Objektorienteeritud modelleerimine.

Objektmudel: Klassid, Objektid ja nende seosed

Loengu eesmärk:

- Ülevaade objektmodelleerimise põhimõistetest
- Klassidiagrammides kasutatavate põhikonstruktsioonide tutvustamine

Sisu

- Objektid ja klassid
- Klassidiagramm
- Kuidas leida klasse ?
- Atribuudid
- Operatsioonid
- Seosed

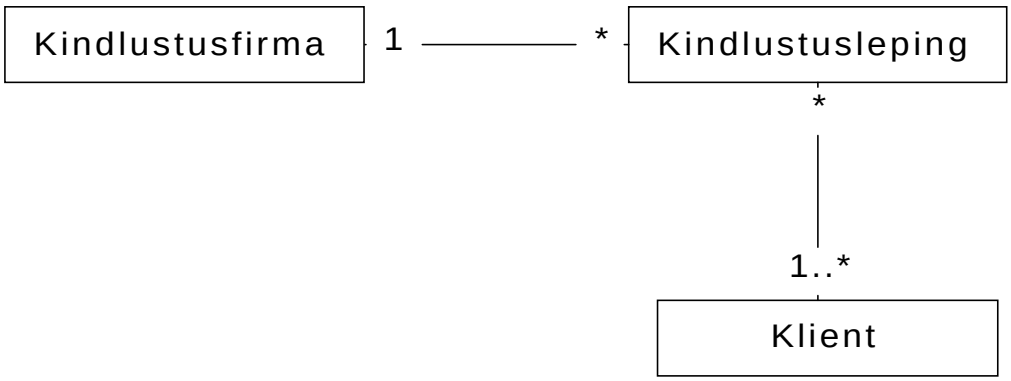
Objektorienteeritud modelleerimises on põhilisteks elementideks *klassid*, *objektid* ning nendevahelised *seosed* . Kui modelleerimise eesmärgiks on tarkvarasüsteemide ehitamine, minnakse objektorienteeritud mudelitelt sujuvalt üle objektorienteeritud programmeerimise mõistetele / konstruktsioonidele, kus klassid ja seosed on teisendatud tegelikuks programmikoodiks.

Objektid ja klassid

Objekt on element, nähtus, asi, millest me räägime ja/või millega tegutseme.
Objekt eksisteerib reaalses maailmas, täpsemalt, meie ettekujutuses sellest maailmast.
Modelleeritavat osa maailmast nimetatakse tavaliselt *objektsüsteemiks* (ka probleemvaldkond, *domeen*)
Objektsüsteemiks võib olla igasugune süsteem, näiteks masin/seade, organisatsioon, ärisüsteem, infosüsteem, tarkvara jne.. Nendes süsteemides käsitletavad objektid (ka info- ja tarkvaraobjektid) on nii või teisiti seotud meie arusaamisega reaalsest maailmast, mis kujuneb seal eksisteerivate objektide struktuuri ja käitumise analüüsi tulemusena.

Klass on objekti tüübi kirjeldus. Objektid on konkreetse klassi liikmed ehk eksemplarid (instance), kusjuures klass kirjeldab ühe objektitüübi omadusi ning käitumist. Klassi moodustavad ühesuguste omaduste ja käitumisega objektid. Objekt on seotud klassiga sarnaselt nagu muutuja on seotud tüübiga tavalises programmeerimiskeeles.

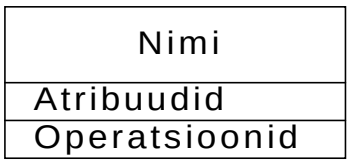
Mudeli üheks põhifunktsiooniks on olla kommunikatsiooni vahend süsteemi erinevate osapoolte (kasutajad / tellijad, arendajad,..) vahel. Äri-, info- jm. süsteemi modelleerimisel tuleks kasutada sellele süsteemile (probleemvaldkonnale, domeenile) iseloomulikke mõisteid. Näiteks kindlustusfirma (info)süsteemi mudel peaks “rääkima” kindlustusäris osalejatele arusaadavas keeles. Pangasüsteemi mudel peaks kasutama panganduse termineid (n. arved, tehingud, saldod,..) jne..



Joonis. Fragment kindlustusfirma mudelist. Üks kindlustusfirma omab palju (0 või enam) kindlustuslepinguid. Üks kindlustuse klient omab mitu (0 või enam) kindlustuslepingut. Üks kindlustusleping on seotud ühe kindlustusfirmaga. Kindlustusleping on seotud paljude (ühe või enama) kindlustuse kliendiga. Joonisel toodud olemeid nimetatakse *klassideks.*

Klassidiagramm

Klassidiagramm on mudeli tüüp, mis esitab süsteemi staatilist vaadet, kasutades klasse ja nendevahelisi seoseid.
Klassidiagramm sarnaneb andmemudelitele, kuid väljendab lisaks infostruktuuridele ka käitumist (klass sisaldab käitumist).
Klassidiagrammi üheks eesmärgiks on defineerida alus (vundament) teistele diagrammidele, kus väljendatakse süsteemi muid aspekte (objektide seisundeid ja objektide koostoimet e. kollaboratsioone väljendatakse dünaamika diagrammidega).
Klassidiagrammi klassi saab otseselt realiseerida objektorienteeritud programmeerimiskeeles (n. Java,C++,..), mis toetab klassi konstruktsiooni.
Klassidiagramm esitab ainult klasse, kuid eksisteerib ka objektidiagramm, kus näidatakse klasside objektieksemplare.



Joonis 2. Klass UML-is. Klass joonistatakse ristkülikuna, mis on jagatud kolme ossa. Neis kasutatav süntaks on programmeerimiskeelest sõltumatu.

Kuidas leida klasse ?

