Andmebaaside projekteerimiseks kasutatavad mudelid

Ver 2.14 (04. märts 2012)

Sisukord

UML keelega tutvumiseks soovitame raamatut:

Fowler, M., 2007. *UMLi kontsentraat. 3. redaktsioon. Objektmodelleerimise standardkeele UML2.0 lühijuhend*. Cybernetica AS. [WWW] http://www.cyber.ee/publikatsioonid/40-raamatud/45-raamat-umli-kontsentraat (23.01.2012)

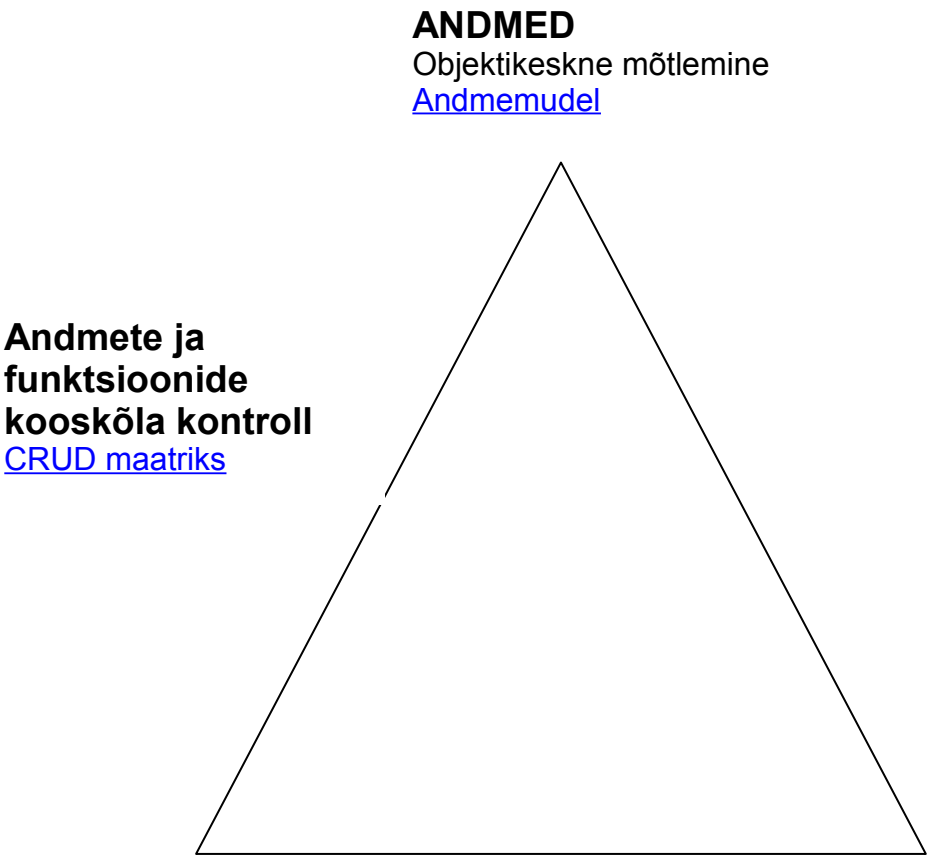
UMList õieti rääkimiseks ja kirjutamiseks ning räägitust ja kirjutatust aru saamiseks vaadake UMLi sõnastikke:

- eesti-inglise:
  http://www.cyber.ee/publikatsioonid/40-raamatud/uml-2-ja-rup-sonastik-eesti-inglise
- inglise-eesti:
  http://www.cyber.ee/publikatsioonid/40-raamatud/46-raamat-infoturbe-uml2-ja-rup-terminite-tolked

1. Sissejuhatus

Järgnevalt antakse ülevaade mõnedest andmebaasi projekteerimise juures kasutatavatest mudeli tüüpidest. Iga mudeli tutvustuse lõpus on küsimused, millele vastamine aitab hinnata mudelite kvaliteeti. **Dokumendi lõpus esitatakse soovitused selle kohta, kuidas parandada mudelite loetavust.**

Mikli (1999) esitab süsteemi modelleerimist iseloomustava diagrammi. Modelleerimise kasutatakse aja-, protsessi ja objektikeskset mõtlemist.



Joonis 1Modelleerimise "kolmnurk".

2. Kasutusjuhtude mudel

Kasutusjuhtude mudeleid kasutatakse, et näidata, mida uus süsteem peaks tegema või mida olemasolev süsteem juba teeb.

Kasutusjuhtude modelleerimises vaadatakse süsteemi "musta kastina", mis pakub tegutsejatele **teenuseid** (kasutusjuhte). Kuidas süsteem seda teeb, kuidas kasutusjuhud on realiseeritud ja kuidas nad sisemiselt töötavad pole esialgu oluline. Seda ei pruugi teada veel isegi projekteerijad.

Kasutusjuhtude mudeli koostamise sammud.

- Süsteemi defineerimine
- Tegutsejate ja kasutusjuhtude leidmine
- Kasutusjuhtude kirjeldamine
- Seoste defineerimine kasutusjuhtude vahel
- Mudeli õigsuse kontroll

Kasutusjuhtude mudel koosneb:
- kasutusjuhtude diagrammid,
- kasutusjuhtude tekstilised kirjeldused.

Kasutusjuhtude diagrammi elemendid:
- süsteem,
- tegutsejad,
- kasutusjuhud,
- seosed eelnimetatud elementide vahel.

Kasutusjuhtude mudel võib olla jagatud e. tükeldatud paljudeks kasutusjuhtude diagrammideks. See tagab, diagrammide selguse ja ülevaatlikkuse. Tükelduse võib teha näiteks lähtudes süsteemi põhifunktsioonidest (st. igal diagrammil on kasutusjuhud, mis vastavad ühele põhifunktsioonile).

2.1 Süsteem

Kasutusjuhtude modelleerimine defineerib arendatava süsteemi piirid. Süsteemi piirid defineeritakse süsteemi poolt käsitletava funktsionaalsusega. Funktsionaalsus esitatakse hulga kasutusjuhtudega, millest igaüks kirjeldab selle tervikliku alamosa.

Piiride täpne paikapanemine (mis võtta sisse, mis jätta välja) on eduka süsteemiarenduse esimene tingimus.

2.2 Tegutsejad

Tegutseja on keegi või miski, mis suhtleb süsteemiga / kasutab süsteemi. Kasutaja saadab ja/või saab sõnumeid süsteemile/süsteemilt. Tegutsejad viivad läbi kasutusjuhte.

Tegutseja võib olla:
- inimene,
- süsteem.

Tegutseja esitab rolli (nt. õppejõud), mitte üksikkasutajat süsteemis (nt. Karin Kool, Kaspar Kala, Taavi Kask).

Ühe füüsilise isiku kohta võib olla mitu tegutsejat, sõltuvalt tema rollist süsteemis. Näiteks õppejõud võib samal ajal olla ka tudeng. Teiselt poolt võib õppejõud töötada ka õppetooli juhatajana.

Ilmavaatlusjaamas aga on tegutsejateks näiteks mõõteseadmed, mis teevad automaatselt mõõtmisi ja edastavad need infosüsteemile.

Tegutseja nimi peab väljendama tema rolli.

Kasutusjuht algatatakse tegutseja poolt, kes saadab kasutusjuhule sõnumi. Süsteemi jaoks on see sõnum sündmus, mis käivitab vajaliku teenuse.

Kui kasutusjuht on läbi viidud, peab kasutusjuht saatma sõnumi ühele või mitmele tegutsejale. Need sõnumid võivad minna ka teistele tegutsejatele lisaks sellele, kes algatas kasutusjuhu.

2.3 Tegutsejate ülesleidmine

Tegutsejate ülesleidmisega pannakse paika üksused/osapooled, mis/kes on huvitatud süsteemiga suhtlemisest ja selle kasutamisest. Siis on võimalik asetuda tegutseja positsioonile ning üritada identifitseerida tegutseja nõuded süsteemile ning milliseid kasutusjuhte ta vajab. Tegutsejate ülesleidmiseks võib vastata järgmistele küsimustele:

- Kes hakkab kasutama süsteemi põhifunktsionaalsusi?
- Kes vajab toetust süsteemilt oma igapäevaste ülesannete täitmisel?
- Keda on vaja süsteemi hooldamiseks, administreerimiseks ja töös hoidmiseks?
- Milliseid riistvaraseadmeid peab süsteem käsitlema?
- Milliste teiste süsteemidega peab süsteem suhtlema? Nii süsteemid, mis algatavad kontakti meie süsteemiga, kui ka süsteemid, millega meie süsteem ise kontakteerub.
- Kes või mis omab huvi tulemuste (väärtuse) vastu, mida süsteem toodab?

Tegutsejate leidmiseks võib küsitleda olemasoleva (arvuti-) süsteemi kasutajaid, uurida milliseid erinevaid rolle nad täidavad igapäevatöös süsteemiga.

Tegutseja eksemplaride (näiteks inimeste) nimetamise kaudu saab kontrollida, kas selline tegutseja tegelikkuses eksisteerib. Tegutseja peab omama assotsiatsiooni ühe või enama kasutusjuhuga. Kui ta ka ühtegi kasutusjuhtu ei käivita, peab ta mõnes nendest mingis punktis osalema.

2.4 Kasutusjuhud

Definitsioon 1

Kasutusjuht on jutustav dokument, mis kirjeldab sündmuste jada, mis tekib, kui tegutseja, süsteemiväline agent kasutab süsteemi mingi protsessi läbiviimiseks (Ivar Jacobson).

Definitsioon 2

Kasutusjuhud moodustavad lepingu süsteemist huvitatud osapoolte vahel, mis kirjeldab mainitud süsteemi käitumist erinevates olukordades ja on organiseeritud tegutsejate eesmärkide järgi.

Kasutusjuht on suhteliselt mahukas protsessi kirjeldus, selle algusest kuni lõpuni, mis tüüpiliselt sisaldab palju samme.

Kasutusjuhuks ei ole tavaliselt üksik samm või toiming protsessis.

Kasutusjuht kirjeldab interaktsiooni süsteemiga. Tüüpilisemad viisid süsteemi piiritlemiseks on järgmised.

- Mingi seadme või arvutisüsteemi riistvara/tarkvara.
- Osakond organisatsioonis.
- Organisatsioon.

Kasutusjuht esindab terviklikku funktsionaalsust tegutseja jaoks. Kasutusjuhu omadused:

- **Kasutusjuhte käivitatakse alati tegutseja poolt.** Kasutusjuht viiakse alati läbi tegutseja poolelt. Tegutseja peab otseselt või kaudselt käskima süsteemil kasutusjuhtu täita.

Mõnikord ei pea tegutseja otseselt kasutusjuhtu käivitama. Näiteks võib kasutusjuht käivituda, kuna saabus mingi kindel ajahetk (näiteks aasta lõpp) või sai läbi mingi määratud ajaperiood. Sellisel juhul võib ette kujutada, et kasutusjuhu algatab tegutseja "Aeg".

- **Kasutusjuht annab tegutsejale väärtuse**. kasutusjuht peab kasutajale üle andma “käegakatsutava” väärtuse. Väärtus ei pea olema väljapaistev, võib olla lihtsalt tajutav.

- **Kasutusjuht peab olema täielik.** Tavaline viga: jagatakse kasutusjuht väiksemateks kasutusjuhtudeks, mis realiseerivad üksteist nagu üksteist väljakutsuvad funktsioonid programmeerimiskeeles. Kasutusjuht pole täielik, kui see pole tootnud lõppväärtust kasutaja jaoks, isegi kui see sisaldab kasutajadialooge.

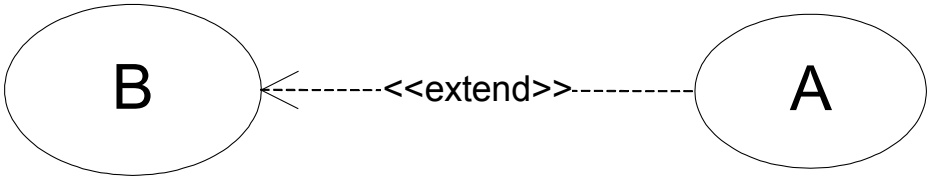
2.5 Seosed kasutusjuhtude vahel

Kasutusjuhu mahukamad sammud ja harutegevused tuleks teha eraldi *kasutusjuhuks* kui on täidetud vähemalt üks järgmistest tingimustest.
- Nad korduvad teistes *kasutusjuhtu*des.
- Nad on keerukad ja pikad, ning nende lahutamine aitab muuta *kasutusjuhud* selgemateks ja arusaadavamateks.

Kasutusjuhtude vahel eristab UML kahte liiki seoseid:
- <<include>>,
- <<extend>>.

<<extend>>
Extend – An extends relationships from use case A to use case B indicates that an instance of use case B may include (subject to specific conditions specified in the extension) the behavior specified by A. Behavior specified by several extenders of a single target use case may occur within a single use case instance. (UML Notation Guide 1.1)

Laiendamisseos kasutusjuhust A (*laiendav* kasutusjuht) kasutusjuhtu B (*laiendatav* kasutusjuht) näitab, et kasutusjuhu B eksemplar võib sisaldada (vastavalt laiendavas kasutusjuhus toodud tingimustele) A poolt spetsifitseeritud käitumist. Ühe kasutusjuhu mitme laiendaja poolt spetsifitseeritud käitumine võib leida aset ühes kasutusjuhu eksemplaris.



Joonis 2<<extend>> seos kasutusjuhtude vahel.

Joonise interpretatsioon: Kasutusjuht A *laiendab* kasutusjuhtu B.

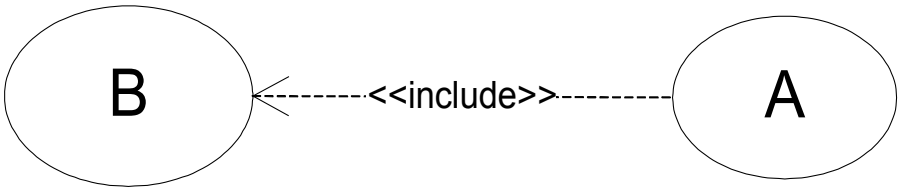
Kasutusjuhu B spetsifikatsioonis ei nimetata laiendavat kasutusjuhtu (kasutusjuht A). See on kasulik juhul, kui kasutusjuhu B puhul soovitakse kasutada kuitahes palju laiendavaid kasutusjuhte. Kui lisandub uus laiendav kasutusjuht, ei tule laiendatava kasutusjuhu (B) spetsifikatsiooni täiendada.

Kasutusjuht B on põhiline kasutusjuht. Kasutusjuhus A on määratud tingimused, mille täitumisel kasutusjuhus B katkestatakse kasutusjuhu B täitmine ja täidetakse kasutusjuht A. Seejärel jätkub kasutusjuhu B täitmine sealt kus see pooleli jäi.

<<include>> (vana nimega <<use>>)

Include – An include relationship defines that a use case contains the behavior defined in another use case (UML 2.0 spetsifikatsioon).

Kasutamisseos kasutusjuhust A kasutusjuhtu B näitab, et kasutusjuht A sisaldab käitumist, mis on spetsifitseeritud B poolt.



Joonis 3<<include>> seos kasutusjuhtude vahel.

Joonise interpretatsioon: Kasutusjuht A *kasutab* kasutusjuhtu B

Kasutusjuht A on põhiline kasutusjuht. Kasutusjuhus A on määratud tingimused, mille täitumisel täidetakse kasutusjuht B. Seejärel jätkub kasutusjuhu A täitmine sealt kus see pooleli jäi.

Programmeerimiskeeles on selle vasteks alamprogrammi väljakutsed põhiprogrammi täitmisel.

Extend seose kasutamine võimaldab näidata erinevate tingimuste täidetusest sõltuvaid alternatiive ja *Include* seose kasutamine võimaldab taaskasutada mõne kasutusjuhu kirjeldust erinevate kasutusjuhtude kirjeldamisel.

Lisaks on võimalik näidata kasutusjuhtude diagrammil pärimisseoseid tegutsejate vahel ja pärimisseoseid kasutusjuhtude vahel.

2.6 Kasutusjuhtude ülesleidmine

Järgnevalt antakse lühiülevaade erinevatest meetoditest, kuidas kasutusjuhte üles leida.

Meetod 1 (põhineb tegutsejatel)

1. Leia tegutsejad, kes on seotud süsteemi või organisatsiooniga (üldvaate tegutsejate nimekiri).
2. Leida nende tegutsejate *eesmärgid* seoses selle süsteemiga. Eesmärkidest tulenevalt identifitseeri iga tegutseja jaoks protsessid, mida ta algatab, või milles ta osaleb. "Poe tegutsejale pähe", ürita end mõelda tema olukorda. Identifitseeri võimalikud täiendavad teenused, mida tegutseja võiks vajada.

Iga üles leitud tegutseja kohta võiks küsida järgnevat.

- Millised on tegutseja eesmärgid seoses süsteemiga?
- Milliseid funktsioone tegutseja vajab süsteemilt? Mida tegutseja peab tegema?

- Kas tegutseja peab lugema, looma, hävitama, või salvestama süsteemis andmeid?
- Kas tegutseja peab saama teateid sündmustest või ise registreerima sündmusi süsteemis? Mida need sündmused esindavad funktsionaalsuse mõttes?
- Kas tegutseja igapäevatööd saab lihtsustada või muuta efektiivsemaks süsteemi uute funktsioonide kaudu (mis siiani oli automatiseerimata)?

Küsimused, mis ei puuduta ühte konkreetset tegutsejat.

- Milliseid sisendeid/väljundeid süsteem vajab? Kust sisendid tulevad ja kuhu väljundid lähevad?
- Millised on süsteemi praeguse realisatsiooni põhiprobleemid

Viimaste küsimustega identifitseeritakse esmalt kasutusjuhud ning seejärel nendega seotud tegutsejad. Kasutusjuht peab olema seotud vähemalt ühe tegutsejaga.

Tegutsejate vahel võib näidata üldistusseoseid. Seda on mõtet kasutada, kui mitu erinevat tegutsejat võivad olla seotud ühe ja sama kasutusjuhuga. Siis võib nende tegutsejate põhjal teha üldistuse ja näidata seost kasutusjuhuga vaid üldisema tegutseja tasemel.

Meetod 2 (põhineb sündmustel)

1. Leia välised sündmused, millele süsteem peab reageerima (üldvaate sündmuse nimekiri).
2. Seosta sündmused *tegutsejate* ja *kasutusjuhtudega.*

Uuri, kes sündmuseid algatas ja kuidas peaks süsteem sündmusele reageerima.

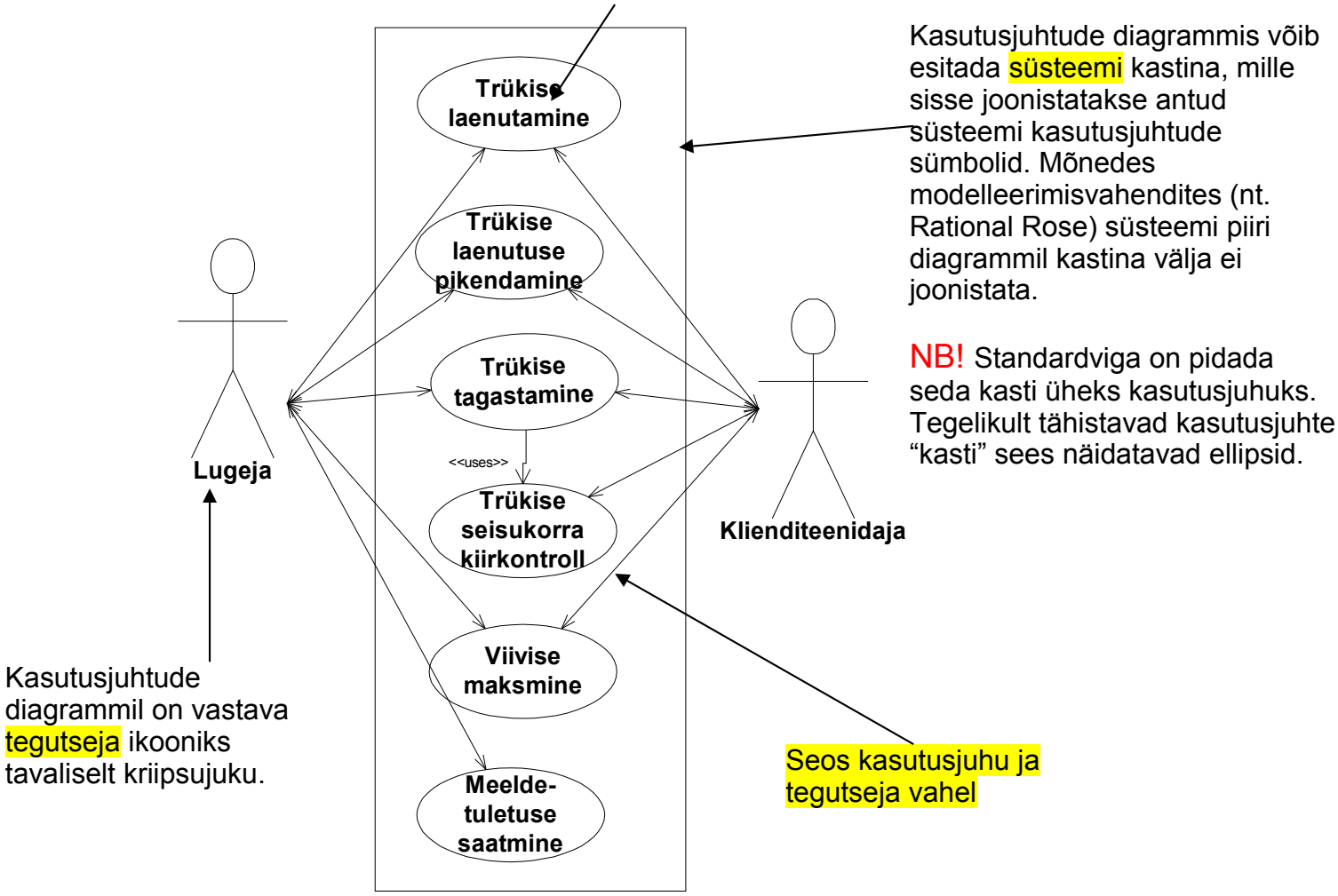
Millisel tasemel kasutusjuhte kirjeldada?

Soovitav oleks leida ja kirjeldada kasutusjuhud, millest igaüks vastab mingile elementaarsele äriprotsessile (*elementary business process, EBP*). Iga selline protsess viiakse läbi ühe tegutseja poolt ühel ajahetkel ja ühes kohas. Iga selline kasutusjuht käivitub reaktsioonina mingile sündmusele, selle läbimine annab tegutsejale väärtuse ning selle läbimise järel on süsteemis terviklikud andmed.

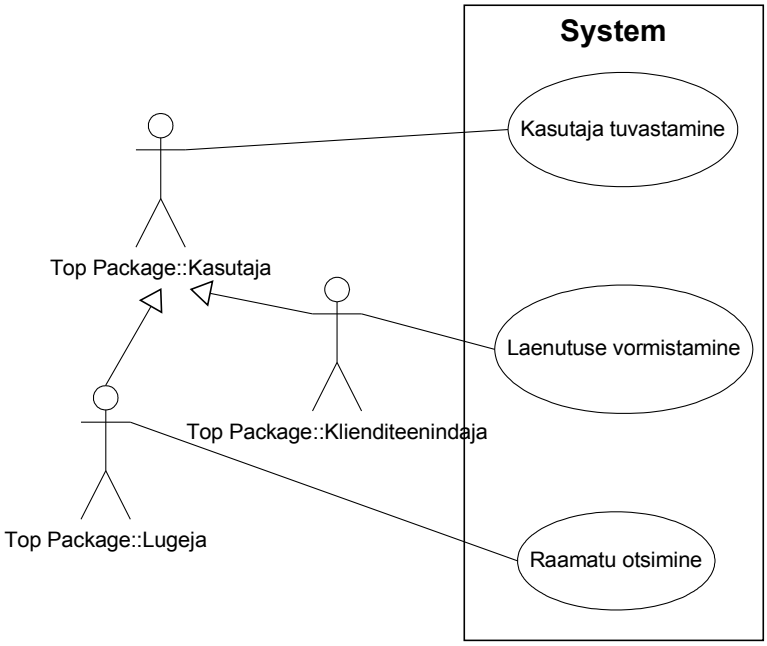
Sellest soovitusest tulenevalt ei tohi ühe kasutusjuhu läbimine võtta päevi (nt. "pea lepingu üle läbirääkimisi"). Teisalt ei tohi kasutusjuht kirjeldada ühte väikest sammu suuremas protsessis (nt. "lisa tellimusse uus kaup").

