Ohjelmistoprojektin työmäärän arviointimallit

Tiina Kirmanen

26.03.2002

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Ohjelmistoprojektin työmäärän arviointi on vaikeaa, koska työmäärän vaikuttavia tekijöitä on paljon, eikä kaikkien näiden tekijöiden ja työmäärän välistä riippuvuutta usein tunneta riittävän hyvin. Työmääräarvioiden tarkkuuden parantaminen on kiinnostanut sekä tutkijoita että tietojärjestelmien rakentajia pitkään ja arviointimallien mahdollisuuksia tuottaa tarkempia arvioita on tutkittu jo usean vuosikymmenen ajan. Tässä tutkielmassa esitellään tekniikoita, joita on käytetty arviointimallien rakentamiseen. Pisimpään tutkitut mallit ovat perustuneet tilastollisiin menetelmiin, yleensä regressiotekniikkaan. Koska näiden mallien tuottaman arvioinnin tarkkuus on ollut vaatimaton, on viime vuosina tutkittu, voitaisiinko ohjelmistotuotannon dynamiikkaa mallintaa paremmin tekoäly-tekniikoilla. Tutkielmassa esitellään tilastolliseen regressiotekniikkaan perustuva regressiomalli ja kaksi tekoälytekniikkaan perustuvaa mallia: analogia, joka perustuu tapauskohtaiseen päättelyyn sekä induktiiviseen päättelyyn perustuva regressiopuu. Tutkielmassa esitellään joitain viime vuosina tehtyjä tutkimuksia. Niiden tulosten perusteella ei voida esittää, että mikään arviointitekniikka olisi toista tarkempi. Eri tekniikat soveltuvat eri tilanteisiin riippuen siitä millainen kokemushistoria on käytettävissä. Mallin tarkkuuteen näyttäisi tekniikkaa enemmän vaikuttavan rakentamiseen käytetyn aineiston määrä ja laatu. Yritysten tulisi kehittää työmäärän arviointia prosessiksi, jossa omien toteutuneiden projektien ja yrityksen tilanteen kannalta merkittävien työmäärään vaikuttavien attribuuttien perusteella kehitettäisiin omaa arviointimallia, jotta ymmärrettäisiin oma tuotantoprosessi paremmin ja päästäisiin tarkempiin arvioihin.

Avainsanat: työmääräarviointi, regressiopuu, analogia, regressioanalyysi

Sisältö

1 Johdanto **1**

1.1 Arviointiprosessi 1

1.2 Tutkielman rakenne 4

2 Työmääräarvioinnin elementit **5**

2.1 Ohjelmiston koko 6

2.2 Uudelleenkäyttö 9

2.3 Projektin olosuhteet 10

2.4 Kokemushistoria 13

3 Arviointitekniikoita **15**

3.1 Arviointitekniikoiden luokittelu 15

3.2 Regressiomalli 17

3.2.1 Regressiomallin periaate 17

3.2.2 Regressiomallin soveltaminen 18

3.3 Analogia 21

3.3.1 Analogian periaate 21

3.3.2 Analogian soveltaminen 23

3.4 Regressiopuu 27

3.4.1 Regressiopuun periaate 27

3.4.2 Regressiopuun soveltaminen 32

3.5 Mallin hyvyyden arviointi 37

3.6 Arviointitekniikoista tehtyjä tutkimuksia 42

4 Arvioinnin epävarmuus ja riskienhallinta **49**

4.1 Riskienhallinta 49

4.2 Arvioinnin epävarmuustekijät ja niiden huomioiminen arvioinnissa 51

5 Yhteenveto **56**

Viitteet **58**

Liite 1: Esimerkkiaineisto **61**

Liite 2: Regressiopuumallin 2 säännöt **62**

Liite 3: Regressiopuumallin 3 säännöt **63**

1 Johdanto

Ohjelmistoprojektin työmääräarviota tarvitaan koko projektin elinkaaren ajan. Projektin alkuvaiheessa tarvitaan alustava arvio tarjouksen hinnoitteluun tai projektin hyötyjen ja kustannusten vertailuun. Projektin aikana tarvitaan tarkempi työmääräarvio projektin suunnittelua varten. Koko projektin ajan arvioitua työmäärää verrataan toteutuneeseen työmäärään. Osa projektin hallintaa on suorittaa määräajoin työmäärän uudelleenarviointi ja korjata tarvittaessa projektin resurssi- ja aikataulusuunnitelmia (Fenton ja Pfleeger 1997).

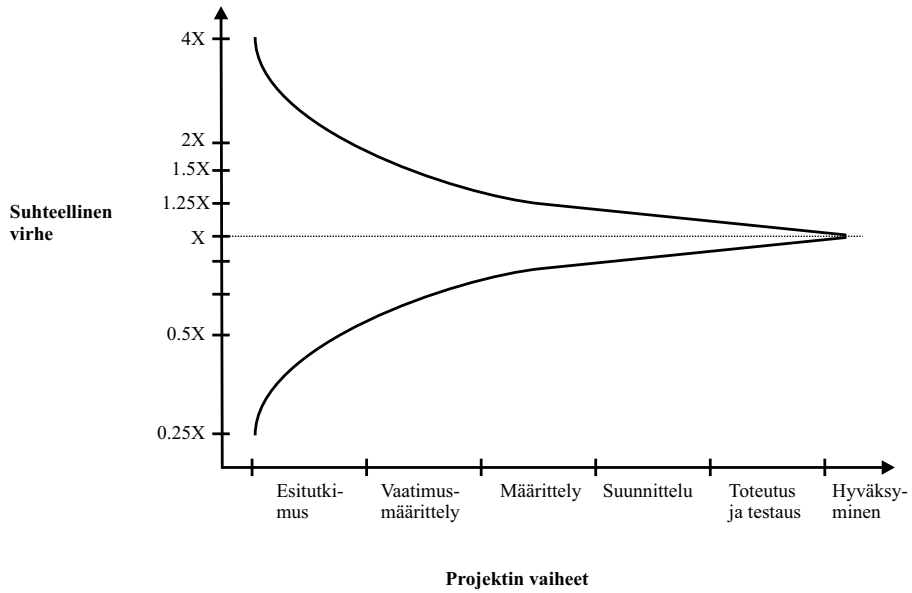
Työmäärän luotettava arviointi on vaikeaa, sillä ohjelmistoprojektiin liittyy erityispiirteitä, joita ei monilla muilla aloilla tavata, ohjelmistoprojektit ovat harvoin toistensa kaltaisia, käytössä oleva teknologia on nuorta ja työkalut muuttuvat nopeaan tahtiin, ohjelmistokehittäjät joutuvat usein ratkaisemaan ongelmia, jollaisia eivät ole aikaisemmin ratkaisseet, ohjelmistotuote on abstrakti ja kompleksinen ja sitä on vaikea mitata, ohjelmistokehitysprosessi on luonteeltaan luovaa työtä, jonka tuottavuutta on vaikea ennakoida (Fenton ja Pfleeger 1997, Jones 1998).

Kuvassa 1 on esitetty kuinka arviointi Boehmin (1981) mukaan tarkentuu projektin elinkaaren aikana. Vaikeinta työmäärän arviointi on projektin alkuvaiheessa, jolloin ohjelmistotuotteesta ja projektin olosuhteista tiedetään vielä vähän. Projektin edetessä ohjelmiston määrittely tarkentuu, jolloin ohjelmiston koko voidaan arvioida tarkemmin ja pystytään paremmin arvioimaan projektin olosuhteita.

Jos arviointi suoritetaan huolimattomasti eikä sitä tarkenneta projektin edetessä, on projekti vaarassa ajautua aikataulun ja kustannusten ylityksiin ja usein aikatauluongelmista seuraa myös laatuongelmia.

1.1 Arviointiprosessi

Fentonin ja Pfleegerin (1998) mukaan työmäärän arvioinnin tavoitteena ei ole pelkästään tuottaa projektille arvio tarvittavasta työmäärästä, vaan tavoitteena tulisi olla prosessi, jolla tuotetaan tarkkoja arvioita. Arviointiprosessi on jatkuvaa arvioinnin kehittämistä, joka vaatii perustakseen arviointimallin, joka kuvaa parametrien avulla ohjelmistotuotantoprosessia. Tutkimalla mallin tuottamia arvioita, saadaan lisää tietoa parametrien käyttäytymisestä ja pystytään säätelemään



Kuva 1: Arvioinnin tarkkuus projektin elinkaaren aikana (Boehm 1981).

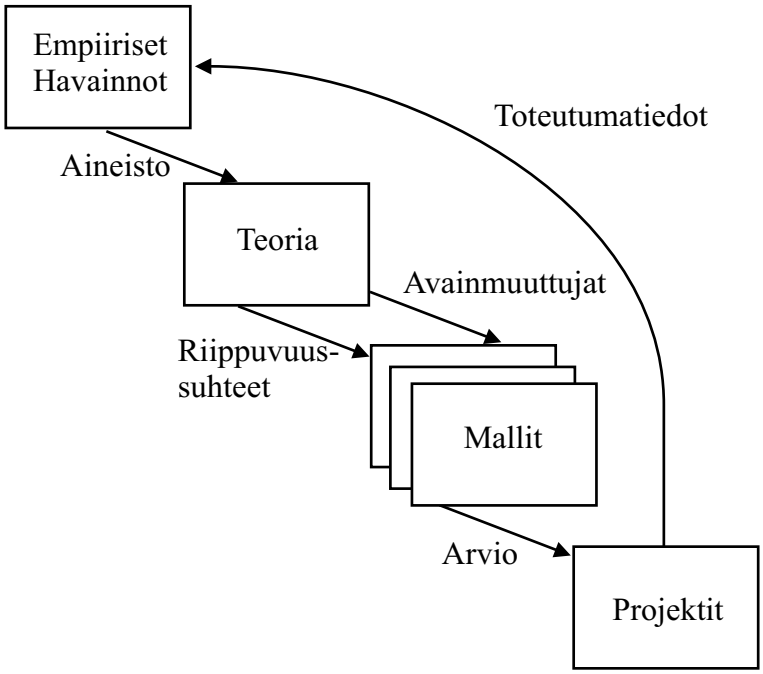
mallia ja näin voidaan tavoitella parempaa arvioinnin tarkkuutta tulevissa projekteissa.

Ohjelmistoprojektin elinkaaren aikana arvioidaan työmäärän lisäksi lukuisia muitakin asioita kuten vaatimusten muuttumista, tarkastusmenettelyissä löytyneiden virheiden määrää, järjestelmän monimutkaisuutta, uudelleenkäyttöä jne. Nämä kaikki voidaan nähdä samoja perusperiaatteita noudattavina kuvan 2 mukaisina prosesseina (Fenton ja Pfleeger 1997).

Arviointiprosessin perustana ovat teoriat ja mallit. Toteutumatiedoista koottu kokemuskanta muodostaa empiiriset havainnot, joiden perusteella voidaan löytää työmäärään vaikuttavia avainmuuttujia. Havaintoja analysoimalla voidaan muodostaa teoria, jonka pohjalta voidaan tutkia, mitkä ovat työmäärään vaikuttavien tekijöiden keskinäiset riippuvuudet ja niiden vaikutus työmäärään. Tämän jälkeen voidaan rakentaa malli, joka kuvaa näitä riippuvuussuhteita (Fenton ja Pfleeger 1997). *Arviointimallilla* tarkoitetaan algoritmia, joka tietyillä syöteparametreillä, kuten projektin koko ja odotettavissa oleva tuottavuustaso, arvioi tarvittavan työmäärän. *Arviointitekniikalla* tarkoitetaan menetelmää, jolla malli muodostetaan (Hughes 2000).

Arviointimalli ei ole staattinen, vaan se muuttuu ja tarkentuu sitä mukaa, kun tietämys ohjelmistotuotantoprosessista, tehtävistä ja tuotteista lisääntyy projektien toteutumatietojen kautta.

Työmäärän arvioinnin luotettavuuden parantaminen on pitkään kiinnostanut sekä tietojärjestel-



Kuva 2: Arviointiprosessin yleinen malli (Fenton ja Pfleeger 1997).

mien rakentajia että tutkijoita. Erilaisten tekniikoiden sopivuutta on tutkittu paljon 1970-luvulta lähtien. Sekä tutkijat, että ammattilaiset ovat kehittäneet erilaisia tekniikoita, joiden avulla voidaan tunnistaa työmäärän ja järjestelmän ominaisuuksien, projektin tilanteen, henkilöstötekijöiden ja muiden työmäärään vaikuttavien tekijöiden välisiä riippuvuuksia. Yleisimmät arviointitekniikat tuottavat arvion tilastollisin menetelmin hyödyntäen toteutuneista projekteista kerättyä historiatietoa.

Hyväkään arviointimalli ei takaa hyvää tulosta ilman huolellisesti tehtyä arviointityötä. Luotettavan arvioinnin saamiseksi ei riitä, että käytettävälle mallille annetaan syötteenä sen vaatimat parametrit ja hyväksytään mallin antama tulos, vaan arviointi tulisi toteuttaa kuten projekti: se on suunniteltava ja katselmoitava ja sen toteutumista on seurattava projektin kuluessa (Boehm 1981).

1.2 Tutkielman rakenne

Työmääräarvion peruselementteinä voidaan pitää järjestelmän kokoa, projektin olosuhteita, uudelleenkäytön astetta ja kokemushistoriaa. Luvussa 2 esitellään näiden tekijöiden merkitystä työmäärän arviointiprosessissa.

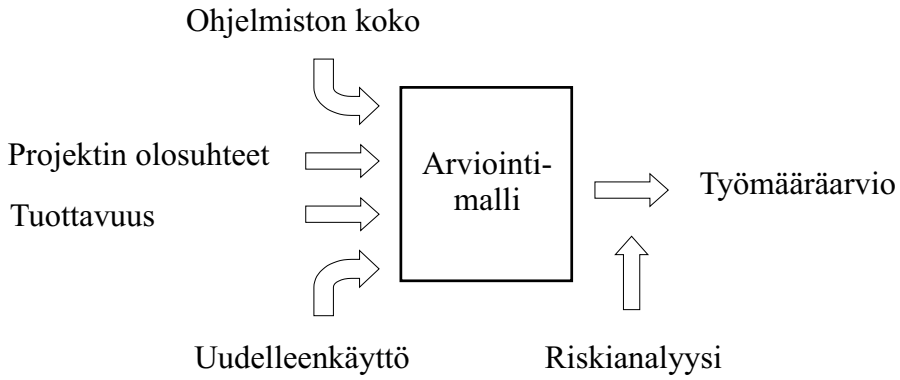
Työmäärän arviointi voi tapahtua manuaalisesti asiantuntija-arviona tai automaattisesti arviointityökalun avulla. Markkinoilla on useita kymmeniä ohjelmistoprojektin työmäärän arviointiin tarkoitettua työkalua. Nämä työkalut perustuvat arviointimalliin, joka on rakennettu hyödyntäen jotain ohjelmistotuotantoprosessin dynamiikkaa kuvaamaan sopivaa tekniikkaa. Luvussa 3 tarkastellaan millaisia tekniikoita on tutkittu työmäärän arviointimallien rakentamisen yhteydessä. Tarkemmin esitellään kolme erilaista arviointitekniikkaa: regressiotekniikka, analogia ja regressiopuu. Luvussa esitellään kuinka mallin tarkkuutta voidaan arvioida ja eräitä viime vuosina tehtyjä tutkimuksia, joissa on vertailtu eri tekniikoilla rakennettuja malleja.

Arviointiin liittyy aina epävarmuutta, josta voi aiheutua riskejä. Riskienhallinnalla voidaan varautua mahdollisiin riskeihin ja pienentää niistä aiheutuvia haittoja. Luvussa 4 tarkastellaan arviointiin liittyviä epävarmuustekijöitä ja kuinka niistä aiheutuvia riskejä voidaan hallita riskienhallinnan keinoin.

Luvussa 5 on yhteenveto arviointimallien rakentamiseen soveltuvista tekniikoista ja arviointitekniikoista tehtyjen tutkimusten tuloksista.

2 Työmääräarvioinnin elementit

Työmäärän arviointi perustuu ennakoituun tuottoasteeseen. Tuottoasteen ennustaminen edellyttää empiiristä tietoa vastaavanlaisista toteutuneista hankkeista. Toteutumatiedot kerätään kokemushistoriaan projektien jälkilaskennan kautta. Jotta työmäärä saadaan arvioitua ennakoidun tuottoasteen avulla, on projektille määriteltävä sen laajuus. Jonesin (1998) mukaan suurin työmäärään vaikuttava tekijä on ohjelmiston koko. Ohjelmiston koolla ei kuitenkaan yksistään kyetä kuvaamaan projektin laajuutta, vaan on huomioitava myös lukuisia muita attribuutteja. Uudelleenkäyttö joko lisää tai vähentää projektin tuottoastetta riippuen siitä, tuottaako projekti uudelleen käytettäviä artefakteja muille projekteille vai hyödyntääkö projekti uudelleenkäytettäviä artefakteja. Projektiin liittyy lisäksi suuri joukko projektiin, lopputuotteeseen, kehitysprosessiin ja henkilöstöön liittyviä olosuhdetekijöitä, jotka vaikuttavat työn tuottoasteeseen. Kuvassa 3 on esitetty edellä mainitut työmäärän arvioinnissa huomioitavat keskeiset elementit .



Kuva 3: Työmääräarvioinnin elementit.

Jos ohjelmistotuotantoprosessista on olemassa mittaustietoja, voidaan rakentaa malli, jolla pyritään selittämään erilaisia ohjelmistotuotantoprosessiin liittyviä ilmiöitä. Conten et al. (1986) mukaan ohjelmistotuotantoprosessin yleinen malli voidaan esittää seuraavassa muodossa:

$$y = f(x_1, x_2, ..., x_n) \tag{1}$$

missä riippuva muuttuja *y* on mittari, kuten työmäärä, kustannukset tai laatu ja x_1, x_2, ... x_n ovat riippumattomia muuttujia, jotka liittyvät joko ohjelmistotuotantoprosessiin tai lopputuotteeseen.

Työmäärää voidaan mallintaa soveltamalla Conten et al. (1986) yleistä mallia ja käyttäen kuvan 3 elementtejä, jolloin malli saa seuraavanlaisen muodon:

$$E = f(S, R, T, D, f) \tag{2}$$

missä *E* on työmäärä, *S* ohjelmiston koko, *R* uudelleenkäytön aste, *T* projektin tilannetekijät, *D* tuottoaste ja *f* riskien vaikutus.

Tässä luvussa käsitellään ohjelmiston koon määrittämistä ja uudelleenkäytön vaikutusta projektin laajuuteen ja keskeisiä tuottavuuteen vaikuttavia projektin tilannetekijöitä sekä kokemushistorian keräämistä tuottavuuden arviointia varten. Työmääräarvioinnin epävarmuuden ja riskien huomioimista käsitellään luvussa 4.

2.1 Ohjelmiston koko

Ohjelmiston koko on keskeisin ohjelmistotuotannon tuotemittareista. Fentonin (1991) mukaan ohjelmiston koon mittari olisi määriteltävä siten, että se huomioi ohjelmiston monimutkaisuuden ja kuvaisi ohjelmistotuotteen todellista toiminnallisuutta. Yhden attribuutin sijasta, koko voidaankin ymmärtää kolmen attribuutin: ohjelmakoodin pituuden, toiminnallisuuden ja sovellusalueen monimutkaisuuden yhdistelmänä.

Ohjelmiston koon määrittely on käytännössä osoittautunut vaikeaksi erityisesti ohjelmiston elinkaaren aikaisessa vaiheessa. Koko voidaan määritellä vaatimusmäärittelyn perusteella, mutta edellytyksenä on, että määrittely on tehty riittävän yksityiskohtaisella tasolla (Fenton ja Pfleeger 1997).

Nykyisin ohjelmiston koon arviointiin käytetään tilanteesta riippuen eri mittausmenetelmiä (Forselius 1999):

- lähdeohjelmarivien laskenta
- lähdeohjelman lausemäärien laskenta
- toimintopistelaskenta

- dokumenttisivujen laskenta
- takaisinlaskenta.

Lähdeohjelmakoodin rivien lukumäärä on kenties edelleen yleisin sovelluksen koon mittari. Rivilukumäärien laskenta voidaan ymmärtää useilla tavoilla. Joidenkin käytäntöjen mukaan esimerkiksi kommenttirivit lasketaan mukaan, joidenkin ei. Fenton (1991) määrittelee ohjelmakoodiriviksi kaiken ohjelmakooditekstin, joka ei ole kommenttia tai tyhjä rivi. Tällöin ei oteta huomioon sitä, kuinka monta ohjelmointikielen loogista lausetta tai loogisen lauseen osaa rivi sisältää.

