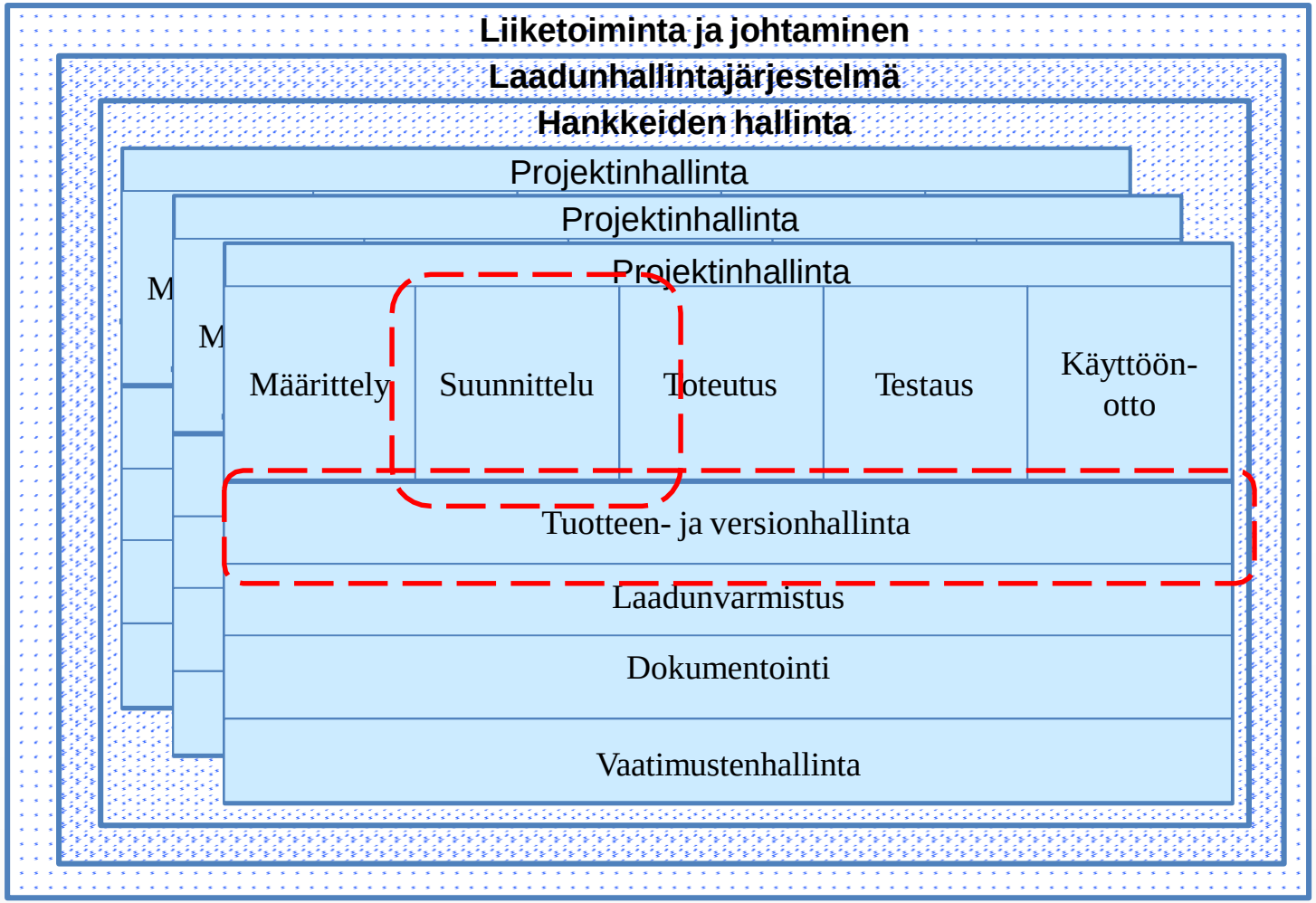
LUT
University

Ohjelmistotuotanto – CT60A4002

Luento 10

Jussi Kasurinen

Ohjelmistotuotannon osa-alueet (Kuva 2.1)