2.7 Kasutusjuhtude mudeli näide

Kasutusjuhtu tähistatakse kasutusjuhtude diagrammil ellipsiga, mis sisaldab kasutusjuhu nime. Nimi võib olla ka kasutusjuhu sümboli all. Tavaliselt paigutatakse kasutusjuht süsteemi piiride sisse ning seostatakse tegutseja(te)ga assotsiatsiooni või kommunikatsiooni assotsiatsiooni abil.



Joonis 4Kasutusjuhtude diagrammi näide.



Joonis 5Kasutusjuhtude diagrammi näide koos pärimisseosega tegutsejate vahel.

Pärimisseost tegutsejate vahel on kasulik näidata, kui mõni kasutusjuht on ühine paljudele eri tüüpi tegutsejatele, mõni aga ainult mõnele spetsiifilisele tegutsejale.

Kasutusjuhtude diagramm on piltlikult nagu raamatu sisukord. See võimaldab saada kiire ülevaate süsteemist, kuid ei kirjelda kasutusjuhte täpselt.

Lisaks diagrammile esitatakse **kõigi** kasutusjuhte kirjeldused tekstilises vormis. See on kirjeldus sellest, kuidas tegutsejad ja kasutusjuhud (süsteem) suhtlevad.

Tekstikirjeldusi on võimalik esitada erineva täpsusastmega. Erinevad autorid pakuvad välja erinevaid viise, kuidas kasutusjuhtude kirjeldusi kirja panna. Siinkohal lähtutakse Larmani (1997) poolt pakutud kirjelduse struktuurist.

Kõrgtaseme formaat:

Nimi: Kasutusjuhu nimi
Tüüp: Määrab kasutusjuhu prioriteedi. Kasutusjuhtude prioriteet määrab nende analüüsimise ja nendele vastava infosüsteemi valmistegemise järjekorra. Võimalikud väärtused:

- Primaarne kasutusjuht. Tähistab süsteemi toimimise jaoks kõige olulisemaid protsesse.
- Sekundaarne kasutusjuht. Tähistab harvemini toimuvaid protsesse.
- Valikuline kasutusjuht. Käsitleb protsesse, mida ei pruugi esialgu analüüsida.

Üheks võimalikuks veaks, mida kasutusjuhtude kirjelduste koostamisel tehakse on, et kõik kasutusjuhud määratakse ühte tüüpi (nt. primaarsed).

Tegutsejad: Kasutusjuhtu kasutavate tegutsejate nimekiri.
Kirjeldus: Kasutusjuhu lühike tekstikirjeldus.

Laiendatud formaat (üks võimalik struktuur):

Nimi: Kasutusjuhu nimi
Tegutsejad: Kasutusjuhtu kasutavate tegutsejate nimekiri.
Osapooled ja nende huvid: Iga kasutusjuht peab kirjeldama sellist, ainult sellist ja täpselt sellist käitumist, mis rahuldab osapoolte huve (aitab neil saavutada oma eesmärke). Osapoolte ja huvide kirjeldamine aitab süsteemselt ja metoodiliselt kontrollida, et iga kasutusjuht kaitseb kõigi sellest huvitatud osapoolte huvisid. Kui ei kaitse, siis tuleb kasutusjuhus teha muudatusi. Seega aitab antud kirjeldus määrata ka kasutusjuhu piire (mis jääb sisse ja mis välja). Näiteks kirjutades, et müügimehe huviks on komisjonitasu saamine, tuletab see edasise kasutusjuhu kirjutamise käigus meelde, et süsteem peab võimaldama komisjonitasu suuruse välja arvutada ning selle väärtuse registreerida. See käitumine tuleb kasutusjuhu kirjelduses kajastada.
Tüüp: primaarne, sekundaarne, valikuline
Käivitav sündmus: Kasutusjuhu käivitava sündmuse kirjeldus. Sageli kiputakse siinkohal esitama lauset kujul: "Kasutusjuht käivitub, sest tegutseja X soovib seda käivitada". Tegelikult tuleks siinkohal kirjeldada selle soovi põhjuseid ja tagamaid.
Kirjeldus: Kasutusjuhu lühike tekstikirjeldus (võib olla sama, mis kõrgtaseme formaadis). Samas ei tohiks siin olla üks-ühele ümberjutustus sama kasutusjuhu tüüpilisest sündmuste käigust.
Eeltingimused: Tingimused, mis peavad olema täidetud selleks, et kasutusjuht võiks käivituda. Siin kirjeldatakse andmed, mis peavad süsteemile teada olema, et see kasutusjuht võiks alata.
Järeltingimused: Süsteemi seisund peale kasutusjuhu läbimist. Siin kirjeldatakse:
- andmeobjektide tekkimist/muutmist/kustutamist;
- andmeobjektide atribuudi väärtuste lisamist/muutmist/kustutamist;
- andmeobjektide vaheliste seoste tekkimist/kustutamist.

Tüüpiline sündmuste käik: Kasutusjuhu tüüpilise läbimise stsenaarium. See esitatakse süsteemi ja kasutaja vahelise dialoogina. Soovitatakse, et see dialoog peaks olema 7+/-2 sammu pikk. Pikema dialoogi korral tuleb otsida võimalusi kasutusjuhu jagamiseks mitmeks eraldi kasutusjuhuks ning nende vahel <<includes>> või <<extends>> seose kasutamiseks.
Alternatiivid: Tüüpilisest kasutusjuhu läbimisest erinevad, alternatiivsed sammud. Igale alternatiivile lisatakse viide tüüpilise stsenaariumi sammule, millega see on seotud (st. mille alternatiiviks see on).

2.8 Eesmärkide vaade

Järgnevalt mõni sõna eesmärkide kirjeldamise kohta.

Eesmärgid moodustavad hierarhia (Marshall, 1999). On üldised eesmärgi (visioon ja missioon), mis on abstraktsed ja mille täitmist on suhteliselt raske

mõõta. Nt. “hõlbustada infotööd” – Mida tähendab hõlbustamine? Millisel juhul saab väita, et infotöö tegemine on hõlpsamaks muutunud? Üldiste eesmärkideni jõudmiseks tuleb saavutada alameesmärkide täitmine, mille täitmine on juba täpsemalt mõõdetav. Iga kasutusjuht kirjeldab, kuidas ühte või mitut sellist alameesmärki saavutada. Iga süsteemi funktsionaalsusega seotud alameesmärgi kohta peab leiduma vähemalt üks kasutusjuht, mis kirjeldab, kuidas seda eesmärki saavutada.

Eesmärk peab vastama järgmistele tingimustele (Robertson ja Robertson, 1999).

- *Mõistlik* – eesmärgi täitmisel saavutatav eelis peab olema suurem kui süsteemi loomiseks kuluv pingutus.
- *Teostatav* – eesmärk peab olema mõõdetav. **Mõõt** on kriteerium, mille alusel saab hiljem hinnata eesmärgi täidetust. Näiteks võib luua infosüsteemi teehooldusega tegelevale ettevõttele. Teehooldajad peavad tagama talvel libedaga teede liivatamise. Infosüsteem aitab teede hooldamist jälgida ja hooldajaid vajalikku kohta suunata. Näiteks võib püstitada eesmärgi, et liiva kulu peab vähenema 20 %. See saavutatakse nii, et juba liivatatud ja korras teelõiku uuesti ei liivatata. Järelikult peab teehooldajatele olema kättesaadav värske informatsioon teede olukorra ja teehoolduseks juba tehtud tegevuste kohta.
- *Saavutatav* – arendajal peab olema oskus eesmärk täita ja tellijal peab olema oskus eesmärgis kirjeldatud süsteemi kasutada.

Jälgige, et infosüsteemi eesmärgid aitavad kaasa ka organisatsiooni üldiste eesmärkide saavutamisele.

2.9 Küsimustik kasutusjuhtude mudeli kohta

1. Kas kasutusjuhtude diagramm ja kasutusjuhtude tekstikirjeldused on kooskõlas?
1.1 Kas kasutusjuhtudel/tegutsejatel on diagrammil ja tekstikirjeldustes ühesugune nimi?
1.2 Kas tekstikirjeldustes ja diagrammil on täpselt ühesugused kasutusjuhu ja tegutseja vahelised seosed?
1.3 Kas tekstikirjeldustes ja diagrammil on täpselt ühesugused kasutusjuhtude vahelised seosed?
2. Kas süsteemi piir on diagrammis ja tekstikirjeldustes üheselt määratud (infosüsteem, organisatsioon)?
3. Kas diagrammis on kasutatud korrektset tähistust?
4. Kas tekstikirjeldustes on olemas kõik vajalikud osad?
5. Kas on iga kasutusjuhu kohta saab leida tegutseja eesmärgi, mida see kasutusjuht täidab?
6. Kas mõni kasutusjuht on liigne?
7. Kas mõni kasutusjuht on puudu?
8. Kas iga kasutusjuht kirjeldab suhteliselt mahukat protsessi, mitte üksikut sammu või toimingut protsessis?
9. Kas kasutusjuht pole mitte liiga mahukas (süsteemi ja kasutaja dialoog on pikem kui 7 sammu) ning seetõttu tuleks see lahutada mitmeks väiksemaks kasutusjuhuks?

10. Kas kasutusjuhu tekstikirjelduses viidatakse igale kasutusjuhule, mida see kasutab läbi <<include>>- seose?

3. Kontseptuaalne andmemudel

Infosüsteemi projekteerimise käigus läbiviidava strateegilise ja detailanalüüsi olulises tulemuseks on *kontseptuaalne andmemudel*.

Kontseptuaalne andmemudel on *mittetehniline* mudel, mis kirjeldab süsteemi baasandmeid. See mudel kirjeldab *nõudmisi* andmebaasile, aga mitte andmebaasi tehnilist realisatsiooni. Seega kontseptuaalses andmemudelis ei pöörata tähelepanu ühelegi andmebaasi *realiseerimisega* seotud asjaolule, nagu:

- andmemudel (näiteks relatsiooniline mudel),
- kasutatav andmebaasisüsteem,
- kasutatav riistvara platvorm,
- kasutatava arvutivõrgu ülesehitus,
- töökiiruse küsimused.

Kontseptuaalne andmemudel on aluseks andmebaasi tehniliste kavandite loomisele, mis arvestavad andmemudeliga (näiteks relatsiooniline mudel) ning andmebaasi loomiseks kasutatava tarkvara- ja riistvara platvormiga (sealhulgas kasutatava andmebaasisüsteemiga). Kontseptuaalse andmemudeli esialgsed visandid e. eskiisid luuakse strateegilise analüüsi käigus ning mudeli koostamine viiakse lõpule detailanalüüsi käigus.

Date (2003) nimetab kontseptuaalse andmemudeli koostamist andmebaasi *semantiliseks modelleerimiseks*. Semantiline mudel üritab kirjeldada süsteemi tähendust. Kontseptuaalne andmemudel kirjeldabki süsteemis säilitama hakatavate andmete tähendust, struktuuri ning andmetega seotud kitsendusi. Öeldakse ka, et kontseptuaalne andmemudel valmib andmebaasi *kontseptuaalse disaini* tulemusena.

Kontseptuaalse andmemudeli oluliseks koostisosaks on visuaalne mudel (diagramm), mis kirjeldab andmebaasi kontseptuaalset struktuuri. Kaks levinud diagrammi tüüpi (ja tegelikult ka vastavat modelleerimise viisi) taolise informatsiooni esitamiseks on:

- Olemi-suhte diagramm (ingl. k. *entity-relationship diagramm, ERD*)
- Objekti-rolli mudel (ingl. k. *object-role model, ORM*) (http://www.orm.net)

Kontseptuaalse andmemudeli võimalikku struktuuri kirjeldab järgmine valem:

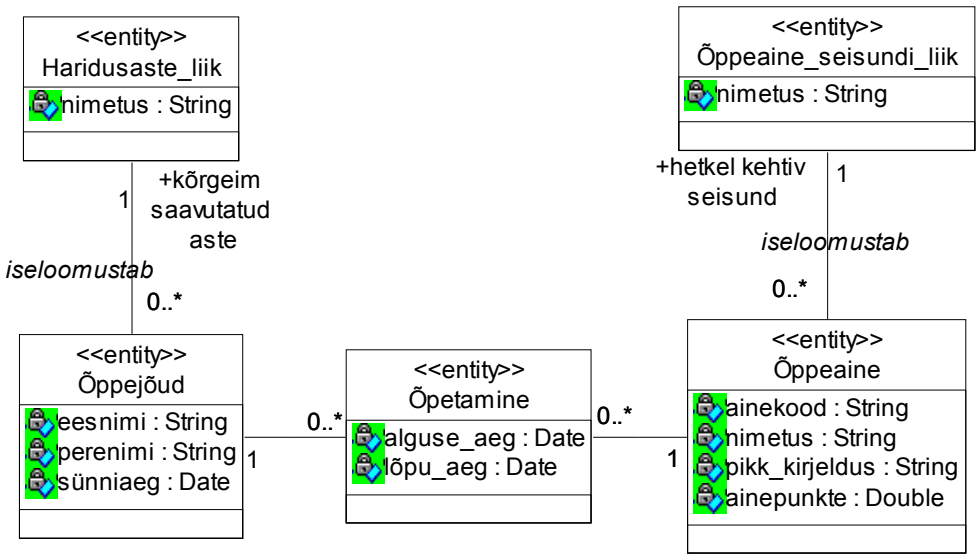
Kontseptuaalne andmemudel = andmebaasi kontseptuaalset struktuuri esitavad diagrammid + diagrammide elementide tekstilised spetsifikatsioonid (semantika kirjeldus) + andmetega seotud kitsenduste spetsifikatsioonid.

Väga levinud meetodiks on kasutada kontseptuaalse andmebaasi disaini tulemuste esitamiseks olemi-suhte diagramme. Tänapäeval luuakse olemi-suhte diagrammid sageli kasutades UMLi *klassidiagramme*. UML oli algselt mõeldud objekt-orienteeritud analüüsi ja disaini läbiviimiseks, kuid selle laiendusmehhanismid võimaldavad teda kasutada universaalse keelena paljude modelleerimisülesannete lahendamiseks.

3.1 Olemi-suhte diagramm

Väga levinud meetodiks on kasutada kontseptuaalse andmebaasi kirjeldamiseks olemi-suhte diagramme. Olemi-suhte diagrammid pakuti Peter Cheni poolt välja 1970-ndate keskel (Chen, 1976). Chen soovis luua esitusviisi, millega saaks modelleerida nii hierarhilisi-, võrk-, kui ka relatsioonilisi andmebaase. 1986 pakuti välja laiendatud olemi-suhte diagramm (Theory et al., 1986). Muuhulgas võimaldas see laiendus kirjeldada üldistusseoseid.

Aja jooksul on välja pakutud palju erinevaid märgisüsteeme e. notatsioone, mida saab kasutada olemi-suhte diagrammide joonistamiseks. Üks tuntud märgisüsteem on näiteks Richard Barkeri poolt väljapakutud Barkeri märgisüsteem (Barker, 1990). Selle võttis kasutusele Oracle Corporation integreerides selle arendusmetoodikaga Oracle*CASE ning CASE (Comupter Aided System Engineering) vahendiga Oracle Designer. Taolist märgisüsteemi kasutatakse ka näiteks L. Silverstoni koostatud universaalseid andmemudeleid esitavates raamatutes (Silverston, 2001a; Silverston, 2001b). Olemi-suhte diagramm on staatikadiagramm, mis kirjeldab süsteemi struktuuri aspekti.



Joonis 6 Olemi-suhte diagrammi näide.

Tänapäeval luuakse olemi-suhte diagrammid sageli kasutades UMLi *klassidiagramme*. UML oli algselt mõeldud *objekt-orienteeritud* analüüsi ja disaini läbiviimiseks, kuid selle laiendusmehhanismid võimaldavad teda kasutada universaalse keelena paljude modelleerimisülesannete lahendamiseks.

Käesolevas raamatus käsitleme me kontseptuaalse andmemudeli koostamist kasutades selleks UMLis loodud olemi-suhte diagramme.

3.2 Olemi-suhte diagrammi komponendid

Olemi-suhte diagrammil esitatakse olemitüübid, olemitüüpide vahelised seosetüübid ning olemitüüpide atribuudid.

Igale kontseptuaalses andmemudelis esitatud olemitüübile vastab reaalses maailmas (objektsüsteemis) eksisteeriv konkreetne või abstraktne objekt (asi, olem), mille kohta tuleb andmebaasis andmeid säilitada.

Valiku aitavad teha modelleerija enda kogemused ning teiste arendajate kogemused, mis võivad olla esitatud mustrite või taaskasutatavate mudelite abil.

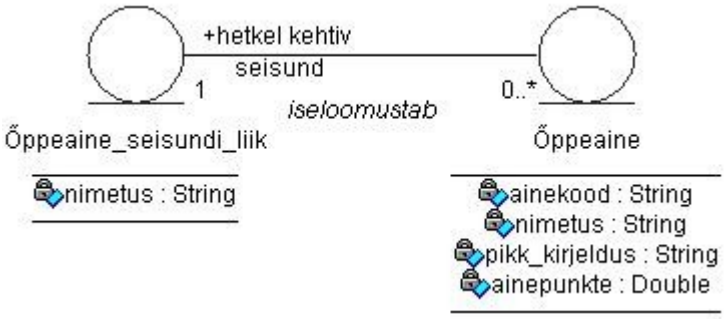
Olemitüüp

Olem: Suvaline konkreetne või abstraktne asi, mis eksisteerib, eksisteeris või võiks eksisteerida, kaasa arvatud nende asjade ühendused. Üksikuid olemeid olemi-suhte diagrammil ei esitata. Olemi-suhte diagrammil esitatakse olemite üldistused – olemitüübid. Näiteks on olemid konkreetsed õppeained nagu "Andmebaaside projekteerimine" ainekoodiga IDU3381 ja "Andmebaaside programmeerimine" ainekoodiga IDU0120.

Olemitüüp: Ühiste nimeliste omadustega olemite hulk. Võib öelda, et tegemist on reaalses maailmas eksisteerivate olemite üldistusega. Kontseptuaalses andmemudelis tuleb esitada sellised ja ainult sellised olemitüübid, millele vastavate olemite kohta soovitakse hakata hoidma andmebaasis andmeid. Segadust tekitab, et mõned allikad kasutavad termini "Olemitüüp" asemel terminit "Olem" ja termini "Olem" asemel terminit "Olemieksemplar".

Olemi-suhte diagrammil esitatakse olemitüüp klassisümboli abil. Klassile võib määrata *stereotüübi* <<entity>> tõstmaks esile, et tegemist ei ole mitte tavalise tarkvaraklassiga, vaid see esitab kontseptuaalses andmemudelis olemitüüpi. Mis on stereotüüp? Stereotüüp on UMLi laiendusmehhanism, mis võimaldab luua uut tüüpi mudelielemente, tuletades neid olemasolevatest elementidest (nt. klass, atribuut, assotsiatsioon, pakett). Neil uutel elementidel on oma spetsiifilised omadused (sildid), semantika ja visuaalne tähistus. Joonis 6 oleval olemi-suhte diagrammil esitatakse olemitüübid *Õppejõud*, *Õpetamine*, *Õppeaine*, *Haridusaste_liik* ja *Õppeaine_seisundi_liik*.

Nagu öeldud, võimaldab stereotüüp kasutada alternatiivset visuaalset tähistust. Joonis 7 esitatakse olemitüübid (*Õppeaine_seisundi_liik* ja *Õppeaine*), kasutades stereotüübi <<entity>> poolt määratud alternatiivset visuaalset tähistust. Segaduse vältimiseks on oluline, et ühes mudelis ei kasutataks läbisegi erinevaid visuaalseid tähistusi.



Joonis 7 Olemi-suhte diagrammi näide.

Seosetüüp

Seos, suhe, side: Seos on tunnetatud ühendus olemite vahel. Üksikuid seoseid olemi-suhte diagrammil ei esitata. Olemi-suhte diagrammil esitatakse seoste üldistused – seosetüübid.

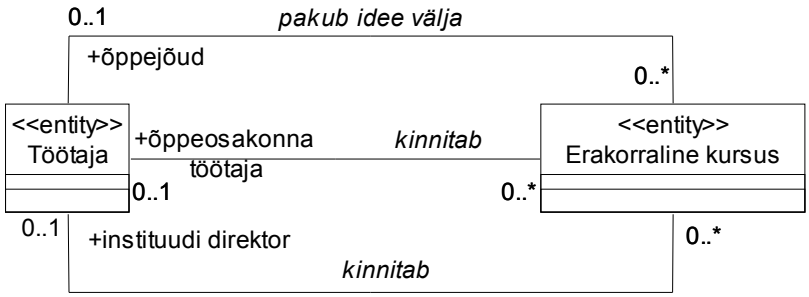
Näiteks: Õppejõud "Andres Mets" õpetab õppeainet "Tuumafüüsika" alates kuupäevast 1. september 2008 kuni kuupäevani 1. veebruar 2009.

Seosetüüp e. suhtetüüp: Kontseptuaalses andmemudelis esitatav seosetüüp on selliste seoste üldistus, mille kohta tuleb andmebaasis andmeid säilitada. Olemi-suhte diagrammil esitatakse seosetüübid joontena. Seosetüübid võivad olla erinevat liiki (assotsiatsioon, osa-terviku seos, üldistusseos) ning nendest liikidest tuleb juttu edaspidi. Iga seosetüübi kohta tuleb kindlaks teha ning dokumenteerida ka selle omadused.

