Analiza și proiectarea sistemelor software

curs 2

PLAN CURS

Analiză arhitecturală

Analiza cazurilor de utilizare

Adaptat după un curs IBM pentru OOAD (OO Analysis and Design)

ANALIZĂ BAZATĂ PE COMPONENTE

În RUP, arhitectura unui sistem software este:

- Organizarea/structura componentelor și a subcomponentelor semnificative ale sistemului şi modul de interacţiune al acestora prin interfeţe.

Activităţile disciplinei de Analiză şi Proiectare sunt organizate în jurul a 2 teme majore:

- Structura - responsabilitate a arhitectului software
  - Nivele arhitecturale
  - Componente şi interfeţe
- Conţinutul - responsabilitate a proiectanţilor
  - Clasele de analiză şi clasele de proiectare

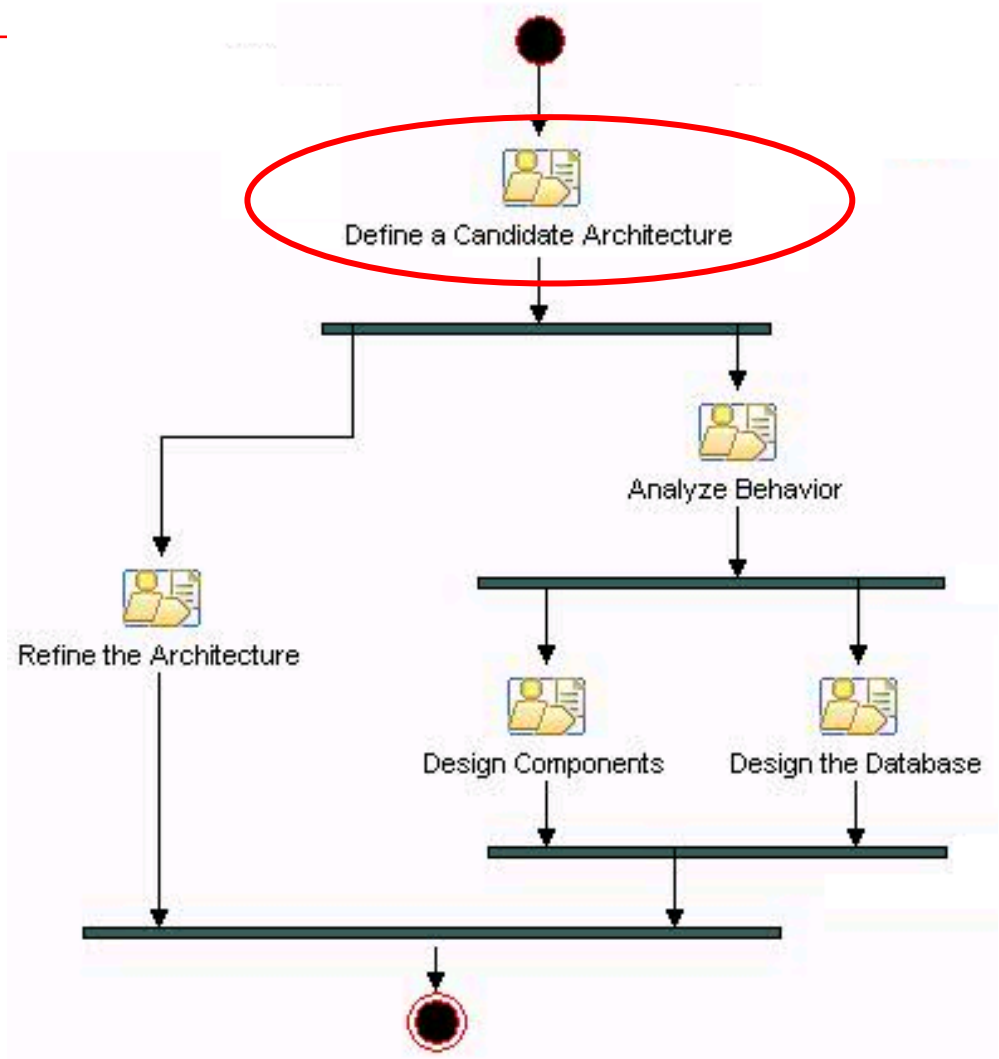
ANALIZĂ BAZATĂ PE COMPONENTE
Etape

Analiză

- Analiză arhitecturală (definire arhitectură candidat)
- Analiza UC (analiză comportament)