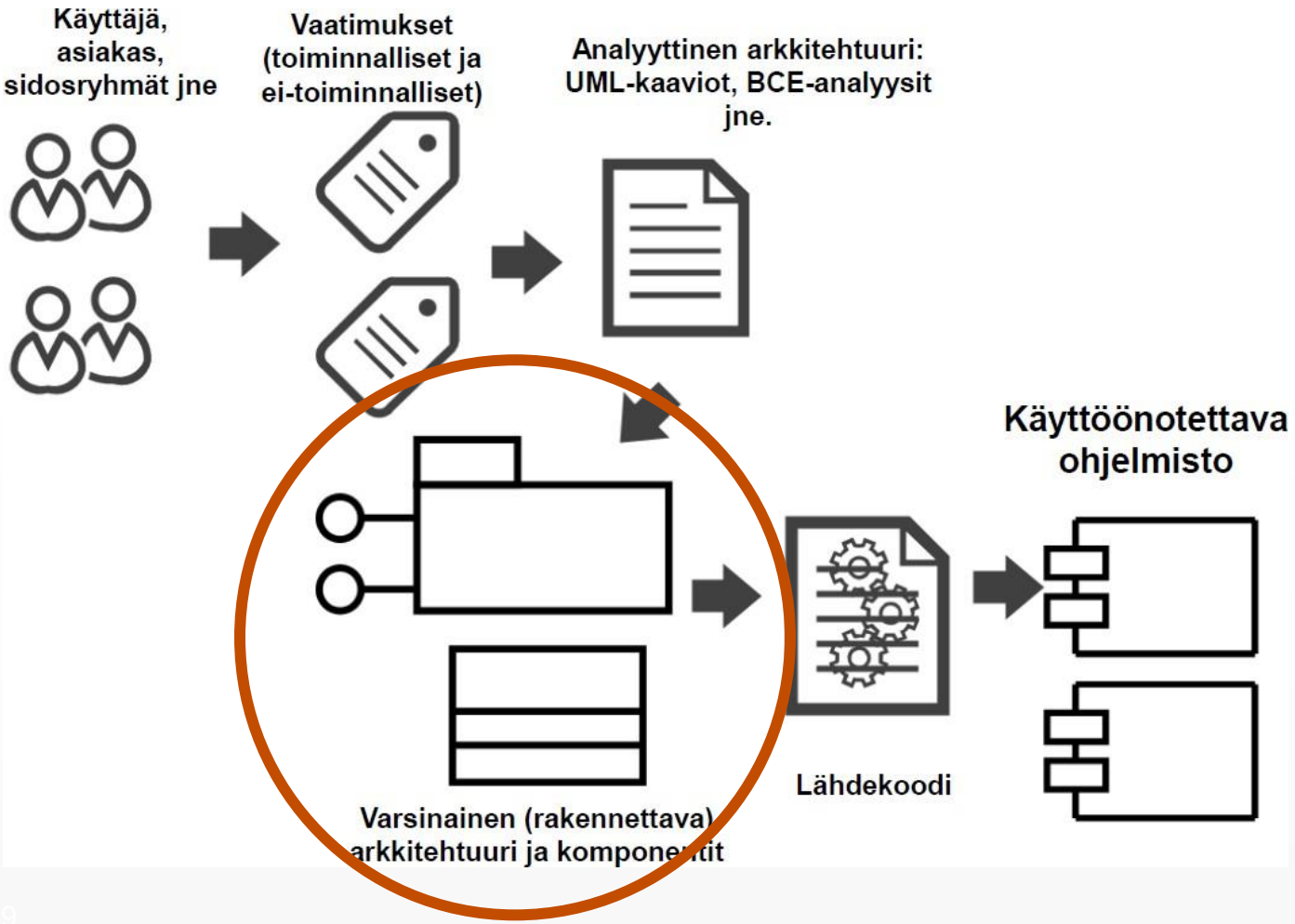


Ohjelmiston arkkitehtuuri ja sen suunnittelu

Mistä on kyse?

Miksi ohjelmistosuunnittelu on tärkeää?

