

Ohjelmistotuotanto

Vaativusanalyysi
Ohjelmiston määrittely

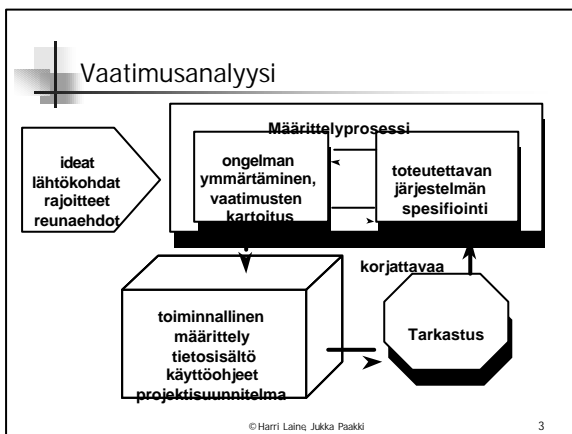
1

Vaativusanalyysi

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

© Harri Laine, Jukka Paakki

2



Vaativusanalyysi

- Ohjelmiston vaativusanalyysin edeltäjänä tai sen osana voidaan katsoa olevan *systemisuunnittelun* (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")

© Harri Laine, Jukka Paakki

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

© Harri Laine, Jukka Paakki

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

© Harri Laine, Jukka Paakki

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)?

© Harri Laine, Jukka Paakki

7

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.

© Harri Laine, Jukka Paakki

8

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. Mittavuus ja konkreettiset numeroarvoihin perustuvat vaatimukset ovat helpoimmin todennettavissa.

© Harri Laine, Jukka Paakki

9

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.

© Harri Laine, Jukka Paakki

10

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 tunnistus ja ne luokitellaan prioriteettiryhmiin.

© Harri Laine, Jukka Paakki

11

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.

© Harri Laine, Jukka Paakki

12

Vaatimusanalyysin vaiheet

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

© Harri Laine, Jukka Paakki

13

Vaatimusten analysointi

- Vaatimuksia voidaan kartoittaa haastattelulla, 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. Prototyypin tai alustava käytön opas voi olla hyvä pohja kommunikoinnille.

© Harri Laine, Jukka Paakki

14

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

© Harri Laine, Jukka Paakki

15

Vaatimusten analysointi

- Vaatimusten löytämisen jälkeen ne priorisoidaan. Myöhemmissä vaiheissa 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".

© Harri Laine, Jukka Paakki

16

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.

© Harri Laine, Jukka Paakki

17

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)

© Harri Laine, Jukka Paakki

18

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 menetelmä ei sovellu kaikkiin tilanteisiin, joten jokaisessa projektissa on valittava juuri siihen sopiva mallinnusmenetelmä

© Harri Laine, Jukka Paakki 19

Mallinnus

- Menetelmät käyttävät yleensä graafisia kuvaustapoja
 - kaavioiden ylläpito voi olla hankalaa (automaatisointi?)
 - 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

© Harri Laine, Jukka Paakki 20

Formaalit spesifikaatiot

- Formaali näkemys: ohjelmisto tuotetaan jonona asteittain tarkentuvia spesifikaatioita

© Harri Laine, Jukka Paakki 21

Formaalit spesifikaatiot

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

$$\text{järjestetty}([x,L]) = (" y\hat{I} L): x\text{ } \& y \hat{U}$$

$$\text{järjestetty}(L).$$

© Harri Laine, Jukka Paakki 22

Formaalit spesifikaatiot – plussia

- täsmällisyys
- tuotettu ohjelmisto vastaa alkuperäistä määrittelyä eli tekee sen, mitä vaaditaan (jolloin ei periaatteessa tarvita testaamista lainkaan)
- automaatisointi: 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)

© Harri Laine, Jukka Paakki 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.

© Harri Laine, Jukka Paakki 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

© Harri Laine, Jukka Paakki 25

Menetelmien synty

© Harri Laine, Jukka Paakki 26

Menetelmien synty

© Harri Laine, Jukka Paakki 27

Menetelmien synty

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

© Harri Laine, Jukka Paakki 28

Ongelman osiinjakoon – ositus

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

© Harri Laine, Jukka Paakki 29

Ongelman osiinjakoon – ositus

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

1	2	3
---	---	---

© Harri Laine, Jukka Paakki 30

Ongelman osiinjakoon – ositus

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

© Harri Laine, Jukka Paakki 31

Ongelman osiinjakoon – ositus

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

© Harri Laine, Jukka Paakki 32

Ongelman osiinjakoon – 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

© Harri Laine, Jukka Paakki 33

Ongelman osiinjakoon – abstrahointi

© Harri Laine, Jukka Paakki 34

Ongelman osiinjakoon – projisointi

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

© Harri Laine, Jukka Paakki 35

Näkökulmia mallinnukseen

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

© Harri Laine, Jukka Paakki 36