Lähdeohjelmakoodin rivien laskentaan sisältyy kuitenkin joitain ongelmia, eikä se sen vuoksi ole aina kovin luotettava mittari. Conten et al. (1986) mukaan rivien lukumäärä ei sovellu tuottavuuden tai työmäärän arviointiin, koska joidenkin rivien tuottaminen on huomattavasti vaikeampaa kuin toisten. Forseliuksen (1999) mukaan eri ohjelmointikielillä toteutetut järjestelmät eivät rivimäärällä mitattuna ole toistensa kanssa vertailukelpoisia. Eri organisaatioiden ohjelmointityylit voivat poiketa ja myös ohjelmoijilla samassa organisaatiossakin voi olla erilainen tyyli kirjoittaa ohjelmia. Näin edes samalla ohjelmointikielellä toteutettujen ohjelmien kokoa ei voida mitata luotettavasti rivimäärällä. Rivien lukumäärän arvioiminen projektin aikaisessa vaiheessa on vaikeaa ja luotettava arvo saadaankin vasta ohjelmiston valmistuttua.

Lähdeohjelman loogisten lauseiden laskeminen on sukua lähdeohjelmarivien laskennalle. Siinä laskenta tehdään etsimällä koodista ohjelmakielen varattujen suorituskäskyjä (Forselius 1999). Ero lähdeohjelmakoodin rivien laskentaan on siinä, että yksi suoristuskäsky saattaa olla useita koodirivejä ja päinvastoin yhdellä koodirivillä on mahdollista esittää useita lauseita. Lähdeohjelmakoodin rivien laskemisen etuna on, että mittari voidaan määritellä kieliriippumattomasti, mutta ohjelman lauseiden laskeminen sen sijaan on selkeästi kieliriippuvainen, riippuen siitä kuinka lause kullakin kielellä määritellään (Conte et al. 1986). Lauseiden laskemisessa on samat ongelmat ja edut kuin lähdeohjelmakoodin rivien laskennassa. Molempien laskenta on helposti automatisoitavissa, mutta tulos on saatavissa vasta, kun ohjelma on valmis (Sallis et al. 1995).

Toimintopistelaskennassa (Forselius 1999) kehityshankkeen koko lasketaan järjestelmän toiminnallisesta näkökulmasta. Toiminnallisuus järjestelmässä näkyy niinä loogisina käyttäjätoimintoina, joita systeemiltä edellytetään. Laskennan käyttäjälähtöisyyden vuoksi menetelmä on riippumaton toteutustekniikoista ja sovellusalueesta.

Toimintopistelaskennan ensimmäisenä vaiheena on järjestelmän toiminnallisten osien tunnistaminen. Toiminnallisia osia ovat sisäiset loogiset tiedostot, ulkoiset liitäntätiedostot, syötteet, tulosteet ja kyselyt. Kunkin toiminnallisen osan toimintopistemäärä määräytyy sen monimutkaisuuden perusteella. Monimutkaisuuden määrittely tapahtuu tietoalkiotyyppien lukumäärän ja viitattujen tiedostojen tai tiedostojen tapauksessa tietuealkiotyyppien perusteella (Garmus ja Herron 2000).

Jonesin (1998) mukaan toimintopistelaskennasta on olemassa useita kymmeniä eri versioita. Perusversiossa huomioidaan edellä mainitut viisi toiminnallista osaa. Joissain versioissa järjestelmän toiminnallisen koon laskentaa on tarkennettu ottamalla monimutkainen käsittely mukaan erillisenä toiminnallisen osan tyyppinä.

Toimintopistelaskentaa on arvosteltu siitä, että se soveltuu lähinnä hallinnollisten järjestelmien laskentaan. COSMIC-ryhmä on kehittänyt toimintopistelaskennasta COSMIC-FFP-menetelmän, jossa on huomioitu myös reaaliaikaisten valvontajärjestelmien vaatimukset koon laskennalle (Abran et al. 2001).

Dokumenttisivujen laskentaa voidaan käyttää yrityksissä, joissa kuvausstandardit ovat tarkkaan määritellyt, jolloin ohjelmiston kuvaus kertoo myös sen laajuuden (Forselius 1999).

Taulukko 1: Ohjelmistoon koon arviointimenetelmien soveltuminen eri tyyppisiin projekteihin (Forselius 1999).

Projektityyppi	*Rivi-määrät*	*Lause-määrät*	*Toiminto-pisteet*	*Takaisin-laskenta*	*Dokumenttien sivumäärät*
Uuskehitysprojektit	–	–	++	–	–
Lisäävä ylläpito	–	–	++	–	–
Muuttava ylläpito	+	++	+	++	+
Konversioprojektit	o	oo	+	++	o
Vuotuinen kunnossapito	o	oo	+	++	+
Valmisohjelmiston hankinta	–	–	+	–	o
Ulkoistamisprojektit	o	o	+	++	o

Takaisinlaskenta (backfiring) on yhdistelmä toimintopistelaskennasta ja lähdeohjelman lausemäärien laskennasta. Menetelmä perustuu konversiotauluihin, jotka on koottu kokemustietojen

perusteella. Konversiotaulun avulla toimintopisteet voidaan muuttaa lähdekoodiriveiksi tai lähdekoodirivit toimintopisteiksi (Forselius 1999). Takaisinlaskentaa voidaan tarvita esim. jos halutaan käyttää jotain lähdekoodiriveihin perustuvaa työmääränarviointimenetelmää kuten COCOMO (Boehm 1981).

Erityyppisiin projekteihin soveltuvat erilaiset menetelmät. Taulukossa 1 on esitetty eri laskentamenetelmien soveltuminen erilaisiin hankkeisiin. Taulukossa on käytetty seuraavia merkintöjä (Forselius 1999):

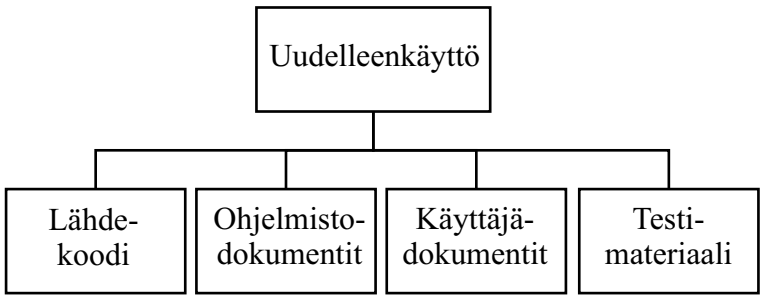
- ++ tarkoittaa, että menetelmä sopii erinomaisesti
- + tarkoittaa, että menetelmää voidaan käyttää
- o tarkoittaa, että menetelmää voidaan käyttää rajatuissa tapauksissa
- - tarkoittaa, että menetelmä ei sovellu käytettäväksi.

2.2 Uudelleenkäyttö

Uudelleenkäytöllä pyritään tuottavuuden ja laadun parantamiseen läpi koko ohjelmistotuotantoprosessin. Tuottavuutta voidaan lisätä paitsi lyhentämällä ohjelmointiin kulunutta aikaa, myös vähentämällä testaukseen ja dokumentointiin käytettyä aikaa. Uudelleenkäyttöä arvioitaessa on lähdekoodin lisäksi huomioitava kaikki uudelleenkäytettävät artefaktit, joita ovat ainakin kuvassa 4 esitetyt määrittelyn ja suunnittelun dokumentaatio, lähdekoodi, testimateriaali ja käyttäjäohjeet (Jones 1998, Pfleeger 1998).

Pfleegerin (1998) mukaan määrittelyjen ja suunnittelun uudelleenkäyttö on tuottavuuden kannalta merkittävintä, sillä esimerkiksi suunnittelun uudelleenkäyttö johtaa automaattisesti ohjelmakoodin uudelleenkäyttöön. Komponenttien uudelleenkäytön seurauksena ohjelmiston luotettavuus lisääntyy.

Jones (1998) pitää uudelleenkäyttöä tuottavuuteen eniten vaikuttavana tekijänä. Jonesin mukaan laadukkaan materiaalin uudelleenkäyttö voi tehostaa ohjelmistoprojektin tuottavuutta jopa 350 prosentilla. Jotta näin suureen tuottavuuden nousuun päästäisiin, on Jonesin mukaan tuotteessa oltava vähintään 75 % uudelleenkäytettävää materiaalia ja materiaalin on oltava virheetöntä.



Kuva 4: Uudelleenkäyttö.

Uudelleenkäytettävät artefaktit, joissa on paljon virheitä, vaikuttavat tuottavuuteen päinvastaisesti. Jonesin (1998) mukaan huonolaatuinen materiaali saattaa heikentää tuottavuutta 300 prosentilla ja huonosti onnistunut uudelleenkäyttö onkin merkittävin tuottavuutta heikentävä tekijä.

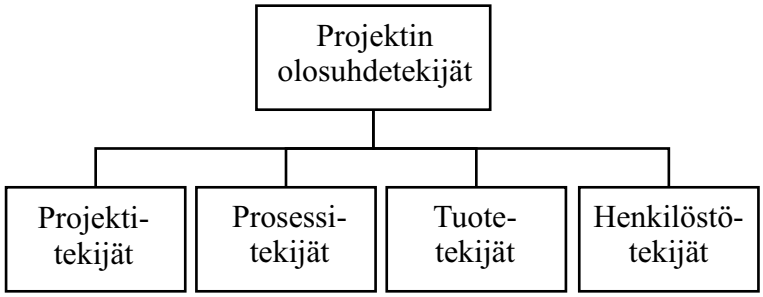
Lim (1994) on tutkinut uudelleenkäytön vaikutuksia tuottavuuteen, laatuun ja kustannuksiin Hewlett-Packardilla. Hän osoitti tutkimuksessaan uudelleenkäytön seurauksena tuottavuuden parantuneen 57 % ja virhetiheyden laskeneen puoleen. Uudelleenkäyttö ei kuitenkaan ole ilmaista, sillä uudelleenkäytettävän ohjelmakoodin tuottaminen vaatii enemmän aikaa kuin ohjelmakoodin, jossa uudelleenkäytettävyyttä ei tarvitse huomioida.

Työmäärää arvioitaessa uudelleenkäyttö huomioidaan projektin laajuudessa. Uudelleenkäytön arviointi ei ole aivan helppoa. Ensiksi on määriteltävä mikä lasketaan uudelleenkäytöksi ja toiseksi on oltava menetelmä, jolla se määrällistetään. Ohjelmakoodin uudelleenkäyttöä voi tapahtua monen tasoisena, ohjelmakoodi voidaan käyttää ilman muutoksia, pienin muutoksin tai siihen tarvitaan suuria muutoksia. Uudelleenkäytön aste voidaan ilmaista ordinaaliasteikollisena tai prosenttiosuutena. Uudelleenkäytettävän materiaalin käyttö voidaan huomioida esimerkiksi vähentämällä projektin laajuudesta kunkin toiminnon uudelleenkäytön astetta vastaava määrä toimintopisteitä (Fenton ja Pfleeger 1997).

2.3 Projektin olosuhteet

Olosuhdetekijät ovat sellaisia projektia luonnehtivia attribuutteja, jotka vaikuttavat projektin tuottavuuteen. Jonesin (1998) mukaan tällaisia tekijöitä on satoja. Olosuhdetekijät voidaan kuvan 5 mukaisesti ryhmitellä neljään pääryhmään: projektitekijät, prosessitekijät, tuotetekijät ja

henkilöstötekijät (STTF 2002).



Kuva 5: Projektin olosuhdetekijät.

Projektitekijät kuvaavat projektin erityispiirteitä kuten asiakkaan sitoutuminen, kehitysympäristön riittävyys, avainhenkilöiden saatavuus, projektiryhmän koko ja projektiryhmän rinnakkaisten projektien lukumäärä, asiakasorganisaatioiden lukumäärä ja aikataulupaineet (STTF 2002, Boehm 1981).

Asiakasorganisaation ja sen käyttäjien sitoutuminen projektiin on edullista projektin tuottavuudelle. Erityisen tärkeänä pidetään käyttäjien osallistumista toimintojen määrittelyyn. Esimerkiksi STTF:n (2002) ExperiencePro-tilanneanalyysissä asiakkaan sitoutuminen on keskimääräistä parempaa, jos asiakas on määritellyt ja hyväksynyt yli 70 % toimintojen määrittelyistä.

Projektiryhmän rakenteen ja ryhmän jäsenten henkilökohtaisten ominaisuuksien vaikutuksia työn tuottavuuteen ei ole vielä tutkittu paljon. Suuressa projektiryhmässä ryhmän kommunikaatio saattaa monimutkaistua. Jotkut projektiryhmät muodostavat hyvin toimivan tiimin ja jäsenet ovat saattaneet tehdä useita projekteja yhdessä, toisissa ryhmissä jäsenten välinen vuorovaikutus ei toimi. Voidaan olettaa, että eri tavoin toimivat ryhmät ovat erilaisia tuottavuuden suhteen (Fenton ja Pfleeger 1997).

Jos projektilla on paljon sidosryhmäriippuvuuksia, sen tuottavuus heikkenee. Sidosryhmien lukumäärä voi lisääntyä joko rinnakkaisten projektien kautta tai tilanteessa, jossa projekti on usean organisaation yhteishanke (STTF 2002).

Prosessitekijät kuvaavat ohjelmistotuotantoprosessin sisäistä systematiikkaa. Tällaisia tekijöitä ovat esimerkiksi standardien, menetelmien ja työkalujen käyttö sekä vaatimusten hallinnan taso. Clark (2000) on tutkinut prosessin kypsyyden vaikutusta tuottavuuteen. Tutkimuksessa organisaatiot on luokiteltu SEI:n määrittelemän prosessien kypsyystasoluokittelun (CMM) mukaisesti

ja tutkimustulosten perusteella yhden kypsyystason nousu voi tehostaa ohjelmistotuotantoa 4 - 11 %.

Tuotetekijät ovat yleisiä ohjelmistotuotetta kuvaavia tekijöitä. Tällaisia tekijöitä voivat olla esimerkiksi tuotteen luotettavuusvaatimukset, tietokannan koko ja sovelluksen monimutkaisuus (Boehm 1981). Ohjelmiston luotettavuusvaatimuksia voidaan arvioida sen mukaan, millaisen haitan järjestelmän heikentynyt käytettävyys aiheuttaa. Esimerkiksi, jos järjestelmä voi olla pitkiä aikoja pois käytöstä ilman taloudellista haittaa, on sen luotettavuusvaatimus hyvin pieni. Jos järjestelmässä oleva puute aiheuttaa vaaran ihmishengelle, on sen luotettavuusvaatimus erittäin suuri.

Sovelluksen monimutkaisuus on vaikeasti arvioitava tekijä. Sillä pyritään kuvaamaan sekä sovelluksen toiminnallista että teknistä monimutkaisuutta. Monimutkaisessa sovelluksessa on useita monimutkaisia algoritmeja tai poikkeuksellisen vaikeasti toteutettavaa tietokantaohjelmointia (STTF 2002).

Henkilöstötekijät kuvaavat projektiryhmän kokemustasoa. Jonesin (1998) mukaan henkilöstön kokeneisuus on uudelleenkäytön jälkeen merkittävin tuottavuuteen vaikuttava tekijä. Kokemustason tarkoituksenmukainen mittaaminen on kuitenkin vaikeaa (Fenton ja Pfleeger 1997). Kokemustason mittaamisessa on huomioitava sekä yksittäisen henkilön kokemustaso että projektiryhmän taso. Tuottavuuden kannalta merkittäviä kokemustekijöitä ovat esimerkiksi projektiryhmän suunnittelutaidot, sovellusaluetuntemus, työvälineiden tuntemus ja projekti- ja ryhmätyötaidot sekä henkilöstötekijöistä merkittävimpänä projektipäällikön kokeneisuus (Boehm 1981, Fenton ja Pfleeger 1997, Jones 1998).

Kokemustaso voidaan ilmaistaan esimerkiksi kokemusvuosina, kuukausina tai aikaisempien vastaavien projektien lukumääränä. Kokemustasoa kuvaavat mittarit ovat yleensä ordinaaliasteikollisia, jolloin projektiryhmän kokemus voidaan laskea projektiryhmän jäsenten kokemustasojen mediaanina tai keskiarvona ja luokitella sen perusteella vastaamaan mittaria (Fenton ja Pfleeger 1997).

Esimerkiksi COCOMO-malli määrittelee sovellusaluekokemuksen erittäin alhaiseksi, jos projektiryhmän keskimääräinen kokemus sovellusalueelta on alle neljä kuukautta. Vuoden keskimääräinen kokemus on luokiteltu alhaiseksi, kolmen vuoden keskimääräiseksi, kuuden vuoden kokemus korkeaksi ja yli 12 vuoden keskimääräinen kokemus erittäin korkeaksi (Boehm 1981).

2.4 Kokemushistoria

Kokemushistorialla tarkoitetaan jo päättyneistä projekteista kerättyä tietojoukkoa, joka koostuu projekteja luonnehtivista attribuuteista sekä toteutumatiedoista, kuten toteutunut työmäärä. Kokemushistoriaa tarvitaan, jotta arvioinnin pohjaksi saataisiin todellista tietoa työn tuottavuudesta. Kokemushistoriaa analysoimalla voidaan muodostaa arviointimalleja, joilla kuvataan projektia luonnehtivien attribuuttien ja työmäärän tai tuottavuuden välistä riippuvuutta.

Arviointimallin rakentamiseen voidaan käyttää yrityskohtaisia tai yritysten yhteisiä kokemuskantoja. Yrityskohtaisen kokemuskannan etuna on, että tiedonkeruuprosessia voidaan kontrolloida paremmin, projektia luonnehtivat mittarit ja mittaustavat ymmärretään paremmin, yritys voi sisäisesti helpommin räätälöidä muuttujia omiin tarkoituksiinsa sopiviksi ja yhtenäinen tietojoukko tuottaa luotettavamman lopputuloksen (Boehm 1981).

Yrityskohtaisen kokemushistorian kerääminen on kuitenkin hidasta, sillä usein yrityksissä valmistuu vain korkeintaan muutamia projekteja vuodessa ja monet arviointimallien rakentamiseen soveltuvat tekniikat vaativat suuren määrän projektitietoa riittävän luotettavan mallin rakentamiseen. Riittävän laajan kokemushistorian kerääminen yritystasolla voi kestää niin kauan, että kokemuskannan projektit ennättävät vanhentua teknologialtaan, eivätkä ole enää kelvollisia uusien projektien arviointiin. Nopeasti kehittyvällä alalla kokemushistoria vanhenee muutamassa vuodessa. Yrityksellä, joka on juuri siirtynyt uuden teknologian käyttöön tai, joka on juuri aloittanut toimintansa, ei ole käytettävissä omaa kokemushistoriaa (Angelis et al. 2001).

Arviointimalli voidaan rakentaa yrityksen ulkopuolisella tai yritysten yhteisellä kokemuskannalla ja kalibroida omiin olosuhteisiinsa sopivaksi omalla kokemushistorialla. Yritysten yhteisten tietokantojen ongelmana on usein epäyhtenäiset tiedonkeruumenetelmät ja heterogeeninen tietojoukko (Angelis et al. 2001).

Jos kokemuskannan tiedot ovat puutteellisia, ei niiden perusteella rakennettu arviointimallikaan voi tuottaa luotettavaa tulosta. Tämän vuoksi tiedonkeruussa on oltava huolellinen ja huomioitava tiedonkeruutapa, mietittävä mitä tietoja kerätään, kuinka paljon tietoja tarvitaan ja kuinka varmistetaan riittävä tiedon laatu.

Conten et al. (1986) mukaan kokemuskantaan kerättävien mittareiden on oltava yksinkertaisia, valideja, ohjaavia, analysoitavia ja vakaita. Yksinkertaisuuden vaatimus tarkoittaa, että mittarin

arvo on helposti ymmärrettävissä. Se ei koostu useista arvioista tai ole monimutkaisen algoritmin tuottama. Validiteetilla tarkoitetaan, että mittari todella kuvaa sitä mitä sen halutaan kuvaavan. Ohjaavat mittarit ovat sellaisia, joita voidaan mitata jo ohjelmistokehitysprosessin aikana ja voidaan näin käyttää ohjaamaan prosessia. Analysoitavuuden vaatimus tarkoittaa, että mittareita voidaan analysoida esimerkiksi tilastollisin menetelmin. Vakaat mittarit eivät ole herkkiä keinotekoiselle manipuloinnille, jolloin mittarin paraneminen esimerkiksi tuottavuuden suhteen ei merkitse todellista tuottavuuden paranemista.