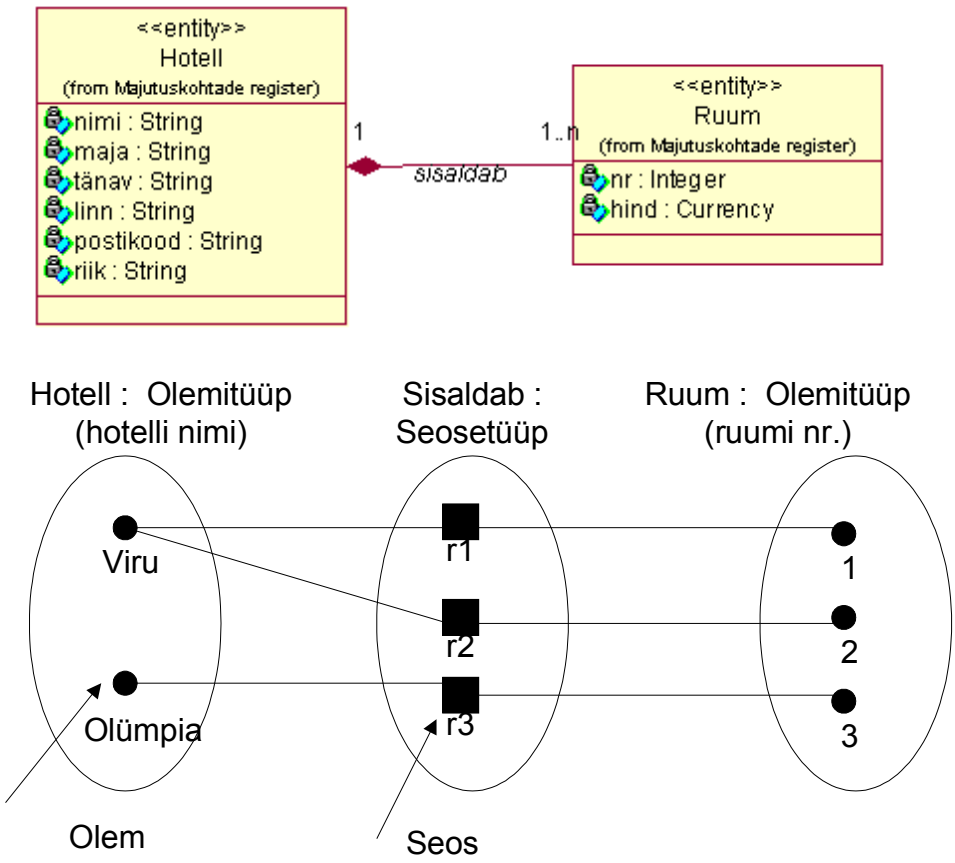
Mis on ühe modelleerija jaoks seosetüüp, see on teise jaoks olemitüüp ja vastupidi. Näiteks süsteemi kirjeldava lause "Õppejõud õpetab õppeainet" põhjal võib leida olemitüübid *Õppejõud* ja *Õppeaine*. Kuid *Õpetamine* on võimalik modelleerida nii seosetüübina *Õppejõud* ja *Õppeaine* vahel kui ka olemitüübina. Kui seosetüübil on nimelisi omadusi, millele vastavaid andmeid tuleb andmebaasis hoida, siis tuleb see seosetüüp modelleerida olemitüübina. Antud juhul tuleks õpetamise kohta registreerida ka õpetamise alguse aeg ja lõpu aeg ning seetõttu on Joonis 6 esitatud ka olemitüüp *Õpetamine*.

Roll: Roll kirjeldab olemitüübi tähendust seosetüübi kontekstis, milles ta osaleb. Rollid aitavad selgitada seosetüüpide tähendust. Olemi-suhte diagrammil esitatakse rollid seosetüüpide otstesse kirjutatud tekstina "+rollinimi". Eriti oluline on rollid määrata, kui kahe olemitüübi vahel on rohkem kui üks seosetüüp.

Joonis 8 esitatud diagrammil on *Töötaja* ja *Erakorraline kursus* vahel kolm seosetüüpi. Iga erakorralise kursuse idee on pakutud välja null või ühe töötaja poolt, kes on rollis "õppejõud". Iga erakorraline kursuse idee on kinnitatud null või ühe töötaja poolt, kes on rollis "õppeosakonna töötaja". Iga erakorraline kursus on kinnitatud null või ühe töötaja poolt, kes on rollis "instituudi direktor".



Joonis 8 Mitu seosetüüpi kahe olemitüübi vahel.



Joonis 9 Seosetüübi semantika.

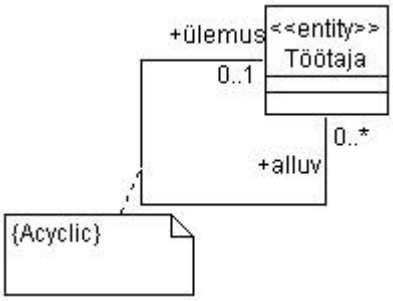
Seosetüübi aste: Seosetüübi aste on seosetüübis osalevate olemitüüpide (osaliste) arv. Seega näiteks:

- üheliikmelises e. unaarses seosetüübis osaleb üks olemitüüp (erinevates rollides) (vt. *rekursiivne seosetüüp*).
- kaheliikmeliikmelises e. binaarses seosetüübis osaleb kaks olemitüüpi.
- kolmeliikmelises e. ternaarses seosetüübis osaleb kolm olemitüüpi.

Kõik Joonis 6 ja Joonis 8 esitatud seosetüübid on kaheliikmelised. Kaheliikmelised seosetüübid on olemi-suhte diagrammidel kõige levinumad.

Rekursiivne seosetüüp: Rekursiivne seosetüüp on seosetüüp, kus seosetüübi mõlemas otsas on osaliseks sama olemitüüp, kuid *erinevates* rollides. Rekursiivset seosetüüpi kutsutakse ka *unaarseks* seosetüübiks.

Joonis 10 esitatud diagrammil on rekursiivne seosetüüp mille mõlemas otsas on sama olemitüüp (*Töötaja*), kuid erinevates rollides (*ülemus*, *alluv*). Diagrammi kohaselt on igal töötajal null või üks otsene ülemus ja null või rohkem otsest alluvat.



Joonis 10 Rekursiivse seosetüübi näide.

Mida tähendab märkmelehele kirjutatud "{Acyclic}"? Tegemist on kitsendusega, mille kohaselt töötajate vahelistes seostes ei tohi leiduda "tsükleid". See tähendab, et töötaja ei tohi olla iseenda otsene ega kaudne ülemus (näiteks ülemuse ülemus) ja otsene ega kaudne alluv (näiteks alluva alluv). Taoliste kitsenduste leidmine ja dokumenteerimine on vajalik, sest tegemist on ärireeglitega, millele vastavad kitsendused tuleb jõustada ka andmebaasis (andmebaasi kitsendustena).

Kui joonistada olemi-suhte diagrammi UMLis, siis võib kitsenduste kirjapanemiseks kasutada UMLi formaalset objektikitsenduste keelt (Object Constraint Language e. OCL), aga seda võib kirja panna ka vabas vormis tekstina. UML 2.0 ei nõua kitsenduste kirjeldamisel kindla keele kasutamist. Nõutud on vaid, et kitsendus tuleb panna looksulgudesse {}.

Millist keelt oleks soovitav kitsenduste kirjeldamiseks kasutada? Esitame järgnevalt näite OCL kitsendusest, mille kohaselt identifitseerib atribuudi *tellimuse_nr* väärtus unikaalselt iga olemitüüpi *Tellimus* kuuluvat olemit. Täpsemalt, tellimused t1 ja t2 on erinevad tellimused, kui nende tellimuse numbrid on erinevad.

```
[context Tellimus]
Tellimus.allInstances -> forAll ( t1, t2 | t1<>t2 implies
t1.tellimuse_nr<>t2.tellimuse_nr)
```

Kui kitsenduse kirjeldamiseks kasutada formaalset keelt, siis saab panna kitsenduse kirja täpselt ja üheselt arusaadavalt. Sellises keeles kitsenduste kirjeldusi suudab analüüsida ja interpreteerida ka tarkvarasüsteem. Samas, taolised kirjeldused pole ilma eriettevalmistuseta inimestele arusaadavad. Mudeleid peaksid üle vaatama ja nendest aru saama ka infosüsteemi tulevased kasutajad. Seega, kasulik oleks kitsendused dokumenteerida kahel viisil:

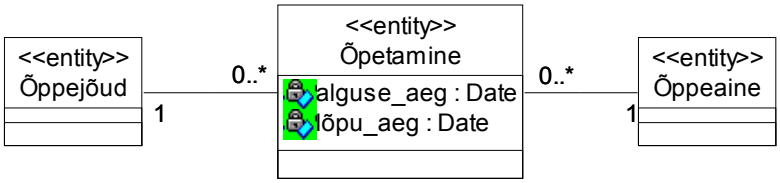
- Täpne ja formaalne kirjeldus kasutades näiteks OCLi.
- Laiemale kasutajate ringile arusaadav tavakeelne kirjeldus.

Vaatleme lauset: "Õppejõud õpetab õppeainet". *Õppejõud* ja *Õppeaine* on olemitüübid.

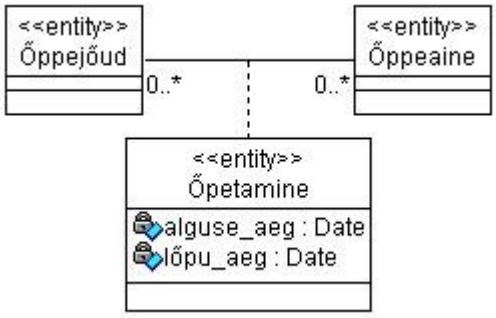
Mis on ühe modelleerija jaoks seosetüüp, see on teise jaoks olemitüüp ja vastupidi. Näiteks süsteemi kirjeldava lihtlause e. lausendi "Üliõpilane õpib õppeainet" põhjal võib leida olemitüübid *üliõpilane* ja *õppeaine*. Kuid *õppimine* on võimalik modelleerida nii seosetüübina *üliõpilane* ja *õppeaine* vahel kui ka olemitüübina.

Kui seosetüübil on atribuute, st. omadusi, mille kohta tuleb andmebaasis andmeid hoida, siis tuleb see seosetüüp modelleerida olemitüübina.

a)



b)



Joonis 11 Seosetüübi modelleerimise võimalused UMLi joonistatud olemi-suhte diagrammil.

Variandi b puhul kasutatakse seosetüübi esitamiseks UMLi sidemeklassi (ingl. k. *association class*). Fowler (2007) kirjutab: "Sidemeklass lisab ühe lisakitsenduse, mis seisneb selles, et iga kahe sidemes osaleva objekti vahel saab olla vaid üks sidemeklassi isend."

Variandid a ja b ei ole samaväärsed. Variandi b korral kehtib täiendav reegel, et üks õppejõud ei saa õpetada ühte õppeainet rohkem kui üks kord.

Atribuut

Atribuut: Atribuut on "nimeline olemi omadus" (EVS-ISO/IEC 2382, 1998). Kontseptuaalses andmemudelis tuleb kirjeldada sellised atribuudid, millele vastavaid andmeid soovitakse hakata andmebaasis hoidma.

Tüüp: Iga atribuut on mingit tüüpi. Atribuudi tüüp on võimalike väärtuste hulk, mille hulgast saavad tulla atribuudi väärtused. Mitmel atribuudil võib olla ühesugune tüüp.

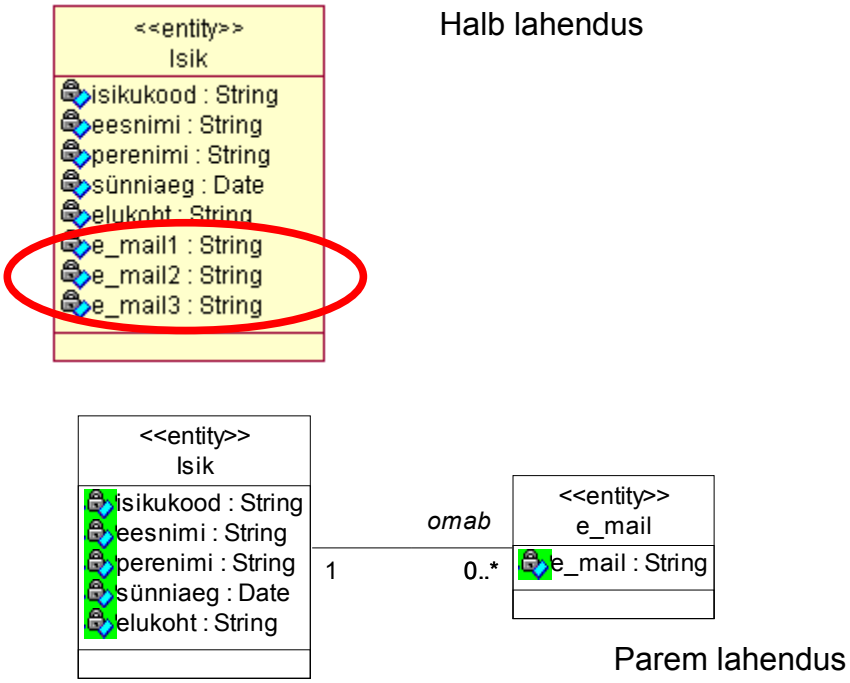
Fakultatiivne atribuut: Fakultatiivne atribuut on atribuut, millel mõne olemi korral ei ole väärtust. Näiteks olemitüübi *Isik* atribuut võib olla *neiupõlvenimi*. Kuid atribuudil *neiupõlvenimi* saab olla väärtus vaid naissoost isikute korral.

Kohustuslik atribuut: Kohustuslik atribuut on atribuut, millel iga olemi korral on vähemalt üks väärtus, mis on antud atribuudi tüüpi. Näiteks olemitüübi *Isik* atribuut võib olla *sünniaeg*. Igal isikul on *sünniaeg*.

Üheväärtuseline atribuut: Üheväärtuseline atribuut on atribuut, millel on iga olemi kohta maksimaalselt üks väärtus, mis on antud atribuudi tüüpi. Näiteks igal isikul on vaid üks *sünniaeg*.

Mitmeväärtuseline atribuut: Mitmeväärtuseline atribuut on atribuut, millel võib olla mõne olemi kohta rohkem kui üks väärtus, mis on antud atribuudi tüüpi.. Näiteks võib isikul olla üks või mitu telefoni või e-maili aadressi.

Kuidas modelleerida mitmeväärtuselist atribuuti?
- Modelleerida UMLis mitmeväärtuselise atribuudina.
- Modelleerida eraldi olemitüübina.



Joonis 12 Mitmeväärtuselise atribuudi modelleerimine.

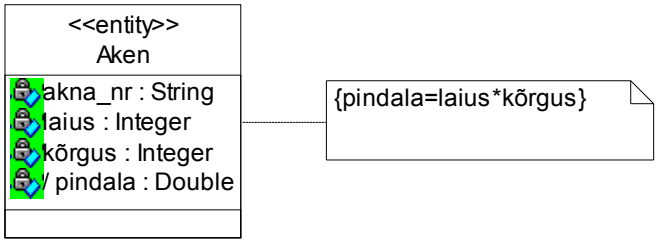
Esimene lahendus on halb, sest ühel isikul võib olla rohkem või vähem kui kolm e-maili aadressi.

Tuletisatribuut: Tuletisatribuut on atribuut, mille väärtus on mingi *arvutusreegli* alusel arvutatav muude väärtuste põhjal (näiteks sama olemitüübi teiste atribuutide väärtustest).

Tuletisatribuudi nime ette tuleks UMLis joonistatud olemi-suhte diagrammil panna kaldkiips "/", et seda atribuuti tähistada. Samuti tuleb dokumenteerida

arvutusreegel. Arvutusreegli võib panna kirja diagrammile paigutatud märkmelehele.

Näiteks hotellis ruumi üheks ööpäevaks *reserveerimise hind* on arvutatav reserveeritava ruumi ühe päeva hinna ja reserveeritud päevade arvu kaudu. Akna pindala on arvutatav akna kõrguse ja laiuse kaudu.



Joonis 13 Tuletisatribuudi näide.

Tuletatud väärtuste andmebaasis hoidmise eelised:
- Kiirendab päringuid.

Tuletatud väärtuste andmebaasis hoidmise puudused:
- Suureneb risk, et andmebaasi satuvad vastuolulised andmed
  - kõrgus*laius <> pindala
- Suurendab andmemahte

Märgime, et *vanuse* esitamine tuletisatribuudina ei ole otstarbekas, sest vanus on pidevas muutumises ning seega tuleks selle atribuudi väärtust sagedasti muuta. Seetõttu oleks mõistlik hoida andmebaasis olemi sünni/loomise aega. Teades hetke kuupäeva ja kellaaega on selle põhjal võimalik alati korrektne vanus välja arvutada (eeldusel, et süsteemis on loodud operaator, mis võimaldab leida kahe ajahetke vahelise intervalli).

Seosetüübi omadused

- Järk e. võimsus
- Osaluskohustus e. tugevus

Need omadused tuleb määrata iga seosetüübis osaleva olemitüübi kohta.

Tugevus

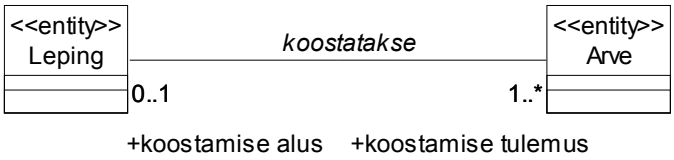
Tugevus näitab vaadeldava seosetüübi ST ja selles osaleva olemitüübi OT kontekstis minimaalset (tüüpi ST kuuluvate) seoste arvu, milles peab osalema iga (tüüpi OT kuuluv) olem.

Tugevuseks võib olla suvaline positiivne arv või 0.

Seosetüüp ST on olemitüübi OT jaoks *kohustuslik*, kui iga tüüpi OT kuuluv olem peab osalema vähemalt ühes tüüpi ST kuuluvas seoses (tugevus on suvaline positiivne arv).

Seosetüüp ST on olemitüübi OT jaoks *mittekohustuslik*, kui on lubatud tüüpi OT kuuluvad olemid, mis ei osale üheski tüüpi ST kuuluvas seoses (tugevus on 0).

Näiteks Joonis 14 on seosetüüp *koostatakse* olemitüübi *Leping* jaoks kohustuslik, sest iga leping peab olema seotud vähemalt ühe arvega. Samas on seosetüüp *koostatakse* olemitüübi *Arve* jaoks mittekohustuslik, sest võib leiduda arveid, mis pole seotud ühegi lepinguga.



Joonis 14 Seosetüübi kohustuslikkus.

Näide:
- Hotell **peab** olema seotud ühe või mitme ruumiga. Seega seosetüüp *Hotell ja Ruum vahel* on *Hotell* jaoks kohustuslik.
- Ruumiga **peab** olema seotud üks hotell. Seega seosetüüp *Hotell* ja *Ruum vahel* on *Ruum* jaoks kohustuslik.

Võimsus e. järk

Võimsus näitab vaadeldava seosetüübi ST ja selles osaleva olemitüübi OT kontekstis *maksimaalset* (tüüpi ST kuuluvate) seoste arvu, milles võib osaleda üks (tüüpi OT kuuluv) olem.

Järguks võib olla suvaline positiivne arv või * (piiramatu arvu korral).

Näide:
- Hotell on seotud ***ühe või mitme*** ruumiga.
- Ruumiga on seotud ***üks*** hotell.

- Võimsus ruumi puhul on üks, ses konkreetne ruum osaleb ühes seoses hotelliga. Ruum on seotud ühe kindla hotelliga.
- Võimsus hotelli puhul on *, sest konkreetne hotell osaleb ühes või rohkemas seoses ruumidega. Hotellis on üks või mitu ruumi.

Joonis 14 näitel on seosetüübi *koostatakse* kontekstis olemitüübi *Leping* tugevus 1 ja võimsus *. Seosetüübi *koostatakse* kontekstis on olemitüübi *Arve* tugevus 0 ja võimsus on 1.

Tugevus ja võimsus koos määravad ära *võimsustiku* (ingl. k. *multiplicty*). Võimsustikud määratakse ära alumise ja ülemise piiriga ning seega määrab tugevus ära alapiiri ja võimsus ülapiiri. Kui ala- ja ülapiir on sama võib kasutada ühte arvu. Järgnevalt esitame näiteid võimalikest võimsustikest.

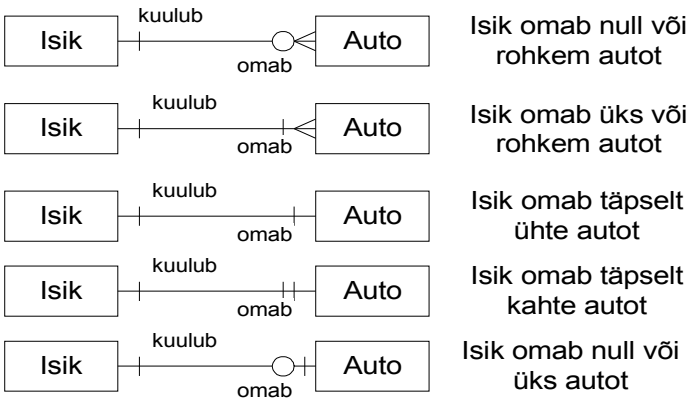
Tähistus	*Sõnaline selgitus*
0..1	Null või üks
1..1 (või 1)	Täpselt üks

Tähistus	*Sõnaline selgitus*
0..*	Null või rohkem
1..*	Üks või rohkem
5..10	Viis kuni kümme

Võimsustikud esitavad kitsendusi, mille põhjal luuakse andmebaasis kitsendusi.

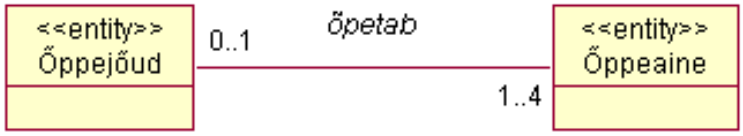
Erinevates notatsioonides esitatakse võimsustikke erinevalt.

Näide:



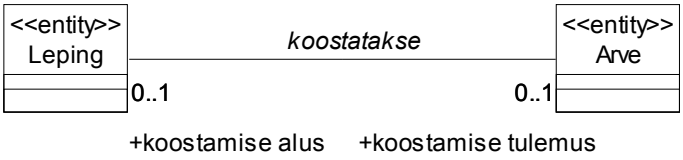
Joonis 15 Erinevad seoste tüübid esitatuna *varesejalgade notatsioonis*.

Õppejõud *peab* õpetama ühte kuni nelja õppeainet. (1..4)
Õppeainet *võib* õpetada üks õppejõud. (0..1)



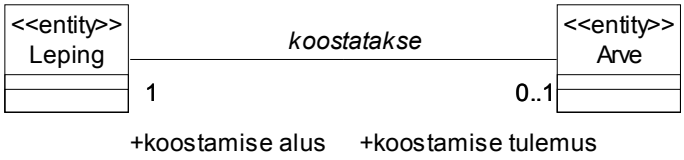
Joonis 16 Võimsustike näide.

Iga lepingu alusel *võib* koostada ühe ja ainult ühe arve. (0..1)
Iga arve *võib* olla koostatud ühe ja ainult ühe lepingu alusel. (0..1)



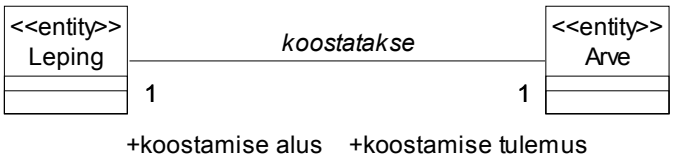
Joonis 17 Võimsustike näide.

Iga lepingu alusel *võib* koostada ühe ja ainult ühe arve. (0..1)
Iga arve *peab* olema koostatud ühe ja ainult ühe lepingu alusel. (1)



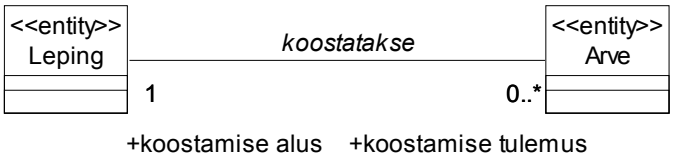
Joonis 18 Võimsustike näide.

Iga lepingu alusel *peab* koostama ühe ja ainult ühe arve. (1)
Iga arve *peab* olema koostatud ühe ja ainult ühe lepingu alusel. (1)



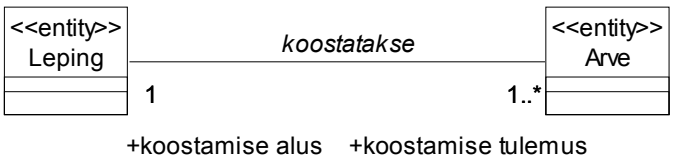
Joonis 19 Võimsustike näide.

Iga lepingu alusel *võib* koostada ühe või mitu arvet. (0..*)
Iga arve *peab* olema koostatud ühe ja ainult ühe lepingu alusel. (1)



Joonis 20 Võimsustike näide.

Iga lepingu alusel *peab* koostama ühe või mitu arvet. (1..*)
Iga arve *peab* olema koostatud ühe ja ainult ühe lepingu alusel. (1)



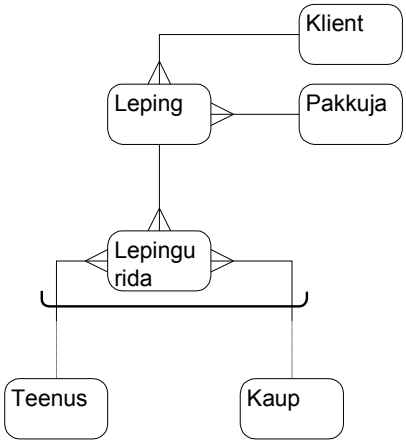
Joonis 21 Võimsustike näide.

