensamt notationssätt för alla diagram är elementet “Kommentar” som kan användas för att ytterligare förtydliga betydelsen av ett element. I bild 3 har kommentarer använts för att visa de olika stereotyperna.

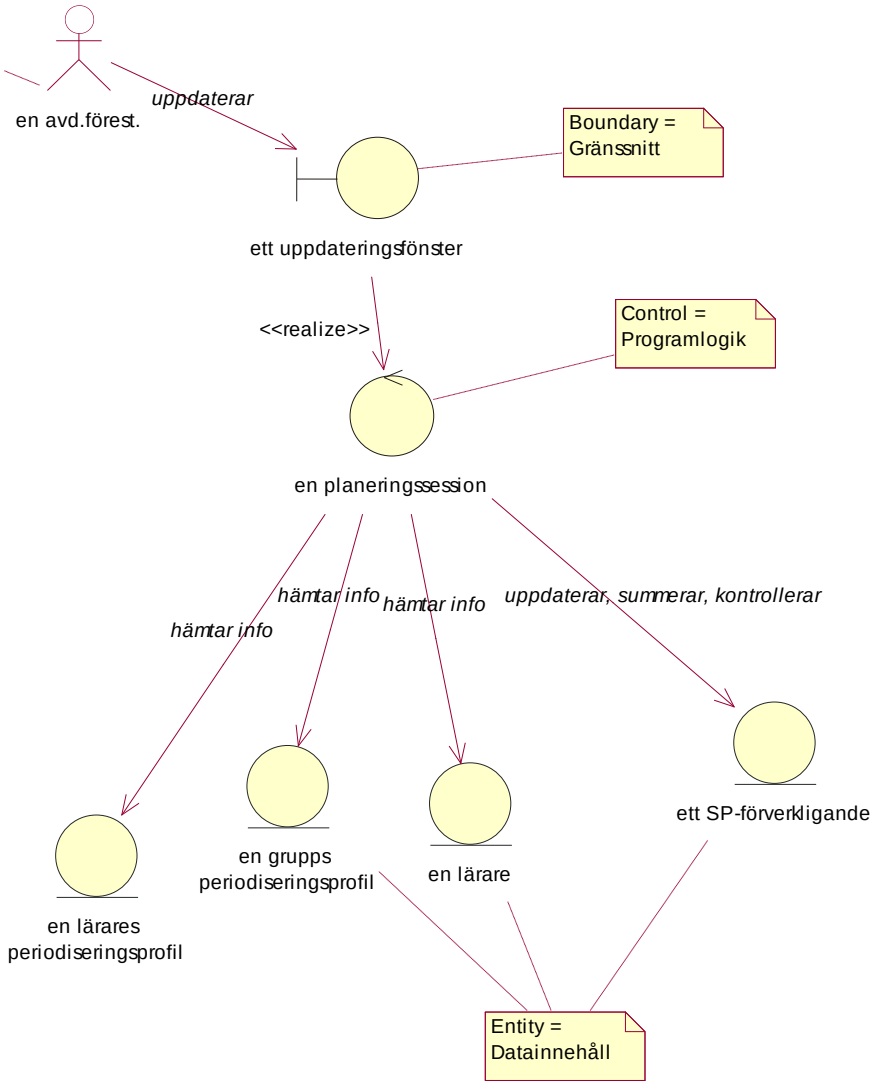


Bild 3. Exempel på ett objektdiagram som det används i detta arbete.

2.2. Användningsfallsdiagram

Målet med användningsfallsdiagrammet är att beskriva vilka situationer i verksamheten datasystemet skall ta hand om, hur det planerade systemet är tänkt att användas och vilka aktörer som är involverade. Både berörda användningsfall och aktörer identifieras och analyseras. En aktör är oftast en fysisk person men kan även vara ett angränsande system som interakterar med systemet, d.v.s. ger eller får information av systemet. Ett användningsfall består

av ett händelseförlopp som bildar en funktionell helhet som logiskt hör ihop. Det är oftast fördelaktogast att inleda analysarbetet med användningsfallsanalysen. Med hjälp av användningsfallsanalysen strävar man till att hitta och gruppera systemets väsentliga objekt och händelser. Elementen är aktörer och användningsfall. Varje aktör beskrivs kort och för varje användningsfall analyseras händelseförloppet. Ett vanligt misstag är att tro att diagrammet skall visa ett flöde, vilket inte är fallet. Diagrammet visar endast aktörer som deltar i användningsfall. Det är sjäva analysen av händelseförloppet som är det väsentliga. Med hjälp av analysen hittas involverade objekt. Väsentliga objekt plockas ut och återfinns i objektdiagrammet. Det finns två vanliga stereotyper som brukar användas vilka visar relationerna mellan två användningsfall. Stereotyperna är «extend»

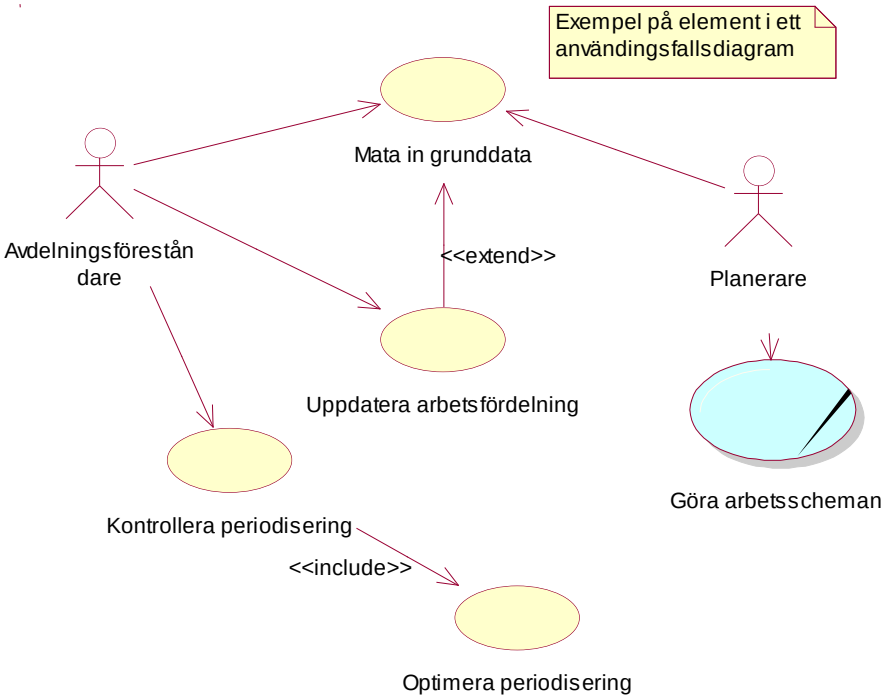


Bild 4. Exempel på ett användningsfallsdiagram.

och «include». Stereotypen «extend» innebär att användningsfallet *i vissa bestämda situationer* kan vara aktuellt medan «include» innebär att

användningsfallet ifråga *alltid* skall tas med. Fördelen med uppdelningen är att användningsfallen kan analyseras separat och att ett visst användningsfall kan utnyttjas av flera andra (jämför med en programmodul som anropar en undermodul). Bild 4 visar ett exempel på ett användningsfallsdiagram. Användningsfallsdiagrammet kan också innehålla användningsfall som inte är direkt föremål för analysen, men som finns med för att förtydliga bilden av problemområdet. Här åtskiljs 'Use case' och 'Business Use Case', se bild 5.

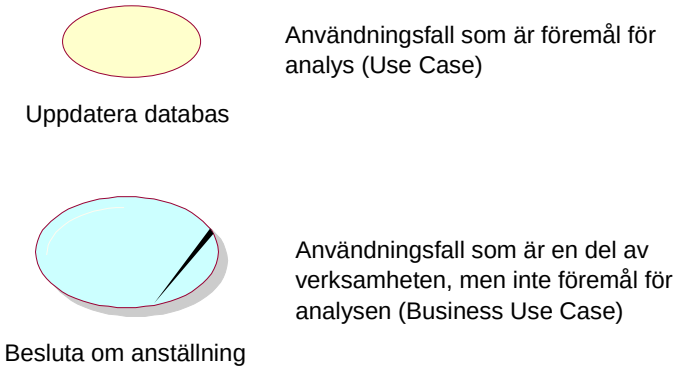


Bild 5. Två olika typer av användningsfall

2.3. Klassdiagram

Syftet med klassdiagrammet är att presentera modellkomponenten (läs: databaslösningen). I kriterierna för denna analys sades det att den slutgiltiga databasen skall vara en relationsdatabas. Därför har det här valts att inte framställa ett renodlat klassdiagram. Alla klasser har tilldelats stereotypen «table». Klassdiagrammet är i själva verket ett ER-schema. Stereotypen «table» har en viss grafisk presentation (bild 6) och de attribut som angetts kommer alltså att bli kolumner i tabellerna. Här har primärnyckeln för tabellen tilldelats stereotypen «Pkey» och sekundärnyckeln «Fkey». Multipliciteten har angetts i relationen mellan tabellerna och därmed kan detta diagram direkt ligga till underlag för skapandet av databasen senare i designskedet.

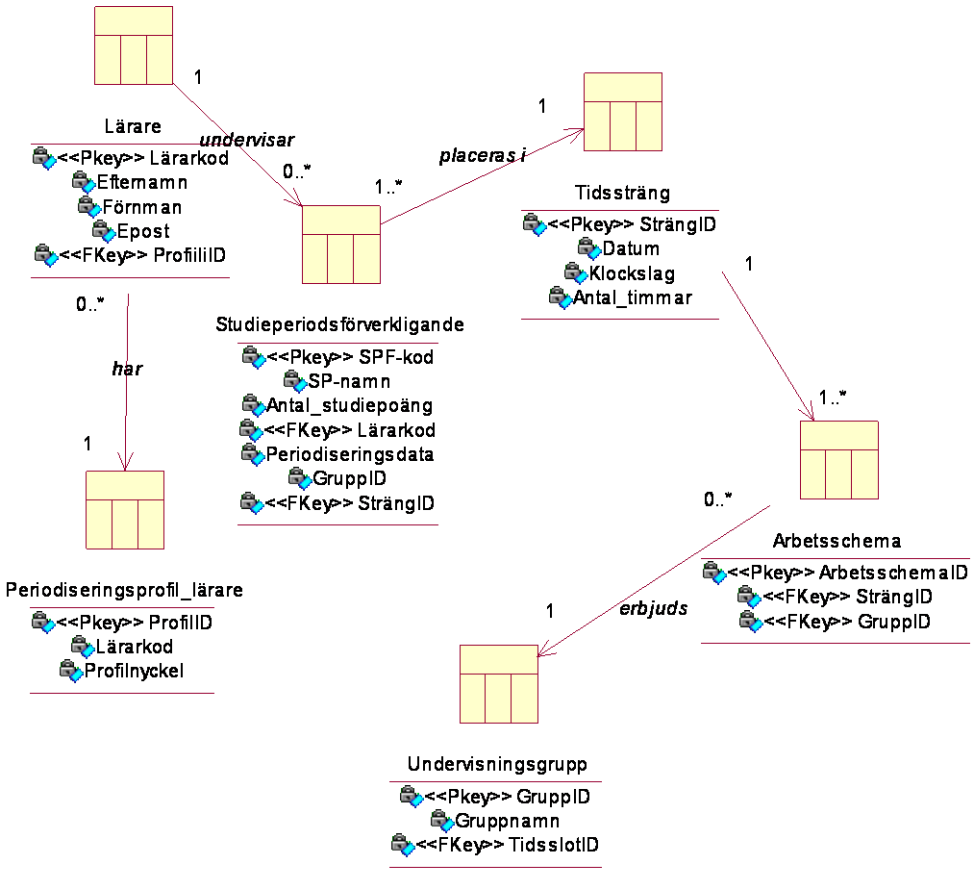


Bild 6. Exempel på ett anpassat klassdiagram, där klasserna tilldelats stereotypen «table» och attributen är synliga.

2.4. Tillståndsdiagram

Tillståndsdiagram används för att visa vilka olika tillstånd ett objekt kan ha under sin livstid samt hur övergångarna mellan tillstånden sker. Detta kan vara mycket användbart då man vill följa centrala objekt i ett datasystem såsom en faktura i ett faktureringssystem eller ett ärende i ett ärendehanteringssystem. Grundläggande koncept är tillstånd, övergång, händelse och villkor.

