Harri Laine

Johdatus sovellussuunnitteluun

Lentomoniste – osa 1

Helsingin yliopisto

Tietojenkäsittelytieteen laitos

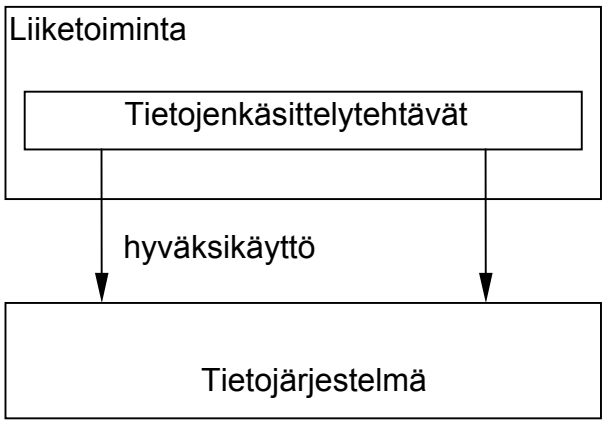
syksy 2002

Sisältö:

1 Järjestelmän kehittämisen vaiheet

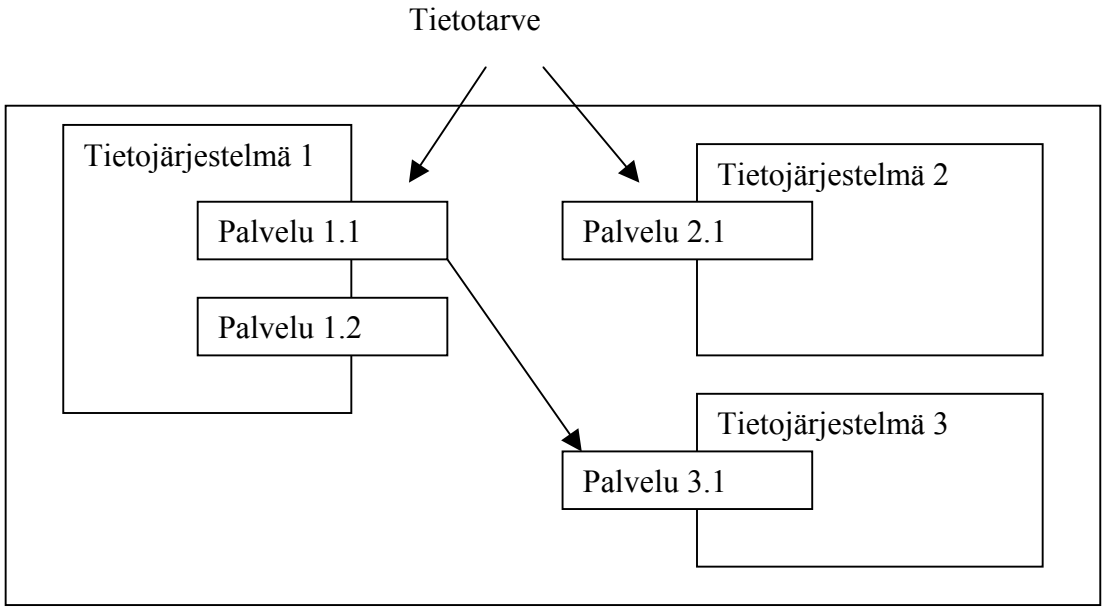
1.1 Peruskäsitteitä

Tietojärjestelmä kehitetään tukemaan jotain toimintaa, jota kutsumme jatkossa liiketoiminnaksi. Toiminta voi olla aitoa yrityksen harjoittamaa liiketoimintaa tai jotain muuta aktiviteettia, jota voidaan tehostaa tai järkeistää suorittamalla joitakin toiminnassa tarvittavia tietojenkäsittelytehtäviä tietojärjestelmän palveluja käyttäen (kuva 1.1).



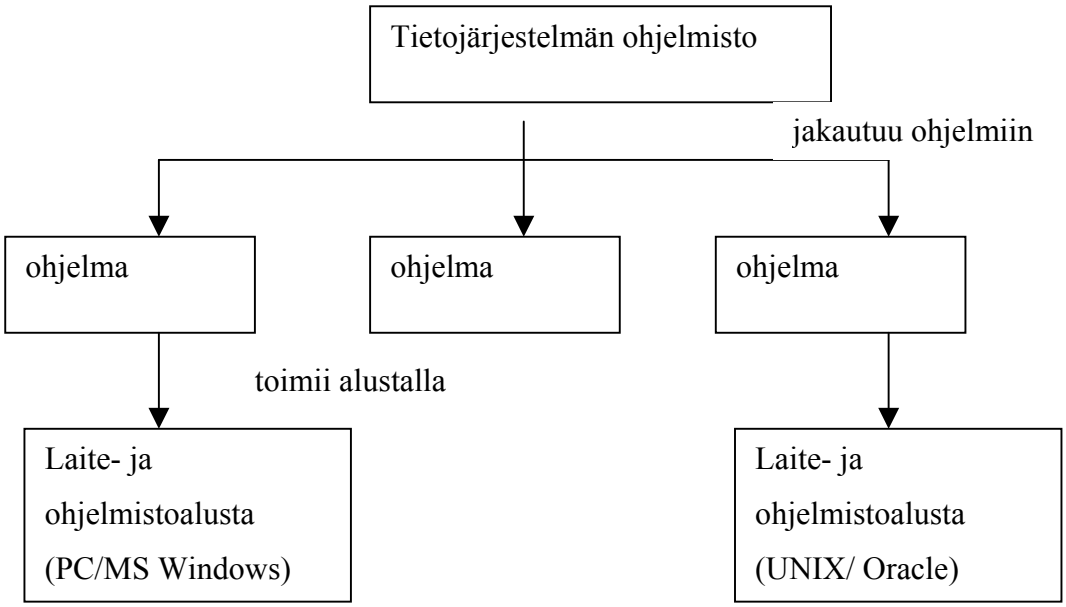
Kuva 1.1: Tietojärjestelmä liiketoiminnan tukena.

Tietojärjestelmä voidaan nähdä tietojenkäsittelypalveluiden tarjoajana. Tietojenkäsittelypalvelun tehtävänä on tyydyttää jokin tietotarve. Tämä voi olla tiedon tallentaminen myöhempää käyttöä varten, tiedon haku tai jokin laskenta- tai tiedonmuokkaustoimenpide. Palvelun tuottamiseen voi osallistua useita tietojärjestelmiä. (kuva 1.2).



Kuva 1.2: Palvelujen hyödyntäminen tietotarpeen tyydyttämiseksi.

Tietojärjestelmä on jollakin perusteella muodostettu hallinnollinen kokonaisuus, jonka voi katsoa muodostuvan tietosisällöstä ja siihen perustuvista palveluista sekä palvelujen aikaansaamiseen tarvittavasta ohjelmistosta ja laitteistosta. Tietojärjestelmän ohjelmisto muodostuu yhdestä tai useammasta jollakin laitteisto- ja ohjelmistoalustalla toimivasta ohjelmasta. Tällaisia alustoja voi yhden järjestelmän puitteissa olla useita, esimerkiksi MS-Windows ohjelmistolla varustetut työasemat ja UNIX-käyttöjärjestelmällä ja relaatiotietokantaohjelmistolla varustettu tietokantapalvelin (kuva 1.3). Ohjelmistoalustan tarjoavia ohjelmia kutsutaan varusohjelmiksi ja niiden päällä toimivia tietojärjestelmän palveluita tuottavia ohjelmia sovellusohjelmiksi.



