Ohjelmistoarkkitehtuurit, syksy 2012

Harjoitus 5, 10.10. – 11.10.2012

1. Ulkopuolisia liitännäiskomponentteja (plugin) käytetään joissain sovelluksissa ohjelmiston muuntelutekniikkana. Selvitä *Gecko*-selainmoottorin (jota käytetään Firefox-selaimessa) liitännäiskomponenttien (Gecko Plug-In API) perusominaisuudet ja vastaa seuraaviin kysymyksiin:
    i. Mihin Gecko-liitännäisiä käytetään?
    ii. Miten ne toimivat?
    iii. Miten selain löytää liitännäiset?
    iv. Minkälainen liitännäisen rakenne on? Minkälainen on Gecko-moottorin ja liitännäisen välinen rajapinta?

    i) Selaimen vakiotoimintojen laajentamiseen, flash plug-in, videoiden toistaminen, java-tuki (appletit)

    ii) Toimii browserin kontekstissa, selain tunnistaa sivulla käytetyn mime-tyypin, etsii vastaavan plug-inin, plug-in renderöi mediakontentin selaimen ikkunaan käyttäen selaimen tarjoamia palveluja, voi olla myös ”näkymätön”, joka ei tuota mitään näkyvää selainikkunaan

    iii) Selain käy läpi käynnistyessään käyttöjärjestelmästä riippuvat kansiot, joista se etsii plug-ineja ja rekisteröi löytämänsä plug-init vastaavalle mime-tyypille. Plug-ineja voi ehkä ottaa käyttöön myös suoritusaikana?

    iv) Kasa C-funktiota – tietty konventio funktioiden nimeämiseen, jotta selain osaa tunnistaa ne -> käytännössä plug-in on DLL, jonka selain lataa. Gecko moottori tarjoaa rajapinnan, jonka kautta plug-in käyttää selaimen tarjoamia palveluja. Hieman epäselvää, millaisessa hiekkalaatikossa plug-inin koodia suoritetaan…

    Katso lisäinfoa: https://developer.mozilla.org/en-US/docs/Gecko_Plugin_API_Reference/Plug-in_Basics#How Plug-ins Work

    Plug-in:n security-mallista puhuttiin, mutta selvyyttä ei oikein tullut. Vaikuttaa siltä, että plug-in periaatteessa voi tehdä mitä vaan selainkin saa tehdä, esimerkiksi lukea ja kirjoittaa tiedostoja käyttäjän oikeuksilla. Näistä linkeistä löytyy jotakin infoa:

    http://www.w2spconf.com/2009/papers/s1p1.pdf
    http://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-roberto_liverani-nick_freeman-abusing_firefox.pdf (koskee laajennoksia/extensions)

2. OSGi ’standardi’ on pohjana monissa java-ympäristön liitännäiskomponenttitoteutuksissa. Selvitä tähän tekniikkaan pohjautuen tarkemmin komponenteille ja alustalle asetetut vaatimukset, komponenttien yhteistyö ja uusien liitännäiskomponenttien käyttöönotto (materiaalia esim: http://www.vogella.de/articles/OSGi/article.html, http://eclipse.org/equinox/documents/quickstart.php)

    Rakenne: JAR pakkauksia, manifestissa OSGi kohtaisia metatietoja: nimi, riippuvuudet, tarjotut ja vaaditut rajapinnat, aktivaattoriluokka, alustavaatimukset.
    Alusta (OSGi) tarjoaa riippuuvuuksien hallinnan (lataa riippuvat bundlet), tarkistaa ajonaikana että ei yritetä käyttää muita kuin sallittuja ulkoisia riippuvuuksia, piilottaa ei-julkiset (ei-tarjotut) rajapinnat muilta bundleilta.

OSGi määrittelee abstrakteja palveluita (esim loggaus). Bundlet voivat tarjota toteutuksia näille palveluille, jotka alusta rekisteröi, ja jotka se käynnistää, kun niitä joku haluaa käyttää.

Katso myös http://www.cs.helsinki.fi/u/aptuovin/arkkit/s12/OSGi.pdf

3. Tutustu Ventus-merkkisten kotikäyttöön tarkoitettujen sääasemien ominaisuuksiin http://www.ventusdesign.com/products/products-weatherstations/
Minkälaisia tuoteperheitä sääasemista voisi muodostaa ja millä perusteilla?

