TESTAAMINEN JA TESTAUSYMPÄRISTÖT

SYSTEEMITYÖ 2019

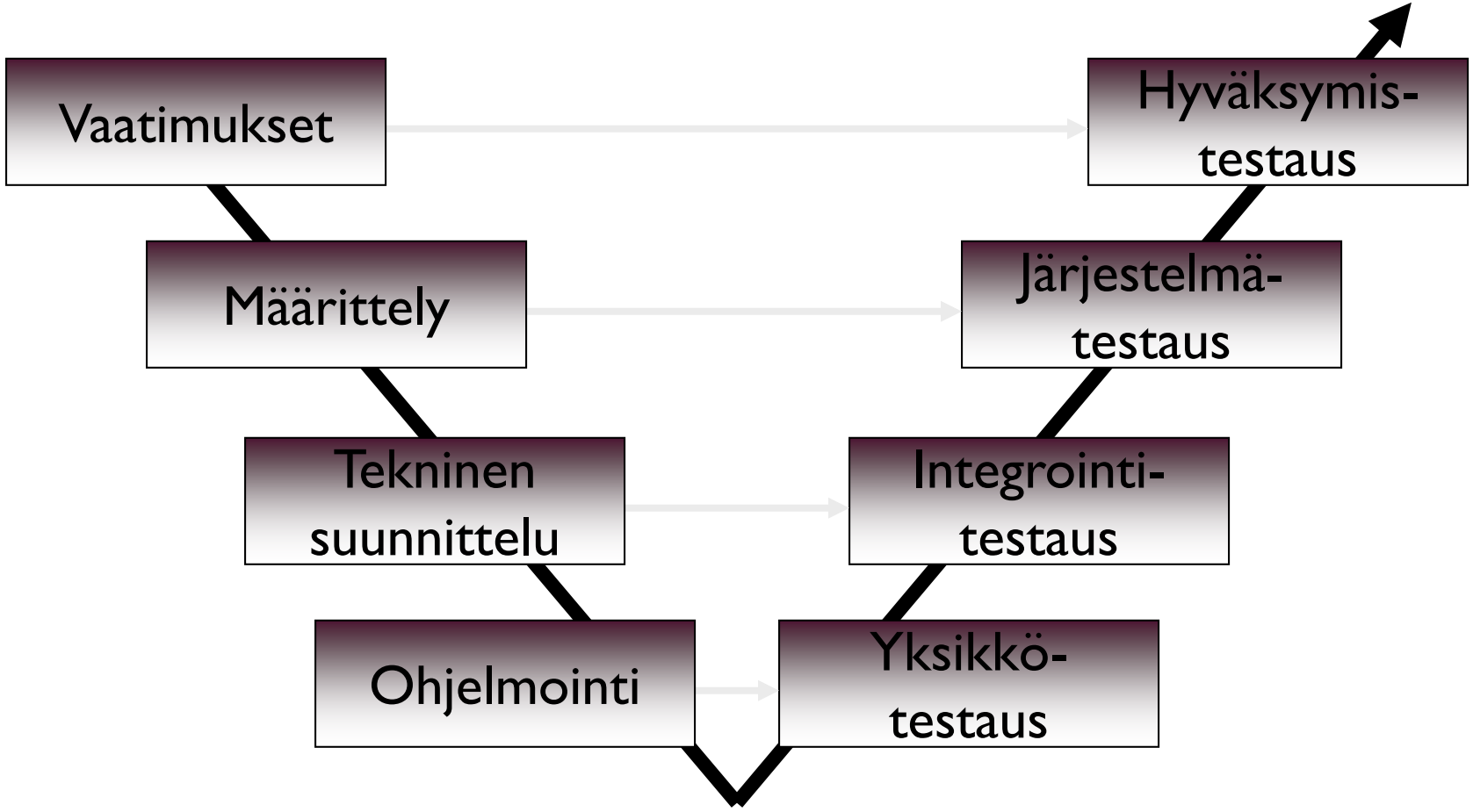
KURSSIN TAVOITTEET

- Tiedät mitä testauksen rooli ohjelmistoprojektissa
- Osaat suunnitella, määritellä ja toteuttaa joitakin testausprosessin osia
- Tiedät mikä on testaustaso ja testaustyyppi ja osaat tehdä ohjelmistotestaukseen liittyviä dokumentteja testaustyön tueksi