Kuva 1.3: Tietojärjestelmän ohjelmisto

Tietojärjestelmä on usein tarkoituksenmukaista jakaa osajärjestelmiksi. Osajärjestelmän muodostamisperusteena voisi olla esimerkiksi:

- avustettava toimintokokonaisuus, esimerkiksi kurssihallinnon toimintoja voisi avustaa kurssihallinto-osajärjestelmä
- yhteiset käyttäjät, esimerkiksi kirjaston asiakkaita varten olisi tarjolla erityinen asiakasjärjestelmä
- yhteinen tietosisältö, esimerkiksi kaikki tilauksiin kohdistuvat palvelut keskitettäisiin tilaustenkäsittelyjärjestelmään,
- maantieteellinen sijainti, esimerkiksi kullakin varastolla voisi olla oma paikallinen varastonhallintaosajärjestelmänsä,
- yhteinen tehtävätyyppi, esimerkiksi raporttien tuottaminen olisi keskitetty raportointiosajärjestelmään,
- samanlainen ajoitus, esimerkiksi päivittäin tarvittavat palvelut ja vain kirjanpitovuoden vaihtuessa tarvittavat palvelut olisi erotettu eri osajärjestelmiin,
- yhteinen toteutustekniikka tai –väline, esimerkiksi jonkin kehittimen käyttöön perustuvat palvelut.

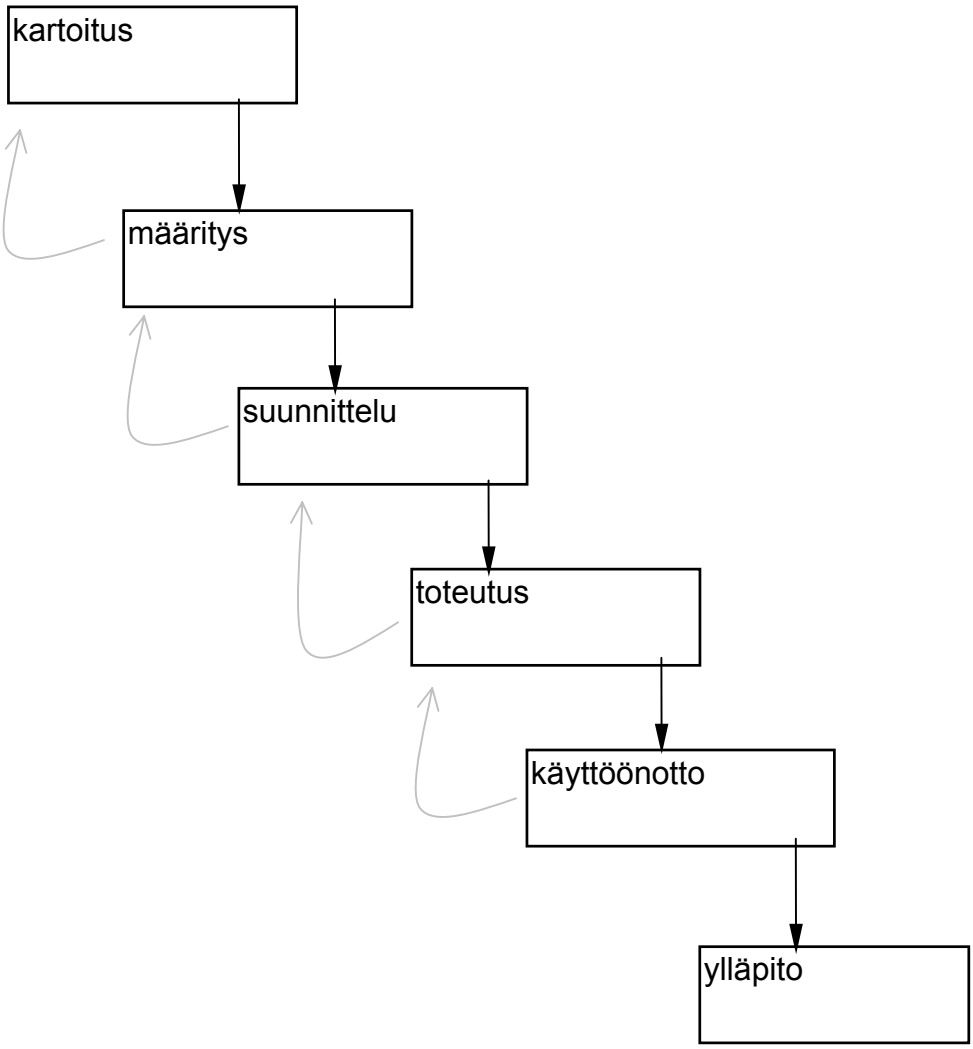
Näitä samoja perusteita voi käyttää koko järjestelmän rajauksessa. Jos järjestelmä jaetaan hierarkkisesti osajärjestelmiin voivat jakoperusteet vaihdella tasoittain ja jopa saman tason sisällä voi olla eri perustein muodostettuja osajärjestelmiä.

Tietojärjestelmän kehityksen yhteydessä on selvitettävä:

- **mihin tarkoitukseen järjestelmä tehdään (why?)**
  - millaista toimintaa järjestelmällä pitäisi tukea
  - mitä tietojenkäsittelytehtäviä järjestelmä avustaa
  - ketkä suorittavat avustettavia tietojenkäsittelytehtäviä
  - mitä hyötyä / kustannuksia järjestelmästä on
  - millaisia ongelmia järjestelmällä halutaan poistaa
- **millainen järjestelmä tehdään (what?)**
  - ketkä ovat järjestelmän käyttäjät
  - mitä palveluja järjestelmä tarjoaa / kenelle
  - millainen on järjestelmän tietosisältö
  - millaisessa ympäristössä järjestelmä toimii
- **miten järjestelmä tehdään (how?)**
  - millainen on järjestelmän tietokannan rakenne
  - miten järjestelmää käytetään
  - millainen on ohjelmiston rakenne

1.2 Järjestelmän elinkaari

Tietojärjestelmän **elinkaarella** tarkoitetaan aikaa, joka kuluu kehittämisen aloittamisesta järjestelmän poistamiseen käytöstä. Järjestelmän elinkaari voidaan jakaa vaiheisiin monella eri tavalla. Järjestelmien kehittämismenetelmät sisältävät yleensä jonkinlaisen vaihejakomallin, jonka mukaisesti järjestelmän kehityksen oletetaan etenevän. Vaihejakomallit voivat olla hyvinkin yksityiskohtaisia. Karkealla tasolla voidaan yleensä erottaa ainakin seuraavat päävaiheet: kartoitus, määritys, suunnittelu, toteutus (sisältää testauksen), käyttöönotto ja ylläpito (kuva 1.4). Kuhunkin vaiheeseen liittyy omat käsitteistönsä, työtapansa ja työkalunsa.



Kuva 1.4: Järjestelmän elinkaaren vaiheet.