Tiedonkeruu on toteutettava systemaattisesti käyttäen sovittuja sääntöjä. Kokemuskantaan tulisi kerätä vain sellaista tietoa, joka on objektiivisesti mitattavissa. Jos esimerkiksi käytetään luokittelevia muuttujia, on luokittelu määriteltävä niin tarkasti, että lopputulos ei riipu luokittelijasta (Conte et al. 1986).

Kokemuskanta koostuu eri mitta-asteikollisista muuttujista. Luokittelevia muuttujia on sekä nominaali- että ordinaaliasteikollisia. Nominaaliasteikollisia ovat esimerkiksi käyttäjäorganisaation tyyppi, sovellusaluetyyppi tai sovellusalustan tyyppi. Ordinaaliasteikollisia ovat esimerkiksi prosessin tai henkilöstön kyvykkyys. Ohjelmiston koko tai toteutunut työmäärä ovat suhdeasteikollisia. Luokittelevien muuttujien määrittelyssä on huomioitava niiden luokkien lukumäärä. Usein hyödyllisemmäksi saattaa osoittautua hienojakoinen luokittelu, jota voidaan käytössä ryhmitellä tarpeen mukaan (Angelis et al. 2001).

Kokemuskannan keruu on suunniteltava huolellisesti pitäen mielessä sen käyttötarkoitus. Kun arviointimallia ryhdytään rakentamaan, on analysoitava, mitkä ovat ne tekijät, jotka ovat merkittäviä tuottavuuden kannalta arvioivan yrityksen ohjelmistokehitysprosessissa, sillä tekijät vaihtelevat yrityskohtaisesti. Analysointi voidaan tehdä esimerkiksi faktorianalyysin avulla (Conte et al. 1986). Malliin kannattaa sisällyttää vain ne tekijät, jotka todella vaikuttavat tuottavuuteen. Lisäksi on huolehdittava, että malliin valitut attribuutit ovat toisistaan riippumattomia. Jos attribuuttien välillä havaitaan riippuvuutta, voidaan ne yhdistää tai valita malliin vain yksi näistä attribuuteista.

3 Arviointitekniikoita

Työmäärän arviointi voidaan tehdä joko asiantuntija-arviona tai arviointimallin avustamana. Asiantuntija-arvio on yleisesti käytetty työmäärän arviointimenetelmä, joka perustuu yhden tai useamman asiantuntijan aikaisempaan kokemukseen samantyyppisistä projekteista. Arvioijilla voi olla käytettävissä työkaluja ja menetelmiä, jotka avustavat arvion tekemisessä, mutta ensisijaisesti arvio perustuu kuitenkin asiantuntijoiden aikaisempaan kokemukseen (Boehm 1981). Arviointimalliin perustuva arviointi on asiantuntija-arviointia formaalimpi tapa. Arviointimallit perustuvat empiiriseen aineistoon, joka koostuu projekteja luonnehtivista työmäärään vaikuttavista attribuuteista.

Arviointimallin rakentamiseen soveltuvia tekniikoita on useita. Tässä luvussa esitellään yleinen luokittelu tekniikoille, joita on käytetty arviointimallien rakentamiseen ja tarkastellaan tarkemmin mallin rakentamista kolmella eri tekniikalla: regressioanalyysi, analogia ja regressiopuu.

3.1 Arviointitekniikoiden luokittelu

Arviointimallien rakentamiseen soveltuvat tekniikat voidaan karkeasti jakaa kahteen luokkaan: parametrisiin ja ei-parametrisiin tekniikoihin (Briand et al. 1999).

Parametriset tekniikat muodostavat tilastollisen analyysin perusteella mallin, joka on yhtälö, jossa syöteparametreinä on yksi tai useampi työmäärään vaikuttava attribuutti. Yleisimmin käytettyjä tilastollisia menetelmiä ovat regressioanalyysi ja varianssianalyysi. Regressiomalli voidaan toteuttaa monella tavalla. Suhdeasteikolliset muuttujat sovitetaan yleensä eksponentiaaliseen malliin ja niille muodostetaan logaritmimuunnos, jolla malli saadaan lineaariseen muotoon. Yhden muuttujan regressioanalyysissä käytetään yleensä selittävänä muuttujana projektin kokoa ja muut työmäärään vaikuttavat attribuutit huomioidaan mallin sovituskertoimina. Usean muuttujan regressiomalli etsitään usein valikoivan askelluksen menetelmällä, jossa selittäjiä lisätään malliin yksitellen ja lisäyksen jälkeen tarkastellaan aiemmin malliin lisättyjen muuttujien tilastollista merkitsevyyttä P-arvolla; jos P-arvo ylittää sille asetetun rajan, muuttuja poistetaan mallista. Usean muuttujan regressioanalyysissa attribuutit, jotka ovat mitta-asteikoltaan luokittelevia, otetaan selittäjiksi ns. dummy-muuttujina. Jos kaikki selittäjät ovat tällaisia dummy-muuttujia, on kyseessä varianssianalyysi (Laininen 2000). Robusti regressio suorittaa iteratiivisesti painote-

tun pienimmän neliösumman regression. Sitä käytetään silloin, kun on tarvetta pyrkiä estämään poikkeavien arvojen voimakas vaikutus lopputulokseen. Paino lasketaan painofunktiolla, jonka parametrinä on residuaalin itseisarvo. Funktio antaa ykköstä lähellä olevan arvon pienille residuaaleille ja pienemmän painon suurille residuaaleille. Näin heikennetään poikkeavien arvojen vaikutusta (Laininen 2000).

Viime vuosina tilastollisten tekniikoiden rinnalle ovat tutkimuksissa nousseet *ei-parametriset tekniikat*. Ne ovat koneoppimiseen (machine learning) perustuvia tekoäly-tekniikoita, joissa malli luo säännöt syöteaineiston perusteella. Tutkittuja tekniikoita ovat esimerkiksi neuroverkot (neural networks), tapauspohjainen päättely (case-based reasoning) ja induktiivinen oppiminen (inductive learning) (Mair et al. 1999).

Neuroverkoissa jokainen neuroniksi kutsuttu yksikkö, joka muodostaa neuroverkon solmun, kuvaa aktiviteettia, jolla on useita syötteitä ja yksi vaste. Jokainen vaste on syöte seuraavalle yksikölle, kunnes saavutetaan päätesolmu, joka tuottaa lopullisen tuloksen, esimerkiksi työmäärän. Kullakin syötteellä on painokerroin, jonka perusteella vasteen merkitystä vahvistetaan tai heikennetään. Mallin käyttäjä opettaa verkkoa kokemuskannasta erotetulla opetusdatalla. Esimerkiksi käyttäen vastavirta-algoritmia (back-propagation). Kun verkko on sovitettu kokemuskannan projekteille, voidaan sitä käyttää uusien projektien työmäärän arviointiin (Pfleeger 1998).

Tapauspohjaista päättelyä käytetään analogiaperustaisessa arvioinnissa. Perusajatuksena on etsiä kokemuskannasta yksi tai useampi arvioitavaa projektia vastaava projekti, jonka tai joiden perusteella työmäärä arvioidaan.

Induktiivisen oppimisen perusajatuksena on luoda yleinen sääntöjen joukko yksittäisten tapausten perusteella. Säännöt muodostetaan sattumanvaraisesti valitulla tai algoritmisesti poimitulla opetusdatalla kokemuskannasta. Säännöt testataan lopulla aineistolla eli ns. testidatalla. Säännöt esitetään päätöspuun muodossa ja arvio muodostetaan puun lehtisolmuihin osuneiden tapausten perusteella (Mair et al. 1999).

Seuraavassa tarkastellaan tarkemmin kolmea arviointimallin luontiin soveltuvaa tekniikkaa. Regressiomalli edustaa tilastolliseen analyysiin perustuvaa tekniikkaa. Ei-parametrisistä tekniikoista esitetään tapauspohjaista päättelyä edustava analogia-tekniikkaa ja induktiivista oppimista edustava CART-algoritmiin perustuva regressiopuu. Regressiotekniikka on valittu mukaan, koska ensimmäiset työmäärän arviointiin rakennetut mallit perustuvat siihen ja se on edelleen

yleisimmin mallien rakentamiseen käytetty tekniikka (Fenton et al. 1998, Jeffery et al. 2001). Analogia ja regressiopuu ovat viime vuosina olleet tutkijoiden mielenkiinnon kohteina ja niillä toteutettuja malleja on vertailtu sekä keskenään että regressiomallien kanssa. Kaikkiin kolmeen tekniikkaan on myös saatavilla valmiita ohjelmistoja mallin luomista varten.

Kunkin tekniikan kohdalla on esitetty esimerkkiaineiston (liite 1) avulla mallin luomiseen liittyviä vaiheita. Esimerkkiaineistona on käytetty kuvitteellista kokemuskantaa, jonka perusta (ohjelmiston koko ja toteutunut työmäärä) on saatu Kemererin aineistosta (Shepperd 2001).

Esimerkkiaineistossa on ohjelmiston koko ilmoitettu sovittamattomina toimintopisteinä ja työmäärä päivinä. Aineistossa on kahdeksan ordinaaliasteikollista tilanneattribuuttia, jotka voivat saada arvoja 1 - 5. Kaikilla attribuuteilla keskimääräistä tilannetta kuvaa arvo 3, sitä pienemmät arvot kuvaavat keskimääräistä vaikeampaa tilannetta ja korkeammilla arvioilla tilanne on työmäärän kannalta keskimääräistä parempi.

3.2 Regressiomalli

Ensimmäiset ja yleisimmin kirjallisuudessa esitellyt arviointimallit perustuvat regressiotekniikkaan. Regressiomalli rakennetaan tutkimalla aiemmista projekteista kerätyn kokemuskannan avulla eri attribuuttien välisiä riippuvuuksia. Kokemuskannasta etsitään attribuutti, joka vaikuttaa projektin työmäärään eniten ja tämä valitaan ensisijaiseksi tekijäksi. Mallia voidaan tämän jälkeen sovittaa etsimällä muita työmäärään vaikuttavia tekijöitä (Fenton ja Pfleeger 1997).

3.2.1 Regressiomallin periaate

Yleinen regressiomalleissa käytetty yhtälö työmäärälle on seuraava (Shepperd et al. 1996):

$$E = aS^b \tag{3}$$

missä *E* on työmäärä, *S* on ohjelmiston koko, joka yleensä mitataan ohjelmakoodiriveinä, *a* on tuottavuutta kuvaava vakio ja vakio *b* kuvaa projektin laajuuden vaikutusta tuottavuuteen.

Epälineaarinen ohjelmiston koon ja työmäärän välinen yhteys saadaan logaritmimuunnoksella linearisoitua, jolloin voidaan soveltaa lineaarista regressioanalyysiä (Laininen 2000).

Kun regressiomallin perusyhtälö on rakennettu, analysoidaan tekijöitä, jotka aiheuttavat poikkeaman perusyhtälön antaman arvion ja toteutuneen työmäärän välillä. Voimme esimerkiksi havaita, että 80 % eroista samankokoisten järjestelmien työmäärissä voidaan selittää projektiryhmien välisen kokemuksen eroavuudella. Näitten työmäärään vaikuttavien lisäparametrien löytämiseen voidaan käyttää esimerkiksi faktorianalyysiä. Löydetyt tekijät lisätään malliin ja niille määritellään painoarvo, jolla säädellään tekijöiden vaikutusta mallin antamaan lopputulokseen. Painoarvoa ei tulisi antaa arvioimalla, vaan sen pitäisi perustua empiiriseen tietoon. Näin yhtälö saadaan sovitettuun muotoon (Fenton ja Pfleeger 1997):

$$E = (aS^b)F \tag{4}$$

missä F on sovituskerroin. Sovituskerroin muodostuu toissijaisille tekijöille määriteltyjen painoarvojen tulona. Näin voidaan tehdä silloin, kun käytetyt tekijät ovat toisistaan riippumattomia.

3.2.2 Regressiomallin soveltaminen

Seuraavassa rakennetaan kaksi regressiomallia. Mallien perusyhtälöiden rakentamiseen käytetään SPSS-ohjelmistoa ja mallit rakennetaan liitteen 1 aineistolla. Ensimmäisen regressiomallin (malli 1) rakentamiseen käytetään kaikkia aineiston 15 projektia. Regressiomallin perusyhtälöksi työmäärälle ohjelmiston koon suhteen saadaan:

$$E = 0,3790S^{0,8927} \tag{5}$$

Mallin selitysasteeksi (R^2) vapausasteilla 13 saadaan 0,661. Tuloksena saadaan Lainisen (2000) tulkintaa soveltamalla tilastollisesti merkitsevä riippuvuus (F=23,37 ja P=0,000), jossa ohjelmiston koon vaihtelu selittää siis 66,1 % työmäärän vaihtelusta.

Vertailun vuoksi luodaan toinen malli (malli 2), jossa mallin rakentamiseen käytetään kymmentä projektia ja loput viisi projektia jätetään testidataksi. Testidata eristetään aineistosta poistamalla

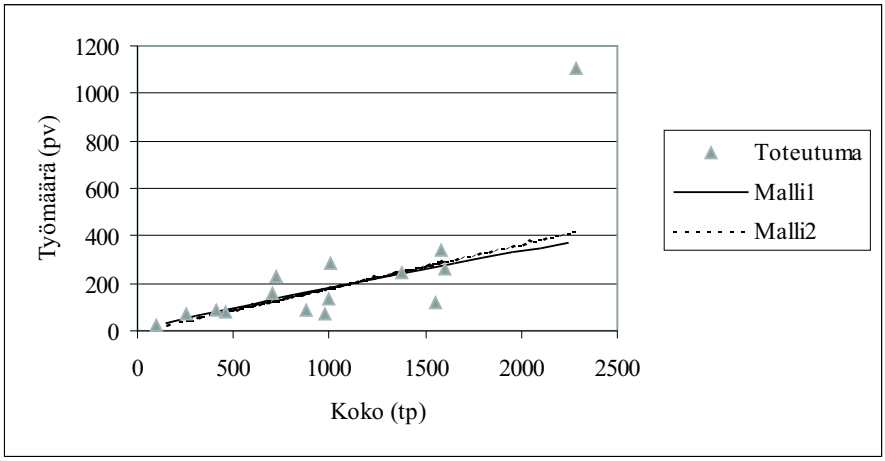
projektien työmäärän suhteen järjestetystä aineistosta joka kolmas projekti ts. projektit 1,2,7,9 ja 12.

Mallin 2 regressioyhtälöksi saadaan:

$$E = 0,1512S^{1,0257} \tag{6}$$

Mallin 2 selitysaste (0,618) on vain hieman heikompi kuin mallin 1 selitysaste ja riippuvuus on edelleen tilastollisesti merkitsevä (F=12,94 ja P=0,007).

Mallien 1 ja 2 muodostamat käyrät on esitetty kuvassa 6. Lisäksi kuvassa on esitetty esimerkkiaineiston mukaiset toteutuneet työmäärät.



Kuva 6: Regressiomallit 1 ja 2.

Taulukko 2: COCOMO-mallin sovituskertoimet (Boehm 1981).

Luokittelu	*LUOT*	*SUUN*	*MENET*	*SOVMON*	*SOVTUN*	*TYOK*	*TKKOKO*
Erittäin huono(1)	1,40	1,46	1,24	1,30	1,29	1,42	1,16
Huono (2)	1,15	1,19	1,10	1,15	1,13	1,17	1,08
Keskimäär. (3)	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Hyvä (4)	0,88	0,86	0,91	0,85	0,92	0,86	0,94
Erittäin hyvä (5)	0,75	0,71	0,82	0,70	0,82	0,70	0,94

Mallien 1 ja 2 yhtälöitä (5) ja (6) tarkennetaan tarkastelemalla kokemukannan projektien muita attribuutteja sopivien sovituskertoimen estimointia varten.

Sovituskertoimien löytäminen vaatii aineiston tilastollista analysointia, joka on rajattu tämän työn ulkopuolelle. Aineiston perusteella muodostettujen kertoimien sijaan tarkastellaan mallin tarkentamista COCOMO-mallin [1] tilanneattribuuttien sovituskertoimien avulla. COCOMO-malli sisältää 15 attribuuttia, joille malli tarjoaa sovituskertoimet (Fenton ja Pfleeger 1997). Taulukossa 2 on esitetty eräitä COCOMO-mallin kertoimia (Boehm 1981). Sovelluksen monimutkaisuus on COCOMO-mallissa luokiteltu kuuteen luokkaan, joista taulukosta 2 on jätetty pois korkein luokitus (extra high), jonka kerroin on 1,65. Tietokannan koko on COCOMO-mallissa luokiteltu neljään luokkaan, minkä vuoksi esimerkissä on kahdelle alimmalle luokalle käytetty samaa kerrointa 0,94.

Taulukossa 3 on esitetty testidatan projektien sovituskertoimet ja niiden tulona laskettu kokonaissovituskerroin.

Taulukko 3: Arvioitavien projektien sovituskertoimet.

Projekti	*LUOT*	*SUUN*	*MENET*	*SOVMON*	*SOVTUN*	*TYOK*	*TKKOKO*	*Yhteensä*
1	1,00	1,00	1,10	1,15	1,13	1,00	1,08	1,54381
2	0,88	1,00	1,00	1,15	0,92	1,00	0,86	0,80069
7	1,40	1,19	1,10	0,70	1,13	1,00	0,94	1,36261
9	1,00	0,86	1,00	1,15	0,92	1,00	1,00	0,90988
12	1,00	1,00	1,10	1,00	1,00	0,86	1,00	0,94600

Taulukossa 4 on esitetty arvioitavien projektien arviot molempien mallien perusversioilla ja sovitetuilla malleilla sekä projektien toteutunut työmäärä.

Yleensä tilastollisen tekniikan ongelmana on, että luotettavan mallin rakentamiseen tarvitaan kymmeniä tai mieluiten satoja projekteja. Esimerkkimallien tilastollinen merkitsevyys ei juurikaan heikentynyt, vaikka käytettävän aineiston määrä pieneni. Tarkempi arvio mallien tarkkuudesta on esitetty kohdassa 3.5.

[1] Tarkastelussa sovelletaan COCOMO81-mallia.

Taulukko 4: Regressiomallien arviot testidatan projekteille.

Projekti	*Malli 1 perus*	*Malli 1 sovitettu*	*Malli 2 perus*	*Malli 2 sovitettu*	*Toteutunut työmäärä*
1	182,2	281,3	182,4	281,6	287,0
2	89,8	71,9	80,9	64,8	82,5
7	22,5	30,7	16,5	22,5	23,2
9	267,7	243,6	283,8	258,2	116,0
12	135,4	128,1	129,7	122,7	230,7

3.3 Analogia

Analogiaan perustuvan arvioinnin perustana on kokemuskanta, johon on kerätty aikaisempien projektien toteutumatietoa. Arvioitava projekti kuvataan muuttujien avulla ja analogiatietokannasta etsitään päättyneitä samankaltaisia projekteja. Löytyneiden projektien toteutumatiedon perusteella muodostetaan arvio uudelle projektille.

3.3.1 Analogian periaate

Analogiaan perustuvassa arviointitapahtumassa joudutaan ratkaisemaan kolme peruskysymystä (Shepperd et al. 1996, Shepperd ja Schofield 1997). Ensin määritellään arvioitava projekti etsimällä ne muuttujat, jotka parhaiten kuvaavat projektia. Kuvaavia muuttujia voivat olla esim. sovellusalue, syötteitten lukumäärä, tietoalkioiden lukumäärä, näyttöjen lukumäärä jne. Muuttujat on määriteltävä siten, että tiedot ovat saatavilla jo projektin aikaisessa vaiheessa. Riippuvana muuttujana voidaan käyttää joko työmäärää tai tuottavuutta.

Muuttujien määrittelyn jälkeen etsitään niiden avulla kokemuskannasta samankaltaisia projekteja. Tätä varten on sovittava sääntö, jolla samankaltaisuus määritellään.

Kolmanneksi on määriteltävä sääntö, jolla löytyneiden projektien työmääristä voidaan muodostaa arvioitavan projektin työmääräarvio.

Lopputuloksen tarkkuuteen vaikuttaa se, kuinka monen analogian löytymiseen pyritään. Jos käytetään vain yhtä analogiaa, saattaa valituksi tullut projekti olla jollain tavoin poikkeuksellinen ja

aiheuttaa siten virheellisen arvion. Liian suuren analogiajoukon käyttäminen sen sijaan saattaa heikentää lähimpänä olevan analogian vaikutusta (Shepperd et al. 1996).