Proiectare

- Identificare elemente de proiectare (rafinarea arhitecturii)
- Identificare mecanisme de proiectare (rafinarea arhitecturii)
- Proiectare clase (proiectare componente)
- Proiectare subsisteme (proiectare componente)
- Descrierea arhitecturii la execuţie şi a distribuirii (rafinarea arhitecturii)
- Proiectarea BD



ANALIZĂ ARHITECTURALĂ

Scop

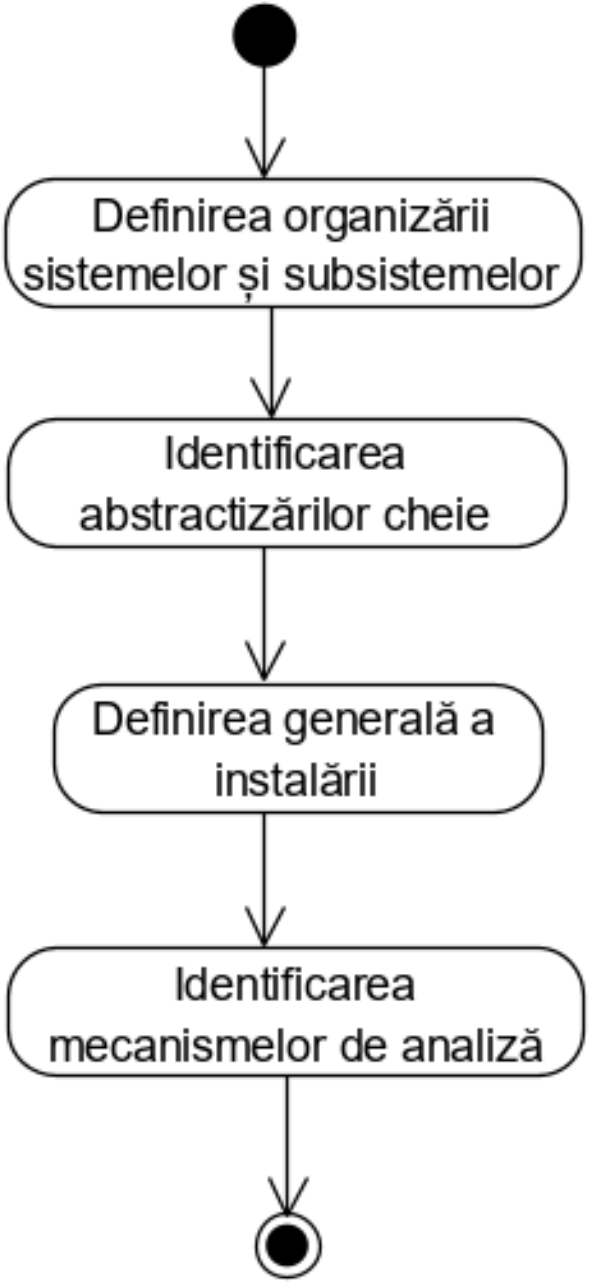
- Definirea unei *arhitecturi candidat* pentru sistem (în general, pe baza experienţei dobândite pentru sisteme similare sau domenii de problemă similare).
- Definirea *şabloanelor arhitecturale*, *mecanismelor cheie* şi *convenţiilor de modelare* pentru sistem.

Rol (responsabil)

- Arhitectul software

Etape majore

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea și descrierea mecanismelor de analiză



DEFINIREA ORGANIZĂRII LA NIVEL ÎNALT A SUBSISTEMELOR

Scop

- Crearea unei *structuri iniţiale* a modelului proiect

În mod tipic, arhitectura este organizată pe nivele (layer-e) – **şablon arhitectural** comun pentru sisteme de dimensiuni medii sau mari.

Analiza arhitecturală a aplicaţiilor software – concentrare pe 2 nivele superioare: **aplicaţie** şi **business.**

ŞABLOANE şi CADRE (frameworks)

•Definirea organizării la nivel înalt a subsistemelor
•Identificarea abstractizărilor cheie
•Definirea generală a instalării
•Identificarea mecanismelor de analiză

Şablon (pattern)

- Oferă o soluţie comună la o problemă comună într-un context dat.

Şablon de analiză/proiectare

- Oferă o soluţie la o problemă tehică pe un domeniu îngust
- Oferă un fragment al unei soluţii

Cadru (framework)

- Defineşte o abordare generală a soluţionării problemei
- Oferă o soluţie schelet, ale cărei detalii pot fi şabloane de analiză/proiectare.

