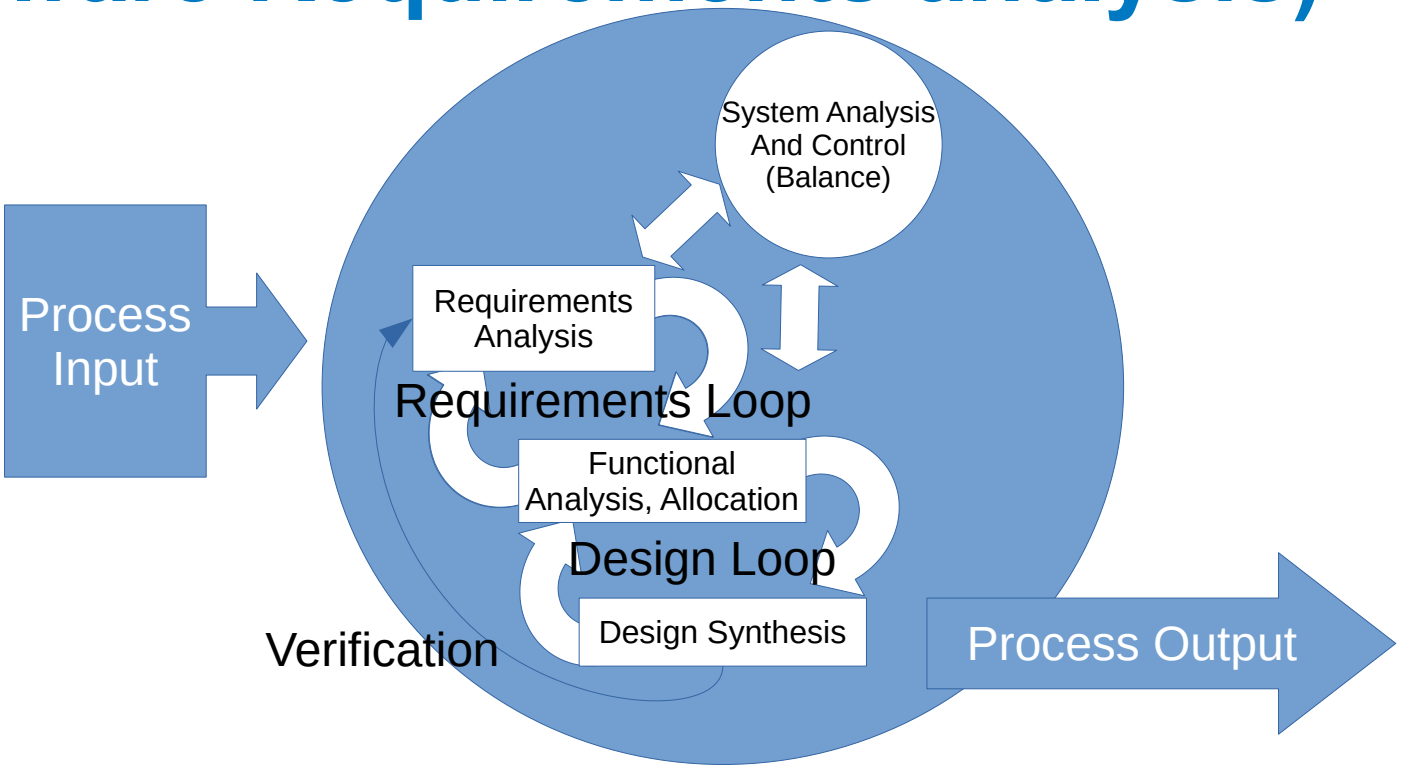
Sovellutuksen elinkaari

Software lifecycle

13.04.2021

Sauli Isonikkilä

Vaatimusmäärittely (Software Requirements analysis)



Vaatimusmäärittely...

- Stakeholder identification (intressiryhmien tunnistaminen)

  Customer (asiakas), Employee (työntekijä), Investors (sijoittajat, kannattavuus), Suppliers and vendors (alihankkijat, toimittajat), Communities (kunnat, työturvallisuus, muu turvallisuus), Government (hallinto, verot)

- Functional requirements (toiminnalliset vaatimukset

  Määrittelee sovellutukseen tulevan funktion tai komponentin käyttäytymisen tai toiminnan.