Projektia määritteleviä muuttujia kannattaa valita mahdollisimman paljon. Käytännön rajoitteena saattaakin olla ainoastaan tietojen saatavuus. Käytettävät muuttujat voivat olla joko suhdeasteikollisia tai luokittelevia. Muuttujien valinnassa käytetään asiantuntijoita, jotka tunnistavat merkittävät ominaisuudet arvioinnin kannalta. Projektin koko on yksi merkittävimmistä työmäärää määrittävistä muuttujista ja siksi käytettävien muuttujien joukossa on oltava vähintään yksi sellainen, jolla kuvataan projektin kokoa (Shepperd et al. 1996).

Analogiat löytyvät käyttämällä euklidista poikkeamaa n-dimensioisessa avaruudessa (Shepperd et al. 1996, Shepperd ja Schofield 1997, Briand et al. 2000). Dimensiot vastaavat muuttujia ja muuttujien arvot normalisoidaan siten, että kaikille muuttujille tulee yhtä suuri painoarvo (Shepperd et al. 1996).

Jokaiselle *n*:lle muuttujalle *k* saadaan näin laskettua arvioitavan projektin P_{jk} ja analogiaehdokasprojektin P_{ik} välinen etäisyys $\delta(P_{ik}, P_{jk})$ (Briand et al. 2000):

$$\delta(P_{ik}, P_{jk}) = \begin{cases} (\frac{|P_{ik}-P_{jk}|}{max_k - min_k})^2 & \text{,jos k on suhdeasteikollinen} \\ 0 & \text{,k on luokitteleva ja } P_{ik} = P_{jk} \\ 1 & \text{,k on luokitteleva ja } P_{ik} \neq P_{jk} \end{cases} \quad (7)$$

Tästä saadaan kahden projektin kokonaisetäisyydeksi:

$$distance(P_i, P_j) = \sqrt{\frac{\sum_{k=1}^{n} \delta(P_{ik}, P_{jk})}{n}} \quad (8)$$

Kun samankaltaiset projektit on löydetty kokemuskannasta, voidaan tietoa niiden toteutuneista työmääristä hyödyntää usealla eri tavalla. Arvioitavan projektin työmäärä voidaan arvioida laskemalla näiden projektien työmäärien painottamaton tai painotettu keskiarvo. Painotetun keskiarvon painottavana tekijänä käytetään samankaltaisuutta, jolloin lähimpänä olevan projektin merkitys kasvaa. (Shepperd et al. 1996, Shepperd ja Schofield 1997). Arviointi voidaan tehdä myös käyttämällä riippuvana muuttujana tuottavuutta. Tällöin työmäärä lasketaan jakamalla arvioitavan projektin koko samankaltaisten projektien tuottavuuden keskiarvolla (Briand et al. 2000).

3.3.2 Analogian soveltaminen

Seuraavassa esimerkissä arvioidaan liitteen 1 aineistolla uusi projekti, jonka attribuutit on esitetty taulukossa 5.

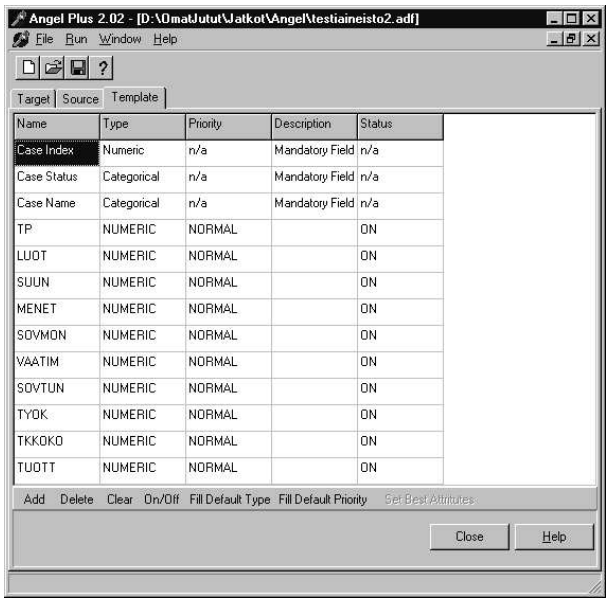
Taulukko 5: Arvioitavan uuden projektin attribuutit.

Attribuutti	*Arvo*
Toimintopisteiden lukumäärä	872
Luotettavuusvaatimukset	4
Suunnittelumenetelmien tuntemus	2
Menetelmien taso ja käyttö	3
Sovelluksen monimutkaisuus	3
Vaatimusten hallinnan vakaus	3
Sovellusaluetuntemus	2
Työkalukokemus	2
Tietokannan koko	3

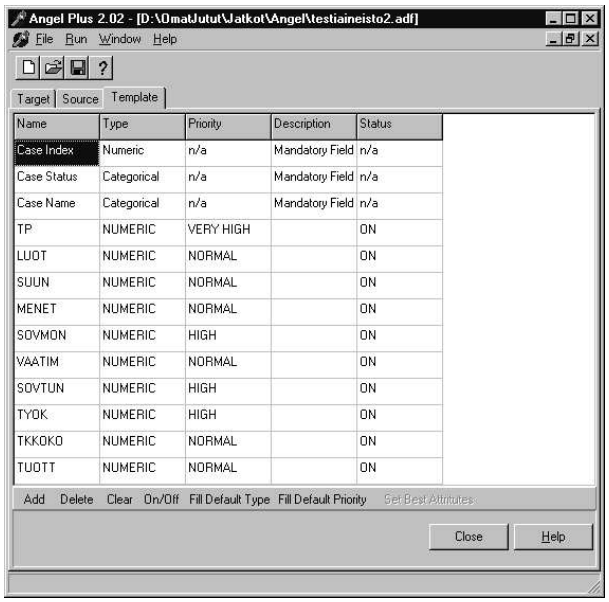
Mallin rakentamiseen käytetään ANGEL-ohjelmistoa, joka on Bournemouthin yliopiston kehittämä internetistä vapaasti asennetta sovellus (Shepperd 2001). ANGEL-ohjelmisto käsittelee attribuutteja, joko luokittelevina tai numeerisina. Esimerkkiaineiston kaikki attribuutit määriteltiin numeerisiksi.

Mikäli halutaan vahvistaa tai heikentää jonkin attribuutin vaikutusta samankaltaisuuden määrittelyssä, voidaan attribuutti priorisoida ANGEL-ohjelmistossa nelitasoisesti (low, normal, high, very high). Ensimmäisessä mallissa (malli 1) kaikki attribuutit priorisoidaan samanarvoisiksi (kuva 7). Ohjelmiston laskemat projektien kokonaisetäisyydet arvioitavasta projektista on esitetty taulukossa 6. Samankaltaisimmaksi projektiksi löytyy projekti kahdeksan. Työmäärä arvioidaan käyttäen yhtä ja kahta analogiaa. Yhden analogian mallilla työmääräarvioksi saadaan 130 päivää. Kahden projektin analogiaa käyttäen työmääräarvioksi saadaan 107,2 päivää laskemalla kahden samankaltaisimman projektin, projektien kahdeksan ja kuusi, työmäärien painottamaton keskiarvo. Vertailun vuoksi sama arvio tehdään arvioimalla työmäärän sijasta tuottavuutta. Tässä tapauksessa yhdellä analogialla tuottavuusarvio on projektin kahdeksan tuottavuus eli 7,659248 toimintopistettä päivässä, jolla työmääräarvioksi saadaan 113,8 päivää. Vastaavasti kahden analogian mallilla työmääräksi arvioidaan 138,9 päivää. Suorittamalla arviointi tuottavuuden kautta,

voidaan paremmin säädellä arviota arvioitavan projektin koon suhteen (Briand et al. 2000).



Kuva 7: Malli 1: Analogia-mallin muuttujien priorisointi.



Kuva 8: Malli 2: Analogia-mallin muuttujien priorisointi.

Mallin 1 heikkous on, että se huomioi kaikki attribuutit samanarvoisina. Toisessa mallissa (malli 2) asetetaan attribuuteille painoarvot siten, että koko asetetaan erittäin merkittäväksi attribuutiksi,

Taulukko 6: Malli 1: etäisyydet painottamattomilla attribuuteilla.

Projekti	*Etäisyys*
8	0,333886087895333
6	0,422549518686625
13	0,449485001243405
5	0,474845354255645
15	0,478975916456758
2	0,488435038353201
1	0,491888959739505
11	0,538694059799855
4	0,552772500430311
9	0,554863044659631
12	0,559471931801533
7	0,565932878950571
14	0,575085405038447
10	0,577783018152490
3	0,626094903453976

sovelluksen monimutkaisuus ja projektitiimin kokemus ja sovellusaluetuntemus merkittäväksi sekä muut attribuutit merkitykseltään normaaleiksi (kuva 8) .

Taulukossa 7 on esitetty arvioitavan projektin etäisyys kokemuskannan projekteista mallilla 2. Samankaltaisin projekti on edelleen 8, mutta nyt toisena on projekti 13. Yhdellä analogialla arvio muodostuu siis edelleen samaksi. Kahden analogian mallissa, jossa käytetään tuottavuutta riippuvana muuttujana, työmääräarvioksi saadaan 143,5 päivää. Yhteenveto analogia-mallien tuottamista arvioista on esitetty taulukossa 8.

Analogia-mallin ymmärtäminen on helppoa, sillä menetelmä vastaa samankaltaista päättelyä, jollaista arvioija käyttäisi asiantuntija-menetelmää käyttäessään. Malli tarjoaa kuitenkin systemaattisemman tavan etsiä samankaltaisia toteutuneita projekteja. Mallin rakentajalle jää analysoitavaksi, kuinka paljon attribuutteja kannattaa käyttää samankaltaisuuden määrittelyyn ja kuinka eri attribuutteja tulisi painottaa. Kokemuskannan projektien lukumäärällä ei ole merkitystä analogia-mallin tarkkuudelle, ongelmaksi jää ainoastaan, milloin kokemuskannan samankaltai-

Taulukko 7: Malli 2: etäisyydet painotetuilla attribuuteilla.

Projekti	*Etäisyys*
8	0,264221976798645
13	0,339457229687671
6	0,345717106843906
15	0,358780551453318
1	0,362256042146487
5	0,395499011987212
2	0,398865237955275
11	0,431305719179543
7	0,431748267729192
10	0,434034198193187
12	0,437585494156241
9	0,444013163866771
14	0,453992318830804
4	0,456437525856230
3	0,478733940885473

Taulukko 8: Yhteenveto analogia-mallien tuottamista arvioista.

Malli	*Arviointiperusta*	*Analogiat*	*Työmäärä*
Malli 1 - 1 analogia	Työmäärä	8	130,0
Malli 1 - 2 analogiaa	Työmäärä	8 ja 6	107,0
Malli 1 - 1 analogia	Tuottavuus	8	113,8
Malli 1 - 2 analogiaa	Tuottavuus	8 ja 6	138,9
Malli 2 - 1 analogia	Tuottavuus	8	113,8
Malli 2 - 2 analogiaa	Tuottavuus	8 ja 13	143,5

simmat projektit ovat riittävän samankaltaisia, jotta niitä voidaan käyttää luotettavan arvion perustana. Arvioinnin jälkeen on helppo verrata arvioinnin perustana käytettyjä projekteja arvioitavaan projektiin. Arvioinnin epävarmuutta voidaan sen jälkeen arvioida vaikkapa riskienhallinnan keinoin.

3.4 Regressiopuu

CART (Classification and Regression Tree) on yksinkertainen, mutta tehokas binääripuupohjainen luokittelumenetelmä. Se soveltuu sekä luokitteluun että lokaalien mallien ja niiden vaikutusten etsimiseen (Koivisto 2000). Työmäärän arviointi on regressio-ongelma, joten siihen sopii käytettäväksi regressiopuu. Puu muodostuu joukosta kysymyksen muodossa olevia luokittelusääntöjä, jotka esitetään puun polkuina. Vastaus kuhunkin kysymykseen on joko kyllä tai ei ja vastausten perusteella edetään puussa kohti lehtisolmua, joka kuvaa ennustettavaa ilmentymää. Regressiopuun avulla voidaan käsitellä eri mitta-asteikollisia muuttujia (Salford Systems 2001).

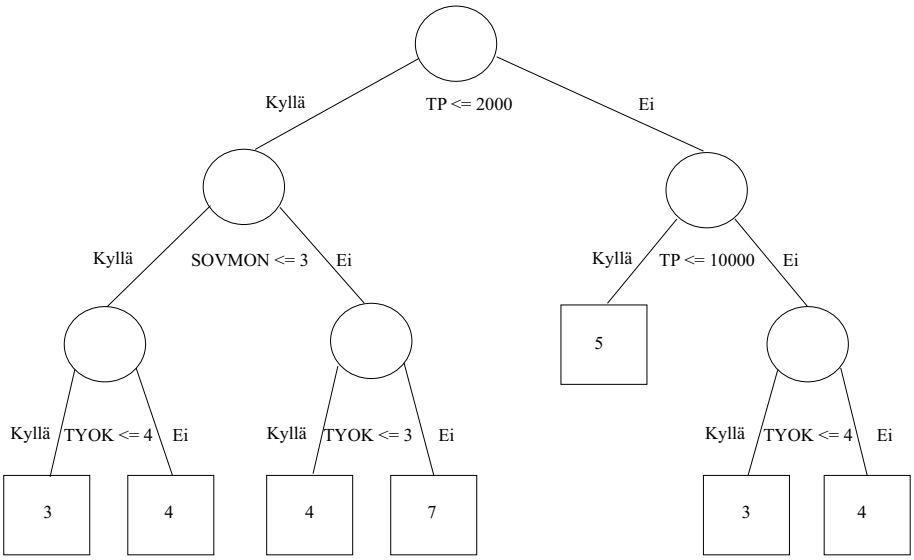
3.4.1 Regressiopuun periaate

Kuvassa 9 on kuvitteellinen esimerkki regressiopuusta työmäärän arviointia varten. Esimerkin juurisolmu esittää kysymyksen "Onko toimintopisteiden lukumäärä korkeintaan 2000". Vastauksella "kyllä" edetään vasempaan lapsisolmuun ja vastauksella "ei" oikeaan. Vasemmassa lapsisolmussa kysytään "Onko sovelluksen monimutkaisuuden luokittelu korkeintaan 3 (normaali tai normaalia monimutkaisempi)". Vastauksella "ei" edetään oikeanpuoleiseen lapsisolmuun, joka esittää kysymyksen "Onko projektitiimin keskimääräinen työkokemus luokiteltu pienemmäksi tai yhtä suureksi kuin 3 (normaali tai normaalia pienempi)". Vastauksella "kyllä" edetään vasemmanpuoleiseen lapsisolmuun, joka on puun lehtisolmu ja antaa arvon riippuvalle muuttujalle. Esimerkkipuussa riippuvana muuttujana on käytetty tuottavuutta. Lehtisolmujen arvot ovat tuottavuuslukuja (toimintopistettä päivässä), joka muodostetaan laskemalla solmun projektien keskimääräinen tuottavuus.

Regressiopuu muodostetaan siten, että valitaan riippuva muuttuja, jollaiseksi työmäärää arvioitaessa sopii tuottavuus tai työmäärä. Lisäksi valitaan riippumattomat muuttujat, jotka vaikuttavat riippuvaan muuttujaan. Kukin solmu puussa määrittää yhteen tällaiseen riippumattomaan muuttujaan liittyvän ehdon. Regressiopuun muodostaminen tapahtuu rekursiivisesti jakamalla käsiteltävä tietojoukko kunnes saavutetaan lopetusehto (Briand et al. 2000).

Puun muodostamiseksi määritellään kolme sääntöä (Breiman et al. 1984):

- sääntö kunkin solmun jakamiseksi



Kuva 9: Esimerkki regressiopuusta.

- lopetusehto jakamiselle
- kunkin päätössolmun ennustearvon määrittäminen.

Solmujen jakaminen tapahtuu siten, että jaon tuloksena olevat joukot jakautuvat mahdollisimman poikkeaviksi joukoiksi riippumattoman muuttujan suhteen (Breiman et al. 1984). Regressiopuussa jakosääntönä käytetään joko pienimmän neliösumman menetelmää (Least Squares, LS) tai pienimmän poikkeamien itseisarvon menetelmää (Least Absolute Deviation, LAD). Seuraavassa esitetään solmujen jakaminen pienimmän neliösumman menetelmällä.

Jakaminen aloitetaan laskemalla juurisolmun kaikkien projektien riippuvien muuttujien keskiarvon ja kunkin projektin riippuvan muuttujan erotuksen neliöitten keskiarvo. Koska solmun arvio riippuvalle muuttujalle on solmun projektien riippuvien muuttujien keskiarvo, kuvaa laskettu luku solmun virhettä, joka merkitään *R(t)*. Voidaan siis esittää (Breiman et al. 1984):

$$R(t) = \frac{1}{N} \sum_{x_n \in t} (y_n - \overline{y}(t))^2 \tag{9}$$

missä *N* on projektien lukumäärä solmussa *t*, y_n on projektin *n* riippuvan muuttujan arvo, *x* on riippumaton muuttuja ja $\overline{y}(t)$ on solmun *t* projektien riippuvien muuttujien keskiarvo.

Joukko eli juurisolmun projektit järjestetään jonkin riippumattoman muuttujan suhteen suuruusjärjestykseen. Tämän jälkeen muodostetaan juurisolmulle kaikki jakoehdokkaat eli mahdolliset solmuparit t_L ja t_R siten, että kaikkien solmun t_L projektien riippumattoman muuttujan arvo on pienempi kuin solmun t_R. Kullekin syntyneelle lapsisolmuehdokasparille lasketaan vastaavasti *R(*t_L*)* ja *R(*t_R*)*. Jako ja laskenta tehdään kaikkien riippumattomien muuttujien suhteen ja paras jako kullekin solmulle *t* on se, jolla saadaan maksimoitua erotus *R(t) - R(*t_L*) - R(*t_R*)* (Breiman et al. 1984).

Jakoa jatketaan rekursiivisesti kullekin lapsisolmulle, kunnes saavutetaan jaon lopetusehto. Jako lopetetaan, jos solmussa on vain yksi tapaus tai jos kaikilla solmun projekteilla on sama riippuvan muuttujan arvo. Lopetukselle voidaan antaa myös muita lopetusehtoja, esimerkiksi tapausten minimimäärä solmussa. Lehtisolmua kutsutaan *päätesolmuksi* ja näin muodostettua puuta *maksimaaliseksi puuksi* (Salford Systems 2001).

Koko puun *T* virhemittari *R(T)* saadaan laskemalla lehtisolmujen $\widetilde{T}$ virhemittarien arvot yhteen (Breiman et al. 1984):

$$R(T) = \sum_{t \in \widetilde{T}} R(t) \tag{10}$$

Tarkoituksena on toisin sanoen löytää paras jako s^* solmulle *t* kaikkien jakojen joukosta *S*, jolla virhemittarin *R(T)* arvo pienenee eniten:

$$\Delta R(s^*, t) = \max_{s \in S} \Delta R(s, t) \tag{11}$$

missä

$$\Delta R(s, t) = R(t) - R(t_L) - R(t_R) \tag{12}$$

Pelkkä binääripuun rakentaminen tuottaa useimmiten liian tiheän jaon ja tällöin on vaarana, että puu sopii ainoastaan puun rakentamiseen käytetylle joukolle, mutta saattaa antaa vääristyneen tuloksen muilla tietojoukoilla. Sopivan kokoinen puu etsitään muodostamalla maksimaalisesta puusta pienempiä puita karsimalla siitä oksia. Näin saadaan joukko alipuita. Maksimaalinen puu

ja alipuut testataan testiaineistolla parhaan puun löytämiseksi (Salford Systems 2001).

Regressiopuun karsiminen (pruning) tapahtuu yleensä poistamalla kaksi lehtisolmua kerrallaan. Tämän vuoksi alipuita syntyy usein paljon, toisin kuin luokittelupuilla, joissa voidaan poistaa laajempia oksia kerrallaan. Pienentäminen on optimointia puun virhemittarin ja kompleksisuuden suhteen. Tätä varten määritellään virhe-kompleksisuusmittari R_α lisäämällä virhemittariin hinta puun kompleksisuudelle (Breiman et al. 1984):

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}| \tag{13}$$

missä α on painotuskerroin ja $|\widetilde{T}|$ puun lehtisolmujen lukumäärä.

Karsinta-prosessin tuloksena syntyy maksimaaliselle puulle T_{max} joukko pieneneviä alipuita T_1 > T_2 > ... > $\{t_1\}$, jossa puu T_1 on pienempi tai yhtäsuuri kuin maksimaalinen puu T_{max} ja vastaavasti kasvava painotuskertoimien α jono *0* = α_1 < α_2 <... siten, että $\alpha_k \leq \alpha \leq \alpha_{k+1}$ ja T_k on pienin puun T_{max} alipuista, joka minimoi mittarin R_α*(T)* arvon (Breiman et al. 1984).