Ideaalitilanteessa järjestelmän kehitys etenee kuvan 1.4 mukaisesti ylhäältä alas suoraviivaisesti. Käytännössä joudutaan kuitenkin usein palaamaan takaisin aiempaan vaiheeseen ja uusimaan kyseisessä vaiheessa tehtyjä ratkaisuja. Tätä esittävät kuvan paluunuolet. Mitä kauemmaksi joudutaan palaamaan sitä kalliimmaksi uudelleen tekemisen kustannukset muodostuvat. Järjestelmän kehittämiskustannusten jakautumisesta vaiheiden kesken on laadittu useita selvityksiä. Taulukossa 1.1 on erään tällaisen tuloksena saatu kustannusjakautuma

Vaihe	% kaikista	% käyttöönottoon asti
Kartoitus	3	9
Määrittely	5	15
Suunnittelu	7	20
Ohjelmointi *	6	18
Moduulitestaus *	7	20
Integrointitestaus *	6	18
Ylläpito	67	

Taulukko 1.1: Elinkaaren aikaisten kustannusten jakautuminen vaiheisiin[1]
Tähdellä (*) merkitys sisältyvät kuvan 1.4 toteutusvaiheeseen.

1.2.1 Kartoitus

Kartoitusvaiheessa selvitetään järjestelmän tarkoitus, järjestelmän yhteys tuettavaan liiketoimintaan sekä järjestelmän kehittämisen kannattavuus. Tässä vaiheessa tutkitaan myös erilaisia vaihtoehtoja järjestelmän hankkimiseksi. Tällaisia voisivat olla valmiin järjestelmän ostaminen, yleiskäyttöisen järjestelmän hankkiminen ja sen virittäminen omiin tarpeisiin sopivaksi, tai oman 'räätälöidyn' järjestelmän teettäminen. Kartoitusvaiheessa asetetaan myös tavoitteita rakennettavalle järjestelmälle. Tavoitteet ovat aluksi karkeita ja ne täsmennetään määritysvaiheessa.

1.2.2 Määritysvaihe

Määritysvaiheessa laaditaan täsmällinen suunnitelma laadittavan järjestelmän sisällöstä. Vaiheen osatehtävänä voi olla **nykyjärjestelmän analysointi**, jonka yhteydessä selvitetään millainen nykyinen tietojärjestelmä on ja mitä ongelmia sen käyttöön liittyy. Tähän on periaatteessa olemassa 'oikea ratkaisu', joka vain pitäisi löytää. Koska kuva nykyjärjestelmästä perustuu ihmisten käsityksiin, ei esiin saatava kuva ole kuitenkaan välttämättä objektiivinen.

[1] Schach S.R.: Software engineering, 2. ed., Aksen Associates, 1993.

Tavoitteiden asetus -osatehtävässä kuvataan täsmällisesti kehitettävän järjestelmän tavoitteet. Tavoitteet saadaan joko käyttäjiltä tai muilta sidosryhmiltä. *Sidosryhmällä* tarkoitetaan tahoa, joka on jollain tavoin tekemisissä järjestelmän kanssa, esimerkiksi saa järjestelmän tuottamia raportteja, tai tuottaa järjestelmässä kirjattavaa informaatiota. Tavoitteet voivat liittyä ongelmien poistamiseen, liiketoiminnan edistämiseen tai järjestelmän välttämättömään sopeutumiseen ympäristön muutoksiin. Tavoitteet voivat kohdistua eri asioihin. Tyypillisesti tavoitteita esitetään koskien:

- suorituskykyä,
- tietotarpeiden tyydytystä,
- kustannuksia,
- valvonnan kehittämistä,
- tehokkuuden parantamista,
- palvelujen kehittämistä,
- nykyjärjestelmän työnkulkua tai
- muita nykyjärjestelmän ongelmia.

Tavoitteet pitäisi esittää mahdollisimman täsmällisesti, jos mahdollista niin selkeinä numeroina (esim. tavoite ‘ilmoittautumiseen käytettävä aika on pudotettava 50 % nykyisestä’ on parempi kuin ‘ilmoittautumista on nopeutettava’). Jos tavoitteet liittyvät uusiin palveluihin ja uusiin tietotarpeisiin, on nämä määriteltävä riittävän täsmällisesti. Tavoitteiden asettamiseen liittyy myös tavoitteiden tärkeysjärjestyksen määrittely.

Tavoitteet voidaan saavuttaa useilla eri keinoilla. **Järjestelmän tavoitetilan määrityksessä** laaditaan karkea suunnitelma järjestelmästä. Tehtävänä on määritellä, millainen tulevan järjestelmän pitäisi olla. Tämän määrittelemiseksi on kuvattava täsmällisesti

- järjestelmän käyttäjät ja
- järjestelmän käyttäjilleen tarjoamien palvelujen sisältö.

Palvelujen sisällön täsmällinen kuvaaminen edellyttää yleensä myös järjestelmän tietosisällön määrittelyä. Järjestelmän tekniseen toteutukseen otetaan kantaa vain periaatteiden osalta. Yksityiskohtiin ei puututa. Niinpä tavoitetilan yhteydessä määritellään yleensä järjestelmän yleisarkkitehtuuri: minkälaisessa ympäristössä järjestelmä

toimii, millaisia periaateratkaisuja esimerkiksi tietokannan (keskitetty / hajautettu, tietokannanhallintajärjestelmä / tavalliset tiedostot), tietoliikenteen ja käyttöliittymien suhteen sovelletaan. Joskus on tarpeen kiinnittää jo tässä vaiheessa myös palvelujen muoto (esim. käyttöliittymän ulkoasu). Tehtävän kuluessa pitäisi laatia useita vaihtoehtoisia suunnitelmia. Näistä suunnitelmista valitaan jokin pohjaksi jatkotyöskentelylle.

1.2.3 Suunnitteluvaihe

Suunnitteluvaiheessa tarkennetaan määrittelyssä laadittua karkeaa suunnitelmaa. Tarkennuksen yhteydessä otetaan kantaa siihen, miten määrityksen yhteydessä kuvattu järjestelmän tavoitetila teknisesti saadaan aikaan. Suunnittelun osa-alueita ovat käyttöliittymän suunnittelu, tietokannan suunnittelu, ohjelmistosuunnittelu, laitteiston valinta ja työnkulun suunnittelu.

Käyttöliittymän suunnittelu on sidoksissa **ohjelmiston suunnitteluun**. Kummankin peruslähtökohtana ovat järjestelmän tarjoamat palvelut ja näiden paketointi. Syötteiden ja tulosteiden rakenteen suunnittelu voidaan katsoa osaksi käyttöliittymän suunnittelua.

Käyttäjien tarpeet kohdistuvat lähinnä tulosteisiin. Perinteinen suunnittelujärjestys onkin edetä tulosteista tiedostojen kautta syötteisiin. Tässä mallissa eli ns. **sovelluskeskeisessä suunnittelussa**, tulosteiden rakenne ja sisältö määräävät varastoidun tiedon rakenteen ja sisällön. Useat sovellukset voivat kuitenkin käyttää samoja tietoja. Jos varastoitavan tiedon rakenne sidotaan nykyisiin tulosteisiin, joudutaan sitä jossakin vaiheessa muuttamaan. Koska ohjelmat perustuvat tietojen rakenteeseen joudutaan myös ohjelmia muuttamaan. Tästä aiheutuu runsaasti kustannuksia. **Tietokeskeisessä suunnittelussa** varastoitavan tiedon rakenne pyritään suunnittelemaan tietojen välisten riippuvuuksien mukaisesti. Rakenne ei tällöin riipu tulosteista. Seurauksena on yleensä tulosteiden tuottamisen kustannusten jonkin asteinen kallistuminen. Toisaalta tulosteita voidaan helposti muuttaa kajoamatta varastoidun tiedon rakenteeseen. Muutokset pysyvät paikallisina ja ovat halvempia kuin sovelluskeskeisessä suunnittelussa.