Määrittelystä toteutukseen

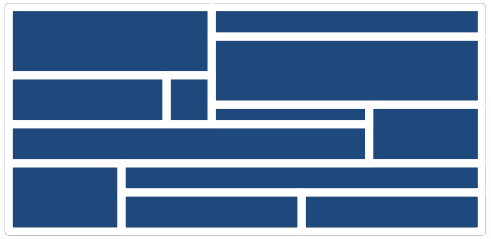


Määrittely – arkkitehtuurisuunnittelu – yksityiskohtainen suunnittelu

Määrittely

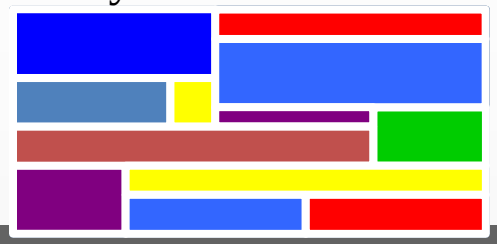
Tuote asiakkaan kannalta

Arkkitehtuurisuunnittelu



Jako komponentteihin
Rajapintojen määrittely

Yksityiskohtainen suunnittelu



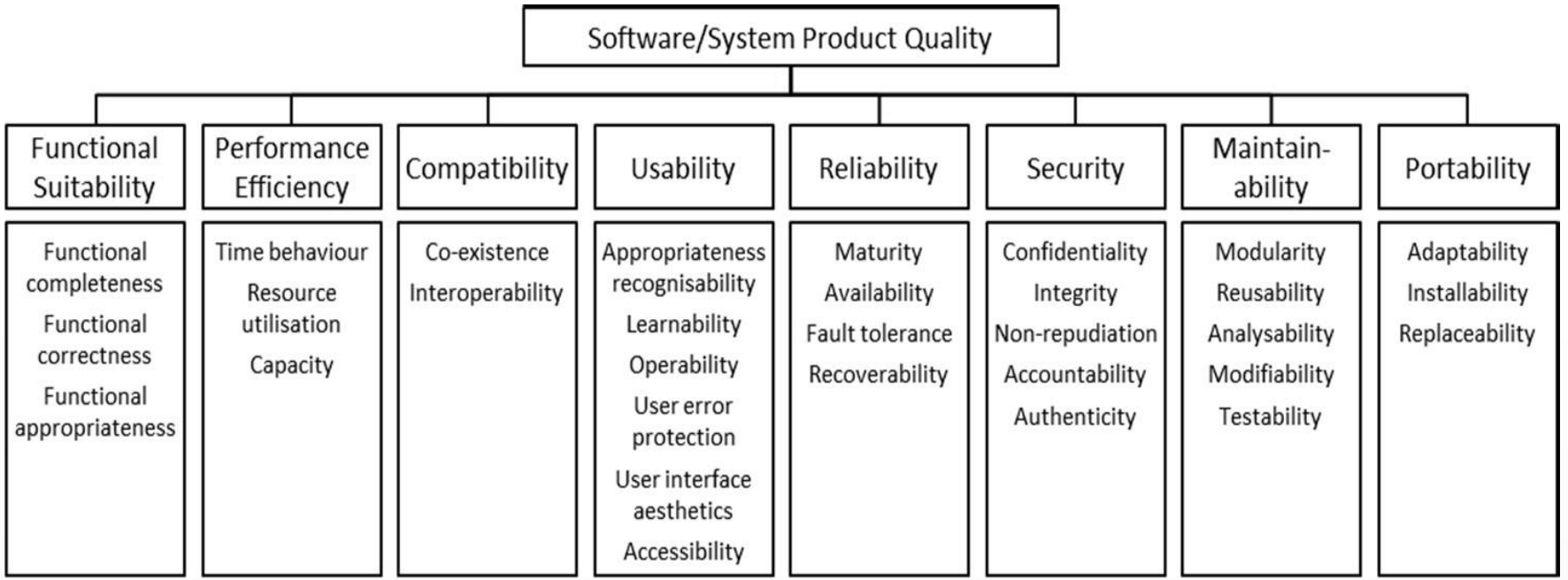
Komponenttien suunnittelu ja toteutus

Hyvän suunnitelman tunnusmerkit

- Suunnitelman tulisi
  - kuvata arkkitehtuuri, joka
    - noudattaa tunnettuja arkkitehtuureja (esim. suunnittelumallit)
    - koostuu komponenteista, jotka noudattavat hyviä periaatteita (esim. modulaarisuus)
    - voidaan toteuttaa evolutionaarisesti
  - olla modulaarinen
    - kuvata erikseen tieto, arkkitehtuuri, rajapinnat ja komponentit
    - johtaa tietorakenteisiin, jotka ovat sopivia toteutettaville luokille
    - johtaa komponentteihin, jotka esittävät itsenäisiä toiminnallisia kokonaisuuksia
    - johtaa rajapintoihin, jotka poistavat liitäntöjen mutkikkuutta
    - olla kuvattu helposti ymmärrettävällä esitystavalla

ISO/IEC 25000, Systems and software Quality Requirements and Evaluation (SQuaRE)