Klasside identifitseerimine on loominguline tegevus, mida tehakse koostöös probleemvaldkonna ekspertidega.
Klassid tulenevad meie arusaamisest antud probleemvaldkonnast ning selle järgi peavad klaasid saama ka oma nime.
Klasside ülesleidmisel võib abiks olla objektide üldine liigitus:

- Käegakatsutavad (füüsilised) objektid: isik, auto, maja,..
- Rollid: tudeng, õppejõud
- Sündmusobjektid (ühe ajaparameetriga): eksam (algusaeg)
- Tegevusobjektid (kahe või enama ajaparameetriga): õppimine (algus, lõpp), õpetamine (algus, lõpp)
- Spetsifikatsioonid: õppeaine

Klasside ülesleidmisel võivad olla abiks standardsed küsimused:

- Millist informatsiooni (asjad, mõisted, sündmused, tegevused) modelleeritavast süsteemist on tarvis salvestada või analüüsida?
- Millised on välissüsteemid, millega modelleeritav süsteem suhtleb? Millised klassid võiksid neid välissüsteeme esindada modelleeritavas süsteemis ?
- Kas me saame kasutada mudeleid, lahendusi või komponente oma varasematest projektidest, kolleegide töödest, muudelt tootjatelt ? Milliseid klasse me sealt kasutaksime?
- Milliseid seadmeid meie süsteem peab käsitlema? Iga süsteemiga ühendatud tehniline seade võib anda klassi, mis seda seadet käsitleb.
- Kas meil on organisatsiooniüksusi? Organisatsioon esitatakse klasside kaudu, eriti ärimudelites.
- Milliseid rolle ärisubjektid täidavad? Rolle võib vaadata klassidena: klient, töötaja, kasutaja,..

Kõige enam toetavad klasside leidmist teised (Use Case ning

dünaamika-) diagrammid. Ilma süsteemi eesmärke ja funktsionaalsust modelleerimata pole võimalik otsustada konkreetsete klasside vajalikkuse üle.

Atribuudid

Klassi atribuudid kirjeldavad objektide omadusi.
Süsteemi eesmärk ja funktsionaalsus mõjutavad klassi kirjeldavaid atribuute. Kirjeldatakse ainult modelleeritava süsteemi kontekstis huvi pakkuvaid atribuute.

Atribuudil on tüüp:
- Primitiivsed tüübid: integer, Boolean, real, point, area, enumeration
- Spetsiifilised (programmeerimiskeele jaoks)
- Teised klassid võivad olla atribuudi tüübiks

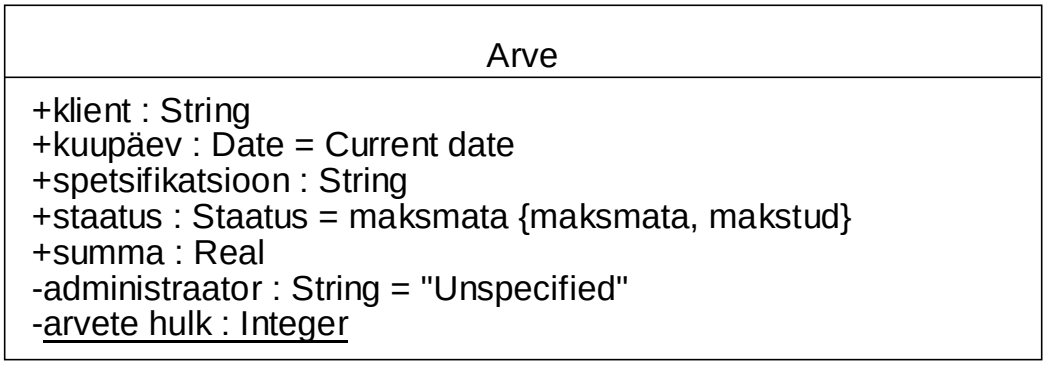
Atribuudi nähtavus (visibility):
- Public (+): nähtav ja kasutatav väljaspoolt antud klassi
- Private (-): nähtav ja kasutatav ainult antud klassi sees
- Protected: kasutatakse koos üldistamise/pärimise seosega

Atribuudil võib olla vaikimisi väärtus.

Saab defineerida klassi skoobiga atribuute (muutujaid), mida jagavad kõik antud klassi objektid. Sellised atribuudid on alla joonitud.
Omadusstringi (property string) võib kasutada atribuudi lubatud väärtuste esitamiseks, eriti loetelutüüpide, nagu värv, staatus, suund, korral.

Atribuudi formaalne süntaks:

visibility name : type-expression = initial-value { property-string }



Operatsioonid

Klass omab atribuute ja operatsioone. Atribuudid iseloomustavad klassi objekte. Atribuutide väärtusi kasutatakse objekti seisundi kirjeldamiseks. Operatsioone kasutatakse atribuutide manipuleerimiseks või muude toimingute läbiviimiseks (näiteks päringud).
Operatsioonid sarnanevad funktsioonidele, kuid nad paiknevad klassi sees ja rakenduvad ainult selle klassi objektidele.

Operatsioon kirjeldatakse tulemustüübi (return-type), nime ja 0 või enama parameetriga.
Tulemustüüp, nimi ja parameetrid üheskoos moodustavad operatsiooni *signatuuri.* Signatuur kirjeldab kõike, mis on vajalik operatsiooni kasutamiseks.
Klassi operatsioonid näitavad, mida see klass “oskab” teha, milliseid teenuseid ta osutab, seega võib operatsioone vaadata klassi liidesena. Sarnaselt atribuutidele võib operatsiooni jaoks kirjeldada nähtavuse ja skoobi.

