

## Dokumentointi ketterissä menetelmissä

- Dokumentointi kuuluu ketteriin menetelmiin niin kuin kaikkeen ohjelmistotuotantoon
- Dokumentointi itsessään yksi vaatimus, jonka prioriteetti pitää arvioida (asiakkaan kanssa)
- Inkrementaalinen ja iteratiivinen lähestymistapa myös dokumentointiin
- Yleensä dokumentointi painottuu projektin loppuun
- Myöhäisellä dokumentoinnilla pyritään takaamaan ettei dokumentoida turhaan

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

123

## Kommunikoinnista

- Ketterissä menetelmissä painotetaan kommunikointia kasvotusten, ei niinkään dokumenttien välityksellä
- Toisaalta dokumentoinnilla voidaan säästää resursseja vähentämällä kommunikoinnin tarvetta
- Brooks' law: "When you add more programmers to a late project, it gets even later." – kirjasta "The Mythical Man-Month" (1975)
- Kun kaikki kommunikoivat keskenään, vuorovaikutuspareja on  $n(n-1)/2$ , eli kommunikoinnin aikavaativuus on  $O(n^2)$

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

124

## Ketterät menetelmät ja vaatimusdokumentti

- Vaatimusdokumentin tuottaminen nähdään riskinä, joka voi johtaa resurssien hukkaamiseen, koska vaatimusten ajatellaan muuttuvan nopeasti  
Vaatimusdokumentti siksi aina vanhentunut
- Vaatimusten "jatkuvaa muuttumista" voidaan liioitellakin
- Johtuuko dokumentoinnin välttäminen aina *todella* vaatimusten muuttumisen uhasta vai siitä että dokumentointi ei ole hauskaa?
- Hyvä tiimi tekee ketterissä menetelmissä dokumentointia *aina kun se on hyödyllistä* (hyödyllisyyttä arvioidaan yhdessä asiakkaan kanssa)

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

125

## Dokumentointi – riski vai resurssi?

- Ketterissä menetelmissä dokumentointi nähdään riskinä, koska voidaan dokumentoida turhaan (ollaanko yhtä huolestuneita siitä, että saatetaan koodata turhaan?)
- Suunnittelukeskeisissä lähestymistavoissa huolellinen dokumentointi nähdään keinona estää resurssien hukkaaminen myöhemmissä työvaiheissa
- Prosessimallin sopivuus ja tarvittavan dokumentoinnin yksityiskohtaisuus riippuu tehtävästä

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

126

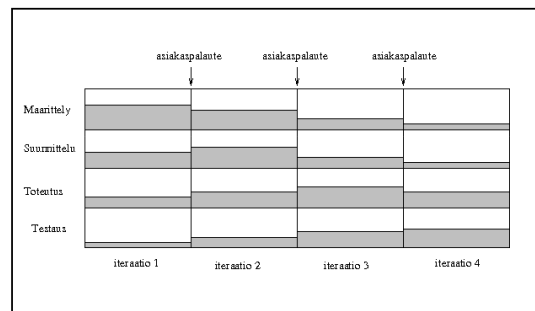
## Vaatimusmäärittely ketterissä menetelmissä

- Ketterissä menetelmissä iteratiivinen lähestymistapa vaatimusmäärittelyyn(kin)
- Vaatimusten etsintä painottuu projektin alkupään iteraatioihin
- Ensimmäiseksi keskitytään
  - Asiakkaalle tärkeimpiin vaatimuksiin
  - Ohjelmiston arkkitehtuurin kannalta keskeisiin vaatimuksiin
- Vaatimukset yleensä valtaosaltaan kiinnittyvät jo alkuvaiheissa projektia

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

127



© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

128

## Vaatimusmäärittelyprosessi

## Vaatimusmäärittelyn perustehtävät

1. Kelpoisuus selvitys
  - Onko toteutettavasta järjestelmä hyötyä?
2. Kartoitus ja analyysi
  - Vaatimusten etsiminen, luokittelu, priorisointi
3. Spesifiointi (määrittely)
  - Vaatimusten esittäminen standardoidussa muodossa
4. Validointi
  - Tarkastetaan määrittelevätkö vaatimukset todella sellaisen järjestelmän, jonka asiakas haluaa

