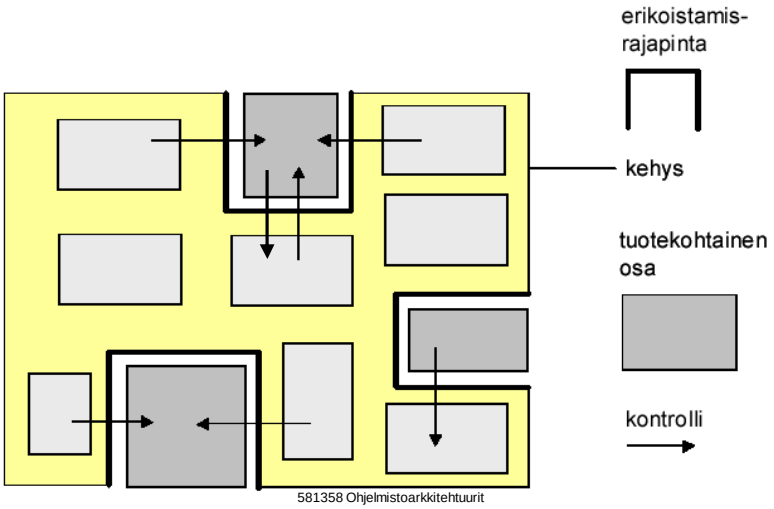
Ohjelmistoarkkitehtuurit

Ohjelmistokehykset

Ohjelmistokehykset (software frameworks)

- Osittain abstraktiksi jätettyjä ***ohjelmistorunkoja***, joita eri tavoin täydentämällä saadaan rakennettua kokonaisia uusia sovelluksia tai sovelluksen osia
- Tavoitteena laajamittainen ja systemaattinen ohjelmistojen (sekä koodin että yleisrakenteen eli arkkitehtuurin) ***uudelleenkäyttö***
  - Suosittu tekniikka tuoterunkoarkkitehtuurien toteuttamiseksi
- Olioperustaisissa kehyksissä ***ohjelmistorunko*** = kokoelma luokkia, komponentteja ja rajapintoja
  - Ohjelmistokehykset sisältävät ohjelmakoodia, joten ne ovat sidoksissa ohjelmointikieleen
  - Kehys on usein osa laajempaa *ohjelmistoalustaa* (platform) ja kehykseen voi kuulua omia työkaluja ja apuvälineitä

Ohjelmistokehykset



Ohjelmistokehykset

- Kehykset eroavat (luokka-, funktio-) *kirjastoista* (library)[1]
  - *Inversion-of-control:* kehyksen sisäänrakennettu koodi ja sen logiikka ohjaa sovelluksen suoritusta (kontrollinkulkua), ei sovelluskohtainen koodi kuten aliohjelmakirjastoja käytettäessä
  - *Oletustoiminnallisuus:* kehys tarjoaa käyttökelpoista oletustoiminnallisuutta, ei pelkästään tynkätoteutuksia (no-ops, stubs)
  - *Laajennettavuus:* kehyksen käyttäjä voi valikoiden syrjäyttää, erikoistaa tai lisätä kehyksen tarjoamaa toiminnallisuutta oman sovelluksena tarpeisiin kehyksen tekijän etukäteen määräämillä tavoilla
  - *Kehyksen muuttumaton koodi:* kehyksen koodia ei ole (yleensä) tarkoitettu käyttäjän muutettavaksi - paitsi laajentamalla sitä tietyillä tavoilla erikseen määrätyissä kohdissa (kts. edelllinen kohta)

[1] http://en.wikipedia.org/wiki/Software_framework

Ohjelmistokehykset

- Ensimmäinen laajalti käytetty ohjelmistokehys: Smalltalk-80-ympäristön *Model-View-Controller*
  - Alkuperäisen Model-View-Controller-kehyksen arkkitehtuurin pohjalta määritelty MVC-arkkitehtuurityyli
- Erityisesti käyttöliittymätoteutukseen on kehitetty useita kehyksiä työasemasovelluksiin, web-sovelluksiin, useille eri ohjelmointikielille, kaupallisia – ei kaupallisia