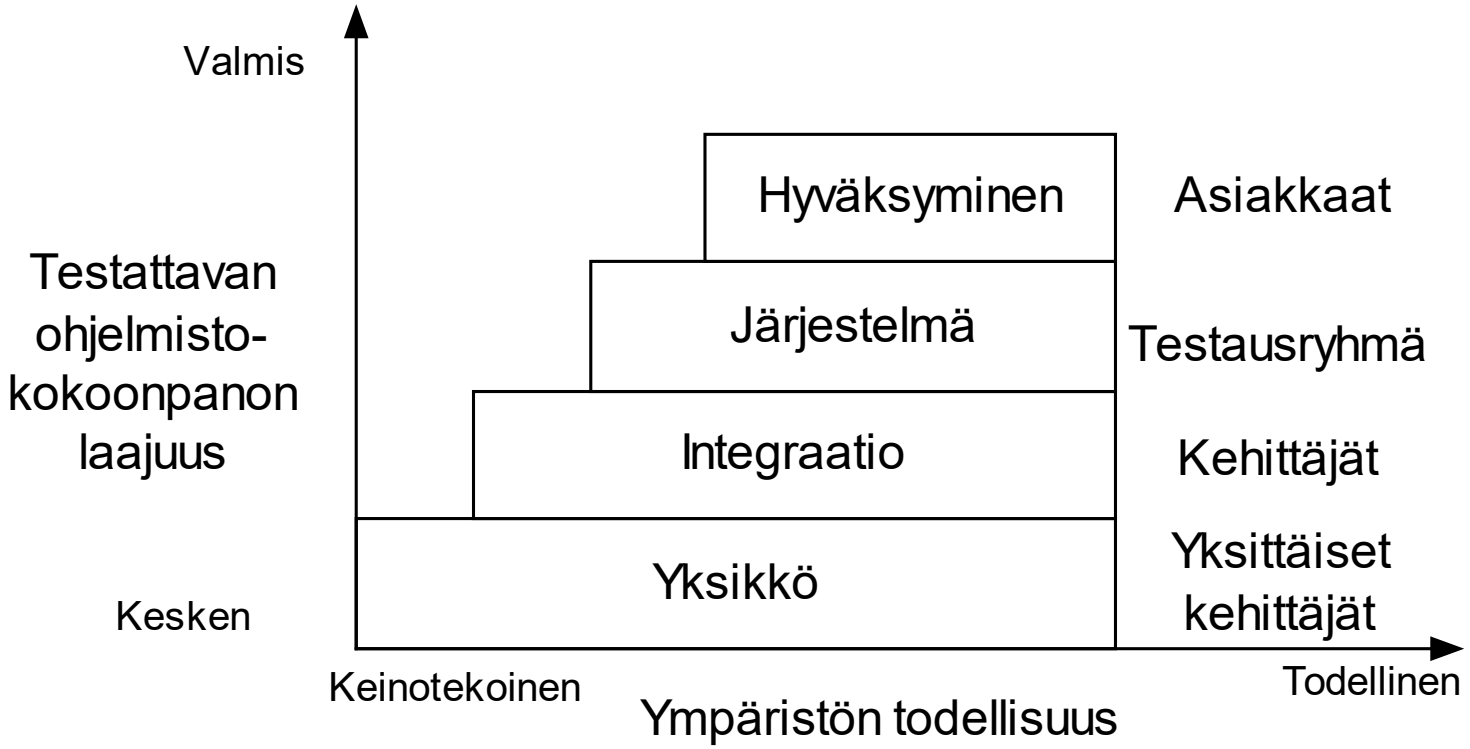
Sisältö OPS:

- Testauksen V-malli
  - moduulitestaus (ohjelmoinnin yksikkötestaus)
  - integrointitestaus
  - järjestelmätestaus
  - hyväksymistestaus
- testaussuunnitelman, testitapausten ja testausraporttien laatiminen

