Ohjelmistoarkkitehtuurit

Arkkitehtuurin kuvaaminen
2
Rajapinnat

24.9.2012 1

Arkkitehtuurin kuvaaminen

- Arkkitehtonisen näkymän esittäminen
  - Luonnollinen kieli + vapaamuotoinen grafiikka
    - *taustalla on oltava joku malli, jonka mukaisia asioita kuvaukseen otetaan mukaan,*
      - *- noudatetaanko mallia – malli elää*
      - *- tulkintojen yhdenmukaisuus kyseenalaista, etenkin grafiikan osalta*
      - *- tuleeko kaikki tarpeellinen mukaan, ohjaava tekijä puuttuu*
      - *+ Esitystapaa voi säätää tarpeen mukaan - havainnollisuus voi olla parempi kuin formaaleissa 'laatikkoverkoissa'*

24.9.2012 2

Arkkitehtuurin kuvaaminen

- Arkkitehtonisen näkymän esittäminen
  - Johonkin ohjelmiston mallinnustekniikkaan pohjautuva kuvaus
    - *esimerkiksi UML-kaaviotekniikat*
      - *- Ovatko tekniikat oikealla abstraktiotasolla, ohjelmointiläheisyys ongelmana*
      - *-+ Taustalla on standardoitu malli, mutta ovatko käsitteet tarpeeksi täsmällisiä ja hyvin määriteltyjä*
      - *- Liiallinen oliokeskeisyys tuottaa ongelmia*
      - *- Tulkinnat käsitteistä ja niiden käytöstä vaihtelevat*
      - *+ Staattisia ja dynaamisia näkymiä*
      - *+ Käyttö jatkuvasti lisääntymässä*
      - *+ Laajennusmekanismit*
      - *+ Hyvä työkalutuki*

24.9.2012 3

Arkkitehtuurin kuvaaminen

- Arkkitehtonisen näkymän esittäminen
  - Erityisellä arkkitehtuurikielellä laadittu kuvaus
    - *- kielet lähinnä tutkimuskäytössä*
    - *- ei juuri käytössä teollisuudessa*
    - *+ käsitteet täsmällisesti määriteltyjä*
    - *+ analysointimahdollisuuksia*

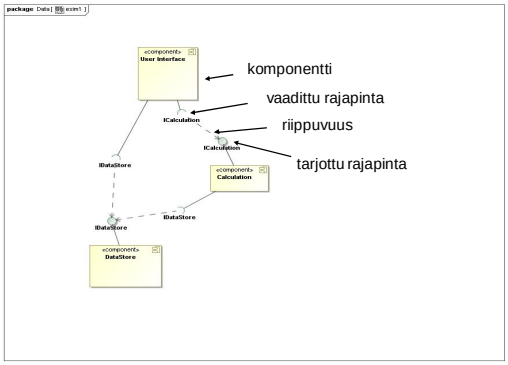
24.9.2012 4

Arkkitehtuurin kuvaaminen

- UML- tekniikat
  - Unified Modeling Language
  - UML-kielen tekniikat painottuvat oliokeskeiseen yksityiskohtaiseen ohjelmistosuunnitteluun
    - *Uusimmissa versiossa 2.0 mukana myös ylemmän tason tekniikoita, esimerkiksi komponenttikaaviot*
    - *Vaativat käsitteiden soveltamista*
    - *Kuvaukset ensisijaisesti graafisia – tulevat helposti isokokoisiksi*
    - *Raportointimuodot riippuvat työkaluista*
    - *Kuvaesitystä täydennettävä tekstikuvauksin*