Ohjelmistokehykset

- Käyttöliittymäkehysten menestys ja suuri määrä leimasivat ohjelmistokehykset pitkäksi aikaa pelkästään käyttöliittymien toteutukseen soveltuvaksi tekniikaksi
- Ohjelmistokehyksiä on kuitenkin toteutettu monille sovellusalueille
  - Esimerkiksi: hypermediajärjestelmät, kaavioeditorit, käyttöjärjestelmät, kääntäjät, laskutusjärjestelmät, palohälytysjärjestelmät, laitetuotantolinjojen mittausohjelmistot, eLearning, web-sovellukset, …
  - Käyttöliittymäkehykset ovat yleiskehyksiä, jotka keskittyvät käyttöliittymäosan toteutukseen, mutta eivät rajaa sovellusaluetta.
    - *Muut kehykset ovat yleensä enemmän sovellusalueesta riippuvia*

Ohjelmistokehykset

- Kehys voi olla kokonaisvaltainen koko sovellusta hallitseva tai osittaisongelmaan tarkoitettu tukikehys
  - esim. käyttöliittymäkehykset rajautuvat käyttöliittymän toteutukseen, etäkäyttökehykset etäkäytön hallintaan
- Ohjelmistokehys ei ole valmis ohjelmistototeutus, vaan toteutus saadaan aikaan kehystä täydentämällä tai mukauttamalla

Ohjelmistokehykset

- Ohjelmistokehyksen ***erikoistaminen*** (framework specialization, framework adaptation) = ohjelmiston (osan) toteuttaminen täydentämällä kehyksen tarjoamaa ohjelmarunkoa
  - Abstraktien luokkien erikoistaminen,
  - Toiminnallisuuden koostaminen kehyksen konkreettisista luokista
- ***Sovellus****kehys (application framework)* = kehys, josta voidaan erikoistaa kokonainen sovellus
- ***Komponenttikehys*** (framelet) = ”minikehys”, jonka erikoistamisen tuloksena syntyy yksittäinen komponentti

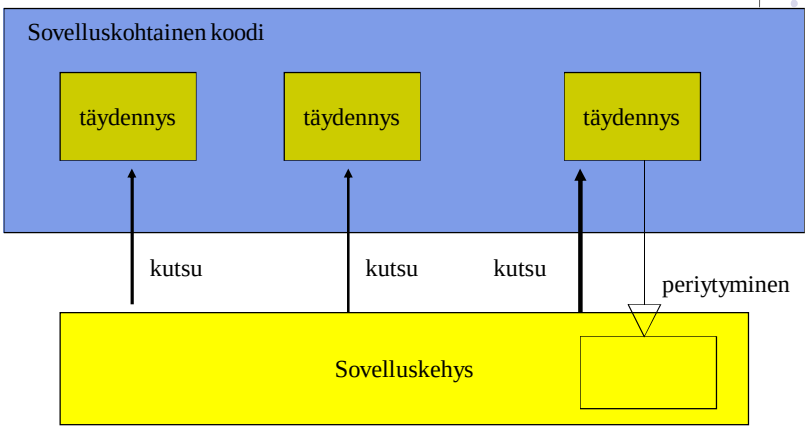
Ohjelmistokehykset

- ***Laajennoskohta***, ***variaatiopiste*** (hot spot, variation point) = kehyksen ”aukkokohta”, jota täydentämällä voidaan sovelluksen puolella varioida ja/tai ottaa käyttöön tietty kehyksen toiminnallisuus/ominaisuus
- ***Erikoistamisrajapinta*** (specialization interface) = laajennoskohtien ja niihin liittyvien vaatimusten kokoelma
  - Kehyksen erikoistamisrajapinta on tyypillisesti paljon monimutkaisempi kuin yksittäisen komponentin rajapinta
  - Erikoistamisrajapinnan laajennoskohtien välillä on riippuvuuksia, joiden ymmärtäminen on edellytys kehyksen oikealle käytölle

Ohjelmistokehykset

- *Erikoistamiseen voidaan käyttää* esim.
  - rajapintojen toteutusta
  - periytymistä ja syrjäyttämistä (jos kieli tarjoaa)
  - olioiden luontia ja kompositiota
  - geneerisiä rakenteita
  - myöhäistä sidontaa ja sitä tukevia rakenteita
- Sovelluskehyksissä päälogiikasta vastaa kehys ja sovelluskohtaiset erityistoimet toteutetaan täydennyksinä