Sõltuvalt võimsustikest võib kahte olemitüüpi siduv kaheliikmeline seosetüüp olla:

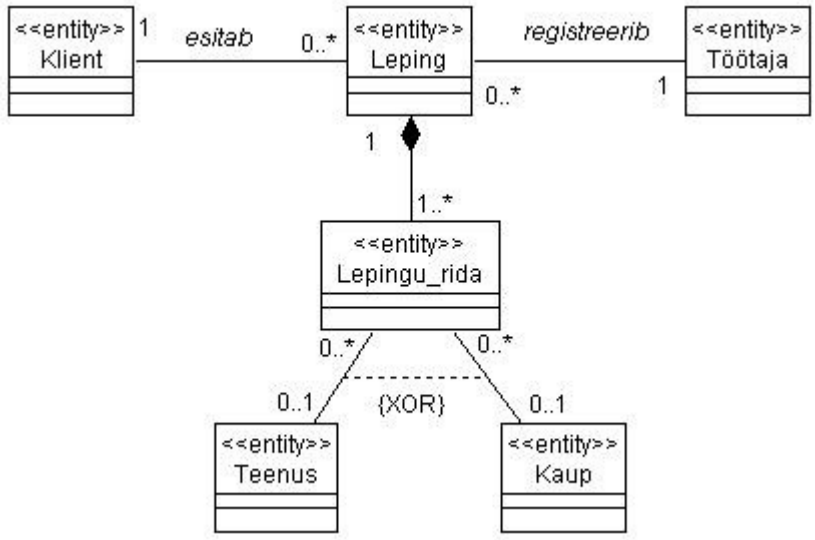
- üks-ühele (1:1) seosetüüp (vt. Joonis 17, Joonis 18, Joonis 19);
- üks-mitmele (1:M) seosetüüp (vt. Joonis 16, Joonis 20, Joonis 21);
- mitu-mitmele (M:N) seosetüüp;

Olemi-suhte diagrammil võib esitada **välistava kaare**. Välistavat kaart saab kirjeldada, kui olemitüüp OT on seosetüüpide kaudu seotud kahe või rohkema olemitüübiga. Kaar esitab kitsenduse, et iga olemi puhul (mis on tüüpi OT) peavad eksisteerima täpselt ühte kaarega hõlmatud seosetüüpi kuuluvad seosed.

Antud juhul peab iga lepingu rida olema seotud kas kauba või teenusega, aga mitte mõlemaga korraga. Teisisõnu, ei ole lubatud lepingu read, mis ei ole seotud ei kauba ega teenusega. Samuti ei ole lubatud lepingu read, mis on seotud nii kauba kui ka teenusega. Pange tähele, et seetõttu on iga lepingu rida seotud null või ühe teenusega ja null või ühe kaubaga, mitte täpselt ühe teenusega ja täpselt ühe kaubaga.

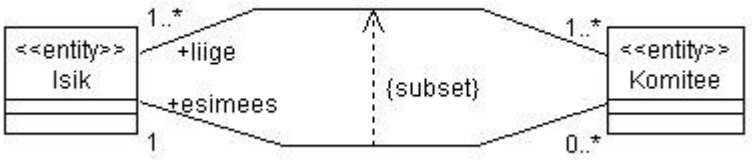


Joonis 22 Kaare esitamine olemi-suhte diagrammil, kasutades *varesejalgade notatsiooni*.



Joonis 23Kaare esitamine UMLis joonistatud olemi-suhte diagrammil.

Alamhulga kitsendus. UMLis saab esitada *alamhulga kitsenduse* (Halpin, 2001). Näite kohaselt saab iga isik olla vaid sellise komitee esimees, mille liige ta on (komitee esimeeste hulk on komitee liikmete alamhulk). Pange tähele, et kuna igal komiteel on täpselt üks esimees, siis peab tänu alamhulga kitsendusele igal komiteel olema üks või rohkem liiget.



Joonis 24Alamhulga kitsenduse esitamine UMLis joonistatud olemi-suhte diagrammil.

3.3 Laiendatud olemi-suhte diagramm

Üldistusseos

Ülatüüp on olemitüüp, millel on üks või rohkem alamtüüpi. Iga alamtüüpi olem on ühtlasi ka ülatüüpi olem, kuid vastupidine ei pruugi alati kehtida (st. iga ülatüüpi olem ei ole alati ka alamtüüpi olem). Alamtüüpi olem pärib kõik ülatüüpi olemi seosed ja atribuudid. Seetõttu nimetatakse üldistusseost ka *pärimisseoseks*. Alamtüübil võivad (aga ei pruugi olla) olla võrreldes ülatüübiga täiendavad omadused (atribuudid) ja/või see osaleb täiendavates seosetüüpides. Ühel ülatüübil võib olla mitu alamtüüpi. Ühel alamtüübil võib olla mitu ülatüüpi (*mitmene pärimine*).

Näiteks ülatüübi *Isik* atribuutideks võivad olla *isikukood*, *eesnimi*, *perenimi sünniaeg* ja *elukoht*. Ülatüübi *Isik* alamtüübid võivad olla *Üliõpilane* ja *Töötaja*. Nii töötaja kui ka üliõpilane on isikud. Alamtüüp *Töötaja* osaleb seosetüübis olemitüübiga *Amet*. Alamtüübil *Üliõpilane* on atribuut *üliõpilaskood*.

Üldistusseos ühendab omavahel ülatüübi ja selle alamtüübi. Diagrammil esitab seda ülatüüpi ja alamtüüpi ühendav joon. Selle joone ülatüübi poolses otsas on kolmnurk, mille tipp on suunatud ülatüübi poole.

Kuidas sõnaliselt väljendada olemi-suhte diagrammil esitatud üldistusseost? Joonis 28 esitatud diagrammi kohaselt: Üliõpilane *on* isik. Töötaja *on* isik.

Ülatüübiks/alamtüübiks olemine on suhteline – olemitüüp, mis on ühe üldistusseose kontekstis ülatüüp võib olla teise üldistusseose kontekstis alamtüüp.

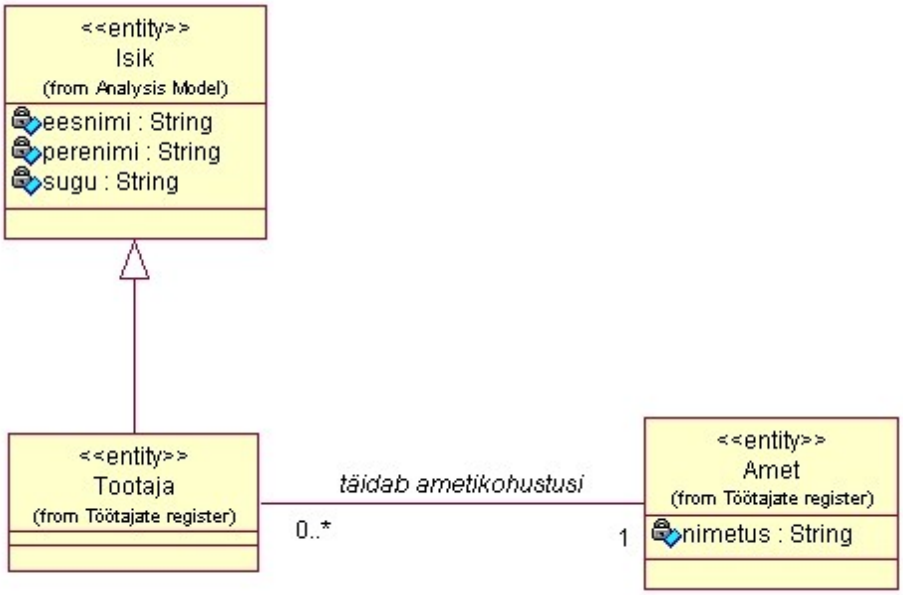
Alamtüüp peab vastama järgmistele kriteeriumitele (Larman, 2002, lk. 399).
- Ülatüübi definitsioon peab 100% kehtima ka selle alamtüübi suhtes.
- Kõik alamtüüpi olemid on ka ülatüüpi.

Üldistusseoste kaudu seotud olemitüübid moodustavad *tüüpide hierarhia*. Tüüpide hierarhiate leidmiseks võib kasutada **spetsialiseerimise** või **üldistamise** meetodit.

Spetsialiseerimine tähendab, et üritatakse leida olemitüüpe eristavaid omadusi. Näiteks leitakse olemitüüp *Isik* ja seejärel tehakse kindlaks, et isik võib olla nii *füüsiline* kui ka *juriidiline isik*.

Üldistamine tähendab, et üritatakse minimeerida erinevusi olemitüüpide vahel ja leida nende ühiseid omadusi. Näiteks leitakse olemitüübid *Ostutellimus* ja *Müügitellimus* ja tehakse üldistus, et tegemist on *Tellimusega* ning veel üldisemalt *Lepinguga* ostjate ja müüjate vahel.

Praktikas tuleb neid meetodeid modelleerimisel kasutada üheskoos.



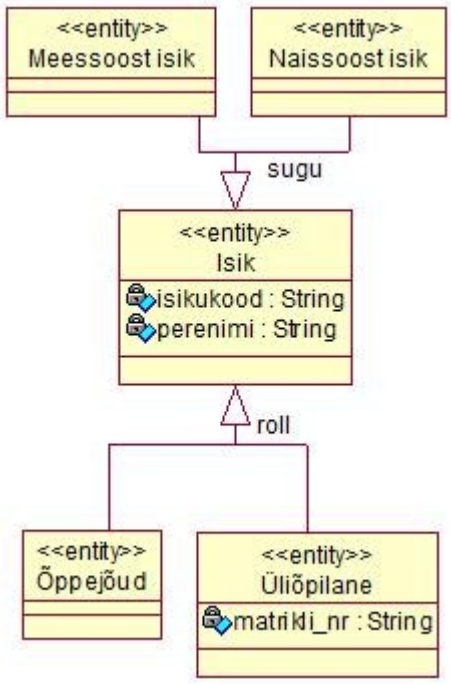
Joonis 25 Üldistusseose näide.

Isik on ülatüüp ja *Tootaja* on alamtüüp. Seost loetakse järgnevalt: "Töötaja **on** Isik"

Millal oleks kasulik leida ja mudelil esitada nii ülatüüp kui ka selle alamtüübid?

- Osa atribuute on ühised kõikidele alamtüüpidele ja need tuleks esitada ülatüübis. Osa atribuute aga on spetsiifilised mingile kindlale alamtüübile.
- Osa seosetüüpe on ühised kõikidele alamtüüpidele ja need tuleks esitada ülatüübi tasemel. Osa seosetüüpe aga on spetsiifilised mingile kindlale alamtüübile.

Üldistuste hulk on kogum üldistusseoseid, mis üheskoos kirjeldavad, kuidas ülatüüpi mingil kindlal viisil alamtüüpideks jagada. Joonis 26 esitatakse kaks üldistuste hulka, mille nimed on *roll* ja *sugu*. Ühte hulka kuuluvad üldistusseosed on visuaalselt „kokku tõmmatud“. Üldistuste hulk *sugu* jagab ülatüübi *Isik* alamtüüpideks vastavalt isiku soole. Üldistuste hulk *roll* jagab ülatüübi *Isik* alamtüüpideks vastavalt isiku rollile. Mõlemasse üldistuste hulka kuuluvate üldistusseoste korral on *Isik* ülatüüp.



Joonis 26 Üldistuste hulgad.

Üldistusseostega koos tuleks ära määrata ka nendega seotud kitsendused (Connolly ja Begg, 2002). Nende kitsenduste alusel luuakse hiljem jällegi andmebaasi kitsendusi.

Osaluskohustus määrab, kas iga ülatüüpi kuuluv olem peab kuuluma ka mõnda alamtüüpi:

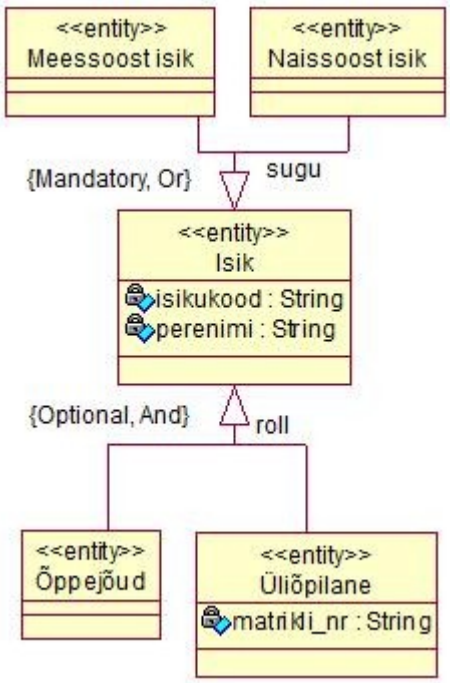
- Kui iga ülatüüpi kuuluv olem peab kuuluma vähemalt ühte alamtüüpi, siis tähistatakse seda diagrammil {Mandatory}.
- Kui iga ülatüüpi kuuluv olem ei pea kuuluma mõnda alamtüüpi (see tähendab, et võib leiduda ülatüüpi kuuluvaid olemeid, mis ei kuulu ühtegi alamtüüpi, siis tähistatakse seda diagrammil {Optional}.

Kuuluvus määrab maksimaalse alamtüüpide arvu kuhu ülatüüpi olem võib kuuluda:

- Kui iga ülatüüpi olem saab kuuluda maksimaalselt ühte alamtüüpi, siis tähistatakse seda diagrammil {Or}.
- Kui iga ülatüüpi olem saab kuuluda rohkem kui ühte erinevasse alamtüüpi, siis tähistatakse seda diagrammil {And}.

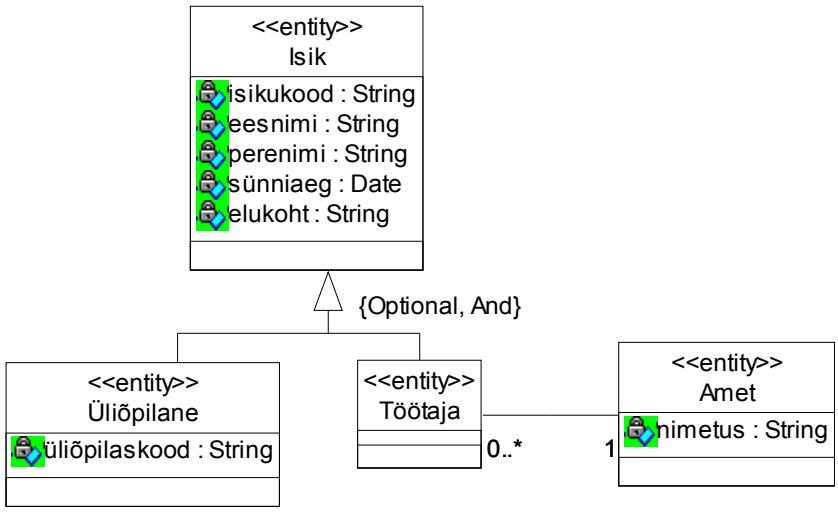
Kuuluvuse kitsendust pole vaja esitada, kui ülatüübil on ainult üks alamtüüp.

Osaluskohustuse ja kuuluvuse kitsendused tuleb esitada eraldi iga üldistuste hulga kohta (vt. Joonis 27).



Joonis 27 Üldistuste hulgad koos kitsendustega.

Vaatleme Joonis 28 esitatud kitsendust "{Otional, And}". "Optional" tähendab, et võib leiduda isikuid, kes ei ole ei üliõpilased ega töötajad. "And" tähendab, et võib leiduda isikuid, kes on korraga nii üliõpilased kui ka töötajad. Pange tähele, et diagrammil võib, aga ei pea kitsendust kirjutama märkmelehele.



Joonis 28 Üldistusseose näide.

Oletame, et Joonis 28 oleks esitatud üldistusseosega seotud kitsendus "{Mandatory, Or}". "{Mandatory}" tähendab, et iga isik peab olema ka mingit alamtüüpi (üliõpilane, töötaja). "{Or}" tähendab, et isik saab olla kas üliõpilane või töötaja, aga mitte mõlemat korraga.

Osa-terviku seos

Agregatsioon esitab osa-terviku seost olemitüüpide vahel kus üks olemitüüpidest esitab *osa* ja teine *tervikut*. Üks osa võib kuuluda mitmesse

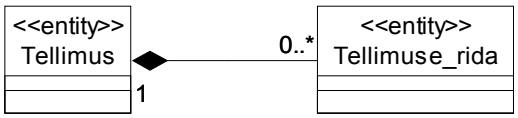
tervikusse ning osa võib eksisteerida ka ilma tervikuta. Näiteks iga meeskond kuulub nulli või rohkemasse liigasse ja igasse liigasse peab kuuluma vähemalt kolm meeskonda. Larman (2002) märgib, et selline seos eksisteerib valdavalt abstraktsete, mitte füüsiliste objektide vahel.

Diagrammil esitatakse agregatsiooni seosetüüpi tähistava joone otsas paikneva seest tühja rombina (valge rombina), mis paikneb *tervikut* esitava olemitüübi poolel. Joonis 29 esitatud diagrammil on tervikuks *Liiga* ja osaks *Meeskond*.

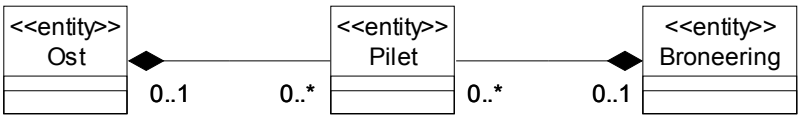
Joonis 29 Agregatsiooni näide.

Kompositsioon on osa-terviku seose erijuhus, mille puhul peab *osa* rollis olev olem kuuluma täpselt ühte *terviku* rollis olevasse olemisse. Näiteks sõrmed on osa täpselt ühest käest (kui just pole tegu siiami kaksikutega) ja tellimuse rida on osa täpselt ühest tellimusest (vt. Joonis 30). Joonis 31 kohaselt peab iga konkreetne pilet olema kas täpselt ühe broneeringu osa või täpselt ühe ostu osa. Konkreetne pilet ei tohi korraga kuuluda nii mingisse broneeringusse kui ka ostu.
Lisaks kehtestab kompositsioon reegli, et terviku rollis oleva olemi kustutamisel tuleb kustutada ka kõik sellele kuuluvad *osa* rollis olevad olemid (Fowler, 2007).

Diagrammil esitatakse kompositsiooni seosetüüpi tähistava joone otsas paikneva täidetud rombina (musta rombina), mis paikneb tervikut esitava olemitüübi poolel. Joonis 30 esitatud diagrammil on *Tellimus* tervik ja *Tellimuse rida* selle osa.



Joonis 30 Kompositsiooni näide.



Joonis 31 Kompositsiooni näide.

Kui *kahtlete* kas osa-tervikus seost esitada, siis Larman (2002) soovitab seda mitte teha. Millal osa-terviku seost kasutada (Larman, 2002)?

- Loogilises mõttes on tegu osa ja tervikuga.
- Osa ja tervik on seotud eluea kaudu. Osa ei saa eksisteerida ilma tervikuta. Sellisel juhul on tegu kompositsiooniga.
- Mõned terviku omadused (nt. asukoht) või käitumised (nt. kustutamine) kanduvad üle ka osadele.

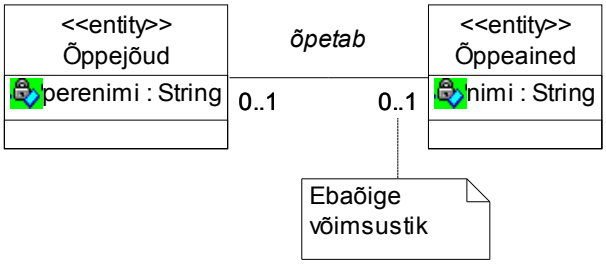
3.4 Nimedest

Kasutage olemitüüpide, atribuutide ja seosetüüpide nimetamisel sisulisi, infosüsteemi tulevastele kasutajatele arusaadavaid nimesid. Nii saavad ka nemad mudelite ülevaatamisel osaleda.

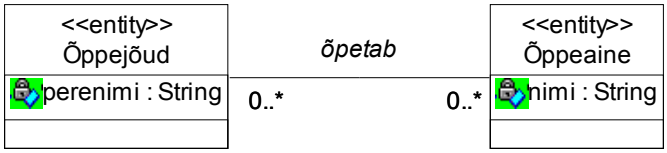
Kasutage olemitüüpide nimedes ainsuse vorme, sest nii on hiljem lihtsam määrata seosetüüpidega seotud võimsustikke. Oletame, et süsteemi kirjeldavad lausendid: Iga õppejõud õpetab null või rohkem õppeainet. Iga õppeainet õpetab null või rohkem õppejõudu. Iga õppejõudu iseloomustab perenimi. Iga õppeainet iseloomustab nimi.

Joonis 32 osas a) on ühe olemitüübi nimi mitmuses (*Õppeained*) ja sellest tulenevalt on võimsustiku määramisel langetatud vale otsus. Joonise osa b) esitab korrektse mudeli.

a)



b)



Joonis 32 Võimsustike määramine.

Ärge lähtuge nimetamisel andmebaasisüsteemide tehnilistest piirangutest nagu: nimi ei tohi olla pikem kui 30 märki, ei tohi sisaldada täpitähti. Pidage meeles, et kontseptuaalne andmemudel on mittetehniline mudel, mille põhjal võib luua tehnilise kavandi väga erinevate platvormide jaoks.

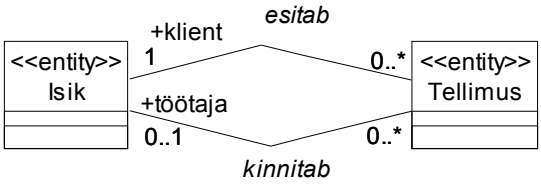
3.5 Ei ole ühte ainuõiget mudelit

Rõhutame, et kontseptuaalse andmemudeli koostamisel võivad erinevad modelleerijad koostada sama lähteinformatsiooni põhjal erinevad mudelid, mis kõik annavad edasi vajaliku informatsiooni

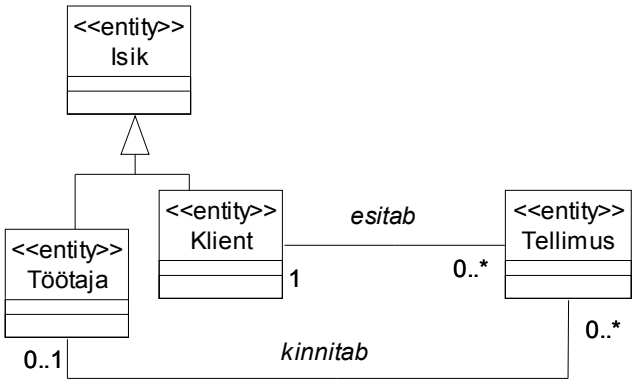
Oletame, et tuleb koostada kontseptuaalne andmemudel järgneva spetsifikatsiooni põhjal: Iga klient esitab null või rohkem tellimust. Iga tellimus on esitatud täpselt ühe kliendi poolt. Iga töötaja kinnitab null või rohkem tellimust. Iga tellimus kinnitatakse null või ühe töötaja poolt. Klient on isik. Töötaja on isik.

Järgnev joonis esitab kaks võimalikku olemi-suhte diagrammi, mida sellise spetsifikatsiooni põhjal võib koostada.

a)



b)



Joonis 33 Alternatiivsete modelleerimisviiside näide.

Variandis a on näidatud olemitüüpide rollid seosetüüpide kontekstis. Variandis b aga esitatakse rollid olemitüüpidena.