Parhaan puun valinta tapahtuu erillisen testijoukon avulla. Ennen puun rakentamista käytettävissä oleva tietojoukko jaetaan kahteen joukkoon, joista ensimmäistä käytetään puun rakentamiseen ja toista puun testaamiseen. Testijoukon tapaukset sijoitetaan löydettyjen puiden lehtisolmuihin ja kullekin puulle lasketaan R_α (Breiman et al. 1984).

Aina ei ole käytettävissä niin suurta tietojoukkoa, että erillisen testijoukon muodostaminen olisi mahdollista. Tällöin parhaan puun etsimiseen voidaan käyttää v-kertaista ristiinarviointia (v-fold cross-validation). Tässä menetelmässä tietojoukko jaetaan satunnaisesti keskenään mahdollisimman samankokoisiin osiin. Esimerkiksi kymmenkertaisessa ristiinarvioinnissa yhdeksää ensimmäistä osaa käytetään maksimaalisen puun luomiseen ja kymmenettä virhemittarin arvon laskemiseksi alipuille. Sama prosessi toistetaan kymmenen kertaa siten, että kullakin kierroksella käytetään puun rakentamiseen ja testaukseen eri osaa tietojoukosta (Salford Systems 2001).

Puun virhemittarin *R(T)* arvo vaihtelee riippuvan muuttuja mitta-asteikon mukaan. Tästä syystä regressiopuumallin tarkkuutta arvioidaan usein suhteellisena virheenä, jolloin paras puu on se, jonka suhteellinen virhe on pienin. Suhteellisen virheen laskentatapa riippuu siitä, mitä testimenetelmää on käytetty. Jos testaus on tehty yhdellä testiotoksella, merkitään mittari tunnuksella $RE^{ts}(T)$ ja se lasketaan seuraavasti (Breiman et al. 1984):

$$RE^{ts}(T) = \frac{R^{ts}(T)}{R^{ts}(\overline{y})} \tag{14}$$

missä $R^{ts}(T)$ on puun virhemittari testidatalla laskettuna ja $R^{ts}(\overline{y})$ on testiaineiston riippuvan muuttujan arvojen ja niiden keskiarvosta poikkeamien neliöitten keskiarvo.

Jos puun testaus tehdään ristiinarvioinnilla, lasketaan suhteellinen virhe seuraavasti (Breiman et al. 1984):

$$RE^{CV}(T) = \frac{R^{CV}(T)}{R(\overline{y})} \tag{15}$$

missä $R^{CV}(T)$ on ristiinarvioinnin tuottama virhemittarin arvo ja $R(\overline{y})$ on koko aineistosta laskettu erotusten neliöitten keskiarvo.

Kun jakosääntönä on LAD regressio, on tavoitteena edelleen löytää jako, jolla virhekertoimen arvo pienenee eniten, mutta nyt poikkeama lasketaan solmun riippuvan muuttujan mediaanista ja solmun *t* virhemittarin ***R(t)*** arvo lasketaan seuraavasti (Breiman et al. 1984):

$$R(t) = \frac{1}{N} \sum_{x_n \in t} |y_n - \nu(t)| \tag{16}$$

missä $\nu(t)$ on mediaani solmun *t* riippuvan muuttujan arvoista.

LAD-tekniikalla toteutetun puun suhteellinen virhe lasketaan testijoukolla testattaessa seuraavasti (Breiman et al. 1984):

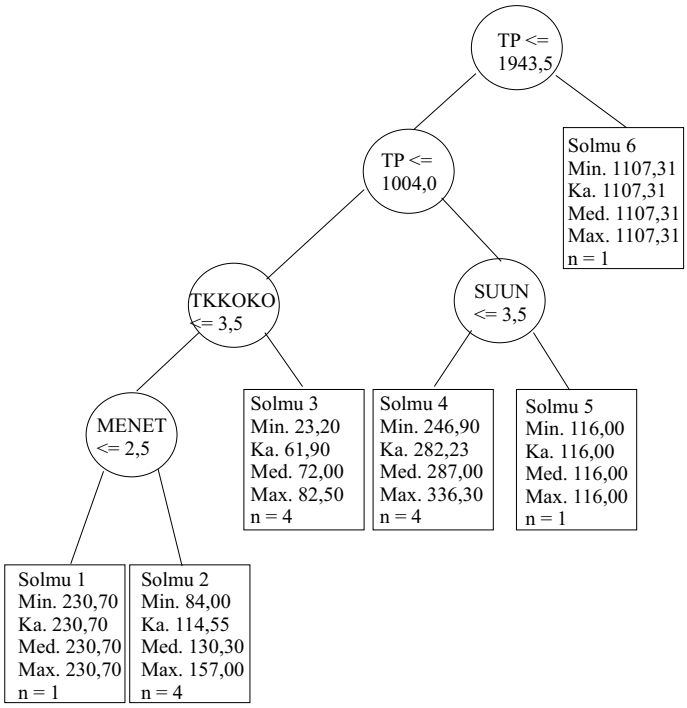
$$RE^{ts}(T) = \frac{R^{ts}(T)}{R^{ts}(\hat{\nu})} \tag{17}$$

ja vastaavasti ristiinarvioinnilla testattaessa seuraavasti (Breiman et al. 1984):

$$RE^{CV}(T) = \frac{R^{CV}(T)}{R(\hat{\nu})} \tag{18}$$

3.4.2 Regressiopuun soveltaminen

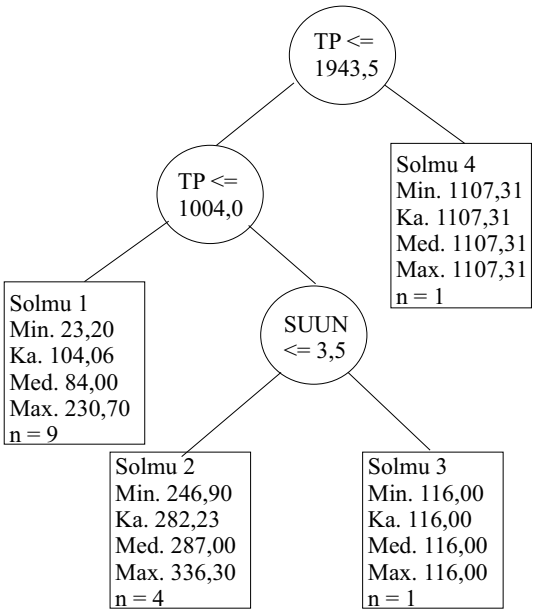
Seuraavassa luodaan esimerkkiaineistosta (liite 1) kolme mallia työmäärän arvioimiseksi. Mallien luontiin käytetään Salford Systemsin (2001) CART-ohjelmistoa. Kaikkien mallien jako tehdään pienimmän neliösumman menetelmällä ja jaon lopetusehdoiksi määritellään vähintään yksi esiintymä lehtisolmussa ja vähintään viisi esiintymää isäsolmussa. Puun testaukseen käytetään kymmenkertaista ristiinarviointia.



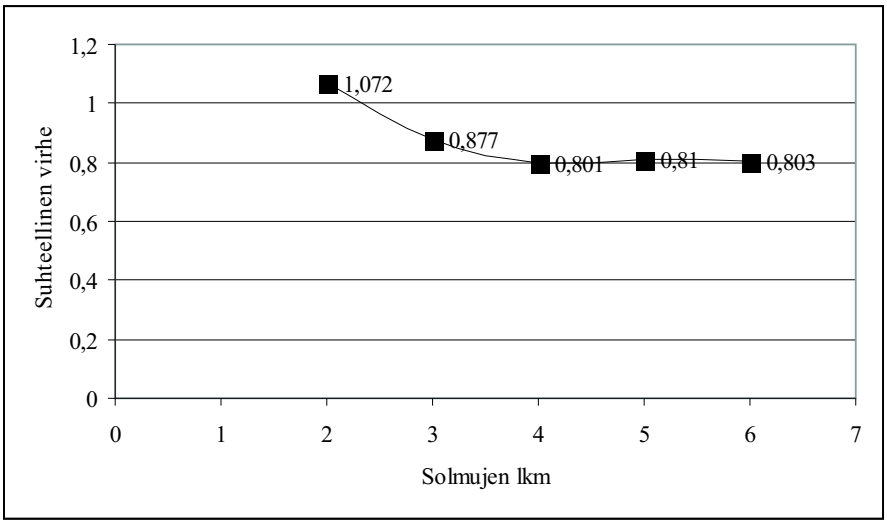
Kuva 10: Malli 1: Maksimaalinen puu työmäärälle.

Mallissa 1 riippuvaksi muuttujaksi valitaan työmäärä. Riippumattomiksi muuttujiksi valitaan loput yhdeksän esimerkkiaineiston attribuuttia.

Kuvassa 10 on esitetty mallille 1 luotu maksimaalinen puu ja kuvassa 11 mallin optimaalinen puu. Puiden lehtisolmuissa on esitetty solmun projektien työmäärien minimi- ja maksimiarvot sekä keskiarvo, mediaani ja solmun projektien lukumäärä. Eri alipuiden suhteelliset virheet on esitetty kuvassa 12. Optimaalisen puun suhteellinen virhe on 0,801 ja maksimaalisen puun 0,803. Maksimaalisessa puussa on kuusi päätesolmua, joista solmut 1, 2 ja 3 eli ehdot tietokannan koko <= 3,5 ja menetelmien taso <= 2,5, karsiutuivat optimoinnin tuloksena. Näin optimaaliseen puuhun jää neljä päätesolmua.



Kuva 11: Malli1: Regressiopuumalli työmäärälle.



Kuva 12: Malli 1: Alipuiden suhteelliset virheet.

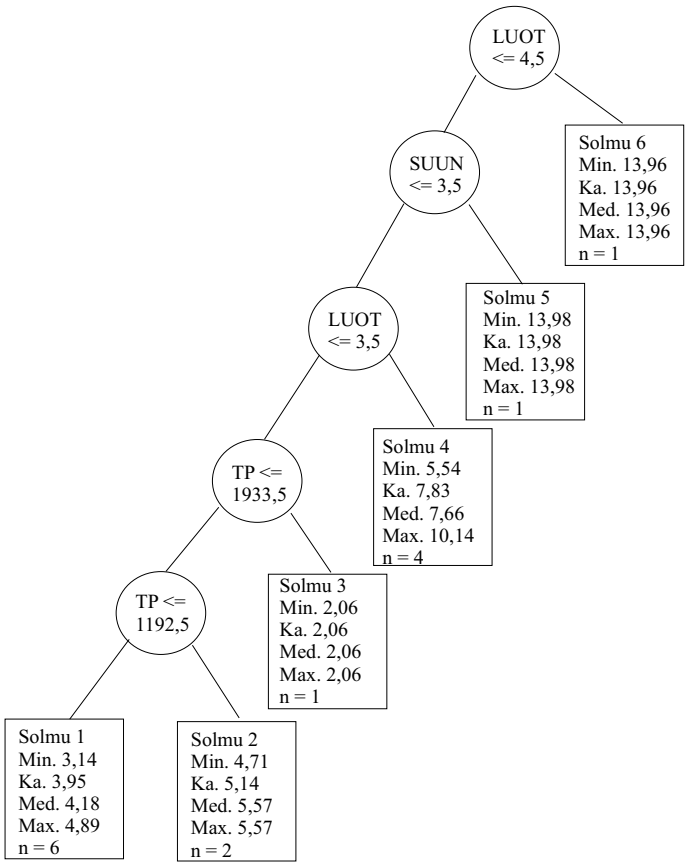
Malli muodostaa kuvan 13 mukaiset säännöt. Sääntöjen perusteella muodostetaan työmääräarvio projektille, jota on käytetty analogia-esimerkissä kohdassa 3.3 (taulukko 5). Projektin attribuutit täsmäävät päätesolmun 1 ehtoihin, jolloin työmääräarvioksi saadaan solmun projektien työmäärien keskiarvo eli 104,06 päivää.

```
/*Terminal Node 1*/
if
(
    TP <= 1004
)
{
    terminalNode = -1;
    mean = 104.056
}

/*Terminal Node 2*/
if
(
    TP > 1004 &&
    TP <= 1943.5 &&
    SUUN <= 3.5
)
{
    terminalNode = -2;
    mean = 282.225
}

/*Terminal Node 3*/
if
(
    TP > 1004 &&
    TP <= 1943.5 &&
    SUUN > 3.5
)
{
    terminalNode = -3;
    mean = 116
}

/*Terminal Node 4*/
if
(
    TP > 1943.5
)
{
    terminalNode = -4;
    mean = 1107.31
}
```

Kuva 13: Malli 1: Regressiopuumallin säännöt.

Mallissa 2 riippuvana muuttuja käytetään työmäärän sijasta tuottavuutta, jolloin CART-ohjelmisto muodostaa kuvan 14 mukaisen puun ja liitteessä 2 esitetyt säännöt. Arvioimalla esimerkkiprojektia tällä mallilla, saadaan tuottavuudeksi 7,38 toimintopistettä päivässä, jonka perusteella työmääräarvioksi voidaan laskea 118,15 päivää. Optimaalisen puun suhteelliseksi virheeksi tulee 0,796, joka on sama kuin mallin maksimaalisella puulla, jossa on seitsemän päätesolmua.

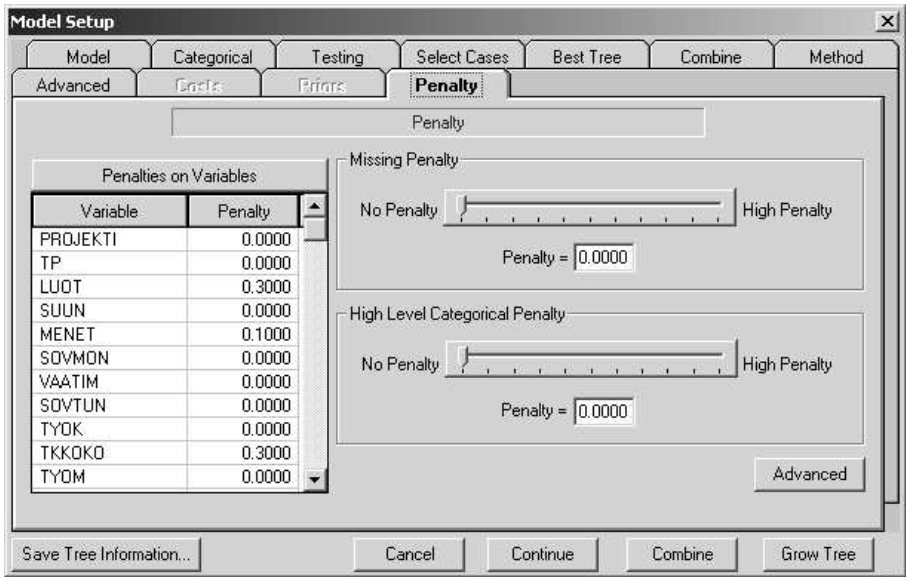
Malleissa 1 ja 2 kaikki riippumattomat attribuutit käsitellään solmujen jaossa samanarvoisina.



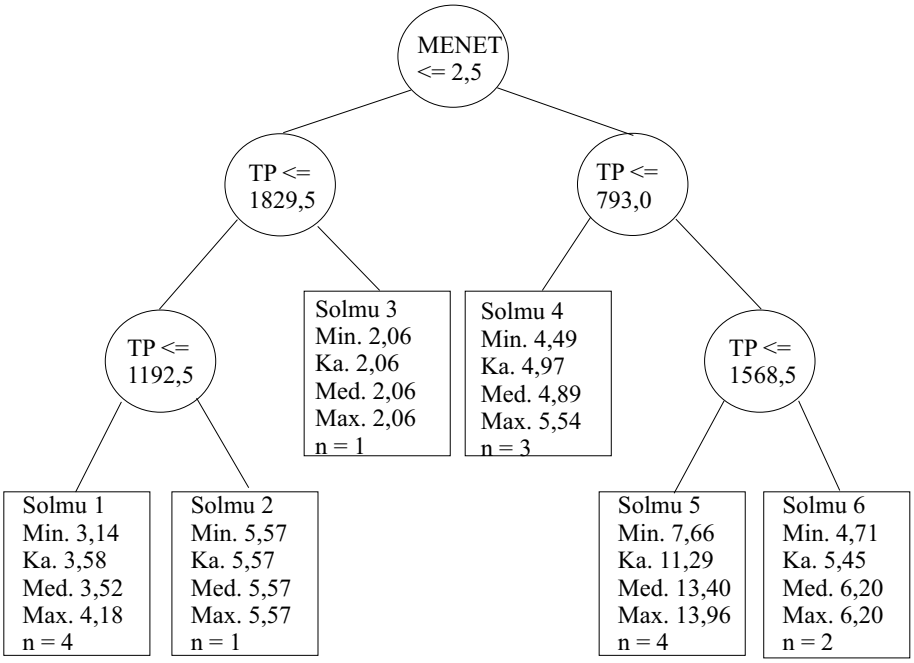
Kuva 14: Malli2: Regressiopuumalli tuottavuudelle.

CART-ohjelmisto mahdollistaa attribuuttien merkityksen muokkaamisen sakkokertoimien avulla. Asettamalla attribuutille sakkokerroin, voidaan pienentää sen mahdollisuutta tulla valituksi ensisijaiseksi jakotekijäksi siten, että sakkokerroin pienentää attribuutilla suoritetun jaon aikaansaamaa virhemittarin arvon paranemista. Mallissa 3 asetetaan edelleen riippuvaksi muuttujaksi tuottavuus ja asetetaan muille attribuuteille sakkokertoimet kuvan 15 mukaisesti. Tavoitteena on pienentää tietokannan koon, luotettavuusvaatimusten ja menetelmien tuntemisen painoarvoa. Malliksi muodostuu kuvan 16 mukainen puu (säännöt on esitetty liitteessä 3).

Arvioitava esimerkkiprojekti toteuttaa säännöt MENET > 2,5 ja TP > 793 ja TP <= 1568,5, joten arvio tuottavuudelle saadaan solmusta 5, jossa tuottavuus on 11,29 TP/päivä. Työmääräarvioksi saadaan 77,24 päivää. Maksimaalinen puu on myös optimaalinen puu. Sen suhteellinen virhe on 0,891 ja siitä saadaan karsittua kaksi alipuuta, joiden molempien suhteelliset virheet ovat yli yhden.



Kuva 15: Malli 3: Regressiopuumalli attribuuttien sakkokertoimet.



Kuva 16: Malli 3: Regressiopuumalli tuottavuudelle painottamalla attribuutteja.

Esimerkkimalleilla pienin työmääräarvio saadaan mallilla 3 ja suurin, jopa yli 53 % suurempi arvio, saadaan mallilla 2. Optimaalisten puiden suhteelliset virheet vaihtelevat 0,796 - 0,891. Yhteenveto regressiopuumalleista on esitetty taulukossa 9.

Taulukko 9: Regressiopuumalleilla arvioidut työmäärät.

Malli	*Riippuva muuttuja*	*Arvioitu työmäärä*	*Optimaalisen puun virhe*
Malli 1	Työmäärä	104,06	0,801
Malli 2	Tuottavuus	118,15	0,796
Malli 3	Tuottavuus	77,24	0,891

CART-mallin rakentaminen on periaatteessa yksinkertaista. Käytännössä mallin rakentaja joutuu kuitenkin ratkaisemaan monta asiaa, kuten solmujen jakamiseen käytettävän säännön, lopetusehdon jakamiselle ja attribuuttien painottamisen sakkokertoimilla. Jo pelkästään se, käytetäänkö riippuvana muuttujana työmäärää vai tuottavuutta, vaikuttaa mallin tuottamaan lopputulokseen. Kovin pienellä aineistolla eri vaihtoehtojen analysointi on vaikeaa, niinpä CART näyttäisikin soveltuvan osaksi arviointiprosessia, silloin kun käytettävissä on riittävän suuri kokemuskanta. Esimerkkiaineistolla rakennetuissa malleissa projektit jakautuvat solmuihin epätasaisesti ja useissa solmuissa on vain yksi projekti.

Regressionpuun suhteellinen virhe on aina positiivinen ja yleensä alle yhden. Joissain tapauksissa puun tarkkuus saattaa muodostua niin huonoksi, että suhteellinen virhe ylittää yhden (Breiman et al. 1984). Esimerkkitapauksissa suhteellinen virhe oli kohtalaisen suuri. Breimanin et al. (1984) mukaan pienien puiden ongelmana onkin, että niiden tarkkuus jää usein heikoksi.

