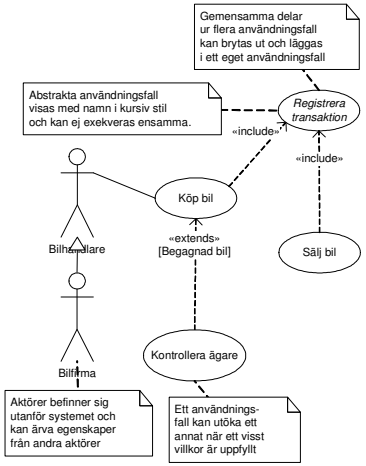
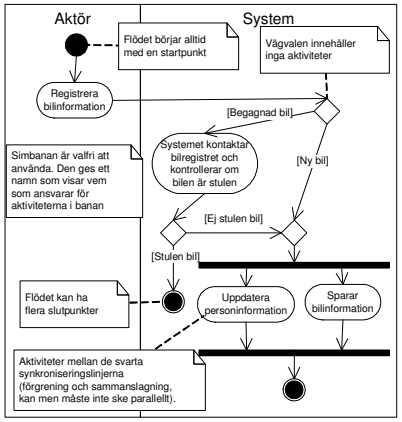
1. Användningsfallsdiagram

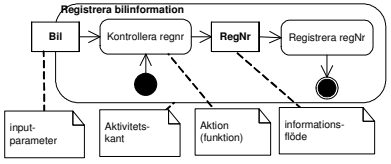


Ett användningsfall representerar en interaktion mellan systemet och en eller flera aktörer som resulterar i nytta för aktören. Användningsfallet används för att ge en övergripande beskrivning av systemets funktionalitet och vilken nytta kunden kan ha av det. Användningsfallet används för att fånga specifika, funktionella krav. Övriga krav dokumenteras i en kompletterande kravspecifikation.

2. Aktivitetsdiagram

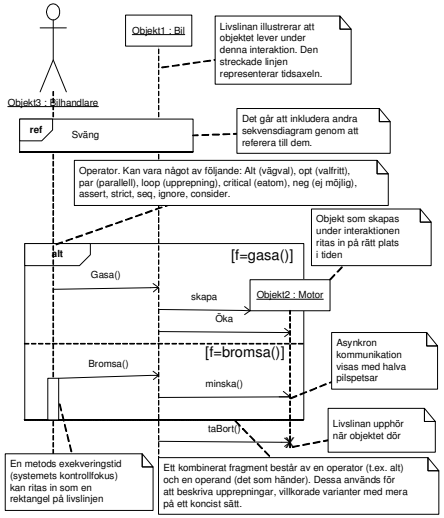


Aktivitetsdiagrammet är ett traditionellt flödesschema. Det kan t.ex. användas för att specificera interaktionen mellan aktör och system i användningsfallet och/eller för att beskriva verksamhetsprocesser. Diagrammets fokus ligger på aktiviteternas sekvens och villkor för att koordinera dessa



En aktivitet innehåller aktioner och dessa kan också visas i diagrammet, tillsammans med dataflödet och in-/utparametrar.

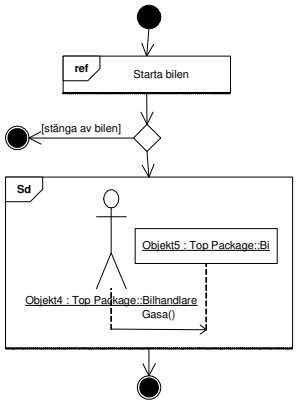
3. Sekvensdiagram



Sekvensdiagrammet används för att visa hur (system)objekt samverkar med varandra. T.ex. för att åstadkomma funktionaliteten i ett användningsfall Returmeddelanden är implicita, men kan också visas som en streckad returpil.

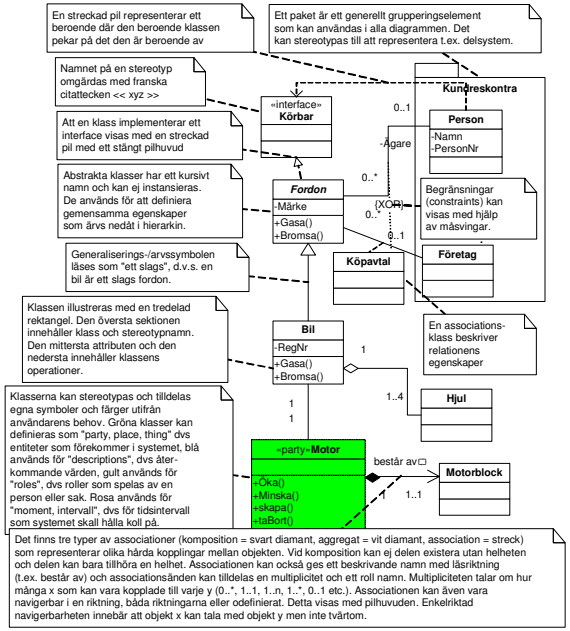
OBS OBS Sekvensdiagram ritas inte för alla interaktioner i systemet utan endast för de "intressanta"