Nagu näete, on variant a) kompaktsem. Variant b võiks olla eelistatum kui olemitüüpidel *Töötaja* ja/või *Klient* on oma spetsiifilisi atribuute või seosetüüpe (mis ei käi kõigi isikute kohta). Samuti lihtsustab variant b) kitsenduste esitamist. Näiteks, kui kehtib reegel, et iga isik peab olema kas töötaja või klient, aga ta ei tohi olla mõlemat korraga, siis saab üldistusseosele lisada kitsenduse {Mandatory; Or}.

3.6 Olemitüüpide liigitus

Olemitüüpe on võimalik liigitada selle järgi, kuidas identifitseeritakse nendesse tüüpidesse kuuluvaid olemeid.

Nõrk olemitüüp on olemitüüp, millesse kuuluvate olemite eksistents sõltub mõnda teise olemitüüpi kuuluvate olemite eksistentsist (Connolly ja Begg, 2002). Seda tüüpi olem ei saa eksisteerida ilma mõne teise olemita. Seda tüüpi olem ei saa eksisteerida ilma mõne teise olemita. Kui olemitüüp OT on nõrk, siis sellesse kuuluvate olemite identifitseerimiseks ei piisa OT atribuutide väärtustest. Sellise olemi identifitseerimiseks on vaja kasutada ka selle seoseid teiste olemitega.

Näiteks on nõrgad olemitüübid:

- Joonis 6 esitatud olemitüüp *Õpetamine.* Õpetamine seob õppejõu ja õppeaine ning ei saa eksisteerida ilma, et eksisteeriksid õppejõud ja õppeaine.
- Joonis 28 esitatud olemitüübid *Töötaja* ja *Üliõpilane*. Töötaja ja üliõpilane on isikud.
- Joonis 30 esitatud olemitüüp *Tellimuse_rida*. Tellimuse rida on tellimuse osa ja ei saa eksisteerida ilma tellimuseta.

Tugev olemitüüp on olemitüüp, millesse kuuluvate olemite eksistents ei sõltu mõnda teise olemitüüpi kuuluvate olemite eksistentsist (Connolly ja Begg, 2002). Näiteks Joonis 6 esitatud diagrammil on tugevad olemitüübid *Haridusaste_liik*, *Õppejõud*, *Õppeaine* ja *Õppeaine_seisundi_liik*.

Kui olemitüüp OT on tugev, siis sellesse kuuluvate olemite identifitseerimiseks piisab OT atribuutide väärtustest. Näiteks konkreetse õppeaine identifitseerimiseks piisab atribuudi *ainekood* väärtusest.

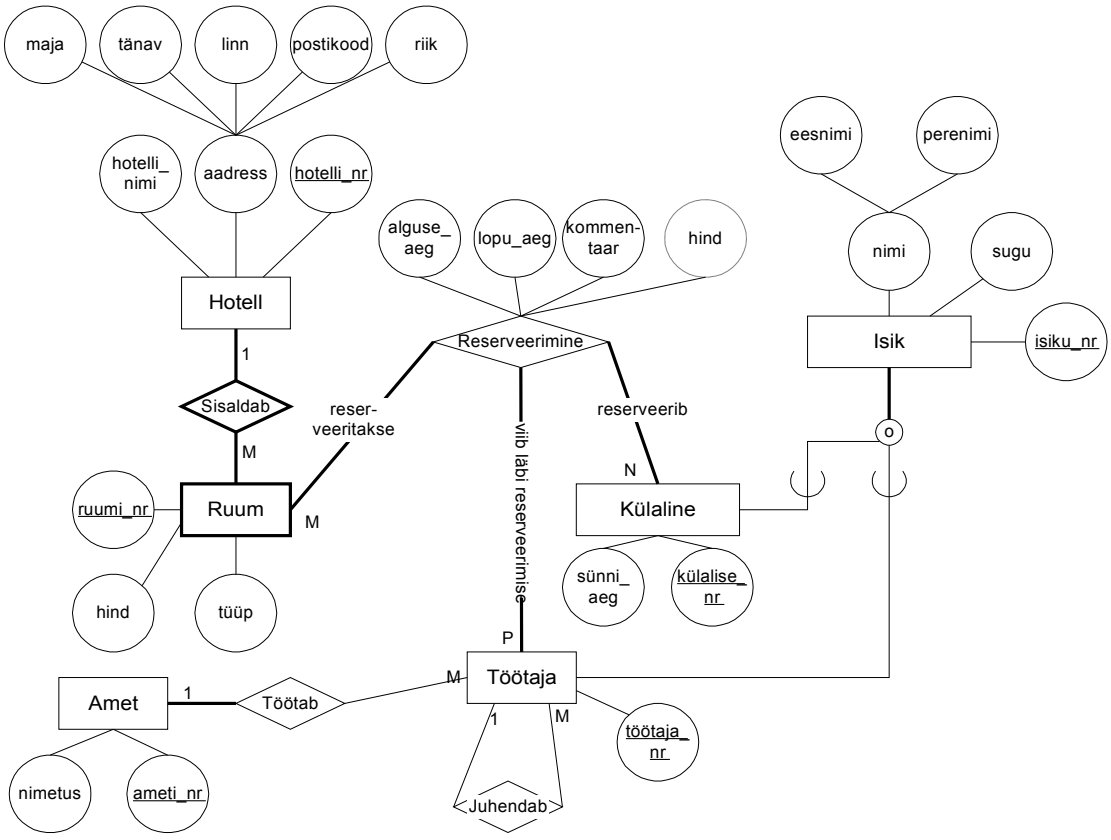
Codd (1979) esitab veel ühe võimaliku olemitüüpide liigituse.

- *Kernel*. Olemitüübid, millesse kuuluvad olemid võivad eksisteerida sõltumata teiste olemite olemasolust. Joonis 6 esitatud diagrammil on kernelid *Haridusaste_liik*, *Õppejõud*, *Õppeaine* ja *Õppeaine_seisundi_liik*.
- *Kirjeldav olemitüüp.* Seda tüüpi olemite peamine eesmärk on iseloomustada / kirjeldada teisi olemeid. Nende eksistents sõltub olemitest, mida nad iseloomustavad. Kirjeldava olemitüübi näiteks on Joonis 12 esitatud olemitüüp *e_mail*.
- *Assotsiatsioon.* Olemitüüp, mis esitab mitu-mitu seoseid. Joonis 6 esitatud diagrammil on assotsiatsioon *Õpetamine*.

3.7 Olemi-suhte diagrammi erinevad notatsioonid

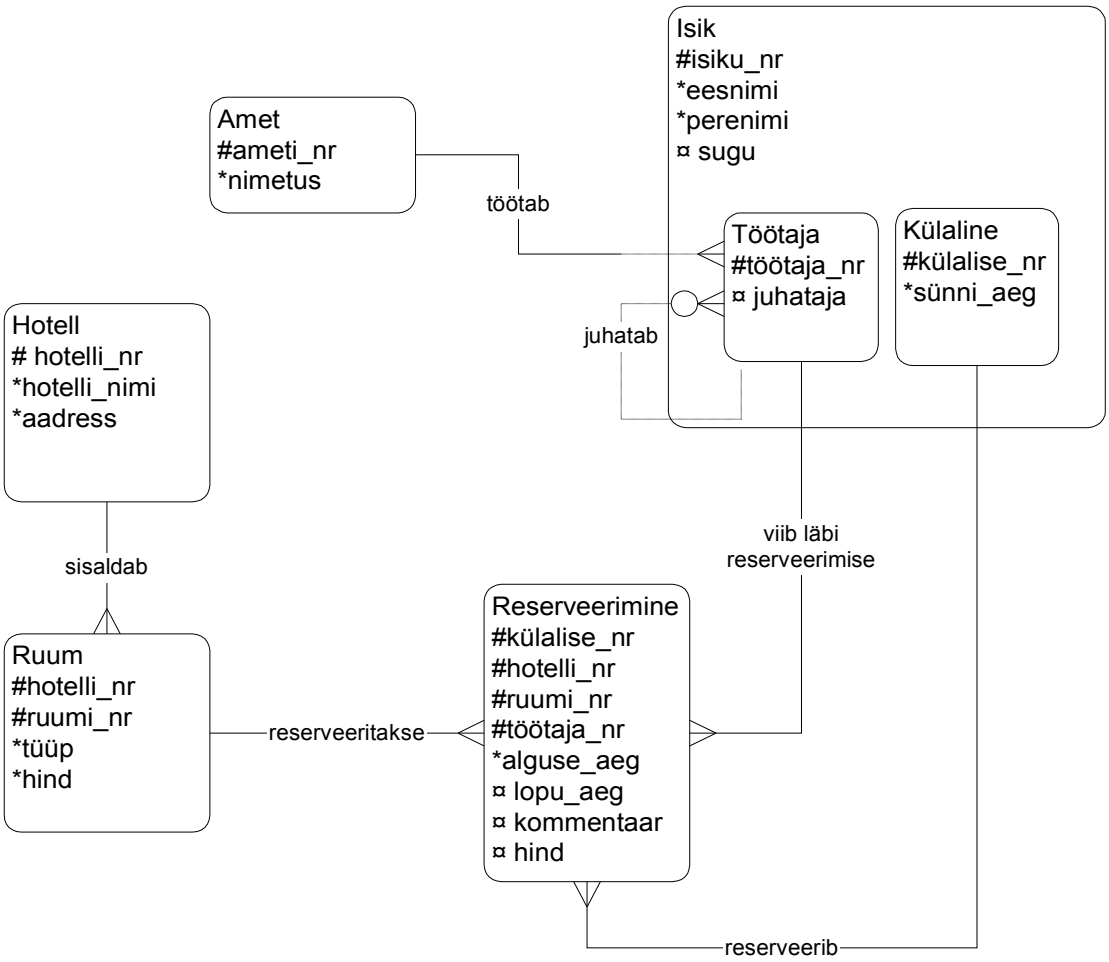
- Chen
- Barker
- IE (Information Engineering)
- IDEF1X (Integration DEFinition for Information Modeling)
- UML (Unified Modeling Language)

Chen



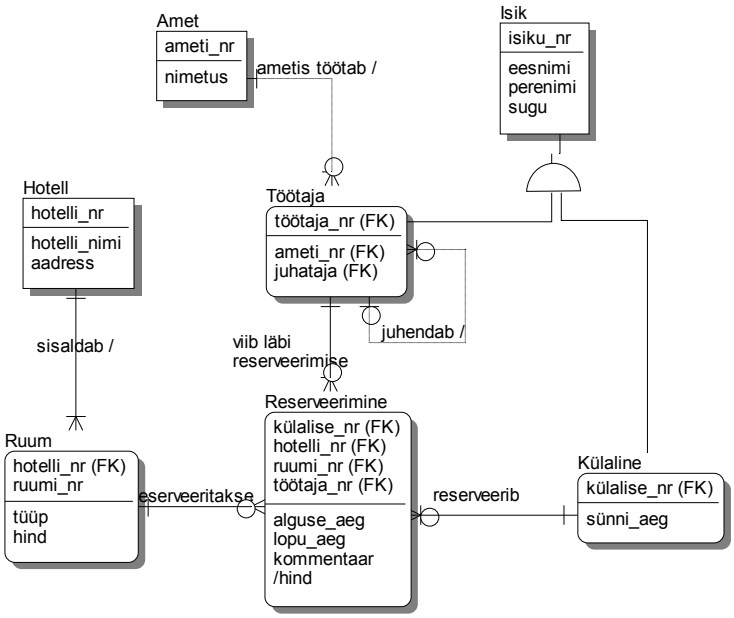
Joonis 34 Olemi-suhte diagrammi esitus Cheni notatsioonis.

Barker



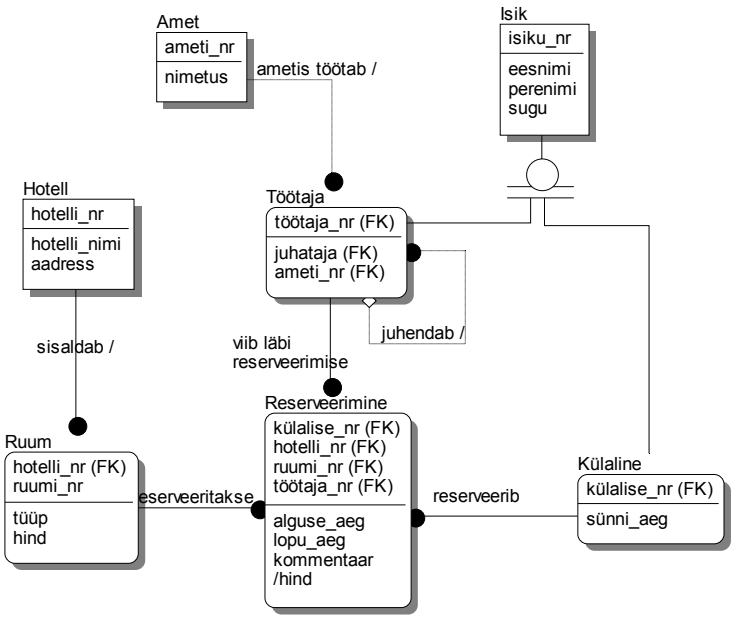
Joonis 35 Olemi-suhte diagrammi esitus Barkeri notatsioonis.

IE (Information Engineering)

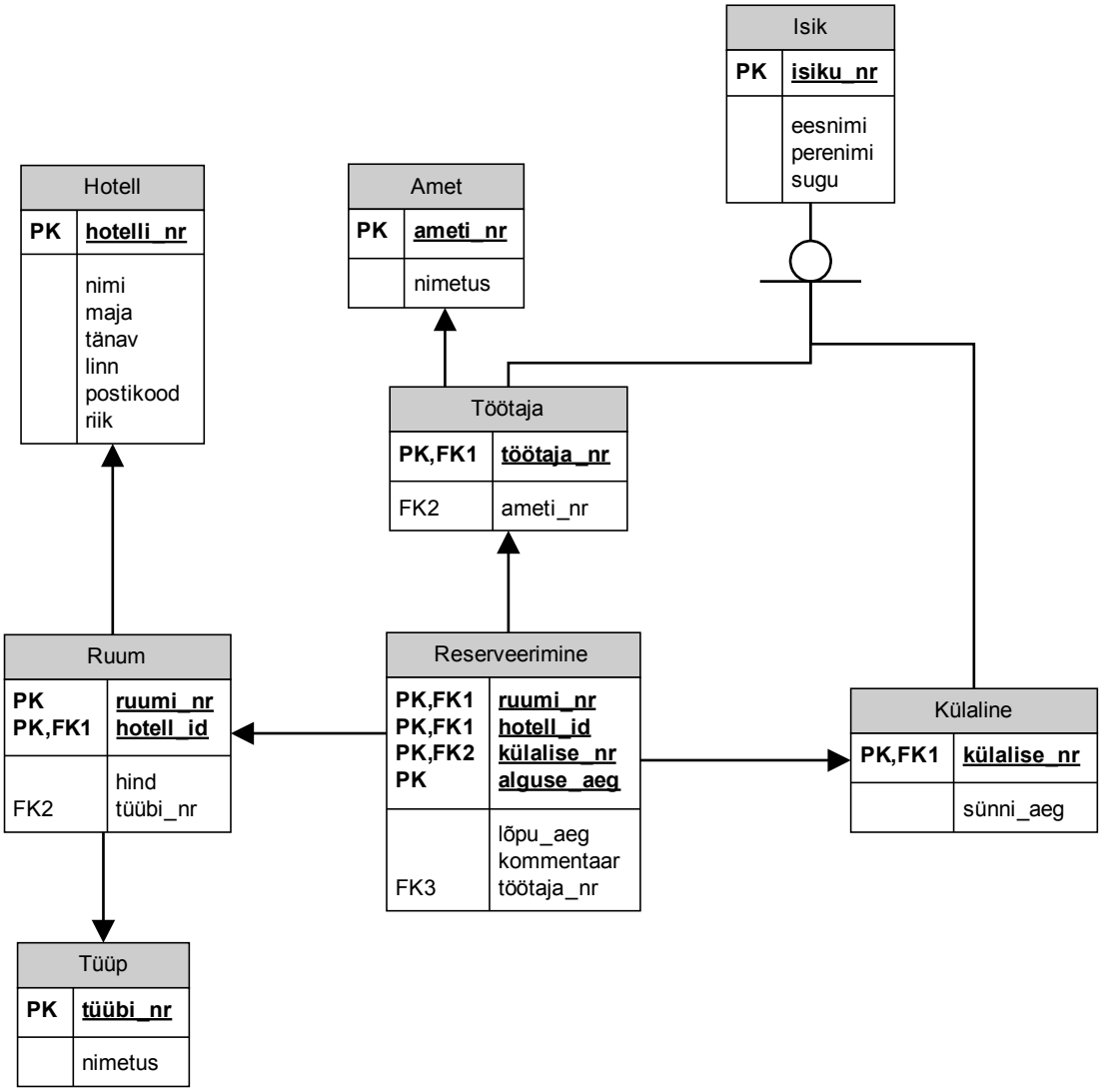


Joonis 36 Olemi-suhte diagrammi esitus IE notatsioonis.

IDEF1X (Integration DEFinition for Information Modeling)

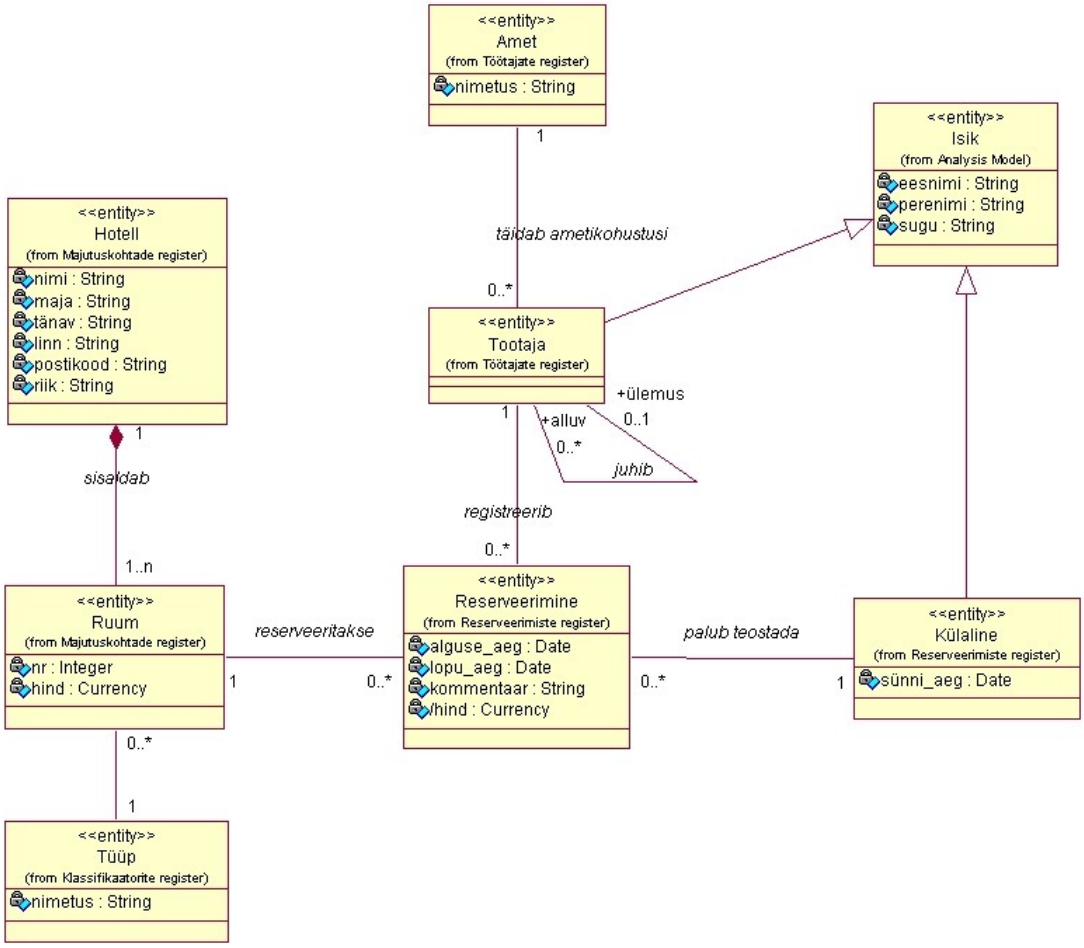


Joonis 37 Olemi-suhte diagrammi esitus IDEF1X notatsioonis.



Joonis 38 Olemi-suhte diagrammi esitus IDEF1X notatsioonis.

UML (Unified Modeling Language)



Joonis 39 Olemi-suhte diagrammi esitus UML notatsioonis.

Joonis 39 olev diagramm on joonistatud CASE vahendiga Rational-Rose.

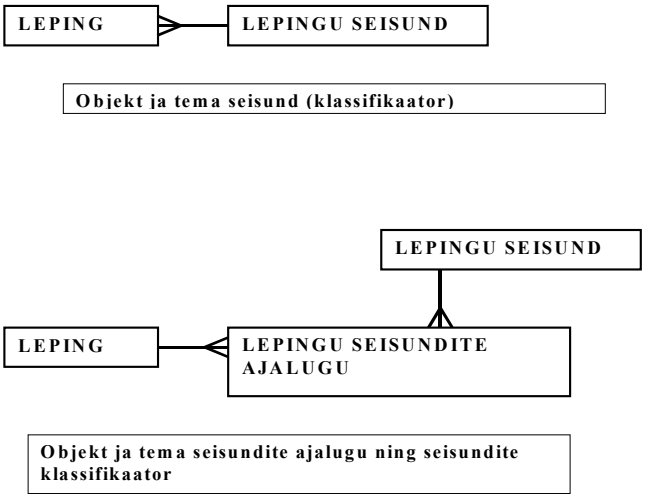
Olemi-suhte diagrammi puudused ja probleemid

- Diagrammi elementide (olemitüüpide, seosetüüpide, atribuut) tähenduses ja definitsioonides pole siiani täpselt kokku lepitud. Mis ühe jaoks on olemitüüp, see on teise jaoks atribuut või seosetüüp.
- Aegade jooksul on kasutusel olnud (ja on siiani) palju erinevaid notatsioone (märgisüsteeme), mida olemi-suhte diagrammide koostamiseks on kasutatud. Erinevates notatsioonides on diagrammidel lubatud erinevat tüüpi elemendid – näiteks IE ja IDEF1X notatsioonis diagrammidel näidatakse välisvõtmeid.
- Pole piisavalt väljendusrikas. Palju informatsiooni (nt. andmetega seotud kitsenduste kohta) on võimalik esitada ainult diagrammiga kaasa tulevate tekstiliste kirjeldustega.

3.8 Küsimused kontseptuaalse andmemudeli kohta

1. Kas teadmine mida mudel loob on süsteemi eesmärkide saavutamiseks piisav?
2. Kas iga olemitüübi/atribuudi/seosetüübi poolt loodav teadmine on süsteemi jaoks oluline?
3. Kas olemitüüpide ja atribuutide semantika dokumendis on kirjeldatud kõigi olemi-suhte diagrammil olevate olemitüüpide ja atribuutide semantika?
4. Kas olemitüüpide ja atribuutide semantika kirjelduses pole olemitüüpe/atribuute, mis pole esitatud olemi-suhte diagrammil?
5. Kas on leitud täpse mudeli jaoks vajalikud klassifikaatorid?
6. Kas põhiobjektiga, millel on võimalikud erinevad seisundid, on seotud olekute klassifikaator?

