W

CURS 1

1) Ce diagramă UML se folosește pentru modelarea interacțiilor dintre obiectele sistemului software?

Raspuns: Se folosește diagrama de secvențe, care arată fluxul de mesaje între obiecte în funcție de timp.

2) Ce diagramă UML se folosește pentru modelarea comportamentului generat de evenimente interne și externe al sistemului software?

Raspuns: Se folosește diagrama de stări (State Machine Diagram), care reprezintă stările active, tranzițiile dintre ele și evenimentele care declanșează schimbările de stare.

3) Pe ce nivele de detaliu se realizează modelarea comportamentului?

Raspuns: Modelarea comportamentului se face pe două nivele de detaliu:

Modelarea comportamentului sistemului în contextul său (diagrame de secvențe la nivel de sistem, diagrame de stări și tranziții la nivelul sistemului).

Modelarea comportamentului intern al sistemului (identificarea obiectelor implicate, secvențe de interacțiuni între obiecte interne, diagrame de stări și tranziții pentru obiectele relevante).

4) Ce conține un obiect de date ?

Raspuns: Un obiect de date conține **un set de atribute** care definesc proprietățile sale.

5) Un obiect de date este definit printr-un set de _____.

Raspuns: atribute.

6) conexiune relevantă între obiecte de date este definită printr-o _____.

Raspuns: relatie

7) Ce se reprezintă pe diagrama de activitate? Indicați variante de utilizare a acesteia?

Raspuns: Diagrama de activitate reprezintă fluxul de acțiuni și activități din cadrul unui proces.

Variante de utilizare:

Modelarea fluxului logic al unui proces.

Identificarea cazurilor de utilizare.

Reprezentarea proceselor business (prin diagrame Swimlane).

8) Ce se reprezintă pe diagrama cazurilor de utilizare?

Raspuns: Se reprezintă **interacțiunile actorilor externi cu sistemul**, ilustrând totalitatea modurilor utile de utilizare ale acestuia.

9) Ce se reprezintă pe diagrama de secvențe la nivel de sistem?

Raspuns: Se reprezintă **interacțiunea dintre actor și sistem**, la nivelul interfeței acestuia.

10) Ce conține prototipul interfeței grafice cu utilizatorul ?

Raspuns:

Ecranul de pornire.

Ecranele și fluxul acestora pentru fiecare caz de utilizare.

Se recomandă folosirea diagramei de stări și tranziții pentru reprezentarea fluxului ecranelor.

11) Deoarece în realitate desfășurarea activităților din setul generic se poate suprapune, propuneți o ordine de lansare a acestora.

Raspuns Ordinea recomandată a activităților:

Revizuirea cerințelor.

Expandarea și rafinarea scenariilor.

Modelarea informațiilor.

Modelarea funcțiilor.

Modelarea comportamentelor.

Analiza și modelarea interfeței utilizator.

Revizuirea completitudinii și consistenței modelelor.

CURS 2

12) Care este regula de comunicare între nivelele unei arhitecturi multinivel?

Raspuns:

Interacțiunea are loc doar între nivele adiacente, un nivel superior accesând serviciile oferite de nivelul inferior. Dacă se schimbă interfața unui nivel, doar nivelele adiacente sunt afectate.

13) Enumerați criterii de separare a elementelor pe diferite nivele ale unei arhitecturi multinivel.

Raspuns:

Nivel de abstractizare – gruparea elementelor cu același nivel de abstractizare. Separarea problemelor – gruparea elementelor corelate și separarea celor disparate. Flexibilitate – menținerea unei cuplări slabe între nivele și încapsularea schimbărilor. Domeniul de aplicabilitate – separarea elementelor aplicației de cele ale modelului domeniului.

14) Cum se poate reprezenta pe modelul UML un nivel arhitectural ?

Raspuns:

Nivelele arhitecturale pot fi modelate în UML utilizând pachete stereotipate cu <<layer>>.

15) Identificați abstractizările cheie, atribute ale lor și relații dintre ele, pe baze următoarei descrieri a unei aplicații software simplă:

Aplicaţie simplă pentru o bibliotecă:

Abonatul poate face o rezervare on-line indicând un domeniu de interes. Sistemul afișează o listă de cărți din domeniul respectiv pentru care există exemplare disponibile, indicând numele și autorii. Sistemul face rezervarea și returnează abonatului un număr de rezervare valabil pentru ziua curentă. Ajuns la bibliotecă abonatul indică bibliotecarului codul rezervării, pe care bibliotecarul îl introduce în aplicaţie. Sistemul anulează rezervarea și permite înregistrarea împrumutului în fișa abonatului indicând datele fiecărei cărți împrumutată: titlul cărții, autorul și numărul de inventar al exemplarului de carte împrumutată, precum și termenul de returnare. Când abonatul returnează una sau mai multe cărți, bibliotecarul utilizează aplicaţia pentru a înregistra această acţiune în fișa abonatului. Dacă abonatul întârzie returnarea unei cărţii, sistemul declanșează o procedură de calcul și înregistrare, pe numele abonatului, a unei penalizări și a unei interdicții de împrumut pentru următoarea lună.