Bild 7 visar ett exempel på vilka tillstånd objektet studerande kan tilldelas under sin livstid i systemet. Ett tillståndsdiagram har oftast ett start- och ett sluttillstånd som presenteras i form av olika element som påminner om

punkter. Ett tillstånd anges i bestämd form och anger alltså objektets status vid ett visst tillfälle. Övergången från ett tillstånd till ett annat sker genom en händelse som kan ha ett argument. Argument anges inom parentes (så här) och villkor som anges inom hårda parenteser [så här]. I designskedet tas sedan ställning till hur denna händelse och tillståndsförändring på bästa sätt realiseras i datasystemet.

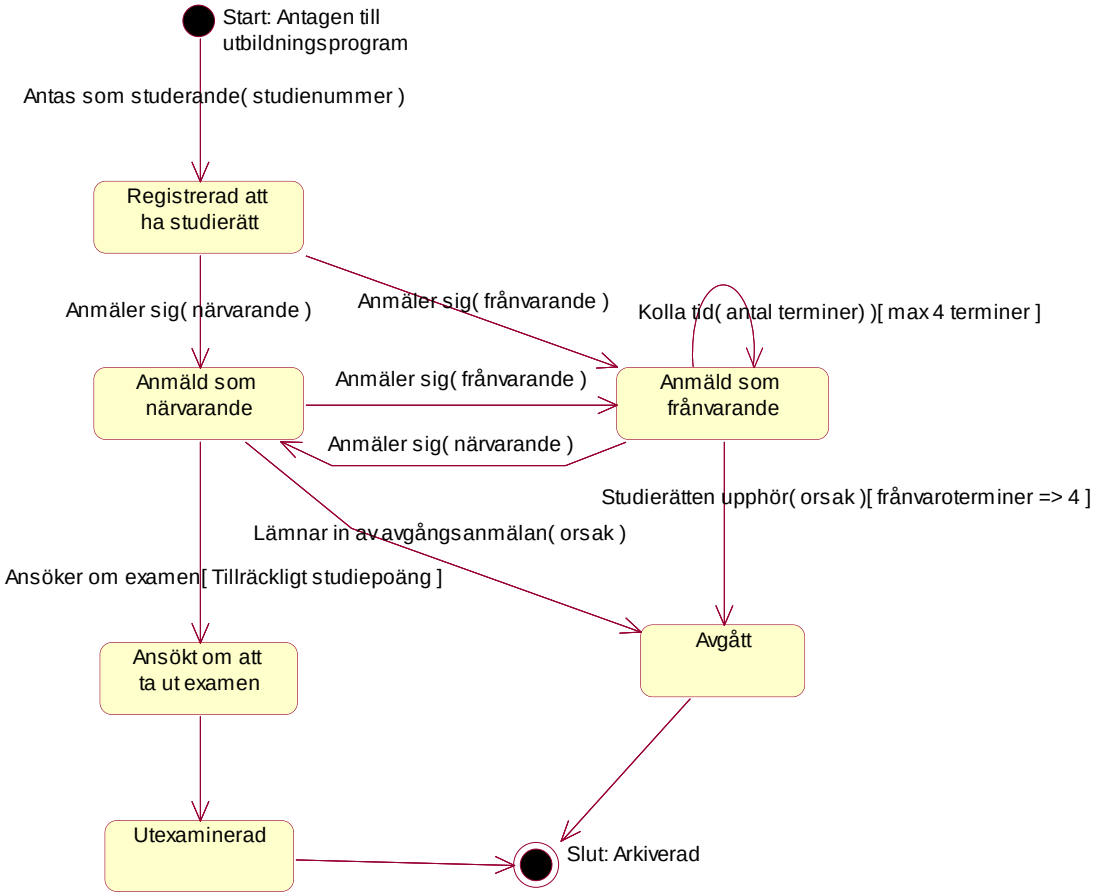


Bild 7. Exempel på ett tillståndsdiagram.

3. ANALYSEN

Syftet med analysen är att beskriva hur datasystemet är tänkt att fungera och detta kapitel utgör den dokumentation som skall ligga till grund för förverkligandet av systemet.

3.1. Definition av uppgiften

3.1.1. Syfte

Syftet med denna analys är att planera ett datasystem för arbetstidsplanering för undervisningen vid Vasa yrkeshögskola. Systemdefinitionen är beskriven med hjälp av VATOFA-kriteriet (Mathiassen m.fl. 2001: 58), vilket ger en bra överblick över det planerade systemet. Problemområdet finns beskrivet i kapitel 3.1.3. Det viktigaste i definitionen av problemområdet är att identifiera de viktigaste objekten och dessa finns sammanfattade i ett objektdiagram. För att få en bra överblick av användningsområdet har en affärsprocesskarta använts (Fagerström m.fl. 1998: 277). Denna ligger som underlag för beskrivningen av informationssystemet.

3.1.2. Systemdefinition

Systemdefinitionen för ATP-systemet enligt VATOFA-kriteriet definieras nedan.

.

Villkor:

Systemet bör *lagra och underhålla* nödvändig information för att planera arbetsfördelningen för kommande läsår; lärare, studiegrupper, studieperioder, utbildningsprogram, förverkligande av studieperioder för nästa läsår och periodiseringsinformation för grupper och lärare. Systemet skall tillhandahålla funktioner för att *skapa* arbetsfördelningen för nästkommande läsår och det skall finnas en koppling till läsordningssystemet Mimosa. Systemet bör vara visuellt översiktligt och lättanvänt. För planeraren skall uppdatering och kontroll vara lätt att utföra. För användaren skall informationen vara

lättarbetad och det skall vara möjligt att behandla enskilda ansvarsområden (avdelningar). Eftersom man laborerar med gemensamma lärarresurser skall det även vara möjligt att se hur egna ändringar inverkar på helheten. Det skall finnas möjlighet att sortera och filtrera informationen på olika sätt och lätt att skriva ut kontrollistor. Dialogfönstren skall vara på finska eller svenska beroende på användarens språkprofil.

Användningsområde:
Systemet skall användas av de personer i organisationen som ansvarar för planeringen av arbetsfördelningen: prefekt, avdelningsföreståndare och planerare.

Teknologi:
Ett fleranvändarsystem i Windows-miljö med möjlighet för användaren att via sin egen arbetsstation i skolans nätverk kunna läsa och uppdatera information enligt eget ansvarsområde. Databasen är en relationsdatabas och informationen skall vara tillgänglig via SQL även för angränsande system som behöver information såsom schemaläggningssystemet och lönesystemet.

Objekt:
Viktiga objekt/ aktörer i verksamheten som berörs är lärare, studieperiod, studiegrupp, utbildningsprogram, studieperiodsförverkligande, periodiserings-nyckel (undervisningstimmar per period), prefekt, avdelningsföreståndare och planerare.

Funktionalitet:
Följande funktioner skall finnas med i systemet: underhåll av grunddata, skapande av studiegrupper, skapande av studieperiodsförverkligande för kommande läsår, kontroll av studiegruppens periodisering, kontroll av lärarens periodisering, underhåll av förverkligande samt flexibla utskriftsrutiner med olika sorterings och filtreringsmöjligheter.

Ansvar:
Datasystemet skall i huvudsak lagra och underhålla sin egen information, men det skall finnas möjlighet att föra över grundinformation om studieperioder och deras periodisering till läsordningssystemet. I detta skede skall systemet lagra

uppgifter om vilka studieperioder en viss lärare undervisar i. I ett senare skede skall även uppgifter om övrigt arbete, såsom projekt- eller utvecklingsarbete lagras. Vid inloggningen skall systemet använda sig av skolans behörighetssystem via LDAP och får via det användarnas profil med bl.a. användarens språkkod som styr på vilket språk användargränssnittet visas.

3.1.3. Definition av problemområde och användningsområde

I analysen utgår resonemanget från verksamhetens funktioner, vilka delas upp i olika ansvarsområden. Hur skolans organisationsschema ser ut är i detta skede av underordnad betydelse. Med verksamheten avses Vasa yrkeshögskolas enhet för företagsekonomi och turism. För att klargöra bilden av det kommande datasystemet har analysen av problemområdet sammanfattats i ett objektdiagram. Utgångspunkten har varit de objekt som identifierats i systemdefinitionen. Objekten har kategoriserats beroende på komponent-tillhörighet kommande systemarkitekturen. Objekten har delats in i användargränssnitt-, funktions- och dataobjekt, vilka via sina stereotyper visualiseras på olika sätt. Bild 8 förtydligar vad som menas.

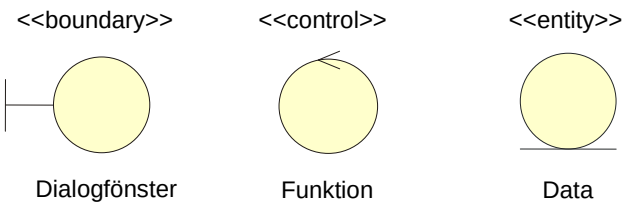


Bild 8. Förklaringar till objekt och stereotyper.

Objektdiagrammet i bild 9 visar en tidig skiss över datasystemets uppbyggnad. Skissen presenterar planerarens uppfattning om hur olika objekt borde samarbeta. Utgångspunkten har varit att bygga upp systemet som en 3-skiktsarkitektur. Alla «boundary»-objekt kommer att bilda gränssnitts-komponenten, alla «control»-objekt funktionskomponenten och alla «entity»-objekt modellkomponenten.

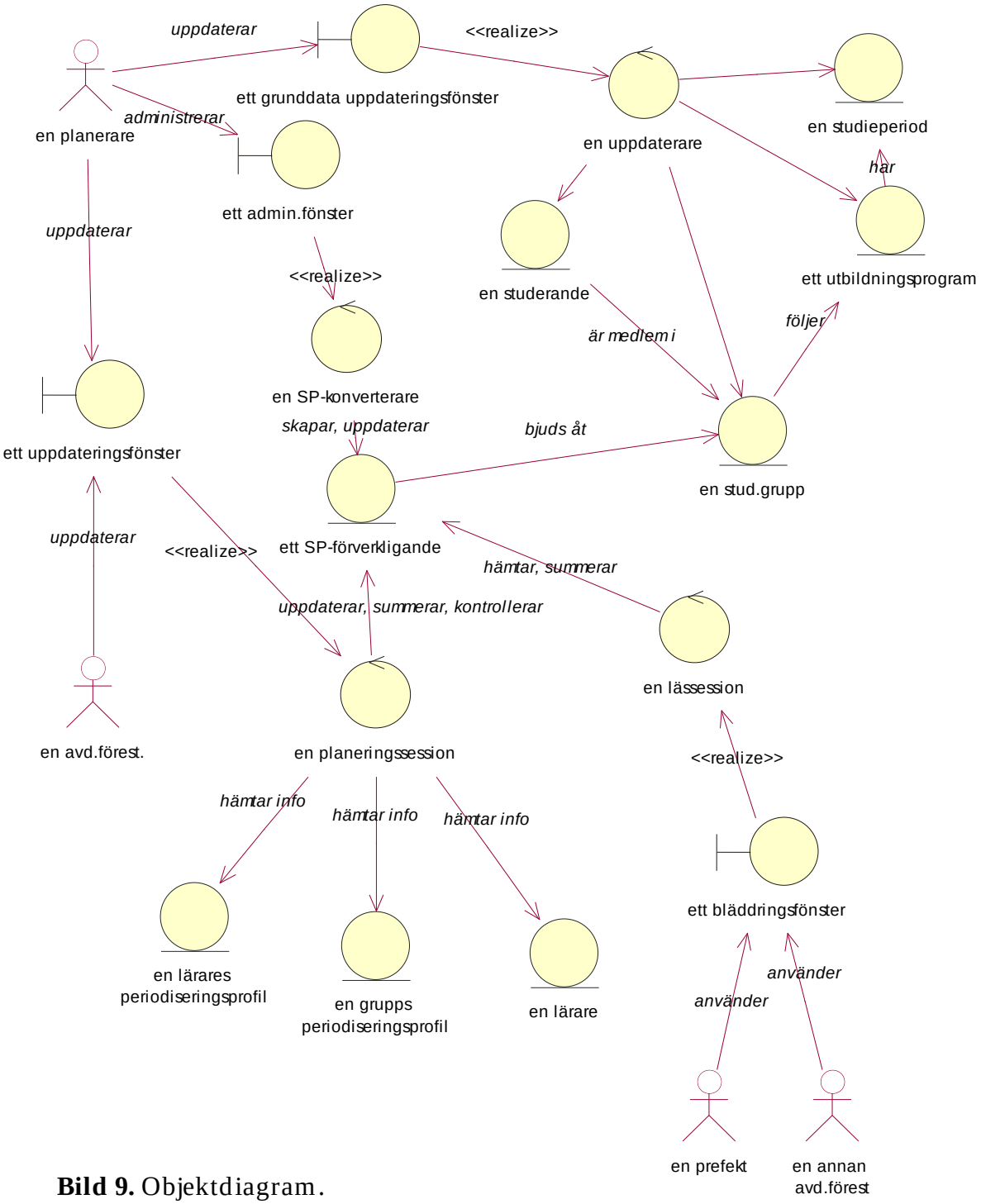


Bild 9. Objektdiagram.