V-MALLI – TESTAUKSEN TASOT

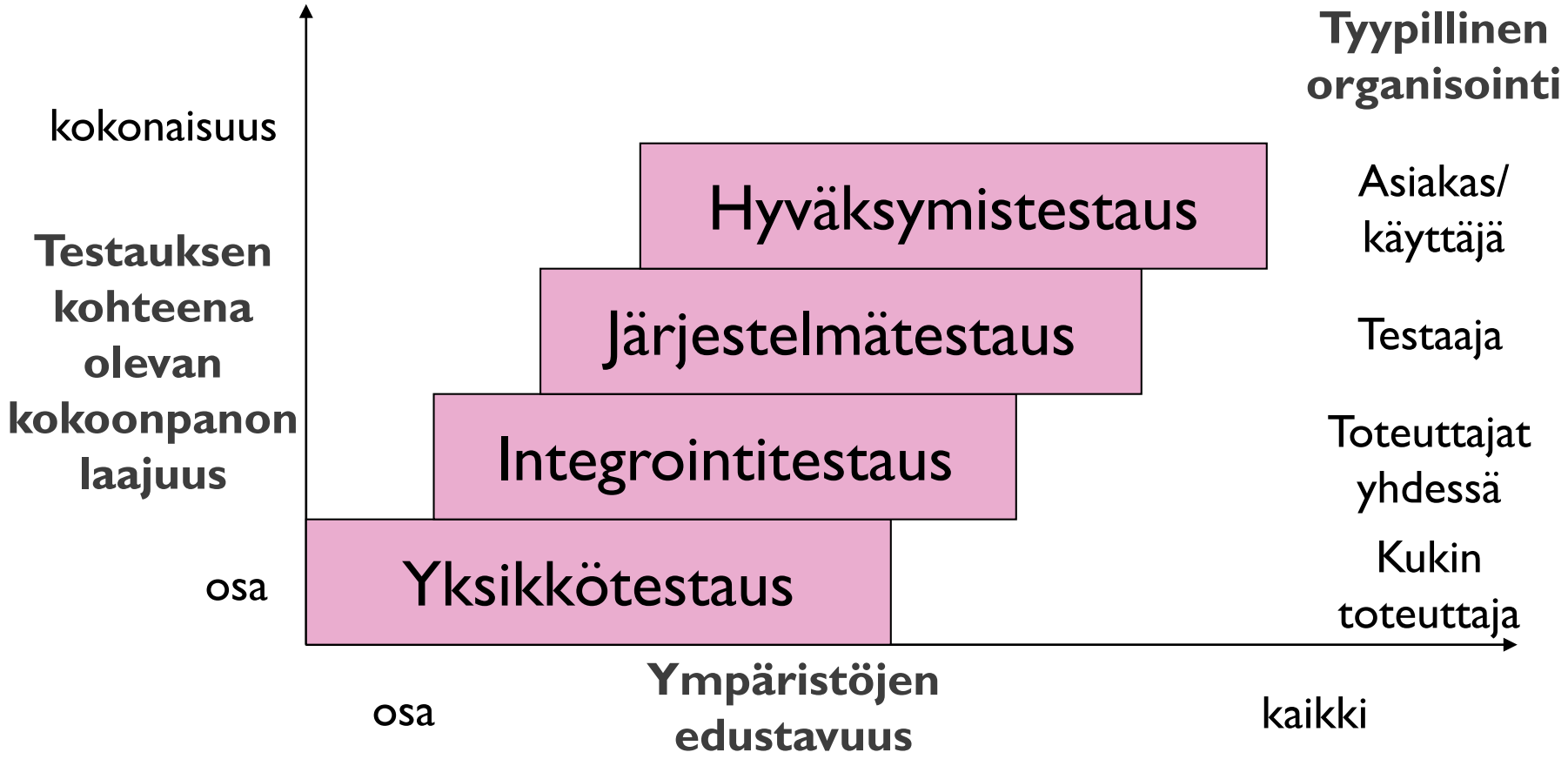


TYYPILLISET TESTAUKSEN TASOT



[Craig&Jaskiel 2002]