Työtehtävien suunnittelussa ratkaistaan, miten käyttäjät toimivat kehitettävän järjestelmän kanssa. Työtehtävä on kokonaisuus, joka voidaan osoittaa jollekin käyttäjälle tai käyttäjäryhmälle. Työtehtäviä suunniteltaessa on varmistettava, että kaikki järjestelmän toiminnot otetaan huomioon. Suunnittelussa on kiinnitettävä huomiota myös siihen, millaiseksi henkilöiden työnkuva muodostuu.

1.2.4 Toteutus, käyttöönotto ja ylläpito

Toteutusvaiheessa laaditaan ohjelmistosuunnitelmassa määritellyt ohjelmat ja testataan ne. Samoin laaditaan järjestelmän käyttöön liittyvät ohjeistot ja koulutusmateriaali. Tarvittavat laitteet on myös hankittava.

Käyttöönotossa ohjelmisto asennetaan toimintaympäristöönsä. Tiedot siirretään vanhoista järjestelmistä uuteen. Käyttäjien käyttöluvat hoidetaan kuntoon ja käyttäjät koulutetaan. Järjestelmä otetaan käyttöön aluksi mahdollisesti rinnan vanhan järjestelmän kanssa. Kun järjestelmä on otettu käyttöön sen toimintaa seurataan ja mahdolliset puutteet ja ongelmat kirjataan ylös.

Ylläpitovaiheessa korjataan järjestelmästä löytyneitä puutteita, täydennetään järjestelmää uusilla palveluilla ja sopeutetaan järjestelmää toimintaympäristössä tapahtuviin muutoksiin. Järjestelmien kaikista kehityskustannuksista ylläpidon kustannukset ovat noin 70 prosenttia, näistä suurin osa on ns. kehittävää ylläpitoa eli palveluiden täydentämistä ja parantamista.

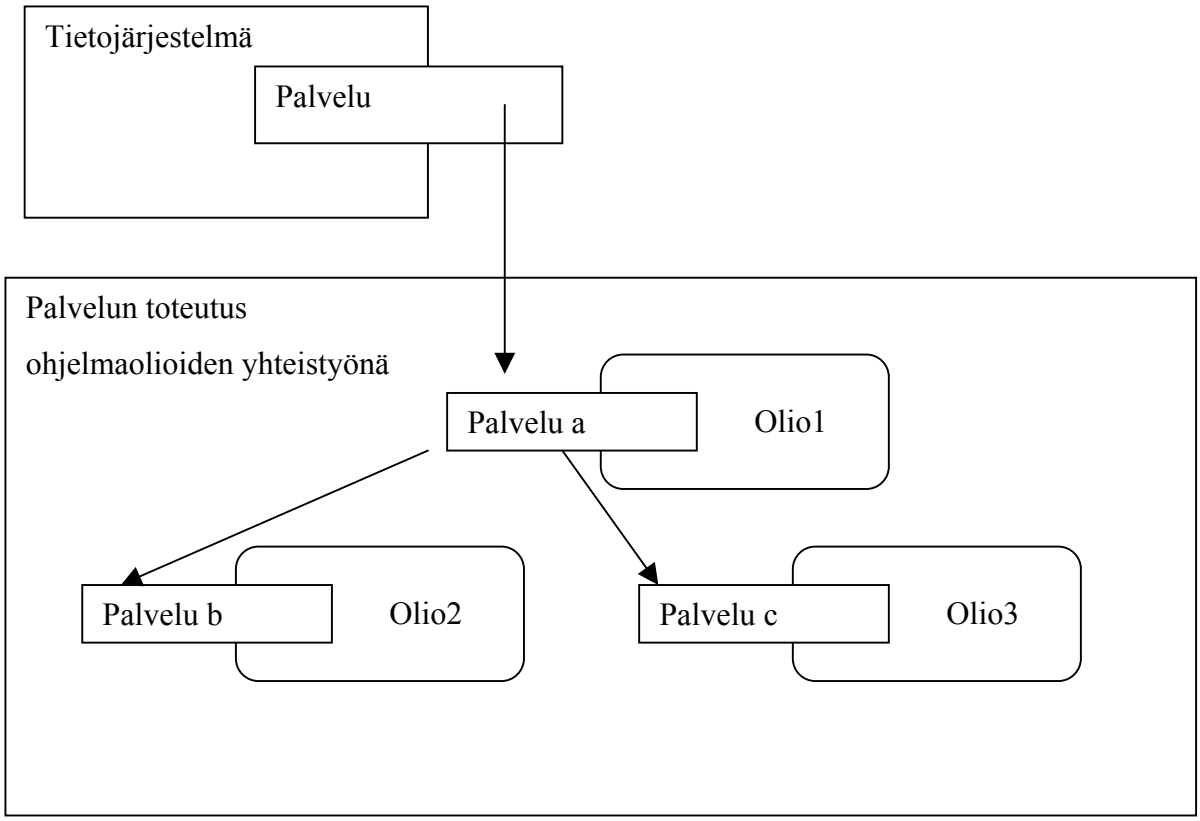
2 Tietojärjestelmän kuvaaminen

Tietojärjestelmän elinkaaren aikana järjestelmästä laaditaan lukuisia kuvauksia. Kartoitusvaiheessa syntyy alustava hyvin karkea hahmotelma järjestelmässä Määritysvaiheessa tämä täsmentyy järjestelmän ulkoisen olemuksen kuvaavaksi täsmälliseksi kuvaukseksi. Suunnitteluvaiheessa kuvauksen yksityiskohtaisuus edelleen kasvaa, myös järjestelmän sisäinen tekninen ratkaisu kuvataan, aluksi

karkealla tasolla (arkkitehtuurina) ja lopuksi hyvinkin yksityiskohtaisesti. Ohjelmoinnissa järjestelmä kuvataan tietokoneelle ohjelmointikielen avulla.

Laadittavien kuvausten tarkoituksena on välittää tietoa järjestelmästä sen kehittämiseen osallistuvien henkilöiden välillä. Järjestelmän kuvaamisen pohjana on aina oltava jokin malli siitä millaisia kuvattavia asioita järjestelmästä pitäisi hahmottaa ja mitä näistä on oleellista esittää. Jotta kuvauksia olisi helppo lukea tarvitaan myös yhteinen kuvaustekniikka. Tietojärjestelmiin liittyen on esitetty useita malleja ja vielä useampia kuvaustekniikoita. Paljous johtuu osittain siitä, että järjestelmiä on kovin monenlaisia ja kovin erilaisilla sovellusalueilla. Esimerkiksi tietoliikennejärjestelmää kuvattaessa tärkeät kuvattavat asiat ovat jossakin määrin erilaisia kuin vaikkapa kuvankäsittelyjärjestelmää tai asiakaspalvelujärjestelmää kuvattaessa. Järjestelmien abstraktisuus ja vakiintuneen teoriapohjan puute tuottavat uusia malleja.