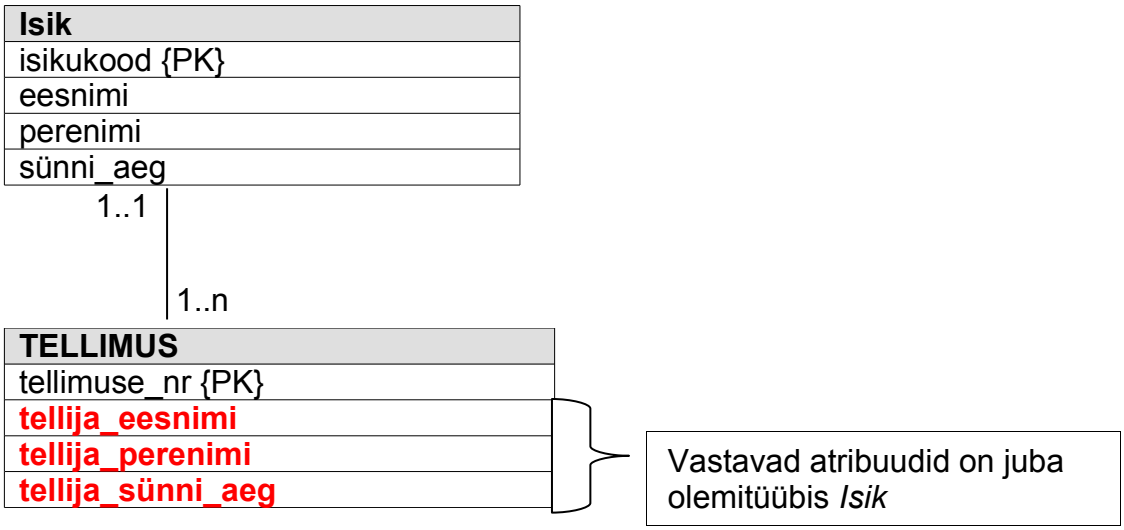
Näide:



Joonis 40Seisundiklassifikaatorite esitamine olemi-suhte mudelis.

7. Kas pole olemitüüpe, mis kirjeldavad sama asja ja tuleks ühendada?
8. Kas pole olemitüüpe, mis tuleks jagada mitmeks erinevaks olemitüübiks?
9. Kas olemitüüpide juures on esitatud kõik teadaolevad atribuudid?
10. Kas ei näidata välisvõtmeid? Välisvõtmeid analüüsi mudelis ei esitata, sest need on *tehniliseks* vahendiks seoste realiseerimisel relatsioonilises andmebaasis.
11. Iga atribuudi puhul küsige endalt:
    - Kas see atribuut on selle olemitüübi atribuut?

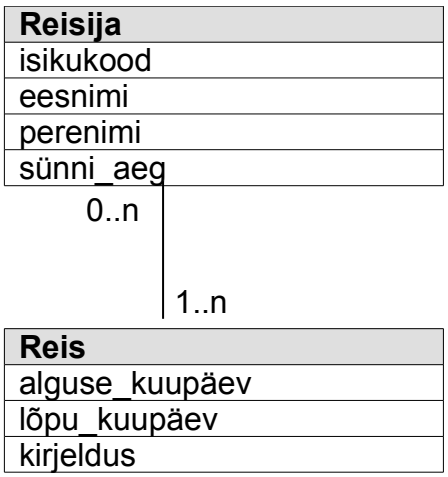
Negatiivne näide:



- Milliseid väärtusi atribuut omab?
- Kas atribuut ei peaks olema iseseisev olemitüüp?

Näide:

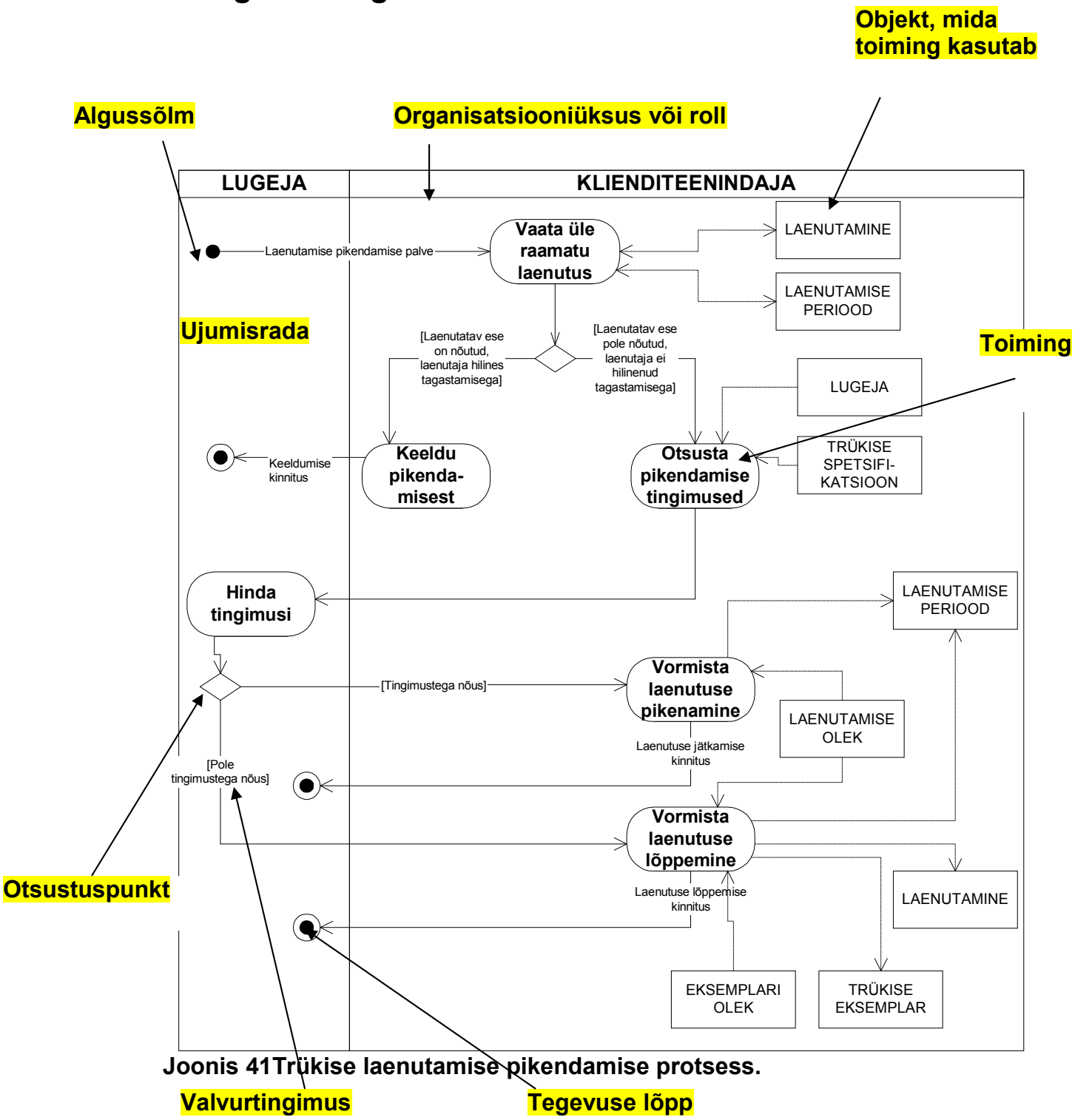
Atribuudi *reisijad* taga peitub tegelikult iseseisev olemitüüp:



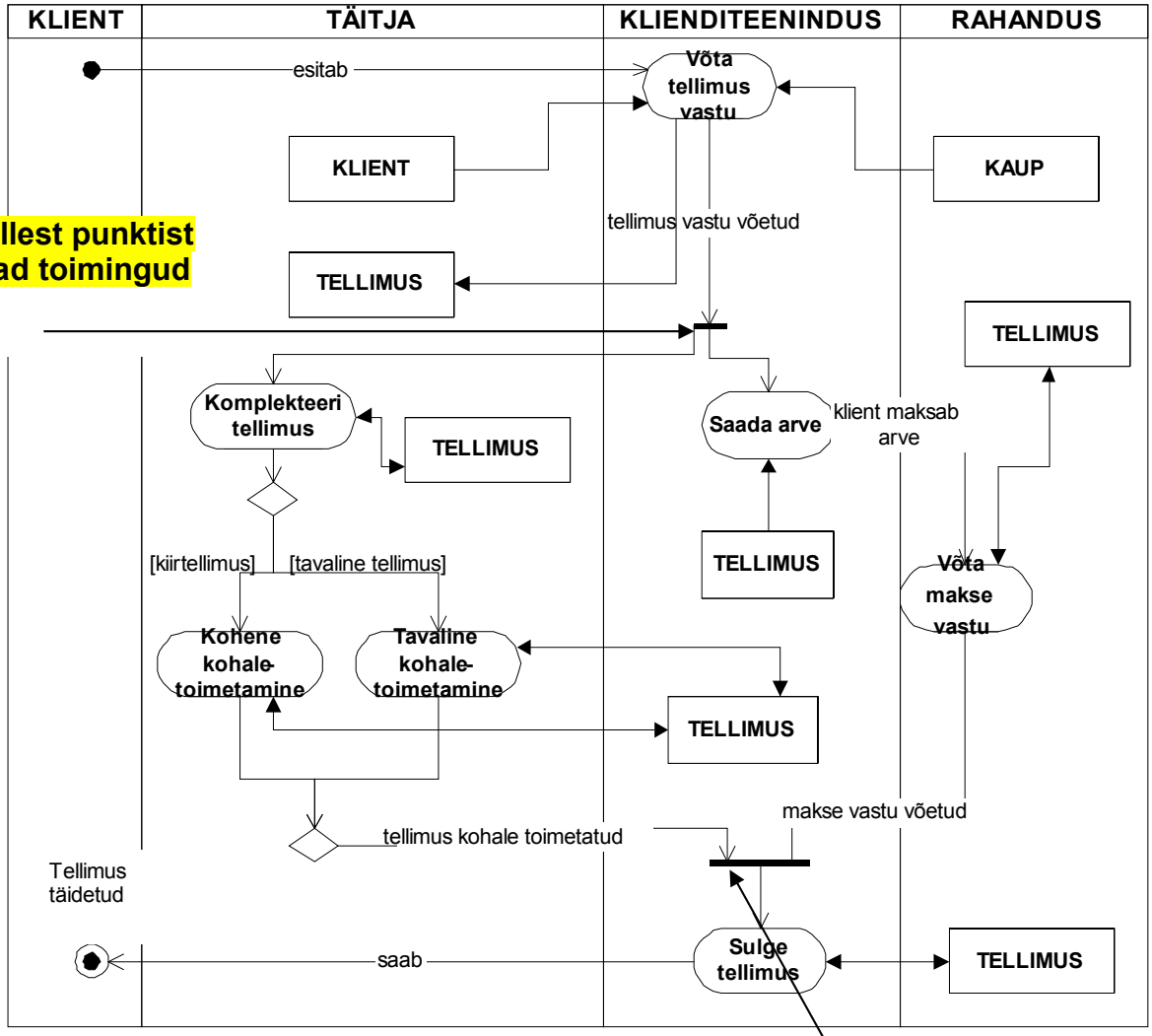
12. Kas tuletatud atribuut on diagrammil selgelt välja toodud?
13. Kas mitmeväärtuseline atribuut on diagrammil selgelt välja toodud?
14. Kas olemitüüpide juures on leitud atribuudid, mille väärtused identifitseerivad olemeid?
15. Kas olemitüüpides ei eksisteeri atribuutide gruppe? (Nt. aadress1, aadress2 ja aadress3 olemitüübis *Isik*. Nende põhjal saab luua uue olemitüübi *Aadress* ja seostada see olemitüübiga *Isik*.)

16. Kas iga seosetüübi (v.a. üldistusseose) puhul on määratud seotud olemitüüpide jaoks järgud ja osaluskohustused?
17. Kas seosetüübile on antud arusaadav ja seost kirjeldav nimi?
18. Kas seosetüübiga koos on esitatud selles osalevate olemitüüpide rollide nimed?
19. Kui kahe olemitüübi vahel on mitu seosetüüpi, kas siis iga seosetüübi otstes on näidatud olemitüübi roll?
20. Kas üldistusseose puhul on määratud osaluskohustus ja kuuluvus?
21. Kas seosetüübid on diagrammil seotud välistava kaarega, seal kus seda on vaja?
22. Kas 1-1 seost on tõepoolest vaja, või võib kaks olemitüüpi ühendada üheks?

4. Tegevusdiagramm



Joonis 41Trükise laenutamise pikendamise protsess.



Hargmik – sellest punktist alates jätkuvad toimingud paralleelselt

Liitumispunkt, paralleelsete toimingute lõpp

Joonis 42Kauba tellimise protsess.

Tegevusdiagramm on mõeldud protsesside modelleerimiseks.

Mis on äriprotsess? (Marshall, 1999) esitab erinevaid definitsioone.
Object Management Group: Äriprotsess esitab organisatsioonis toimuvat töö- ja informatsioonivoogu. Äriprotsessid suhtlevad äriobjektidega, et tagada organisatsiooni toimimine. Äriprotsess võib olla pikaajaline, nagu tellimuse elutsükkel või lühiajaline, nagu aasta lõpus aruande koostamine.

ISO-9000: Iga organisatsioon eksisteerimise põhjus on lisandväärtuse tekitamine. Lisaväärtuse saavutamiseks kasutatakse protsesside võrgustikku. Igal protsessil on sisendid ja protsessi tulemusena saadavad väljundid. Protsesside võrgustiku struktuur pole tavaliselt lihtne jadastruktuur, vaid see võib olla üsna keerukas.

CMU Software Engineering Institute: Protsess on hulk järjestatud samme, mida tehakse mingi eesmärgi saavutamiseks. Protsessi saab tükeldada (dekomponeerida) väiksemateks sammudeks kuni jõutakse elementaartoiminguteni.

Fowler (2007) märgib, et rangelt võttes on tegevus toimingute jada. Tegevusdiagramm kirjeldab toimingute jada ning võimaldab näidata nii tingimuslikku kui paralleelset käitumist. Toimingud on omavahel seotud voogude e. kaarte abil. Kaarele võib soovi korral anda ka nime, kuid enamasti piisab lihtsalt noolest (Fowler, 2007).

"Äriprotsessi kontekstis on toiming diskreetne samm, mis võib olla seotud teiste toimingutega sõltuvuste võrgu kaudu." (Marshall, 1999)

Tegevusdiagrammil on võimalik näidata otsustuspunkte. Otsustuspunkti siseneb üks voog ja otsustuspunktist väljub mitu valvurtingimustega (lisatingimusega) voogu. Vastavalt valvurtingimusele valitakse toiming, mida järgmisena täita. Igas otsustuspunktis saab valida vaid ühe väljuva voo – seega peavad valvurtingimused olema üksteist vastastiku välistavad. Valvurtingimus on nurksulgudesse pandud loogikaavaldis. Valvurtingimus [else] tähistab voogu, mis tuleb valida, kui otsustuspunktis kõik muud valvurtingimused on väärad.

Tegevusdiagrammil on võimalik näidata paralleelseid toiminguid. Paralleelsete toimingute kirjeldamiseks kasutatakse tegevusdiagrammil sümboleid "Hargmik" ja "Liitumispunkt". "Hargmiku" ja "Liitumispunkti" vahel esitatavad toimingud toimuvad paralleelselt. "Liitumispunkt" näitab, et kõik eelnevad toimingud peavad olema lõppenud, et toimingud saaks jätkuda. Viimase näite puhul toimuvad paralleelselt näiteks arve saatmine ja tellimuse komplekteerimine.

Tegevusdiagramm on organiseeritud vastavalt rollidele. "Ujumisrajad" e. "rajad" e. "sektsioonid" eraldavad erinevate organisatsiooniüksuste ja rollide poolt läbiviidud toiminguid.

Toimingud kasutavad oma tööks andmeid. Tegevusdiagrammil võib esitada ka objekte, mille kohta käivad andmed on toimingule sisendiks või väljundiks. Objekti võib diagrammil esitada mitu korda. Iga objekti ilmumine esitab erinevat etappi selle elutsüklist. Iga kord võib objekti tähistada viitega selle seisundile konkreetsel ajahetkel.

Äriprotsess võib olla liiga keerukas, et seda esitada ühel diagrammil. Sellisel juhul tuleb see esitada mitme lihtsama diagrammi abil. Mitu protsessi toimingut asendatakse ühe alamprotsessiga, mis kirjutatakse lahti eraldi diagrammil.

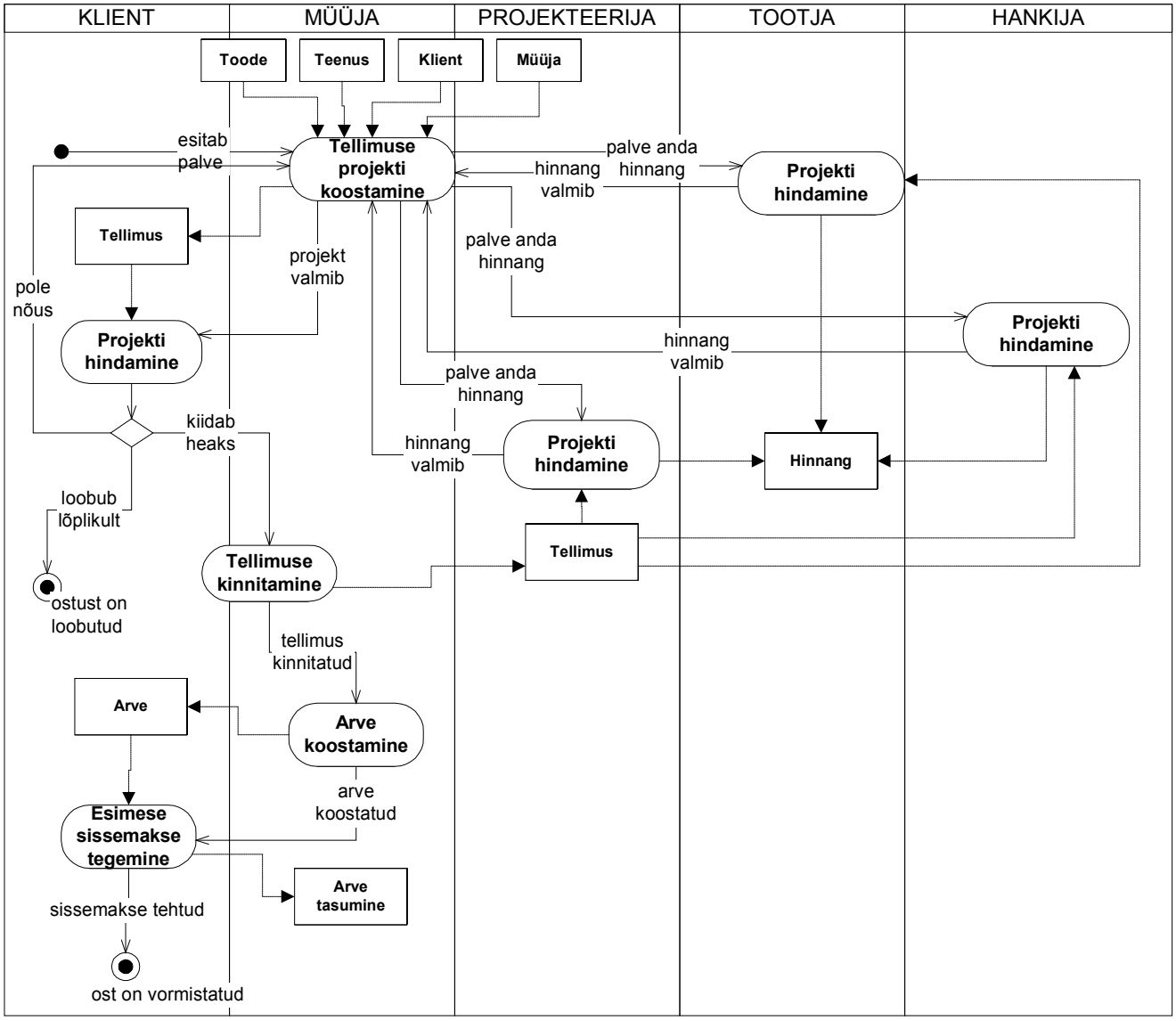
Milleks kasutada protsesside tükeldamist?
- Võimaldab saada ülevaate suurtest protsessidest.
- Võimaldab esitada ülevaatlikke protsessimudeleid.
- Aitab peita harva toimuvaid, erandlikke töövooge.

Tegevusdiagrammi eelised.
- Üldiselt mõistetavad ka mitte-IT inimesele.
- Saab näidata paralleelset käitumist.
- Igale tegevusele saab lisada viite selle eest vastutavale rollile (ujumisrajad).

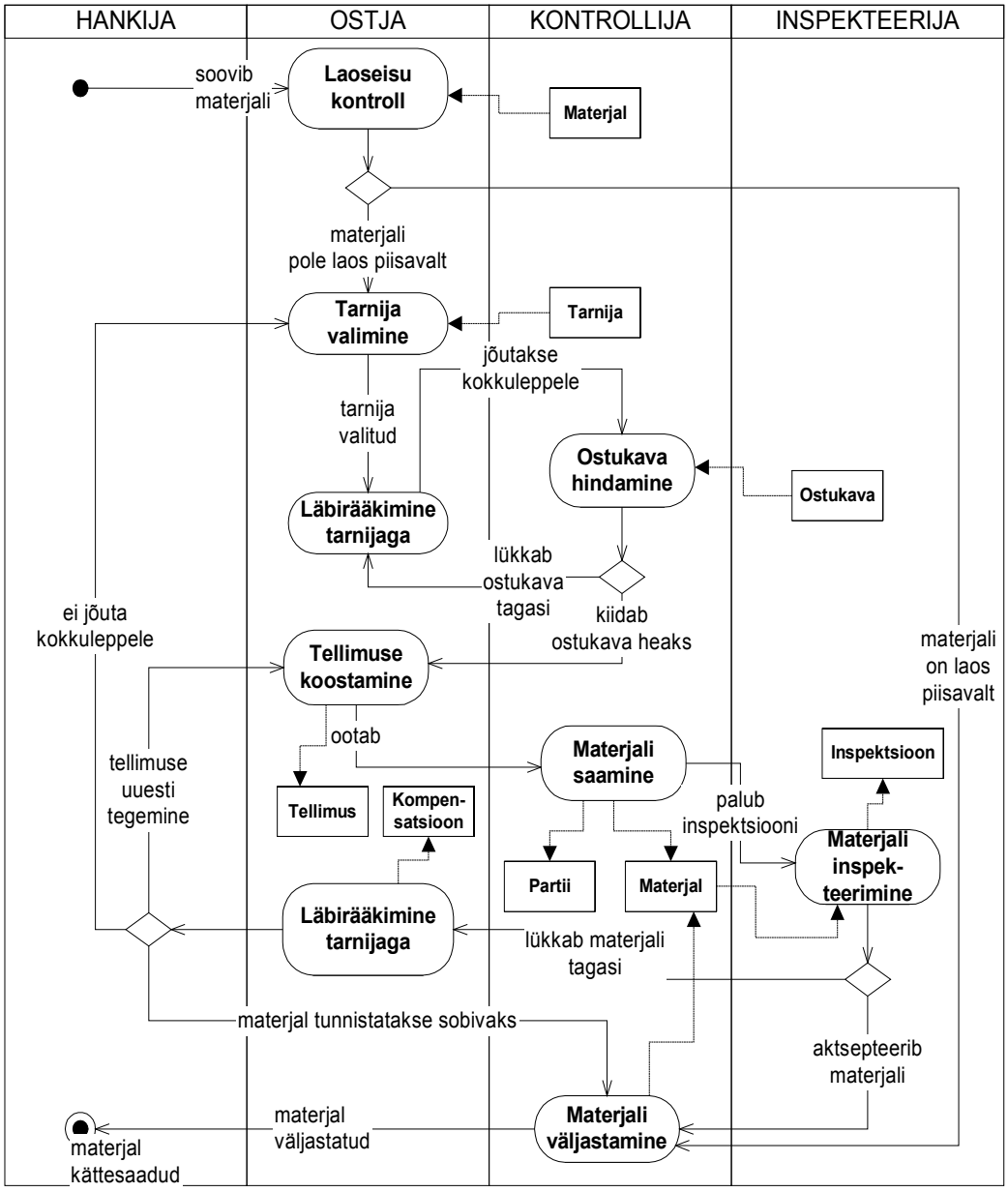
Olukorrad, kus on sobilik kasutada tegevusdiagrammi.
- Töövoo mõistmine organisatsiooni uurimise käigus. Annab üldise ülevaate organisatsioonis toimuvatest tööprotsessidest. Kasutatakse äriprotsesside ümbertöötlemiseks.
- Kasutusjuhu analüüsimine. Iga kasutusjuht on kirjeldatav stsenaariumitega. Need aitavad hinnata alternatiivseid protsessi läbimise viise.
- Kirjeldada keerukat algoritmi, kus on palju paralleelseid tegevusi.
- Kasutajaliideses navigeerimise kirjeldamine (vt. http://www-128.ibm.com/developerworks/rational/library/4697.html)

Ära kasuta tegevusdiagrammi:
- Objektide suhtlemise näitamiseks.
- Kirjeldamaks objektide käitumist oma eluea jooksul.