Raspuns:

Abstractizări cheie (Clase principale):

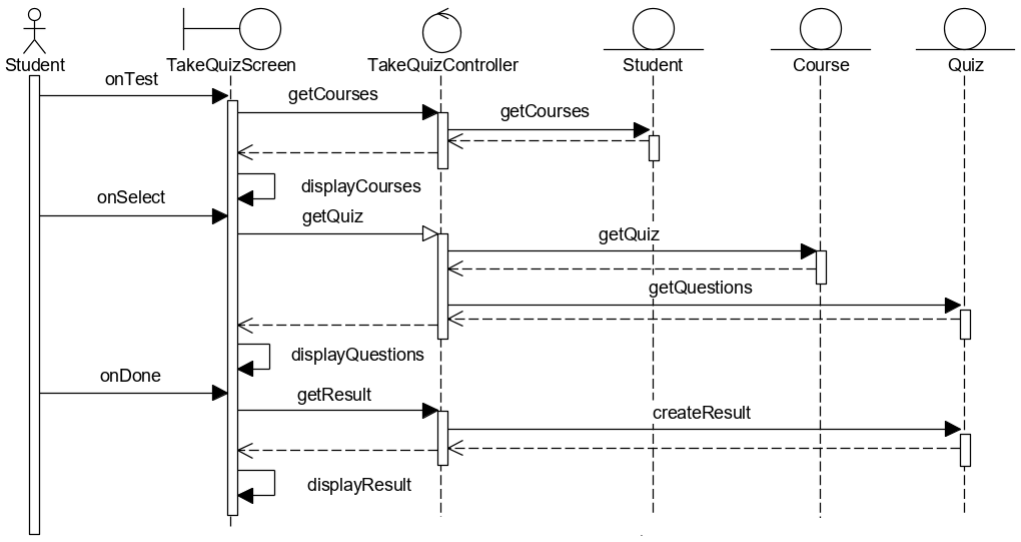
- Abonat (id, nume, email)
- Rezervare (număr rezervare, dată)
- Carte (titlu, autor, număr inventar)
- Bibliotecar (id, nume)
- Împrumut (abonat, carte, dată împrumut, termen returnare)

Relații:

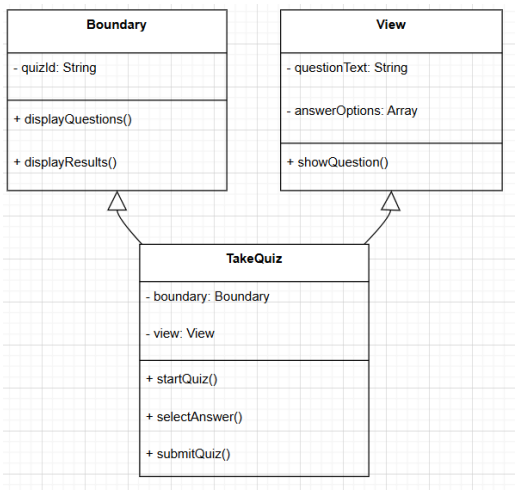
- Abonat face o Rezervare.

- Bibliotecar procesează Împrumuturile.
- Carte poate avea mai multe Împrumuturi.
- Dacă termenul de returnare este depășit, se creează o **penalizare**.

16) Considerând diagrama de secvențe din figură, desenați diagrama claselor participante (VOPC).



Raspuns:



CURS 3

17) Enumerați etapele procesului de proiectare software.

Raspuns:

Reprezentarea arhitecturii sistemului. Definirea interfețelor cu utilizatorii finali, cu alte sisteme și cu dispozitive externe. Definirea componentelor constitutive și a relațiilor dintre acestea. Proiectarea componentelor constitutive.

18) Ce se urmărește la evaluarea modelelor ?

Raspuns:

Detectarea erorilor, inconsistențelor și omisiunilor. Compararea variantelor multiple și alegerea celei optime. Fezabilitatea implementării în cadrul unor restricții de timp și cost.

19) În procesul de proiectare software, ce rezultate are separarea problemelor ?

Raspuns:

Creșterea independenței între componente. Legături puternice intra-componentă și legături slabe inter-componente. Izolare relativă a componentelor, ceea ce facilitează mentenanța și reutilizarea.

20) Care sunt elementele de care are nevoie orice metodologie pentru proiectarea software-lui ?

Raspuns:

Notație: limbaj pentru exprimarea fiecărui model. Proces: activități ce conduc la construirea ordonată a modelelor sistemului. Instrumente: artefacte utilizate pentru construirea modelului, care ajută la detectarea erorilor și impun reguli de modelare.

21) Enumerați câteva recomandări pentru creșterea calității proiectării.

Raspuns:

Utilizarea unor șabloane sau stiluri arhitecturale. Crearea unor componente bine proiectate și independente funcțional. Definirea unor interfețe clare, pentru a reduce complexitatea conexiunilor. Adoptarea unei metode iterative, bazată pe analiza cerințelor. Utilizarea unei notații clare pentru reprezentarea modelului de proiect.

22) Conform definițiilor prezentate, ce au în comun abstractizarea procedurală și abstractizarea datelor ?

Raspuns:

Ambele separă proprietățile logice de detaliile implementării. Abstractizarea procedurală separă logica unei acțiuni de detaliile sale de implementare. Abstractizarea datelor separă logica datelor de detaliile de reprezentare.

23) Descrieți pe scurt principiul încapsulării.