  Esim: FR1: Käyttäjän pitää pystyä lisäämään uusi tilaus data

- Non-functional requirements (ei-toiminnaliset vaatimukset)

  Tässä määritellään kriteerejä, joiden avulla voi arvioida järjestelmän toimintaa. Tarkoituksena tässä ei ole siis määritellä toimitaa tai sen käyttäytymistä (behaviour). Ei-funktionaalisia vaatimuksia ovat tavallisesti: Availability (esim 24/7), Efficiency (tehokkuus), Flexibility (joustavuus), Portability (asennettavissa ei käyttöjärjestelmille esim), Integrity (eheys), Performance (”toiminnan nopeus”, esim. MIPS eli million instructions per second tms.), Reliability (luotettavuus) , Reusability (uudelleen käytettävyys), Robustness (kestävyys), Scalability (”laajennettavuus”), Usability (käytettävyys).

  Esim NFR1: Järjestelmän pitää olla skaalautuva (laajennettavissa laajemman käyttäjämäärän tarpeisiin).

Vaatimusmäärittely...

- Use cases (käyttötapaukset)

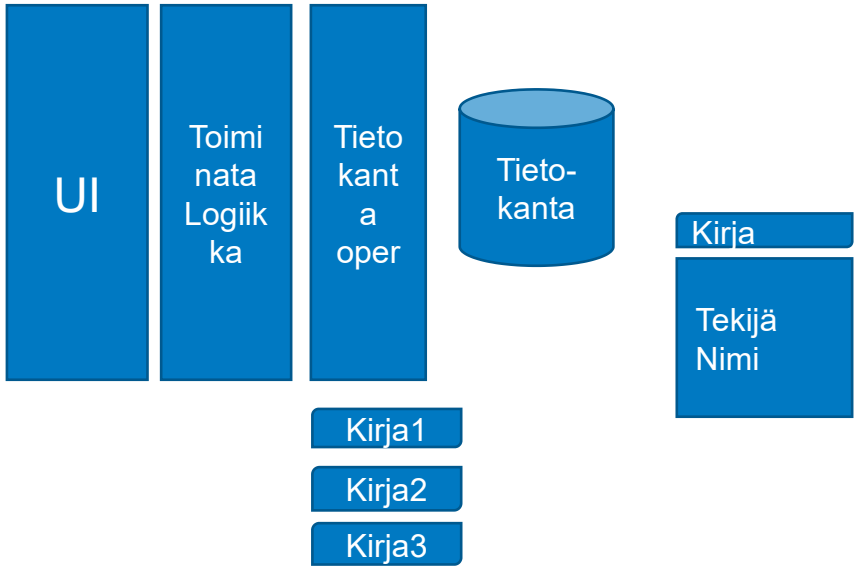
  Käyttäjän kannalta kirjoitettu tarina siitä miten käyttötapaus etenee. Esim. käyttäjä valitsee tuotteen, lisää otsoskoriin, ostoskorissa editoi lukuäärän, valitsee maksutavan, toimitustavan, ja hyväksyy tilauksensa.

- Requirements specification (vaatimusmäärittely specifikaatio, dokumentaatio)

  Functional requirements (functional partitioning, control description), non-functional requirements, use cases, environment characteristics (hardware, peripherials, users)

Ohjelmiston suunnittelu (Software design)

- Software architecture (sovellutusarkkitehtuuri)
- Control hierarchy (kontrollihierarkia)
- Data structure (tietorakenne)
- Information hiding (tiedonkätkentä)



Ohjelmiston toteutus (Software implementation)

- Software coding (ohjelmiston ohjelmointi)

Ohjelmiston testaus (Software testing)

- Static testing (staattinen testaus), dynamic testing (dynaaminen testaus) and passive testing (passiivinen testaus)
- Black box testing, white box testing
- Testing levels (testaus tasot):
  Unit testing (yksikkötestaus), Integration testing, System testing (järjestelmätestaus), Acceptence testing (hyväksymistestaus)
- Test types (testaus tyypit):
  Regression testing, Acceptance testing, Alpha testing, Beta testing, Functional testing non-functional testing, Usability testing (käytettävyystestaus)

- Regressiotestaus:

  Regressiotestauksessa testataan, että ohjelmistoon tehty muutos ei vaikuta ohjelmiston toiminnallisuuteen (functionality).

