LIITE 4: Järjestelmäarkkitehtuurikuvaus ja kehittämismalli

Tämä dokumentti on tiivistetty yleis- ja tavoitetilan kuvaus Peppi-järjestelmäarkkitehtuurista ja sen kehitysmallista. Maksullisen koulutuksen projektissa tulee noudattaa tätä kuvausta.

Johdanto

Metropolia Ammattikorkeakoulu tavoittelee järjestelmäarkkitehtuurissaan palvelupohjaisen arkkitehtuurin (SOA) periaatteita ja käytänteitä.

Kaikki käytetyt teknologiat ovat suosittuja ja koeteltuja avoimen lähdekoodin tuotteita. Pääperiaatteena on ollut tehdä arkkitehtuurista niin suoraviivainen, että palvelun kehittäjä voi helposti ymmärtää palvelun toimintaperiatteet sekä palveluiden ja asiakasohjelmien välisen viestinkulkuun liittyvät ratkaisumallit.

Palvelupohjaisuus

Palvelupohjaisuus toteutuu seuraavien periaatteiden mukaan:

Sääntö 1, SOA-metodologia.

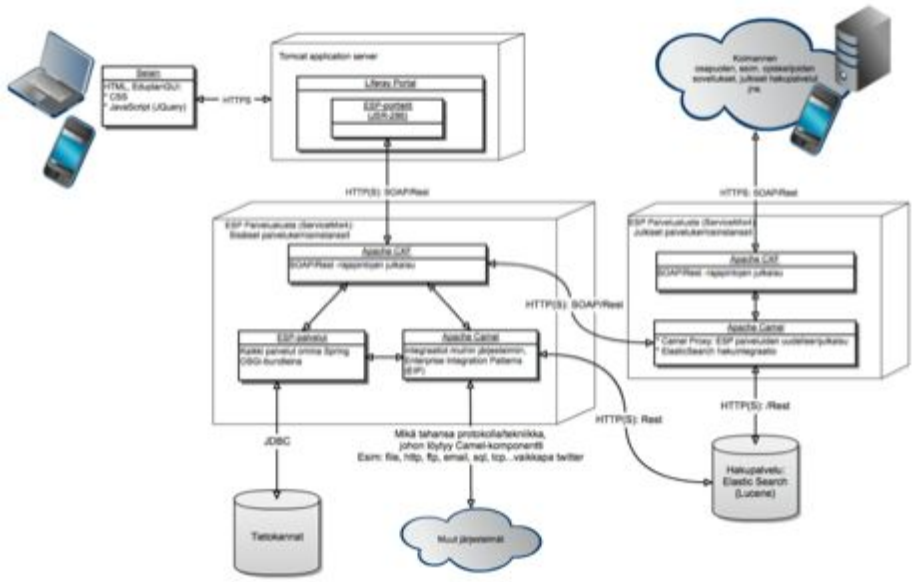
- Tietoa käsitellään palveluiden/palvelurajapintojen kautta.
- Palveluiden tulee olla autonomisia, toinen palvelu ei kontrolloi niiden toimintaa. Niitä voidaan ajaa hajautetusti. Ne eivät ole sidottuja toisen palvelun sisäiseen toimintaan.
- Palveluiden tulee olla löyhästi sidottuja toisiinsa, palvelut ovat sidoksissa toisiinsa vain rajapintojen kautta. Tällöin palvelun sisäinen toteutus on vaihdettavissa.
- Palveluita voidaan uudelleenkäyttää.

Sääntö 2, Standardeihin pohjautuvat rajapintaratkaisut

- Rajapinnat julkaistaan SOAP-pohjaisina webservice-rajapintoina tai Rest-tyyppisinä rajapintoina. Olennaista on, ettei julkaistu rajapinta luo riippuvuutta mihinkään tiettyyn alustaan.

Sääntö 3, Palvelurajapintojen erottaminen käyttöliittymistä

- Käyttöliittymiä ei sidota tiukasti palvelun sisäiseen toteutukseen, jolloin palveluita voidaan uudelleenkäyttää ja käyttöliittymiä voidaan uudistaa moduuli kerrallaan



Yleiskuva järjestelmäkokonaisuudesta

Käyttöliittymät on toteutettu JSR-286 -standardin mukaisina portlet-sovelluksina, joita ajetaan Liferay-portaalissa (ks. alempana kohta). Liferay-portaali tarjoaa monipuolisen ja helppokäyttöisen ympäristön, jonka avulla erilaisia käyttäjä/roolipohjaisia työpöytiä voidaan koostaa.