Härefter gäller det att bilda sig en uppfattning om användningsområdet, d.v.s. den organisation som skall använda datasystemet. Beskrivningen är gjord med hjälp av en affärsprocesskarta (Fagerström m.fl. 1998: 277), vilken ger en bra

översikt över vilka processer som ingår. Affärsprocesskartan ligger till underlag för den efterföljande beskrivningen av informationssystemet.
Ansvarsområden per funktion finns i den nedre delen (bild 10) och informationsflödet i den övre. Affärsprocesskartan beskriver även processen för schemaläggningen för att ge en helhetsbild och öka förståelsen för verksamheten. Den fortsatta analysen av användningsfall avgränsar sig dock till det område som finns innanför den streckade linjen.

Här några förtydligande beskrivningar som förklarar innebörden i affärsprocesskartan; Det administrativa arbetet leds av enhetens prefekt, som har beslutanderätten inom enheten. Prefektens befogenheter finns definierade i de verksamhetsregler som utfärdats av yrkeshögskolans styrelse. Prefekten har budgetansvar och rapporterar till ledande rektorn. Den operationella verksamheten leds av enhetens ledningsgrupp (LG) som består av prefekten och enhetens avdelningsföreståndare. LG sammanträder i regel varannan vecka. I LG tas beslut i ärenden som berör den pedagogiska utvecklingen och den operationella verksamheten.

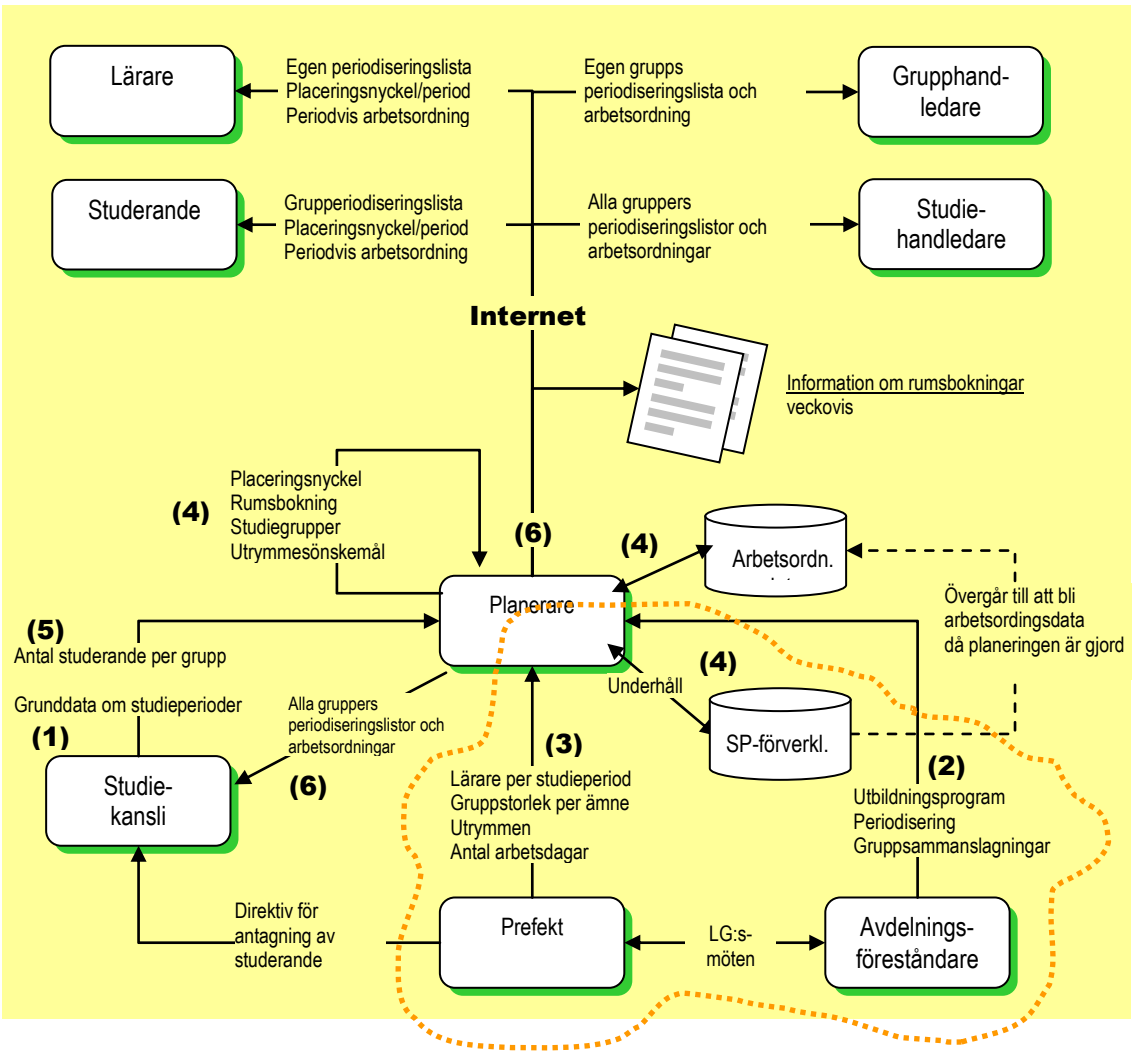
Planeringen av arbetsfördelningen är en del av den operationella verksamheten. Avdelningsföreståndaren har nyckelrollen i denna process. På dennes ansvar vilar planeringen av arbetsfördelningen inom den egna avdelningen och koordineringen av arbete med övriga avdelningar och prefekten. Utgångspunkten vid planeringen är studieplanerna vilka stipulerar vilka studieperioder som bör erbjudas följande studieår. Planeraren deltar i hela processen och har till uppgift att uppdatera grunddata och framförallt studieperiodsförverkliganden. Resultat från arbetsfördelningen ligger även till grund för schemaläggning och läseordningar. En mer ingående analys av väsentliga användningsfall samt aktörers (personers) interaktion med informationssystemet finns i kapitel 3.2.1.

3.1.4. Informationssystemet

Nedan följer en analys av det informationsflöde samt in-/ utdata som har betydelse för analysen. Studiehandboken där alla utbildningsprogram finns beskrivna utgör utgångspunkten för all planering. Analysen följer de sifferbeteckningar som kan ses i affärsprocesskartan på bild 10.

Indata (1): Grunddata om studieperioder.
Källa: Grunddata för studieperioderna administreras och underhålls av studiekansliet via datasystemet Winha som är ett eget separat datasystem.
Överföringssätt: Manuellt.
Information: För varje studieperiod behöver planeraren följande data: kod, namn, omfattning (antal studiepoäng) och studiegrupp (= den grupp som, enligt gällande studiehandbok, studieperioden i första hand skall bjudas åt).

Indata (2): Utbildningsprogram, periodisering, gruppsammanslagningar
Källa: Protokoll och muntlig information om beslut angående nästkommande studieår, utfärdade av prefekten. Informationen är svår att formalisera och kräver under året en aktiv dialog mellan prefekt, avdelningsföreståndare och planerare.
Överföringssätt: Manuellt.
Information: Uppgifter om eventuella förändringar i utbildningsprogrammens struktur (= ändring som kommer att finnas i studiehandboken nästkommande år), uppgifter om vilka studieperioder som skall erbjudas nästkommande studieår, uppgifter om sammansatta studiegrupper (en studieperiod kan, ifall antalet studerande annars skulle bli för litet, erbjudas endast en gång och då åt sammanslagna studiegrupper och/ eller inriktningsalternativ), uppgifter om periodisering (d.v.s. uppgifter om när studieperioden skall infalla för att den pedagogiskt sett skall vara på rätt plats).



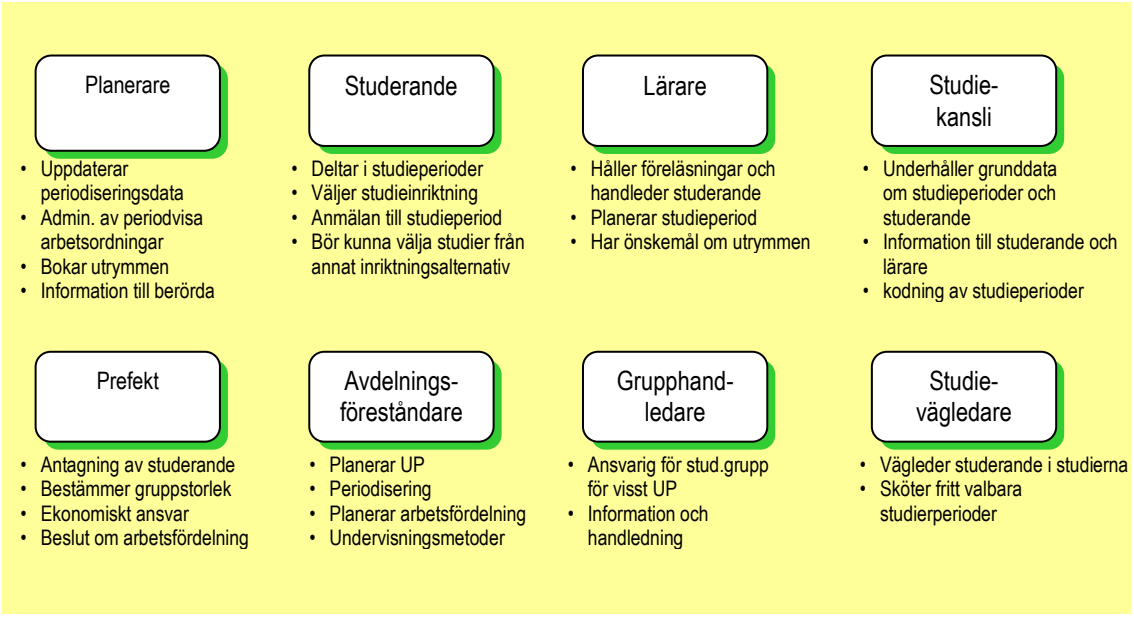


Bild 10. Affärsprocesskarta.

Indata (3): Lärare per studieperiod, gruppstorlek, utrymmen, antal arbetsdagar
Källa: Protokoll gällande beslut om nästkommande studieår, tagna av prefekten under året. Diskussioner mellan avdelningsföreståndare och prefekt angående arbetsfördelningen för nästkommande år.
Överföringssätt: Manuellt.
Information: Uppgifter om arbetsfördelning för nästkommande år (= uppgifter om vem som är lärare för viss studieperiod), antal arbetsdagar och periodindelning för nästkommande studieår, uppgifter om ledigheter och eventuella speciella händelser, uppgifter om tillgängliga utrymmen, beslut om gruppstorlekar (minimi- och maximigränser).

Indata/Utdata (4): Placeringsnyckel, närtimmar/ period, rumsbokningsbehov och studiegrupper.
Källa: Grunddata för arbetstidsplaneringen finns för närvarande i form av en kalkylmodell i Excel som planeraren uppdaterar och underhåller. Utdata skapas i denna process då informationen i kalkylmodellen underhålls. Kalkylmodellen betjänar både arbetstidsplaneringen och sedan även schemaläggningen.
Överföringssätt: Manuellt.
Information: Uppgifter om en lärares arbetsbörda i en viss period (för att undvika överbelastning), uppgifter om studiegrupper (studiegrupper bildas i regel inom det egna utbildningsprogrammet men det skall finnas möjlighet att välja studieperioder utanför det egna programmet), uppgifter om vilka studieperioder som inte får erbjudas samtidigt (studerande bör ges möjlighet att välja), uppgifter om vilka studieperioder som kan erbjudas samtidigt, uppgifter om en lärares eventuella tjänstledigheter och/ eller utlandsvistelser.