Ka klass võib omada klassi skoobiga operatsioone. Viimased saavad pöörduda ainult klassi skoobiga atribuutide poole. Klassi skoobiga operatsioonid defineeritakse üldiste operatsioonide läbiviimiseks, nagu objektide loomine ning objektide leidmine, kus üksikud objektid ei osale (v.a. operatsiooni võimalik tulemus).

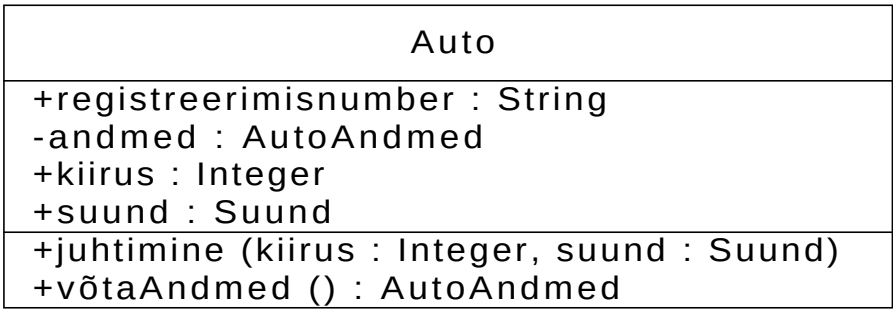
Operatsiooni formaalne süntaks:

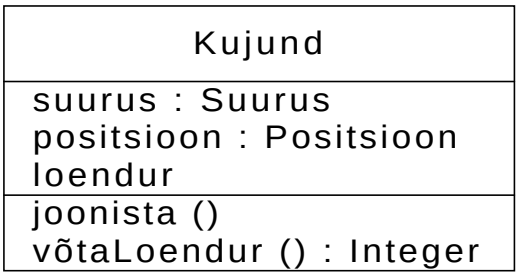
visibility name (parameter-list) : return-type-expression
{ property-string}

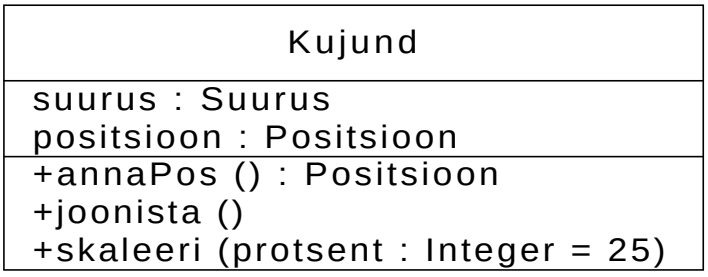
Parameetrite loetelu on komaga eraldatud loetelu formaalsetest parameetritest, millest igaüks järgib süntaksi:

name : type-expression = default-value

Operatsiooni nähtavused on samasugused nagu atribuutidel. Kõik operatsioonid ei pea omama tulemustüüpi, parameetreid või omadusstringi, kuid operatsioonil peab olema unikaalne signatuur (=return type, name, parameters)







Saab kirjeldada vaikimisi väärtusi parameetritele, s.t. kui väljakutsuja ei väärtusta parameetrit, kasutatakse vaikimisi väärtust.

Operatsioon on osa klassi liidesest.
Operatsiooni realisatsiooni nimetatakse meetodiks.
Operatsioon kirjeldatakse signatuuriga või
eeltingimuse, järeltingimuse, algoritmi ning mõjuga objektile.
Eeltingimus peab olema tõene enne, kui operatsiooni saab täita.
Näiteks kujund peab olema joonistatud enne, kui teda saab skaleerida.
Järeltingimuseks võib olla tingimus, et kohe peale joonistamist peab kujundit muutma ja seda ei tohi teha hiljem.
Kui operatsioon muudab objekti seisundit (n. skaleerimine muudab kujundi seisundit), tuleks see dokumenteerida.
Selline dokumentatsioon on operatsiooni omadus, mida tavaliselt ei esitata vahetult klassidiagrammil.

Seosed

Klassidiagramm sisaldab klasse ja nendevahelisi seoseid. Liigi järgi jagunevad seosed:

- Assotsiatsioonid
- Üldistused (generalisation)
- Sõltuvused (dependency)
- Peenendused (refinement)

Assotsiatsioon on ühendus klasside vahel ning ühtlasi nende klasside objektide vahel.

Üldistus on seos üldisema ja spetsiifilisema elemendi vahel., kus spetsiifilisem element võib sisaldada üksnes lisainformatsiooni. Spetsiifilisema elemendi eksemplari võib kasutada kõikjal, kus üldisem element on lubatud.

Sõltuvus on seos ühe sõltumatu ja ühe sõltuva elemendi vahel. Sõltumatu elemendi muudatus mõjutab sõltuvat elementi, mitte vastupidi.

Peenendus on seos kahe sama asja kirjelduse vahel, mis omavad erinevat abstraktsioonitaset.

Assotsiatsioonid

Assotsiatsioon on klassidevaheline ühendus, semantiline ühendus seoses osalevate klasside objektide (eksemplaride) vahel. Assotsiatsioon on tavaliselt kahesuunaline, s.t. kui objekt on seotud teise objektiga, siis mõlemad objektid “tunnevad teineteist”, “on ühendatud” jne.

Tavaline assotsiatsioon

Tavaline assotsiatsioon on lihtsalt ühendus klasside vahel. Joonistatakse pidevjoonena kahe klassi vahel. Assotsiatsiooni nimi (tavaliselt tegusõna, võib olla ka nimisõna) kirjutatakse seosjoone lähedusse. Seosenimed peavad tulenema probleemvaldkonnast, nagu klassinimedki.