Perheitä voi muodostaa monella tavalla. Järkevintä tuotteiden myymisen kannalta on ajatella asiaa käyttäjän ja potentiaalisten asiakasryhmien kannalta. Esimerkiksi:

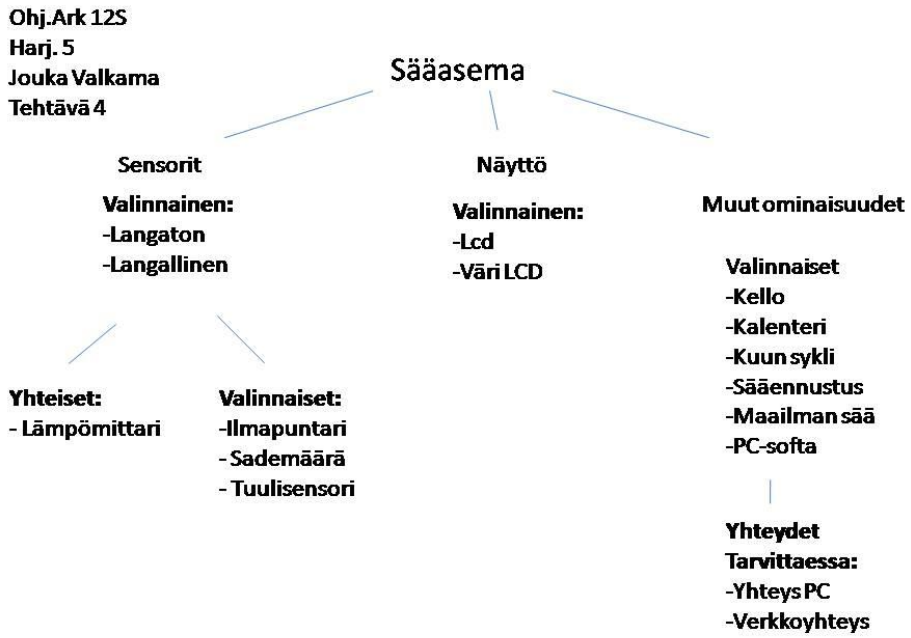
- Peruslämpömittarit
- Itsenäiset sääasemat (stand-alone weatherstation)
- Internet yhteyden vaativat sääsemat (connected weatherstation)

Näiden kategorioiden sisällä voi sitten käyttää alaryhmittelyjä: basic/economy (vähän toimintoja, halpa), mid-range (useita toimintoja), top-line (paljon toimintoja, kallis), style (useita toimintoja, muotoiluun ja ulkoasuun erityistähuomiota).

Toteutuksen kannalta taas voidaan katsoa laitteistoa ja komponentteja, sekä arkkitehtuuria ja miettiä niiden uudelleenkäyttöä, ja muodostaa tältä pohjalta ”järjestelmäperheitä” (system family), jos samojen teknisten komponenttien jakaminen on ryhmittelyn pohjana.

4. Laadi listaus Ventus-sääasemien piirteistä (features) ja selosta millaista muuntelua (variability) niissä on. Lisää sopivia välitasoja listaukseen (abstrakteja piirteitä), jotta saat aikaan hierarkisen, puumaisen rakenteen. Voit käyttää yksinkertaista tekstuaalista esitystapaa tai luennoilla 4.10. kalvolla 27 esitetyn kaltaista UML-piirremallia muunneltavuuden kuvaamiseen.

Yksi mahdollisuus:



Ja toinen:

Abstract feature	*Concrete feature*	*Concrete sub-feature*
remote management		
	PC application	
connecticity		
	wireless PC transmitter	
	USB connection	
	wireless sensors	
	wired connection to sensors	
	RF receiver for meteotime broadcasts	
Sensor		
	Anemometer	
		battery powered
		solar-cell charging
	Pressure	
	Temperature	
		inside
		outside
	Rain gauge	
	humidity	
		inside
		outside
	Soil sensor	
Time management		
	clock	
		alarm
		radio controlled
		time-zone
	calendar	
weather forecast		
	meteotime 4-day satellite forecast for 60 cities	
		2-day extension or 30 cities
	barometric	
display		
	type	
		color
		b&w
	size	
	touch	
weather information		
	temperature	
		inside
		outside
	humidity	
		inside

	outside
soil temperature soil humidity pressure wind direction wind speed wind chill factor dew point heat index trends memory	
	temperature, humidity and atmosph
meteotime weather alert celestial	
	moon phase sunrise, sunset
programmable ice warning	

mandatory
optional
alternative