Ohjelmistotuotanto

Vaatimusanalyysi
Ohjelmiston määrittely

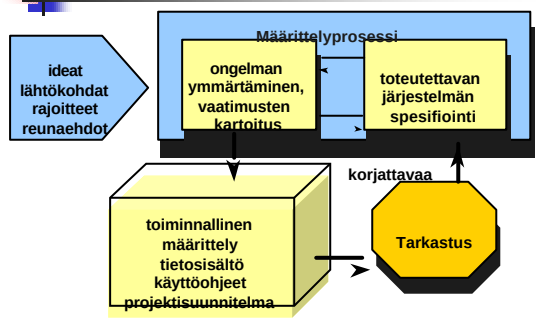
1

Vaatimusanalyysi

- Vaatimusanalyysin tavoitteena on selvittää ohjelmistolle asetettavat vaatimukset niin yksityiskohtaisesti, että niiden perusteella voidaan tuottaa haluttu ohjelmisto.
- Lineaarisissa malleissa työvaiheen lopputuloksena saadaan täydellinen lista vaatimuksista ja analyysi niiden vaikutuksesta lopputulokseen. Iteratiivisissa malleissa saadaan lista tällä iteraatiokierroksella vaikuttavista vaatimuksista.

 2

Vaatimusanalyysi



 3

Vaatimusanalyysi

- Ohjelmiston vaatimusanalyysin edeltäjänä tai sen osana voidaan katsoa olevan *systeemisuunnittelun* (system engineering). Sen tarkoituksena on
  - kartoittaa ohjelmiston rooli yrityksessä, sen sidosryhmät ja vaikutukset
  - tuottaa malli koko yrityksestä tai tuotealueesta ja ohjelmiston asemasta siinä
  - jakaa mahdollisesti järjestelmä laitteistolla ja ohjelmistolla toteutettaviin osiin (ns. “system allocation”)
  - selvittää alueen nykytila ja laatia arvio projektin kaupallisesta ja teknisestä järkevyydestä (ns. “feasibility study”)

 4

Kartoitus

- Ohjelmistolle asetettujen vaatimusten kartoitus
  - selvitetään asiakkaan tarpeet
  - mikä ongelma on ohjelmistolla tarkoitus ratkaista
  - selvitetään halutut palvelut
  - MITÄ asiakas haluaa?
  - asiakas ja tuottaja tekevät kartoituksen yhteistyössä

 5

Määrittely (“speksaus”)

- Ohjelmiston määrittely
  - MITÄ ohjelmisto tarjoaa?
  - tuloksena määrittelydokumentti:
    - sopimus asiakkaan ja tuottajan välillä
    - perusta myöhemmille vaiheille
    - asiakkaan hyväksyttävä
    - oltava sekä asiakkaan että tuottajan ymmärtämässä muodossa

 6

Kartoituksen lähtökohtia

- kuka on asiakas?
- onko käynnissä tarjouskilpailu?
- millaista hyötyä odotetaan?
- millaista tietoa pitäisi käsitellä?
- millaisessa käyttöympäristössä ohjelmiston tulisi toimia?
- onko suorituskykyä mietitty?
- onko asiakkaalla toiveita tai vaatimuksia projektin aikataulusta tai miehityksestä (esimerkiksi projektipäälliköstä)?

Vaatimuksia vaatimuksille

- Dokumentoitavien vaatimusten on oltava:
  - Virheettömiä (correct) eli asiakkaan käsitysten mukaisia. Isoissa järjestelmissä virheettömyyden todentaminen on vaikeaa.
  - Ristiriidattomia (consistent). Vaatimuksia pitää analysoida, jotta löydetään niihin sisältyvät ristiriidat. Ristiriitaisuutta saattaa syntyä kielen moniselitteisyydestä, mutta usein vaatimukset ovat aidosti ristiriitaisia (esimerkiksi “mahdollisimman siirrettävä ja tehokas asiakkaan ympäristössä”).
  - Täydellisiä (complete). Ei jätetä kirjaamatta tiedossa olevia vaatimuksia. Vaatimukset muuttuvat aina projektin elinkaaren aikana. Kun muutokset kohdistuvat jo jäädytettyyn vaatimusdokumenttiin, tarvitaan konfiguraationhallintaa.