Võimalus kasutada navigeeritavaid seoseid, lisades seose lõpuotsa noole. Nool näitab, et seost saab kasutada ainult noole suunas.
Assotsiatsioon võib omada kahte nime, kummaski suunas ühte. Nime suunda näidatakse väikese täidetud kolmnurgaga nime ees või taga, sõltuvalt suunast. Seost on võimalik lugeda lausena, näiteks:
“Autor kasutab arvutit”.

Assotsiatsiooni osaks on ka seoses osalevate objektide (eksemplaride) arv (multiplicity): Autol on 1 või enam omanikku ja isik võib omada 0 või enamat autot.
Arvukuse (multiplicity) variandid:
Nullist üheni (0..1)
Null või enam (0..*) ehk (*)
Üks või enam (1..*)
Kaks (2)
(5..11)
(1, 4, 6, 8..12)
Vaikimisi arvukus on (1)

Mudel peab olema asjaosalistele arusaadav, tõlgitav loomulikku keelde:

- Kindlustusfirma omab kindlustuslepinguid, mis viitavad ühele või enamale kliendile.
- Klient omab kindlustuslepinguid (0 või enam), mis viitavad ühele kindlustusfirmale
- Kindlustusleping sõlmitakse kindlustusfirma ning ühe või enama kliendi vahel. Kindlustusleping viitab nii kliendile (klientidele) kui ka kindlustusfirmale.

- Kindlustusleping väljendatakse (nullis või ühes) kindlustuspoliisis (kirjalik kindlustusleping)
- Kindlustuspoliis viitab kindlustuslepingule

Kui helistate kindlustusfirmasse ja kindlustate oma auto, tekib kindlustusleping, kuid mitte kindlustuspoliis. Kindlustuspoliis tehakse hiljem ja saadetakse kliendile. Niisugust reaalsust on oluline ka modelleerida. Kui modelleeriksime kindlustusäri poliisikeskselt (mitte lepingukeskselt), võib tekkida suuri probleeme. Näiteks kui klient kindlustab oma auto ja veidi hiljem teeb avarii, kui kindlustuspoliisi veel pole, kuid suuline kokkulepe kindlustusagendiga saavutatud.
Mis saab siis, kui juurutatakse uut tüüpi kindlustuspoliisid (kindlustamine Web-is), või üldse äri (ärireeglite) muutumise korral? Modelleerides tegelikku elu, lisab Web kindlustus lihtsalt vastava klassi (Web kindlustuspoliis), millel võib olla teistsugune käitumine, kui tavalisel poliisil (n. kliendid saavad neid ise muuta ja muudatused kanduvad automaatselt kindlustuslepingusse; kindlustuspoliisi saab saata e-maili teel otse kliendile).

Objektidiagramm

Objektidiagramm näitab konkreetseid objekte (eksemplare) ja nendevahelisi ühendusi kindlal ajahetkel. Objektidiagrammi võib vaadata konkreetse näitena klassidiagrammi kohta, mis näitab, kuidas keerukas klassidiagramm võib väljenduda konkreetsete objektide (eksemplaride) tasemel. Kuidas klassidiagrammi objekte saab omavahel kombineerida konkreetsel ajahetkel.

Objekt näidatakse klassina, mille nimi on allajoonitud, kusjuures objekti nime võib näidata või mitte näidata klassinime ees:

objektinimi : klassinimi

Kui objektinime ei näidata, siis allajoonitud klassinimele eelneb koolon:

: klassinimi

mis tähendab, et tegeldakse antud klassist nimetu objektiga.

Kolmas võimalus: näidatakse ainult objektinime (allajoonitud), ilma klassinimeta:

objektinimi

Rekursiivne assotsiatsioon

Seos (assotsiatsioon) võib ühendada klassi iseendaga, s.t. ühendatud objektid kuuluvad samasse klassi. Klassi seost iseendaga nimetakse *rekursiivseks assotsiatsiooniks*, mida kasutatakse keerukates mudelites, nagu tootestruktuurid.

Rollid assotsiatsioonis

Assotsiatsioon võib omada rolle seoses iga assotsiatsioonis osaleva klassiga. Rollinimi on string, mis paikneb seose otsas selle klassi lähedal, millele ta rakendub. Rollinimi näitab, missugust rolli mängib antud klass antud assotsiatsiooni kontekstis. Rollinimed on assotsiatsiooni, mitte klassi osad. Rollinimede kasutamine on mittekohustuslik.

Kvalifitseeritud assotsiatsioon

Kvalifitseeritud assotsiatsioone kasutatakse seoses üks-mitu või mitu-mitu assotsiatsioonidega. Kvalifikaator eristab objekte assotsiatsiooni mitu-poolel. Kvalifikaator näitab, kuidas identifitseeritakse konkreetset objekti assotsiatsiooni mitu-poolel ja teda võib vaadata kui võtit või indeksit kõigi objektide eraldamiseks seoses. Kvalifikaator joonistatakse väikese kastina seose otsas selle klassi lähedal, millest liikumist alustatakse. Kvalifitseeritud assotsiatsioonid vähendavad tegelikku (eksemplaride) arvukust üks-mitmelt üks-ühele. Disainis kasutatakse menüüde modelleerimisel/realiseerimisel.

Or-assotsiatsioonid

Isik (kindlustusvõtja) võib omada kindlustuslepingut kindlustusfirmaga, samuti firma (kindlustusvõtja) võib omada kindlustuslepingut kindlustusfirmaga, kuid isik ja firma ei tohi omada ühte ja sedasama kindlustuslepingut.

Or-assotsiatsioon on piirang (constraint) üle kahe või enama assotsiatsiooni. Ta määrab, et antud klassi objektid võivad osaleda mitte rohkem kui ühes piiratud seostest antud ajahetkel. Or-assotsiatsiooni tähistatakse katkendjoonega or-assotsiatsioonis osalevate assotsiatsioonide vahel ning {or} piiranguga katkendjoonel.