Raspuns:

Încapsularea ascunde detaliile implementării și structurile de date locale în spatele unei interfețe, permițând modulelor să fie independente și să comunice minimal între ele.

24) Care sunt criteriile calitative de evaluare a independenței funcționale și care este relația independenței funcționale cu fiecare dintre acestea ?

Raspuns:

Coeziunea: măsoară cât de bine sunt legate între ele funcțiile unui modul. O coeziune mare indică o funcționalitate clară și bine definită. Cuplarea: măsoară gradul de dependență între module. O cuplare mică indică o mai mare independență funcțională. Independența funcțională este direct proporțională cu coeziunea și invers proporțională cu cuplarea.

25) Dați exemple de tipuri de clase de proiectare care nu sunt rafinări de clase de analiză.

Raspuns:

Clase suport pentru persistență: gestionează accesul la date persistente. Clase de sistem: oferă funcții de management și control pentru operarea sistemului. Clase de proces: gestionează abstractizări de nivel inferior necesare procesării informațiilor.

CURS 4

26) Conform structurii canonice a sistemelor software, enumerați modelele primare ale unui sistem software. Explicați încuibarea recursivă caracteristică unuia dintre aceste modele.

Răspuns: Modelele primare ale unui sistem software sunt:

- Modelul domeniului (modelul de analiză): reprezintă adevărurile durabile despre domeniu, independente de implementare.

- Modelul proiect: descrie sistemul ce va fi construit, incluzând decizii de proiectare, mecanisme, interfețe.
- Modelul codului: descrie codul sursă al sistemului la nivel de limbaj de programare.

Modelul proiect prezintă o încuibare recursivă, având o structură arborescentă de modele:

- Modele de interfață publică vizibilă (boundary), care ascund detaliile interne.
- Modele interne (internals), care rafinează interfețele definite în modelele de interfață.

27) Care sunt principalele categorii de interfețe ale unui sistem software ?

Răspuns: Principalele categorii de interfețe sunt:

- Interfața cu utilizatorul (UI): include elemente de estetică, ergonomie și componente tehnice reutilizabile.
- Interfețele cu sisteme externe: necesară pentru schimbul de date cu alte sisteme, rețele, dispozitive.
- Interfețele interne: specifice comunicării dintre componentele sistemului, definite prin operațiile vizibile din exterior.

28) Ce trebuie să conțină o descriere completă a detaliilor interne ale unei componente software și ce diagrame UML se pot folosi ?

Răspuns: O descriere completă trebuie să conțină:

- Structurile de date necesare fiecărei componente.
- Detaliile algoritmice pentru procesările interne.
- Interfețele de acces la operațiile componentei.

Diagrame UML utilizate:

- Diagramă de clase pentru structura internă.
- Diagramă de activitate sau pseudocod pentru logica procesării.
- Diagramă de componente pentru interfețe (porturi, required/provided).

29) Ce relație există între clasele de analiză și clasele de proiectare ?

Răspuns:

Clasele de analiză modelează obiectele din domeniul problemei și cerințele funcționale, în timp ce clasele de proiectare includ cerințe extra-funcționale și modelează soluția implementabilă. Unele clase de analiză devin clase de proiectare, altele sunt divizate sau combinate, iar unele pot deveni subsisteme sau pachete.

30) Indicații categorii de clase de analiză candidate să devină subsisteme.

Răspuns:

- Clasele care oferă servicii complexe (ex. autorizare securitate).
- Clasele <> pentru GUI sau acces la sisteme externe.
- Clasele cu comportament opțional sau variante de servicii.

- Elemente puternic cuplate.
- Produse existente care exportă interfețe.

31) Care sunt cele 2 zone de descriere a unui subsistem ? De ce este necesară separarea acestora ?

Răspuns:

- Zona de interfață: definește funcționalitățile publice expuse de subsistem.
- Zona internă: conține detaliile de implementare ale subsistemului.

Separarea este necesară pentru modularitate, interschimbabilitate și posibilitatea de a dezvolta componentele independent.

32) Ce presupune rafinarea claselor de analiză referitor la atribute și operații ?

Răspuns:

- Definirea precisă a numelui, tipului și vizibilității atributelor.
- Adăugarea de operații adiționale precum getters/setters, constructori/destructori.
- Optimizarea accesului la date, cum ar fi utilizarea proxy-urilor pentru atribute rareori folosite.

33) Ce înțelegeți prin clasă container ?

Răspuns:

O clasă container este o clasă utilizată pentru a gestiona colecții de obiecte, precum liste, seturi sau dicționare. Aceasta permite manipularea eficientă a obiectelor de același tip.

34) Sub ce forme se poate proiecta reutilizarea unei părți din implementarea unei clase ?

Răspuns:

- Factorizare: crearea unei superclase care conține funcționalitatea reutilizabilă, iar subclasele moștenesc această funcționalitate.
- Delegare: utilizarea compoziției pentru a delega operațiile către o altă clasă, permițând reutilizarea codului fără moștenire.

CURS 5

NU ERAU INTREBARI

CURS 6

35) Ce relație există între arhitectura sistemului software și atributele de calitate ale acestuia ?

Răspuns:

Arhitectura software este esențială pentru îndeplinirea atributelor de calitate ale unui sistem. Aceasta determină modul în care cerințele de calitate, precum performanța, fiabilitatea, securitatea și mentenabilitatea, sunt implementate și susținute. O arhitectură bine proiectată facilitează atingerea acestor obiective prin utilizarea de stiluri arhitecturale adecvate și prin aplicarea unor principii și mecanisme arhitecturale specifice.