4. Interaktionsöversiktsdiagram

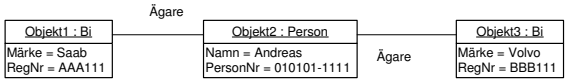


Interaktionsöversiktsdiagrammet är ett slags aktivitetsdiagram där man hänvisar till sekvensdiagram istället för att använda sig av aktivitetsnoder. Antingen så ritas interaktionen in, inuti aktiviteten eller så hänvisar man till ett specifikt sekvensdiagram som beskriver interaktionen. Diagrammet är bra för att visa på samband mellan olika sekvensdiagram genom att koppla ihop dessa med varandra och för att skapa en spårbarhet från användningsfall till design. Interaktionsöversiktsdiagrammet kan använda sig av alla symboler i aktivitetsdiagrammet.

5-6. Klass- och objektdiagram

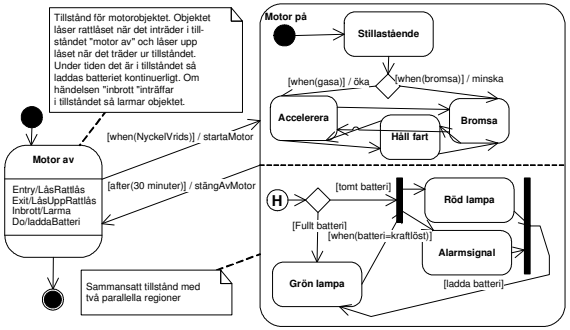


Klassdiagramet är det vanligast förekommande och beskriver en struktur (system, data, information, mål, begrepp m.fl.). Det kan göras på olika nivåer (t.ex. implementationsoberoende-/analysnivå och språkspecifikt/designnivå). Objektdiagrammet exemplifierar klassdiagrammet med en ögonblicksbild.



Objekten visas med tvådelade rektanglar där den översta sektionen innehåller objektnamn+paket+klassnamn. Namnets olika delar är avskilda med kolon. Namnet är också understruket. Den nedre sektionen innehåller objektets aktuella attributvärden. Objekten är sammankopplade med länkar. Objektdiagram är kraftfulla för att "testa tänket" i ett generellt klassdiagram.

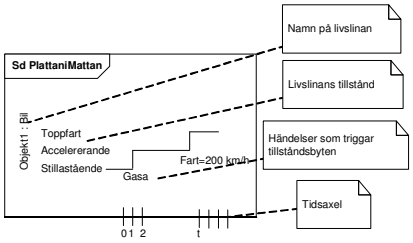
7. Tillståndsdiagram



Tillståndsdiagrammet illustrerar hela eller en del av en livscykel för ett objekt. Ett tillstånd definieras som kombinationen av ett objekts attributvärden och länkar till andra objekt vid ett visst givet ögonblick. En övergång från ett tillstånd till ett annat beskrivs med följande syntax: trigger+villkor+handling. ”When” används för att representera händelsestyrda övergångar och ”after” för tidsstyrda. Tillståndssymbolen har tre sektioner där den översta innehåller tillståndets namn, den mellersta tillståndets interna aktiviteter och den nedersta tillståndets interna övergångar. Entry, exit och do är reserverade namn för aktiviteter och kan ej användas som namn på händelser. Entry-aktiviteter sker vid ingången till tillståndet, exit vid utgången och do sker kontinuerligt under tiden objektet befinner sig i tillståndet.

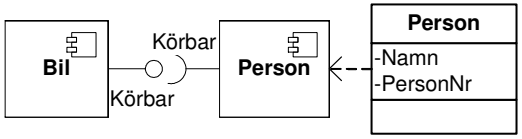
Förgrenings- (fork) och sammanslagningslinjer (join) kan användas på samma sätt som i aktivitetsdiagrammet. Ett H i en cirkel betyder att objektet kommer ihåg vilket deltillstånd det befann sig i senast objektet var i ett sammansatt tillstånd och återvänder direkt till det deltillståndet.