Indata (5): Antal studerande per grupp

Källa: I ett tidigt skede av planeringen (maj-juni-juli) används ett uppskattat antal studerande och senare (augusti-september) används det verkliga antalet studerande (godkända sökande) för respektive utbildningsprogram.

Överföringssätt: Manuellt.

Information: Uppgifter om antalet studerande per utbildningsprogram (dessa uppgifter behövs för att man skall kunna avgöra hur många gånger en studieperiod skall erbjudas).

Utdata (6): Periodiseringslistor och arbetsscheman per grupp, per lärare och per undervisningsutrymme.

Källa: Informationen skapas i användningsfallen ”skapar arbetsordning” och ”distribuerar arbetsordningar”.

Överföringssätt: Periodiseringslistor för både lärare och studiegrupper distribueras via Internet.

Information: Periodiseringslistor för nästkommande studieår för lärare och studiegrupper, arbetsscheman för nästkommande period för lärare och studerande. Dessutom kan man också se bokningsläget per utrymme per vecka.

3.2. Analys av användningsområdet

Användningsfalldiagrammet (bild 11) visar vilka användningsfall som identifierats i analysen. De mindre ljusgula användningsfallen är de som är föremål för analysen medan de större blåa enbart finns med för att ge en helhetsbild (se förklaring i bild 5). Aktörerna och användningsfallen har analyserats och dokumenterats nedan.

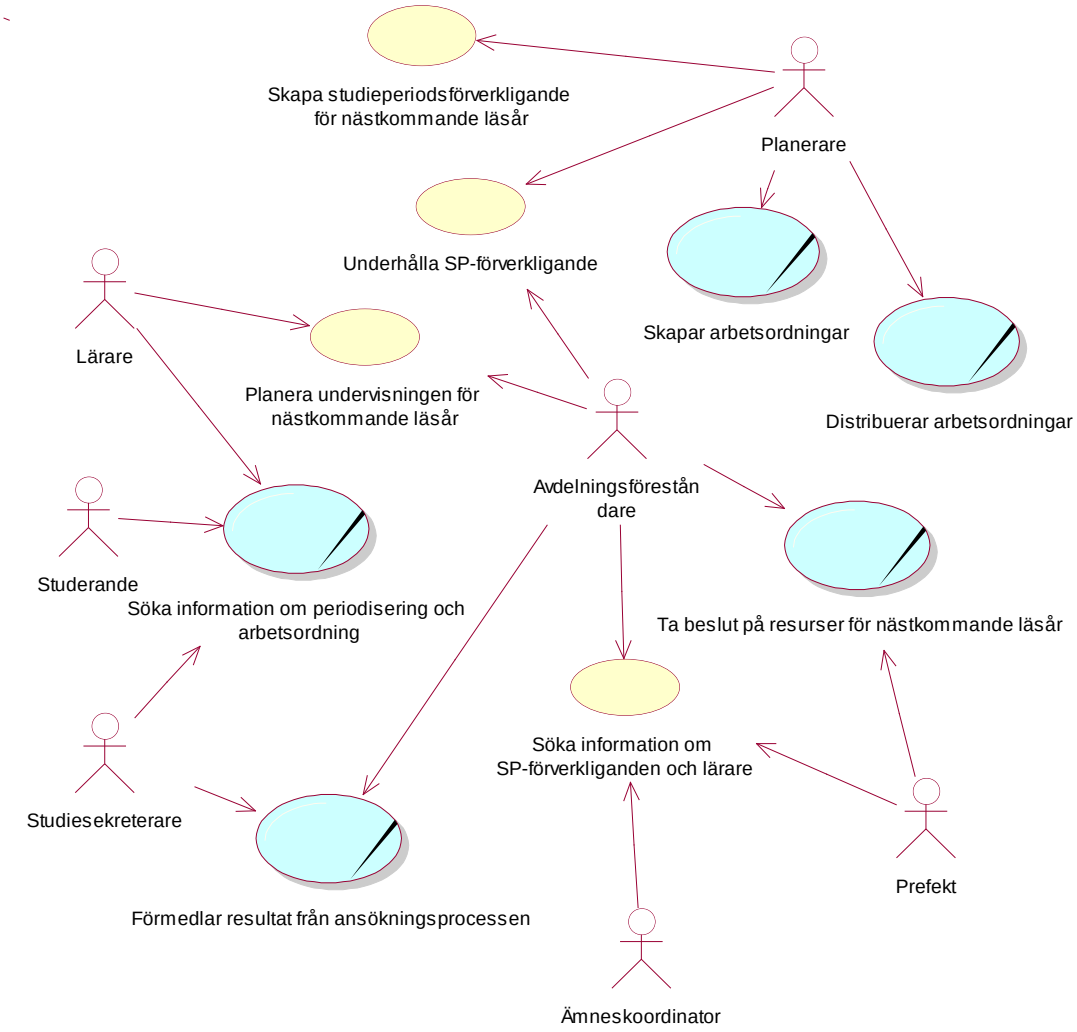


Bild 11. Användningsfallsdiagram.

3.2.1. Analys av aktörer

Följande aktörer har identifierats och analyserats.

Planerare
Person som ansvarar för planeringen av studieperiodernas förverkligande och underhåller utbildningsprogrammen i Winha samt är ansvarig för schemaläggningen per period.

Avdelningsföreståndare
Person som är ansvarig för ett eller flera utbildningsprogram och är närmaste förman för de lärare som undervisar inom ifrågavarande utbildningsprogram.

Prefekt
Person som är ansvarig för en viss bransch inom Vasa yrkeshögskola. Det finns tre olika branscher. Varje bransch har ett flertal olika utbildningsprogram.

Lärare
Person som är lärare vid Vasa yrkeshögskola och ansvarar för planeringen och genomförandet av SP-förverkligande för ett visst kunskapsområde. Läraren undervisar inom ett eller flera kunskapsområden och ofta inom flera olika utbildningsprogram.

Ämneskoordinator
Person som har fått till ansvar att samordna undervisningen inom ett visst ämnesområde (t.ex. matematik).

Studerande
Person som är inskriven som studerande och har studierätt vid Vasa yrkeshögskola. Vissa studerar för att få en examen och andra deltar i mer kortvariga specialutbildningar eller enstaka studieperioder inom Vasa öppna yrkeshögskola. Vasa yrkeshögskola utbildar både ungdoms- och vuxenstuderande.

Studiesekreterare
Person som jobbar på studiekansliet och som ansvarar för de studietjänster som erbjuds studerandena, såsom studiekort, blanketter för olika ansökningar, användarkonton, studiestödsärenden m.m. Studiekansliet handhar statistik av olika slag och ansvarar för administrationen av studerandena med hjälp av Winha.

3.2.2. Analys av användningsfall

Följande användningsfall har identifierats och händelseförloppen har analyserats. Utifrån dessa analyser har sedan datasystemets funktioner definierats och redovisats (kapitel 3.2.3) och väsentliga objekt plockats ut och vidare analyserats (kapitel 3.3). Väsentliga objekt har första gången de förekommer här i analysen angivits med *kursiv* stil.

Användningsfall:
”Skapa *studieperiodsförverkligande* för nästkommande läsår”

Händelseförlopp:

1
Planeraren går igenom och konstaterar vilka *utbildningsprogram* och vilka *studiegrupper* som är aktuella för nästkommande läsår

2
Planeraren skapar en ny databas (nuläget i Excel) innehållande alla *studieperiodsförverkiganden* per utbildningsprogram och per studiegrupp

3
Planeraren lägger in all data som finns tillgänglig för varje SP-förverkligande. Databasen innehåller följande data per förverkligande:

- studieperiodens kod
- unik kod för studieperiodens förverkligande (en studieperiod kan ha ett eller flera förverkliganden under läsåret)
- studieperiodens namn
- studieguidens version (årtal)
- språkkod (undervisningsspråk, t.ex. FIN)
- omfattning i studiepoäng
- studiegrupp(er), som förverkligandet skall erbjudas åt
- studiegruppstorlek (antal studerande)
- *periodisering* (antal närtimmar per period)
- totalt antal timmar
- uppgift om timmarna skall vara löneberättigande

- undervisningsmetod
- andel virtuell undervisning (%)
- ämneskod
- *uppgifter om den som skall undervisa (lärarkod)*
- uppgift om vem som har undervisat i studieperioden föregående läsår (lärarkod)
- *typ av arbete* (kod från överenskommelsen om heltidsarbete)
- uppgift om *antalet timmar undervisningsskyldighet* beroende på form av undervisning (kan vara branschspecifikt)
- önskemål om utrymme (*rumskod*)
- *kommentarer (föregående läsår)*
- *kommentarer (följande läsår)*

4

Innan överföringen görs till Mimosa för schemaläggning sker periodisering per vecka. Planeraren utgår ifrån det totala antalet närtimmar och gör en veckovis periodisering för varje studieperiodsförverkligande. Därefter gör en engångsflyttning från Excel till Mimosa där uppgifterna sedan bearbetas vidare.

Användningsfall:
”Underhålla *studieperiodsförverkliganden* för nästkommande läsår”

Informationen som styr uppdateringen av studieperiodsförverkliganden (SPF) finns hos avdelningsföreståndaren eller hos planeraren. Avdelningsföreståndaren ansvarar för att informationen är korrekt och för de beslut som tas angående resurser för nästkommande läsår. Avdelningsföreståndaren ansvarar också för att resultat som kommer ur ansökningsprocessen tas i beaktande och uppdateras i SPF-databasen. SPF-databasen ligger direkt till underlag för arbetstidsfördelningen och uppdateras vartefter arbetet med arbetsfördelningen fortskrider. Databasen bör hela tiden visa aktuellt läge.

Händelseförlopp:

1
Planeraren och avdelningsföreståndaren har under arbetsfördelningsarbetet kontinuerligt kontakt med varandra och båda har egna ansvarsområden för uppdateringen av SP-förverkligandena

2
Användaren öppnar en viss rad (post) i SPF-databasen

3
Användaren uppdaterar, lägger till eller tar bort rad (post) från databasen

Användningsfall:
” Planera undervisningen för nästkommande läsår”

Målet är att komma överens om vilka studieperioder en viss lärare skall undervisa i följande läsår. Beslutet tas av avdelningsföreståndaren i samråd med läraren. Utgångspunkten är föregående läsårs undervisning och utfall.

Händelseförlopp

1
Avdelningsföreståndaren skickar ut en förfrågan om huruvida läraren står till förfogande nästkommande år och ber om önskemål angående undervisning och övrigt arbete

2
Läraren svarar på förfrågan och ger in sina önskemål om nästkommande års undervisning och övrigt arbete

3
Avdelningsföreståndaren har ett utvecklingssamtal med respektive lärare där både föregående år och nästkommande år diskuteras

4
Ämneslärarna har egna möten där de tar ställning till arbetsfördelningen inom sina egna ämnesområden

5
Avdelningsföreståndaren och planeraren registrerar tillsammans den information som fåtts och beaktar denna vid uppdateringen av SP-förverkliganden

Användningsfall:
” Söka information om SP-förverkliganden och lärare”

Under arbetsfördelningens gång behöver olika personer information från den gemensamma SPF-databasen. Olika avdelningar kan ha gemensamma lärare. Dessa lärares arbetstid behöver koordineras. Förutom information om undervisningen finns det också uppgift om övriga arbetstimmar såsom internt utvecklingsprojektarbete, deltagande i arbetsgrupper, forsknings- och utvecklingsprojekt, handledning av slutarbeten, administrativt arbete, utveckling av egen undervisning m.m. Samtliga timmar påverkar den slutgiltiga arbetsfördelningen.

Händelseförlopp

1
Användaren söker information ur databasen, oftast information om en viss lärare. För att söka information bör användaren ha tilldelats behörighet att läsa databasen

2
Användaren kan se och bläddra i information om periodisering per lärare, studiegrupp och/ eller studieutrymmen

3
Användaren har möjlighet att söka information på olika sätt och få summeringar över en viss lärares arbetsbörda

4
Användaren kan även uppdatera informationen om han/ hon tilldelats dessa rättigheter.

5
Användaren har möjlighet att ta ut en kontrollista på önskad lärare

3.2.3. Funktioner

Utifrån analysen av användningsfallen har den funktionalitet här definierats som datasystemet behöver. Funktionaliteten har delats upp i ett antal funktioner som finns redovisade i tabell 1.