## Vaatimusten muuttuminen

- Mitä isommasta järjestelmästä on kyse, sitä varmemmin vaatimukset muuttuvat:
  - Sidosryhmien tarpeet vaihtelevat
  - Käyttöön oton jälkeen löytyy uusia vaatimuksia
- Toisaalta sanotaan, että suunnittelukeskeinen malli sopii parhaiten suuriin järjestelmiin ristiriita?
- Vaatimusmäärittelyprosessissa täytyy aina jollakin tavalla huomioida muuttuvien vaatimusten vaikutus aikatauluun ja tuotteeseen

## Iteratiivinen vaatimusmäärittely

- Muutosten huomioiminen tarkentamalla vaatimuksia vähitellen spiraali
- Sallii myös vaatimusmäärittelyn rinnalla tehtävän ohjelmiston suunnittelun ja toteutuksen
- Vaatimusmäärittelyprosessi on enemmän tai vähemmän iteratiivinen esim. ketterissä malleissa, prototyypimalleissa ja komponenttimalleissa

## Kelpoisuus selvitys

- Kelpoisuus selvitys (*feasibility analysis*) on lyhyt esivaihe vaatimusmäärittelylle
  - Kelpoisuus selvitysraportti
    - arvioidaan kannattaako järjestelmän kehitystyötä jatkaa
1. Tuoko kehitettävä järjestelmä lisäarvoa asiakkaalle?
  2. Voidaanko järjestelmä toteuttaa nykyisellä teknologialla, projektille varatulla aikataululla ja budjetilla?
  3. Voidaanko järjestelmä integroida jo olemassa oleviin järjestelmiin?
  4. Kannattaako järjestelmä toteuttaa, vai voidaanko vastaava järjestelmä ostaa valmiina? Onko ostettava järjestelmä sovittavissa asiakkaan tarpeisiin?

## Vaatimusten kartoitus ja analyysi

- Yhteistyö asiakkaan ja loppukäyttäjien kanssa
  - mitä palveluja järjestelmältä vaaditaan,
  - mitä järjestelmän suorituskyvyltä vaaditaan,
  - mitä laitteisto- ja ympäristörajoituksia on huomioitava jne.
- Työvaiheessa huomioidaan *sidosryhmät*
- *Sidosryhmä* (stakeholder) on henkilö tai ryhmä, joka suoraan tai välillisesti on tekemisissä kehitettävän järjestelmän kanssa.

## Vaatimusanalyysin ongelmia

- Sidosryhmien edustajat eivät aina tiedä mitä he todella haluavat tai toiveet voivat olla epärealistisia (esim. liian kalliita toteuttaa)
- Sidosryhmien edustajat esittävät asioita käyttäen omaa ammattiterminologiaansa
- Eri sidosryhmillä erilaisia tarpeita vaatimukset voivat olla keskenään ristiriidassa
- Organisaatioon liittyvät ja poliittiset tekijät saattavat vaikuttaa vaatimukseen
- Sidosryhmiinkin voi tulla muutoksia ja sitä(kin) kautta vaatimukset voivat muuttua prosessin aikana

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 135

## Kartoituksen ja analyysin vaiheet

- *Vaatimusten etsiminen*
- *Vaatimusten luokittelu*
  - Järjestämätön vaatimusten joukko ryhmitellään
- *Vaatimusten priorisointi*
  - Vaatimusten tärkeysjärjestys ja konfliktivien vaatimusten käsittely neuvottelemalla sidosryhmien kanssa
- *Vaatimusten dokumentointi*
  - Dokumentti lähtökohta spiraalin seuraavalle kierrokselle

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 136

## Kartoituksen ja analyysin spiraali

- Kartoitus ja analyysi on iteratiivinen prosessi.
- Prosessi aloitetaan isoista vaatimuksista ja edetään kohti pienempiä ja pienempiä yksityiskohtia.