TESTAUSTASOJEN TYYPILLISET MERKITYKSET



YKSIKKÖTESTAUS

- Yksikkötestauksella tarkoitetaan koodin toteutukseen kohdistuvaa teknistä testausta
  - Keskittyy virheisiin, jotka tyypillisesti syntyvät koodia kirjoittaessa
- Yksikkö voi olla:
  - **Tehtävänanto**: ”Koodaan ja testaan tämän ominaisuuden”
  - **Komponentti, moduuli, luokka**: ”toteutan kaikki toiminnallisuuden näihin kooditiedostoihin”
- Laajuuden tulkinnassa paljon vaihtelua
  - Usein näkee suurissa hankkeissa jaettavan yksikkötestaustaso kahteen osaan, yksikkö- ja komponenttitestaukseen
  - Tällöin yksikkötestaus kohdistuu kooditiedostolaajuuteen ja komponenttitestaus vastaa kehittäjän omalle komponentilleen tekemää osajärjestelmätestausta ennen integrointia kokonaisjärjestelmään

YKSIKKÖTESTAAMINEN: JAVASCRIPT

- JavaScript:in testaamiseen tämän hetken suosituin työkalu on Jest
- Yksikkötestaus (funktiot, luokat)
  - Annetaan tietty input ja tarkistetaan että output on se mitä odotetaankin
- Jestin avulla voi:
  - Automatisoida yksikkötestejä (test template, assertions)
  - Tutkia testauksen laajuutta (code coverage)
  - Rakentaa testausjärjestelmän jossa ulkoiset rajapinnat on korvattu stub:eillä tai mock:eilla

YKSIKKÖTESTAUKSEN HARJOITTELUA

- Tee uusi projektikansio, siirry siihen CMD:llä ja aja:

```
npm install --save-dev jest
```

- Käynnistä Visual Studio Code, tee uusi summa.js tiedosto:

```
function sum(a, b) {
        return a + b;
}
module.exports = sum;
```

- Tee uusi summa.test.js tiedosto:

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () =>
{ expect(sum(1, 2)).toBe(3); }

);
```

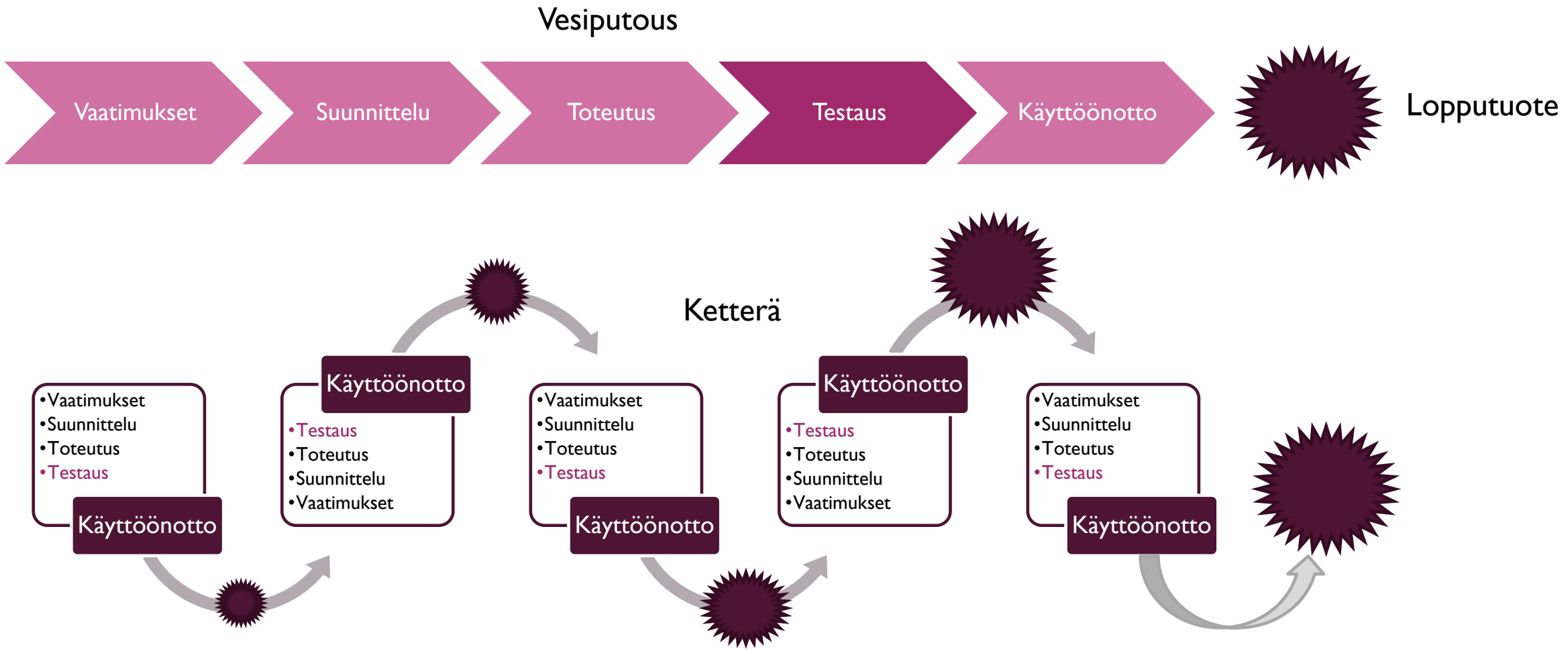
- Lisää seuraava koodi package.json-tiedostoon:

```
{ "scripts": { "test": "jest" } }
```

- Käynnistä testit CMD:n kautta:

```
npm run test
```

OHJELMISTOPROJEKTIMENETELMIÄ



CONTINUOUS INTEGRATION / CONTINUOUS DELIVERY (CI/CD)

- Ketterässä ohjelmistotuotannossa käytetään ns. jatkuvaa integrointia (CI)
  - Ohjelmoijat committoivat koodinsa versionhallintaan heti kun uusi vaatimus on toteutettu ja tekevät integrointipyynnön (merge request), jolloin CI-järjestelmä ajaa koodille automaattisesti yksikkötestit (committoidaan koodin mukana)
  - Jos yksikkötestit menevät läpi ja koodi myös hyväksytään koodikatselmoinnissa, se voidaan integroidaan päähaaraan (master branch), yleensä useamman ohjelmoijan on hyväksyttävä muutokset
  - Nyt koko ohjelma voidaan koostaa (master build), ja ajetaan sille järjestelmätason testit ja suoritetaan järjestelmätason manuaalista testausta (QA, laadunvarmistusta) ennen kuin se julkaistaan

INTEGROINTITESTAUS

- Integrointitestauksella tarkoitetaan kaiken tasoisten osajärjestelmien yhdistämiseen kohdistuvaa teknistä testausta
- Integrointitestaus usein vaikea erottaa yksikkötestauksesta, koska integrointi voi sisältää ohjelmointitehtäviä
  - Usein toimiva perussääntö: enemmän kuin yksi toteuttaja → integrointitestaustarve
  - Arkkitehtuuri apuna integrointitestaustarpeiden tunnistamisessa
- Testaus kohdistuu integroitavaksi valittujen osajärjestelmien laajuuteen
  - Integrointitestauksessa osajärjestelmien liittymän näkökulma

JÄRJESTELMÄTESTAUS

- Järjestelmätestauksella tarkoitetaan kokonaisjärjestelmän laajuuteen kohdistuvaa testausta
  - Voi sisältää sekä teknisen että käyttäjän näkökulman
- Testaus kohdistuu kulloinkin olemassa olevaan kokonaisjärjestelmän laajuuteen
  - Laajuus kasvaa uusien integrointien myötä
  - Järjestelmätestauksessa kokonaisjärjestelmän näkökulma

HYVÄKSYMISTESTAUS

- Hyväksymistestauksella tarkoitetaan testausnäkökulmaa, jossa tarkastellaan järjestelmää sen todelliseen ympäristöönsä soveltuvuuden kannalta
- Sopimuspohjainen hyväksymistestaus
  - Voi olla juuri ennen tuotantoon siirtoa tai keskellä järjestelmätestausta
  - Etenkin monitoimittajaympäristössä hyväksymistestaus liittyy laskunmaksuun eikä voida odottaa tuotantoon siirtoa
  - Voi toistua jokaisen uuden alijärjestelmän versioasennuksen jälkeen
- Alpha- ja betatestaus, pilotointitestaus
  - Virhetilanteiden löytäminen, joita on vaikeaa tai kallista löytää laboratorio-olosuhteissa – todellisten ympäristöjen monimuotoisuuden toistaminen haaste

ERITYISESTI HUOMIOI

- Suunnittele testit perustuen jokaisen suunnitteluvaiheen tuotokseen, mutta suorita testit siinä järjestyksessä joka on kaikkein järkevin
  - Testauksen vaiheistus voi olla erilainen kuin tasot peräkkäisinä vaiheina