3.5 Mallin hyvyyden arviointi

Kohdissa 3.2 - 3.4 on esitelty ohjelmistoprojektin työmäärän arviointiin sopivien mallien rakentamista eri tekniikoilla. Hughesin (2000) mukaan tehokas arviointimalli vaatii vakaan arviointitekniikan ja mallin on oltava oikein sovitettu käytettävään ohjelmistokehitysympäristöön. Hyväkään arviointitekniikka ei tuota hyvää lopputulosta, jos käytetään vääriä tuottavuusarvoja. Mallin rakentaminen ei ole vaikeaa, mutta rakentamisen jälkeen on varmistuttava siitä, että malli kykenee selittämään ohjelmistotuotantoprosessia ja arvioimaan työmäärän riittävän tarkasti. Ensin on varmistettava, että malli toimii sillä aineistolla, jolla se on rakennettu, mutta ideaalitilanteessa mallin tulisi toimia ilman suuria muutoksia myös eri aikoina ja erilaisista ympäristöistä kerätyillä aineistoilla (Conte et al. 1986). Mallien arviointitekniikoita on lukuisia. Joissain tapauksissa

voidaan tutkia arviointimallin kykyä tuottaa tarkkoja arvioita tietyssä ympäristössä ja joskus voidaan olla kiinnostuneita eri tekniikoiden kyvystä tuottaa tehokkaita malleja (Hughes 2000).

Koska tiedetään, että mikään malli ei voi huomioida kaikkia työmäärään vaikuttavia tekijöitä ja, että monien tekijöiden tarkka mittaaminen on vaikeaa, hyväksytään, että mallin antama arvio poikkeaa toteutuneesta (Conte et al. 1986). Seuraavassa esitellään yleisimmin käytettyjä mittareita, joilla arviointimallin tehokkuutta voidaan arvioida.

Mallin antaman arvion ja toteutuneen työmäärän välistä suhdetta voidaan tarkastella mittarilla, joka ilmaisee arvion suhteellisen virheen *RE* (Relative Error) (Conte et al. 1986):

$$RE = \frac{A - E}{A} \tag{19}$$

missä *E* on arvioitu työmäärä ja *A* toteutunut työmäärä.

Jos toteutunut työmäärä on suurempi tai yhtä suuri kuin arvioitu, on suhteellinen virhe positiivinen ja sen arvo on nollan ja yhden välillä. Jos toteutunut työmäärä on pienempi kuin arvioitu, on *RE* negatiivinen, eikä sillä ole alarajaa. Nyt voidaan määritellä keskimääräinen suhteellinen virhe *N*:lle arvioidulle projektille (Conte et al. 1986):

$$\overline{RE} = \frac{1}{N} \sum_{n=1}^{N} RE_n \tag{20}$$

missä *N* on projektien lukumäärä.

Mittari $\overline{RE}$ kuvaa huonosti mallin tarkkuutta, koska positiiviset yliarvioinnit ja negatiiviset aliarvioinnit kumoavat toisensa. Tämän vuoksi arvioinnin tarkkuuden perusmittarina käytetään suhteellisen virheen itseisarvoa *MRE* (Magnitude of Relative Error) (Conte et al. 1986):

$$MRE = \frac{|A - E|}{A} \tag{21}$$

Mitä pienempi MRE on, sitä paremmin arviointi on onnistunut.

Arviointimallin tarkkuutta *N*:lle arvioidulle projektille voidaan nyt arvioida laskemalla suhteellisten virheiden itseisarvojen keskiarvo *MMRE* (Mean Magnitude of Relative Error) (Conte et al.

1986):

$$MMRE = \frac{1}{N}\sum_{n=1}^{N} MRE_n \tag{22}$$

Silloin kun mittarin arvo on pieni, tuottaa malli keskimäärin hyviä arviointeja, vaikka joukossa voikin olla projekteja, joiden arvioinnit ovat olleet huonoja. Conte et al. (1986) pitävät hyvänä mallia, jonka *MMRE* on pienempi kuin 0,25.

Usein keskiarvon lisäksi tai sen sijasta mallin hyvyyttä arvioidaan suhteellisten virheiden mediaanina (MdMRE). Mediaanin etu keskiarvoon verrattuna on, että se ei ole yhtä herkkä poikkeavan suurille tai pienille suhteellisen virheen *MRE* arvioille (Briand et al. 1999).

Kun halutaan tarkastella, kuinka suuren osan arvioista malli on pystynyt tuottamaan tavoitellun *MRE*:n puitteissa, käytetään mittaria *PRED(q)*, joka kuvaa niiden projektien *k* osuutta kaikista projekteista *N*, joiden *MRE* <= q (Conte et al. 1986):

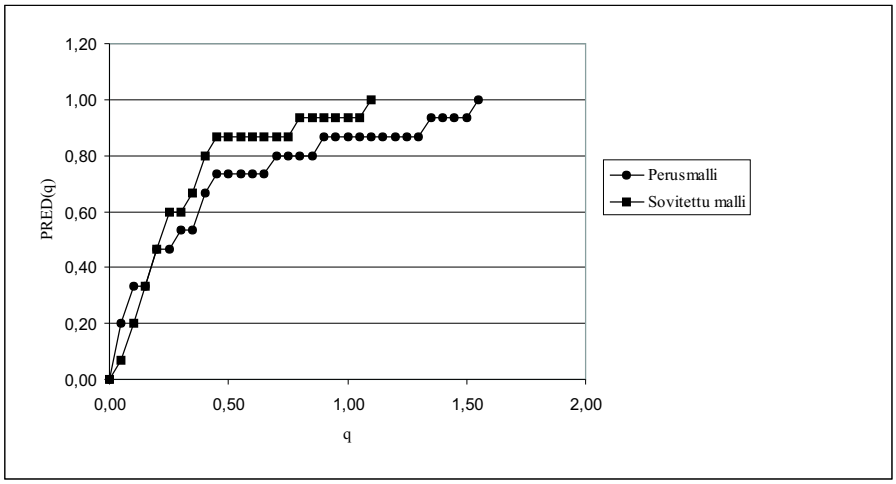
$$PRED(q) = \frac{k}{N} \tag{23}$$

Jos *PRED(0,25)* = 0,47, niin 47 % arvioinneista on poikennut korkeintaan 25 % toteutumasta. Conte et al. (1986) ehdottavat, että tekniikan arviointikykyä voidaan pitää hyväksyttävänä, mikäli *PRED(0,25)* on vähintään 0,75.

Kuvassa 17 on esitetty *PRED*-arvot nollasta yhteen eri *q*:n arvoilla kohdassa 3.2 esitellyille ensimmäisen regressiomallin (malli 1) perusmallille ja sovitetulle mallille. Käyrät risteävät *q*:n arvolla 0,15. Tämänkaltainen epäsäännöllisyys on Conten et al. (1986) mukaan tyypillistä pienillä tietojoukoilla, joissa muutamalla projektilla voi olla suuri vaikutus *PRED*-arvoon.

Taulukossa 10 on esitetty *MRE*-arvot arvioinneille, jotka on suoritettu regressiotekniikalla rakennetuilla mallin 1 perusmallilla ja sovitetulla mallilla.

Taulukossa 11 on esitetty regressiomallin 1 perusmallin ja sovitetun mallin tehokkuutta kuvaavat mittarit *MMRE, MdMRE* ja *PRED(0,25)*. Kaikilla mittareilla mitattuna sovitettu malli näyttäisi olevan tehokkaampi kuin perusmalli. Aina mallin tehokkuuden arvioiminen ei ole näin helppoa. Eri mittareilla mitattuna mallin hyvyys saattaa vaihdella. Siksi mallin arvioija joutuu arvioimaan



Kuva 17: *PRED* regressiomallin 1 perusmallille ja sovitetulle mallille.

Taulukko 10: Esimerkkiprojektien *MRE*-arvot regressiomallin 1 perusmallilla ja sovitetulla mallilla.

Projekti	*Perusmallin arvio*	*Sovitetun mallin arvio*	*Toteutunut työmäärä*	*Perusmallin MRE*	*Sovitetun mallin MRE*
1	182,22	281,31	287,0	0,36509	0,01982
2	89,77	71,88	82,5	0,08816	0,12872
3	377,53	693,98	1107,3	0,65906	0,37328
4	161,29	121,29	86,9	0,85609	0,39570
5	272,15	287,94	336,3	0,19074	0,14380
6	81,66	91,26	84,0	0,02784	0,08645
7	22,50	30,66	23,2	0,03007	0,32164
8	180,29	158,65	130,3	0,38363	0,21759
9	267,70	243,57	116,0	1,30775	1,09978
10	52,39	54,80	72,0	0,27231	0,23891
11	275,22	208,29	258,7	0,06387	0,19487
12	135,37	128,06	230,7	0,41321	0,44490
13	132,20	132,20	157,0	0,15799	0,15799
14	239,99	261,09	246,9	0,02797	0,05748
15	176,73	124,60	69,9	1,52839	0,78251

subjektiivisesti eri mittareiden tärkeyttä (Conte et al. 1986).

Taulukko 11: Regressiomallin 1 perusmallin ja sovitetun mallin tehokkuusmittarit koko aineistolla.

Mittari	*Perusmalli*	*Sovitettu malli*
MMRE	0,42481	0,31090
MdMRE	0,27231	0,21759
PRED(0,25)	0,47	0,60

Mallin hyvyyden arviointia vaikeuttaa se, että malli saattaa tuottaa hyviä arvioita jollain aineistolla, mutta toimiikin huonosti jollain toisella (Conte et al. 1986). Regressiomallin 1 mittarit on laskettu samalla aineistolla, jolla malli on rakennettu. Kitchenhamin (1998) mukaan luotettavammat arviot mallin tarkkuudesta saataisiin, jos käytettävä projektikanta olisi niin suuri, että se voitaisiin jakaa kahteen joukkoon, joista toisella satunnaisesti koko projektikannasta erotetulla opetusdatalla rakennettaisiin malli ja toisella ns. testidatalla suoritettaisiin mallin tarkkuuden arviointi.

Tarkastellaan vielä regressiomallien 1 ja 2 perusmallien tarkkuutta. Taulukossa 12 on esitetty mallien *MRE*-arvot mallin 2 testaamista varten erotetuilla projekteilla.

Taulukko 12: Regressiomallien 1 ja 2 perusmallien *MRE*-arvot testidatan projekteilla.

	Malli 1	*Malli 2*
1	0,365086	0,364373
2	0,088159	0,019666
7	0,030066	0,288957
9	1,307754	1,446612
12	0,413213	0,437998

Taulukossa 13 on esitetty molempien perusmallien tehokkuusmittareita testidatalla. Käytetty testiaineisto on niin pieni, ettei siitä voi tehdä johtopäätöksiä. Tulokset näyttävät kuitenkin tukevan väittämää, että mallin testaaminen samalla aineistolla, jota käytetään mallin rakentamiseen, tuottaa paremmat arvot mallin tarkkuutta kuvaaville mittareille kuin jos käytettäisiin erillistä testidataa, ja antaa siten liian optimistisen kuvan mallin tarkkuudesta.

Taulukko 13: Regressiomallien 1 ja 2 perusmallien tehokkuusmittarit testidatalla.

Mittari	*Malli 1*	*Malli 2*
MMRE	0,44086	0,51152
MdMRE	0,36509	0,36437
PRED(0,25)	0,40	0,20

3.6 Arviointitekniikoista tehtyjä tutkimuksia

Arviointitekniikoita ja malleja on tutkittu jo useiden vuosikymmenien ajan. Useissa viime vuosina tehdyissä tutkimuksissa on vertailtu ei-parametrisillä tekniikoilla toteutettujen mallien tarkkuutta erilaisten regressiomallien tarkkuuteen sekä vertailtu eri kokemuskannoilla rakennettujen mallien tarkkuutta.

Tässä kappaleessa esitellään neljä tutkimusta, joissa on vertailtu eri tekniikoilla tehtyjen arviointimallien tehokkuutta ja erilaisten kokemuskantojen käyttöä mallin rakentamiseen. Tehdyt tutkimustulokset antavat ristiriitaisen kuvan eri tekniikoiden paremmuudesta. Myös johtopäätökset yrityskohtaisen kokemushistorian paremmuudesta yleiseen kokemushistoriaan verrattuna ovat ristiriitaisia. Kitchenhamin (1998) mukaan käytettävissä olevan kokemuskannan koko ja kokemuskannan tietojen mitta-asteikollisuus vaikuttavat siihen, millä tekniikalla aineiston analysointi onnistuu parhaiten. Jos kaikki mallin rakentamiseen käytettävät attribuutit ovat suhdeasteikollisia, sopii analysointiin parhaiten usean muuttujan regressioanalyysi. Jos attribuuttien lukumäärä ei ole suuri ja ne ovat nominaali- tai ordinaaliasteikollisia, ovat sopivia tekniikoita regressionanalyysi, jossa käytetään dummy-muuttujia tai varianssianalyysi. Jos käytettävissä on suuri kokemuskanta, voidaan käyttää koneoppimiseen perustuvia algoritmeja kuten CART.

Shepperd et al. (1996)

Shepperd et al. (1996) ovat verranneet analogia-mallin, lineaarisen regressiomallin ja vaiheittaisen regressiomallin tuottamia arvioita. Vertailu tehtiin käyttäen kuutta eri kokemuskantaa. Analogia-mallilla *MMRE* vaihteli käytetystä aineistosta riippuen 39 prosentista 78 prosenttiin. Lineaarisella regressiomallilla ja vaiheittaisella valikoivan askelluksen regressiomallilla *MMRE* oli 45 - 252 %. Kaikilla aineistoilla analogia-malliin perustuvat arvioinnit onnistuivat paremmin kuin kummallakaan regressiotekniikkaan perustuvilla malleilla tehdyt arvioinnit. Analogiaan pe-

rustuvan mallin eduiksi todettiin lisäksi, että arviointi onnistuu, vaikka riippumattomien muuttujien ja toteutuneen työmäärän välille ei löydettäisi tilastollisesti merkitsevää riippuvuutta ja, että analogia-tekniikka ei kaipaa yrityskohtaista kalibrointia. Tutkijat pitivät sitä myös helpommin ymmärrettävänä, mikä lisää luottamusta arvioinnin tulokseen ja mahdollistaa arvioinnin tuloksen järkevyyden arvioinnin. Yhteenveto tutkimuksen ensimmäisen vaiheen tuloksista on esitetty taulukossa 14.

Taulukko 14: Mallien MMRE-arvot eri kokemuskannoilla (Shepperd et al. 1996).

Kokemuskanta	*Analogia*	*Lineaarinen Regressio*	*Vaiheittainen Regressio*
Albrecht	0,62	0,85	0,90
Atkinson	0,39	0,57	0,45
Desharnais	0,64	0,66	0,66
Finnish	0,62	1,33	1,01
Kemerer	0,62	1,07	1,07
Mermaid	0,78	2,52	2,52

Taulukko 15: Mallien MMRE-arvot jaetuilla kokemuskannoilla (Shepperd et al. 1996).

Kokemuskanta	*Analogia*	*Lineaarinen Regressio*	*Vaiheittainen Regressio*
Desharnais-1	0,37	0,39	0,41
Desharnais-2	0,29	0,29	0,29
Desharnais-3	0,26	0,36	0,36
Mermaid-E	0,53	0,62	0,62
Mermaid-N	0,60	2,26	-

Tutkimuksen toisessa vaiheessa kaksi kokemuskannoista jaettiin osiin, Desharnaisin kanta jaettiin kehitysympäristön perusteella ja Mermaidin kannasta erotettiin ylläpitoprojektit uuskehitysprojekteista. Tarkoituksena oli tutkia, kuinka aineiston homogeenisuus vaikuttaa arvioinnin tarkkuuteen. Shepperdin et al. (1996) mukaan tulokset vahvistivat aikaisempien tutkimusten tuloksia, joiden mukaan regressiomallien tarkkuus paranee pienemmillä homogeenisilla kokemuskannoilla. Kehitysympäristön perusteella jaetulla kokemuskannalla rakennetun lineaarisen regressiomallin *MMRE* parani 66 %:sta 29 - 39 %:iin ja vaiheittaisen regressiomallin 29 - 41 %:iin. Toisella

kannalla, jossa jako tehtiin ylläpito- ja uuskehitysprojekteihin, tarkkuus parani yläpitoprojekteilla rakennetuilla molemmilla regressiomalleilla 252 %:sta 62 %:iin. Uuskehitysprojekteilla lineaarisen regressiomallin tarkkuus parani ainoastaan 226 %:iin, eikä vaiheittaisen regression mallilla saatu tilastollisesti merkitsevää riippuvuutta, sillä projektien lukumäärä jäi liian pieneksi (kahdeksan). Regressiomallien tarkkuuden paranemisesta huolimatta analogiamallin tarkkuus oli edelleen MMRE:llä mitattuna parempi kaikilla kuudella kokemuskannalla (Shepperd et al. 1996). Toisen vaiheen tulokset on esitetty taulukossa 15.

Briand et al. (1999)

Briand et al. (1999) tutkivat suomalaisella Laturi-kokemuskannalla eri arviointitekniikoiden tarkkuutta. Tutkimuksen tarkoituksena oli verrata eri tekniikoilla rakennettujen mallien tarkkuutta ja tutkia vaikuttaako mallin tarkkuuteen se, onko se rakennettu yrityskohtaisella kokemushistorialla vai yritysten yhteisellä kokemushistorialla.

Tutkimukseen oli valittu seuraavat tekniikat: usean muuttujan regressioanalyysi (OLS), varianssianalyysi (ANOVA), regressiopuu (CART) ja analogia. Analogiamalleja muodostettiin kaksi: yhden analogian malli ja kahden analogian malli. Taulukossa 16 on esitetty eri mallien tarkkuus koko Laturi-kannasta rakennetuilla malleilla.

Taulukko 16: Koko kokemuskannalla rakennettujen mallien tarkkuus (Briand et al. 1999).

Tekniikka	*MMRE*	*MdMRE*	*PRED(0,25)*
Regressio	0,53	0,440	0,31
ANOVA	0,57	0,446	0,34
CART	0,52	0,390	0,34
Analogia-1	1,16	0,610	0,24
Analogia-2	1,34	0,570	0,17

Taulukossa 17 on esitetty yrityskohtaisella kokemushistorialla rakennettujen mallien tarkkuus. Shepperdin et al. (1996) tutkimuksesta poiketen, tässä tutkimuksessa yrityskohtaisen aineiston käyttö paransi mittareilla *MMRE* ja *MdMRE* mitattuna analogiamallien tarkkuutta, mutta regressiomallin tarkkuus heikkeni. Mittarilla *PRED(0,25)* mitattuna ainoastaan kahden analogian malli pystyi tuottamaan hieman tarkemman arvion.

Briandin et al. (1999) mukaan analogiamallien muita heikompi tarkkuus saattoi johtua mene-

Taulukko 17: Yrityskohtaisella aineistolla rakennettujen mallien tarkkuus (Briand et al. 1999).

Tekniikka	*MMRE*	*MdMRE*	*PRED(0,25)*
Regressio	0,670	0,410	0,22
ANOVA	0,787	0,424	0,22
CART	0,569	0,462	0,29
Analogia-1	0,710	0,480	0,14
Analogia-2	0,750	0,470	0,21

telmästä, jolla analogiat etsittiin. Kaikki mallin attribuutit huomioitiin samanarvoisina samankaltaisuutta määriteltäessä. Briandin et al. (1999) tutkimuksen perusteella yksikään tekniikka ei kuitenkaan noussut ylitse muiden. Tutkijat toteavatkin, että arvioinnin tarkkuus ei riipu käytetystä tekniikasta, vaan mallin rakentamiseen käytetyn kokemuskannan tietojen laadusta ja riittävyydestä. Yksikään malleista ei kyennyt tuottamaan riittävän tarkkaa arvioita, vaikka mallin rakentamisessa huomioitiin suuri joukko attribuutteja ja kokemuskannan keruussa oli huomioitu aineiston yhtenäisyyden vaatimus. Tutkijat ehdottavat, että malliin huomioitaisiin yrityskohtaisesti sopivia attribuutteja, joilla malli kalibroitaisiin huomioimaan paremmin yrityksen olosuhteet (Briand et al. 1999).

Briand et al. (2000)

Briand et al. (2000) toistivat Laturi-kannalla tehdyn tutkimuksen (Briand et al. 1999) käyttäen ESA:n (European Space Agency) kokemushistoriaa.