Korrastatud (ordered) assotsiatsioon

Ühendused objektide vahel võivad omada kindlat korda: näiteks ekraaniaknad võivad paikneda ekraanil korrastatult (üks üleval, teine all jne.). Tavaline väärtus antud omaduse jaoks assotsiatsioonil on *korrastamata*, seepärast seda spetsiaalselt ei näidata. Kui eksisteerib kindel kord ühenduste (links) vahel, siis seda näidatakse piiranguga {ordered} assotsiatsioonijoone kõrval selle klassi lähedal, mille objektid on korrastatud. Kuidas korrastus toimub (järjestatud), määratakse kas assotsiatsiooni omadusega või loogeliste sulgude sees (näiteks, {järjestatud ajas kasvavalt}).

Assotsiatsiooniklass

Assotsiatsiooniga võib siduda klassi, mida nimetatakse assotsiatsiooniklassiks. Assotsiatsiooniklass pole ühendatud assotsiatsiooni ühegi otsaga, vaid tegeliku assotsiatsiooni enesega (=assotsiatsiooni modelleerimine klassina). Assotsiatsiooniklass on (nagu) tavaline klass: ta võib omada atribuute, operatsioone ja teisi assotsiatsioone. Assotsiatsiooniklasse kasutatakse lisainformatsiooni lisamiseks objektide ühendustele (links), näiteks konkreetse ühenduse loomise aeg. Assotsiatsiooni iga ühendus (eksemplar) on seotud assotsiatsiooniklassi konkreetse objektiga (eksemplariga).

Kolmandat järku assotsiatsioon

Seostada on võimalik enam kui kahte klassi; kolmandat järku (ternary) assotsiatsioon seob kolme klassi.
Klient (poliisivaldaja rollis) omab palju (0 või enamat) kindlustuslepingut, iga kindlustusleping on seotud kindlustusfirmaga (kindlustaja rollis). Kliendi ja kindlustuslepingu vahelisel seosel on (0 või üks) kindlustuspoliis. Kolmandat järku seos esitatakse suure rombiga. Rollid ja arvukus (multiplicity) võivad olla näidatud, kuid kvalifikaatoreid ja agregatsioone (järgmine teema) pole lubatud. Kolmandat järku seosega võib olla ühendatud assotsiatsiooniklass, mis ühendatakse katkendjoone abil ühega rombi neljast tipust.

Agregatsioon

Agregatsioon on assotsiatsiooni erijuhtum. Agregatsioon näitab “osa-terviku” seost klasside vahel. Näiteks auto koosneb neljast rattast, mootorist, kerest, käigukastist, jne.
Agregatsioon kirjeldab sageli erinevaid abstraktsioonitasemeid (auto koosneb roolist, mootorist,..). Agregatsiooni väljendavad võtmesõnad “koosneb”, “sisaldab”, “on osaks”, mis väljendavad osa-terviku seost klasside ja neisse kuuluvate objektide vahel.

Agregaati näidatakse väikese rombiga assotsiatsioonijoone ühes otsas, tervikut väljendava klassi poolel. Kuna agregatsioon on agregatsiooni erijuhtum, saab temaga siduda arvukust (multiplicity), rolle (osa poolel) ja kvalifikaatoreid. Agregaat võib omada nime (koos suunaga) ning olla navigeeritav.

Jagatud (shared) agregatsioon

Jagatud agregatsioon on niisugune, milles osad võivad olla osadeks erinevates (ükskõik millistes) tervikutes. Kui agregatsioon on jagatud, siis on arvukus terviku poolel suurem kui üks. Jagatud agregatsioon on tavalise agregatsiooni erijuhtum.

Kompositsiooni agregatsioon

Kompositsiooni agregatsioonis tervik omab osasid. Osad “elavad” terviku sees; nad hävivad koos oma tervikuga. Arvukus terviku poolel peab olema 0 või 1 (0..1), osa poolel suvaline. Kompositsiooni agregatsioon moodustab puustruktuuri, jagatud agregatsioon võrkstruktuuri.
Kompositsiooni agregatsiooni esitamise kolm võimalust:
1) täidetud rombiga terviku poolel
2) kui osasid rohkem kui üks, võib tervikupoolsed otsad ühendada ühte rombi (puustruktuur). Seda saab teha ka tavalise agregatsiooni puhul
3) panna osaklassid tervikklassi sisse

Rollid saab viia atribuutideks ning klassid atribuuditüüpideks.
Kompositsiooniagregatsiooni realiseerimisel peab tervikklass juhtima oma osade elutsüklit, hävitama oma osad koos iseenda hävitamisega..
Alternatiivne viis kompositsiooniagregatsiooni realiseerimiseks on realiseerida osad klassi liikmesobjektidena, s.t. füüsiliselt kapseldada osad tervikklassi sisse.

Üldistusseos (generalisation)

Üldistusseos (ka pärimine) on seos üldisema ja spetsiifilisema elemendi vahel. Spetsiifilisem element on täielikult kooskõlas üldisema elemendiga ning sisaldab lisainformatsiooni. Spetsiifilisema elemendi eksemplari saab kasutada kõikjal, kus üldisem element on lubatud.
Üldistamist kasutatakse klasside, use case – ide jm. mudelielementide (näiteks pakettide) jaoks. Üldistamist kasutatakse tüüpide, mitte eksemplaride tasemel.
Üldistusseost võib lugeda spetsiifilisema elemendi poolt üldisema poole "on" või "on liiki" (auto on sõiduk, müügijuht on töötaja)

Tavaline üldistusseos

Spetsiifilist klassi nimetatakse alamklassiks (subclass) ja üldisemat ülemklassiks (superclass). Alamklass pärib ülemklassi kõik omadused: atribuudid, operatsioonid, assotsiatsioonid. Avaliku nähtavusega atribuudid ja operatsioonid ülemklassis jäävad avalikuks alamklassis. Privaatse nähtavusega atribuudid ja operatsioonid päritakse, kuid pole kasutatavad (ei saa pöörduda) alamklassi seest. Kaitstud (protected) atribuute ja operatsioone (tähistatakse märgiga #) ei saa kasutada teised klassid, välja arvatud klass ja tema kõik alamklassid.