Tabell 1. Funktionstabell.

Nr	Funktion	Komplexitet	Typ
1.1	Uppdatering och underhåll av SPF. Möjlighet att uppdatera, lägga till och ta bort önskad information rad- och kolumnvis.	Omfattande	Uppdaterande
1.2	Skapande av SPF-databas för nästa läsårs SP-förverkliganden. Utgångspunkten är föregående års uppgifter och nästa års utbildningsplaner för respektive utbildningsprogram och årskurs.	Omfattande	Uppdaterande En gång per år
2	Arbetstidsplanering. Planera arbetsfördelningen för nästa år. Möjlighet att kontrollera årssummeringar och göra uppdateringar av lärare och/ eller grupper.	Omfattande	Uppdaterande
3	Sökning och bläddring i lärares och gruppers bokade studieperioder. Möjlighet till utskrifter.	Normal	Läsande
4	Uppdatering av styrdatatabeller. Parameteruppgifter som styr programlogiken	Normal	Uppdaterande Funktionsstyrande
5	Uppdatering av behörigheter. Det bör finnas möjlighet att ge användaren olika rättigheter inom ATP-systemet (tillgång till dialogfönster, läs- och uppdaterings-rättigheter).	Omfattande	Uppdaterande Styrande

3.2.4. Användargränssnitt

I detta avsnitt tas ställning till hur dialogen mellan användare och system skall utformas och hur användargränssnittet skall utformas. Detta konkretiseras med hjälp av ett exempel på ett användargränssnitt. En översikt över dialogen beskrivs i ett navigeringsdiagram (bild 12) som visar huvuddialogerna och hur olika dialogfönster är sammankopplade. För att visa hur dialogfönster och menyer kan tänkas utformas, presenteras nedan ett exempel på ett dialogfönster (bild 13). Exemplet bör ses som riktgivande och inte som definitivt. Navigeringsdiagrammet visar vilka dialoger/ fönster systemet borde innehålla samt visar hur man rör sig mellan olika dialoger. Navigeringsdiagrammet har ritats som ett aktivitetsdiagram i UML.

Vid inloggningen identifieras användaren via skolans LDAP och användarens profil används för att bestämma vilket språk som skall användas i dialogerna. Dialogerna kan visas antingen på finska eller på svenska. I nästa steg väljer användaren ingångsdialog och systemet skall kontrollera vilken behörighet och vilka rättigheter användaren har innan dialogfönstret kan öppnas. Om behörighet saknas visas ett felmeddelande.

Varje dialogfönster ansvarar för en viss funktionshelhet som beskrivs i funktionstabellen. Varje huvuddialog innehåller ett antal flikar som döljer ett antal underdialoger som behövs för en viss funktion. I exemplet på bild 13 visas dialogen ”Lärare” och den information som är relevant. När man jobbar med arbetsfördelningen behöver man växla mellan lärare och grupp och detta görs genom att byta flik. Övre delen av fönstret visar summeringar och analysdata medan nedre delen visar detaljer. Det skall vara möjligt att på ett snabbt sätt förflytta sig från analysläge till uppdateringsläge. Detta kunde förverkligas med en ”drag-and-drop” funktion. Till höger på övre delen av dialogfönstret finns några uppdateringsikoner inramade för lärare, grupp och SPF. Genom att från en detaljrad i nedre delen av dialogfönstret dra (drag) en viss gruppkod eller en studieperiodskod och föra den ovanpå (drop) rätt uppdateringsikon kunde användaren lätta förflytta sig till ett uppdateringsfönster för vald grupp eller SPF. Uppdateringsfönstret skulle öppnas i ett nytt fönster och efter genomförd uppdatering skulle fönstret stängas och användaren återvända till den

ursprungliga dialogen. Detta är ett förslag till flexibel lösning. Det återstår att se om detta är möjligt att förverkliga senare i realiseringsskedet.

Denna beskrivning bör ses som en första ytlig analys av användargränssnittet och funktionaliteten. Det krävs ännu många diskussioner och genomgångar med programmeraren innan denna modell är realiseringsbar.

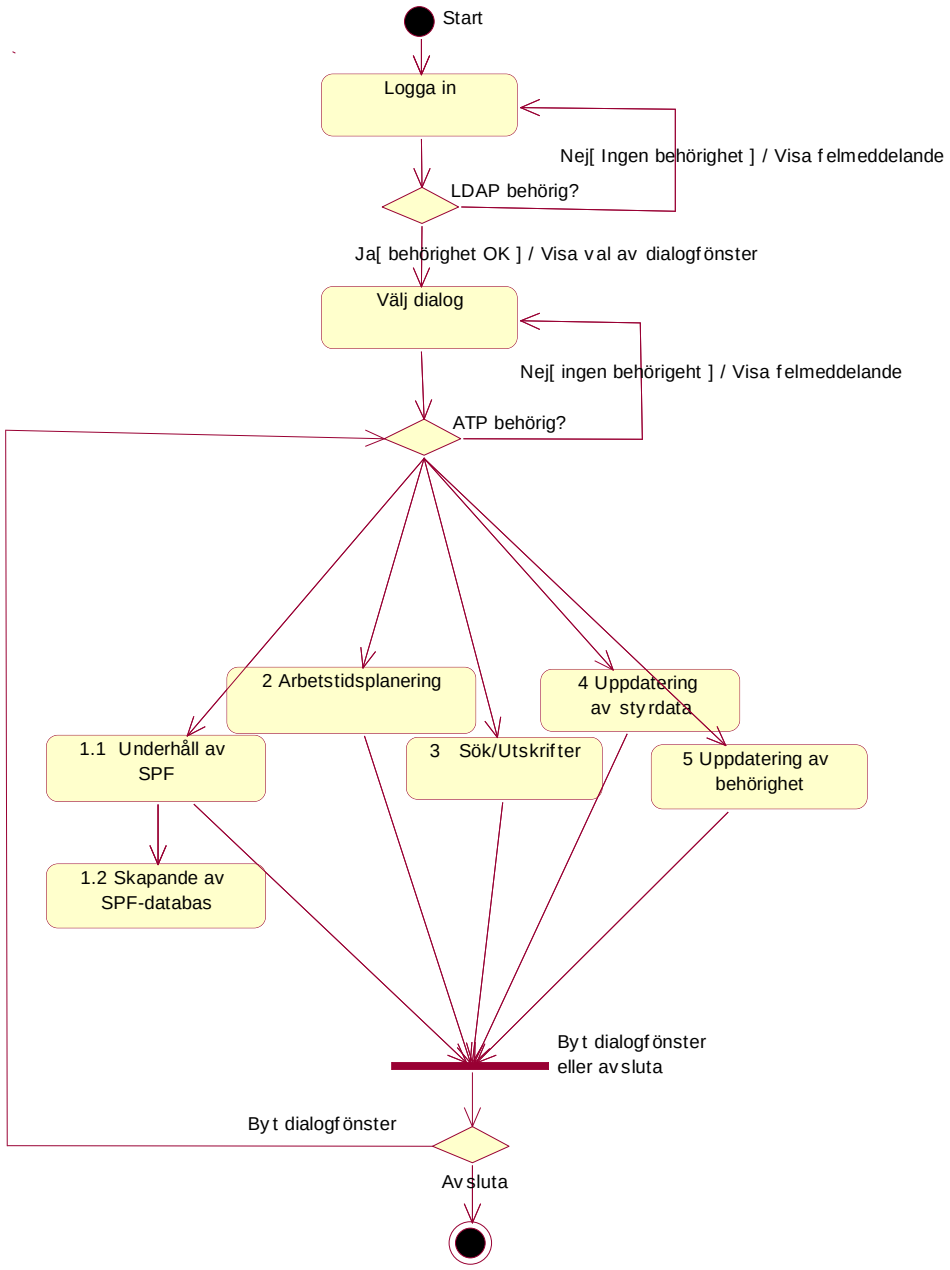


Bild 12. Navigeringsdiagram.

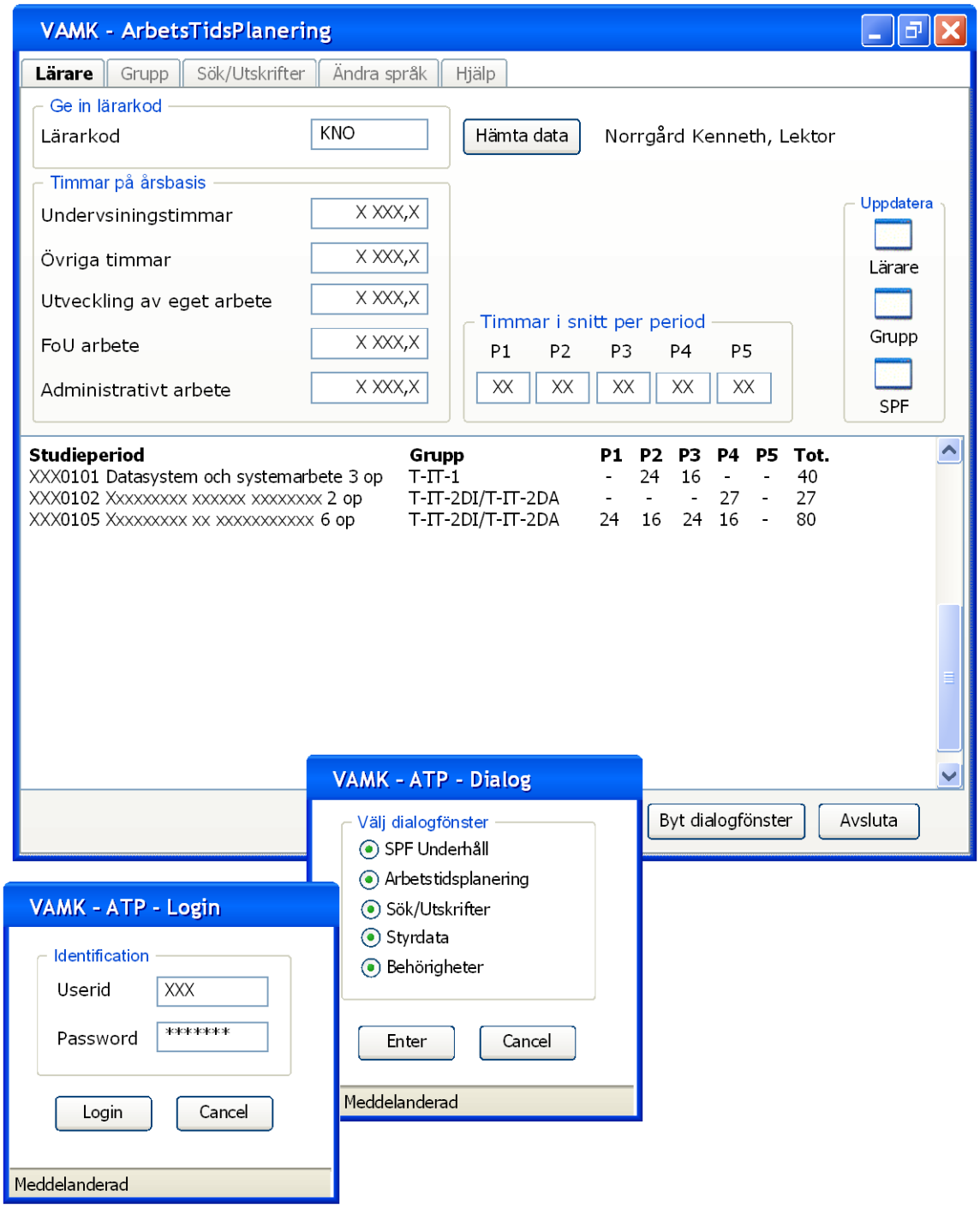


Bild 13. Exempel på dialogfönster för inloggning och analys av lärare.

3.3. Analys av problemområdet

Detta kapitel innehåller analysen av problemområdet. Som tidigare nämnts är målet att skapa en relationsdatabas som lagrar nödvändig information för att systemet skall fungera.

3.3.1. Klassdiagram

Syftet med klassdiagrammet är här att presentera den tänkta databaslösningen. Utgångspunkten var att den slutgiltiga databasen skulle vara en relationsdatabas varför klassdiagrammet påminner om ett ER-schema. Alla klasser har tilldelats stereotypen «table». Attributen som angetts kommer att bli kolumner i tabellerna.