36) Ce este un conflict arhitectural și cum se poate rezolva ? Dați un exemplu.

Răspuns:

Un conflict arhitectural apare atunci când două sau mai multe atribute de calitate intră în opoziție, ceea ce face dificilă optimizarea lor simultană. Soluționarea acestuia implică realizarea unui compromis prin alegerea unei arhitecturi care oferă un echilibru între cerințele contradictorii.

Exemplu: Creșterea performanței prin utilizarea unor componente mari (granularitate grosieră) poate reduce mentenabilitatea sistemului, deoarece modificările devin mai dificile. O soluție ar fi utilizarea unei arhitecturi modulare, unde doar anumite componente critice sunt optimizate pentru performanță, menținând restul sistemului flexibil și ușor de întreținut.

37) Enumerați perspective fundamentale de reprezentare a arhitecturii software.

Răspuns:

- Perspectiva statică: ilustrează structura codului și unitățile de implementare (ex. module, clase, pachete).
- Perspectiva dinamică: descrie structura de execuție a sistemului (ex. componente și conectori, fluxuri de date, interacțiuni runtime).
- Perspectiva alocării: detaliază modul în care software-ul este distribuit pe resursele hardware și software.

38) Realizați corespondența corectă între perspectivă și tipuri de elemente reprezentate ?

Răspuns:

- Perspectiva statică → module, clase, pachete
- Perspectiva dinamică → componente, conectori, fluxuri de date
- Perspectiva alocării → resurse hardware, fișiere executabile, fișiere de date

39) Realizați corespondența corectă între stil și forme de comunicare între componente.

Răspuns:

- Pipe-and-filter → Flux de date secvențial
- Client-server → Mesaje între client și server
- Event-based → Comunicarea bazată pe evenimente
- Repository → Acces direct la o bază de date centralizată
- Layered → Apeluri de funcții între straturi

40) Realizați corespondența corectă între mecanism și categoria căreia îi aparține.

Răspuns:

- Concurență → Planificator de task-uri, management procese
- Persistență → Baze de date, fișiere de configurare
- Distribuire → Broker, middleware

41) Ce relație există între stiluri arhitecturale și mecanisme arhitecturale

Răspuns:

Stilurile arhitecturale stabilesc organizarea generală a componentelor unui sistem software, în timp ce mecanismele arhitecturale se concentrează pe aspecte specifice ale comportamentului sistemului. Un stil arhitectural poate utiliza unul sau mai multe mecanisme arhitecturale pentru a implementa cerințele funcționale și non-funcționale ale sistemului.

42) Enumerați principalele etape în proiectarea arhitecturii unui sistem software.

Răspuns:

1. Reprezentarea sistemului în context și definirea interfețelor externe.
2. Partiționarea sistemului și definirea interfețelor interne.
3. Proiectarea arhitecturii datelor.
4. Rafinarea arhitecturii în componente.
5. Descrierea instanțierilor arhitecturii.

43) De ce este importantă crearea și evaluarea mai multor variante ale arhitecturii unui sistem software ?

Răspuns:

Evaluarea mai multor variante arhitecturale permite identificarea celei mai potrivite soluții pentru cerințele sistemului. Aceasta ajută la reducerea riscurilor, optimizarea performanței și îmbunătățirea atributelor de calitate, precum securitatea, scalabilitatea și mentenabilitatea. Metode precum ATAM (Architecture Tradeoff Analysis Method) sunt utilizate pentru a compara alternativele și a alege arhitectura optimă.

CURS 7

44) Cum se reprezintă un șablon de proiectare și cum se reprezintă o instanță a sa în UML ?

Răspuns:

Un șablon de proiectare se reprezintă în UML sub forma unei colaborări parametrizate, unde parametrii sunt marcați cu stereotipul <>. Instanța unui șablon se reprezintă prin asocierea concretă a claselor și obiectelor reale cu rolurile definite în șablon.

45) Ce conține documentația realizată de arhitectul software pentru un mecanism de implementare?

Răspuns:

Documentația unui mecanism de implementare include:

- Descrierea scopului și funcționalității mecanismului.
- Diagrame UML relevante, cum ar fi diagrame de clase și secvențe.
- Definiția interfețelor utilizate pentru integrarea cu alte componente.
- Instrucțiuni de utilizare și exemple de cod pentru implementare.
- Eventuale restricții și considerații legate de performanță sau securitate.

46) În ce condiții sunt interschimbabile două subsisteme ?

Răspuns:

Două subsisteme sunt interschimbabile dacă:

- Implementarea lor respectă aceeași interfață publică.
- Furnizează aceleași servicii și respectă contractele de utilizare definite.
- Nu introduc modificări în structura și fluxul logic al sistemului.
- Se comportă conform cerințelor de calitate specificate (performanță, securitate etc.).

47) Ce diagrame UML se folosesc pentru modelarea comportamentului intern al subsistemelor ?

Răspuns:

Pentru modelarea comportamentului intern al subsistemelor se folosesc:

- Diagrame de secvență pentru a reprezenta fluxul de mesaje între componentele subsistemului.
- Diagrame de clase pentru a arăta structura statică a subsistemului.
- Diagrame de activitate pentru a evidenția logica de execuție a operațiunilor interne.
- Diagrame de stare pentru a ilustra schimbările de stare ale obiectelor din subsistem.