- Toiminnalliset ja ei-toiminnalliset vaatimukset
- Tuotteen vaatimukset



Suunnittelun peruskäsitteet 1/2

- Abstraktio
  - Keino hallita mutkikkuutta
    - Esim. avata ovi, ovi
- Arkkitehtuuri
  - Rakenne, jolla voidaan lisätä järjestelmän käsitteellistä yhdenmukaisuutta
- Suunnittelumallit
  - Mahdollistavat kokeiltujen ratkaisujen käytön suunnittelussa
- Ositus (separation of concerns)
  - Vaikean ongelman ratkaisu on helpompaa, kun sen jakaa osiin
- Modulaarisuus
  - Tekee ohjelmasta hallittavan
- Tiedon kätkentä
  - Moduulien välinen kommunikointi tapahtuu hallitusti rajapintaa käyttäen, moduulin sisältöä ei näytetä
- Toiminnallinen riippumattomuus
  - Saavutetaan korkean koheesion ja alhaisen kytkennän avulla
    - Koheesio = komponentti tai luokka kapseloi vain sellaiset attribuutit ja operaatiot, jotka liittyvät läheisesti toisiinsa ja itse luokkaan tai komponenttiin
    - Kytkentä/Kytkeytyneisyys = komponenttien ”seurustelu”, esim. globaalin muuttujan käyttö, operaatio A herättää operaation B

Suunnittelun peruskäsitteet 2/2

- Askeleittain tarkentaminen
  - Esimerkki top-down suunnittelustrategiasta
  - Abstraktio ja tarkentaminen ovat toisiaan täydentäviä
- Piirteet (aspects)
  - Liittyy osittamiseen ja asetettuihin vaatimuksiin
  - Esim. jos kaksi vaatimusta leikkaavat jonkin piirteen osalta, tämä piirre kannattaa toteuttaa omana moduulina
- Refaktorointi
  - Yksinkertaistaa suunnitelmaa tai koodia muuttamatta itse toimintaa ja käyttäytymistä (tutkitaan käyttämättömät elementit, päällekkäisyys, tehottomat tai tarpeettomat algoritmit, huonot tietorakenteet)
- Suunnittelutason luokat
  - Suunnittelija luo tyypillisesti viidentyyppisiä luokkia
    - Rajapintaluokat, liiketoiminta-alueen luokat, prosessiluokat, talletusluokat, järjestelmäluokat

Suunnitteluperiaatteet

- Ohjelmistotuotanto on aina oppimis- ja kommunikointiprosessi, vaikka keskiössä on tekninen ongelma
  - Korostuu eritysesti ohjelmiston koon kasvaessa
- Käsitteet keskiössä
  - Sovellusalueeseen liittyvät käsitteet: Tunnistetaan ja nimetään vaatimusmäärittelyn yhteydessä
  - Ratkaisuun liittyvät käsitteet: Tulevat ohjelmistotuotannosta ja ohjelmoinnista
    - Usein ei vastinetta ongelma-/sovellusalueella
    - Esim. tietokanta, tiedosto, muistialue, sanomajono, jne.
- Keskeisiä periaatteita selkeisiin ja ymmärrettäviin ratkaisuihin pyrittäessä ovat mm.
  - Yksinkertaisuus ja suoraviivaisuus
  - Osittaminen
  - Lokaalisuus
  - Abstraktioiden hyödyntäminen
  - Rajapinnat
  - Yhdenmukainen toteutusfilosofia

Arkkitehtuurityylejä

- Abstrakteja kuvauksia
- **Auttaa hahmottamaan kokonaisuutta**
- Monet käytännön yksityiskohdat määrittelemättä
- Esimerkiksi
  - Monoliittinen
    - ”Kaikki yhdessä palikassa”-malli
  - Kerrosarkkitehtuuri
    - Asiakas-palvelin (client-server)
  - Tietovarastoarkkitehtuurityylit
    - Kuvataan tietokannat ja tietokantoja käyttävät ohjelmat
  - Ryhmitteleviä tyylejä
    - Esim. pääohjelma/aliohjelma
  - Palveluperustaiset (SOA)
    - Palvelun tarjoaja ja käyttäjä
  - Peer-2-Peer-mallit
  - Mikropalveluarkkitehtuuri

Suunnittelun eteneminen

Rajapinnat
- Komponentin tarjoamat palvelut, oletukset ympäristöstä ja käyttäjän toiminnasta, toteutuksen asettamat rajoitteet
- Tasapainoilua nopeuden ja monikäyttöisyyden välillä

Suunnittelun eteneminen
- Yleiset periaatteet, filosofia
- Perusrakenne komponenttitasolla
- Komponenttien tietosisällöt
- Komponenttien rajapinnat
- Komponenttien näkyvyysalueet
- Testataan suunnitelmat
- Komponenttien sisäinen toteutus
- Kriittisten kohtien toimivuuden varmistus prototyypeillä
- **Suunnitelmien dokumentointi**