  - Pyri saamaan oikea näkökulma, jolla löydetään tärkeät virheet, mukaan suoritukseen mahdollisimman aikaisin
- Uusintatestauksella entistä suurempi painoarvo kun lisätään uusia ominaisuuksia olemassa oleviin järjestelmiin (vrt. regressiotestaus)
  - Selvitä minkä testaustason näkökulmasta uusintatestausriski on taloudellisinta hallita

TESTAUSKÄSITTEIDEN KOHTEITA

Testauskäsite	Vastaa kysymyksiin
Testauksen missio	Mitä? Miksi?
Testausstrategia	Mitä? Miksi? Miten? Entä jos?
Testaustekniikat	Miten?
Testauskäytännöt	Miten?
Testausorganisaatio	Kuka?
Testausympäristö ja – välineet	Millä?
Testausprosessi	Mitä? Kuka? Missä? Milloin? Miten?

TESTAUSTYYPPI

- **Testaustyyppi** – Ryhmä testausaktiviteettejä, joilla on yhteisiä ominaisuuksia joiden perusteella ne voidaan tunnistaa omana luokkanaan, ja jotka on ryhmitelty arvioimaan yhtä tai useampaa toisiinsa liittyvää laatuominaisuutta.
  - Voi sijoittua yhdelle tai useammalle testaustasolle ja testausvaiheeseen.
  - Kaikki eivät aina tarpeellisia - käytännössä testaustyypit muodostavat tarkastuslistan mahdollisesti katettavista asioista
- Testaustyypit jakaantuvat toiminnallisiin ja ei-toiminnallisiin

TOIMINNALLISEN TESTAUKSEN TESTAUSTYYPPEJÄ

- Toiminnallisuustestaus (functionality testing, feature testing)
- Yhtäaikaisuustestaus (concurrency testing)
- Asennustestaus (installation testing)
- Alustatestaus (platform testing)
- Aloitustestaus (build verification testing, smoke testing)
- Konfiguraatiotestaus (configuration testing)
- Yhteensopivuustestaus (compatibility testing)
- Rinnakkaistestaus (end-to-end testing)
- Rajapintatestaus (interface testing)
- Poikkeustilannetestaus (recovery testing)
- Lokalisointitestaus (localization testing)
- Dokumentaation testaus (documentation testing)
- Aineiston laadun testaus (data quality testing)
- Alfatestaus (alpha testing)
- Betatestaus (beta testing)
- Muuntotestaus (conversion testing)
- Tuotantotestaus (production testing, operational testing)
- Standardien testaus (standards testing)

EI-TOIMINNALLISEN TESTAUKSEN TESTAUSTYYPPEJÄ

- Luotettavuustestaus (reliability testing)
- Suorituskykytestaus (performance testing)
- Kuormitustestaus (load testing)
- Rasitustestaus (stress testing)
- Paljoustestaus (volume testing)
- Kestävyystestaus (endurance testing)
- Tietoturvatestaus (security testing)
- Käyttöturvallisuuden testaus (safety testing)
- Käytettävyystestaus (usability testing)
- Esteettömyystestaus (accessibility testing)
- Palautettavuustestaus (recoverability testing)
- Tuettavuustestaus (supportability testing)
- Ylläpidettävyystestaus (maintainability testing)
- Siirrettävyystestaus (portability testing)
- Koodin laadun testaus (code quality testing)