ȘABLOAN ARHITECTURAL

•Definirea organizării la nivel înalt a subsistemelor
•Identificarea abstractizărilor cheie
•Definirea generală a instalării
•Identificarea mecanismelor de analiză

Un şablon arhitectural exprimă o schemă de organizare structurală fundamentală pentru sisteme software

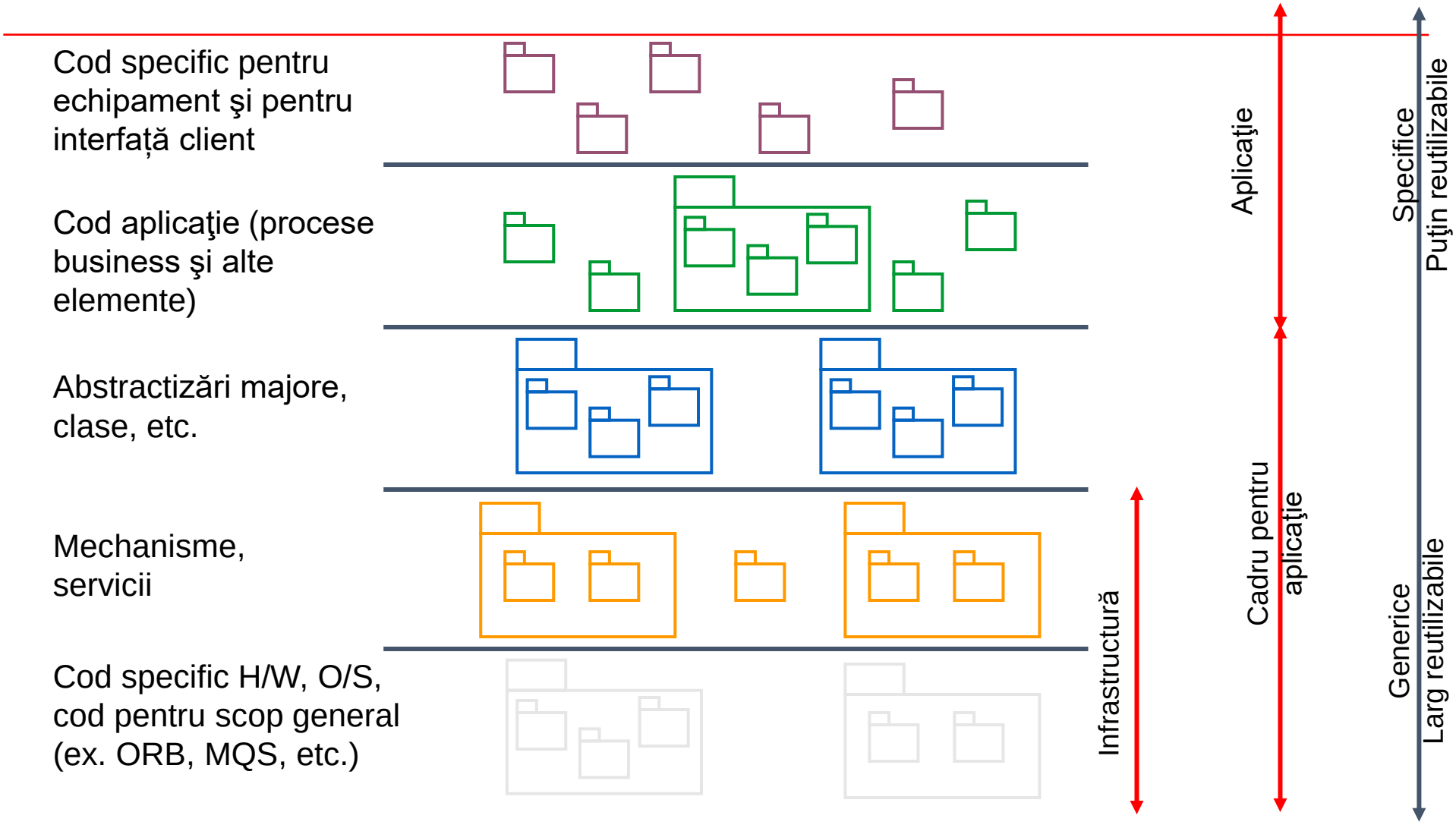
- Oferă un set de sisteme predefinite, le specifică responsabilităţile şi include reguli şi directive pentru organizarea relaţiilor dintre ele. – *Buschman et al, "Pattern-Oriented Software Architecture — A System of Patterns"*

Exemple:

- Layers (multi-nivel)
- Model-view-controller (MVC)
- Pipes and filters
- Blackboard

ARHITECTURI MULTI-NIVEL

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză



ARHITECTURI MULTI-NIVEL

•Definirea organizării la nivel înalt a subsistemelor
•Identificarea abstractizărilor cheie
•Definirea generală a instalării
•Identificarea mecanismelor de analiză

Organizează unităţile de implementare ale sistemului pe mai multe ***nivele de abstractizare***.