Taulukossa 18 on tutkimustulokset kolmelle eri tavalla kokemuskantaa hyödyntäville malleille. Ensimmäisessä vaiheessa on rakennettu mallit koko kokemuskannalla. Toisessa kokeessa aineistosta on erotettu yhden yrityksen kokemushistoria, jolla on rakennettu mallit ja arvioitu tämän saman yrityksen projekteja. Tämä jäljittelee tilannetta, jossa yritys käyttää omaa kokemushistoriaa mallin rakentamiseen. Kolmannessa kokeessa jäljitellään tilannetta, jossa yritys käyttää ulkopuolisella aineistolla rakennettuja malleja. Kokemuskannasta erotetaan yhden yrityksen projektit ja malli rakennetaan lopulla aineistolla. Aineistosta erotetun yrityksen projekteja käytetään mallin arviointiin.

Tässä tutkimuksessa on aiemman Laturi-tutkimuksen lisäksi kaksi ANOVA- ja CART-mallia, joista toisessa on käytetty riippuvana muuttujana työmäärää (ANOVA_e ja CART_e) ja toisessa

tuottavuutta (ANOVA_p ja CART_p). Analogiamalli perustuu yhden analogian etsimiseen ja siinä riippuvana muuttujana on käytetty tuottavuutta.

Taulukko 18: Eri tekniikoiden *MdMRE*-tarkkuus erilaisilla kokemuskannoilla (Briand et al. 2000).

Tekniikka	*Koko kanta*	*Yrityskohtainen*	*Ulkopuolinen*
OLS	0,32	0,34	0,32
ANOVA_e	0,40	0,26	0,30
ANOVA_p	0,62	0,73	0,69
CART_e	0,70	0,41	0,69
CART_p	0,80	0,73	0,80
Analogia	0,76	0,73	0,76

Briandin et al. (2000) tutkimuksen perusteella OLS ja ANOVA_e toimivat selvästi muita tekniikoita tarkemmin sekä koko kannan aineistolla, että yrityskohtaisella aineistolla. Tekniikat osoittautuivat yhtä tarkoiksi mutta, koska tutkijoiden mukaan ANOVA-tekniikan automatisointi on vaikeampaa kuin OLS-tekniikan, ei sen käytöllä saavuteta merkittäviä etuja ainakaan tässä tutkimuksessa käytetyn aineiston kaltaisilla tapauksilla (Briand et al. 2000).

Yrityskohtaisen aineiston käyttäminen ei näyttänyt parantavan mallien tarkkuutta verrattuna yrityksen ulkopuolisella aineistolla rakennettuihin malleihin. Eniten tarkkuus parani CART_e-tekniikalla. Yrityskohtaisten erityispiirteiden huomioiminen arviointimalleissa onnistuu parhaiten tutkimalla yrityksen kannalta merkittäviä tekijöitä ja suunnittelemalla niille räätälöidyt mittausmenetelmät. Yleisten COCOMO-mallin tyyppisten attribuuttien kerääminen yrityskohtaisesti ei lisää mallin tarkkuutta (Briand et al. 2000).

Kahdessa tarkimmassa tekniikassa riippuvana muuttujana käytetty työmäärää ja työmäärä-koko suhde on esitetty eksponentiaalisena. CART_e-mallissa ohjelmiston kokoa on käytetty riippumattomana muuttujana, mutta malli ei muodostanut yhteyttä työmäärän ja koon välille. Tuloksien perusteella tutkijat päättelevät, että tarkimpia arvioita tuottavat mallit, jotka kykenevät huomioimaan työmäärän ja ohjelmiston koon välisen epälineaarisen riippuvuuden (Briand et al. 2000).

CART-tekniikan etuina on, että se ei vaadi tilastollista analyysia ja sen tulkinta ja mallin rakentaminen on helppoa. Tutkimustuloksista huolimatta CART saattaa olla varteenotettava vaihtoehto

tilanteessa, jossa kyetään rakentamaan tasapainotettu puu. Analogia-tekniikan monimutkaisuus on samankaltaisuuden määrittelyssä, jolle ei ole olemassa selkeää ja hyvin ymmärrettyä menetelmää. Tämän vuoksi analogia on suositeltava vaihtoehto ainoastaan, jos ei ole käytettävissä riittävän suurta kokemushistoriaa OLS-regressiomallin tai CART regressiopuun rakentamiseen ja, mikäli on käytettävissä asiantuntijoita samankaltaisuuden määrittelyyn (Briand et al. 2000).

Jeffery et al. (2001)

Jeffery et al. (2001) ovat tutkineet arviointitekniikoiden tarkkuutta International Software Standards Benchmarking Groupin (ISBSG) tietokannalla. Tutkimusongelman asettelu on samanlainen kuin kahdessa edellä esitetyssä tutkimuksessa (Briand et al. 1999 ja Briand et al. 2000). Tutkimuksessa oli mukana OLS regression, analogian, varianssianalyysin ja CART-tekniikan lisäksi vähemmän tutkittu tekniikka robusti regressio, jolla pyritään pienentämään OLS regressiossa esiintyvää poikkeavien esiintymien aiheuttamaa ongelmaa (Jeffery et al. 2001).

Tutkimuksessa on rakennettu eri tekniikoilla mallit jäljitellen tilanteita, joissa yritys käyttää omaa kokemuskantaa ja yritysten yhteistä kantaa mallin rakentamiseen. Taulukossa 19 on esitetty eri tekniikoiden tarkkuus yrityksen omien projektien perusteella rakennetuilla malleilla. Paras tarkkuus on saatu analogia-, OLS- ja CART_p-tekniikoilla (Jeffery et al. 2001).

Taulukko 19: Yrityskohtaisella aineistolla rakennettujen mallien tarkkuus (Jeffery et al. 2001).

Tekniikka	*MMRE*	*MdMRE*	*PRED(0,25)*
Analogia	0,305	0,208	0,58
OLS	0,254	0,228	0,58
Robusti Regressio	0,320	0,263	0,50
ANOVA_p	0,489	0,475	0,00
ANOVA_e	0,297	0,276	0,42
CART_p	0,238	0,178	0,67
CART_e	0,760	0,451	0,25

Taulukossa 20 on esitetty usean yrityksen yhteisellä aineistolla rakennettujen mallien tarkkuus yhden yrityksen projekteja arvioimalla. Kaikilla tunnusluvuilla mitattuna mallien tarkkuus on erittäin heikko (Jeffery et al. 2001).

Tutkimuksen tuloksiin on vaikuttanut se, että tutkimuksessa käytetty ISBSG-kannan aineisto oli

Taulukko 20: Usean yrityksen aineistolla rakennettujen mallien tarkkuus (Jeffery et al. 2001).

Tekniikka	*MMRE*	*MdMRE*	*PRED(0,25)*
Analogia	1,145	0,701	0,17
OLS	0,895	0,683	0,00
Robusti Regressio	0,848	0,638	0,00
ANOVA_p	1,234	1,124	0,00
ANOVA_e	1,304	1,137	0,00
CART_e	1,935	2,129	0,08
CART_e	1,563	1,316	0,08

erittäin heterogeenista. Koko kannassa oli 324 projektia, joiden tuottoaste (tuntia per toimintopiste) vaihteli 0,4:stä 73,5:een, projektit olivat kooltaan 9 - 17518 toimintopistettä ja työmäärä oli 97 - 59809 tuntia. Tutkimuksessa käytetyn yrityksen projektien tiedot olivat huomattavasti homogeenisempia. Projekteja oli käytettävissä 14, niiden koko oli 56 - 579 toimintopistettä, työmäärä 170 - 1238 tuntia ja tuottoaste 1,4 - 3,8 tuntia / toimintopiste (Jeffery et al. 2001). On ilmeistä, ettei heterogeenisella aineistolla voida mallintaa ohjelmistotuotantoprosessia riittävän hyvin ellei siinä ole projektin tilannetta luonnehtivien attribuuttien kuten ohjelmiston tyyppi, ympäristö ja organisaation tyyppi lisäksi ohjelmistotuotantoprosessin kypsyystasoa kuvaavia mittareita (kuten SPICE tai CMM). Kypsyystasomittareiden linkittäminen kokemuskantoihin mahdollistaisi ohjelmistotuotantoprosessin dynamiikan paremman ymmärtämisen (Jeffery et al. 2001).

4 Arvioinnin epävarmuus ja riskienhallinta

Riippumatta siitä millä menetelmällä ja tekniikalla työmäärän arviointi on suoritettu, liittyy tulokseen epävarmuutta, joka voi vaihdella yksittäisten arviointien välillä. Ohjelmistoprojektiin sisältyy työmäärän arviointiin liittyvien epävarmuustekijöiden lisäksi monia muitakin epävarmuustekijöitä. Kaikki epävarmuustekijät eivät edusta riskiä, mutta toisaalta riskejä ei ole ilman epävarmuutta (Pittman 1999).

4.1 Riskienhallinta

Riskillä tarkoitetaan tappion mahdollisuutta tai menettämisen uhkaa (Pfleeger 1998). Projektin eri sidosryhmät tarkastelevat projektin onnistumista eri näkökulmista. Asiakkaille ja suunnittelijoille projektin epäonnistumista kuvaavat aikataulujen ja kustannusten ylitykset. Käyttäjille ei-toivottu lopputulos on virheellinen toiminnallisuus tai huono käytettävyys ja ylläpitäjille lopputuotteen huono laatu (Boehm 1991).

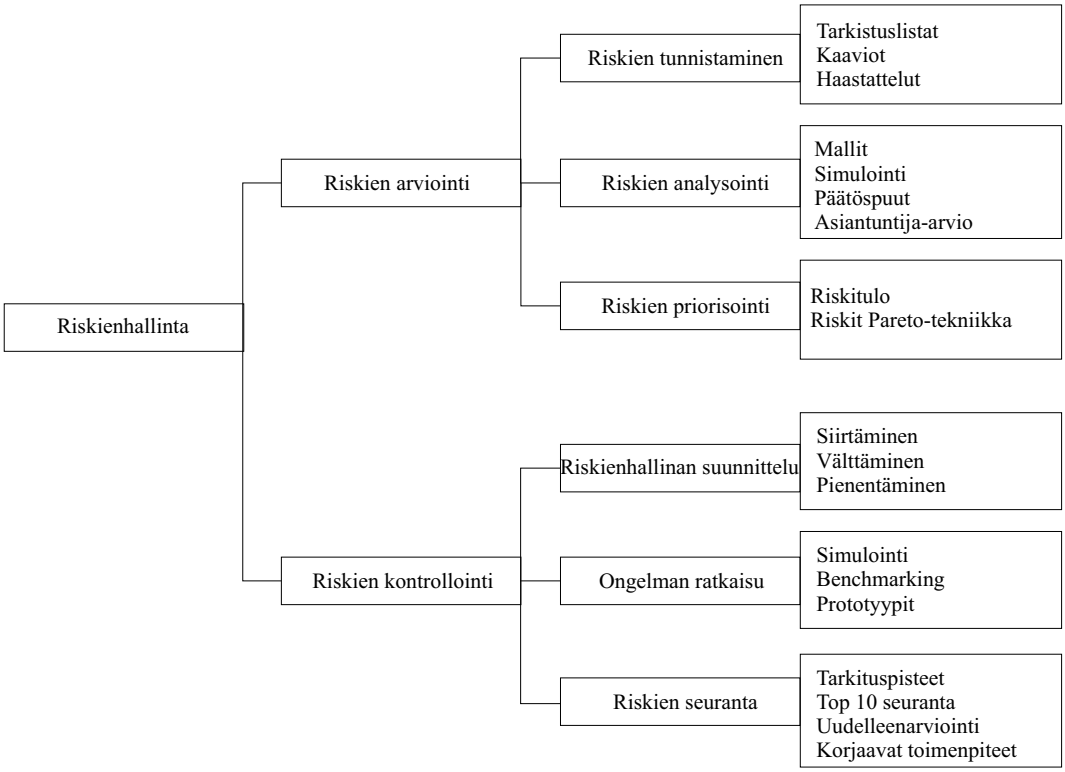
Riskienhallinnalla tarkoitetaan aktiviteettia, jonka tarkoituksena on tunnistaa, analysoida ja kontrolloida projektin riskejä ennen kuin niistä muodostuu uhka onnistuneelle projektille. *Riskienhallintaprosessilla* tarkoitetaan systemaattista, täsmällisesti määriteltyä, jatkuvasti kehitettävää riskienhallinnan aktiviteettia (Kontio 2001).

Kuvassa 18 on esitetty ohjelmistoprojektin riskienhallinnan vaiheet. Riskienhallinta-aktiviteetti käsittää kaksi päävaihetta: riskien arviointi ja riskien kontrollointi (Boehm 1991, Kontio 2001, Rimpiläinen ja Peltonen 2000).

Ensimmäinen vaihe, *riskien arviointi*, sisältää riskien tunnistamisen, analysoinnin ja priorisoinnin.

Riskien tunnistamisen tarkoituksena on tunnistaa projektin kaikki mahdolliset riskit. Käytettyjä riskien tunnistamisen menetelmiä ovat tarkistuslistat, aivoriihi-tekniikka, Delphi-menetelmä ja päättyneistä projekteista koottuun kokemuskantaan perustuva riskirekisteri. Apuna voidaan käyttää kaavioita ja haastatteluja (Rimpiläinen ja Peltonen 2000).

Riskien analysoinnin tarkoituksena on hankkia tunnistetuista riskitekijöistä niin paljon tietoa, et-



Kuva 18: Riskienhallinnan vaiheet (Boehm 1991, Pfleeger 1998, Kontio 2001).

tä riskienhallintasuunnitelman teko on mahdollista. Riskit voidaan ryhmitellä hallittaviksi kokonaisuuksiksi esimerkiksi riskin tyypin tai kriittisyyden perusteella ja riskeille arvioidaan niiden laajuus ja toteutumisen todennäköisyys. Analysointi voi perustua esimerkiksi arviointimalleihin tai tilastolliseen analyysiin (Boehm 1991, Kontio 2001).

Projektiin liittyy paljon riskejä, mutta kaikkien niiden kontrolloimiseen ei kannata käyttää resursseja. *Riskien priorisoinnilla* pyritään löytämään ne riskit, jotka ovat kaikkein merkittävimpiä ja joiden kontrolloimiseen on tärkeintä panostaa (Kontio 2001). Riskien priorisointi voidaan tehdä riskin laajuuden ja todennäköisyyden tulona lasketun riskitulon avulla. Tärkeimpiä ovat ne riskit, joiden riskitulo on suurin (Pfleeger 1998).

Kontion (2001) mukaan riskien määrällistäminen riskitulon avulla on ongelmallista, sillä riskin toteutumisen todennäköisyys joudutaan usein arvaamaan. Näin riskin määrällistämiseen itseensä liittyy epävarmuustekijä. Riskitulon sijaan Kontio (2001) ehdottaa priorisointiin Riskit Pareto-luokittelutekniikkaa, joka on osa Riskit-riskienhallintamenetelmää. Pareto-tekniikassa muodostetaan taulukko, jonka sarakkeina on riskien todennäköisyysprioriteetit ja riveinä riskien laajuu-

den prioriteetit, riskit sijoitetaan taulukon soluihin ja riskien prioriteetti tulkitaan sen sijainnista taulukossa. Pareto-tekniikka tuottaa ainoastaan osittaisen järjestyksen, sillä kaikkien riskien prioriteettia muihin nähden se ei kykene ilmaisemaan. Tekniikka on Kontion (2001) mukaan kuitenkin riittävä riskienhallintasuunnitelman tekoa varten.

Riskienhallinnan toinen päävaihe *riskien kontrollointi* koostuu riskienhallinnan suunnittelusta, riskiongelman ratkaisusta ja riskien seurannasta.

Riskien kontrollointia varten muodostetaan riskianalyysin perusteella *riskienhallintasuunnitelma*, jonka seuraaminen on osa projektinhallintaa (Sallis et al. 1995). Suunnitelmaan kirjataan toimenpiteet riskien siirtämiseksi, välttämiseksi tai vaikutuksen pienentämiseksi. Suunnitelmassa huomioidaan riskienhallinnan kustannukset, jotka vaikuttavat siihen, mitkä vaihtoehtoisista toimenpiteistä kunkin riskin osalta toteutetaan. Menetelminä voidaan käyttää tarkistuslistoja, hyöty-kustannus analyysiä tai standardoituja lomakepohjia (Boehm 1991, Kontio 2001).

Ongelman ratkaisulla pyritään löytämään keino, jolla riskin toteutumisen todennäköisyyttä voidaan pienentää. Apuna voidaan käyttää esimerkiksi simulointia tai protoilua. Esimerkiksi, jos työmääräarvioon näyttää liittyvän liian suuria epävarmuustekijöitä, voidaan projekti toteuttaa inkrementaalisesti.

Riskien seuranta jatkuu koko ohjelmistoprojektin ajan. Riskien seurantaa varten on hyödyllistä käyttää sopivia mittareita, jotka on kirjattu riskien kontrollointisuunnitelmaan (Kontio 2001). Seuranta voidaan toteuttaa esimerkiksi seuraamalla projektin kymmentä tärkeintä riskiä. Riskienhallintasuunnitelmaan kirjatuissa seurantapisteissä arvioidaan tärkeimpien riskien tilanne ja niiden tärkeysjärjestys. Projektin edetessä riskien järjestys listalla voi muuttua, osa riskeistä voidaan kenties poistaa ja projektille saattaa ilmaantua uusia riskejä (Boehm 1991).

4.2 Arvioinnin epävarmuustekijät ja niiden huomioiminen arvioinnissa

Ohjelmistoprojektiin liittyy riskejä, jotka kohdistuvat kustannuksiin, aikatauluun, lopputuotteen laatuun ja toiminnallisuuteen (Sallis et al. 1995). Osa riskeistä aiheutuu arviointivälineistä. Kitchenham ja Linkman (1997) jakavat arviointimallin avulla suoritetussa arvioinnissa esiintyvät epävarmuutta aiheuttavat virheet neljään ryhmään: mittausvirheet, mallista aiheutuvat virheet, virheelliset olettamukset ja puitevirheet.

Mittausvirheet syntyvät siitä, että arviointiin vaikuttavat muuttujat ovat luonnostaan epätarkkoja. Esimerkiksi ohjelmiston koon arviointi on aina epätarkkaa. Jonesin (1998) mukaan yleisin virhe arvioinnissa on ohjelmarivien lukumäärään perustuva koon määritys, sillä keskimääräisessä ohjelmistoprojektissa yli puolet työmäärästä on muuta työtä kuin ohjelmakoodin tuottamista. Kemererin (1993) mukaan myös toimintopisteinä laskettu koko on epätarkka ja laskennassa on vähintään 12 prosentin epätarkkuus.

Mittausvirheitä voidaan pienentää, kun ymmärretään mittareiden rajoitukset. Esimerkiksi toimintopistelaskentaa sovellettaessa tulisi Kitchenhamin (1997) mukaan laskentaa yksinkertaistaa siten, että peruslaskennat voidaan kerätä automaattisesti järjestelmäkuvauksesta jo projektin alkuvaiheessa ja perusarvoja ei tulisi laskea yhteen, vaan niistä tulisi muodostaa vektori, jolla kuvataan järjestelmää.

Mallista aiheutuu virheitä, koska empiirinen malli voi aina olla vain abstraktio todellisuudesta (Kitchenham ja Linkman 1997). Mikään arviointimalli ei voi sisältää kaikkia niitä tekijöitä, jotka vaikuttavat työmäärään. Ne tekijät, jotka vaikuttavat työmäärään, mutta eivät eksplisiittisesti sisälly malliin, aiheuttavat mallinnusvirheen.

Mallista aiheutuvia virheitä tulisi korjata arviointiprosessi kautta analysoimalla tehtyjä arvioita ja korjaamalla mallia vastaamaan yrityksen omaa ohjelmistokehitysprosessia. Analysoinnin tuloksena voidaan löytää uusia tuottavuuteen vaikuttavia tekijöitä, joilla mallia voidaan korjata. Kitchenhamin (1997) mukaan yritysten tulisi kehittää omia malleja, jotka hyödyntävät omaa kokemuskantaa.

Virhe olettamuksissa tarkoittaa mallin syöteparametreissä esiintyvää epävarmuutta (Kitchenham ja Linkman 1997). Epävarmuus johtuu siitä, että projektin tilanneattribuuttien määrittely on aina todennäköinen arvio tulevista olosuhteista. Esimerkiksi ohjelmiston koon laskenta perustuu olettamukseen, että kaikki asiakkaan vaatimukset on tarkkaan otettu huomioon. Esimerkiksi projektin tilanneattribuutit saattavat muuttua, jos projektin henkilöillä ei olekaan oletettua kokemusta tai projektin henkilöstö vaihtuu.