CURS 8

48) Dați exemple de două criterii de alocare procese la noduri.

Răspuns:

- **Localizarea datelor:** Procesele sunt plasate pe nodurile care conțin datele necesare pentru a minimiza latența și utilizarea rețelei.
- **Echilibrarea încărcării:** Procesele sunt distribuite pe noduri pentru a evita supraîncărcarea unui singur nod și a asigura utilizarea optimă a resurselor.

49) Care este semnificația stereotipului <> din diagrama de instalare (delopyment) ?

Răspuns:

Stereotipul <<>> este utilizat pentru a desemna tipul unui element UML. De exemplu:

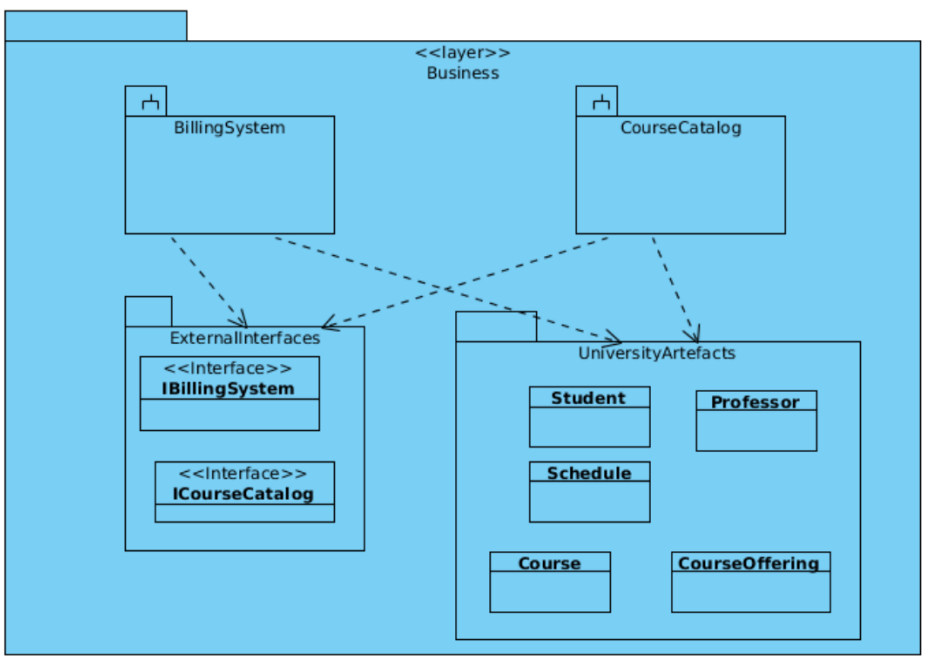
- <<node>> indică un nod fizic.
- <<component>> marchează un component software.
- <<artifact>> indică un artefact implementabil.

50) Dați două exemple de criterii pentru grupare clase în pachete.

Răspuns:

- **Responsabilități comune:** Clasele cu funcționalități similare sunt grupate împreună pentru a îmbunătăți modularitatea.
- **Dependențe minime între pachete:** Se evită interdependențele complexe între pachete pentru a crește flexibilitatea și mentenabilitatea.

Care este punctul slab al acestui model de organizare ? Ce modificări sugeraţi?



Răspuns:

- **Punct slab:** Interfețele dintre subsisteme nu sunt clar definite, iar relațiile dintre clase sunt prezentate doar schematic. De asemenea, nu sunt evidențiate toate dependențele și fluxurile de date.
- **Sugestii:** Introducerea unor componente intermediare pentru separarea logică a responsabilităților și utilizarea unor mecanisme de abstractizare mai bine definite.

51) Realizați corespondența corectă între elemente ale claselor persistente din diagrama de clase și elemente din baza de date relațională.

Răspuns:

- Student → Tabel_Studenti
- Professor → Tabel_Profesori
- Course → Tabel_Cursuri

- CourseOffering → Tabel_Oferte_Cursuri
- Schedule → Tabel_Orar

52) Enumerați strategiile de implementare a persistenței.

Răspuns:

- **Mapare Obiect-Relațională (ORM):** Utilizarea unor framework-uri precum Hibernate pentru conversia automată între obiecte și tabele.
- **Persistență bazată pe fișiere:** Stocarea datelor în fișiere JSON, XML sau CSV.
- **Persistență bazată pe baze de date relaționale:** Utilizarea SGBD-urilor precum MySQL, PostgreSQL.
- **Persistență bazată pe baze de date NoSQL:** MongoDB, Cassandra pentru date nestructurate.

CURS 9

53) Plecând de la informațiile sintetice de mai jos, realizați o analiză mai detaliată referitoare la păstrarea informațiilor sesiunilor în memoria volatilă a procesului server:

Alegere variantă de stocare date sesiuni adaptată la context : • în process (performanță, număr limitat de sesiuni concurente) • utilizare serviciu oferit de server Web (pentru date costisitor de recreat) • utilizare server SQL centralizat (în arhitecturi Web farm) OBS. Protejarea (SSL, IPSec) canalului de comunicare cu serverul utilizat separat pentru stocare sesiuni.