Vaatimuksia vaatimuksille

- Jatkuu…
  - Realistisia (realistic). Vaatimusten pitää olla toteutuskelpoisia. Selvitettävä käyttöympäristön ja sidosryhmien asettamat rajoitukset ja suorituskykyvaatimukset.
  - Tarpeellisia (needed). Tarpeellisten vaatimusten määrittely on yllättävän vaikea tehtävä. Vaatimukset kannattaa priorisoida, jolloin on helpompaa vetää raja tarpeellisten ja vähemmän tarpeellisten vaatimusten välille. Todellinen tarve ei välttämättä ole se, minkä asiakas esittää.
  - Todennettavissa (verifiable). Vaatimuksen toteutuminen tuotteessa täytyy olla osoitettavissa. Mitattavuus ja konkreettiset numeroarvoihin perustuvat vaatimukset ovat helpoimmin todennettavissa.

Vaatimuksia vaatimuksille

- Jatkuu ...
  - Jäljitettävissä (tracable). Jokaisen vaatimuksen kohdalla tulee kirjata ylös henkilö tai ryhmä, joka on asettanut tämän vaatimuksen. Näin voidaan tarpeen vaatiessa palata vaatimusketjua takaisin päin vaatimuksen esittäjään.

Vaatimusanalyysin vaiheet

- Kartoitus
  - Tässä vaiheessa selvitetään ongelma-alueet ja yleiset tuotteen komponentit. Osa tästä työstä on tehty jo projekti- ja systeemisuunnitelman yhteydessä.
- Arviointi
  - Kartoituksen perusteella saadaan selkeä kuva tuotteen yleisistä vaatimuksista. Vaatimukset dokumentoidaan ja analysoidaan yksityiskohtaisesti. Uusia vaatimuksia voi ilmetä tässä vaiheessa. Palataan takaisin kartoitukseen, kunnes uusia vaatimuksia ei löydy.
- Priorisointi ja kokoaminen
  - Kun vaatimukset on löydetty, niille annetaan yksikäsitteinen tunniste ja ne luokitellaan prioriteettiryhmiin.

Vaatimusanalyysin vaiheet

- Mallinnus
  - Kun tuotteesta on kerätty riittävästi tietoa, siitä rakennetaan abstrakti malli (jollakin mallinnusmenetelmällä / -kielellä). Mallinnus on selkeä keino kuvata tuotteen tietosisältö, tiedon kulku järjestelmässä sekä järjestelmän yleiset algoritmit ja sidosryhmät. Mallin avulla voidaan edelleen tarkentaa vaatimuksia. Mallinnusvaiheessa saatetaan palata kartoitukseen, jos mallin avulla löydetään puutteita tai virheellisiä vaatimuksia.
- Määrittely
  - Kun edelliset vaiheet on hyväksytty, vaatimusdokumentti voidaan viimeistellä. Tällöin määritellään yksityiskohtaisesti kaikki vaatimukset prioriteetteineen ja vaatimuksista saatu malli.

Vaatimusanalyysin vaiheet

- Jäädytys
  - Vaatimusdokumentti tarkastetaan, hyväksytään ja "jäädytetään" (minkä jälkeen siihen ei saa tehdä muutoksia ilman eksplisiittistä sopimusta asiakkkaan kanssa). Jos dokumentti ei ole riittävän täydellinen, palataan takaisin aikaisempiin osavaiheisiin.
  - Iteratiivisissa, inkrementaalisissa tai XP-projekteissa vaatimusdokumentti elää koko ajan eikä sitä "jäädytetä" koskaan kokonaisuutena vaan pala palalta.

