Fakulteti i Shkencës Kompjuterike

Inxhinieria e Softverit

UML Component Diagramet duke përdorur Microsoft Office Visio 2013

MSc Fatime Gashi

UML Diagramet e Komponenteve

UML Diagramet e Komponenteve i tregojnë varshmëritë ndër komponentet softverike, përfshirë klasifikuesit që i specifikojnë ato, si klasat implementuese, dhe artefaktet që i implementojnë ato, si p/sh. File-i source-code, file-i binarycode, file-i ekzekutues, skriptat dhe tabelat. Diagramet e komponenteve krijohen për:

- Modelimin e konfigurimit të dizajnit të nivelit të ulët për sistem;
- Modelimin e infrastrukturës teknike;
- Modelimin e arkitekturës biznesore për organizatë;

Për komponentet

Në figurën 1 komponentet janë modeluar me katërkëndësha, dhe duke e theksuar llojin e komponentës *<<komponenta>>* apo duke e përdorur simbolin . Komponentet mund të realizojnë një apo më shumë interfejsa, të cilat paraqiten në formë të lëpisës, dhe mund të ketë varësi nga komponentet apo interfejsat tjerë.

Siç shihet nga figura, komponenta *Persistence* ka varësi nga komponenta *DB e Korporatës*.

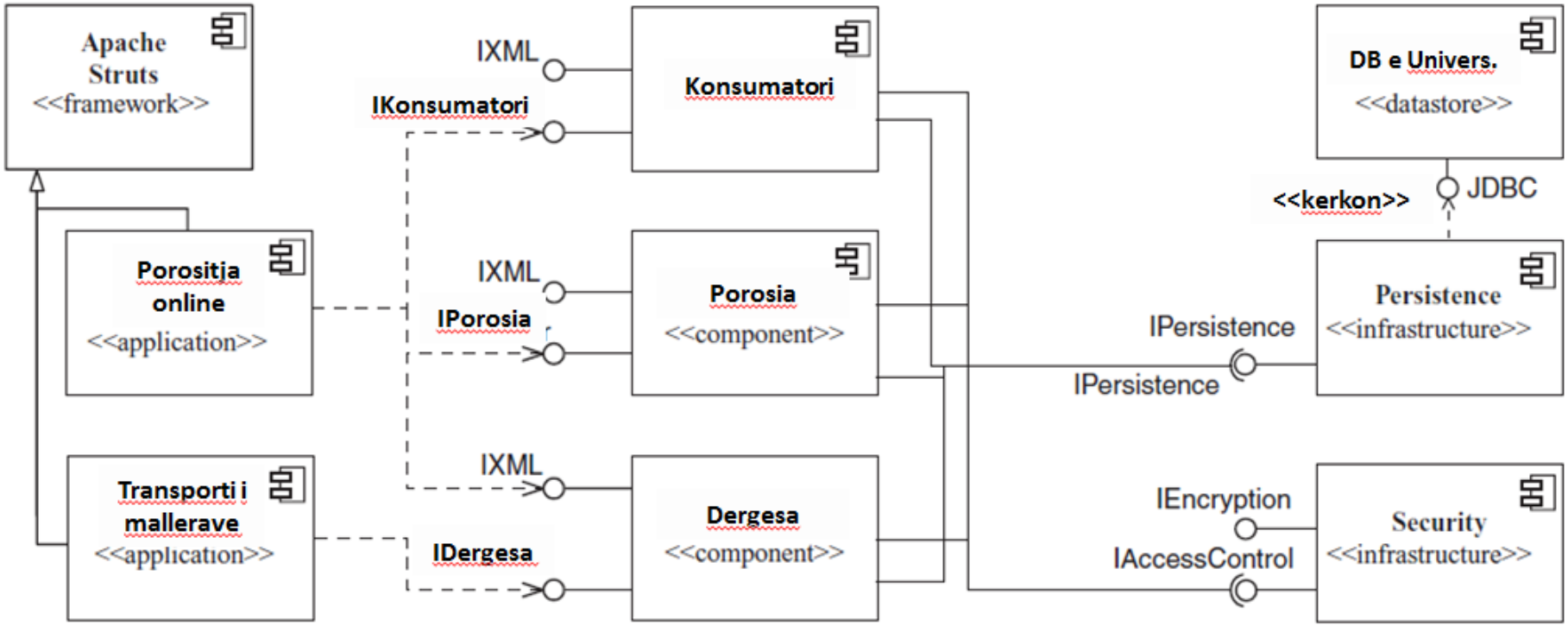

Figura 1: UML Diagrami i Komponentave që paraqet arkitekturën logjike të një sistemi të thjeshtë e-komerc