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 137

## Vaatimusten etsimisen tekniikoita

- *Vaatimusten etsiminen: mitä sidosryhmät odottavat järjestelmästä?*
- *Näkökulmat (viewpoints)*
  - mitä suoria tai epäsuoria yhteyksiä sidosryhmillä on järjestelmään?
- *Haastattelut (interviews)*
  - sidosryhmiltä selvitetään keskustelemalla ja sopivin kysymyksin, mitä he odottavat järjestelmästä
- *Dokumentaatioon tutustuminen*

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 138

## Etsimistekniikoita 2

- *Skenaariot (scenarios) ja käyttötapaukset (use cases)*
  - kirjataan suoraviivaisia kuvauksia tulevan järjestelmän käytöstä, skenaarioita
  - toisiinsa sidoksissa olevat skenaariot ryhmitellään tarvittaessa yhteen käyttötapauksiksi
  - niin suunnittelukeskeisissä kuin ketterissä menetelmissä paljon käytetty tapa
- *Etnografia*
  - Loppukäyttäjien havainnointi todellisessa työskentely-ympäristössä

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 139

## Näkökulmat

- Tapa jäsentää vaatimuksia tarkastelmalla järjestelmää eri sidosryhmien näkökulmista
- Suorat näkökulmat
  - Ihmiset tai toiset järjestelmät jotka suoraan tekemisissä järjestelmän kanssa
- Epäsuorat näkökulmat
  - Sidosryhmät jotka eivät itse käytä järjestelmää, mutta vaikuttavat vaatimukseen
- Ympäristöön liittyvät näkökulmat
  - Ympäristön piirteet ja rajoitukset, jotka vaikuttavat vaatimukseen

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 140

## Sosiaaliset ja organisaatioon liittyvät tekijät

- Järjestelmää käytetään jossakin sosiaalisessa ympäristössä ja organisaatiossa
- Näihin liittyvä, usein vaikeasti esille saatava tieto voi vaikuttaa oleellisesti vaatimusten määrittelyyn onnistumiseen
- Jos oleellisia tekijöitä jää piiloon, järjestelmää ei välttämättä todellisuudessa käytetä
- Nämä tekijät eivät ole yksi näkökulma muiden joukossa, vaan ne vaikuttavat kaikkiin näkökulmiin
- Hyvän vaatimusten analyysoijan pitäisi olla sensitiivinen näille tekijöille, ei kuitenkaan olemassa mitään systemaattista tapaa niiden analyysoimiseksi

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 141

## Etnografia

- **Idea ohjelmistotuotannossa:** asiakas ei välttämättä tiedä, tiedosta tai osaa pukea sanalliseen muotoon kaikkia loppukäyttäjille tarpeellisia vaatimuksia mennään paikan päälle katsomaan, miten tulevan järjestelmän loppukäyttäjät työskentelevät
- Etnografia-käsite kulttuurintutkimuksesta: *kulttuurin kuvaaminen* sisältä päin, kulttuuriin liittyvien *merkitysten ymmärtäminen*
  - Historiallisena lähtökohtana vieraiden ("alkukantaisina" pidettyjen) kansojen tutkimus 1900-luvun alussa
  - Perusmenetelmä *osallistuva havainnointi*

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 142

## Etnografia (2)

- Vaatimusten pohjana siis tapa, jolla ihmiset todellisuudessa työskentelevät vastakohtana sille, miten heidän "virallisesti" kerrotaan tai ajatellaan työskentelevän
- Osallistuvan havainnoinnin ongelmana voi joskus olla se, että havainnointi saattaa vaikuttaa ihmisten työskentelytapoihin

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 143

## Haastattelu/keskustelu koskien vaatimuksia

- Sidosryhmien haastattelu koskien nykyistä sekä kehitettävää järjestelmää
- Suljetut (strukturoidut) haastattelut
  - Etukäteen määritelty joukko kysymyksiä sidosryhmien vastattavaksi
- Avoimet haastattelut
  - Etukäteen määritelty korkeintaan aihepiirit, joista keskustellaan vapaasti sidosryhmien kanssa

