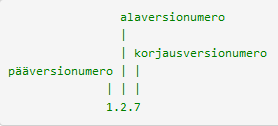
## Ohjelmiston versiohallinta

**Mitä tarkoittaa ohjelmiston versiohallinta? Mihin osa-alueisiin se voidaan jakaa ja mitä eri osa-alueet tarkoittavat?**

Ohjelmistoprojektin tuotosten hallinta ja se mahdollistaa ohjelmiston kehityksen seurannan ja hallitun kehityksen. Semanttisessa versioinnissa ohjelmiston versio muodostuu kolmesta komponentista: pää-, ala- ja korjausversionumerosta.



**Mitä tarkoittaa versiointi? Millaisia perusteita versioinnilla voi olla? Mainitse kolme.**

Versioinnin tarkoituksena lähtökohtaisesti on, että ohjelmistoversiomuutoksista tulisi ymmärrettäviä ja johdonmukaisia.

Perusteet/Ulottuvuudet:

* Historiallinen
  + uudempi versio syrjäyttää aiemman
* Looginen
  + luodaan haaroja, jossa kukin haara kuvastaa kehityksen vaihtoehtoista kulkua
* Yhteistoiminnallinen
  + väliaikaisuuden hallinta, tarkoituksena on luoda hetkellisiä versioita

**Mitä tarkoittaa versioiden merkitseminen?**

Ohjelmiston eri versiot voidaan järjestää ja merkitä numeroin.

**Miksi versioiden välisten erojen tunnistaminen on tärkeää?**

Se on tärkeää, koska toisiaan seuraavat versiot saattavat olla hyvin toistensa kaltaisia ja eroavaisuuksien tarkan määrittelyn avulla voidaan keskittyä muutokseen ja säästää resursseja.

**Mikä on eri versioiden tallentamisen tarkoitus?**

Ohjelmiston kehityksen ja ohjelmiston aikaisempien versioiden uudelleenrakentamisen mahdollistamiseksi on vanhat ja uudet versiot voitava tallentaa pysyvästi. Yleistä on käyttää varastoa, jonne ohjelmiston edeltävät versiot ja niitä koskevat konfiguraation hallinnan tiedot tallennetaan.

**Mikä on versionhallintajärjestelmä? Kuvaile ainakin kaksi avoimen lähdekoodin ja kaksi kaupallista versionhallintajärjestelmää.**

Avoimet:

* Fossil
  + hajautettu versioiden valvonta- ja vianseurantajärjestelmä
* Veracity
  + integroitu vianseurantajärjestelmä ja ketteränkehityksen hallintatyökalu

Kaupalliset

* StarTeam
  + versionhallintajärjestelmä, jota käytetään ohjelmistokehityksessä, ja varsinkin silloin kun hankkeeseen liittyy useita työryhmiä eri paikoissa
* PTC Integrity
  + ohjelmiston ja sovelluksen elinkaaren hallintatyökalu

**Ohjelmistosta löytyvät versiot 1.0, 2.0, 2.1, 1.1.1. Mitkä seuraavista ovat mahdollisia seuraavia versioita? Perustele vastauksesi.**

1. **2.2.1 (ei voi tapahtua, koska 2.1:stä ei voi hypätä kuin 2.2:een tai 3.0:een)**
2. **3.1 (voi tapahtua, jos 2.1:n jälkeen tehdään iso päivitys)**
3. **1.1 (voi tapahtua, jos 1.0:n jälkeen tehdään pieniä korjauksia)**
4. **1.1.2 (voi tapahtua, jos 1.1.1:n jälkeen tehdään korjauksia)**
5. **2.2 (voi tapahtua 2.1:n jälkeen)**
6. **2.2.2 (ei voi tapahtua, koska 2.1:stä ei voi hypätä 2.2.2;een)**
7. **1.0.1 (voi tapahtua, jos 1.0:n jälkeen tehdään korjauksia)**
8. **1.1.1 (ei voi tapahtua, koska versio on jo olemassa)**
9. **1.2 (voi tapahtua 1.0:n jälkeen)**
10. **1.1.3 (voi tapahtua, jos 1.1.1:n jälkeen tehdään korjauksia)**
11. **1.2.1 (ei voi tapahtua, koska 1.0:sta ei voi hypätä 1.2.1:seen)**

**Ohjelmistosta löytyy versiot 1.0, 2.0, 1.1, 2.1, 3.0, 2.2., 2.1.1, 2.1.2, 3.1., 1.2, 1.3, 1.2.1. Teet korjauksen, joka on tarkoitus periyttää kaikkiin ohjelmiston versioihin. Mitä versioita sinun on ohjelmistosta tehtävä sisältäen uuden korjauksen?**

* Ei ole käytännöllistä, mutta jos oikein on pakko, niin jokaisen version loppuun lisätään \*.1 tai \*.a

**Ohjelmistosta löytyy versiot 1.0, 2.0, 1.1, 2.1, 3.0, 2.2., 2.1.1, 2.1.2, 3.1., 1.2, 1.3, 1.2.1. Päivität ohjelmistoon uuden ominaisuuden uusimpaan versioon. Mikä tulee versioksi?**

* 3.2 tai 3.1.1 (riippuen ominaisuuden suuruudesta), koska viimeisin versio on ollut 3.1

**Ohjelmistosta löytyy versiot 1.0, 2.0, 1.1, 2.1, 3.0, 2.2., 2.1.1, 2.1.2, 3.1., 1.2, 1.3, 1.2.1. Teet ohjelmistosta uuden julkaisun. Mikä tulee sen versioksi?**

* 3.2 tai 4.0 (riippuen muutoksien koosta), sillä viimeisin versio on ollut 3.1