Tällä kurssilla tarkastellaan järjestelmien oliopohjaista (object oriented) kuvaamista. Oliopohjaisuus sinällään on metatason malli (malli mallista), jonka mukaan minkä tahansa järjestelmän (ei siis pelkästään tietojärjestelmän) voidaan katsoa muodostuvan olioista, jotka yhteistyössä toimien ja toistensa palveluja hyväksikäyttäen tuottavat järjestelmän palvelut. Järjestelmä itsekin on tällöin olio. Tällaista metamallia voidaan luontevasti soveltaa useisiin tilanteisiin. Liiketoiminnassa on tyypillisesti käytetty organisaatioyksikköjakoa. Organisaatioyksiköllä on tietyt vastuut ja se tuottaa palveluja muille organisaatioyksiköille. Se on siis ilmiselvä olio. Yritys pyörii sen organisaatioyksiköiden yhteistyön ansiosta. Olio-ohjelmoinnissa tietojärjestelmän tarjoamia tietojenkäsittelypalveluita tuotetaan tietokoneen sisuksissa toimivien tietojenkäsittelyolioiden yhteistyön tuloksena (kuva 2.1). Oliot toimivat osina isommissa ja monimutkaisemmissa olioissa, jotka tuottavat omat laajemmat ja monimutkaisemmat palvelunsa alkeellisempien olioiden yksinkertaisempia palveluita hyödyntäen.



Kuva 2.1: Tietojärjestelmän palvelun tuottaminen

Koska oliopohjaisuus on metamalli sitä voidaan soveltaa tietojärjestelmienkin kuvaamiseen eri tavoin. Eri menetelmät esittelevät erilaisia soveltamistapoja eli malleja järjestelmien kuvaamiseen oliopohjaisesti. Menetelmät esittelevät myös kuvaustekniikoita, joita tulisi käyttää. Kuvaustekniikat ovat usein ainakin osittain graafisia, koska graafisen kuvauksen avulla on tekstikuvausta helpommin saatavissa kokonaiskuva kuvattavasta kohteesta. Tällä kurssilla ei käsitellä mitään tiettyä menetelmää järjestelmän kehittämiseksi. Kurssilla käsitellään oliopohjaisten kuvausten laatimiseksi kehitettyä UML-kuvaustekniikkaa sekä joitain keskeisiä eri menetelmille yhteisiä järjestelmäkuvauksia.

UML (Unified Modeling Language) kuvaustekniikka kehitettiin yhdistämällä kolmen tunnetuimman ns. ensimmäisen sukupolven oliomenetelmän käyttämät kuvaustekniikat: Boochin oliotekniikka[2], Rumbaugh:n ja kumppaneiden OMT [3] ja

[2] Booch G.: Object Oriented Design with Applications, Benjamin/Gummings, 1991

[3] Rumbaugh J. et al: Object-Oriented Modeling and Design, Prentice-Hall, 1991

Jacobsonin OOSE [4]. UML-tekniikan peruskehittämisestä on huolehtinut Rational Software –yritys, jonka osakkaina em. tekniikkojen kehittäjät ovat. Oliotekniikan käyttöä edistämään perustettu OMG (Object Management Group) - yhdistys valitsi UML:n omaksi kuvausjärjestelmästandardikseen loppuvuodesta 1997. OMG:ssä ovat jäseninä useat merkittävät ohjelmistoyritykset, joten valinta on varsin merkittävä askel oliopohjaisten kuvausten yhtenäistämiseksi. Hyvin monet tietokoneavusteisten suunnitteluvälineiden (CASE[5]-välineiden) tuottajat ovat jo muuttaneet tai muuttamassa välineitään UML-tekniikkaan perustuviksi.

UML määrittelee joukon kaaviotyyppejä käytettäväksi järjestelmän kuvaamiseen. Tiettyä kaaviotyyppiä voi käyttää useassa tilanteessa. Esimerkiksi luokkakaaviota (class diagram) voi käyttää ohjelman rakenteen kuvaamiseen, järjestelmän osajärjestelmäjaon kuvaamiseen, järjestelmän tietosisällön kuvaamiseen, jne. UML ei määrittele missä tilanteissa tiettyä tekniikkaa pitäisi käyttää. Tämä on kehittämismenetelmien tehtävä. UML on laajennettava kuvauskieli. Kielen käyttäjällä on mahdollisuus lisätä siihen omia piirteitä.

UML:n tarjoamat kaviotekniikat ovat:

- luokkakaavio (class diagram),
  tiedot ja mitä niillä tehdään
- oliokaavio (object diagram),
  tietojen välisten riippuvuuksien havainnollistaminen
- käyttötapauskaavio (use case diagram),
  järjestelmän palvelut
- tilakaavio (statechart)
  olioiden käyttäytyminen, elinkaari
- olioiden yhteistyökaaviot
  miten oliot toimivat yhdessä
  - palveluketjut (sequence diagram)
  - yhteistyörakenteet (collaboration diagram)
- aktiviteettikaavio (activity diagram)

[4] Jacobson I., et al: Object-Oriented Software Engineering, ACM Press/Addison-Wesley, 1992

kontrollin kulku ohjelman suorituksessa

- komponenttikaavio (component diagram)
  ohjelman muodostuminen erillisistä komponenteista
  erikoistapaus oliokaaviosta
- sijoittelukaavio (deployment diagram)
  ohjelmiston sijoittelu laitteistoille

Tällä kurssilla käsitellään näistä olio- ja luokkakaaviot, käyttötapauskaaviot ja olioiden yhteistyökaaviot.

3 Järjestelmän palvelut

Tietojärjestelmät tarjoavat tietoa sekä käyttäjilleen että epäsuorasti muille tahoille. Tahoja, jotka ovat järjestelmän ulkopuolella, mutta kuitenkin palvelujen kautta kytkeytyneitä järjestelmään kutsutaan järjestelmän sidosryhmiksi. Tällainen taho voi toimia

- tiedon tuottajana järjestelmään tai
- tiedon hyväksikäyttäjänä

Sidosryhmien tunnistaminen on ensimmäinen tehtävä järjestelmän palveluja määriteltäessä. Tässä tehtävässä eri sidosryhmät nimetään sekä määritellään lyhyellä tekstikuvauksella. Sidosryhmien löytämiseksi voidaan esittää kysymykset:

- kuka / mikä saa tulosteita järjestelmästä?
- kuka / mikä toimittaa tietoa järjestelmään?
- kuka käyttää järjestelmää?
- mihin muihin järjestelmiin kehitettävä järjestelmä on yhteydessä.

[5] CASE = Computer Aided Software Engineering

Näiden kysymysten perusteella luokitellaan samankaltaiset tahot sidosryhmiksi. Tietojenkäsittelytieteen laitoksen opetustietojärjestelmän sidosryhmiä ovat esimerkiksi

- opiskelijat
- opettajat
- vastuuhenkilöt
- laitoksen johtoryhmä
- siivoojat
- vahtimestarit
- opintosuoritusrekisteri-järjestelmä.