Arvioinnin syöteparametrien epävarmuudesta johtuvaa riskiä voidaan tarkastella riskianalyysin avulla käyttäen riskituloa, tällöin riskitulon laajuus ymmärretään työmäärän lisäyksenä. Riskitulo voidaan esittää seuraavan kaavan muodossa (Kitchenham ja Linkman 1997):

$$Kokonaisriski = (E2 - E1)P2 \quad (24)$$

missä *E1* = todennäköinen työmäärä, mikäli alkuperäinen arvio pitää paikkansa, *E2* = todennäköinen työmäärä, mikäli vaihtoehtoinen olettamus toteutuu ja *P2* on todennäköisyys sille, että vaihtoehtoinen olettamus toteutuu.

Oletetaan projektin työmääräarvioksi esimerkiksi 200 henkilötyöpäivää, joka perustuu siihen, että ohjelmiston koko on 1000 toimintopistettä ja tuottoaste 0,2 päivää/toimintopiste. Lisäksi arvioidaan, että on olemassa riski, että ohjelmiston koko laajenee 1100 toimintopisteeksi, jolloin työmääräksi saadaan 220 henkilötyöpäivää. Jos laajuuden kasvamisen todennäköisyydeksi arvioidaan 0,1, voidaan laskea kaavan (24) avulla riskituloksi (220 - 200) x 0,1 = 2 henkilötyöpäivää (Kitchenham ja Linkman 1997).

Taulukko 21: Riskin laajuden arviointi COCOMO-kertoimien avulla.

Attribuutti	*Muutoksen vaikutus kertoimeen*	*Uusi työmäärä*
Sovellusaluetuntemus	1,13 / 0,92 = 1,23	88,3
Suunnittelumenetelmien tuntemus	1,19 / 1 = 1,19	86,6
Yhteisvaikutus	(1,13 * 1,19) / (0,92 * 1) = 1,46	105,1

Toinen tapa huomioida syöteparametrien epävarmuus on käyttää *herkkyysanalyysiä*. Herkkyysanalyysiin voidaan käyttää työmäärän arviointimalleja arvioimalla mallin attribuuttien vaihteluja ja suorittamalla vaihtoehtoisia arvioita (Sallis et al. 1995). Arvioidaan seuraavassa herkkyysanalyysin avulla mahdollinen työmäärän lisäys liitteen 1 esimerkkiaineiston projektille 2. Käytetään perusarviona sovitetun regressiomallin 1 avulla saatua työmääräarviota, joka on 71,9 päivää (taulukko 4). Arvioidaan, että on olemassa riski, että projektin toteutusresursseihin tulee muutos ja muutoksen vaikutuksesta sovellusaluetuntemus heikkenee hyvästä huonoksi ja suunnittelumenetelmien tuntemus heikkenee keskimääräisestä huonoksi. Esimerkkilaskelma riskin vaikutuksesta työmäärään on esitetty taulukossa 21 esittämällä vaikutus työmäärään COCOMO-kertoimien avulla. Uusi työmääräarvio saadaan COCOMO-kertoimien suhteen avulla lasketulla korjauskertoimella. Sovellusaluetuntemuksen muutos yksistään kasvattaisi työmääräarviota 88,3 päiväksi ja muutos suunnittelumenetelmien tuntemisen tasossa 86,6 päiväksi. Molempien tekijöiden yhteisvaikutus muodostuu niiden kertoimien tulona jolloin uusi työmääräarvio olisi 105,1 päivää

eli työmäärän lisäys olisi 33,2 päivää. Arviointimallin käyttökelpoisuus riskien analysointiin on rajallinen, sillä se rajoittuu niihin tekijöihin, joita käytetään mallin attribuutteina (Sallis et al. 1995).

Syöteparametrien epävarmuuden huomioiminen ei vähennä tai lisää mallissa tai mittareissa olevien virheiden aiheuttamaa epävarmuutta. Riskien tarkastellussa on varottava epävarmuustekijöiden kaksinkertaista huomioimista. Näin voi tapahtua, jos mallia korjataan ja sama tekijä huomioidaan myös syöteparametrien epävarmuutena riskitulon avulla. Esimerkiksi projektihenkilöstön kyvykkyyden huomioiminen voi sisältyä malliin tai sitä voidaan tarkastella riskitulon avulla (Kitchenham ja Linkman 1997).

Mallin puitevirheet ovat olettamusvirheiden erikoistapaus. Tällainen virhe voi syntyä, jos arvioitava projekti ei ole mallin puitteissa. Yleinen virhekuvitelma on, että pienen projektin toteutumatietoa voidaan hyödyntää suuren projektin arvioinnissa. Käytännössä suuressa projektissa joudutaan kuitenkin tekemään useampia työvaiheita kuin pientä järjestelmää rakennettaessa. Suuren projektin kustannusrakenne on myös erilainen kuin pienen. Esimerkiksi pienessä n. 100 toimintopisteen projektissa ohjelmakoodin kirjoittamisen osuus on n. 54 % kokonaistyöajasta kun se suurella 10000 toimintopisteen järjestelmällä on vain 18 % (Jones 1998).

Malli on myös asiakasyrityksen toimialasta riippuvainen. Tuottavuuden on havaittu poikkeavan melkoisesti eri toimialojen ohjelmistoprojekteissa. Suomalaisella kokemuskannalla tehdyssä benchmarking tutkimuksessa Maxwell ja Forselius (2000) havaitsivat teollisuuden projektien tuottavuuden olleen 0,377 toimintopistettä tunnissa kun pankki- ja vakuutusalalla tuottavuus oli vain 0,116 toimintopistettä. Tutkimustuloksissa he esittävät myös eri toimialojen ohjelmistoprojektien merkittävimpiä tuottavuuteen vaikuttavia tekijöitä. Eri toimialoilla poikkeavat sekä tuottavuuteen eniten vaikuttavat tekijät, että niiden painoarvo. Arviointimalliksi ei näin ollen kelpaa eri toimialan kokemuskantaan perustuva malli.

Mallin puitevirheiden aiheuttamaa epävarmuutta pienennetään, kun pidetään huolta, ettei samaan malliin sisällytetä esimerkiksi uuskehitysprojekteja, lisäävää ylläpitoa ja korjaavaa ylläpitoa, ellei kokemushistorian analysoinnilla ole havaittu, että ne käyttäytyvät saman mallin mukaisesti (Kitchenham 1997).

Boehmin (1991) mukaan riskienhallinnan tehtävä on tunnistaa projektin kriittiset menestystekijät ja huomio tulisi keskittää näiden tekijöiden seurantaan. Työmäärän arviointiin liittyy niin monia

epävarmuustekijöitä, että riskianalyysi kannattaa kytkeä työmäärän arviointiin ja riskien kontrollointi osaksi projektin hallintaa. Projektien riskienhallinta kannattaa ottaa työmäärän arvioinnin rinnalle omaksi prosessiksi ja kerätä kokemustietoa päättyneistä projekteista tulevia projekteja varten.

Projektia ei voida täysin turvata kaikilta riskeiltä eikä riskienhallintaan ole olemassa valmiita yleispäteviä ratkaisuja. Monimutkaisten henkilöstöön ja teknologiaan liittyvien menestystekijöiden huomioiminen vaatii aina tapauskohtaista harkintaa (Boehm 1991).

5 Yhteenveto

Ohjelmistoprojektin työmäärän arviointi pitäisi ymmärtää prosessina, jossa päättyneiden projektien kokemuskantaa hyödyntäen analysoidaan projektia luonnehtivien attribuuttien ja työmäärän välistä riippuvuutta ja pyritään kehittämään ohjelmistotuotantoprosessia kuvaavia malleja, joiden avulla päästäisiin tarkempiin työmääräarvioihin.

Työmäärän arvioiminen on kuitenkin vaikeaa, sillä ohjelmistotuotannon tuottavuuteen vaikuttavat monet tekijät. Työmääräarvio joudutaan tekemään myös niin aikaisessa vaiheessa, että monista projektin attribuuteista ei ole vielä täyttä varmuutta. Lisäksi luovan suunnittelutyön tuottavuutta on vaikea ennakoida.

Työmäärään eniten vaikuttavana tekijänä pidetään ohjelmiston kokoa, joka voidaan määritellä usealla tavalla. Yleisimmin käytetään joko lähdekoodirivien lukumäärää tai ohjelmiston toiminnallisuutta kuvaavaa toimintopistelaskentaa, joka huomioi myös ohjelmiston monimutkaisuutta. Työmäärään vaikuttavat myös monet muut projektia, ohjelmistotuotantoprosessia, tuotetta ja henkilöstöä luonnehtivat tekijät.

Projektien toteutumatiedoista kerätään kokemushistoriaa, jotta tietoja voitaisiin jälkikäteen analysoida ja ymmärtää paremmin tuottavuuteen vaikuttavia tekijöitä. Kun kokemushistorian perusteella on kyetty muodostamaan teoria työmäärän tai tuottavuuden ja projektin attribuuttien välisistä riippuvuuksista, voidaan rakentaa arviointimalleja ja pyrkiä niiden avulla tarkempiin arvioihin.

Työmäärän arviointimalleja on tutkittu useita vuosikymmeniä ja niitä on rakennettu useilla eri tekniikoilla, jotka voidaan jakaa karkeasti kahteen luokkaan: parametrisiin ja ei-parametrisiin. Parametrisissä tekniikoissa luodaan yhtälö työmäärän ja siihen vaikuttavien attribuuttien välille. Perinteisin parametrinen tekniikka on tilastolliseen analyysiin perustuva regressioanalyysi. Silloin kun riippuvan ja riippumattomien muuttujien välillä ei ole tilastollista riippuvuutta, voidaan käyttää tekoäly-tekniikkaan perustuvia koneoppimisen menetelmiä. Arviointimallien rakentamista on tutkittu ainakin neuroverkoilla, analogia-tekniikalla ja regressiopuilla.

Viime vuosina on tehty paljon tutkimuksia, joissa on pyritty selvittämään, mikä olisi paras tekniikka arviointimallin rakentamiseen. Tutkimustulokset ovat ristiriitaisia mutta myös lohduttomia, sillä näyttäisi siltä, että tutkimuksia varten rakennetuilla arviointimalleilla ei kyettäisi tuot-

tamaan riittävän tarkkoja arvioita. Tekniikoiden lisäksi tutkijat ovat olleet kiinnostuneita vertaamaan yrityskohtaisen ja yritysten yhteisten kokemuskantojen vaikutusta mallin tarkkuuteen. Myös näissä tutkimuksissa tulokset ovat olleet ristiriitaisia. Tutkimustuloksista voidaan kuitenkin päätellä, että tekniikkaa tärkeämpi tekijä onnistuneen mallin rakentamisessa on, millainen kokemushistoria on käytettävissä. Mallin rakentaminen voi onnistua, jos käytettävissä on kokemushistoria, joka on homogeeninen, siinä on riittävästi projekteja ja projektien tiedot on kerätty yhtenäisillä menetelmillä.

Arviointiin liittyy paljon epävarmuustekijöitä, joista voi aiheutua riskejä projektille. Riskienhallinta on osa projektinhallintaa ja sillä pyritään löytämään, analysoimaan, luokittelemaan ja kontrolloimaan projektin riskejä.

Vaikka arviointimallin eivät olekaan täydellisiä, on niistä hyötyä projektipäällikölle. Mallin käyttö systematisoi arviointityötä ja se toimii muistilistana, jotta tärkeät työmäärän vaikuttavat tekijät tulisivat huomioiduiksi. Projektin jälkilaskennan kautta kerrytetään kokemushistoriaa, jonka avulla voidaan analysoida omaa ohjelmistotuotantoprosessia ja ymmärtää sitä paremmin ja kenties päästä jatkossa tarkempiin arvioihin. Projekteja pitää hallita ja ne ovat kurinalaista toimintaa. Arviointimalli voi omalta osaltaan ohjata tähän kurinalaisuuteen. Ohjelmistotuotantoala kuitenkin muuttuu jatkuvasti, joten arviointimallienkin on muututtava. Mallin hyvyydestä riippumatta, tarvitaan arvioinnissa edelleen projektipäällikön kokemusta ja tapauskohtaista harkintaa.

Viitteet

Abran, A., Desharnais, J-M., Oligny, S., St-Pierre, D., Symons, C. , *COSMIC-FFP Measurement Manual. Version 2.1*, 2001. http://www.cosmicon.com (10.12.2001)

Angelis, L., Stamelos, I., Morisio,M., Building a Software Cost Estimation Model Based on Categorical Data. *Software Metrics Symposium*, 2001, 4–15.

Boehm, B.W., *Software Engineering Econimics*. Prentice-Hall, New Jersey, 1981.

Boehm, B.W., Software Risk Management: Principles and Practices. *IEEE Software*, 8(1) (Jan), 1991, 32-41.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., *Classification and Regression Trees*. Wadsforth & Books/Cole Advanced Books & Software, California, 1984.

Briand L.C., El Emam K., Surmann, D, Wieczorek, I, Maxwell, K.D., An Assessment and Comparison of Common Software Cost Modeling Techniques. *Proceedings of the 21st International Conference on Software Engineering*, ICSE 99, 1999, 313 - 322.

Briand L.C., Langley, T., Wieczorek, I., A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. *Proceedings of the 22nd International Conference on Software Engineering*, ICSE 2000, 2000, 377 - 386.

Clark, B.K., Quantifying the Effects of Process Improvement on Effort. *IEEE Software*, 17(6) (Nov/Dec), 2000, 65–70.

Conte, S.D., Dunsmore, H.E., Shen, V.Y., *Software Engineering Metrics and Models*. Benjamin-Cummings, Menlo Park, 1986.

Fenton, N.E., *Software Metrics. A Rigorous Approach*. Chapman and Hall, London, 1991.

Fenton, N.E., Pfleeger, S.L., *Software Metrics. A Rigorous & Practical Approach*. 2nd Edition. PWS Publishing Company, Boston, 1997.

Forselius, P., Ohjelmistojen koon mittaaminen erityyppisissä hankkeissa. *Systeemityö* No 1, 1999.

Garmus, D., Herron, D., *Function Point Analysis. Measurement Practices for Succesfull Software Projects*. Addison-Wesley, Boston, 2000.

Hughes, B., *Practical Software Measurement*. McGraw-Hill, London, 2000.

Jeffery, R., Ruhe, M.,Wieczorek, I., Using Public Domain Metrics to Estimate Software Development Effort. *Proceedings of the 7th International Software Metrics Symposium*, April 2001, 16 - 27.

Jones, C.T., *Estimating Software Costs*. McGraw-Hill, USA, 1998.

Kemerer, C.F., Reliability of Function Points Measurement: A Field Experiment. *Communications of the ACM*, 36(2) (Feb), 1993, 85-97.

Kitchenham, B. A., The Problem with Function Points. *IEEE Software*, 14(2) (Mar/Apr), 1997, 29-31.

Kitchenham, B. A., A Procedure for Analysing Unbalanced Datasets. *IEEE Transactions on Software Engineering*, 24(4) (April), 1998, 278-301.

Kitchenham, B. A., Linkman, S., Estimates, Uncertainty and Risks. *IEEE Software*, 14(3) (May/June), 1997, 69–74.

Koivisto, H., *Sumea laskenta*. Luentoaineisto. Tampereen Yliopisto, 2001.
http://www.ae.tut.fi/courses/fuzzy/ (8.5.2001)

Kontio, J., *Software Engineering Risk Management. A Method, Improvement Framework and Empirical Evaluation.* Suomen Laatukeskus, Helsinki, 2001.
http://lib.hut.fi/Diss/2001/isbn951225655X/ (11.1.2002)

Känsälä, K., Integrating Risk Assessment with Cost Estimation. *IEEE Software*, 14(3) (May/June), 1997, 61–67.

Laininen, P., *Tilastollisen analyysin perusteet*. Otatieto, Helsinki, 2000.

Lim, W., Effect of Reuse on Quality, Productivity and Economics. *IEEE Software*, 11(5) (September), 1994, 23–30.

Mair, C, Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Sheppard, M., Webster, S., *An Investigation of Machine Learning Based Prediction Systems*. Bournemouth University, 1999. http://dec.bmth.ac.uk/ESERG (2.1.2001)

Maxwell, K.D., Forselius, P., Benchmarking Software Development Productivity. *IEEE Software*, 17(1) (Jan/Feb), 2000, 80-88.

Pittman, M., *A Landscape of Risk. Project Risk Management*. Pittman Consulting, 1999. http://www.pmnetwork.com.au/risk.pdf (20.1.2002)

Pfleeger, S.L., *Software Engineering. Theory and Practice*. Prentice Hall International, New Jersey, 1998.

Rimpiläinen, A., Peltonen, M., *Riskien ja kokemustiedon hallinta projekteissa: Menetelmät, Standardit, Ohjelmistot*. VTT Automaatio, Tampere, 2000.

Salford Systems, *Salford Systems White Paper Series*, 2001. http://www.salford-systems.com/whitepaper.html#An (9.5.2001)

Sallis, P., Tate G., MacDonnel, S., *Software Engineering. Practice, Management, Improvement*. Addison-Wesley Publishing Company, Australia, 1995.

Shepperd, M., *Automated Project Cost Estimation : Using Analogies*. The ANGEL Project. 2001, http://dec.bmth.ac.uk/ESERG/ANGEL/ (9.5.2001)

Shepperd, M., Schofield, C., Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(12), (November), 1997, 736-743.

Shepperd, M., Schofield, C. Kitchenham, B., Effort Estimation Esing Enalogy. *Proceedings of the 18th International Conference on Software Engineering*, ICSE 18, 1996, 170-178.

STTF, *Productivity Factors of Experience Situation Analysis*. http://www.sttf.fi/html/1Expeng2sitanalysis.html (2.2.2002)

Liite 1: Esimerkkiaineisto

Projekti	TP	LUOT	SUUN	MENET	SOVMON	VAATIM	SOVTUN	TYOK	TKKOKO	TYOM
1	1010	3	3	2	2	3	2	3	2	287
2	457	4	3	3	2	3	4	3	4	82,5
3	2284	2	3	2	2	2	3	2	2	1107,31
4	881	4	3	3	3	3	4	4	2	86,9
5	1583	3	3	3	2	3	4	3	3	336,3
6	411	3	3	3	3	3	2	4	3	84
7	97	1	2	2	5	3	2	3	4	23,2
8	998	4	3	3	3	3	3	3	3	130,3
9	1554	3	4	3	2	3	4	3	3	116
10	250	3	2	2	4	2	3	3	4	72
11	1603	4	3	3	3	2	3	4	3	258,7
12	724	3	3	2	3	3	3	4	3	230,7
13	705	3	3	3	3	2	3	3	3	157
14	1375	3	3	2	2	3	3	4	3	246,9
15	976	5	3	3	3	2	3	3	4	69,9

TP	Koko toimintopisteinä
LUOT	Ohjelmiston luotettavuusvaatimukset
SUUN	Henkilöston suunnittelumenetelmien tuntemus
MENET	Menetelmien taso ja käyttö
SOVMON	Järjestelmän monimutkaisuus
VAATIM	Vaatimusten vakaus
SOVTUN	Henkilöstön sovellusaluetuntemus
TYOK	Henkilöstön työkalutuntemus
TKKOKO	Tietokannan koko
TYOM	Toteutunut työmäärä

Liite 2: Regressiopuumallin 2 säännöt

```
*Terminal Node 4*/
if
(
    SUUN <= 3.5 &&
    LUOT > 3.5 &&
    LUOT <= 4.5
)
{
    terminalNode = -4;
    mean = 7.38327
}
/*Terminal Node 5*/
if
(
    LUOT <= 4.5 &&
    SUUN > 3.5
)
{
    terminalNode = -5;
    mean = 13.3966
}
/*Terminal Node 6*/
if
(
    LUOT > 4.5
)
{
    terminalNode = -6;
    mean = 13.9628
}
```

Liite 3: Regressiopuumallin 3 säännöt

```
/*Terminal Node 1*/
if
(
    MENET <= 2.5 &&
    TP <= 1192.5
)
{
    terminalNode = -1;
    mean = 3.57767
}
/*Terminal Node 2*/
if
(
    MENET <= 2.5 &&
    TP > 1192.5 &&
    TP <= 1829.5
)
{
    terminalNode = -2;
    mean = 5.56906
}
/*Terminal Node 3*/
if
(
    MENET <= 2.5 &&
    TP > 1829.5
)
{
    terminalNode = -3;
    mean = 2.06266
```

```
}
/*Terminal Node 4*/
if
(
    MENET > 2.5 &&
    TP <= 793
)
{
    terminalNode = -4;
    mean = 4.97423
}
/*Terminal Node 5*/
if
(
    MENET > 2.5 &&
    TP > 793 &&
    TP <= 1568.5
)
{
    terminalNode = -5;
    mean = 11.2892
}
/*Terminal Node 6*/
if
(
    MENET > 2.5 &&
    TP > 1568.5
)
{
    terminalNode = -6;
    mean = 5.45174
}
```