- Yksikkötestaus:

  Yksikkötestauksessa (myös tunnetaan nimellä komponenttitestau) testataan, että yksikkö (tai yksiköt) tai komponentit ohjelmistossa toimivat niin kuin ne on suunniteltu toimivan.

- Staattinen testaus:

  Staattinen testaus tehdään ilman, että siinä ajetaan ohjelmistoa. Staattisessa testauksessa tarkoituksena on löytää virheitä, epäselvyyksiä dokumentaatiossa (vaatimusmäärittely dokumentaatiossa, ohjelmiston suunnitteludokumentaatiossa, käyttötapauksissa jne).

- Dynaaminen testaus:

  Dynaamisessa testauksessa on tarkoitus testata löytyykö ohjelmistoa ajettaessa heikkoja alueita sen käyttäytymisessä (behaviour) eri muuttujien arvioilla tai jos muuttujat pidetään vakoinina.

- Integraatio testaus:

  Integraatio testauksessa erikseen jo testattuja komponentteja testataan yhdessä muiden sovellutukseen tulevien komponenttien kanssa. Integraatio testaus tehdään yksikkötestaukseen jälkeen ja ennen validation testausta. Integraatio testaus tehdään testaussuunnitelman (test plan) mukaisesti.

- Alfa testaus:

  Alfa testaus on se muoto validation testauksesta, jossa pyritään löytämään kaikki mahdolliset ongelmat tai virheet (bugs) ennen kuin ohjelmisto julkaistaan julkaisukelpoisena tuotteena.

- Beta testaus:

  Beta testaus tehdään alfa testauksen jälkeen. Beta testauksessa julkaisukelpoiseksi todettu ohjelmistotuote annetaan rajoitetulle ulkopuoliselle yleisölle (tunnetaan nimellä beta testaajat) testattavaksi. Tässä testauksessa tarkoituksena on varmistaa, että ohjelmisto tuotteessa on mahdollisimman vähän vikoja ja virheitä. Ohjelmisto tuotteen versioita voidaan myös julkaista laajemmalle yleisölle testattavaksi, jotta saadaan palautetta, lisää mahdollisia käyttäjiä ja toimittaa heille uusia ominaisuuksia aikaisemmin.

- Black-box testaus:

  Mustalaatikko testauksessa nimensä mukaisesti suoritetaan ohjelmiston testaus siitä, miten se toimii (functionality), ilman että perehdytään ohjelmiston sisäiseen toimintaan ja rakenteeseen.

- White-box testaus (tunnetaan myös nimillä clear box, glass box testaus):

  Tässä testauksessa testaaja tuntee hyvin testattavan ohjelmiston sisäisen rakenteen ja miten se toimii sisäisesti. Testaajan hyviä tietoja sitten käytetään testi tapausten suunnitteluun ja sitten ohjelmiston testaukseen niiden mukaisesti.

- Hyväksymistestaus (Acceptance testing):

  Tavallisesti hyäksymistestaukseen kuuluvat käyttäjän hyväksymistestaus, toiminnallinen hyväksymistestaus (operational acceptance testing), sopimuksellinen ja lakien sekä muiden velvoitteiden täyttämiseen liittyvä hyväksymistestaus ja myös alfa testus sekä beta testaus.

Ohjelmiston julkaiseminen (Software delivery/deployment)

- Release

  Julkaiseminen, julkaistaan sovellutuksesta julkaisukelpoinen versio, jonka asiakkaat voivat hankkia. Ohjelmisto tuotteesta voi olla omat versiot eri alustoille, kuten Microsoft Windows, Linux, Apple iOS, Android.

- Installation and activation/deactivation

  Ohjelmiston asennus ja sen aktivointi ja aktivoinnin poistaminen.

- Update

  Ohjelmiston päivitys

- Version tracking

  Ohjelmiston eri versioiden seuranta. Eri asiakkailla voi nimittäin olla käytössään eri versioita julkaistusta ohjelmisto tuotteesta. Heidän ilmoittamiaan puutteita ja virheitä on pystyttävä myös päivittämään. Tässä versiointi on tärkeää. Versiointiin on olemassa erilaisia version hallintajärjestelmiä, kuten Git (GitHub), Microsoftin Team Foundation Server, AWS CodeCmmit jne.