S01-7 Komponens alapú szoftverfejlesztés 1

1. A szoftverfejlesztési modell fogalma.
2. A komponens és komponens modell fogalma.
3. UML kompozíciós diagram fogalma.
4. A szoftverarchitektúrák fogalma, összetevői.
5. A KobrA programfejlesztési modell alapjai.
6. A KobrA modell környezeti térképe: vállalati vagy üzleti modell, használati modell, strukturális modell, viselkedési modell.
7. Komponens specifikáció részei: funkcionális modell, viselkedési modell és strukturális modell.

A szoftverfejlesztési modell fogalma

Alapvetően eljárások, ajánlások gyűjteménye

- Segíti a szoftverfejlesztés teljes ciklusát
- Megmondja, hogy egy adott fejlesztési lépésben mit kell tenni és azt hogyan kell megtenni
- Egyetlen mondatba sűrítve a szoftverfejlesztési modell egy recept a programok létrehozására

Minden szoftver rendelkezik életciklussal, amely meghatározza létét a feladat kitűzésétől a program használatának befejeztéig. Az életciklus általában négy fő fázisra bontható:

1. specifikáció: a szoftver funkcionalitásának és megszorításainak megadása
2. tervezés és implementáció: a specifikációnak megfelelő szoftver előállítása
3. verifikáció és validáció: a szoftver ellenőrzése a specifikációnak történő megfelelésre
4. evolúció: a szoftver továbbfejlesztése a változó elvárásoknak megfelelően

A szoftverfejlesztési modell határozza meg az említett fázisok közötti kapcsolatot, időbeliséget.

Szoftverfejlesztési modellek például:

- Vízesés modell
- Spirál modell
- Komponensalapú
- stb.

A komponens és komponens modell fogalma

Komponens

Szyperski féle szoftver komponens definíció:

- „A szoftver komponens
  - egy kompozíciós egység, aminek
  - szerződéssel definiált interfészei
  - és csak explicit környezeti függőségei vannak.
- A szoftver komponenst
  - függetlenül lehet telepíteni és
  - független fél használhatja fel kompozícióban."

“szerződéssel definiált interfészei”:

- interfész: a komponens kapcsolódási felületeinek specifikációja (szolgáltatott és elvárt)
- szerződés: a szolgáltatót és a klienst kölcsönösen megkötő feltételek
  - funkcionális és nem funkcionális aspektusok
  - műveletek elő- és utófeltétele

“explicit környezeti függőségei vannak”:

- környezeti függőségek: a telepítési és futási környezet specifikációja
  - pl. milyen eszközökre, platformokra, erőforrásokra vagy más komponensekre van szükség

“függetlenül lehet telepíteni és független fél használhatja fel kompozícióban”:

- komponens platformra telepíthető és utána az interfészein keresztül elérhető
- a felhasználó különbözhet a fejlesztőtől és telepítőtől

Komponens modell

A komponens modell azon szabványok és konvenciók specifikációja, amelyek szükségesek ahhoz, hogy az egymástól függetlenül fejlesztett komponensek kompozícióját elő lehessen állítani. Ilyenek például:

- a kompozíciós mechanizmusok
- a sikeres együttműködéshez a komponensek által betartandó konvenciók

Komponens technológiának nevezzük a komponens modell megvalósítását, szabványok és szoftver eszközök biztosításával a komponensek megvalósításához, összeállításához és működtetéséhez. Példák:

- .Net,
- Corba Component Model,
- Enterprise Java Bean, ...

UML kompozíciós diagram fogalma

Az UML kompozíciós (komponens) diagrammal a rendszer komponenseit, azok kapcsolatait, valamint nyújtott és elvárt interfészeiket ábrázolhatjuk.