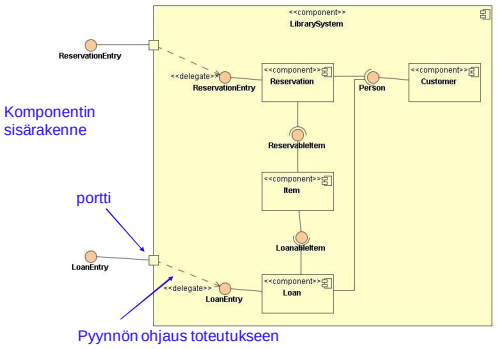
24.9.2012 5

UML-komponenttikaavio



6

UML-komponenttikaavio



Komponentin sisärakenne

portti

Pyynnön ohjaus toteutukseen

24.9.2012 7

UML:n laajennettavuus

- UML:n perustana yleinen oliopohjainen metamalli, joka kiinnittää mallinnuksen peruskäsitteet
- Muokattavuus mahdollistaa käsitteiden sovittamisen
- Käsitteitä voidaan erikoistaa käyttäen stereotyyppejä, leimoja (tagged values) ja semanttisia sääntöjä
  - Miltei mikä tahansa metamallin käsite (esimerkiksi luokka, olio, yhteys, ominaisuus, palvelu) voidaan erikoistaa stereotyypin avulla
  - Stereotyyppi nimeää pääkäsitteen alikäsitteen ja liittää siihen täsmäsemantiikan
    - *Periaatteessa stereotyyppiin voisi liittää myös oman graafisen esityksen*
    - *Sopiva työkalu mahdollistaa laajennusten tekemisen mallinnustilanteessa*
  - Leimoilla (nimi-arvo pari) voidaan kiinnittää joitain kohteiden ominaisuuksia
  - Semanttisilla säännöillä voidaan asettaa rajoituksia esimerkiksi käsitteiden kytkennöille.

24.9.2012 8

UML:n laajennettavuus

- UML-profiilit ovat valmiiksi määriteltyjä laajennoksia
  - SysML – järjestelmänkuvauskieli (ohjelmat, laitteet, ihmiset, ympäristö) – arkkitehtuuritasoista
  - SoaML – palvelukeskeisten järjestelmien kuvaaminen – arkkitehtuuritasoista
  - Useita hajautettujen arkkitehtuurien kuvausprofiileja - arkkitehtuuritasoista

24.9.2012 9

Arkkitehtuurin kuvaaminen

- Arkkitehtuurikieliä - kiinnitetty käsitteistö, yleiskäyttöisiä
  - Darwin
  - Rapide
  - Wright
- Arkkitehtuurikieliä sovellusaluekohtaisia
  - Koala (viihdeelektroniikka)
  - Weaves (väylät ja suodattimet)
  - AADL
- Arkkitehtuurikieliä – laajennettava käsitteistö
  - Acme
  - ADML
  - xADL

24.9.2012 10

Arkkitehtuurin kuvaaminen

- Perustuvat yleisiin arkkitehtuurielementteihin
  - Komponentit
  - Konnektorit (connector)
  - Portit
  - Rajapinnat (interface)
  - Pitkälti samoja asioita kuin UML:n komponenttimallissa
  - Usein sekä tekstuaalinen että graafinen esitys
  - Kuvauksen riittävyys analysointitarpeisiin
    - *Esim. vaadittu ja tarjottu rajapinta yhdenmukaisia*
    - *Simuloitavuus*
    - *Kapasiteettilaskelmat*

24.9.2012 11

Arkkitehtuurin kuvaaminen

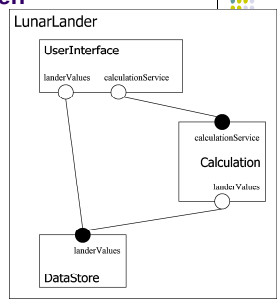
Darwin esimerkki

```
component DataStore{
    provide landerValues;
}

component Calculation{
    require landerValues;
    provide calculationService;
}

component UserInterface{
    require calculationService;
    require landerValues;
}

component LunarLander{
inst
    U: UserInterface;
    C: Calculation;
    D: DataStore;
bind
    C.landerValues -- D.landerValues;
    U.landerValues -- D.landerValues;
    U.calculationService -- C.calculationService;
}
```



(Taylor..)