Răspuns:

Păstrarea informațiilor sesiunilor în memoria volatilă a procesului server este o metodă rapidă și eficientă pentru gestionarea sesiunilor, dar vine cu anumite limitări. Principalele aspecte de luat în considerare sunt:

- **Performanță ridicată:** Accesul la datele din memoria RAM este mult mai rapid decât accesul la o bază de date sau un sistem de fișiere.
- **Număr limitat de sesiuni concurente:** Deoarece memoria RAM este o resursă finită, această metodă funcționează bine pentru un număr limitat de utilizatori simultani.
- **Pierderea datelor la restart:** Dacă serverul se oprește sau este repornit, toate sesiunile sunt pierdute.

Alternative adaptate la context:

- **În process (în memoria serverului):** Bun pentru aplicații cu un număr redus de utilizatori.
- **Utilizare serviciu oferit de serverul Web:** Indicat pentru stocarea datelor costisitor de recreat, păstrând echilibrul între performanță și persistență.
- **Utilizare server SQL centralizat:** Potrivit pentru arhitecturi Web farm unde sesiunile trebuie partajate între mai multe servere.

OBS: Se recomandă protejarea canalului de comunicare (SSL, IPSec) atunci când se utilizează un server separat pentru stocarea sesiunilor.

54) Enumerați elementele modelului rezultat al proiectării aplicațiilor web.

Răspuns:

- **Modelul conținutului:** Structura datelor și obiectelor de conținut.
- **Modelul interacțiunilor:** Definirea cazurilor de utilizare, diagrame de secvențe și stări.
- **Modelul funcțional:** Descrierea funcțiilor sistemului și a comportamentului acestora.
- **Modelul configurației:** Stabilirea infrastructurii și a cerințelor de instalare și interoperabilitate.
- **Modelul navigației:** Structurarea fluxului de navigare între componentele aplicației.
- **Modelul arhitecturii:** Organizarea aplicației pe diferite niveluri (MVC, client-server etc.).

55) Enumerați câteva principii de proiectare a interfeței cu utilizatorul în aplicațiile web.

Răspuns:

- **Simplitate și claritate:** Design intuitiv, ușor de utilizat.
- **Consistență:** Utilizarea aceleiași structuri vizuale și funcționale pe toate paginile.
- **Feedback vizual:** Oferirea de confirmări sau notificări privind acțiunile utilizatorului.
- **Navigare intuitivă:** Mecanisme clare de navigare (meniuri, breadcrumb, căutare eficientă).
- **Flexibilitate:** Adaptabilitate la diferite dispozitive și rezoluții de ecran.
- **Eficiență:** Minimizarea numărului de pași necesari pentru realizarea unei sarcini.
- **Securitate:** Protejarea datelor utilizatorului și prevenirea accesului neautorizat.

56) Ce proiectează inginerul software referior la conținutul aplicațiilor web ?

Răspuns:

- **Structura obiectelor de conținut:** Definirea entităților și a relațiilor dintre ele.
- **Schema de organizare a paginilor:** Modul în care conținutul este structurat și prezentat.
- **Mecanisme de gestionare a conținutului:** Crearea și administrarea datelor (ex. CMS - Content Management System).
- **Adaptabilitatea conținutului:** Implementarea variantelor de conținut în funcție de utilizator și context.
- **Optimizarea conținutului:** Asigurarea încărcării rapide și utilizarea eficientă a resurselor.
- **Mecanisme de căutare și filtrare:** Oferirea de opțiuni avansate de accesare a informațiilor.

CURS 10

57) Care este elementul central al unui framework MVC tipic și ce rol are acesta?

Răspuns:

Elementul central al unui framework MVC tipic este **Controller-ul**, deoarece acesta gestionează fluxul de date dintre Model și View. Controller-ul:

- Primește solicitările utilizatorului.
- Coordonează accesul la Model pentru manipularea datelor.
- Selectează View-ul corespunzător pentru afișarea rezultatelor.

58) Fie o aplicație web dezvoltată pe un framework MVC. Precizați cum ați realiza următoarele sarcini : 1. Crearea unui demo 2. Transformarea interfeței grafice în interfață în mod linie de comandă. 3. Transformarea aplicației în serviciu web 4. Modificarea suportului de persistență

Răspuns:

1. **Crearea unui demo:** Se poate genera rapid un schelet al aplicației folosind un generator de cod, precum rails new demo_app pentru Ruby on Rails.
2. **Transformarea interfeței grafice în interfață în mod linie de comandă:** Se poate crea un script CLI care interacționează direct cu Model-ul, fără a folosi un View HTML.
3. **Transformarea aplicației în serviciu web:** Implementarea unui API RESTful prin adăugarea de rute și controllere care returnează JSON în loc de HTML.
4. **Modificarea suportului de persistență:** Se poate schimba baza de date folosind ORM-uri precum ActiveRecord, iar migrarea datelor se face prin rails db:migrate.

59) Precizați din ce elemente ale modelului analiză al aplicației rezultă fiecare dintre componentele M, V și C.

Răspuns:

- **Model (M):** Provine din entitățile identificate în analiza domeniului și structura bazei de date.
- **View (V):** Se bazează pe scenariile de interacțiune ale utilizatorului și interfața vizuală a aplicației.
- **Controller (C):** Derivă din cazurile de utilizare ale aplicației și regulile de business care trebuie implementate.