A kompozíciós diagram részei:

- Structured Classifier: Olyan (absztakt) osztályt jelöl, melynek van belső struktúrája és működését részben vagy teljesen a Part-ok írják le.
- Part, Role: valamilyen általánosított szereplő, amely csak magát a szerepet (funkcionalitást) jelöli ki, de lehet konkrét osztálypéldány vagy valamilyen absztrakt superclass is akár.
- Port: A komponensek közötti kommunikáció csomópontjai. Definiálják saját required és provided interface-eiket. A portokat connector-okkal összekötve lehet kifejezni a komponensek közötti kommunikációt.
- Provided interface: Azok az interfacek, melyeken keresztül a komponens szolgáltatásokat nyújt. (Mint a HiFi-n a Jack dugalj)
- Required interface: Azok az interfacek, melyek a komponens működéséhez szükségesek, ezekt használja. (Mint egy HiFi-nek az áramforrás csatlakozó)
- Connector
  - Delegation connector: A bennfoglaló kompozit külvilág számára látható portjait köti össze a benne lévő komponensekkel.
  - Assembly connector: Ugyanazon strukturális szinten lévő komponensek portjait köti össze.

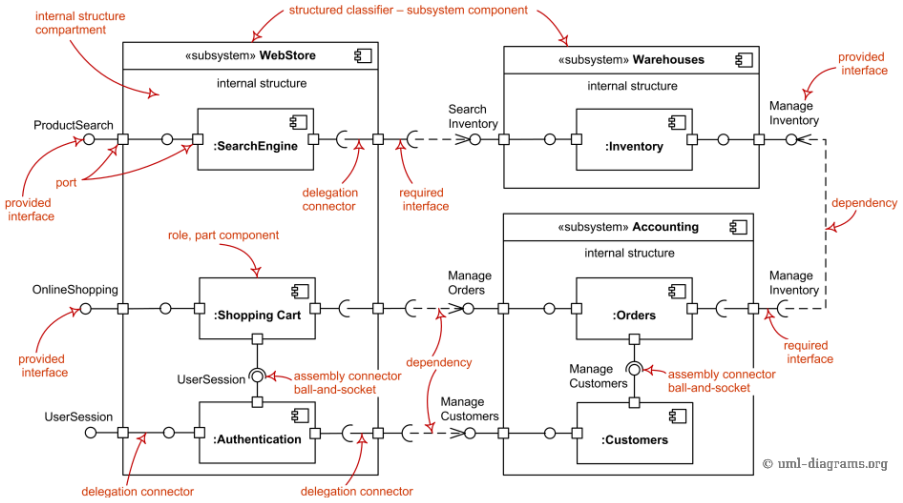

Figure 1: Kompozíciós diagram

A szoftverarchitektúrák fogalma, összetevői

Szoftver architektúrának nevezzük a szoftver fejlesztése során meghozott elsődleges tervezési döntések halmazát.

- Olyan döntések, amelyek megváltoztatása később a szoftver jelentős újratervezését igényelné
- Kihatnak a rendszer
  - felépítésére
  - viselkedésére
  - kommunikációjára
  - nem funkcionális jellemzőire
  - megvalósítására

A szoftver architektúra elsődleges feladata a rendszer magas szintű felépítésének és működésének meghatározása, a komponensek és kapcsolataik kiépítése.

Fontosabb strukturális összetevői:

- Architektúra sémák
  - elemek és reláció típusok leírása, amely a használatukra vonatkozó megszorításokat is tartalmazza
  - pl.: a kliens szerver architektúra egy jól ismert séma
- Referencia modellek
  - egy referencia modell a funkcionalitás egy olyan felosztása, amely az egyes részek közötti adatfolyamot is tartalmazza
  - a referencia modell egy ismert probléma felbontása olyan részekre, amelyek egymással kooperálva megoldják az eredeti problémát
- A referencia architektúra
  - a referencia architektúra egy olyan referencia modell, amelyet a szoftverelemekre és azok kapcsolataira képeztünk le

Szoftverarchitektúrák például:

- monolitikus architektúra
- réteges architektúrák (pl. Model-View-Persistence)
- mikroszolgáltatások architektúra (microservices)