24.9.2012 12

Komponenttien rajapinnoista

- Ohjelmistorakennetta voidaan tarkastella komponentteina
- Komponentti muodostaa ***toiminnallisen kokonaisuuden***, joka tarjoaa joukon loogisesti toisiinsa liittyviä palveluja
  - Palvelut käyttävät samoja tietorakenteita
  - Palveluja käytetään tyypillisesti samassa yhteydessä
  - Toiminnallisuus peruste komponentin olemassaololle
- Komponentti on työnjaon perusyksikkö kaikissa kehitystyön vaiheissa

24.9.2012 13

Komponenttien rajapinnoista

- Yleinen ohjelmistosuunnittelun periaate: erotetaan se ***mitä*** halutaan tehdä siitä ***miten*** toiminta toteutetaan
  - Komponentin ei pitäisi riippua tietyistä toisista komponenteista, vaan joukosta (abstrakteja) palveluja, joita muut komponentit tarjoavat
  - ”Abstrakti palvelu” määritellään ***rajapintana (interface)***

24.9.2012 14

Komponenttien rajapinnoista

- Minimaalinen rajapintamäärittely: ***palveluiden kutsumuodot*** (signatures)
  - kutsu voi olla muutakin kuin paikallinen tai etäproseduurikutsu
    - *Esim. viesti-/tapahtumapohjainen protokolla*
- Muut rajapinnassa määriteltävät asiat:
  - Palveluiden toiminnallinen määrittely – mitä tekee
  - Palveluiden ei-toiminnalliset ominaisuudet (esim. tilan ja ajan käyttö)
  - Sivuvaikutukset (= vaikutukset rajapinnan tarjoavan komponentin ulkopuolelle)
  - Mahdolliset palvelupyynnöistä aiheutuvat poikkeukset
  - Riippuvuudet globaaleista resursseista (globaalit muuttujat, tiedostot, tietovarastot)
  - Palveluiden väliset riippuvuudet (esimerkiksi oikea kutsujärjestys)

24.9.2012 15

Komponenttien rajapinnoista

- Rajapinnat määräävät tavat, joilla komponentit kommunikoivat keskenään
- Myös suurin osa arkkitehtuurityyleistä ja suunnittelumalleista perustuu rajapintojen käyttämiseen
- Huolellinen rajapintasuunnittelu edellytys
  - työn järkevälle jakamiselle
  - ohjelmiston keskeisten laatuominaisuuksien, kuten testattavuuden, ylläpidettävyyden ja muunneltavuuden varmistamiselle
- Rajapintasuunnittelu alkaa tyypillisesti heti keskeisten arkkitehtuuriratkaisujen tekemisen (esim. arkkitehtuurityylien valitsemisen) jälkeen

24.9.2012 16

Komponenttien rajapinnoista

- Ohjelmointikielissä rajapinnan käsite on kehittynyt vähitellen
  - Aliohjelmien kutsumuotojen määrittely (esim. Fortran)
  - Moduulit (Modula-2, Ada, Pascal): tietorakenteiden ja niitä käsittelevien operaatioiden kokoaminen; ulospäin näkyvien osien määrittely → ***abstrakti tietotyyppi***
  - Olioparadigma
    - *Rajapinnan kuvaaminen* **abstraktina luokkana**
    - **Rajapinnan** *(~ abstraktin luokan)* **laajentaminen** *uusilla operaatioilla (***periytyminen***)*
    - *Rajapinnan toteuttava komponentti* **konkreettisena luokkana**
    - *Rajapintojen ja ne toteuttavien komponenttien välinen N:M-suhde* **moniperiytymisen** *avulla (huom. luokkien vs. rajapintojen moniperintä)*
    - *Joissain kielissä rajapinta omana syntaktisena rakenteena: esim. Javan tai C#:n* **interface** *(vrt. C++:n kokonaan abstrakti luokka)*

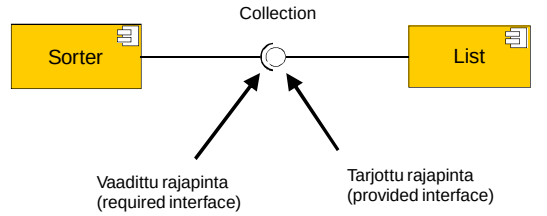
24.9.2012 17

Komponenttien rajapinnoista

- Komponentti voi joko tarjota (eli toteuttaa) tai vaatia rajapinnan → ***tarjotut*** ja ***vaaditut rajapinnat*** (provided and required interfaces)
- Tietty nimetty rajapinta on yleensä toisen komponentin vaatima ja toisen tarjoama
- Ohjelmointikielet, kuten Java tai C++, eivät tarjoa välineitä vaadittujen rajapintojen eksplisiittiseen kuvaamiseen (pitää kuvata ei-formaalilla dokumentaatiolla)