Vaatimusten analysointi

- Vaatimuksia voidaan kartoittaa haastatteluilla, tutkimalla dokumentteja, perehtymällä ohjelmiston ajateltuun toimintaympäristöön, tutustumalla oppikirjoihin ja työoppaisiin, jne.
- On tärkeää huolehtia kommunikoinnin toimivuudesta asiakkaan ja tuottajien välillä. Esitysten havainnollisuuteen ja ymmärrettävyyteen tulee kiinnittää erityistä huomiota. Prototyyppi tai alustava käytön opas voi olla hyvä pohja kommunikoinnille.

Vaatimusten analysointi

- Eräs tapa analysoida vaatimuksia on tarkastella tietojärjestelmää reaktiivisena järjestelmänä. Järjestelmän toiminnot ovat tällöin reaktioita johonkin toimintaympäristön tapahtumaan. Vaatimukset kohdistuvat näihin reaktioihin.
- Kunkin reaktion kohdalla selvitetään
  - mikä tapahtuma aiheuttaa sen, millaisia aiheuttajat ovat
  - miten se vaikuttaa järjestelmään
  - aktivoiko se muita reaktioita
  - mitä sääntöjä siihen liittyy
  - mitä vaatimuksia siihen liittyy

Vaatimusten analysointi

- Vaatimusten löytämisen jälkeen ne priorisoidaan. Myöhemmässä vaiheessa päätetään, kuinka korkean prioriteetin vaatimukset tullaan toteuttamaan ensimmäisessä ohjelmistoversiossa ja kuinka matalan prioriteetin vaatimukset voidaan mahdollisesti jättää lopulta kokonaan toteuttamatta.
- Analyysissa kannattaa varoa spiraali-ilmiötä, jolloin samaa ongelma-aluetta käsitellään uudestaan ja uudestaan yhä pienenevissä kehissä ilman että vaatimusten määrittely etenee oleellisesti.
- Vaatimusanalyysissa kannattaa huomata, että myös sellaiset vaatimukset tulisi saada esille, jotka jäävät yleensä pois "itsestään selvyyksinä".

Vaatimusten analysointi

- Vaatimukset voidaan jakaa kolmeen luokkaan:
  1. Normaalit vaatimukset. Nämä tulevat esille vaatimusanalyysissa ja ne kirjataan vaatimusdokumenttiin. Asiakas *tietää*, että nämä vaatimukset toteutetaan.
  2. Odotetut vaatimukset. Nämä ovat implisiittisesti mukana vaatimuksissa, mutta niitä ei yleensä kirjata. Asiakas *olettaa*, että nämä vaatimukset toteutetaan.
  3. Ekstrat. Näitä ei ole kirjattu vaatimuksiksi, mutta ne ovat mukana lopputuotteessa. Asiakas ei oleta, että näitä tehdään.

Mallinnus

- Järjestelmää kuvaavan käsitteellisen mallin (conceptual model) muodostaminen on määrittelyn keskeinen tehtävä
  - yksinkertaistettu malli
  - käyttäjien ja suunnittelijoiden kommunikoinnin perusta
  - toisaalta täsmällisyys ja toisaalta ymmärrettävyys tärkeää (Huom! Osin ristiriitaiset tehtävät)
- Mallissa pitäisi käsitellä ainakin
  - toimintoja (function)
  - dataa (data)
  - ohjausta (control)
  - käyttäytymistä (behaviour)

Mallinnus

- Järjestelmän mallintamiseen on kehitetty monia menetelmiä
  - perusajatus siitä, mikä mallintamisessa on oleellista, on eri menetelmissä erilainen; menetelmät keskittyvät esimerkiksi joko toimintojen, datan, ohjauksen tai käyttäytymisen kuvaamiseen ja jättävät muut näkökulmat vähemmälle huomiolle
  - kukin menetelmä vaatii käytettävien käsitteiden, symbolien ja merkintätapojen hallitsemisen
  - mikään menetelmistä ei sovellu kaikkiin tilanteisiin, joten jokaisessa projektissa on valittava juuri siihen sopiva mallinnusmenetelmä