Komponentteihin jako

- Arkkitehtuurisuunnittelun yksi perustehtävä on jakaa ohjelmisto pienempiin osiin eli moduuleihin / komponentteihin ja niiden välisiin sidoksiin sekä rajapintoihin
- Moduuli
  - Moduulin tavoitteena on muodostaa selkeä itsenäinen kokonaisuus, joka on helppo ymmärtää – yhtenäisyys eli koheesio
  - Ohjelma voidaan jakaa esimerkiksi kolmeen osaan/moduulityyppiin (tähän palataan uudelleen myöhemmin):
    1. Käyttöliittymä
    2. Tietovarasto
    3. Laskenta / Liiketoimintaan liittyvät tiedot ja ominaisuudet
  - Moduulien väliset riippuvuudet eli kytkennät (coupling) pyritään minimoimaan, jotta moduuleja voisi käyttää hyväksi myös muissa yhteyksissä – uudelleenkäyttö, reuse

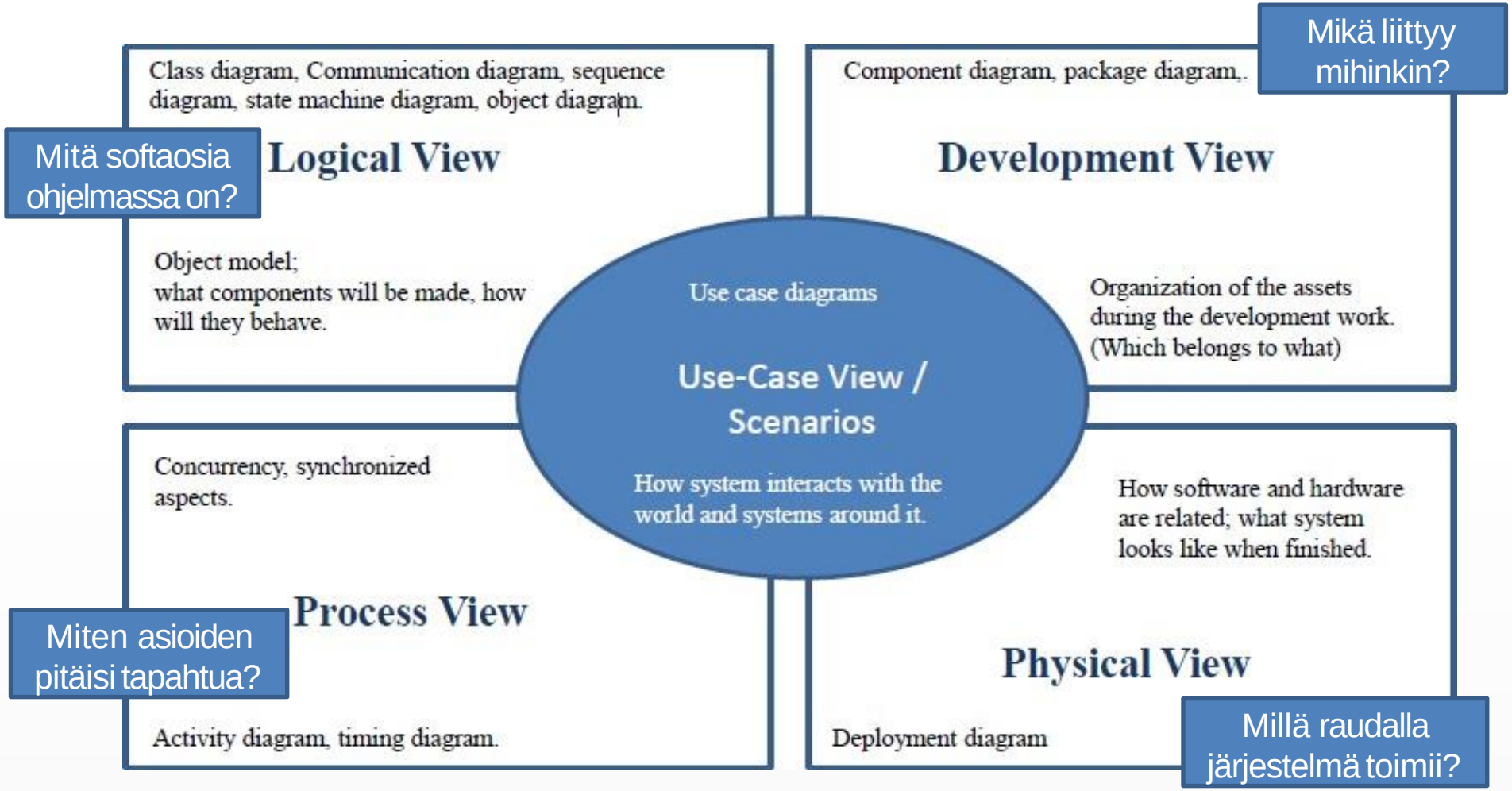
Arkkitehtuuri

- Määritelmiä
  - Komponentit ja niiden väliset suhteet
  - Usein myös suuntaviivat jatkokehitykselle ja suunnitteluperiaatteet
  - Se osa ohjelmistoa, joka ei muutu normaalin ylläpidon aikana
- IEEE 1471-2000:
  - *Software architecture: The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution*.

Ohjelmiston kuvaaminen

- Ohjelmiston kuvaaminen yksikäsitteisesti ei ole helppoa
  - Vrt. talo ja piirustukset: sähkö-, LVI-, pohja-, … kuvat
- 4+1 näkymää ohjelmistoon (Kruchten 1995):
  - Looginen näkymä: toiminnallisuus
  - Toteutusnäkymä: toteutuksen rakenne, jako tiedostoihin
  - Prosessinäkymä: missä prosesseissa laskenta tapahtuu
  - Sijoittelunäkymä: ohjelmiston osien sijoittelu fyysisille
  - +1 eli käyttötapaukset: käyttäjien näkemys ohjelmistosta, idean ydin

4+1 view into architecture



Suunnittelumallit, design patterns

- Samantapaiset ratkaisut luokkarakenteen ja luokkien yhteistyön suunnittelussa toistuu → Hyväksi havaittujen mekanismien kuvaus
- Eräiden arvioiden mukaan suunnittelumalleilla voidaan kuvata normaalitapauksessa 80% koko ohjelmistosta