24.9.2012 18

Rajapinnat UML 2.0:ssa

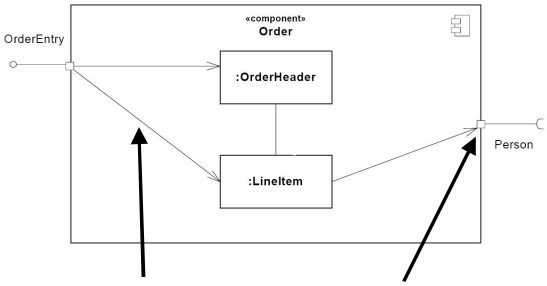


Vaadittu rajapinta
(required interface)

Tarjottu rajapinta
(provided interface)

24.9.2012 19

Rajapinnat UML 2.0:ssa



Konnektori (connector)

Portti (port)

24.9.2012 20

Rajapinnat UML 2.0:ssa

- Portti
  - Liitoskohta, jonka kautta komponentti on yhteydessä ulospäin
  - Mekanismi, jolla voidaan koota yhteen rajapintoja liittämällä ne samaan porttiin, esim. roolirajapinnaksi
  - Portteja ei välttämättä tarvitse käyttää mallinnuksessa – jokaisella rajapinnalla implisiittisesti oma porttinsa

24.9.2012 21

Rajapinnat

- Nykyaikaisissa ohjelmointikielissä rajapinnat määritellään operaatioiden kutsumuotoina
- Lisäksi kuvattava palveluiden semantiikka
- Rajatuilla sovellusalueilla palvelut voidaan määritellä formaalisti
  - Palveluiden ja ohjelmien oikeaksi todistaminen
  - Ei sovellu yleisiin, laajoihin järjestelmiin → epäformaalit semantiikan kuvaukset

24.9.2012 22

Palvelun sisältö

- Palvelun sisältö kuvataan usein ***tulo-*** ja ***jättöehtoina*** (pre/postconditions)
- Tuloehto kuvaa ehtoa, jonka on oltava voimassa ennen palvelun suoritusta
- Jättöehdon on oltava voimassa palvelun suorittamisen jälkeen
- Tulo- ja jättöehdot voidaan nähdä sopimuksena palvelun tarvitsijan ja palvelun tarjoajan välillä → ***sopimuspohjainen suunnittelu*** (design-by-contract)
- Tulo- ja jättöehdot voidaan aina kuvata epäformaalisti
- Jotkut ohjelmointikielet tarjoavat omat erikoistuneet mekanismit tulo- ja jättöehtojen esittämiseksi (esim. Eiffel)

24.9.2012 23

Palvelun sisältö

- Tulo- ja jättöehtojen lisäksi palvelua säätelevät ***invariantit*** = ehdot, joiden on oltava voimassa tietyissä tilanteissa ohjelman suorituksen aikana
- ***Luokkainvariantti***: oltava voimassa aina muulloin paitsi luokan operaatioiden suorituksen aikana

24.9.2012 24

Rajapinta ja sen toteutus

```
interface Observable {
  void addListener(Listener l);
  boolean isListenerRegistered(Listener l);
}

class ObservableImpl implements Observable {
  private Vector _listeners = new Vector();
  public void addListener(Listener l) {
    if (!isListenerRegistered(l))
      _listeners.addElement(l);
    else System.err.println(
      "Listener already registered");
  }
  public boolean isListenerRegistered(Listener l) {
    return _listeners.contains(l);
  }
}
```


Rajapinta ja sen toteutus

```
Observable o = new ObservableImpl();
Listener l = new ListenerImpl();
if (!o.isListenerRegistered(l)) {
  o.addListener(l);
}
else {
  // Error handling...
}
```