Järgnevalt on toodud veel tegevusdiagrammide näiteid:



Joonis 43Müügiprotsessi tegevusdiagramm.



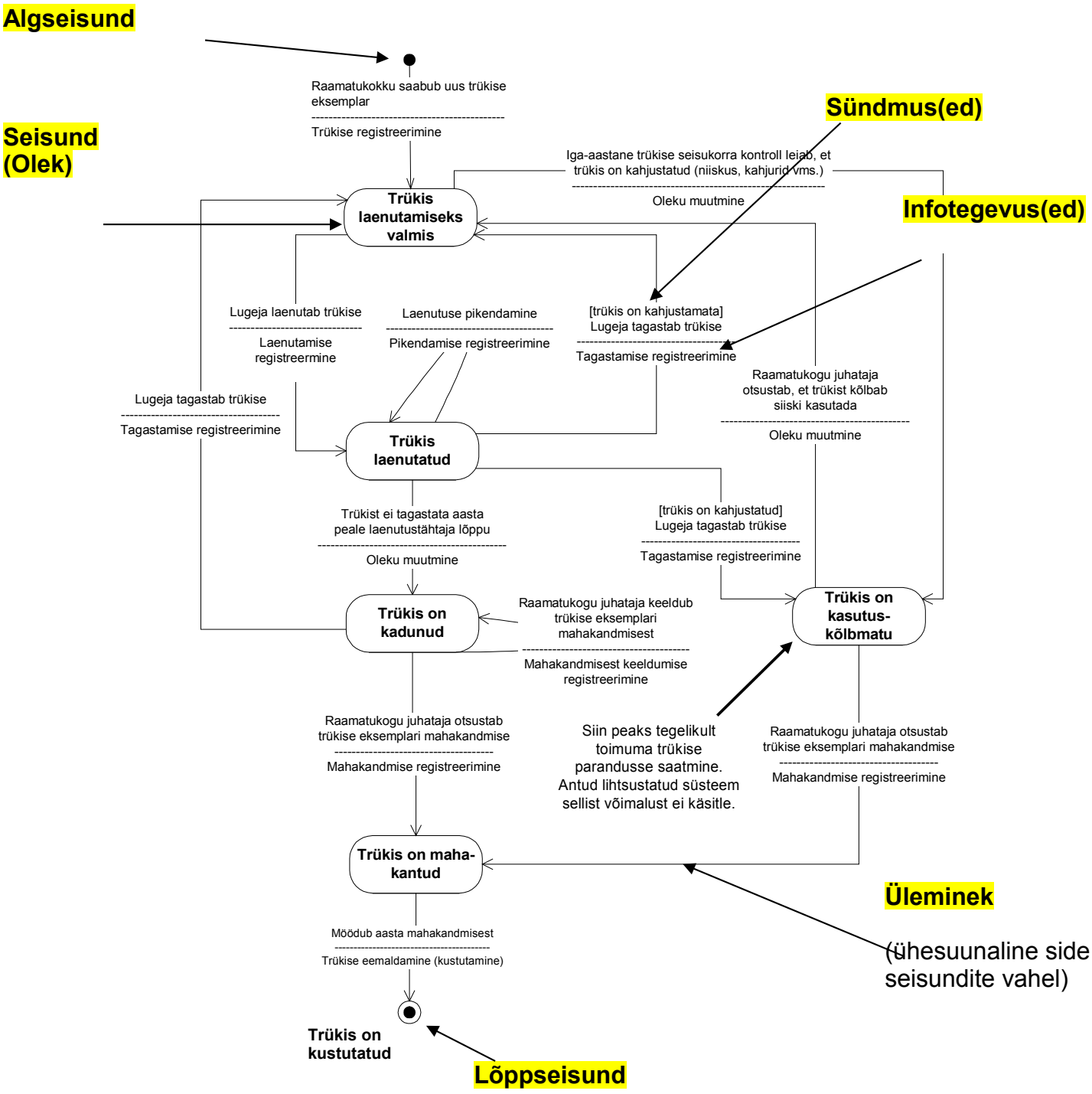
Joonis 44Materjalide hankimise protsessi tegevusdiagramm.

4.1 Küsimustik tegevusdiagrammi kohta

1. Kas tegevusdiagrammil on näidatud algsõlm ja tegevuse lõpp?
2. Kas algsõlmest on võimalik jõuda iga toiminguni?
3. Kas "ujumisrajad" eraldavad erinevate organisatsiooniüksuste ja rollide poolt läbiviidud toiminguid?
4. Kas tegevusdiagrammil on iga toiminguga seostatud andmed, mida see toiming kasutab?
5. Kas otsustuspunktist väljuvate voogude juures on näidatud ka valvurtingimused?
6. Kas diagrammis on kasutatud korrektset tähistust?
7. Kas diagrammid on loetavad ja tehniliselt korrektsed?

5. Seisundidiagramm (olekudiagramm)

Eesti keeles võib kasutada nii mõisteid “seisund” (“seisundidiagramm”, “algseisund”, “seisundimuutus” jne.) kui ka “olek” (“olekudiagramm”, “algolek”, “olekumuutus” jne.). Need on samaväärsed.



Joonis 45Trükise eksemplari (koopia) seisundidiagramm.

Seisundidiagrammi eesmärgid

- Selgitada süsteemis asetleidvad sündmused, mis põhjustavad seisundite muutumist ja mille korral muudetakse atribuutide väärtusi.
- Kirjeldada objektide käitumist mingis seisundis.
- Määratleda süsteemi objektide seisundite muutumise juhtumid.
- Uurida protsesside ja andmete muutmise dünaamilist seost.
- Leida protsesse käivitavad või tingivad juhtumid – süsteemi reaktsioonid.

Seisund on objekti poolt teostatud eelnevate tegevuste tulemus, mis on määratud selle atribuutide ja seoste (lingid teiste objektidega) väärtustega. Ajal, kui objekt on mingis seisundis võivad objektiga teatud asjad juhtuda ja teatud asjad mitte juhtuda. Näiteks kui patsient on seisundis "raskelt haige", siis ei lubata teda haiglast lahkuda. Patsient saab haiglast lahkuda alles siis, kui ta jõuab seisundisse "stabiliseerunud" või "tervenenud".

Seisundidiagramm esitab objekti elutsüklit. See diagramm esitab objekti lubatud seisundeid, sündmuseid, mis on selle objekti jaoks olulised ja lubatud üleminekuid seisundite vahel. See diagramm näitab kõiki võimalikke viise, kuidas objekt reageerib sündmustele.

"Sündmus on ajahetke abstraktsioon. See on väliste objektide ja infosüsteemi (või allsüsteemide) vahel liikuvate infovoogude kokkupuute-punkt, mis nõuab süsteemilt teatud reaktsioone, teiste sõnadega, sündmus tingib või käivitab arvutikasutuse." (Mikli, 1999)

- Väline sündmus (ka süsteemi sündmus) on tingitud millestki või kellestki väljaspool süsteemi piiri.
- Ajast tingitud sündmused. Seda tüüpi sündmus tekib näiteks mingi ajahetke saabumise või mingi ajavahemiku möödumise tõttu.

Üleminek esitab reaktsiooni sündmusele. Kui toimub sündmus, siis minnakse üle ühest seisundist teise. Seisund võib jääda ka samaks. Ülemineku juures näidatakse sündmus(ed), mis selle ülemineku käivitas. Ülemineku juures võib näidata ka (info)tegevus(ed), mis on sellest sündmusest tingitud.

Sündmuseid tegevusi võib diagrammil esitada:

sündmus(ed) / infotegevus(ed)

sündmus(ed)
---
infotegevus(ed)

Üleminekut käivitava sündmuse juures võib olla määratud ka **valvurtingimus**. Kui toimub sündmus ja valvurtingimus on täidetud, siis võib üleminek toimuda.

Algseisund esitab objekti sünnipunkti. Üleminek näitab objekti minekut seisundisse, mille see saab oma loomisel. Objektil võib olla mitu erinevat

algseisundit – sõltuvalt sündmusest, mis põhjustavad objekti loomise. Näiteks teenuse osutamisele võib kuid ei pruugi isik ette registreeruda.

Lõppseisund täitab seda, et objekt lõpetab eksistentsi. Objektil võib olla mitu erinevat lõppseisundit, mis on seotud erinevate sündmustega, mis objekti lõppseisundisse viivad.

Objekti seisundi võib määrata objekti atribuutide ja seoste kaudu.

Näiteks kui haige kehatemperatuur on üle 39 kraadi Celsiuse järgi ja temaga teostatakse teatud raviprotseduure, siis ta on "tõsiselt haige". Samas on nende seisundite teadmine oluline, sest erinevates seisundites reageerib objekt sündmustele erinevalt. Samuti on lubatud üleminekud ainult teatud seisundite vahel.

Seega sageli on objekti seisundi määramine piisavalt oluline selleks, et modelleerida seda eraldi atribuudina. Seisundite kirjeldamiseks on atribuudid, mida tuntakse "seisundimuutujatena". Objektil võib olla mitu seisundit kirjeldavat atribuuti. Näiteks auto puhul auto korrasoleku ja auto kindlustatuse kirjeldamiseks. Selleks võib andmemudelil luua seisundite klassifikaatori.

Keerukamatel seisundidiagrammidel võib esitada alamseisundeid.

Alamseisundid *pärivad* kõik ülemseisundite üleminekud. Näiteks kui telefon on mistahes alamseisundi (nt. tooni mängimine, numbri valimine), siis kui telefoni toru pannakse hargile, toimub üleminek seisundisse mitteaktiivne.

"Kui objekt reageerib sündmusele alati samal viisil siis teda loetakse **seisundist sõltumatuks** selle sündmuse suhtes." (Larman, 1997)

Kui objekt reageerib sündmustele erinevalt, sõltuvalt oma seisundist, siis teda loetakse **seisundist sõltuvaks.**

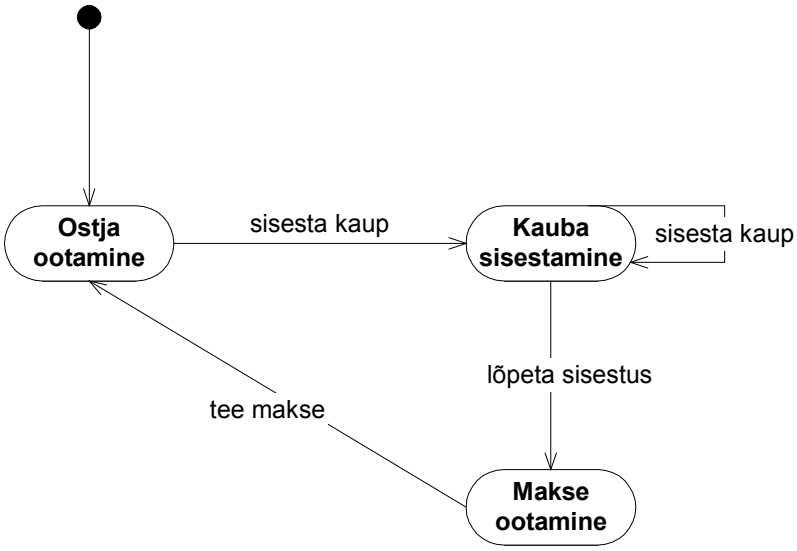
Loo seisundidiagramm objektide kohta, mis on seisundist sõltuvad ja millel on keerukas käitumine.

Sageli on selliste objektidega seotud mingi kindlaksmääratud sündmuste jada.

Kasutusjuhtude seisundidiagramm

Kirjeldab ühe kasutusjuhu käigus lubatud väliste sündmuste järjekorda.

Järgnev seisundidiagramm kirjeldab kasutusjuhus "Kauba ostmine" ilmneda võivate sündmuste järjekorda.



Joonis 46Kasutusjuhu "Kauba ostmine" seisundidiagramm.

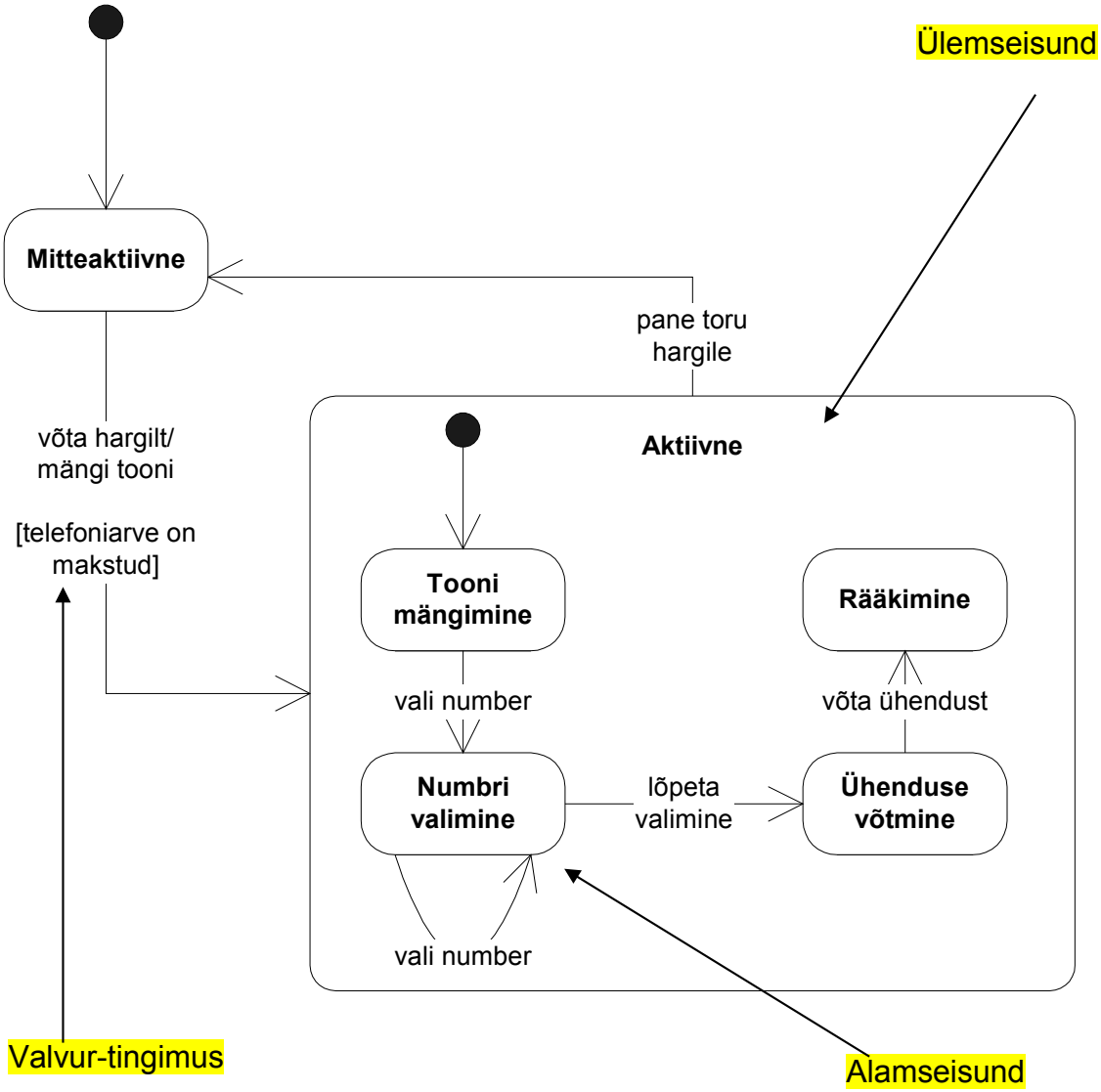
Näiteks ei ole makse tegemine lubatud enne, kui kõik ostetavad kaubad on sisestatud.

Süsteemi seisundidiagramm

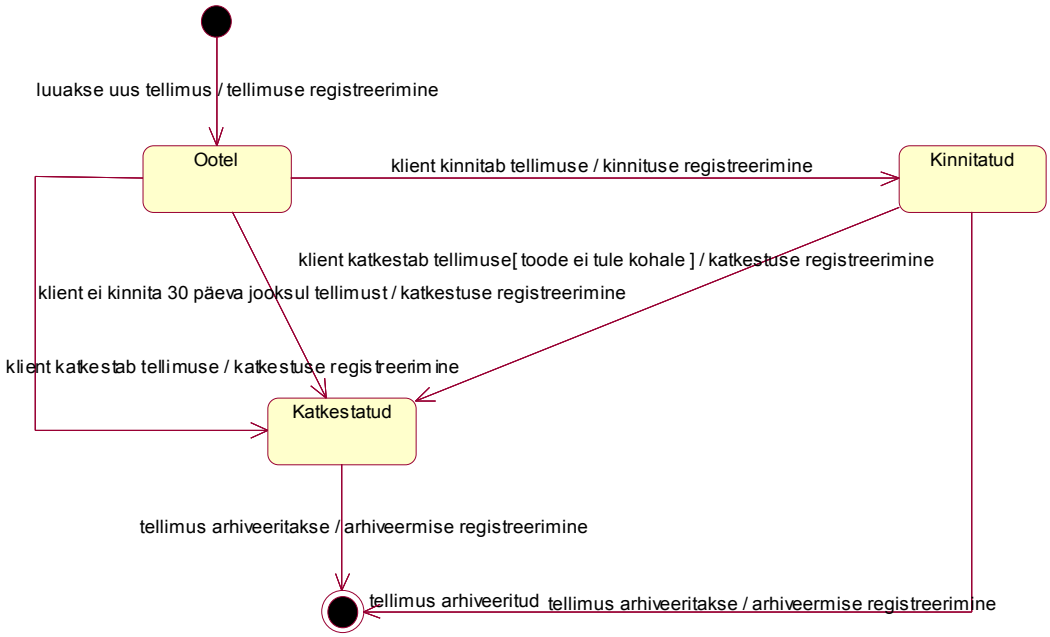
Kasutusjuhtude seisundidiagrammi erijuhtum, mis kirjeldab kogu terviksüsteemi jaoks kõikvõimalikke sündmuseid ja seisundite üleminekuid. See on ühend kõigist kasutusjuhtude seisundidiagrammidest. Selleks, et mudel oleks ülevaatlik, võib seda kasutada kui sündmuste hulk on piisavalt väike.

Objektide seisundidiagrammid

Seisundidiagramme võib koostada kõigi objektide kohta. Seisundidiagramme on mõtet koostada objektide kohta, mis omavad keerukat seisundist sõltuvat käitumist.



Joonis 47"Telefon" seisundidiagramm kus on näidatud ka alamseisundid (Larman, 1997).



Joonis 48Objekti "Tellimus" seisundidiagramm. Joonistatud CASE vahendi Rational Rose abil .

5.1 Küsimustik objekti seisundidiagrammi kohta

1. Kas objekt, mille kohta seisundidiagramm käib, on seisunditest sõltuv s.t. kas reaktsioon teatud sündmus(t)ele sõltub objekti seisundist?
2. Kas seisundidiagramm kirjeldab kõiki võimalikke objekti elutsükleid (sündmuste-seisundite jadasid)?
3. Kas seisundidiagramm välistab kõik võimatud objekti elutsüklid?
4. Kas kõikidesse seisunditesse on võimalik algseisundist jõuda?
5. Kas diagrammil kasutatud tähistus on korrektne?

6. CRUD maatriks

Erinevat tüüpi mudelid esitavad erinevaid vaateid süsteemile. Mudelid peavad olema omavahel kooskõlas, et anda süsteemist korrektne ülevaade ning võimaldada luua korrektne süsteemi realisatsioon.

Andmevaadet ja protsessivaadet esitavate mudelite omavahelise kooskõla kontrollimiseks võib kasutada CRUD maatriksit. Maatriksi nimi on akronüüm, mis tuleneb sõnadest "Create", "Read", "Update", "Delete". (Loo, Loe, Muuda, Kustuta). Akronüüm on mitme sõna algustähtedest moodustatud lühinimetus. Eeldame, et andmevaate esitamiseks kasutatav kontseptuaalne andmemudel sisaldab olemi-suhte diagramme ja protsessivaate kirjeldamiseks kasutatakse kasutusjuhtude mudelit. Sellisel juhul peab iga atribuudi/seosetüübi kohta leiduma kasutusjuht, kus:

- ... on kirjeldatud toiming, mille käigus atribuut väärtustatakse/seos tekitatakse (**C**reate).
- ... on kirjeldatud toiming, mille käigus atribuudi väärtust/seost loetakse (**R**ead).

Näiteks, kui andmeid ainult *luuakse,* kuid ühegi kasutusjuhu käigus ei *loeta*, siis tekib küsimus, miks on vaja koguda andmeid mida keegi ei kasuta. Kui leidub kasutusjuht, mille käigus andmeid loetakse, kuid ühegi kasutusjuhu käigus neid andmeid ei looda, siis on selge, et süsteemi kirjeldus ei ole täielik.

Lisaks võib leiduda kasutusjuht, kus:

- ... on kirjeldatud toiming, mille käigus atribuudi väärtust/seost muudetakse (**U**pdate).
- ... on kirjeldatud toiming, mille käigus atribuudi väärtus/seos kustutatakse (**D**elete).

Kustutada tuleb andmed, mille kustutamist nõuab seadus, määrus või organisatsioonisisene praktika. Andmete kustutamist võib olla vaja sisestusvigade parandamiseks. Andmete kustutamist tuleb alati põhjalikult kaaluda, sest kustutatud andmeid võib tulevikus näiteks analüüsimise otstarbel vaja minna. Infosüsteemi võib luua nii, et enne operatiivandmete andmebaasist kustutamist laaditakse andmed andmeaita või andmevakka, et neid oleks võimalik hiljem analüüside läbiviimiseks kasutada.