Peppi-ekosysteemissä on tällä hetkellä seuraavat työpöydät:

- pääkäyttäjän työpöytä
- korkeakoulupalveluiden työpöytä
- suunnittelijan työpöytä
- opettajan työpöytä
- opiskelijan työpöytä

Maksullisen koulutuksen projektissa tuotettavat palvelut sijoittuvat useille eri työpöydille. Maksulliseen koulutukseen hakeutuvien opiskelijoiden käyttöliittymä sijaitsee Metropolian julkisilla Internet-sivuilla, jotka on toteutettu Typo2-julkaisujärjestelmän päälle.

Käyttöliittymät kutsuvat palvelualustassa (ESP)/(ServiceMix4) ajettavia palveluita SOAP- tai Rest -rajapintojen avulla, jotka on julkaistu Apache CXF:n avulla. Palvelut on toteuttu Java-moduuleina (OSGi). On tärkeää huomioida, että palvelut voivat kutsua toisiaan myös OSGi-säiliön sisällä (ks. alempana kohta Palvelumoduuli).

Integraatiot muihin järjestelmiin on toteutettu Apache Camelin avulla (ks. alempana kohta *Integraatiot*)

Kuvassa on myös erikseen kuvattu täysin julkisten rajapintojen tarjoaminen erillisen julkisen palvelukerroksen avulla. Julkisen palvelukerroksen avulla rajapintoja voidaan uudelleenjulkaista kuormittamatta sisäistä palvelukerrosta. Julkisen palvelukerroksen rooli:

- Välimuistien lisääminen palveluille
- Palveluiden julkaiseminen eri skeemoissa
- Erilaisten autentikointi/auktorisointiratkaisuiden käyttöönotto kuormittamatta sisäistä palvelukerrosta
- Hakupalveluiden toteuttaminen esim. Elastic Searchin avulla

Palvelukerros

Keskeisin komponentti on palvelualusta (ESP), joka koostuu Apache ServiceMix4 -ESB-tuotteesta ja siihen erikseen tehdyistä laajennoksista (ESP Feature). Palvelualusta toimii ajoympäristönä palvelumoduuleille ja integraatioille.

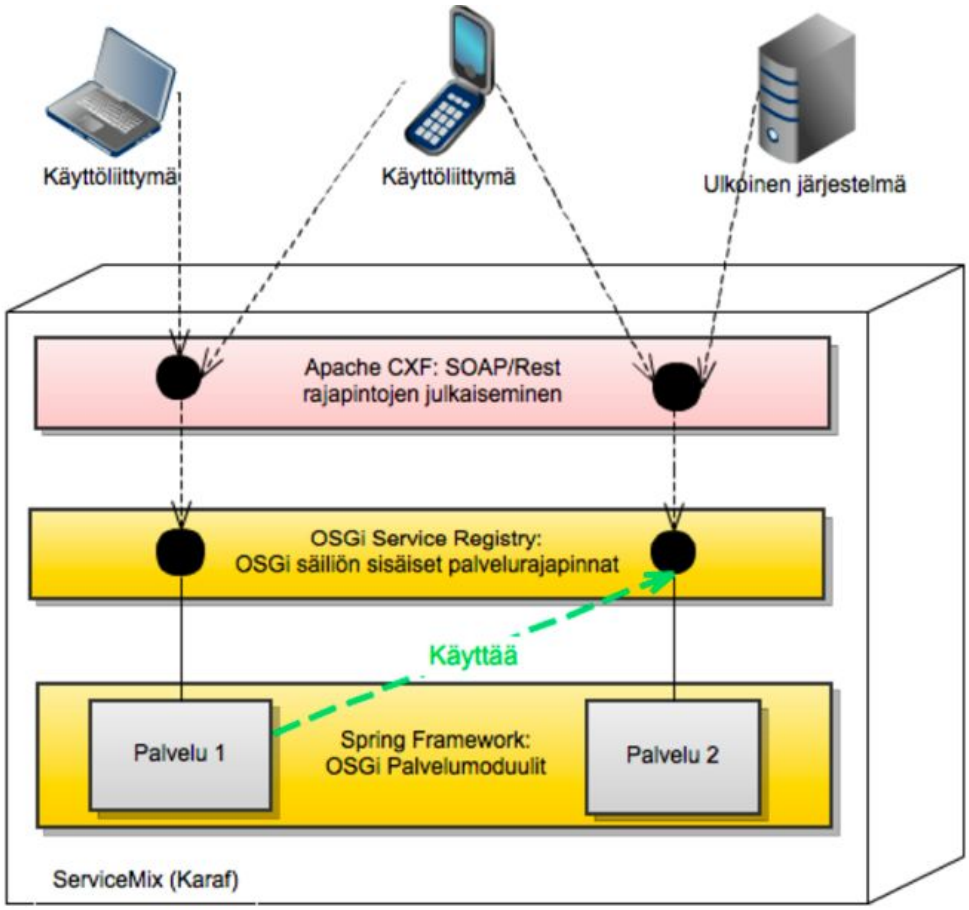
ServiceMix4 on avoimen lähdekoodin Enterprise Service Bus (ESB) tuote, joka perustuu OSGi-teknologiaan. OSGi-säiliönä (container) käytetään ESP:n tapauksessa Apache Felixiä. ServiceMix4:n ydin on Apache Karaf, joka on kevyt OSGi runtime -ympäristö. Karaf laajentaa OSGi containeria tarjoamalla toimintoja mm. OSGi-moduulien hallintaan. ServiceMix4 onkin käytännössä Apache Karaf + mukaan paketoituja OSGi -valmiita teknologioita. Oleellisimmat toiminnot ovat:

- Apache ActiveMQ: viestin välitys (JMS)
- Apache Camel: EIP-integraatiot
- Apache CXF: SOAP/Rest -tyyppiset Web Servicet
- Spring Framework: palveluiden kehittämisessä käytetty sovelluskehys

Palvelumoduulit

Palvelut ovat Javalla ohjelmoituja moduuleita, jotka voivat julkaista rajapintoja myös OSGI-säiliön sisällä. Kutsut palveluiden välillä tehdään OSGI-säiliön sisällä ilman, että viestejä täytyy muuttaa eri muotoon lähetyksen ajaksi. Tällöin palveluiden koostaminen ei kuormita järjestelmää yhtä paljon kuin jos koostaminen tehtäisiin eri SOAP-rajapintoja käyttäen.

Julkaistu SOAP/Rest-rajapinta toimii ainoastaan fasadina käyttöliittymäkerrokselle sekä ulkopuolisille järjestelmille.



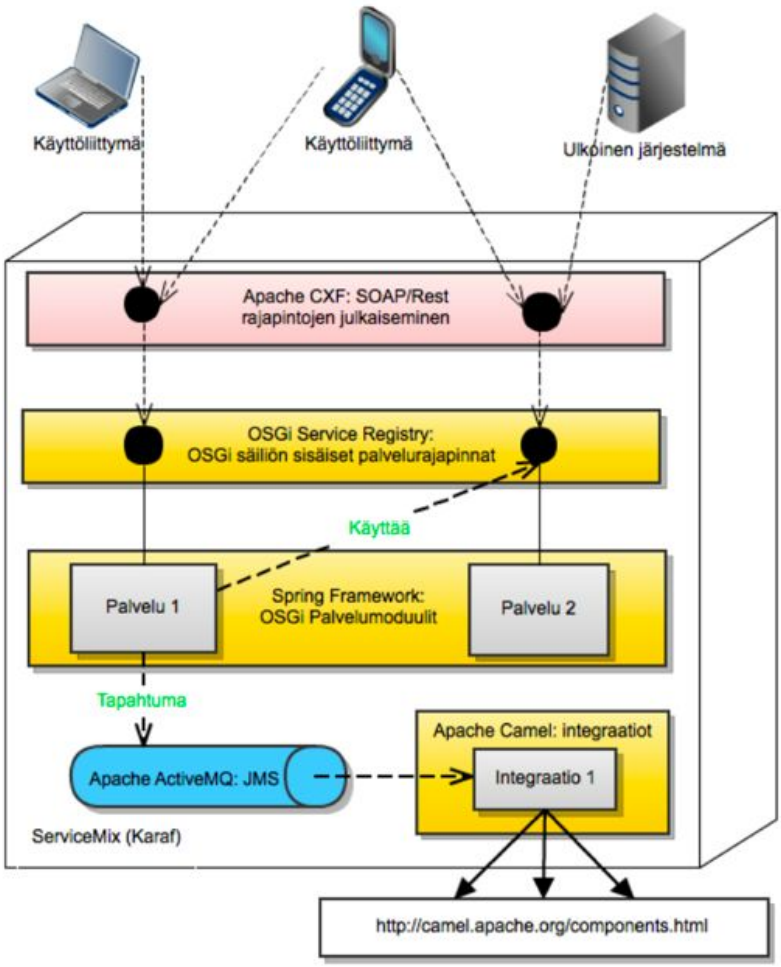
Integraatiot

Palvelut ohjelmoidaan rajapintoja vasten, jotka on erotettu omiin OSGI-bundleihin. On kuitenkin tilanteita joihin sopii paremmin perinteisempi EIP (Enterprise Integration Patterns)-tyyppisten integraatioiden käyttäminen:

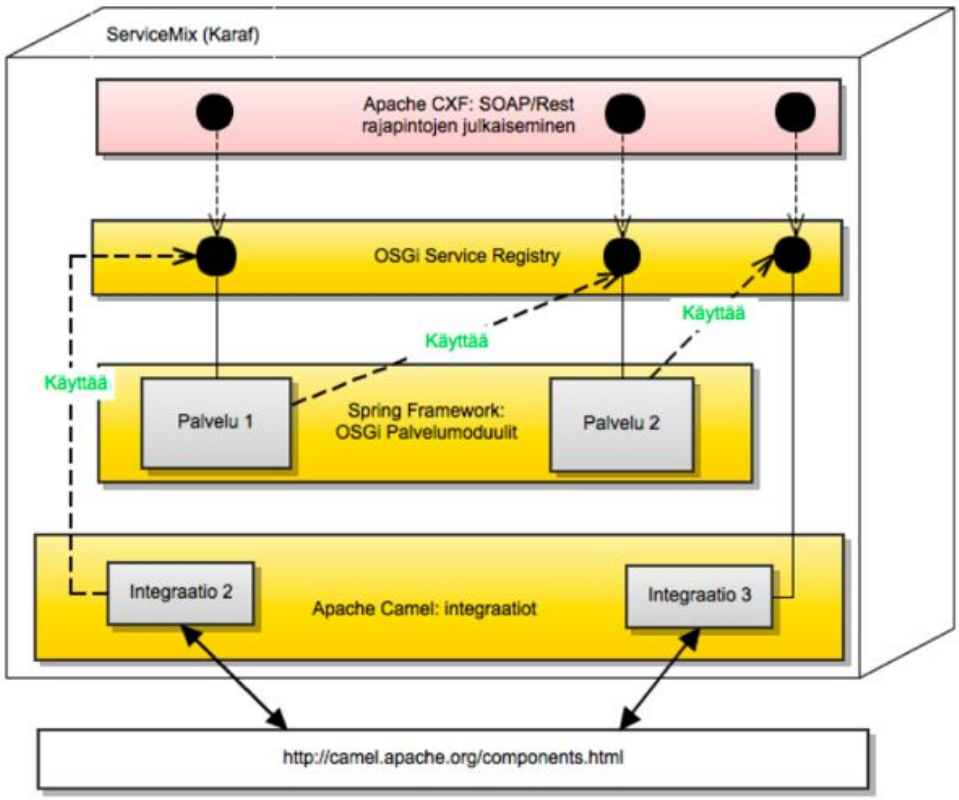
- Palvelut täytyy kytkeä vanhaan ns. legacy-järjestelmään, johon ei ole olemassa ohjelmallista rajapintaa. Käytännössä integraatio täytyy tehdä joko siirtotiedoston tai tietokantayhteyden avulla.
- Palveluiden toimintoihin halutaan liittää asiakaskohtaisia laajennuksia, jotka liittyvät ESP-palvelualustan ulkopuolisiin järjestelmiin. Esimerkiksi Pepissä palvelut lähettävät muutoksista viestejä JMS-jonoon. Tapahtuma-palvelu puolestaan kuuntelee jonoon tulevia viestejä ja lähettää niitä eri integraatioille.