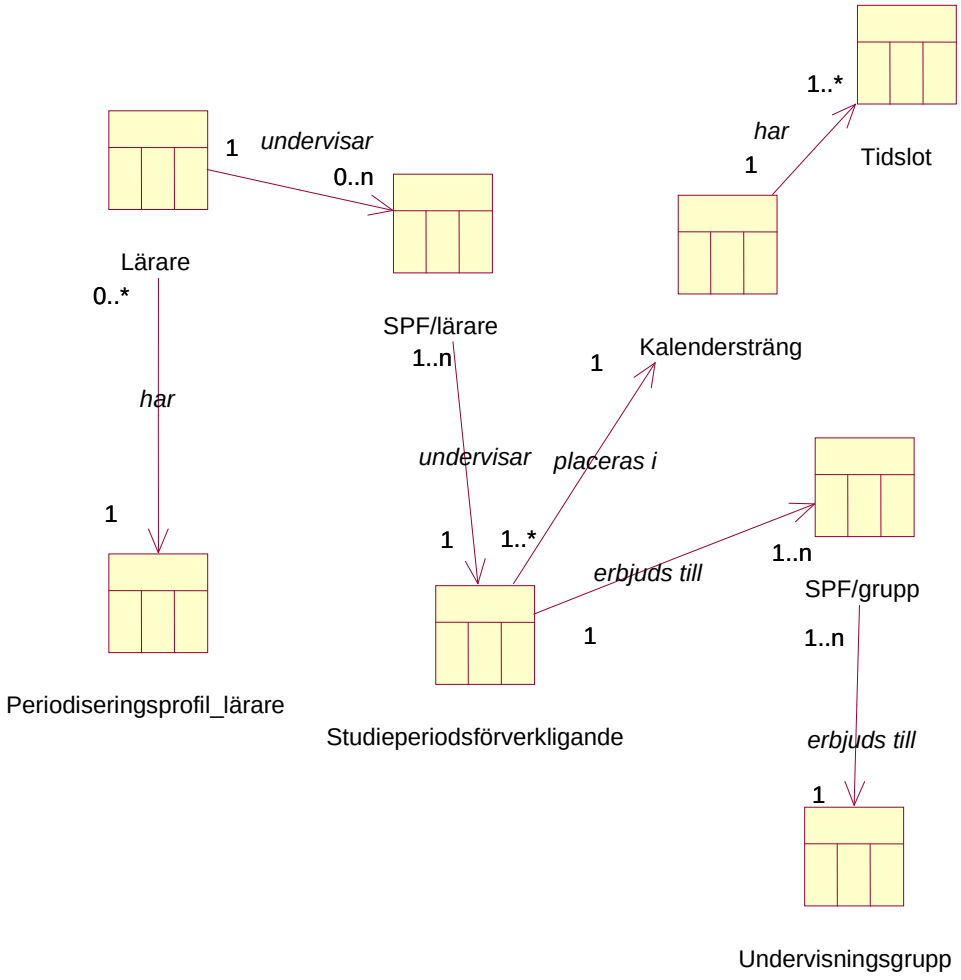


Bild 14. Klassdiagram/ tabellstruktur.

Multipliciteten har angetts i relationen mellan tabellerna och därmed kan detta diagram ligga direkt som underlag för skapandet av databasen i det senare designskedet.

3.3.2. Analys av klasser/ tabeller

Nedan finns en analys av varje tabell som presenteras på bild 14. Dessa bildar databasen. Alla tabeller har namngivits och dessutom har tabellens ansvar och innehåll analyserats. Primärnyckel har angivits med symbolen * och främmande nyckel med symbolen +.

Tabell: Lärare
Ansvar: Ansvarar för lagringen av grunddata om lärarna.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*Lärarkod	Text	Lärarens initialer, t.ex. KNO
Efternamn	Text	
Förnamn	Text	
Titel	Text	
Avdelning	Text	Anger lärarens hemavdelning, t.ex. TK
Telefonnr	Alfanumerisk	
E-post	Text	e-postadresskontroll
+ProfilID	Numerisk	Kopplat till tabellen Periodiseringsprofil

Tabell: Periodiseringsprofil
Ansvar: Lagrar standardiserade periodiseringsprofiler per lärare och styr optimeringen av periodiseringen.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*ProfilID	Numerisk	Unik kod, t.ex. 01 – 20
Profilnyckel	Numerisk	Periodiseringsprofilen styr lärarens planerade periodisering för nästkommande läsår. Periodiseringsnyckeln är 5 siffrig, en siffra för varje

	period som är fem till antalet. Innebörden i koden är: 1=lätt, 2=normal, 3=tung. Summan av kodens fem siffror borde bli 10, t.ex. profil=22222 betyder en jämn periodisering; profil=33220 betyder tung höst, normal vår men inget i period 5; profil=22321 betyder normal höst tung period 3 medan det sedan blir lättare.

Tabell: Studieperiodsförverkligande
Ansvar: Lagrar all grundinformation om studieperiodsförverkliganden. Denna tabell är den viktigaste och mest centrala i datasystemet.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*SPF-kod	Alfanumerisk	Unik kod som följer den logik som nu finns i Winha, t.ex. TDP0101
SP-kod	Alfanumerisk	Studieperiodens kod. En unik kod för studieperioden som bör överensstämma med den kod som grundats i Winha.
Benämning	Text	Studieperiodens namn
Omfattning	Numerisk	Studieperiodens omfattning i antal studiepoäng
Studieguide	Numerisk	Studieguidens versionsnummer = årtal, t.ex. 07
Språkkod	Text	Undervisningsspråk, t.ex. FIN
+SPF/ grupp	Kod	Här anges den unika kod som via tabellen SPF/ grupp kopplar ihop en viss SPF med en viss undervisningsgrupp
Närtimmar	Numerisk	Antal närtimmar totalt
Periodisering	Numerisk	Information om hur många timmar som skall placeras i varje period. Perioderna är fem till antalet.
Virtuell andel	Numerisk	Andel virtuell undervisning, anges i procent Tillåtna värden är 25%, 50% 75% och 100%.
+SPF/ lärare	Kod	Här anges den unika kod som via tabellen SPF/ lärare kopplar ihop en viss SPF med en viss lärare

Timfaktor-kod	Alfanumerisk	Kod som styr uträkningen av hur många årstimmar som skall räknas ut för ett visst SPF. Detta kan variera beroende på till vilken typ av utbildning studieperioden tillhör. Antal årstimmar beräknas enligt formeln: studiepoäng * timfaktor. Här anges en kod och timfaktorn hämtas från styrdatatabellen.
Pfaktor-kod	Alfanumerisk	Kod som styr vilken planeringsfaktor som skall användas för denna lärare för detta SPF. För en ny studieperiod får en lärare mer planeringstid är för en redan inarbetad. Här anges en kod och planeringsfaktorn hämtas från styrdatatabellen. För att få totala antalet årstimmar för en SPF studiepoäng * timfaktor * planeringsfaktor.
Rumskod	Alfanumerisk	Uppgifter om vilket utrymme som önskas. Alla utrymmen har en egen unik rumskod.
Kommentar 1	Lång text	Kommentarer från föregående läsår
Kommentar 2	Lång text	Kommentarer för följande läsår

Tabell: Undervisningsgrupp
Ansvar: Lagrar grundinformation om de undervisningsgrupper som skall erbjudas studieperioder.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*GruppID	Alfanumerisk	Unik kod som identifierar en viss undervisningsgrupp, t.ex. T-TK-1-1 Den skall överensstämma med Winha.
Gruppnamn	Alfabetisk	Beskriver gruppen, t.ex. Tietojenkäsittely 1.vuosi
Antal	Numerisk	Anger antal studerande i studiegruppen

Tabell: SPF/ lärare
Ansvar: Kopplar ihop studieperiodsförverkligande med lärare.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*NyckelID	Numeriskt	En unik kod som identifierar en viss rad som kopplar ihop en lärare med ett SPF
+SPF-kod	Kod	Kopplat till SPF-tabellen
+Lärarkod	Kod	Kopplat till lärar-tabellen

Tabell: SPF/ grupp
Ansvar: Kopplar ihop studieperiodsförverkligande med undervisningsgrupp.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*NyckelID	Numeriskt	En unik kod som identifierar en viss rad som kopplar ihop en grupp med ett SPF
+SPF-kod	Kod	Kopplat till SPF-tabellen
+GruppID	Kod	Kopplat till undervisningsgrupp-tabellen

Följande två tabeller innehåller information som behövs först i det skedet då studieperiodsförverkligandet skall schemaläggas. När schemaläggningen sker kommer de lärare och de grupper som är kopplade till ett visst SPF att få sina scheman bestämda i kalendern via tabellen tidslot. En kalendersträng kan ha en till många tidslots medan en viss tidslot kan gälla bara en viss kalendersträng.

Tabell: Kalendersträng
Ansvar: Innehåller information om en viss sträng.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*SträngID	Numerisk kod	En 6-siffrig unik kod som identifierar en sträng, t.ex. 070101. Logiken i koden är följande: `pos. 1-2` = läsår, t.ex. 07 betyder läsåret 07-08 `pos. 3-4` = period, t.ex. 01 betyder period 1 `pos. 5-6` = strängID, t.ex. 01 betyder sträng 1
+SPF-kod	Kod	Kopplat till SPF-tabellen

Beskrivning	Lång text	Här beskrivs placeringsreglerna för just denna sträng, t.ex. måndagar 8:15-10:00 och torsdagar 8:15 – 10:00. Det är möjligt att bygga upp en egen logik för hur strängarna skall placeras i kalendern med hjälp av tabellen tidslot.

Tabell: Tidslot
Ansvar: Innehåller information om de tider som är möjliga att användas vid schemaläggningen.
Innehåll:

Attribut	*Typ*	*Beskrivning*
*TidslotID	Numerisk	En unik nyckel för varje tidslot. Denna nyckel kan genereras automatiskt av systemet
Datum	Datum	Anger ett visst datum, t.ex. 27.8.2007
Klockslag start	Tid	Anger starttiden för denna slot
Klockslag slut	Tid	Anger sluttiden för denna slot
+SträngID	Numerisk kod	Kopplar en tidslot till en kalendersträng

Dessutom skall det finnas en eller flera tabell som skall innehålla styrdata för summeringar och uträkningar av årstimmar per lärare och per undervisningsgrupp. Tabellerna används av olika program som behöver parameteruppgifter för uträkningar. Dessa tabeller är användarstyrda och uppdateras av planeraren. Tabellerna finns inte med i klassdiagrammet på bild 14, men skall finnas tillgänglig för de program som behöver dem.

Styrdatatabeller
Ansvar: Innehåller parameterdata som styr uträkningar och summeringar i olika program.
Innehåll:
Innehållet i tabellerna är ännu inte analyserade i detalj. Den slutliga lösningen görs först vid realiseringen. Tabellerna skall innehålla åtminstone nedanstående styrdata.

Antal veckor per period
För att räkna ut en lärares eller en undervisningsgrupps genomsnittliga antal timmar per period behövs uppgift om antalet veckor per period.

Timfaktor
För att räkna ut en lärares årstimmar utifrån ett visst SPF behövs en timfaktor. Den variera beroende på till vilken typ av utbildning en viss studieperiod tillhör. För varje timfaktor-kod som finns i SPF-tabellen skall det finna en motsvarande timfaktor som kan användas som multiplikator.

Planeringsfaktor
För att räkna ut den totala tiden som en lärare skall få för en studieperiod. För en ny studieperiod får en lärare mer planeringstid är för en redan inarbetad. För varje pfaktor-kod som finns i SPF-tabellen skall det finna en motsvarande planeringsfaktor som kan använda som multiplikator.

3.3.3. Tillståndsdiagram

Det väsentligaste objektet i denna analys är studieperiodsförverkligandet. Ett tillståndsdiagram beskriver de tillstånd ett objekt kan vara i samt övergångar till andra tillstånd. Bild 15 visar en analys av de tillstånd som ett SPF har under sin livscykel.
Ett SPF skapas för nästa läsår på basen av de studieplaner som föreligger. För att kunna förverkligas bör en ansvarig lärare utses och periodisering bestämmas. När förverkligandet schemalagts och anmälning till förverkligandet påbörjats borde inga uppdateringar vara tillåtna längre. Efter att förverkligandet är avslutat blir SPF arkiverat och kommer därefter att ligga till grund för nästa års planering.