60) Fie www.psswcm.com/lectures/WebMVC URI-ul unei resurse oferită de o aplicație web prin RESTful API. Unde vor fi rutate solicitările cu acest URI ?

Răspuns:

Solicitările către www.psswcm.com/lectures/WebMVC vor fi rutate către **LecturesController**, iar acțiunea va fi determinată în funcție de metoda HTTP utilizată:

- GET /lectures/WebMVC → LecturesController#show
- POST /lectures/WebMVC → LecturesController#create
- PUT /lectures/WebMVC → LecturesController#update
- DELETE /lectures/WebMVC → LecturesController#destroy

61) Ce trebuie să definească proiectantul și ce elemente se generează automat pentru crearea unui element de tip Model.

Răspuns:

- **Definite de proiectant:**
  - Atributele și relațiile modelului.
  - Reguli de validare și asocieri între modele.
  - Logica specifică Model-ului.
- **Generate automat:**
  - Fișierul modelului în /app/models/.
  - Migrarea bazei de date cu schema corespunzătoare.
  - Testele unitare pentru model.

62) Cum se poate realiza, în controller, separarea afișării de modificarea modelului.

Răspuns:

Separarea afișării de modificarea modelului se realizează prin definirea unor acțiuni distincte pentru fiecare responsabilitate. Acțiunile care preiau date din Model sunt separate de cele care modifică datele în Model. Aceasta permite o organizare clară a codului și facilitează testarea, menținerea și reutilizarea componentelor. De asemenea, utilizarea redirecționărilor și a mecanismelor de validare ajută la o mai bună gestionare a logicii de actualizare a datelor.

63) Care sunt tipurile de filtre ? La ce sunt utile filtrele ?

Răspuns:

Filtrele sunt folosite pentru a executa acțiuni înainte sau după rularea unui controller și se împart în:

- **Before filters:** Executate înainte de acțiunea controller-ului (ex. autentificare utilizator).
- **After filters:** Executate după acțiunea controller-ului (ex. logare acțiuni utilizator).
- **Around filters:** Învelesc întreaga acțiune, putând modifica atât intrările, cât și ieșirile (ex. măsurarea timpului de execuție a unei acțiuni). Filtrele sunt utile pentru **gestionarea autentificării, validarea cererilor și optimizarea performanței**.

OBS. Întrebările se referă la framework-ul Ruby on Rails.

CURS 11

CURS 12

64) Prin ce se caracterizează abordarea MDE în dezvoltarea de software ?

Răspuns: MDE (Model Driven Engineering) este o abordare a dezvoltării software în care un sistem este reprezentat ca un set de modele ce pot fi transformate automat în cod executabil.

- Modelele au diferite nivele de precizie:
  - Model ca schiță (pentru comunicare de idei și alternative).
  - Model ca ghid de implementare (documentare decizii de proiectare).

    - Model ca program (generare automată de aplicații).
- Se utilizează pentru comunicarea cu clienții, suport în proiectare, specificații pentru programatori și inginerie inversă (reverse engineering).

65) Care sunt categoriile de modele în MDA?

Răspuns:

- Computation Independent Model (CIM) – Modelează abstractizările importante de domeniu utilizate într-un sistem, fiind numit și model de domeniu.
- Platform Independent Model (PIM) – Modelează modul de funcționare a unui sistem fără referire la implementarea sa, folosind de obicei modele UML pentru a ilustra structura statică și interacțiunile sistemului.
- Platform Specific Model (PSM) – Transformări ale modelului PIM, cu un PSM pentru fiecare platformă de execuție.

66) Care sunt categoriile de elemente modelate cu IFML ?

Răspuns:

- Structura prezentării conținutului (model compoziție site).
- Legarea la modelul domeniului (stratul de persistență).
- Căile de navigare.
- Evenimentele.
- Legarea la logica business.

-

67) Care sunt parametrii de intrare ai acțiunii “Execute the payment”, de unde provin aceștia și pe ce tip de flux circulă fiecare?

Răspuns:

- Parametrii de intrare ai acțiunii "Execute the payment" sunt informațiile necesare pentru efectuarea plății, cum ar fi suma de plată, detaliile cardului, datele utilizatorului și metodele de autentificare.
- Aceștia provin din diferite surse, precum formularele completate de utilizator, datele din baza de date a utilizatorului și notificările de la sistemul bancar.
- Parametrii circulă pe diferite tipuri de flux:
    - Flux de navigare – atunci când utilizatorul selectează metoda de plată.
    - Flux de date – transferul informațiilor de plată către procesatorul de plăți.
    - Flux de eveniment – generarea confirmării plății și notificarea utilizatorului.

68) Ce elemente lipsesc din acest extras dintr-un model IFML?

Răspuns: Elementele care pot lipsi dintr-un extras de model IFML includ:

- Specificația detaliată a evenimentelor (ex. OnSubmit, OnSelect).
- Definirea conexiunilor dintre componente și logica business.
- Parametrii de legătură în fluxurile de date și navigare.

- Specificația completă a structurii de conținut a interfeței utilizatorului.
- Asocierile cu modelul domeniului și cu stratul de persistență.

CURS 13

69) Indicați două beneficii ale abordării orientate pe servicii în dezvoltarea aplicațiilor software.