Integraatiot toteutetaan Apache Camelin avulla ja ne asennetaan ServiceMixiin:

- Integraatioita voidaan hallita ServiceMixin konsolissa kuten muitakin OSGi moduuleja
- Kuvassa palvelu 1 lähettää viestin (esim. opintojakson luonti) JMS-jonoon. Integraatio 1 kuuntelee jonoa, jolloin palvelun jonoon lähettämä viesti laukaisee integraation (esim. työtilojen luonti ja tietojen vienti Winhaan)



Camel integraatioista voidaan myös kutsua ServiceMixiin asennettuja OSGi-palveluita tai julkaista uusia palveluita:



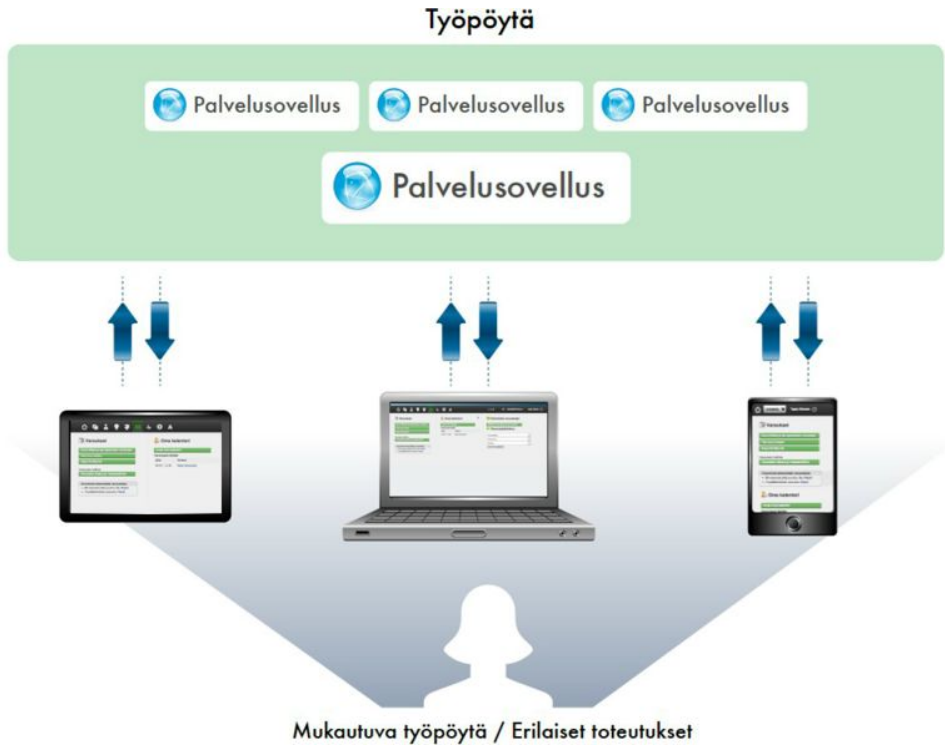
Palvelukerroksessa käytetyt teknologiat palveluiden ja integraatioden kehittämisessä

Teknologia	Kuvaus	Lisätietoa
Java	Ohjelmointikieli, jolla palvelukerroksen komponentit on ohjelmoitu	
OSGI	Java-pohjainen plugin-teknologia	OSGI
ServiceMIx	Integraatiotuote tai eri integraatiotuotteiden muodostama kokonaisuus, joka sisältää keskeiset toiminnot palvelupohjaiselle järjestelmälle	ServiceMix
CXF	Kirjasto, jonka avulla rajapinnat voidaan	CXF

	julkaista SOAP Rest-tyyppisinä käyttöliittymille ja muiden järjestelmien käytettäväksi.	
Spring	inversion of control (IoC) -sovelluskehys, joka helpottaa eri komponenttien sitomista toisiinsa sekä ohjelmoitia tietokantaoperaatioissa ja transaktioiden käsittelyssä.	Spring
Camel	Komponentti viestin välitykseen ja reititykseen	Camel

Käyttöliittymäkerros

Palvelualustan (ESP) käyttäminen ei edellytä sitoutumista mihinkään tiettyyn käyttöliittymäteknologiaan. Käyttöliittymät on kuitenkin päätetty tehty portletteina ja ajonaikaisena ympäristönä käytetään Liferay portaalia. Liferayn avulla koostetaan käyttäjälle hänen rooliinsa perustuva työpöytä.



Käyttöliittymien käyttämät teknologiat

Teknologia	Kuvaus
Java J2EE	Java-pohjainen alusta
Struts	Javalle tehty sovelluskehys, jota käytetään käyttöliittymän logiikan rakentamiseen
FreeMarker	Javalle tehty sovelluskehys, jonka avulla tehdään lomakkeet/näkymät.
Liferay	Java-pohjainen portaali, jonka avulla voidaan koostaa erilaisia työpöytiä/koostenäkymiä sekä hallita niihin liittyviä käyttöoikeuksia.
jQuery	Javascript-kirjasto

Kehitysmalli

Yleistä

Kehitys tehdään arkkitehtuurissa määritellyillä teknologioilla, jotta uusien moduulien jatkokehitys, ylläpito ja testaus olisi mahdollisimman suoraviivaista. Linjatuista teknologiaratkaisuista ei siten saa poiketa ilman erillistä päätöstä. Tämä koskee myös uusien teknologioiden ja sovelluskehysten hyödyntämistä. Kehitysmallia revisioidaan muutaman vuoden välein, yhteensopivuus säilyttäen.