- Tyypillinen suunnittelumallin kuvaus sisältää
  - Nimi
  - Ongelma
  - Ongelmayhteys
  - Ratkaisu
  - Seuraukset
  - Soveltuvuus
- Malli kertoo, mitä luokka tarvitaan, mitkä ovat niiden väliset suhteet ja miten ne kommunikoivat → luokkien toteutus erikseen

Sovelluskehys, Application Framework

- Valmis toteutus perusarkkitehtuurille → arkkitehtuuri kiinnitetty
- Luokka-, komponentti- ja/tai rajapintakokoelma
- Perusratkaisuja ei tarvitse suunnitella
- Ohjelmistorunko, jossa aukkoja
- Sovelluskehyksen oikeaoppinen käyttö vaatii kehyksen arkkitehtuurin syvällistä tuntemista

Uudelleenkäyttö

- Ohjelma ei kulu käytöstä -> uudelleenkäyttö suositeltavaa

- Yleiskäyttöiset komponentit
  - Käyttöliittymän rakentamisen kirjastot, matematiikkakirjastot, tietorakennekirjastot
- Sovellusaluekohtaiset komponentit
- Sovelluskohtaiset komponentit

- Tuottavuus ja laatu paranevat?
  - Yleiskäyttöisyyden luomiseen sitoutuva työ
  - Muutokset
  - Komponenttien löytyminen
  - Komponentin tehokkuus
  - Komponentin monimutkaisuus / perehtymiseen vaadittava aika

Tuoterunkoarkkitehtuuri (Product Line Architecture, PLA)

- Laajasti käytössä tuotevarianttien teossa
- Uudelleen käyttö
- Yksi runko – useita ratkaisuja, vrt. autonvalmistaja
- Vertaa MVP, minimum viable product-lähestymistapa

Miksi puhua arkkitehtuurista tai tuoterungoista?

Päivitys 09.11.2020: Konetyyppi on **edelleen** lentokiellossa ilman tarkkaa aikataulua käyttönotolle; “early 2021, maybe”

Päivitys 15.11.2021: Kone sai lentoluvan alkuvuodesta, ja lensi seuraavan kerran 11. helmikuuta 2021 ollen melko tasan 23 kuukautta käyttökiellossa.

"These guys didn't even know the damn system was on the airplane, nor did anybody else,“ – Michael Michaelis

Kuva: Wikipedia, Acefitt
CC BY-SA 4.0

Tuotteenhallinta

Mistä on kyse?

Miksi tuotteenhallinta on tärkeää?