Järjestelmä voi tarjota eri sidosryhmille erilaisia **palveluita**. Palvelujen tai tarpeiden erilaisuus onkin yksi peruste ryhmien muodostamiselle. Sidosryhmä voi olla suorassa yhteydessä järjestelmään tai joko saada tietoja järjestelmästä tai toimittaa tietoa järjestelmään epäsuorasti. Esimerkiksi yllä olevista sidosryhmistä siivoojat eivät ole suorassa yhteydessä opetustietojärjestelmään kuten ei myöskään laitoksen johtoryhmä. Merkittävän osuuden suorassa yhteydessä järjestelmään olevista sidosryhmistä muodostavat järjestelmän käyttäjät. Yllä olevista sidosryhmistä käyttäjäryhmiä ovat opiskelijat, opettajat, vastuuhenkilöt ja vahtimestarit. Opintosuoritusrekisterijärjestelmä on erillinen järjestelmä, jonka palveluita laitoksen opetustietojärjestelmä hyödyntää (opintosuoritusote, kirjauspalvelut).

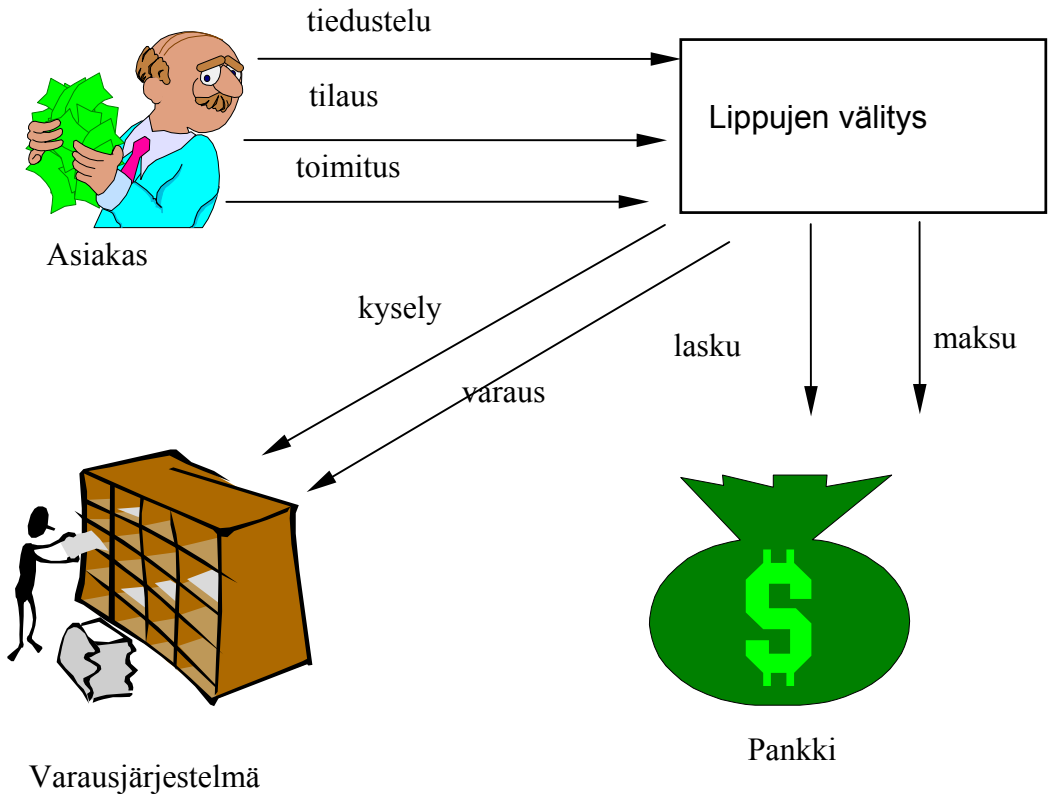
3.1 Sidosryhmäkaavio

Karkean tason yleiskuva järjestelmästä voidaan esittää **sidosryhmäkaaviona (context diagram)**, jossa näkyvät

- järjestelmä, keskeisenä komponenttina,
- järjestelmän sidosryhmät ja
- tärkeimmät järjestelmän tarjoamat ja käyttämät palvelut tai palvelukokonaisuudet.

Palvelut esitetään kaaviossa sidosryhmän järjestelmään yhdistävänä viivana. Sidosryhmäkaavion tulisi antaa yleiskuva järjestelmästä. Tämän vuoksi sitä ei voi esittää kovin yksityiskohtaisella tasolla. Samankaltaisia palveluita on syytä koota yhteen ryhmiksi, jotka sitten esitetään kaaviossa yhtenä palvelukokonaisuutena. Käytämme sidosryhmäkaaviota siten, että liitämme palveluja kuvaavaan viivaan nuolenkärjen, joka osoittaa palvelun tarjoajaan. Jos kyseessä on järjestelmän itsensä tarjoama palvelu osoittaa nuoli siis järjestelmään. Jos taas kyseessä on ulkoisen järjestelmän käyttö, osoittaa nuoli kyseiseen ulkoiseen järjestelmään.

Kuvassa 3.1 on vapaamuotoisin symbolein esitetty yleiskuvaus lipunvälitysjärjestelmästä. Asiakas käyttää järjestelmän palveluita 'tiedustelu', 'tilaus' ja 'toimitus'. Lipunvälitysjärjestelmä käyttää elokuvayhtiökohtaisten varausjärjestelmien palveluita 'kysely' ja 'varaus' sekä pankkijärjestelmien palveluita 'lasku' ja 'maksu'.



Kuva 3.1: Lipunvälitysjärjestelmän yleiskaavio.

Pelkkä kaavio ei ole riittävä järjestelmän yleiskuvan antamiseksi. Kaavion täydennykseksi tarvitaan sanalliset kuvaukset sekä sidosryhmistä että tuotettavista ja käytettävistä palveluista.

3.2 Käyttötapausmalli

Tietojärjestelmä tarjoaa käyttäjilleen palveluita, jotka perustuvat järjestelmän tietosisältöön. Järjestelmän toiminnalliset vaatimukset voidaan määritellä määrittelemällä millaisia palveluita järjestelmän on tarjottava. Käyttötapausmalli (use case model) on viime aikoina suosioon tullut tapa järjestelmän palvelujen määrittelyyn. [6] UML tarjoaa kuvaustekniikan käyttötapausmallin esittämiseen.

Käyttötapauksella mallinnetaan käyttäjän järjestelmän avulla suorittamaa tehtävää eli tapaa käyttää järjestelmää. Luonteva tapa on kytkeä käyttötapaukset käyttäjän työtehtäviin. Käyttötapauksen laajuus on hyvin keskeinen tekijä mallin ymmärrettävyyden ja hallittavuuden kannalta. Käyttötapaukset eivät saisi olla liian laajoja eivätkä myöskään liian suppeita. Keskikokoisessa hankkeissa käyttötapauksia on yleensä muutamia kymmeniä ja isommissa hankkeissa jopa satoja.

Pääperiaatteena on, että yhden käyttötapauksen tulisi muodostaa looginen kokonaisuus, jolla on sekä selvä lähtökohta että merkityksen omaava lopputulos. Lähtökohtana eli herätteenä on tapahtuma tai tarve, joka käynnistää käyttötapauksen. Tapahtuma voi olla joko käyttäjän itsensä, jonkin ulkopuolisen tekijän tai järjestelmän aiheuttama. Se voi olla myös ajan kulumisesta aiheutuva. Kuvan 5.1 esimerkissä asiakkaan saamia palveluita voidaan ajatella käyttötapauksina. Koska kyseessä ei ole asiakkaan työtä avustava järjestelmä, ne eivät liity asiakkaan työtehtäviin. Tiedustelu-käyttötapauksen lähtökohtana on tietotarve ja lopputuloksena haluttu tieto. Tilaus -käyttötapauksen lähtökohtana on tarve varata paikka näytökseen ja lopputuloksena aikaansaatu varaus.