Käyttöliittymien osalta tulee huomata, että esimerkiksi mobiilisovelluksien tekemisessä voidaan hyödyntää kulloinkin sopivinta käyttöliittymäteknologiaa, mutta silloin kun kehitetään käyttöliittymiä käytettäville työpöydille (Liferay), tulee käyttöliittymät tehdä yllä kuvatuilla teknologioilla.

Moduulit

Toteutus aloitetaan tekemällä moduulisuunnitelma, joka selventää toteutukseen liittyvät tekniset ratkaisut ja mahdolliset ongelmakohdat. Lisäksi sen avulla varmistutaan, että toteutus on ESP-käytäntöjen mukainen ja linjassa aikaisempien toteutusten kanssa. Moduulisuunnitelma liitetään osaksi koko järjestelmän kattavaa teknistä dokumentaatiota. Moduulisuunnitelma sisältää seuraavat kohdat:

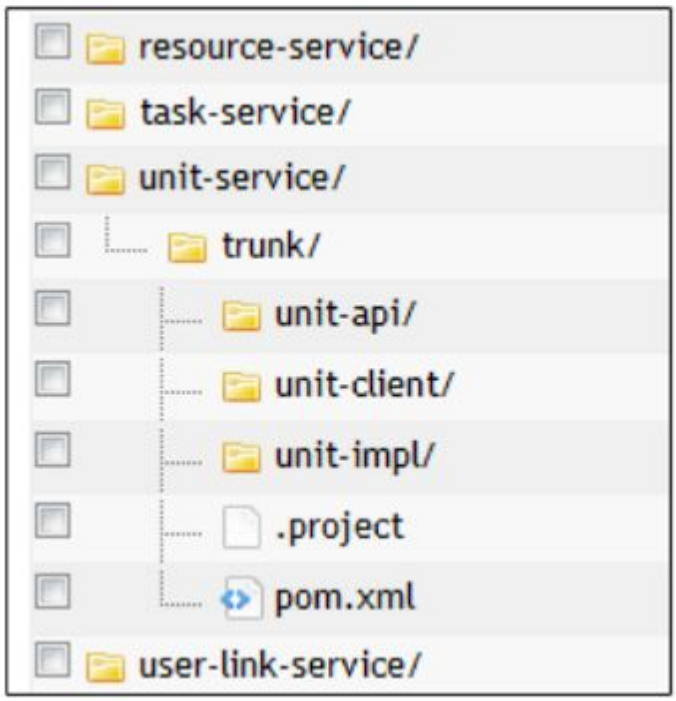
- moduulin tarkoitus
- linkitykset käyttötapauksiin
- moduulin riippuvuudet
- moduulin rakenne ja tiedostot

- asennuksen kuvaus
- moduulin tekniset riskit ja ongelmakohdat
- kehitysympäristö
- poikkeuskäsittely ja virhetilanteista toipuminen
- moduulin testaus

Versionhallinta

Kaikki koodi tallennetaan projektin Subversion-versionhallintaan.

Alla olevassa kuvassa on esimerkkirakenne versionhallinnassa.



Kokoonpanon hallinta

Ohjelmakoodin käännöstyökaluna käytetään Mavenia (http://maven.apache.org/), joka on Java-ohjelmoinnissa http://maven.apache.org de facto -standardin asemassa. Pääominaisuuksinaan Maven yhtenäistää sovellusprojektin koostamisen ja tiedostohierarkian sekä tarjoaa tavan hallita kirjasto- ja projektiriippuvuuksia.

Jatkuva integrointi toteutetaan Bamboo-työkalun avulla. Se automatisoi sovellusprojektin koostamisen, yksikkötestaamisen ja teknisen laadunvarmistuksen. Käytännössä sen avulla käännetään moduulin lähdekoodi heti, kun ohjelmistosuunnittelija on tehnyt muutoksia ja ajetaan yksikkö- ja integraatiotestit.