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 144

## Hyvä haastattelija

- Keskitty kuuntelemaan
- Avoin ja vastaanottavainen: vaatimukset tulevat haastateltavalta
- Ei kannata kysyä: "Mitä haluatte järjestelmältä?"
  - Fokusoidumpia kysymyksiä ja ehdotuksia
  - Haastateltavan sanomien asioiden tarkentamista sopivilla lisäkysymyksillä

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 145

## Skenaariot

- *Skenaariot* ovat todellisia esimerkkejä siitä, kuinka järjestelmää käytetään
- Niihin tulee sisältyä
  - Lähtötilanteen kuvaus
  - Normaalin (=onnistuneen) käyttötilanteen etenemisen kuvaus
  - Kuvaus vaihtoehtoisista tilanteista, joissa jokin menee pieleen
  - Kuvaus samanaikaisista tapahtumista
  - Lopputilan kuvaus

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011 581259 Ohjelmistotuotanto 146

## Käyttötapaukset

- *Käyttötapaukset* (use cases) ovat skenaarioihin pohjautuva UML-tekniikka, jolla määritetään erityyppiset järjestelmän käyttäjät ja käyttötilanteet
- Käyttötapausten joukon pitäisi kuvata kaikki mahdolliset tavat käyttää järjestelmää
- UML-sekvenssidiagrammeja voidaan käyttää kuvaamaan käyttötapausten yksityiskohtia

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

147

## Vaatusmäärittely XP:ssä

- Asiakas mukana tiimissä ja vastaa vaatimuksia koskevista päätöksistä
- Vaatimukset esitetään yleensä korteille kirjoitettavina *käyttäjäkertomuksina* (user stories)
- Kertomukset lähellä käyttötapausta, mutta ilmaistu luonnollisella kielellä, usein vain muutamilla lauseilla

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

148

## Vaatusmäärittely XP:ssä

- Tiimi muokkaa kertomuksesta joukon (*toteutus*)tehtäviä
- Tiimi arvioi tehtävien vaatiman ajan ja kustannukset
- Asiakas valitsee prioriteettien ja aikatauluarvion perusteella kertomukset, jotka toteutetaan seuraavassa syklissä

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

149

## Vaatimusten validointi

- Todennetaan että vaatimuksilla kuvattu järjestelmä on sitä, mitä asiakas haluaa
- Validointi on erityisesti suunnittelukeskeisissä prosessimalleissa erittäin tärkeää, sillä virheelliset vaatimukset heijastuvat koko tuotteen elinkaaren ajan
- Mitä myöhemmin virheellinen vaatimus keksitään, sitä kalliimpaa sen korjaus on

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

150

## Vaatimusten validointi

- Verifioituvuus
  - Voidaanko vaatimusten toteutumista testata?
- Ymmärrettävyys
  - Onko vaatimukset ymmärretty oikein?
- Jäljitettävyys
  - Onko jokaisen vaatimuksen esittäjä ja sidosryhmä tiedossa?
- Mukautuvuus
  - Voidaanko vaatimusta muuttaa ilman laajoja muutoksia muihin vaatimuksiin?

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

151

## Vaatimusten validointi

- *Yhtenäisyys*
  - Onko vaatimukset kuvattu johdonmukaisella tarkkuudella ja tekniikoilla?
- *Täydellisyys*
  - Kuvaavatko vaatimukset koko järjestelmän?

© Juha Taina, Marko Salmenkivi ja Kjell Lemström, 2006-2011

581259 Ohjelmistotuotanto

152

## Vaatimusten validointi: tekniikoita

- Vaatimusten katselmukset
  - Systemaattinen manuaalinen vaatimusten analysointi
- Prototyypit
  - Suorituskelpoisen prototyypin käyttäminen vaatimusten tarkistamiseen
- Testitapausten generoiminen vaatimuksille