Diagramet arkitekturale shpesh vështrohen nga një gamë e gjërë e njerëzve, të cilët nuk mund të jenë të njoftuar me projektin tuaj, prandaj emrat e komponenteve duhet të jenë të kuptueshëm. Për shembull shumica e komponenteve me përjashtim të *DB të Korporatës* emërohen me emra, si *Konsumatori* dhe *Persistence*.

Kur krijoni model të detalizuar të komponenteve, për ta kuptuar konfigurimin fizik të sistemit, atëherë emëroni komponentet e juaja duke përdorur emra që lidhen me rrethanat p.sh. file-i *Java source-code* mund të emërohet *Customer.java*, ose biblioteka e Windows-it mund të emërohet *auditLogger.dll* kurse dokumenti mund të emërohet *User Manual.doc*.

Gjatë diagramit të komponenteve përdoreni një mënyrë të shënimit në mënyrë konsistente ose *<<komponentën>>* ose simbolin  .

Në vijim janë dhënë kuptimet e disa fjalëve që përdoren më shpesh:

Application – Është pjesa frontale e sistemit tuaj, si psh mbledhja e faqeve HTML dhe e ASP/JSP që punojnë me to, për sistemin që bazohet në kërkim.

Datastore (deponimi i të dhënave) – lokacion persistent për deponimin e të dhënave.

Document – dokumenti i printuar apo elektronik.

Executable – komponenta softverike e cila mund të ekzekutohet në node.

File – file-i i të dhënave.

Infractructure – komponentë teknike përbrenda sistemit tuaj, si shërbimi persistencë.

Source code – file-i source-code, si file-i *.Java apo file-i *.cpp.

Table – tabelë e të dhënave përbrenda databazës.

Web service – një apo më shumë shërbime të Web-it.

XML DTD – XML DTD.

Paraqitni vetëm interfejsat relevante

Është mirë që diagramet t'i mbani sa më thjeshtë që të jetë e mundur. Një mënyrë për ta bërë këtë është të paraqiten vetëm interfejsat që janë të aplikueshme për qëllimet e diagramit tuaj. Për shembull interfejsi XML është i modeluar për komponentën *Porosia* por nuk është duke u përdorur, që dmth se ju mund të mos e paraqitni këtë herë. Pra mos e ngarkoni diagramin me informata të panevojshme.

Lidhjet e varshmërisë dhe trashëgimisë (Dependency dhe Inheritance)

Komponentet do të kenë varësi ose nga komponentet tjera ose nga interfejsat e komponenteve tjera. Varshmëritë paraqiten me shigjeta me vija të ndërprera.

Duhet të përpiqeni që t'i rradhitni komponentet e juaja në atë mënyrë që të mund t'i vizatoni varësitë nga e majta në të djathtë. Dhe kjo do ta rrisë konsistencën e diagrameve tuaja, dhe ju ndihmon që t'i identifikoni varësitë qarkore (unazore) në dizajnin tuaj, siç është paraqitur në shembullin në figurën 2:

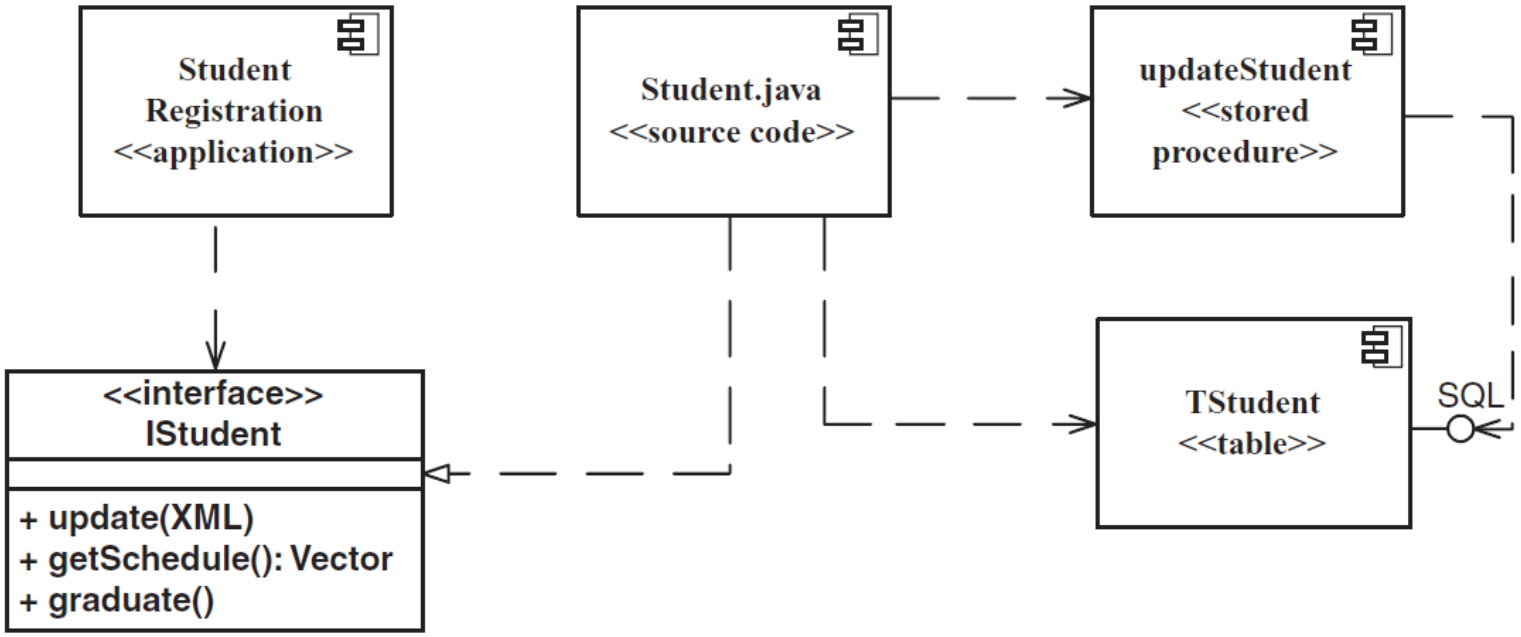

Figura 2: UML Diagrami i Komponenteve për disa aspekte të studentit të sistemit të universitetit

Varësia rrethore është në vijim: *Student.Java* varet nga *updateStudent*,e cila varet nga *TStudent*, e që varet nga *Student.Java*.

E në rastin e modelimit vertikalisht, atëherë mirë është që të modelohen varësitë nga lartë poshtë.

Trashëgimia/Inheritance

Trashëgimia ndërmjet komponenteve mund të ekzistojë në këtë rast ndërmjet *Shipping* dhe *Appache Strouts,* dhe siç shihet komponenta trashëguese paraqitet nën komponentën amë.

Bëni komponentet të varura vetëm nga interfejsat

Duke i bërë komponentet e varura vetëm nga interfejsat, kjo e bën të mundur që të zëvendësohet vetëm komponenta e jo edhe komponentet që varen prej saj.

Shmanguni nga modelimi i varësive kompiluese

Nëse e vëreni se keni modeluar varësi të kompilimit, atëherë ka gjasa që e keni tej-modeluar sistemin tuaj.

Në vazhdim është paraqitur një shembull i diagramit të komponeneteve, në figurën 3:

Ky shembull paraqet diagramin e komponeneteve për blerjen online. Diagrami paraqet tri nënsisteme: WebStore, Warehouses, and Accounting.

Nënsistemi **WebStore** i përmban tri komponente që kanë të bëjnë me blerjen online – **Search Engine**, **Karroca për blerje** dhe **Autentikimi**. Komponenta **Search Engine** e mundëson kërkimin e artikujve nëpërmjet interfejsit **Kerkimiiproduktit** dhe e përdor interfejsin **Kërko Inventarin** të cilën e ka komponenta **Inventari**.

Komponenta **Karroca për blerje** e përdor interfejsin **Menaxho porositë** të cilën e ka komponenta **Porositë**. Komponenta **Autentikimi** ua mundëson klientëve të krijojnë llogari, të kyqen dhe çkyqen nga ndonjë llogari.

Nënsistemi **Accounting** i ka dy interfejsa – **Menaxho Porositë** dhe **Menaxho Konsumatorët.** Dhe nëpërmes këtyre dy interfejsave, pjesët tjera të diagramit lidhen me komponentet **Porosia** dhe **Konsumatorët**.