 19

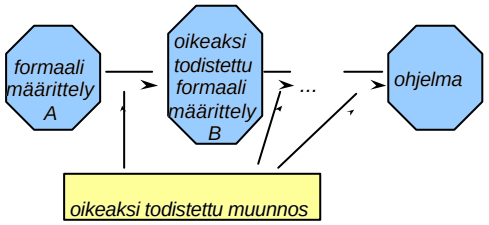
Mallinnus

- Menetelmät käyttävät yleensä graafisia kuvaustapoja
  - kaavioiden ylläpito voi olla hankalaa (automatisointi?)
  - suurien kokonaisuuksien hahmottaminen?
    - yhteen kuvaan ei mahdu kovin paljon asiaa
    - yleiskuvan ja yksityiskohtien esittäminen samassa kuvassa ei toimi
  - kaaviot usein moniselitteisiä ja epätäsmällisiä, jolloin ne jättävät (liian) paljon tulkinnanvaraa myöhemmille projektin vaiheille

 20

Formaalit spesifikaatiot

- Formaali näkemys: ohjelmisto tuotetaan jonona asteittain tarkentuvia spesifikaatioita



 21

Formaalit spesifikaatiot

- Formaali määrittely = ohjelmiston ominaisuudet määrittelevä matemaattinen kaava
- Esim: järjestäminen

järjestetty([x,L]) = (" y Î L): x £ y Ù järjestetty(L).

järjestetty([]).

 22

Formaalit spesifikaatiot – plussia

- täsmällisyys
- tuotettu ohjelmisto vastaa alkuperäistä määrittelyä eli tekee sen, mitä vaaditaan (jolloin ei periaatteessa tarvita testaamista lainkaan)
- automatisointi: spesifikaatiota voidaan analysoida ja havaita sisäiset loogiset virheet
- spesifikaatiota voi osittain suorittaa tai simuloida jo ketjun alussa
- abstraktiotason voi säätää sopivaksi
- erikoiskieliä (VDM, Z, LOTOS, Prolog,...)
- erikoisprosesseja (Cleanroom)

 23

Formaalit spesifikaatiot – miinuksia

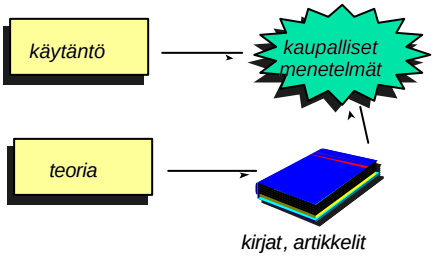
- miten varmistetaan, että ketjun 1. spesifikaatio on “oikein” (suhteessa epäformaaliin asiakkaaseen)?
- miten todistetaan, että muunnosten oikeaksitodistukset ovat oikein?
- entä oikeaksitodistuksen oikeaksitodistukset, jne.

 24

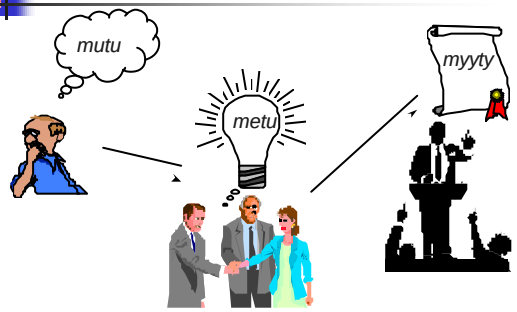
Formaalit spesifikaatiot – miinuksia

- prosessin raskaus
  - lyhyt ketju -> suuria semanttisia välejä spesifikaatioiden välillä -> todistaminen vaikeaa
  - pitkä ketju -> paljon spesifikaatioita, muunnoksia ja todistuksia
- epähavainnollisuus
  - huono kommunikaatioväline
  - onko tämä sittenkin vain notaation ongelma?
- teollisuudelta puuttuva formaali osaaminen
- vähän vakuuttavia näyttöjä
  - käytössä erityisaloilla: tietoliikenne, kääntäjät , avaruusluotaimet