- Nivel – bibliotecă sau colecţie de servicii înrudite.
- Fiecare nivel oferă un set de servicii accesibile printr-o ***interfaţă***(ex. API).
- Suportă ***dezvoltarea incrementală*** a sub-sistemelor de pe nivele diferite.
- Interacţiunea are loc doar între nivele ***adiacente***, un nivel superior accesând serviciile oferite de nivelul inferior.
- Când se schimbă interfaţa unui nivel, doar nivelele adiacente sunt afectate.

DEFINIRE NIVELE ARHITECTURALE

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză

Nivel de abstractizare

- Gruparea elementelor de pe acelaşi nivel de abstractizare

Separarea problemelor

- Gruparea elementelor asemănătoare, corelate
- Separarea elementelor disparate
- Elemente ale aplicaţiei vs. Elemente ale modelului domeniului

Flexibilitate (elasticitate)

- Cuplare slabă între nivele
- Concentrare pe încapsularea schimbărilor

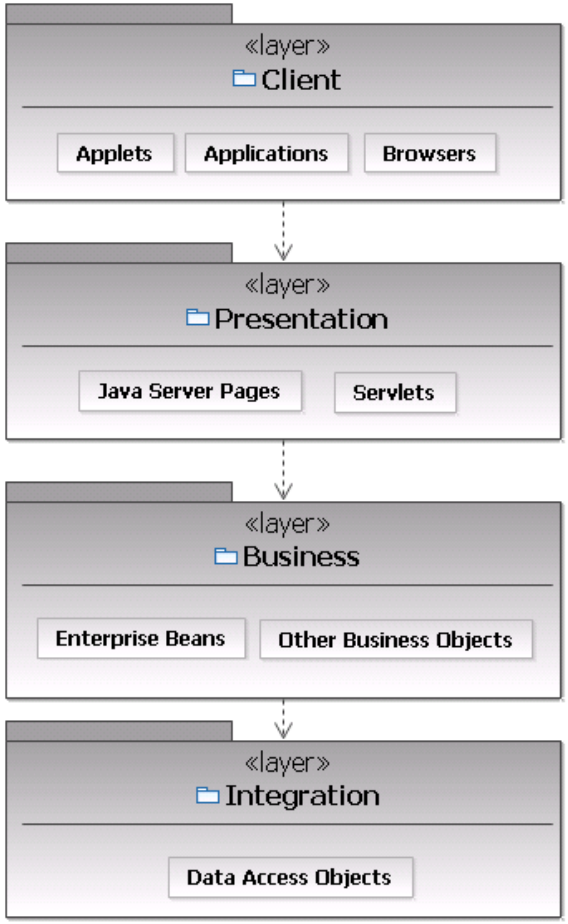
Obs. UI, regulile business şi datele ce trebuie păstrate tind să aibă potenţial mare de modificare. ?

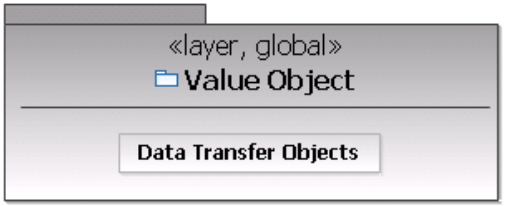
MODELARE NIVELE ARHITECTURALE

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză

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

Exemplu : Nivele software pentru o aplicaţie JavaEE generică





Obs: <<global>> convenţie utilizată pentru nivelele ce pot fi utilizate de toate celelalte nivele.

IDENTIFICAREA ABSTRACTIZĂRILOR CHEIE

Def. ***Abstractizare cheie*** = concept din domeniul problemei pe care sistemul trebuie să fie capabil să-l manipuleze.