Üldistusseosed moodustavad klassihierarhia, milles klass võib olla korraga ülem- ja alamklassiks.
Üldistust esitatakse pidevjoonega spetsiifilisema klassi poolt üldisema klassi poole koos suure täitmata kolmnurgaga, mis osutab ülemklassi poole.
Nagu agregatsioonis, saab üldistusseoses pärimist esitada puustruktuurina, kus kolmnurka jagavad kõik alamklassid.

Abstraktne klass on klass, mis ei tohi omada ühtegi objekti. Ta esitab teiste klasside (alamklasside) jaoks ühiseid atribuute ja käitumist, mille need pärivad.
Näiteks sõiduk on abstraktse klassi näide, mis esitab maa- ja veesõidukite ühised omadused, kuid ei sisalda ühtegi objekti (eksemplari).
Klassi saab kuulutada abstraktseks, lisades nime alla loogelistes sulgudes väärtuse { abstract }.

Abstraktne klass omab tavaliselt abstraktseid operatsioone. Abstraktne operatsioon on niisugune, millel pole realiseerivat meetodit samas klassis, kus ta spetsifitseeritakse, vaid ainult signatuur. Klass, mis omab vähemalt ühte abstraktset operatsiooni, peab olema abstraktne klass.

Klass, mis pärib klassilt, kus on üks või enam abstraktset operatsiooni, peab realiseerima need operatsioonid (andma nende jaoks meetodid) või olema ise abstraktne klass. Abstraktsed operatsioonid näidatakse omadusstringiga { abstract } operatsiooni signatuuri järel.

Näiteks abstraktne klass Sõiduk omab abstraktseid operatsioone *drive , start , stop* . Need on käitumised, mida iga sõiduk peab omama. Sõiduki iga alamklass peab andma meetodid nende operatsioonide jaoks või hakkama ise abstraktseks klassiks.

Konkreetne klass on võimeline looma objekte (eksemplare) ja omab realisatsioone (meetodeid) kõigi operatsioonide jaoks.
Kui Sõiduki klass on spetsifitseerinud abstraktse operatsiooni *drive* , siis nii Auto kui ka Laev peavad realiseerima vastava meetodi (või operatsioonid tuleb kuulutada abstraktseteks), mis saavad olema erinevad. Ühel juhul paneb operatsioon pöörlema rattad, teisel juhul propelleri (laeva kruvi). Alamklassid pärivad ülemklassilt operatsiooni, kuid realiseerivad selle erinevalt.

Alamklassid võivad operatsioone üle defineerida. Üledefineeritud operatsioon peab omama sama signatuuri (tulemustüüp, nimi, parameetrid) nagu ülemklassis. Üledefineeritud operatsioon võib olla nii abstraktne (puudub realisatsioon ülemklassis) kui konkreetne (omab realisatsiooni ülemklassis). Mõlemal juhul kasutatakse üledefineerimist antud klassi kõigi eksemplaride jaoks. Alamklassis võib lisada uusi operatsioone, atribuute, assotsiatsioone.

Näide: Isik juhib Sõidukit. Sõiduk on abstraktne klass s.t. konkreetsed objektid, mida Isik juhib, on konkreetsetest alamklassidest Auto ja Laev. Kui Isik käivitab juhtimisoperatsiooni, sõltub tulemus sellest, kas objektiks juhtub olema Auto või Laev. Kui Auto, pannakse pöörlema rattad (kasutades realisatsiooni, mis kirjeldatud klassis Auto), kui Laev, siis propeller (kasutades realisatsiooni, mis kirjeldatud klassis Laev).

Tehnikat, kus alamklassi objekt toimib nagu ülemklassi objekt, kusjuures defineeritakse üle üks või enam ülemklassi operatsioonidest, nimetatakse *polümorfismiks.* Polümorfism tähendab, et operatsiooni tegelik realisatsioon sõltub objekti tüübist, millele operatsiooni rakendatakse.

Diskriminaatoriga on võimalik näidata, mille alusel üldistatakse/spetsialiseeritakse. Näiteks sõidukite puhul liikumiskeskkond.

Piiratud (Constrained) üldistusseos

Piirangud üldistusseosel määravad, kuidas üldistusseost kasutatakse ning laiendatakse. Enam kui ühe alamklassiga üldistusseoste jaoks saab defineerida järgmisi kitsendusi:

Lõikumine ehk ülekate (Overlapping)
Mittelõikuvus (Disjoint)
Täielikkus (Complete)
Mittetäielikkus (Incomplete).

Taolisi semantilisi piiranguid näidatakse loogelistes sulgudes üldistuse kolmnurga lähedal. Kui üldistusseostel pole ühist kolmnurka,vaid iga alamklassi kohta eraldi kolmnurk, tuleb kõik kokkukuuluvad pärimisseose jooned ühendada ristuva katkendjoonega, mille läheduses esitada piirang(ud) loogeliste sulgude sees.

Lõikuvad ja mittelõikuvad üldistusseosed

Lõikuv (mitmene) pärimine tähendab, et alamklassid võivad pärida enam kui ühelt ülemklassilt ning mitu ülemklassi võivad omada ühist alamklassi pärimispuus (võrkstruktuur).