[6] Jacobson: Object-Oriented Software Engineering: A use case driven approach, Addison-Wesley,1992

Termiä käyttötapaus käytetään yleisesti sekä luokkatason käsitteenä että ilmentymätason käsitteenä. Jatkossa käytetään mahdollisissa epäselvissä tilanteissa luokkatasolla termiä käyttötapausluokka ja ilmentymätasolla termiä käyttötapaus-ilmentymä tai esimerkkitapaus. UML:ssä käyttötapauksia kuvataan luokkatasolla.

Esimerkki:

Käyttötapausluokka:

Paikan varaaminen elokuvanäytökseen

Esimerkkitapaus:

Kalle Kenkkunen varaa paikan 308 Tennispalatsi 12:n näytökseen 20.11.1999 klo 21.

Käyttötapauksella on käyttäjä (actor), joka käyttötapauksessa toimii vuorovaikutteisesti järjestelmän kanssa toteuttaakseen tavoitteensa. Käyttäjän toiminta on syötteiden antamista ja palautteen saamista. Usein käyttäjä on ihminen, mutta käyttäjä voi olla myös ulkoinen järjestelmä. Käyttötapaukseen liittyy aina tavoite = asia, jonka käyttäjä haluaa saada aikaan käyttötapauksen avulla.

Käyttötapausta kuvattaessa kerrotaan käyttäjän ja järjestelmän välisen vuoropuhelun sisältö, mitä käyttötapauksessa tapahtuu ja mistä asioista käyttäjä ja järjestelmä vaihtavat tietoa. Miten kommunikointi käytännössä tapahtuu määritellään vasta suunnitteluvaiheessa. Käyttöliittymää ei siis tässä vaiheessa vielä kiinnitetä, joskin joitain periaatteita saattaa olla tarpeen hahmotella, jotta käyttäjät ymmärtäisivät, mistä on kyse. Käyttötapaus kuvataan esittämällä vuoropuhelun peruskulku. Mahdolliset käyttötapauksen kulkua muuttavat poikkeustilanteet voidaan määritellä erillisinä käyttötapausta täydentävinä käyttötapauksina. Käyttötapaukseen voi liittyä vaatimuksia koskien suojausta, vasteaikoja, yms.

Esimerkkejä ilmoittautumisjärjestelmään liittyen:

Kurssille Ilmoittautuminen

- Suorittajana opiskelija

- Opiskelija antaa tunnistustietonsa ja valitsee kurssin sekä kurssin harjoitusryhmän. Opiskelija saa tiedon ilmoittautumisen onnistumisesta.
- Opiskelija ei voi ilmoittautua täynnä olevaan ryhmään
- Opiskelija ei voi ilmoittautua, jos hänelle on kirjattu osallistumiseste.
- 2 ruuhkahuippua vuodessa, n 1400 ilmoittautumista tunnissa, muulloin vähän
- Esimerkkitapaus: *NN ilmoittautuu JSS/s99 kurssin harjoitusryhmään 3.*

Ilmoittautumisen peruminen

- Suorittajana opiskelija
- Opiskelija antaa tunnistetietonsa ja valitsee ilmoittautumisen, jonka hän haluaa perua.
- Järjestelmä ilmoittaa operaation suorituksesta
- Esimerkkitapaus: *NN peruu ilmoittautumisensa JSS/s99 kurssin harjoitusryhmään 3*

Esimerkkejä lipunvälitysjärjestelmään liittyen

Elokuvan näytösaikojen ja paikkojen selvitys

- Suorittajana asiakas
- Asiakas valitsee elokuvan sekä aikavälin alku- ja loppupäivät ja aikarajat näytöksen alkamisajalle. Lisäksi asiakas voi määritellä näytetäänkö tulos teattereittain vaiko alkamisaikojen perusteella järjestettynä.
- Tuloksena asiakas saa luettelon näytöksistä määrittelemässään järjestyksessä
- Esimerkkitapaus: *NN haluaa saada selville tänään välillä 17-22 alkavien Star Wars Episode 1 näytösten ajat ja paikat Helsingissä*

Lippujen varaus näytökseen

- Suorittajana asiakas
- Asiakas valitsee näytöksen ja saa tiedot vapaina olevista paikoista. Hän valitsee näistä mieleisensä (yhden tai useampia) ja antaa tunnistetietonsa.

Järjestelmä vahvistaa varauksen antamalla asiakkaalle varausnumeron sekä tuottamalla laskun, jonka asiakas maksaa pankkijärjestelmänsä avulla. Lasku jää odottamaan maksamista. Kun asiakas on maksanut laskun hän tulostaa itselleen liput.

- Esimerkkitapaus: *Kalle Kenkkunen varaa paikan 308 Tennispalatsi 12:n näytökseen 20.11.1999 klo 21. Hän saa varausnumeron 1234567 ja laskun jonka hän maksaa Meritan Solo järjestelmän avulla. Hän tulostaa liput omalla mustesuihkukirjoittimellaan.*

Laajoissa järjestelmissä voidaan lähteä liikkeelle käyttäjien työtehtäviin perustuvista käyttötapauksista. Käyttötapauksia analysoitaessa niistä saatetaan löytää yhteisiä osia, jotka voidaan erottaa omiksi käyttötapauksiksi. Samoin erilaiset virhe- ja poikkeustilanteet sekä vaihtoehtoiset kulut käyttötapauksessa olisi syytä erottaa omiksi erillisiksi käyttötapauksiksi. Näin syntyy riippuvuuksia käyttötapausten välille. Jacobson on määritellyt riippuvuuksien esittämiseen soveltuvan kaaviotekniikan. Tässä tekniikassa on kahden tyyppisiä riippuvuuksia laajennoksia ja hyväksikäyttöä.

Esimerkiksi käyttötapaukseen *paikkojen varaus näytökseen* voisi liittyä seuraavia poikkeuksia ja virhetilanteita:

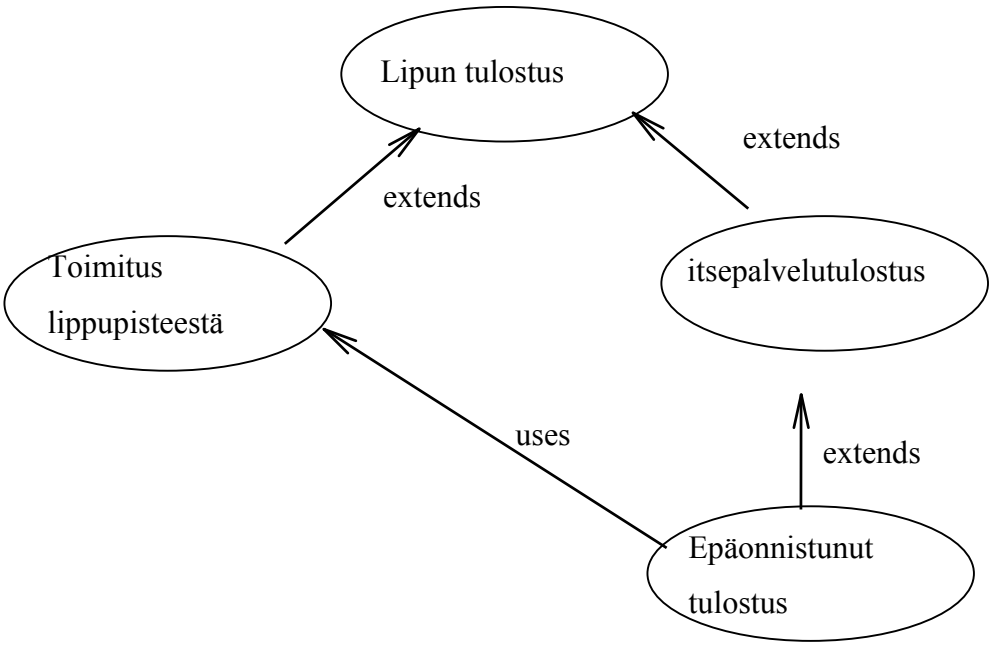
- Valittuun näytökseen ei ole haluttua määrää vapaita paikkoja
- Asiakas ei ole rekisteröitynyt
- Asiakas on unohtanut tunnistetietonsa
- Asiakkaalla ei ole käyttöoikeutta pankkitietojärjestelmään
- Asiakkaalla ei ole kirjoitinta
- Yhteys järjestelmään katkeaa (eri vaiheissa).

Laajennoksilla (extent) tarkoitetaan käyttötapauksia, joissa kommunikoinnin sisältö poikkeaa perussisällöstä. Esimerkiksi ilmoittautumisen laajennos olisi ensikertaa ilmoittautuminen, jossa opiskelijan on annettava kaikki henkilötietonsa. ‘Paikkojen varaus näytökseen’ –käyttötapauksen peruskulussa asiakas tulostaa lipun omalla kirjoittimellaan. Asiakkaalla ei kuitenkaan välttämättä ole kirjoitinta tai hän ei saa lippua tulostettua. Tällöin käyttötapauksen viimeiselle vaiheelle lipun toimitukselle tarvitaan vaihtoehtoinen ratkaisu. Toimitustavoiksi voitaisiinkin määritellä

- *itsepalvelutulostus*

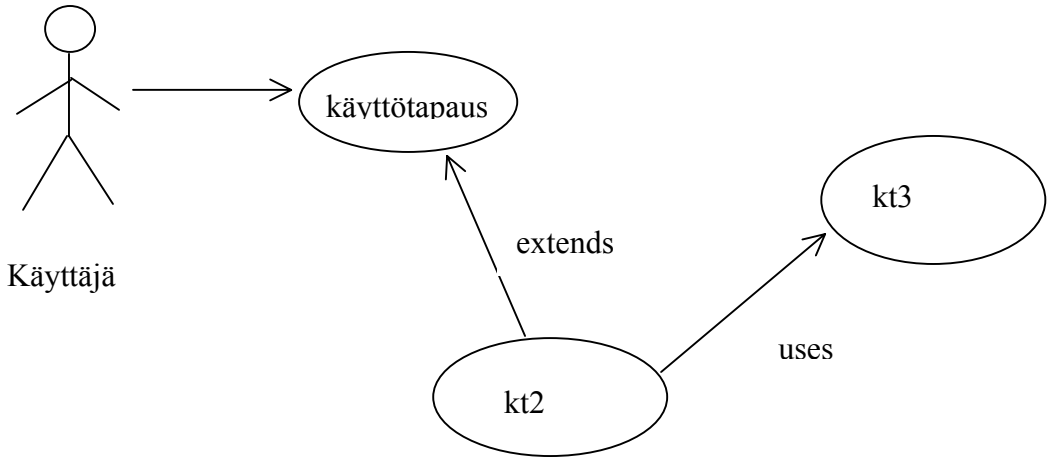
  Lippu kirjataan noudettavaksi teatterin lippupisteestä. Asiakkaan on lippua noutaessaan annettava varausnumeronsa. Kun asiakas on noutanut lipun se kirjataan noudetuksi.
- toimitus lippupisteestä.

  Järjestelmä selvittää, tarvittaessa asiakkaalta kysymällä, millainen kirjoitin asiakkaalla on. Jos kirjoitin on tulostukseen soveltuva muodostetaan lippujen kuvat ja toimitetaan ne asiakkaan työasemaan. Asiakas tulostaa liput ja kuittaa tulostuksen onnistuneeksi.



Kuva 3.2: Käyttötapausten riippuvuuksia.

Kuvassa 3.2 on UML-tekniikalla kuvattu käyttötapausten riippuvuuksia. Siinä on edellämainittujen käyttötapausten lisäksi määritelty poikkeustilanne epäonnistunut tulostus, joka aiheuttaa lipun toimituksen lippupisteestä. Tämä käyttötapaus käyttää hyväkseen toista käyttötapausta. Kuvassa 3.3 on esitetty UML:n käyttötapauskaavion symbolit. Käyttötapausten tekstimuotoinen kuvaus esimerkiksi taulukkomuodossa on kuitenkin oleellisempaa kuin graafinen käyttötapauskaavio, jonka informaatiosisältö on usein varsin vähäinen.



Kuva 3.3: UML-käyttötapauskaavion symbolit.

Käyttäjä (actor) kuvaa käyttäjän roolia. Esimerkiksi henkilö voisi olla roolissa opettaja tai opiskelija. Nuoli käyttäjästä käyttötapaukseen kuvaa sitä, että kyseisessä käyttäjäroolissa oleva käyttäjä käyttää kyseistä käyttötapausta. Laajentaminen ja hyväksikäyttö määriteltiin yllä.

Käyttötapausten avulla kuvataan kaikki järjestelmältä edellytettävät palvelut eli järjestelmän toiminnalliset vaatimukset. Käyttötapauksilla on hyvin keskeinen rooli ohjelmiston kehitysprosessin eri vaiheissa, sillä toiminnallisten vaatimusten määrittelyn lisäksi järjestelmä suunnitellaan ja toteutetaan usein niiden ohjaamina. Käyttötapauksiin liittyvät esimerkkitapaukset voivat toimia järjestelmän testitapauksina.

Edellä on tarkasteltu käyttötapauksen sisällön määrittelyä. Käyttötapauksen toteutuksen suunnittelussa on perustana käyttöliittymän yleissuunnitelma. Sen pohjalta suunnitellaan yksityiskohtaisesti käyttötapauksen läpivienti käyttöliittymän avulla ja määritellään käyttöliittymäoliot ja niille käyttötapauksen tarvitsemia palveluita.

Käyttötapausten esittämisestä graafisesti on hyötyä lähinnä silloin kun on esitettävä käyttötapausten välisiä riippuvuuksia (laajennoksia ja hyväksikäyttöä). Jos näitä ei kuvata riittää käyttötapausten esittämiseen luettelo tai taulukko. Luettelossa käyttötapaukset olisi hyvä ryhmitellä käyttäjäkohtaisiin ryhmiin, esim. asiakkaan käyttötapaukset, virkailijan käyttötapaukset, johdon käyttötapaukset.