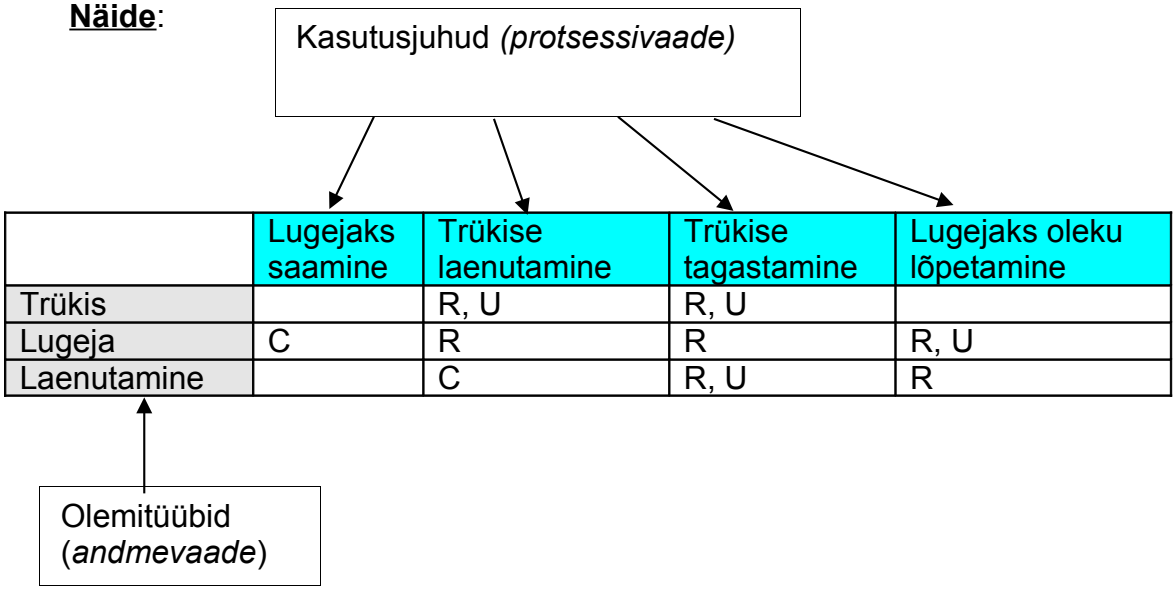
On võimalik, et isegi kui kasutajaliideses jäetakse kasutajale mulje, et ta andis kustutamise korralduse, siis tegelikult muudetakse kustutatava andmeobjekti seisundit ja see säilib andmebaasis. Kasutaja neid andmeid enam ei näe.

CRUD maatriksi võib luua erineva täpsusega.

- **Andmevaate** võib maatriksis esitada näiteks registri, olemitüübi või atribuudi/seosetüübi täpsusega. Järgnevalt tuuakse näide olemitüübi täpsusega maatriksist.

- **Protsessivaate** võib esitada näiteks funktsionaalse allsüsteemi, kasutusjuhu või toimingu täpsusega. Järgnevalt tuuakse näide kasutusjuhu täpsusega maatriksist.

Näide:



	Lugejaks saamine	Trükise laenutamine	Trükise tagastamine	Lugejaks oleku lõpetamine
Trükis		R, U	R, U	
Lugeja	C	R	R	R, U
Laenutamine		C	R, U	R

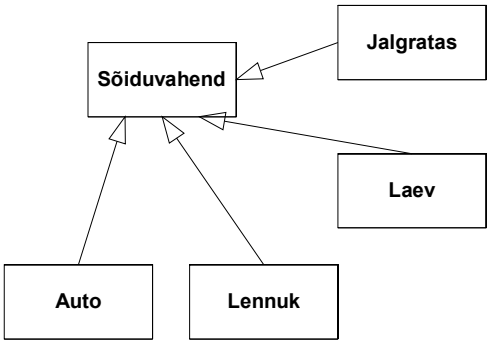
Joonis 49 CRUD maatriksi näide.

- Lugejaks saamise käigus registreeritakse andmebaasis uus lugeja (CREATE).
- Trükise laenutamise käigus loetakse (READ) trükise ja lugeja andmed. Seejärel registreeritakse andmebaasis uus laenutamine (CREATE). Lõpuks muudetakse trükise seisundit (UPDATE), näitamaks selle väljalaenutamist.
- Trükise tagastamise käigus loetakse (READ) kontrolli eesmärgil trükise, lugeja ja laenutamise andmed. Registreeritakse laenutamise tegelik lõppemise aeg (UPDATE). Lõpuks muudetakse trükise seisundit (UPDATE) näitamaks, et trükist saab uuesti laenutada.
- Lugejaks oleku lõpetamise korral loetakse (READ) laenutamise andmeid, et teha kindlaks, ega ei ole veel lõppemata laenutusi. Seejärel loetakse lugeja andmed ning muudetakse lugeja seisundid (seisund:="arhiveeritud") (UPDATE).

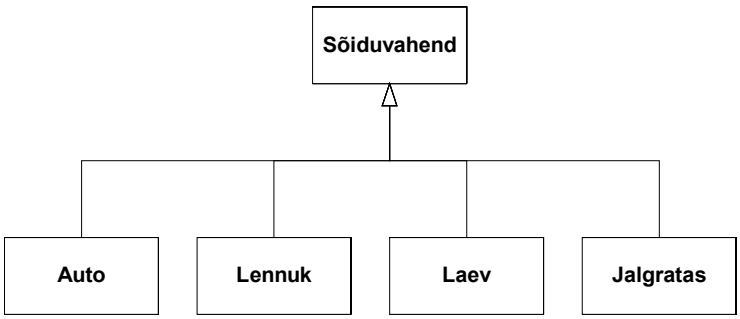
7. Mudelite loetavus

Mudelid peavad olema nende kasutajatele arusaadavad ja hästi loetavad. Seetõttu tuleks järgida soovitusi mudelite kujundamise kohta. Soovituste allikateks on (Evitts, 2000) ja (Hoberman, 2002).

1. **Esitage kontseptuaalmudelis või olemi-suhte diagrammil üldistusseos puukujulisena. Esitage üldisem mõiste kõrgemal kui spetsiifilisemad mõisted.**

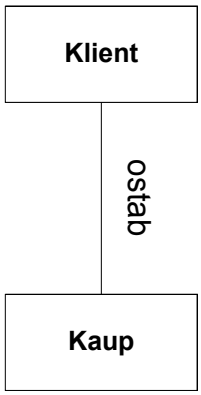


Joonis 50 Halb näide üldistusseose joonistamise kohta.



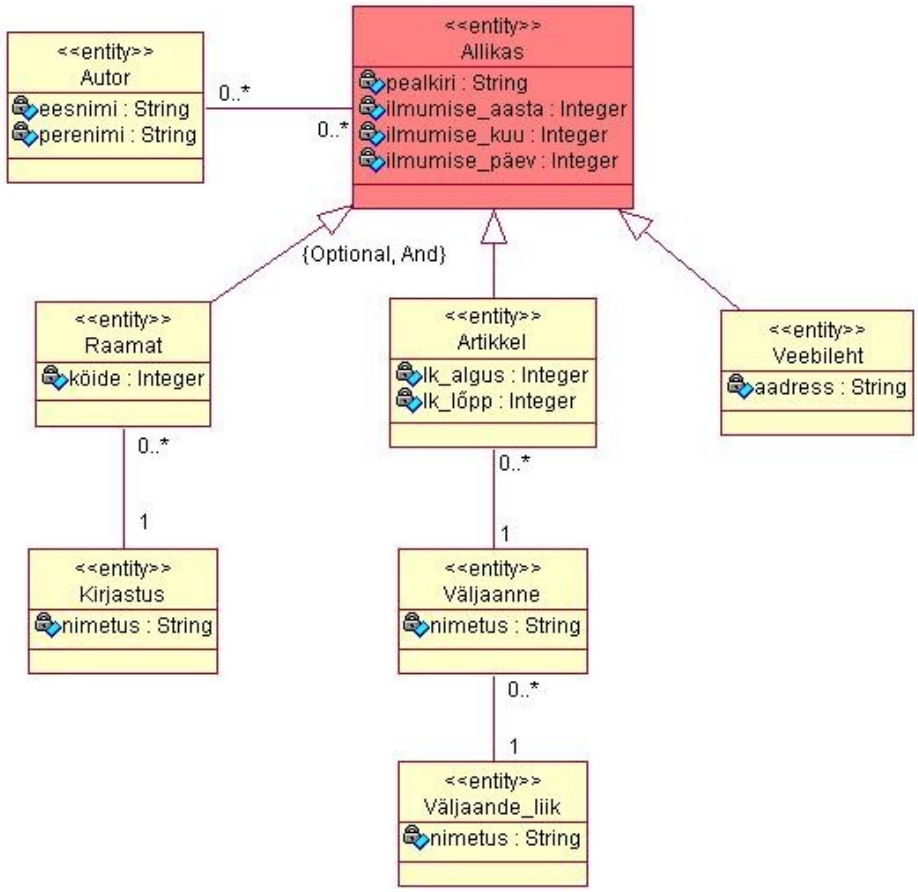
Joonis 51 Hea näide üldistusseose joonistamise kohta.

2. **Kui mudelil on palju jooni, millele tuleb midagi kirjutada, siis ruumi kokkuhoiu mõttes paigutage kirjutis pikki joont.**



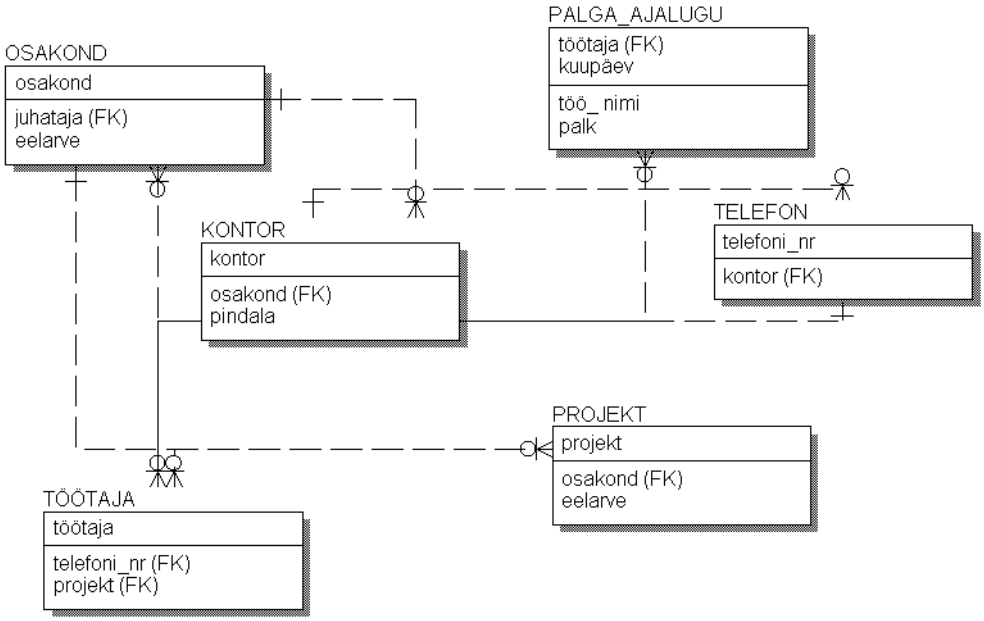
Joonis 52Joonele kirjutatava teksti paigutamine pikki joont .

3. **Tõstke diagrammil esile olulised elemendid kasutades rõhutatud jooni, fonte, varjutatud alasid või värve. Samuti võib näiteks esiletõstmist vajavad mudeli elemendid joonistada suuremalt kui teised.**



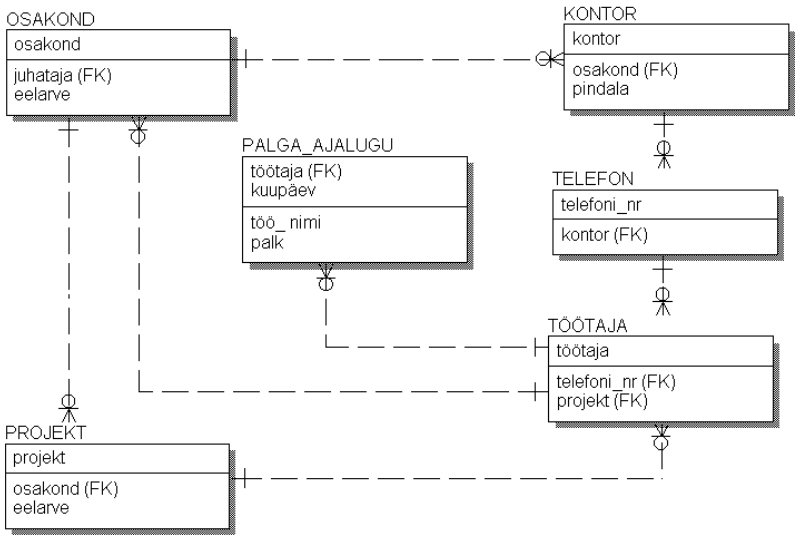
Joonis 53Allikate registri olemi-suhte diagramm.

4. **Vältige ristuvaid ja liiga pikki jooni. Vältige joontes liigseid paindeid ja sikk-sakke. Mitte kunagi ei tohiks jooned jääda teiste mudeli elementide alla.**



Võiks isegi öelda, et modelleerija, kes esitab oma töö lõpptulemusena taolise mudeli, ei austa oma töö tarbijat ja lugejat.

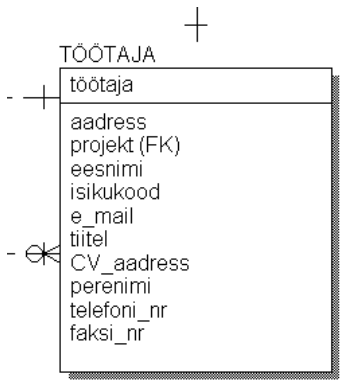
Joonis 54Halvasti paigutatud andmemudeli näide.



Joonis 55Hästi paigutatud andmemudeli näide.

5. Hoberman (2002) kirjeldab, millises järjekorras tuleks loogilise disaini andmebaasi diagrammil ja tabelite kirjeldustes esitada veergude nimed.

Järgnevalt on toodud halb näide veerunimede järjekorrast:



Joonis 56Halb näide veerunimede järjekorrast.

Järgnevalt esitatakse soovitav veerunimede järjekord ning nimetatakse reeglid, mida veerunimede järjekorra määramisel peaks kasutama.

1. ***Primaarvõtmesse kuuluvad veerud***. Kui primaarvõtmes on mitu veergu, siis tuleb kasutajatele rohkem huvi pakkuvad ja sagedamini päringutes kasutatavad veerud panna ettepoole.
2. ***Alternatiivvõtmetesse kuuluvad veerud***. Kui alternatiiv-võtmes on mitu veergu, siis tuleb kasutajatele rohkem huvi pakkuvad ja sagedamini päringutes kasutatavad veerud panna ettepoole. Kui tabelis on mitu alternatiiv-võtit, siis tuleks ettepoole paigutada kasutaja jaoks olulisemad võtmed.
3. ***Välisvõtmetesse kuuluvad veerud***.
4. ***Ülejäänud kasutajale sisulist tähendust omavad veerud***, mis on grupeeritud vastavalt mõistetele, mille kohta nad käivad. Iga grupi sees võivad veeru nimed olla sorteeritud kas tähestiku järgi või selle järgi, millises järjekorras need veerud täidetakse (varem täidetavad veerud on eespool).
5. ***Lõppkasutajale sisulist tähendust mitteomavad veerud***
   (nt. *CV_aadress*, milles olev väärtus viitab CV dokumendi asukohale arvuti kataloogipuus. Selle veeru väärtus pakub huvi vaid rakendusprogrammile.)

Error! Not a valid embedded object.

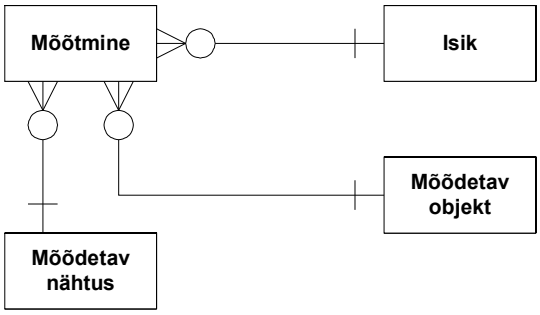
Joonis 57Hea näide veerunimede järjekorrast.

Veeru nimi	Põhjendus asukoha kohta veergude järjekorras
töötaja	primaarvõti
isikukood	alternatiivvõti
tiitel eesnimi perenimi	veergude grupp "Kuidas isiku poole pöörduda?"
aadress telefon e_mail faksi_nr	veergude grupp "Isiku kontaktandmed"
CV_aadress	andmebaasi kasutava programmi jaoks vajalik veerg.

Selline nimede järjestus:
- kiirendab mudeli ülevaatamist ja sellest arusaamist,
- kiirendab vigade ja puuduvate veergude avastamist.

6. Oracle CASE*Method soovitab olemi-suhte diagrammil paigutab olemitüübid nii, et "varesejalad" (>-) on avatud vasakule või üles (Hay, 1996).

Selle tulemusena paigutuvad füüsiliselt käegakatsutavaid objekte esitavad olemitüübid diagrammil alumisse paremasse serva. Mitte käegakatsutavaid objekte esitavad olemitüübid (nt. tegevused) paigutuvad diagrammi ülemisse vasakusse serva.



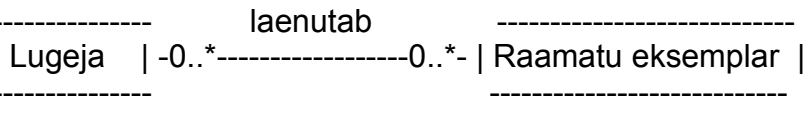
Joonis 58Olemi-suhte diagrammi soovitatav paigutus Oracle CASE*Methodi järgi.

Hoberman (2002) soovitab jälgida tähtskeemi, mille kohaselt paigutatakse andmemudelil põhiobjekt keskele ja sellega seotud objektid tähekujuliselt selle ümber. M:N suhte puhul peaks loodav vahetabel alati paiknema suhtes osalevate olemitüüpide põhjal loodud tabelite vahel.

7. UML klassidiagrammi joonistamine ASCII sümbolite abil

UML sümboli esitusviis ASCII sümbolite abil	UML sümboli nimi
------- | Isik | -------	Klass
<>----------> ^ V | | | V	Agregatsiooniseos
<| ---------- ^ - | | | # |	üldistusseos
---------->^ | | |	Sõltuvusseos

Näide:



[Lugeja] -|-------<**[Laenutab]**>--------|-**[Raamatu eksemplar]**

Joonis 59Olemi-suhte diagrammi joonistamine ASCII sümbolite abil.

Lugege mudelite kujundamise kohta ka artiklit:

Lieberman, B., 2004. "The art of modeling Part III: Visual composition", Rational Edge, January 2004. [WWW] http://www.ibm.com/developerworks/rational/library/2741.html (01.02.2004)

8. Mõisted

Eesti keeles	Inglise keeles
Kasutusjuhtude mudel	**Use case model**
Kasutusjuht	Use case
Tegutseja Tegija	Actor
Süsteem	System
Olemi-suhte diagramm	**Entity Relationship Diagram, ERD**
Olemitüüp	Entity type
Olem Olemi eksemplar Olemi esindaja	Entity Entity occurrence Entity instance
Suhtetüüp, seosetüüp	Relationship type
Suhe, seos, suhte eksemplar	Relationship occurrence
Seosetüübi aste	Degree of a relationship type
Suhe, milles osaleb kaks olemit	Binary relationship
Suhe, milles osaleb kolm olemit	Ternary relationship
Suhe, milles osaleb neli olemit	Quarternary relationship
Rekursiivne suhe	Recursive relationship
Atribuut	Attribute
Atribuudi väärtuste piirkond e. domeen	Attribute domain
Lihtatribuut	Simple attribute
Liitatribuut	Composite attribute
Üheväärtuseline atribuut	Single-valued attribute
Mitmeväärtuseline atribuut (Mitmene atribuut, korduv atribuut)	Multi-valued attribute
Tuletatud atribuut	Derived attribute
Kandidaatvõti	Candidate key
Primaarvõti	Primary key
Liitvõti	Composite key
Nõrk suhtetüüp (identifitseeriv suhe)	Weak relationship type
Nõrk olemitüüp	Weak entity type Child entity type Dependent entity type Subordinate entity type
Tugev seosetüüp	Strong relationship type
Tugev olemitüüp	Strong entity type Parent entity type Owner entity type Dominant entity type
Järk	Cardinality ratio
Osaluskohustus	Participation constraint
Kohustuslik osalemine	Mandatory participation
Valikuline osalemine	Optional participation
Laiendatud olemi-suhte diagramm	Enhanced Entity-Relationship model
Sidemeklass	Association class
Superklass	Superclass

Eesti keeles	Inglise keeles
Alamklass	Subclass
Tüüpide hierarhia	Type hierarchy Specialization hierarchy Generalization hierarchy IS-A hierarchy
Spetsialisatsioon	Specialization
Üldistus	Generalization
Osalemise piirang üldistusseose juures	Participation constraint
Kuuluvuse piirang üldistusseose juures	Disjoint constraint
Iga olem saab kuuluda vaid ühte alamtüüpi	Subclasses are disjoint
Iga olem saab kuuluda mitmesse alamtüüpi	Subclasses are non-disjoint
Agregatsioon	Aggregation
Kompositsioon	Composition
Tegevusdiagramm **Tegevusskeem**	**Activity diagram**
Toiming	Activity Action
Äriprotsesside ümbertöötlemine	Business Process Reengineering
Seisundidiagramm **Olekumuutuste diagramm** **Olekudiagramm**	**State Transition Diagram, STD**
Seisund Olek	State
Sündmus	Event
Üleminek	Transition
Valvurtingimus	Guard condition
Algseisund	Initial state
Lõppseisund	Final state
CRUD maatriks	**CRUD matrix**

9. Kasutatud materjalid

1. Activity Diagrams [WWW] http://www.isds.bus.lsu.edu/cvoc/learn/bpr/cprojects/spring1998/modeling/activity.html (28.09.2001)
2. Barker, R., 1990. *CASE Method: Entity Relationship Modelling*. Addison-Wesley Professional.
3. Codd, E. F., 1979. Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, Vol. 4, No. 4 (December 1979), pp. 397–434.
4. Connolly, T.M. & Begg, C.E., 2002. *Database systems. A Practical Approach to Design, Implementation and Management*. Third Edition. Pearson Education. 1236 p.
5. Date, C. J., 2003. *An Introduction to Database Systems.* Eighth Edition. Addison Wesley. 983 p.

6. Evitts, P.A., 2000. *UML Pattern Language*. Macmillian Technical Publishing. 257 p.
7. Hay, D.C., 1996. *Data model patterns: conventions of thought*. New York : Dorset House Pub., 1996. 268 p.
8. Fowler, M. & Scott, K., 1999. *UML Distilled*. 2nd edition. Addison-Wesley. 224 p.
9. Fowler, M., 2007. *UMLi kontsentraat*. 3. redaktsioon. Objektmodelleerimise standardkeele UML2.0 lühijuhend. Cybernetica AS.
10. Hoberman, S., 2002. *Data Modeler's Workbench: tools and techniques for analysis and design*. Wiley Computer Publishing. 472 p.
11. Larman, C., 1997. *Applying UML and patterns : an introduction to object-oriented analysis and design*. Upper Saddle River (N.J.) : Prentice Hall PTR. 507 lk.
12. Larman, C., 2002. *Applying UML and patterns : an introduction to object-oriented analysis and design.* Second Edition, Prentice Hall, Upper Saddle River (N.J.). 627 lk.
13. Miliauskaite, E. & Nemuraite, L. 2005. Representation of integrity constraints in conceptual models. Information Technology And Control, Kaunas, Technologija, 2005, Vol.34, No.4, pp. 355 – 365.
14. Marshall, C., 1999. *Enterprise modeling with UML : designing successful software through business analysis*. Reading, Mass. : Addison-Wesley. 259 p.
15. Mikli, T., 1999. *Sissejuhatus infosüsteemidesse*. Tallinn : TTÜ. 99 lk.
16. Robertson, S. & Robertson, J., 1999. *Mastering the Requirements Process*. Addison-Wesley. 404 p.
17. Scott, W.A., Introduction to the Diagrams of UML 2.0 [WWW] http://www.agilemodeling.com/essays/umlDiagrams.htm (06.03.2007)
18. Silverston, L., 2001a. *The Data Model Resource Book: A Library of Universal Data Models for All Enterprises*. Revised Edition, Vol. 1. Wiley Computer Publishing.
19. Silverston, L., 2001b. *The Data Model Resource Book: A Library of Universal Data Models by Industry Types*. Revised Edition. Vol. 2, Wiley Computer Publishing.
20. Vendelin, J., 2003. *Rakenduste loomine andmebaasiga MS Access*, Tallinn : TTÜ. 60 lk.