Mittelõikuv üldistusseos tähendab, et alamklass saab pärida vahetult ainult ühelt ülemklassilt ning mitmel ülemklassil ei saa tekkida ühist alamklassi. Vaikimisi on tegemist mittelõikuva pärimisega; kui vastupidist pole näidatud, ei ole ühised alamklassid lubatud.

Täielikud ja mittetäielikud üldistused

Täielik üldistusseos tähendab, et kõik alamklassid on kirjeldatud ning uusi lisada ei saa. Mittetäielik üldistusseos (vaikimisi ongi üldistus mittetäielik) tähendab, et alamklasse saab juurde lisada. Kuna üldistamise üheks põhieesmärgiks on võimaldada edasisi laiendamisi, peetakse mittetäielikke üldistusi tavalisemateks.

Sõltuvuse ja peenendamise seosed

Sõltuvusseos on semantiline ühendus kahe mudelielemendi vahel, millest üks on sõltumatu ja teine sõltuv element. Sõltumatu elemendi muutmine mõjutab sõltuvat elementi. Nagu üldistusseose puhul, võib mudelielemendiks olla klass, pakett, use case, jne..
Sõltuvuste näited: Üks klass käsitleb teise klassi objekti parameetrina; üks klass pöördub teise klassi globaalse objekti poole; üks klass kutsub välja klassiskoobiga operatsiooni teisest klassist.

Sõltuvusseost näidatakse katkendjoonega mudelielementide vahel. Joone ühes otsas on nool, joonel võib olla märgend, mis näitab stereotüüpi (sõltuvuse liiki). Sõltuvuse stereotüübiks võib olla näiteks “sõber”: üks mudelielement saab erilised õigused teise mudelielemendi sisemiste (isegi privaatse nähtavusega) osade / struktuuride poole pöördumiseks.

Peenendus (refinement) on seos sama asja kahe kirjelduse vahel, mis on erineva abstraktsioonitasemega. Peenendusseos võib olla tüübi ja teda realiseeriva klassi vahel, siis nimetatakse seda realisatsiooniks (realization). Samuti on peenendusteks seosed analüüsi klassi ja disaini klassi vahel, mis modelleerivad sama asja, või seosed kõrgtaseme kirjelduse ja madala taseme kirjelduse vahel (näiteks seos kollaboratsiooni üldvaate ning sama kollaboratsiooni detailse diagrammi vahel). Peenendusseost saab kasutada ka sama asja erinevate realisatsioonide (näiteks lihtsa realisatsiooni ehk prototüübi ning keerukama kuid effektiivsema realisatsiooni) modelleerimiseks.

Peenendusseost näidatakse katkendjoonega ja kolmnurgaga (üldistuse sümbol) kahe mudelielemendi vahel. Peenendust kasutatakse mudeli(te) kooskõlastamiseks. Suurtes projektides peavad kõik koostatavad mudelid olema kooskõlastatud. Mudeleid kooskõlastatakse selleks,et:

- Näidata, kuidas erineva abstraktsioonitasemega mudelid on omavahel seotud.
- Näidata, kuidas erinevate arendusfaaside (vajaduste spetsifitseerimine, analüüs, disain, realiseerimine,..) mudelid on seotud.
- Toetada konfiguratsiooni juhtimist
- Toetada jälgitavust (traceability) mudeli(te)s.

Piirangud ja tuletused (Reeglid)

UML-is saab väljendada reegleid (Rules): piiranguid (constraints) ning tuletusi (derivations). Piirang kitsendab mudelit. Piirangute juba tuttavateks näideteks on or-assotsiatsioon, korrastatud (ordered) assotsiatsioon ning pärimise piirangud (overlapping, disjoint, complete, incomplete). Tuletused on reeglid selle kohta, kuidas asju saab tuletada, näiteks isiku vanust (jooksev kuupäev miinus sünnikuupäev). Reegleid saab siduda kõigi mudelielementidega, kõige sagedamini kasutatakse neid atribuutide, assotsiatsioonide, pärimise ja rollide jaoks ning ajapiiranguid dünaamikamudelites (oleku-, jada, kollaboratsiooni ja tegevusdiagrammid). Kõiki piiranguid näidatakse loogelistes sulgudes ({}) mudelielemendi läheduses või sulgudes kommentaari koosseisus, mis on ühendatud mudelielemendiga.

Assotsiatsioonid võivad olla tuletatud või piiratud. Kui firmal on lepinguid paljude klientidega, võib tuletatud seosega esitada tähtsamad ehk VIP kliendid. Tuletatud assotsiatsioonil on märgend seosjoone lähedal, mis algab kaldkriipsuga, millele järgneb tuletuse nimi. Piiratud assotsiatsiooniga on tegemist, kui üks assotsiatsioon on teise assotsiatsiooni alamhulk.

Ka atribuudid võivad olla tuletatud või piiratud. Tüüpiline atribuudi piirang on tema väärtuspiirkond (=atribuudi omadusstring). Näiteks atribuudi *värv* omadusstring on {punane, roheline, kollane}, või hoopis {0<= värv <=255}. Tuletatud atribuut arvutatakse mingil viisil teistest atribuutidest. Arvutusvalem antakse sulgudes klassi all. Tuletatud atribuudi nimi algab kaldkriipsuga, tuletatud atribuudi väärtust ei salvestata selle klassi objektidesse, vaid arvutatakse iga kord.

Üldistusseos omab ainult piiranguid, mitte tuletusi.
Rollid võivad omada piiranguid, mis kitsendavad ühe objekti poolt täidetavate rollide kombinatsioone. (Näiteks isik tavaliselt ei kinnita iseenda poolt registreeritud tegevusi).