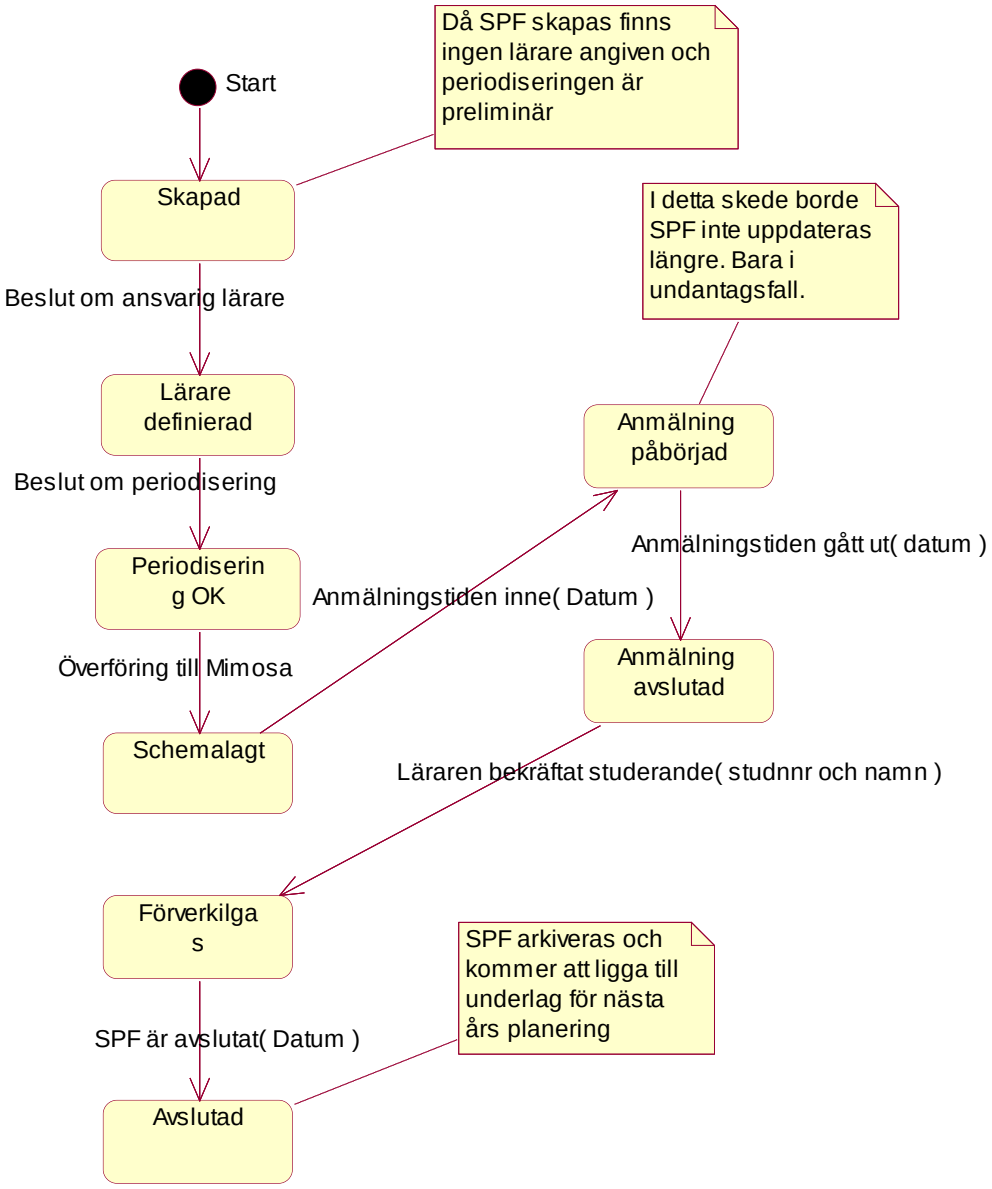


Bild 15. Tillståndsdiagram för ett studieperiodsförverkligande.

3.4. Kommentarer till analysen

Analysen syfte är att bestämma systemkraven och att förstå ett system, dess omgivning och villkoren för dess implementering (Mathiassen m.fl. 2001: 17). Man kan säga att analysens uppgift är att modellera ett system och få fram en överblick över det kommande datasystemet. En analys innehåller i allmänhet många förenklingar och begräsningar och så är fallet även med denna analys. Många aspekter och beskrivningar är fortfarande öppna för diskussion. Denna analys torde i alla fall ge underlag för fortsatta diskussioner om realiseringen av systemet.

Nästa steg i projektet vore att diskutera analysen med kommande användare och testa de lösningar som presenterats. För att få en bra återkoppling från användare vore det på sin plats att rita upp fler exempel på dialogfönster och få dessa kommenterade. I den efterföljande designen borde ställning snabbt tas till den tekniska plattformen och till vilket programmeringsverktyg som skall användas. Analysen har inte fastslagit vilken teknisk lösning som skall användas för databasen utan bara fastställt att det är fråga om en relationsdatabas. Dessa beslut har överlåtits till designen. Ett viktigt mål i analysen har varit att skapa en treskiktsarkitektur som bildas av gränssnitts-, funktions- och modellkomponenten. Detta är viktigt att notera när valet av programmerings- och databasverktyg görs.

Under analysen har vissa aspekter och krav dykt upp som nödvändigtvis inte fanns med i den ursprungliga kravspecifikationen. Dessa bör noteras i den fortsatta utvecklingen.

1. Dialogerna skall fungera på två språk, nämligen finska och svenska. Språkvalet styrs av användarens profil men kan ändras under sessionens gång om så önskas.
2. Analysen har nu fokuserat på endast undervisningstimmar, men önskemålet framöver kommer att vara att även ta med övriga timmar såsom FoU-timmar, projekttimmar och övrigt arbete.
3. Det borde finnas en koppling till lönesystemet så att de timmar som uppdateras och underhålls i ATP kunde överföras till lönesystemet på något sätt, åtminstone i utskriftsform.

4. ATP-databasen ligger till underlag för schemaläggningen och det bör finnas möjlighet att exportera data från ATP till Mimosa som används för schemaläggningen.
5. Åtminstone grundperiodiseringen i olika perioder (P1 – P5) bör kunna utföras i ATP-systemet. Periodiseringsdata borde vara överföringsbart till Mimosa för vidare bearbetning.
6. ATP och Winha (systemet för administration av studerande) borde vara hårt knutna till varandra. Alla SPF som skapas i ATP-systemet skall också finnas i Winha eller tvärtom. Rutinerna för hur ATP och Winha skall koordineras borde ses över i organisationen.
7. Analysen har fokuserat på hur själva planeringen av arbetstiden skall fungera men inte på vilken typ av analyser eller rapporter som kunde produceras. Detta borde diskuteras i det fortsatta arbetet.
8. Under analysen har också diskuterats möjligheterna att ATP-systemet kunde användas till att även rapportera in utförda timmar. Därmed kunde jämförelse göras mellan planerade timmar med verkliga timmar. Framtiden får utvisa ifall detta är ändamålsenligt.

Denna analys kan nu ligga till underlag för beslut om en vidareutveckling av arbetstidsplaneringen. I nuläget fungerar arbetstidsplaneringen i Excel och logiken är inbyggd i olika arbetsark med hjälp av formler och makron. Det nuvarande systemet fungerar nog och producerar de rapporter som behövs men endast för den egna enheten. Det finns ett uttalat behov av att förenhetliga arbetstidsplaneringen inom hela Vasa yrkeshögskola och då räcker inte Excel till som verktyg. Detta kunde vara början till ett gemensamt arbetstids-planeringssystem som skulle betjäna hela yrkeshögskolan.

4. ANVÄNDBARHETEN AV UML

Det har tidigare konstaterats att UML är ett visuellt språk som används för att specificera, konstruera och dokumentera ingående delar i ett system. UML är ett generellt modelleringsspråk i den mening att det lämpar sig för olika verksamhetområden och för olika tekniska plattformar. Med ett modelleringsspråk beskriver man modeller. En modell är en förenklad beskrivning av det planerade systemet. Modeller används för att kommunicera med andra personer i ett projekt. Detta sker genom kommunikation face-to-face, via själva designverktyget samt genom läsning av olika typer av dokument. En kravspecifikation används t.ex. för att kommunicera mellan användare och utvecklare och ett analysdokument används för kommunikation mellan planerare och designers. Modellering är en skapande process som sker i huvudet på planeraren. Man samlar in information om problemområdet och utifrån en given målbeskrivning söker man fram olika alternativa modeller och systemlösningar. När modellerna och lösningarna börjar kännas stabila skall de diskuteras, förbättras, kompletteras och dokumenteras. Det talade och skrivna språket är ett sätt att diskutera och dokumentera designlösningar, men användningen av visuella symboler underlättar oftast informationsutbytet mellan personer i projektet.

Det är här som fördelarna med UML kommer in i bilden. Den första hypotesen som detta arbete presenterade antydde att UML är en beskrivningsteknik som är lätt att förstå, lätt att använda och som resulterar i användbara dokument. UML består av en stor uppsättning element och regler för hur elementen kopplas samman. Fyrkanter, streck, pilar och andra symboler fungerar ofta utmärkt för att beskriva mycket komplexa strukturer och samspel. Elementen i UML har en definierad syntax och semantik, vilket gör att det kan kallas för ett språk. När man bygger ett datasystem så gäller det att få fram en modell som representerar det planerade systemet väl. För att visualisera modellen använder man UML-diagram. Diagrammen i sig är inte det centrala utan det är modellen som skapas när man arbetar med diagrammen. En UML-modell består av bestämda element t.ex. aktörer och användningsfall. Ett visst bestämt UML-diagram visar modellen från en viss synvinkel. UML är ett flexibelt och mångsidigt designspråk som kan användas för alla typer av modelleringsbehov som finns vid utvecklandet av system. Att lära sig att använda UML-element

och att tyda och rita UML-diagram är relativt lätt och dokumentationen är användbar.

Enligt min mening uppfyller UML det som krävs av ett beskrivningspråk i ett projekt. Man kan på ett lätt och överskådligt sätt beskriva en modell av ett datasystem samt använda UML-diagrammen och analysdokumentationen i kommunikationen mellan projektmedlemmarna. UML är egentligen inte ett verktyg för programmeraren även om olika caseverktyg, exempelvis Rational Rose Enterprise, erbjuder koppling mellan UML och objektorienterade språk. Det är snarare ett språk för kommunikation mellan planerare och programmerare för att förmedla den modell som system skall byggas på.

4.1. Objektdiagram

Objektdiagram är speciellt användbara när man skissar på lösningar och vill få en konkret bild av ett system. Det är lättare att diskutera om verkliga objekt än om klasser. Diagrammet kan vara bra att använda i dialogen med personer som inte är insatta i objektorientering. Ett rent klassdiagram kan bli för abstrakt att förstå. I detta arbete har objektdiagrammet använts som en modell för skissning och presentation av den tänkta 3-skiktsarkitekturen. Med hjälp av stereotyperna «boundary», «control» och «entity» har objekten kategoriserats och fått sin tillhörighet i antingen gränssnitts-, funktions- eller modellkomponenten.

4.2. Användningsfallsdiagram

Användningsfallsdiagram används för att ge en överblick av de användningsfall som finns och av deras kopplingar till aktörer. Diagrammet kan även användas för att visa avgränsningen av det planerade datasystemet. Systemet skall ”ta hand om” just de användningsfall som angetts och övriga skall lämnas utanför. Det centrala är analysen av händelseförloppet i användningsfallen. Användningsfall används för att beskriva kraven på systemen ur beställarens perspektiv och kan med fördel användas i kommunikationen med beställaren. Genom att analysera texten i händelseförloppen kan man hitta väsentliga objekt

och utifrån dessa starta diskussionen om klasser och associationer (läs: den kommande databaslösningen).

4.3. Klassdiagram

Klassdiagrammet är ett strukturbeskrivande diagram som används för att visa en statisk bild över modellelement och deras inbördes relationer. Exempel är klasser, klassers innehåll och relationer mellan klasser. Klassdiagram kan användas i många olika sammanhang där strukturelement och deras relation ska modelleras. Klassdiagrammet är det mest centrala diagrammet i UML. De mekanismer som används för klasser fungerar ofta på samma sätt för andra UML-element. Klassdiagrammets syfte i detta arbete har varit att beskriva modellkomponenten, d.v.s. databasen. Alla klasser har getts stereotypen «table» och därmed har innebörden ändrats från att representera en klass till att representera en tabell. Följden blir underförstått att ett objekt, som vanligtvis är förekomsten i en klass, nu blir presenterat som en rad i en tabell. Även den grafiska notationen har ändrats. Se kapitel 2.3. Detta är en av fördelarna med UML. Målet med utvecklingen har varit att göra det till ett flexibelt, utbyggbart och mångsidigt språk som skall passa alla typer av modelleringsbehov.