Menetelmien synty



Menetelmien synty



Menetelmien synty

- ***Tarvitaan välineet***

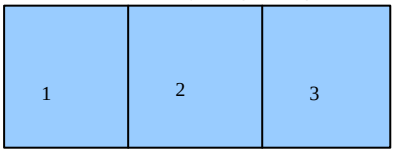
1. tiedon analysointiin
2. toimintojen esittämiseen
3. liittymien kuvaamiseen
4. ongelman osittamiseen
5. abstrahoinnin tukemiseen

Ongelman osiinjako – ositus

- Ositus
  - kokonaisuus jaetaan osiin, toiminnallisuuden perusteella
  - osa voidaan edelleen jakaa pienempiin osiin

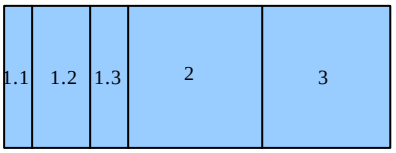
Ongelman osiinjako – ositus

- Ositus
  - kokonaisuus jaetaan osiin, toiminnallisuuden perusteella
  - osa voidaan edelleen jakaa pienempiin osiin



Ongelman osiinjako – ositus

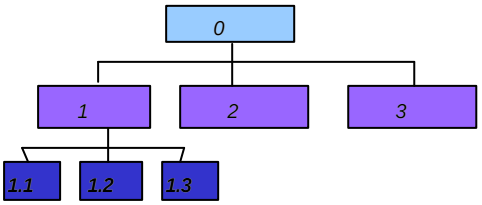
- Ositus
  - kokonaisuus jaetaan osiin, toiminnallisuuden perusteella
  - osa voidaan edelleen jakaa pienempiin osiin



 31

Ongelman osiinjako – ositus

- Muodostuu jakohierarkia
  - kullakin tasolla sama tarkastelukulma ja kuvaustapa, mutta suppeampi kohde



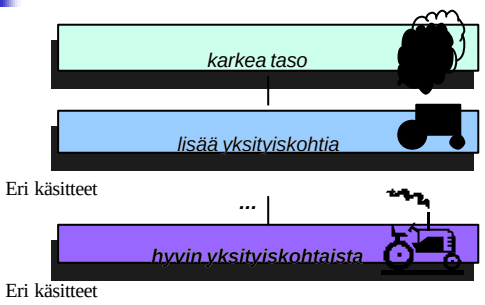
 32

Ongelman osiinjako – abstrahointi

- Abstrahointi
  - järjestelmää tarkastellaan aluksi karkeammalla tasolla
  - lisätään yksityiskohtaisuutta siirryttäessä seuraavalle tasolle
  - tarkastelun kohteena kullakin tasolla koko järjestelmä
  - esim.
    - käsitetaso
    - rakennetaso
    - fyysinen taso
  - kuvaustapa tasoilla voi vaihdella

 33

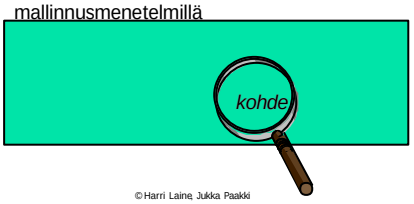
Ongelman osiinjako – abstrahointi



 34

Ongelman osiinjako – projisointi

- Projisointi
  - järjestelmää tarkastellaan jostain tietystä näkökulmasta (esim. arkkitehtuuri, suojaus, käytettävyys,..)
  - eri näkökulmat kuvataan yleensä eri mallinnusmenetelmillä



 35

Näkökulmia mallinnukseen

- toimintokeskeinen
  - syötteistä tulosteita muokkaava systeemi
- tietokeskeinen
  - kohdealueeseen liittyvää tietämystä ylläpitävä ja tarjoava systeemi
- oliokeskeinen
  - olioiden joukkio, joka yhteistyössä toimien tuottaa halutut palvelut (vrt. organisaatioyksikkö)

 36