- Onko rajapinnan määrittelyssä jotain puutteita, jotka aiheuttavat ongelmia?
  - Puutteelliset vastuumäärittelyt, turhia tarkistuksia, käyttäjä ei voi käsitellä virhetilannetta

Rajapinta ja sen toteutus

- Rajapintamäärittelyn tarkentaminen tulo- ja jättöehdoilla (kuvitteellisessa Javan laajennoksessa):

```
interface Observable {
  //...
  void addListener(Listener l) {
    pre getListener(l) == null;
    post getListener(l) != null;
  }
  //...
}
```

- Ehdon rikkoutuminen voi aiheuttaa esimerkiksi poikkeuksen (vrt. Javan assertiot) – tämä riippuu kielen toteutuksesta
- Takaavatko tulo- ja jättöehdot, että palvelu toimii oikein?

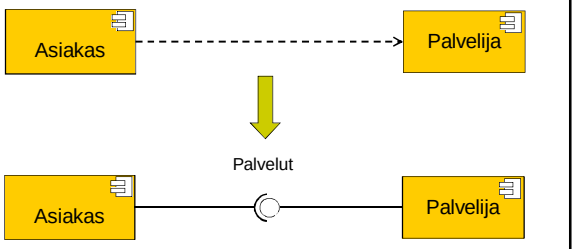
Poikkeukset rajapinnassa

- Standardi-Java tarjoaa mahdollisuuden määritellä rajapintaan poikkeuksia

```
interface Observable {
  // …
  void addListener(Listener l)
    throws ListenerRegisteredException;
  // …
}
class ObservableImpl implements Observable {
  // …
  public void addListener(Listener l)
    throws ListenerRegisteredException {
    if (isListenerRegistered(l)) {
       throw new ListenerRegisteredException(l);
    }
    _listeners.addElement(l);
  }
  // …
}
```


Rajapinnat

- Rajapinnan avulla palvelun käyttäjä voidaan tehdä riippumattomaksi palvelun konkreettisesta toteutuksesta



Rajapinnat

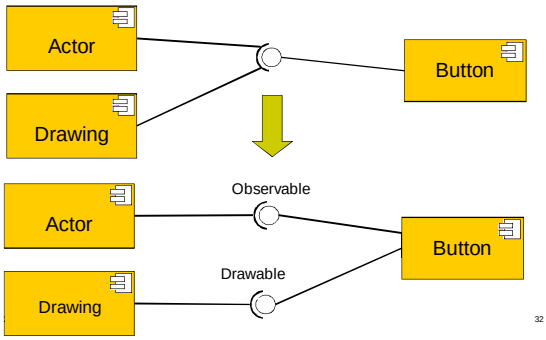
- Arkkitehtuurisuunnittelussa järjestelmän komponenttien vastuut tarkennetaan rajapintojen muodossa
- Jos komponentti A tarvitsee komponentin B palveluita, tulee B:n rajapinnasta A:n vaadittu rajapinta
- Usein komponentin tarjoama palvelujoukko kannattaa kuitenkin jakaa osiin siten, että komponentti toteuttaa monta erilaista rajapintaa.

Rajapinnat

- Komponentti tarjoaa palvelujaan toiselle komponentille aina jossain roolissa, joka ei välttämättä kata kaikkia komponentin palveluita
  - Asiakas käyttää komponenttia jossain roolissa, ei välttämättä kaikissa
- Erillinen rajapinta jokaiselle selkeästi identifioidulle roolille → ***roolirajapinnat***
- Komponentti voi toteuttaa useita roolirajapintoja
- Useat komponentit voivat toteuttaa saman roolirajapinnan

24.9.2012 31

Roolirajapinnat



32

Roolirajapinnat

- Roolirajapinnat muodostavat hienojakoisen rajapintajoukon
  - Muutos tarjotun rajapinnan määrittelyssä vaikuttaa niihin komponentteihin, jotka käyttävät rajapintaa – ei kaikkiin jotka käyttävät komponenttia
  - Muutos vaaditussa rajapinnassa vaikuttaa vain niihin, jotka toteuttavat rajapinnan

24.9.2012 33