Loeng 12. Täiendavad teemad. Kokkuvõte.

- Arhitektuuri esitamine UMLi vahenditega
  - Paketid, n-kihilise arhitektuuri esitamine
  - Realisatsioonidiagrammid: komponendi- ja rakendusdiagrammid (näited)
- UMLi metamudeli lühiülevaade
- Mustri mõiste ja näiteid

- Aine kokkuvõte (eesmärgid ja tulemused)
- Jätkuaine?

Viimase loengu eesmärgiks on puudutada mõningaid olulisi teemasid, mida siiani pole jõutud käsitleda.

1. **UMLi kasutamine** süsteemide loogilise ja füüsilise **arhitektuuri esitamiseks**

1.1. Kihiliste arhitektuuride esitamiseks sobib **UMLi paketi mõiste**. Pakett on mudeli elementide loogilise grupeerimise mehhanism. Iga arhitektuurikihi võib esitada paketina, mille stereotüübiks on <<layer>>. Iga kihipaketi sees saab esitada selle kihi komponente ehk allsüsteeme. Allsüsteemi võib esitada paketina, mille stereotüübiks on <<subsystem>>. Allsüsteem on loogiline mudel-komponent, mis pole veel seotud ühegi konkreetse realisatsioonikeskkonnaga. Allsüsteemist võib edasise arenduse käigus saada tarkvarakomponent. Komponentide füüsilises disainis saab kasutada UMLi realisatsioonidiagramme.

1.2. Realisatsioonidiagrammideks on UMLi **komponent- ja rakendusdiagrammid** (deployment dgm). Komponentdiagramm on programmeerija töövahend. Esitab tarkvarakomponente, nende liideseid ja sõltuvusi. Komponent on konkreetse programmeerimiskeele või muu realisatsioonikeskkonnaga seotud allsüsteem. Komponentdiagramm on klassidiagrammi erijuhtum. Komponent on klass stereotüübiga <<component>>. Samuti rakendusdiagramm on objekti- või klassidiagrammi erijuhtumiks, mis võimaldab esitada võrguarhitektuure. Võrgu sõlm on objekt klassist stereotüübiga <<node>>. Sõlmede ühendus on link assotsiatsioonis stereotüübiga <<connector>>. Võrgu sõlmedesse joonistatakse komponente ja nendevahelisi sõltuvusi. Saab defineerida sõlmede tüüpe ja tüüpkoosseise, kasutades üldistuse ja pärimise seoseid võrgu sõlmede jaoks.

1.3. Arhitekt ja disainer peavad tundma ning oskama kasutada **disainimustreid**. Disainimuster on üldistatud probleem-lahendus paar (üldistatud lahendus üldistatud probleemile), mida võib esitada vaba teksti ja / või näiteks UMLi diagrammide vormis. UMLi terminoloogias muster on tõlgendatav kui parametriseeritav kollaboratsioon ehk koostööeeskiri mustri elementide ehk klasside vahel. Muster esitatakse sel juhul kollaboratsioonisümboli (katkendovaal) abil, millel on sõltuvusseosed (katkendnool) mustri elemente esitavatesse klassidesse (ristküliku sümbol). Parameetrite edastamine mustris osalevatele klassidele toimub sõltuvusseostele kirjutatavate rollinimede kaudu. Nende rollinimede kaudu seostatakse osalevad klassid mustrit täpsustavates klassi- ning interaktsioonidiagrammides. Mustri iga elemendi võib omakorda modelleerida mustrina (composite pattern – liitmuster). Suured liitmustrid on süsteemiarhitekti tööriist. Näiteks valib arhitekt esmalt kihilise mustri, mille igasse kihti valib omakorda sobiva alammustri jne.. Mustrite tundmine ja rakendamise lihtsustab oluliselt OO disaini ühe võtmeküsimuse – vastutuste jagamine ülesande täitmises osalevate klasside vahel – kiiret ja otstarbekat lahendamist taaskasutatavate komponentide baasil.

2. **UMLi metamudel.** UML on isekirjeldav keel, mille tuum on kirjeldatud UMLi enda klassidiagrammi vahenditega kuues nn. tuumpaketis (UML Core Packages). Iga selline pakett esitatakse ühe klassidiagrammi (UMLi metaklassid ja nende seosed) ning pikkade tekstiliste selgituste ja definitsioonide kaudu. UMLi tuumas antakse laiendusmehhanismid, mille abil on võimalik tuuma kuuluvaid mõisteid laiendada ning kohandada konkreetse probleemvaldkonna tarbeks (näiteks probleemvaldkonna konkreetsed klassid defineeritakse tuumast pärit metaklasside (nagu klass kui niisugune, seos kui niisugune, jne..) alusel), kasutades sobivaid stereotüüpe. Modelleerimise ja vahendite arenedes muutub ka UMLi tuum (varsti on oodata versiooni 2.0).

3. **OOM aine kokkuvõte.** Aine eesmärgiks oli 1) omandada UMLi diagrammitehnikate kasutamise põhioskused ning teadmised 2) seostada need oskused ja teadmised ühe konkreetse OOA / OOD iteratiivse protsessiga 3) kasutada saadud teadmisi ja oskusi aine projekti läbiviimisel. Loengutes jõudsime käsitleda iteratiivse arendusprotsessi põhilisi samme ja nende seoseid. Samuti UMLi põhiliste diagrammitehnikate mõisteid ja seoseid. Praktika poole pealt peaksid

aine läbinud tudengil olema teadmised ja oskused OO analüüsi korrektseks läbiviimiseks UMLi vahenditega. OO disaini jõudsime ainult “nuusutada”. See on suur valdkond, mis nõuab jätkuaineid ning iseseisvat õppimist, eriti disainimustrite rakendamise osas.

4. Jätkuained?: Ametlikku OO disaini nimelist jätkuainet, mis seob OO analüüsi sujuvalt OO programmeerimisega, kahjuks ei tea olevat. Annan võimaluse soovijatel sellega tegelda sügissemestri aines “Infosüsteemide arendamise tehnoloogia”