Ohjelmistokehykset

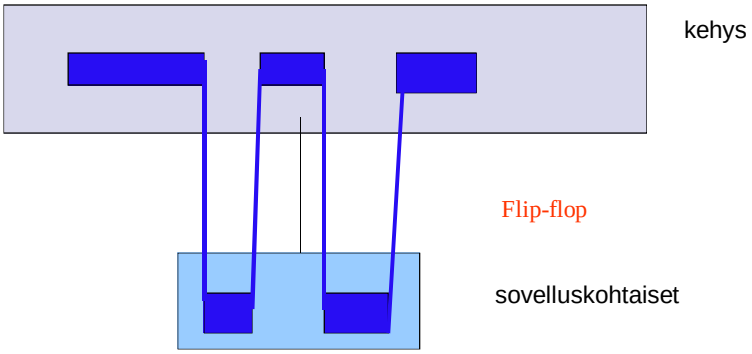


Kutsuissa käytössä ns. käänteinen kutsurakenne =
kehys kutsuu dynaamista sidontaa
hyväksikäyttäen sovelluskohtaisia täydentäviä moduuleja
(Hollywood-periaate: don't call us we call you)

9.10.2014 581358 Ohjelmistoarkkitehtuurit 11

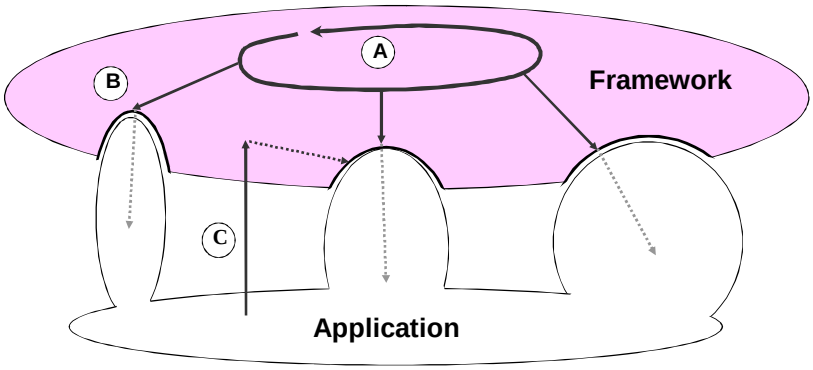
Ohjelmistokehykset

Kehyksissä ohjelman kontrollia ohjaavat vuoroin
kehys vuoroin sovellukohtaiset täydennykset



9.10.2014 581358 Ohjelmistoarkkitehtuurit 12

Ohjelmistokehykset



Sovelluskehyksessä
(A) Pääkontrolli (”main-silmukka”) kehyksen puolella.
(B) Takaisinkutsut sovelluksen puolelle.
(C) Kehyksen tarjoamien palveluiden kutsuminen.

Ohjelmistokehykset

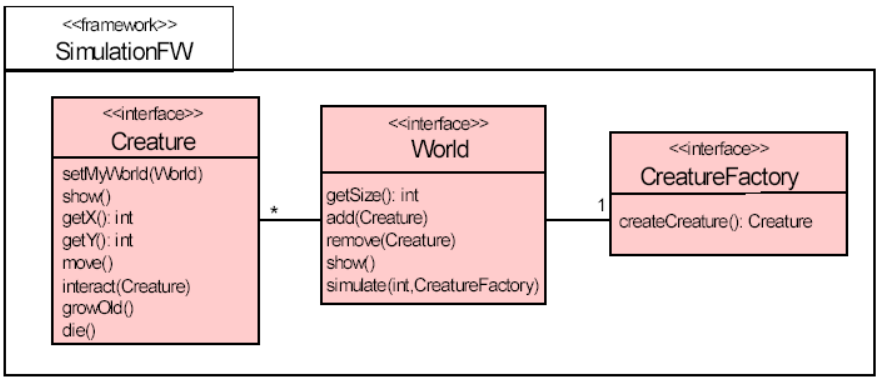
- Kehyksissä sovelletaan tyypillisesti suunnittelumalleja (design patterns) erilaisten hyvinä pidettyjen asioiden aikaansaamiseksi
  - Joidenkin mallien soveltaminen on välttämätöntä kehyksen erikoistamismahdollisuuksien kannalta
- Kehyksen koodi sisältää tyypillisesti useiden suunnittelumallien ilmentymiä
  - ”Alkuperäiset” suunnittelumallit on ”löydetty” analysoimalla olemassa olevia kehyksiä (esimerkiksi Smalltalk, HotDraw)
  - Suunnittelumallit sopivat usein kehysten dokumentointiin [Johnson, 1992]
- Suunnittelumalleissa on tyypillisesti kehykseen kuuluva osa (esim. abstraktit luokat) ja tuotekohtainen osa (esim. vaihtuvat konkreettiset luokat)