Surse

- Cunoştinţele despre domeniu
- Cerinţe
- Glosar
- Modelul domeniului sau modelul business (dacă există)

DESCRIEREA ABSTRACTIZĂRILOR CHEIE

Abstractizări cheie – modelate sub formă de ***clase de analiză***

Pentru fiecare clasă se oferă

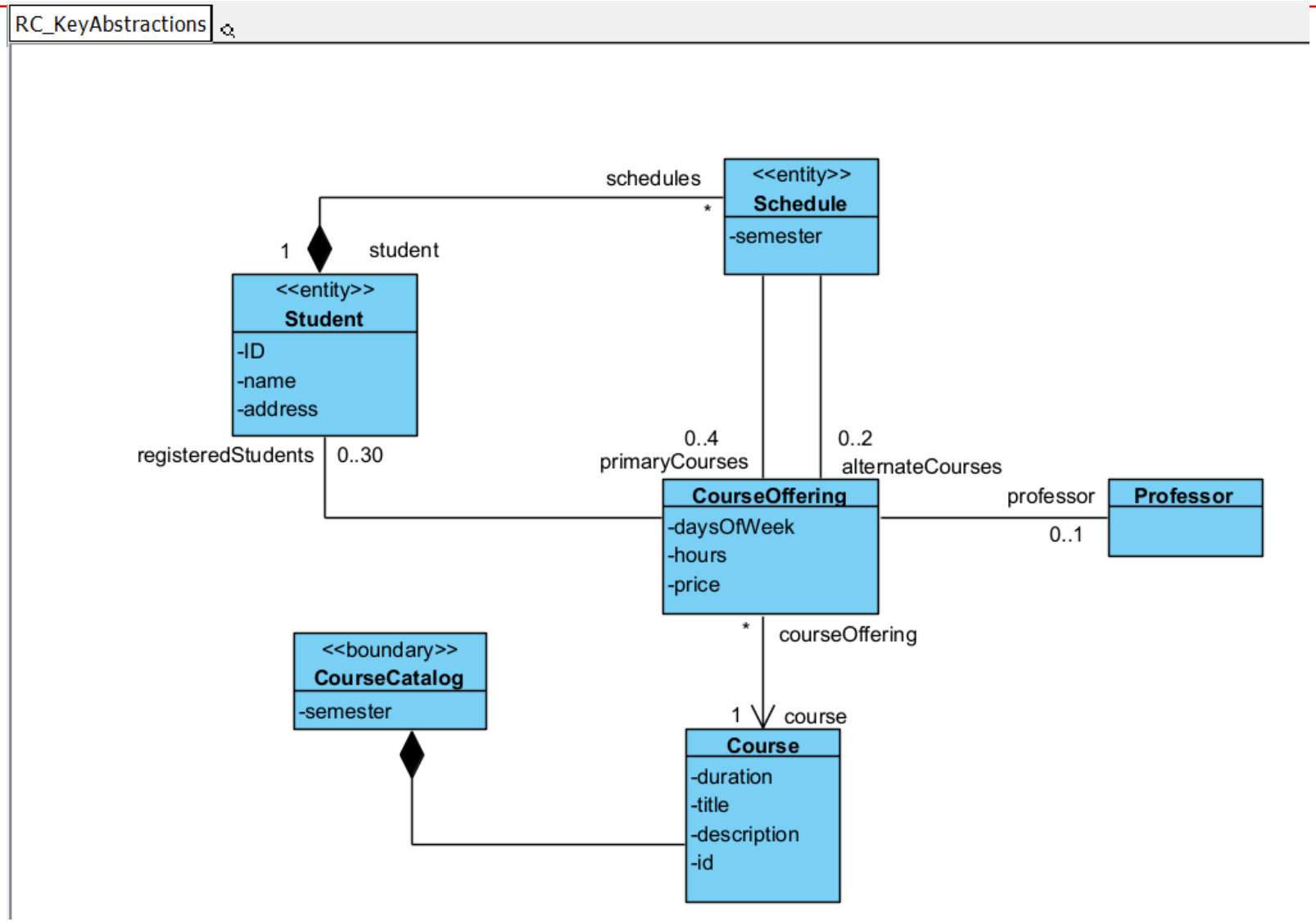
- Descriere scurtă
- Atribute principale
- Relaţiile cu alte clase

Detaliile vor fi lăsate în seama etapelor următoare

- Nu se urmăreşte identificarea completă şi definitivă a claselor aplicaţiei
- E posibil să fie identificate clase care să nu fie neaparat necesare cazurilor de utilizare
- Set iniţial de clase