I den andra hypotesen som framlades i början av arbetet antyddes att en objektorienterad metod med UML som beskrivningsteknik skulle ge tillräckliga underlag för att definiera, dokumentera och presentera en modell av datasystemet. Att man har möjlighet att använda stereotyper, d.v.s. att planeraren tillåts ge vissa element bestämda nya betydelser/ roller, är en viktig egenskap som ger möjlighet att applicera UML på sitt eget projekt. Som tidigare nämnts är klassdiagrammet det mest centrala vid beskrivningen av det planerade systemet. Kapitel 3.3 i detta arbete visar hur klassdiagrammet tillsammans med analyserna av tabellerna kan dokumentera och presentera en modell av det planerade datasystemet.

4.4. Tillståndsdiagram

Tillståndsdiagram beskriver de tillstånd ett objekt eller en interaktion kan vara i samt eventuella övergångar till andra tillstånd. Vanligtvis används diagrammet för att beskriva tillstånd och tillåtna förändringar i tillstånd för en klass eller en grupp av samverkande klasser. Ett annat användningsområde är beskrivningen av en användardialog med dialogrutor (fönster) som tillstånd. Bild 12 visar ett exempel på detta. Tillståndsdiagrammet har i UML 2.0 ändrat namn från ’State Chart Diagram’ till ’State Machine Diagram’ och därmed fått utökade användningsmöjligheter. I detta arbete har tillståndsdiagrammet dock använts och tolkats enligt definitionerna i UML 1.0.

4.5. Övriga UML-diagram

I detta arbete har endast en del av de UML-diagram som står till buds använts. Att ha friheten att använda bara vissa diagram i sitt projekt, kan ses som en del av flexibiliteten i UML. Den grundläggande tanken är att ta med de diagram som tillför analysen någonting och inte behöva rita diagram bara för diagrammets skull. Det finns alltså ytterligare ett antal diagram som kan användas i motsvarande analyser. Tabell 2 visar exempel på ett antal diagram som finns med i UML version 2.0. I kapitel 5 tabell 4 finns en komplett lista på alla 13 diagram som finns upptagna i UML 2.0.

Tabell 2. Användbara UML-diagram (The Unified Modeling Language, Sparx Systems).

Diagram	**Kort förklaring**
Aktivitetsdiagram (Activity Diagram)	Beskriver en högre nivå av verksamhetsprocesser som ofta beskriver dataflöden. Används även för att modellera komplex logik inom system och kunde t.ex. visa en analys av ett komplext användningsfall.
Paketdiagram (Package Diagram)	Ett diagram som beskriver hur modellelement är organiserade i paket och innehåller endast paket och deras inbördes relationer. Diagrammet används för att skapa en övergripande beskrivning av hur element är grupperade.

(Tabell 2, forts)

Komponentdiagram (Component Diagram)	Används för att beskriva de komponenter som tillsammans utgör en applikation, ett system eller en del av ett system. Det som beskrivs är vilka komponenterna är, hur de är organiserade samt vilka beroenden via synliga interface som finns. Används ofta för att visualisera komponentbaserade system och för att visa på vilka delar som är dynamiskt utbytbara.
Kommunikations-diagram (Communication Diagram)	Används för att visa hur anrop sker mellan objekt. Det centrala för diagrammet är att visa strukturen och strukturens koppling till hur kommunikation mellan objekt sker. Anropen sekvensnumreras för att visa i vilken ordning anropen sker. (Detta diagram kallades tidigare för Collaboration Diagram i UML 1.0)
Sekvensdiagram (Sequence Diagram)	Används för att beskriva en sekventiell logik, d.v.s. ordningen för meddelanden som skickas till objekt beskrivs. Sekvensdiagram har en hård koppling till hur program exekveras. För att modellera realiseringen av användningsfall är sekvensdiagram ett av de vikigare verktygen. Kommunikationsdiagram beskriver motsvarande information men visualiserar den på ett annat sätt.

5. NÅGRA REFLEKTIONER

Det har varit både intressant och lärorikt att genomföra denna objektorienterade analys och utforska de möjligheter som Unified Modeling Languange ger. UML har uppdaterats till version 2.0 i oktober 2005. Det finns kompletteringar och nya diagram i UML 2.0 som erbjuder ännu fler användningsmöjligheter. Att en större och mer genomgripande uppdatering inte varit nödvändig, förrän sju år efter det att version 1.0 blivit industristandard, tyder enligt min åsikt på att UML redan från början varit genomtänkt.

Det som slår mig är flexibiliteten i beskrivningsspråket. Det finns goda möjligheter att anpassa diagrammen till sina egna projekt. Man behöver t.ex. inte ta med alla typer av diagram och projektanpassningar kan göras genom att definiera stereotyper och ge vissa elementen eller relationer egna betydelser. Elementet “Notes” ger möjlighet till fria kommentarer vilket i många fall kan öka förståelsen för analysen. Det är att jämföra med när en programmerare lägger in kommentarrader i sin kod eller då man använder gula “post-it-lappar” på kontoret. Trots denna flexibilitet har UML en strikt syntax att följa och varje element har sin bestämda betydelse. De grafiska representationerna känns naturliga och syntaxen är lätt att lära för en person som har tidigare erfarenhet av systemutveckling. Ett system är ju ofta komplext att överskåda och styrkan i UML är att man med ett enskilt diagram har möjlighet att fokusera på problemet (läs: systemlösningen) från en viss synvinkel. När man bygger ett datasystem så gäller det att skapa en bra modell som representerar det planerade systemet. UML-diagrammen visualiserar planeringen och modellen “växer fram” vartefter man arbetar med diagrammen.

En viktig aspekt i sammanhanget är hur lätt det är att underhålla de artefakter som skapas i ett projekt. En artefakt är ofta ett utskrivet dokument men kan lika gärna vara en modell och/ eller ett UML-diagram skapat i ett caseverktyg, t.ex. ett användningsfallsdiagram i Rationa Rose Enterprise. Det går alldeles utmärkt att rita UML-diagram för hand på ett rutigt papper för att få igång en diskussionen om olika systemlösningar. Då tjänar UML som det gemensamma språket mellan olika planerare eller mellan planerare och designer. Jag är helt säker på att detta gör processen snabbare och ökar den gemensamma

förståelsen för problemområdet. Det kan dock vara bra, för att inte säga nödvändigt, att använda ett caseverktyg för att dokumentera analysen/ diagrammen elektroniskt. Detta torde inte förorsaka problem för något projekt, även om budgeten vore snäv. Det finns gott om UML-modelleringsverktyg att ladda ner gratis via Internet. Först när diagrammet kan underhållas med hjälp av dator, är det troligt att det kommer att fungera som det är tänkt. Då kan diagrammen uppdateras vartefter analysen framskrider. Ett bra caseverktyg kombinerat med en vettig versionshantering är en bra förutsättning för ett lyckat projekt. Om detta sedan förstärks med en bra uppgjord rollfördelning där varje projektmedlem vet sin uppgift och har en klar bild av hur olika UML-diagram hänger ihop så finns alla ingridienser till ett lyckat resultat.

Det finns väldigt mycket material och källor på Internet som behandlar objektorienterad analys och UML. Vissa av dessa källor har utnyttjats i detta arbete och återfinns i källförteckningen. För den som vill läsa mer och utforska möjligheterna med UML presenteras några tips nedan.

Tabell 3 innehåller länkar till några UML tutorials som finns på Internet. Tabell 4 visar en översikt på de 13 diagram som för närvarande ingår i UML version 2.0 (Ambler Scott W 2007).

Tabell 3. Rekommenderade Internetkällor om UML

Förklaring	**Internetlänk**
Practical UML: A Hands-On Introduction for Developers, by Randy Miller. “This tutorial provides a quick introduction to the Unified Modeling Language.”	`http://dn.codegear.com/article/31863`
UML Tutorial. “This tutorial provides a technical overview of the 13 UML diagrams supported by Enterprise Architect. UML 2 semantics are explained in detail in the new UML 2.0 tutorial.”	`http://www.sparxsystems.com.au/uml-tutorial.html`
UML Tutorial - Free Online Course, by Gentleware AG. “All you need to know about Object Oriented Modeling and UML. Learn the fundamentals of object oriented programming.”	`http://www.gentleware.com/fileadmin/media/synergy/Course/index.htm`

Tabell 4. Översikt över de 13 diagram som finns i UML 2.0.

1. Paketdiagram (Package Diagram) - nytt i UML 2.0
2. Klassdiagram (Class or Structural Diagram)
3. Objektdiagram (Object Diagram)
4. Strukturdiagram (Composite Structure Diagram) - nytt i UML 2.0
5. Komponentdiagram (Component Diagram) - kompletterats i UML 2.0
6. Fördelningsdiagram (Deployment Diagram)
7. Användningsfallsdiagram (Use Case Diagram)
8. Aktivitetsdiagram (Activity Diagram)
9. Tillståndsdiagram (State Machine Diagram) - kompletterats i UML 2.0
10. Kommunikationsdiagram (Communication Diagram) - i UML 1.0 kallades detta för samarbetsdiagram (Collaboration Diagram)
11. Sekvensdiagram (Sequence Diagram)
12. Timingdiagram (Timing Diagram) - nytt i UML 2.0
13. Interaktionsöversiktsdiagram (Interaction Overview Diagram) - nytt i UML 2.0

KÄLLFÖRTECKNING

Böcker

Bennett Simon, John Skelton & Ken Lunn (2001). *Schaum's Outlines of UML*. L.E.G.O. Spa, Vincenza Italy, McGraw-Hill International (UK), 328 s.

Fagerström Johan, Pierre Bjurhager, Anders Wallström & Thomas Jönsson (1998). *Objektorientering i hela företaget – en integrerad utvecklingsprocess.* Lund, Studentlitteratur, 328 s.

Kruchten Philippe (2002). *The Rational Unified Process – En introduction*, svenska utgåvan, Edinburgh, Pearson Education Limited, 299 s.

Mathiassen Lars, Andreas Munk-Madsen, Peter Axel Nielsen & Jan Stage (2001). *Objektorienterad analys och design*, andra upplagan, Lund, Studentlitteratur, 501 s.

Fowler Martin & Kendall Scott (1997). *UML Distilled – Applying the Standard Object Modeling Language*, 9th Printing (1998). USA and Canada, Addison-Wesley Longman Inc, 183 s.

Kroll Per & Philippe Kruchten (2005). *The Rational Unified Process Made Easy – A Practitioner´s Guide to the RUP,* Boston, Addison-Wesley Pearson Education Inc, 416 s.

Elektroniska källor

Ambler Scott W. (2007). *Introduction to the Diagrams of UML 2.0* [Online]. Ambysoft Inc. Tillgänglig på World Wide Web: <URL:http://www.agilemodeling.com/essays/umlDiagrams.htm>

Booch, Grady (1998). *The Visual Modeling of Software Architecture for the Enterprise* [Online]. USA, Rational Software Corporation, 1998. Tillgänglig på World Wide Web: <URL:http://www.rosearchitect.com/mag/archives/9810/f1.shtml >

Fägnell Urban (2003). *RUP - En introduktion* [Online]. The Pellesoft Community. Publicerad 05.03.2003. Tillgänglig på World Wide Web: <URL:http://www.pellesoft.se/area/articles/article.aspx?artid=566>

The Unified Modeling Language (UML) [Online]. Sparx Systems' UML Resource Page. Tillgänglig på World Wide Web: <URL:http://www.sparxsystems.com.au/platforms/uml.html>

UML Resource Page. [Online]. Object Management Group, Inc. Tillgänglig på World Wide Web: <URL: http://www.uml.org>

UML Resource Center. [Online]. IBM Rational Software's UML Resource Page. Tillgänglig på World Wide Web: <URL:http://www-306.ibm.com/software/rational/uml>

Vasa yrkeshögskolas hemsida [Online]. Vasa yrkeshögskola. Tillgänglig på World Wide Web: <URL:http://www.puv.fi>