(Olio-)Ohjelmistokehykset

- Eri tyyppisiä kehyksiä:
  - Abstrakti kehys
  - Muunneltava kehys (white box framework)
  - Pistokekehys (plugin framework)
  - Koottava kehys (black box framework)

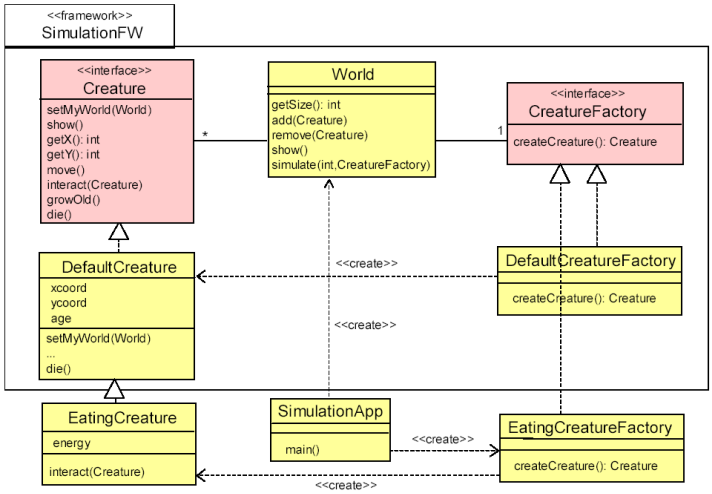
(Olio-)Ohjelmistokehykset

- Abstrakti kehys: vain abstrakteja luokkia (tai rajapintoja); ei toiminnallisuutta
- vuorovaikutus on toteutettava sovelluskohtaisesti



(Olio-)Ohjelmistokehykset

- Muunneltava kehys: Tarjoaa rajapinnat ja ohjelman päälogiikan
- Myös periminen ja syrjäyttäminen erikoistamistekniikkoina

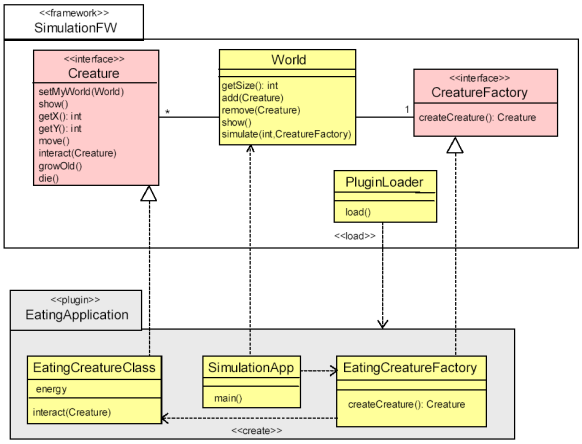


(Olio-)Ohjelmistokehykset

- Muunneltavan kehyksen yhteydessä kehyksen käyttäjän on oltava hyvin perillä kehyksen toiminnasta kyetäkseen käyttämään sitä
- Metodien väliset riippuvuudet voivat aiheuttaa ongelmia. Yhden syrjäyttäminen voi edellyttää jonkin toisenkin syrjäyttämistä, jolloin luokkien määrän kasvaa nopeasti.

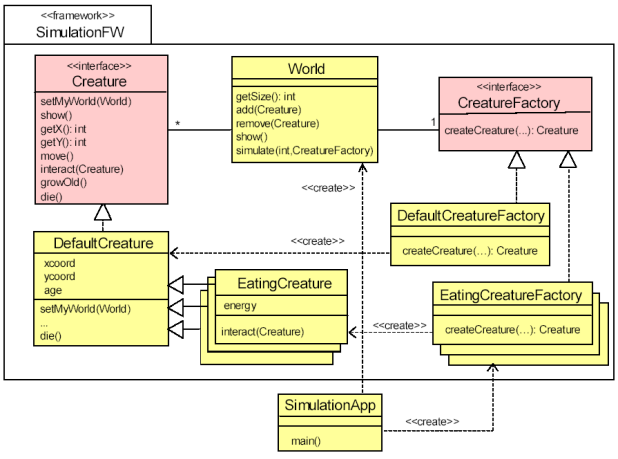
(Olio-)Ohjelmistokehykset

- Pistokekehys: erikoistetaan (miltei yksinomaan) rajapintoja; erikoistus tarjotaan yhtenäisenä plugin-komponenttina, esim Eclipse plugins