Răspuns:

1. Integrarea serviciilor de la furnizori multipli – Aplicațiile pot utiliza servicii provenite de la diferiți furnizori, fie interni, fie externi, facilitând interoperabilitatea și reutilizarea componentelor.
2. Adaptabilitate și scalabilitate – Serviciile pot fi reutilizate și integrate în diferite aplicații, permițând scalabilitatea sistemului și reducerea timpului de dezvoltare.

70) Explicați următoarea afirmație : "Legarea serviciului la aplicație se face la instalare sau la execuție. Rezultă că aplicațiile pot fi reactive și își pot adapta operarea la modificările din contextul de execuție."

Răspuns:

- Prin legarea serviciilor la instalare sau execuție, aplicațiile pot selecta dinamically cel mai potrivit serviciu disponibil în funcție de contextul de execuție.
- Aceasta permite aplicațiilor să fie mai flexibile, să adapteze utilizarea resurselor și să optimizeze performanța, de exemplu, prin schimbarea furnizorului de servicii în funcție de disponibilitate sau costuri.

71) Elementele definiției unui serviciu software sunt interfața (CE), modul de acces (CUM) și punct(e) de acces (UNDE). Explicați ce simplificări aduce abordarea RESTful comparativ cu WSDL ? CE = interfața • Operațiile • Formatele mesajelor trimise și recepționate de serviciu CUM = binding • Corespondența între interfața abstractă și un set concret de protocoale; specifică detaliile tehnice ale comunicării cu serviciul Web. UNDE • Localizarea implementării serviciului (endpoint).

Răspuns:

- CE (Interfața): RESTful folosește metode HTTP standardizate (GET, POST, PUT, DELETE) pentru interacțiuni simple, pe când WSDL impune o structură complexă bazată pe XML.
- CUM (Modul de acces – Binding): RESTful elimină nevoia de binding-uri complicate, folosind un set standardizat de protocoale HTTP, pe când WSDL necesară configurarea explicită a binding-urilor.
- UNDE (Puncte de acces): RESTful identifică resursele prin URI-uri intuitive, reducând complexitatea localizării serviciului, pe când WSDL necesară definirea explicită a endpoint-urilor.
- Alte simplificări: RESTful are un overhead mai redus față de SOAP/WSDL, permite caching și este mai potrivit pentru aplicații distribuite și interacțiuni web moderne.

CURS 14

72) Tipurile fundamentale de procese ale unui sistem RT/E.

Răspuns:

1. Proces de control senzor - colectează informații de la senzori, eventual bufferizând datele.
2. Proces de procesare date - prelucrează datele colectate și calculează răspunsul sistemului.
3. Proces de control element de acționare - generează semnale de control pentru dispozitivele de acționare.

73) Enumerați activitățile de proiectare pentru sistemele RT/E.

Răspuns:

- Selectarea platformei hardware și a sistemului de operare de timp real (RTOS).
- Identificarea perechilor stimul-răspuns.
- Analiza cerințelor de timp și stabilirea termenelor limită (deadlines).
- Proiectarea proceselor concurente.
- Proiectarea algoritmilor pentru optimizarea prelucrării.
- Proiectarea datelor și stabilirea evenimentelor pentru comunicare.
- Planificarea execuției proceselor.
- Verificarea modelului prin simulări sau analize statice.

74) Realizați corespondența corectă între model și diagramele UML folosite.

Răspuns:

- Model de stare → Diagrama de stări și tranziții.
- Model interacțiuni obiecte → Diagrama de secvențe.
- Model comportament timp → Diagrama de timp.
- Model execuție procese → Diagrama de activitate.

75) Ce diagrame UML se folosesc în mod special pentru sistemele RT/E ?

Răspuns:

- Diagrame de stări și tranziții - pentru modelarea comportamentului sistemului.
- Diagrame de secvențe - pentru modelarea interacțiunilor dintre componente.
- Diagrame de timp - pentru analiza comportamentului sistemului în raport cu constrângerile temporale.
- Diagrame de activitate - pentru reprezentarea proceselor sistemului.

76) Enumerați și descrieți pe scurt șabloanele arhitecturale specifice sistemelor RT/E.

Răspuns:

1. Observe and React - Monitorizează un set de senzori, afișează date și inițiază acțiuni în cazul unor condiții excepționale.
2. Environmental Control - Utilizează senzori și elemente de acționare pentru a modifica contextul sistemului.
3. Process Pipeline - Configurare de conducte de procesare a datelor, folosind buffer-e pentru sincronizare.

77) Sistemele de timp real sunt proiectate ca procese cooperante controlate de un executiv de timp real (real-time executive). Care este rolul analizei de timp și care sunt factorii considerați în analiza de timp a acestor procese cooperante.

Răspuns:

- Rolul analizei de timp: Se determină frecvența de execuție a fiecărui proces astfel încât toate intrările să fie procesate la timp și să se respecte cerințele de timp.
- Factorii considerați:
  - Termenele limită (deadlines) - momentele la care trebuie finalizat un proces.
  - Frecvența de execuție - cât de des trebuie rulat un proces.
  - Timpul de execuție - durata procesării unui stimul și generării răspunsului.
  - Sisteme de timp real hard vs. soft - analiza trebuie să ia în considerare cel mai defavorabil caz (worst-case execution time) pentru sistemele hard.