Nënsistemi **Warehouses** i siguron dy interfejsa **Kërko Inventarin** dhe **Menaxho Inventarin** të cilat përdoren nga nënsistemet tjera nëpërmes lidhjeve dependency/varshmëri.

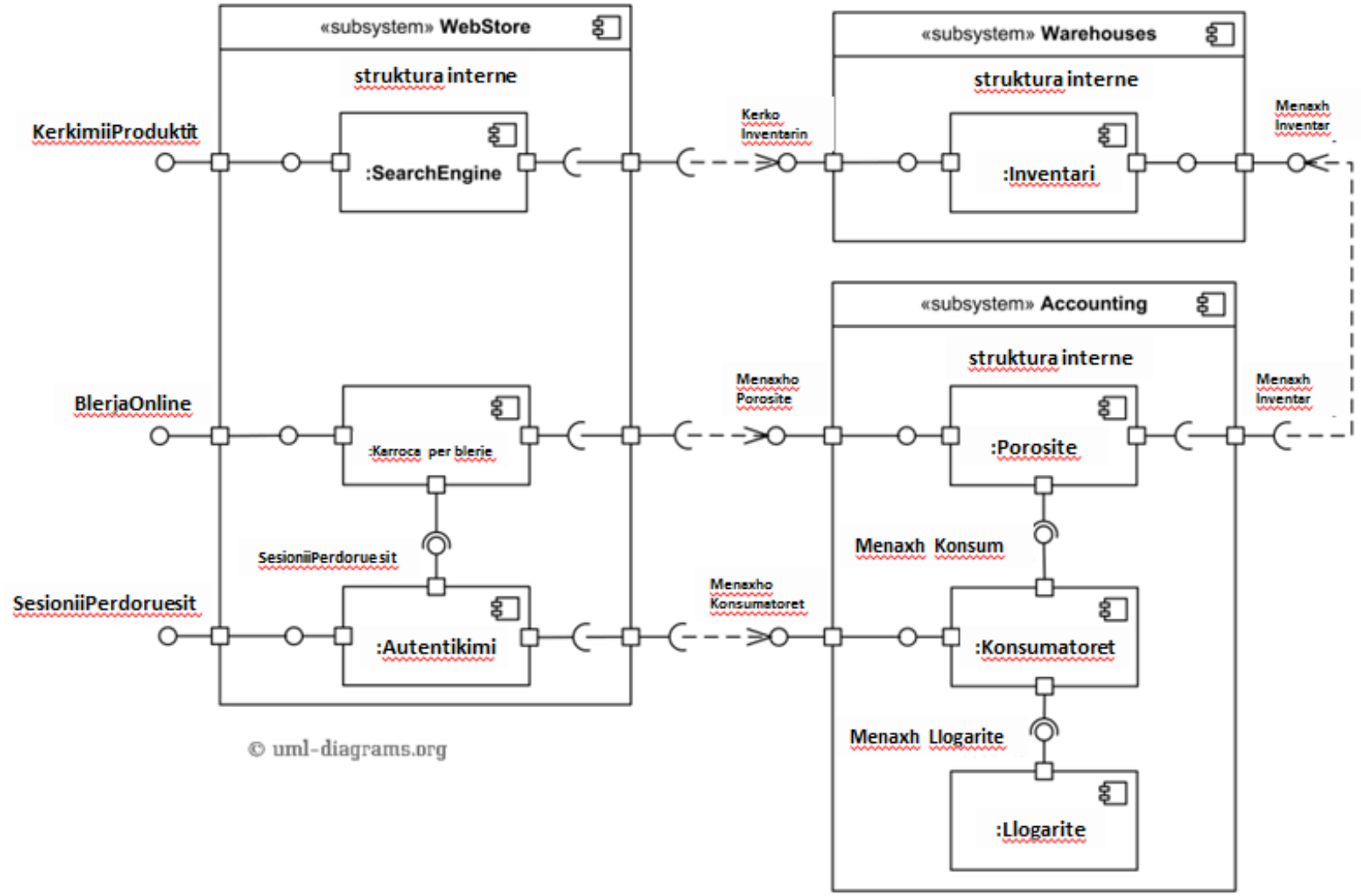

Figura 3: Diagrami i Komponenteve për Blerjen Online