(Olio-)Ohjelmistokehykset

- Koottava kehys: Koostetaan sovellus kehyksen tarjoamista valmiista komponenteista; ei takaisinkutsuja sovelluskoodiin



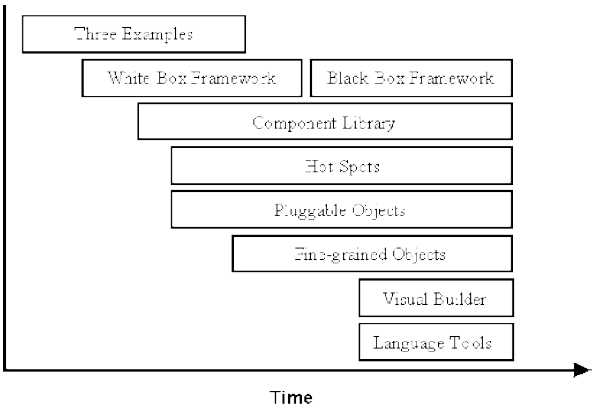
(Olio-)Ohjelmistokehykset

- Koottavat kehykset
  - Uutta toiminnallisuutta saadaan yhdistelemällä tarjolla olevia olioita uusin tavoin ja säätämällä toimintaa parametrein
  - Perintä korvataan toimintojen delegoinnilla ja parametroinnilla

(Olio-)Ohjelmistokehykset

- Koottavat kehykset …
  - Käyttäjän ei tarvitse tuntea kehyksen sisäisiä yksityiskohtia, mutta on tiedettävä millaisia olioita voi luoda ja miten niitä voi kytketään toisiinsa.
  - Luotavien olioiden luokat ovat valmiiksi määriteltyjä. Luotavaan olioon voi vaikuttaa parametreilla.
  - Koottavat kehykset (black-box) ovat *yleensä helpompikäyttöisiä* kuin läpinäkyvät kehykset. Toisaalta niitä on *vaikeampi kehittää* koska kehittämisessä pitää varautua monipuolisemmin eri käyttötilanteisiin.
  - Koska joustavuudella on ennakkomäärittelyn asettamat rajat, eivät koottavat kehykset voi soveltamisalueeltaan olla kovin laaja-alaisia

Ohjelmistokehyksen evoluutio



Kehyksen elinkaaren aikana tyypillisesti esiintyviä ”patterneja”
Roberts: http://st-www.cs.illinois.edu/users/droberts/evolve.html

Ohjelmistokehyksen evoluutio

- Kolme esimerkkijärjestelmää
  - kehyksen laatiminen edellyttää hyvää perehtyneisyyttä sovellusalueeseen
  - kukaan ei pysty kehittämään kehystä tyhjästä
  - perehtyneisyys on saavutettavissa vain tekemällä alueelle sovelluksia, joissa kehyksestä voisi ajatella olevan hyötyä - tätä kautta nähdään
    - *mikä toistuu sovelluksesta toiseen (siis kuuluu kehykseen)*
    - *mikä on sovellusspesifiä (kuuluu erikoistukseen)*
    - *kehys muotoutuu esimerkkien pohjalta*

Ohjelmistokehyksen evoluutio

- Kolmen esimerkkisovelluksen tarkoitus
  - kolmesta voi jo tehdä yleistyksiä
  - sovellukset hieman toisistaan poikkeavia
  - yksi tiimi voi tehdä ne peräkkäin
    - *kokemus heti hyödynnettävissä*
  - tai kolme tiimiä voi tehdä ne rinnakkain
    - *laajempi näkemys*
  - täydellisen sovelluksen asemesta prototyyppikin antaa käsitystä

Ohjelmistokehyksen evoluutio

- Yleistämällä on helpointa edetä läpinäkyvään kehykseen.
- Kun läpinäkyvää kehystä käyttäen kehitetään sovelluksia havaitaan, että joudutaan luomaan samoja aliluokkia useisiin sovelluksiin ->Perustetaan kirjasto
  - *kirjastoa perustettaessa ei välttämättä ole tietoa onko jokin komponentti yleiskäyttöinen vai sovelluskohtainen*
  - *aluksi kirjastoon voidaan ottaa kaikki konkreettiset luokat*
  - *myöhemmin karsitaan sovelluskohtaisiksi osoittautuneet*
  - *usein käytetyistä luokista tulisi johtaa kehykseen liitettävä abstraktio*