Käsitteitä

- Komponentti
  - Ohjelmatiedostot, lähdekieliset
  - Dokumentit
  - Johdettu komponentti (käännetty ohjelma) -> työkaluohjelmat parametri- ja komentotiedostoineen mukaan tuotteenhallintaan
- Konfiguraatio
  - Komponenteista muodostettu kokonaisuus
- Hallinta-alkio (komponentti/konfiguraatio)
- Versio
  - ”jäädytetty” hallinta-alkio
- Vaihetaso / Baseline
  - Viimeisin jäädytetty versio

Tuotteenhallinta

Komponentit

- Versiointi: Mitä versioita on olemassa, miten vanhoihin versioihin päästään käsiksi, …
- Identifiointi: Mikä komponentti tämä on, mitä ominaisuuksia sillä on,…
- Tuottaminen: Millä työkalulla komponentti tuotetaan (esim. kääntäjän versio ja käännöskomento)
- Muutosten hallinta: Miten estetään samanaikainen muutosten teko komponenttiin, mitä muutoksia tehty, …

Konfiguraatiot

- Versiointi: Mitä versioita on olemassa, miten vanhoihin versioihin päästään käsiksi (esim. tuottamalla ne uudelleen), …
- Identifiointi: Mikä konfiguraatio tämä on, mitä komponentteja ja komponenttien versioita on asiakkaan x järjestelmän tietyssä versiossa ,…
- Tuottaminen: Miten rakennetaan asiakkaan x konfiguraatio a.b.c
- Muutosten hallinta: Mihin komponentteihin ja niiden versioihin ehdotettu muutos vaikuttaa, mihin konfiguraatioihin muutos vaikuttaa, …

Toimintatavat

- Vastuut ja toimintavaltuudet
- Miten vaihetuotteet siirtyvät järjestelmästä toiseen
- Miten uudet versiot hyväksytään ja julkistetaan
- Miten muutosesitykset ja virheraportit tehdään ja käsitellään
- Miten arkistointi ja varmuuskopiointi hoidetaan
- …