A KobrA programfejlesztési modell alapjai

Komponensalapú fejlesztés általánosan

- Meglévő szoftverkomponensekből építjük fel a rendszerünket
- A komponensek kellően általánosak $\Rightarrow$ újrafelhasználhatóak
- A komponens alapú fejlesztés alapvetően két diszjunkt területre bomlik:
  - a komponens fejlesztés
    - a komponensek, mint egyedi építőkövek kifejlesztése
  - az alkalmazás fejlesztés

    * a kész komponensekből a rendszer felépítése

KobrA programfejlesztési modell

A KobrA-ban legfontosabb célunk, hogy a rendszer alapvető struktúráját absztrakt formában állítsuk elő, hogy sokféle implementációs technológiára leképezhető legyen.

Legfontosabb elvek:

- Problémák elkülönítése
- Modellalapú fejlesztés
- Komponensek használata
- Objektumorientált technológiák alkalmazása

A KobrA a fejlesztési folyamatot két alapvető dimenzióra osztja

- Kompozíció/dekompozíció:
  - Dekompozíció: a rendszer alkalmas szétvágása önmagukban kezelhető részekre
  - Kompozíció: az implementált részek összeillesztése
- Absztrakció/konkretizáció:
  - A felhasználók számára jól érthető modelltől közelítünk a számítógép által értelmezhető, futtatható program felé

Megtestesítés: a specifikált szolgáltatások tényleges megvalósításának folyamata

Érvényesítés (validáció): az implementált elemek összevetése az absztrakt modellel

A KobrA modell környezeti térképe

- A környezetet definiáló leíróelemek összessége
- A környezet önálló, saját jogú komponens

Vállalati vagy üzleti modell

- A rendszerünk üzleti környezetét írja le
- Banki rendszerek esetében például a vállalati modell reprezentálja azokat a fogalmakat, amelyek a banki világ számára relevánsak
- Azt kell megadni, hogy ezek a fogalmak milyen hatással lesznek az elkészülő szoftverrendszerre

Használati modell

- A felhasználónak a rendszerrel fenntartott magas szintű kapcsolatait specifikálja
- Használati eset diagramok összességéből áll elő
  - Bemutatják a rendszer szereplőit és a használati eseteket, valamint a közöttük lévő kapcsolatokat
  - Minden használati eset valamilyen absztrakt tevékenységet reprezentál, amelyet a rendszer felhasználója hajt végre
- A használati esetek főleg a rendszer összetevői közötti interakciókra, valamint a rendszer és határai közötti kapcsolatokra koncentrálnak

Strukturális modell

- A környezeti térképen belüli strukturális modell definiálja azon entitások struktúráját, amelyek kívül esnek a kifejlesztendő rendszer hatáskörén, de hatással vannak a rendszerre
- Például:
  - input
  - output, amelyet valamilyen módon tovább feldolgoznak

Viselkedési modell

- A rendszer szereplőinek a rendszer határára vonatkozó tevékenységeit írja le
- Az UML aktivitás diagramjával történik

Komponens specifikáció részei

- A komponens specifikáció azon leíró dokumentumok összessége, amely definiálja a komponens működését
- Minden ilyen dokumentum a komponens működésének valamilyen aspektusát vizsgálja és csak arra koncentrál, hogy az adott aspektusból a komponens mit csinál
- Cél: az elvárt és szolgáltatott interfészek leírása
- A komponens specifikáció mindent tartalmaz, amit tudni lehet és kell a komponensről

Funkcionális modell

- A komponens szolgáltatott és az elvárt műveleteit írja le
- Definiálja a szolgáltatott műveletek kívülről látható hatásait

Viselkedési modell

- Megmutatja, hogyan viselkedik külső hatásokra a komponens
- UML állapotdiagramok formájában készül el
- Ha egy komponensnek nincsenek állapotai (tisztán funkcionális komponens), akkor nincs szükség viselkedési modellre

Strukturális modell

- Definiálja a komponens műveleteit, attribútumait és azt, hogy milyen komponensekkel van kapcsolatban
- Rávilágít a jó dekompozíciós lehetőségekre