Ohjelmistokehyksen evoluutio

- Hot spots (erikoistamiskohdat)
- Kehystä käytettäessä joudutaan kirjoittamaan samanlaista koodia toistuvasti eri sovelluksiin, osa koodista taas muutuu sovelluskohtaisesti (erikoistamiskohdat ovat kohtia, joissa esiintyy muuttuvaa koodia) - näiden limittyminen aiheuttaa ongelmia ylläpidossa
  - Muuttuva koodi tulisi eristää ja mieluiten kapseloida luokiksi.
  - Samana toistuva koodi tulisi sisällyttää kehykseen
  - Luokkajakoa voidaan joutua hienontamaan

Ohjelmistokehyksen evoluutio

- Pluggable Objects (pistokkaat)
  - eri sovellusten yhteydessä luodut aliluokat saattavat erota toisistaan vain vähän, esimerkisi vain yksi syrjäytetty metodi - useat luokat hankaloittavat ratkaisun ymmärtämistä
  - Pienet eroavaisuudet voi hoitaa välittämällä muuttuva koodi (pistokas) oliolle parametrina luonnin yhteydessä - tekniikka riippuu ohjelmointikielestä (viite rajapinnan toteuttavaan olioon, viite funktioon,…)
  - luokkarakenne hienojakoistuu

Ohjelmistokehyksen evoluutio

- Black-box framework
  - black-box kehykset ovat käyttäjälle helpompia
  - Suositeltavaa perintähierarkian käyttö komponenttikirjaston jäsentelyssä ja kompositiopohjaiset kytkennät kehyksessä
  - Muuntetaan perintä kompositioksi (perinnän sijaan delegointi)
- Visual builder
  - Lähtökohtana on black-box kehyksen olemassaolo.
  - Työkalu avustaa kokoamista --> ollaan saatu aikaan sovelluskehitin
- Language Tools
  - kehitinympäristö edellyttää yhteensopivia välineitä ratkaisun tarkasteluun ja virheiden jäljitykseen

9.10.2014 581358 Ohjelmistoarkkitehtuurit 29

Ohjelmistokehysten riskejä

- Tekninen vaativuus ja monimutkaisuus
  - Arkkitehdin tunnettava hyvin sovellusalue ja joustavuuteen liittyvät oliotekniikat (esim. suunnittelumallit)
  - Abstraktius voi tehdä kehyksestä erikoistajille hankalasti ymmärrettävän
- Kehysten yhdistely voi olla ongelma
  - Mitä tehdään, jos halutaan käyttää useampaa kehystä, joista kukin määrittelee pääkontrollisilmukan?

9.10.2014 581358 Ohjelmistoarkkitehtuurit 30

Ohjelmistokehysten riskejä

- Monoliittisuus
  - Kehys saattaa kasvaa suureksi ja hallitsemattomaksi
- Laadullinen varianssi
  - Variaatiopisteet liittyvät useimmiten toiminnallisuuden muokkaamiseen
  - Laatuominaisuuksia variointi sen sijaan yleensä hankalaa (esim. tietoturvan lisääminen, suorituskyvyn optimointi, …)
- Dokumentointi
  - Olennaista on erikoistamisrajapinnan kuvaus
  - Tälle ei kuitenkaan vakiintunutta kuvaustapaa

Ohjelmistokehysten riskejä

- Toteutuksen ongelmia
  - Joustavien ratkaisujen toteuttaminen edellyttää asiantuntemusta (esim. suunnittelumallit auttavat)
  - Joustavuus → monimutkaisuus, abstraktit käsitteet
    - *Ylläpito vaativaa*
    - *Testaus? (vrt. tuoterunkojen testaus)*
- Käytön ongelmia
  - Miten kehysten käyttö pitäisi dokumentoida?
    - *Keittokirjat (cookbooks)*
    - *Suunnittelumallipohjainen dokumentointi*
  - Riittääkö tavanomainen dokumentaatio erikoistamisrajapinnan kuvaukseen?
  - Työkalutuki sovellusten rakentamiseen