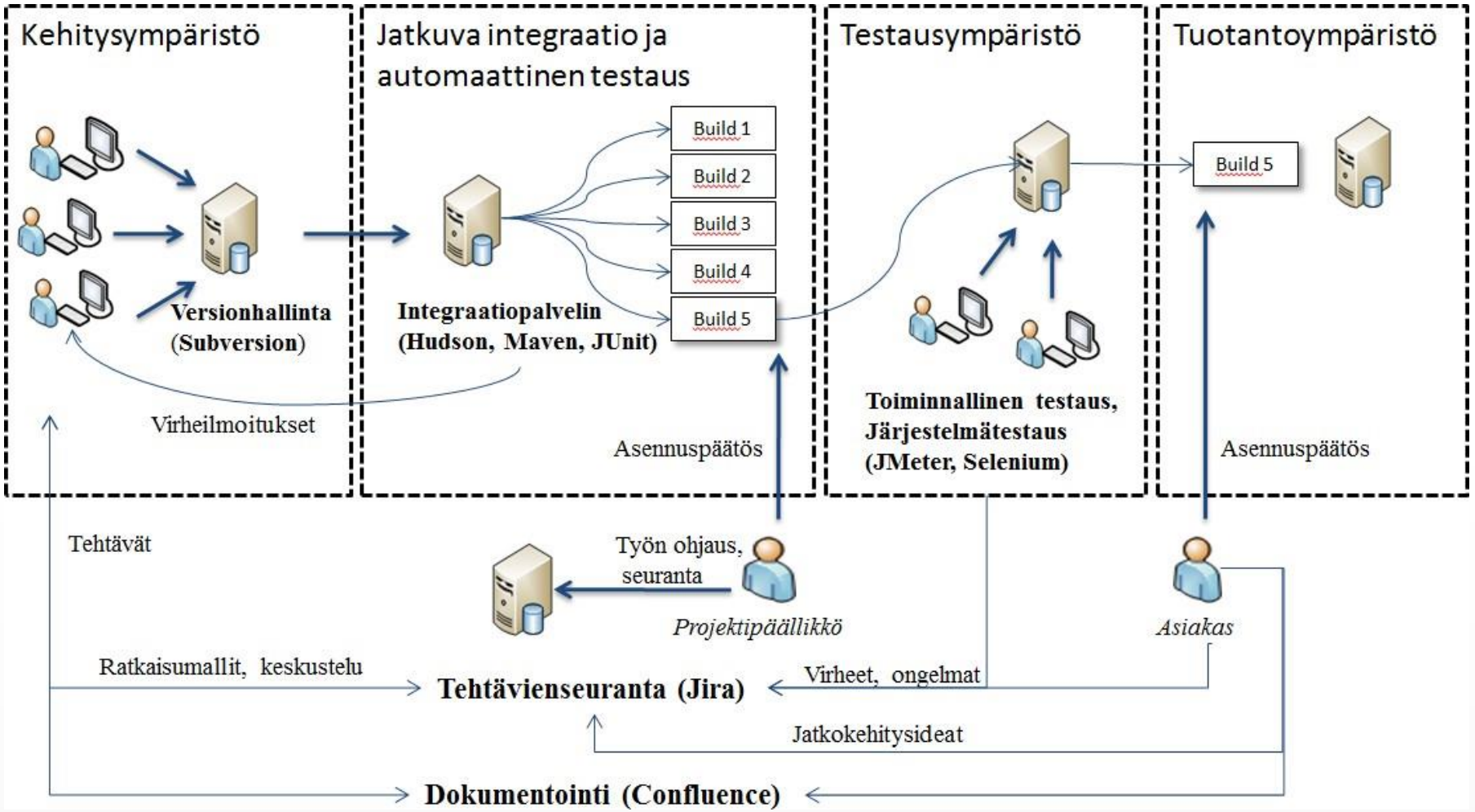
Tuotteenhallinta

- Menetelmät, joilla saman komponentin eri versiot hallitaan
- Menetelmät, joilla konfiguraatioita ja niiden versioita hallitaan
- Versioita ja konfiguraatioita luotaessa ja muutettaessa noudatettavat toimintatavat

Versionhallinta

- Perusta
  - Kehittäjälle stabiili työympäristö
  - Samanaikaisen saman moduulin muokkaamisen estäminen / samanaikaisen moduulin käytön estäminen
  - Hyväksytty komponentti tuotteenhallintaan
  - Muutokset hyväksyntäprosessin kautta
- Versionhallinnan tietoja
  - Versionumero
  - Hallinta-alkion vastuuhenkilö
  - Hallinta-alkion tila (ei aloitettu, aloitettu, valmis)
  - Tilan muuttumispäivämäärät
  - Testiympäristöt, testitapaukset versionumeroineen
  - Ohjelmisto- ja laitteistokokoonpano
  - Käytettyjen työkalujen versionumerot

Asiakasprojektin työnkulku ja työkalut



Versiopuu

- Tavoitteena vain yksi haara
  - Hallittavuus
  - Testaaminen
  - Haarautuminen -> paluu samaan versiopuuhun
- Revisio, peräkkäinen versio
- Logiikka esim.
  - Suurempi muutos 1.x ->2.0
  - Uudet piirteet 1.05 -> 1.06
  - Virheiden korjaus 1.05.00 -> 1.05.01
  - Sisäinen, ei asiakkaalle näkyvä versiointi 1.05.01.001
- Versionhallintaohjelmiston käyttö

Ratkaistavia kysymyksiä

- Mihin muutokset vaikuttavat?
- Millainen konfiguraation on tietyllä asiakkaalla?
- Mikä on komponenttien versioiden yhteensopivuus?
- Miten tietty ympäristö rakennetaan uudelleen?

Jatkuva integrointi (Continuous Integration)

- Kaikki kehittäjät toimivat käytännössä saman version kanssa
- Muutokset pieniä, jatkuva tallennus versionhallintaan -> integrointitestaukseen
  - Testauksen automatisointi
  - Testien optimointi
- Jos käännös / testi ei läpi -> välitön reagointi

Käsitteistä

- Tuotteenhallinta, keskittyy tuotteen tarvitsemien osakokonaisuuksien hallintaan (Configuration Management)
  - Osa-alueena versionhallinta

- Software Product Management (SPM), ohjelmistotuotteen hallinta
  - Liiketoimintataso
  - Tuotepäällikkö, Product Manager
- Software/Application Lifecycle Management, ohjelmiston elinkaarenhallinta
  - “the entire lifecycle from the idea conception, through to the development, testing, deployment, support and ultimately retirement of systems”

Lopuksi

- Luennoilla
  - Ohjelmistosuunnittelu
    - Arkkitehtuurisuunnittelusta yksityiskohtaiseen suunnitteluun
    - Arkkitehtuurityylit, suunnittelumallit, sovelluskehykset
    - Uudelleenkäyttö
  - Tuotteenhallinta
    - Komponentit
    - Konfiguraatiot
    - Toimintatavat

LUT
University