Reeglite (piirangute ja tuletuste) kirjeldamiseks kasutatakse UML koostisosa, mida nimetatakse *navigation expression.* Näited:

Insurance_Contract.Policyholder > 0
Person.~Policyholder. Sum insured > 1000
Car.Driver.driving licence = True

Liidesed

Pakettide, komponentide ja klasside jaoks saab defineerida liideseid. Sel juhul nimetatud elemendid toetavad / realiseerivad liideses defineeritud käitumist. Liidest võib vaadelda lepinguna koostoimivate mudelielementide klastrite (kobarate) vahel. Programmeerimises on ekvivalentideks OLE/COM või Java liidesed (interface), kus liideseid saab kirjeldada klassidest eraldi ning liidese realiseerimiseks saab valida palju klasse (või pakette või komponente).. Liides kirjeldatakse abstraktsete operatsioonidena: signatuuride hulk, mis üheskoos spetsifitseerib käitumise, mida konkreetne element (või elemendid) võivadrealiseerida/toetada. Objektid võivad nüüd sõltuda liidesest üksinda, teadmata midagi liidest realiseerivatest klassidest.

Liides esitatakse väikese ringiga, millel on nimi ning ühendus (sisuliselt 1-1 assotsiatsioon) oma mudelielemendiga. Klass, mis kasutab konkreetse klassi poolt realiseeritavat liidest, ühendatakse läbi sõltuvusseose (katkendnool) liidese ringiga. Sõltuv klass on sõltuv ainult operatsioonidest konkreetses liideses, kuid mitte millestki muust selles klassis (vastasel juhul peaks sõltuvusnool ulatuma klassi sümboloni). Liidesel paiknevaid operatsioone ei näidata otseselt diagrammil. Liidese operatsioonide näitamiseks spetsifitseeritakse liides klassina koos stereotüübiga “interface”, kasutades tavalist klassisümbolit.

Pärimisseost liideste vahel saab näidata klassidiagrammil, kus kõik liidesed omavad stereotüüpi “interface”.

Paketid

Pakett on mudelielementide grupeerimise mehhanism. Kõiki mudelielemente, mida pakett omab või millele ta viitab, nimetatakse paketi sisuks. Paketi eksemplar ei oma mõtet. Pakette kasutatakse modelleerimistöös ega transleerita täidetavateks süsteemideks. Mudelielemendi omanikuks ei saa olla rohkem kui üks pakett. Paketti nimetatakse sageli allsüsteemiks.

Paketid võivad importida mudelielemente teistest pakettidest. Kui element on imporditud paketi poolt, siis see pakett viitab talle justkui omanikpakett. Pakettide vahel on lubatud sõltuvusseosed, peenendusseosed ning üldistusseosed.

Pakett esitatakse suure ristkülikuna koos väikese ristkülikuga vasakus ülanurgas. Kui paketi sisu (klasse) ei näidata, siis paketi nimi antakse suures ristkülikus, vastasel korral väikeses.

Pakett sarnaneb agregatsioonile. Kui pakett omab oma sisu, on tegemist kompositsiooniagregatsiooniga (composed aggregation); kui ta viitab oma sisule (s.t. impordib elemente teistest pakettidest), on tegemist jagatud agregatsiooniga.

Pakett võib omada nähtavust (visibility) nagu klassidki, mis näitab, kuidas teised paketid saavad pöörduda tema sisu poole. Nähtavuse astmeid on 4: privaatne, kaitstud, avalik, realisatsioon. Vaikimisi avalik. Avalik nähtavus tähendab, et teised elemendid saavad näha ja kasutada paketi sisu. Privaatne (private) tähendab, et ainult pakett, mis omab või impordib mudelielementi, saab seda kasutada. Kaitstud (protected) nähtavus on sarnane privaatsele, kuid lisaks omanikule ja importijatele saavad kaitstud paketi sisu kasutada ka pärimishierarhias allpool paiknevad paketid. Realisatsioon on sarnane privaatsele, kuid teised paketid ei saa realisatsiooni nähtavusega paketi sisu importida.

Pakett võib omada liidest, mis publitseerib tema käitumist. Tavaliselt siis üks või enam klasse paketi seest realiseerivad selle liidese.

Templates

Templiteid näidatakse parametriseeritud klassidena. Templiti parameetreid kasutatakse reaalse klassi loomisel, mida saab kasutada. Templit on klass, mis pole veel täielikult spetsifitseeritud ja mille lõplik spetsifikatsioon tehakse läbi parameetrite. Parameetriteks võivad olla klassid või primitiivsed tüübid. Parametriseeritud klassi kasutatakse klassigrupi kirjeldamiseks. Näiteks massiv, mille eksemplarklassideks autode, värvide jne. massiiv.
Parametriseeritud klassi parameetrite loetelu sisaldab iga parameetri jaoks nime ja tüüpi. Kui parameetriks klass, pole tüübi näitamine kohustuslik: [T:class, A:integer] on sama mis [T, A:integer]. Parameetrite loetelu näidatakse katkendjoonega ristkülikus parametriseeritud klassi ristküliku paremas ülanurgas. Templiti eksemplar näidatakse tavalise klassisümboliga, kus klassinimi sisaldab nii templiti nime kui ka parameetrite väärtusi: Array<Car,100>. On võimalik näidata templiti ja tema eksemplari vahelist seost peenendusseosena, kasutades stereotüüpi “bind”, millele järgnevad tegelikud parameetrid klassile.

Mudeli kvaliteet

Modelleerimiskeel üksinda ei saa tagada mudeli head kvaliteeti. Mudelite kvaliteedile tuleb pöörata eraldi tähelepanu, sõltumata kasutatavast modelleerimiskeelest.

Hea mudel on:

- Arusaadav
- Vastab eesmärgile
- Sisaldab olulist
- Normaalsed (mõistetavad nimed)
- Kooskõlas seotud mudelitega
- Mitte liiga keeruline