8. Timingdiagram



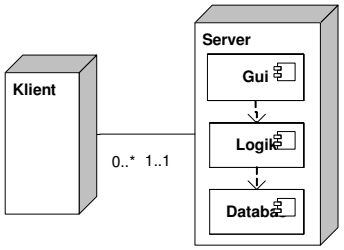
Timingdiagrammet är ett slags interaktionsdiagram där fokus ligger på att visa hur länge ett objekt (en livslina) befinner sig i ett visst tillstånd.

9. Komponentdiagram



En komponent representerar en modulär enhet i systemet som kapslar in sitt innehåll och är utbytbar. Komponenter kan erbjuda ett gränssnitt (interface). Detta illustreras med hjälp av en ”lollypop”, dvs en klubbsymbol. Komponenter som kräver ett gränssnitt kan visa detta med hjälp av gaffelsymbolen ovan. Komponentdiagramet visar på beroenden mellan komponenterna och man kan också peka ut vilka klasser som realiserar vilka komponenter.

10. Deploymentdiagram



Deploymentdiagrammet beskriver systemets hårdvarustruktur. Om man vill kan man även presentera systemets komponenter och visa på vilken hårdvara dessa befinner sig.

Utöver ovanstående diagramtyper innehåller UML ytterligare tre diagram; paketdiagram, kommunikationsdiagram och sammansättningsdiagram (composite structure).

Vart skall jag börja? Beroende på vilken roll man har i projektet använder man sig olika mycket av olika diagram. Utvecklare använder i första hand klass, sekvens och tillståndsdiagram medans verksamhetsrepresentanter arbetar mest med användningsfall, aktivitets- och klassdiagram.

Vart kan jag hitta mer information? På Internet är två bra startpunkter Systemvaruhusets hemsida (www.systemvaruhuset.se) och OMGs umlsida (www.uml.org). Vill man läsa böcker så rekommenderar vi ”UML Distilled” av Martin Fowler och ”UML Reference Manual” av James Rumbaugh. Systemvaruhuset tillhandahåller även kurser i UML och andra områden som är nödvändiga för förutsägbar systemutveckling.

SYSTEMVARUHUSET™

Systemvaruhuset AB Solna strandväg 78 171 54 Solna
Tel:08-50 52 10 42 Fax: 08-50 52 10 10 www.systemvaruhuset.se

Systemvaruhusets UML-lathund

Systemvaruhusets vision är att göra för IT-branschen vad Ford gjorde för bilindustrin! Att förbättra personalens arbetsförhållanden, sänka produktionskostnaden och skapa en förutsägbarhet vad gäller pris, kvalitet och funktionalitet.

Att fritt kunna utrycka sig i C#, C++, Java, C, Ruby, Ajax eller Pascal, att hantera verktyg från Microsoft, Borland eller Rational och att känna till Design Patterns, STL, COM, Active X i Windows, Unix eller OS2 är en självklarhet när så krävs. Men förutsägbar systemutveckling kräver alltid något mer. En förmåga att "dansa med metoder".

Vår kärnkompetens är vår förmåga att driva projekt med ett förutsägbart resultat och att sätta en prislapp på detta. För oss är den tekniska kompetensen bara halva svaret. Den andra halvan är vår förmåga att välja metod, teknik och att prioritera de stora stenarna först.

Vi är ett ungt företag med mycket erfarenhet. Vi startade 2005 och är idag 15 konsulter med kontor i Solna. Vi är dock verksamma från Kiruna i norr till Ringhals i söder. Brukar du betala med kort på bensinstationen och i matbutiken? Då är chansen stor att du redan stött på våra system eftersom vi bl.a. arbetar med intelligenta kort.

Unified Modeling Language är ett modellspråk som vi använder oss av för att designa och dokumentera våra lösningar. UML är en iso-standard (ISO/IEC 19501) och omfattar 13 diagramtyper. Denna lathund beskriver de diagram vi använder mest. Diagrammen presenteras i den ordning som vi vanligen tar fram dem.

UML är en viktigt framgångsfaktor för oss när det gäller att på kort tid bygga förvaltningsbara system och vi delar gärna med oss av vår kunskap inom detta område, i projekt, som mentorer och/eller genom någon av våra utbildningar. Vår vision är ju trots allt att förändra branschen.