Exemplu:
Course Registration System

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză



ANALIZA ARHITECTURALĂ
Evaluare formativă

1. Care este regula de comunicare între nivelele unei arhitecturi multinivel?

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

3. Cum se poate reprezenta pe modelul UML un nivel arhitectural ?

4. 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.

https://forms.gle/JriDafZPkc5uVxBn8

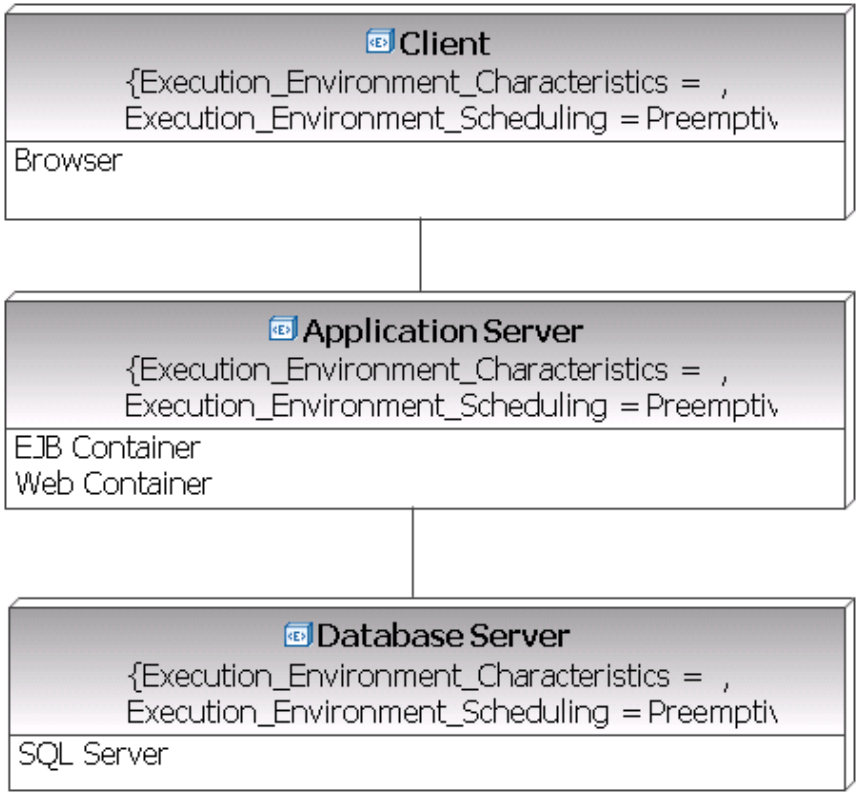
DEFINIREA GENERALĂ A INSTALĂRII

Scop:

- Înţelegerea distribuirii geografice şi a complexităţii operaţionale a sistemului

Dezvoltarea unei scheme generale a modului în care software-ul va fi amplasat, pentru a ilustra:

- Accesul la distanţă
- Distribuirea pe noduri multiple
- Componentele Hw şi Sw existente



Exemplu : Definire generală a instalării unei aplicaţii JavaEE generică.

MECANISME DE ANALIZĂ

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză

Def. Mecanismele de analiză = mecanisme arhitecturale* utilizate în fazele iniţiale ale procesului de analiză şi proiectare:

- Capturează aspectele cheie ale soluţiei în manieră *independentă de aplicaţie*
- Sunt *concepte din domeniul "computer science",* de obicei fără legătură cu domeniul problemei
- Oferă *comportamente specifice* pentru clasele de domeniu sau pentru componente.

Exemple:

- Persistenţă
- Comunicare între procese (IPC)
- Manipulare erori sau avarii
- Notificare
- Transfer de mesaje, etc.

***** Mecanisme arhitecturale = Soluţii arhitecturale concrete comune la probleme frecvent întâlnite.

IDENTIFICAREA ȘI DESCRIEREA MECANISMELOR DE ANALIZĂ

Identificare top-down (cunoaştere a priori) sau bottom-up (descoperite pe parcurs)

- Iniţial ar putea exista doar numele (ex. persistenţă)

Pe măsură ce sunt identificate clasele client (pentru mecanism), este necesară calificarea utilizării fiecărui mecanism

- Pentru persistenţă, se vor identifica caracteristici ca granularitate (dimensiune înregistrare), volum (nr. de înregistrări), mecanism de extragerere, frecvenţă de actualizare, etc.

Mecanismele de analiză vor fi rafinate pentru a deveni mecanisme de proiectare

- Un mecanism de proiectare presupune unele detalii legate de contextul de implementare, dar nu este legat de o anumită implementare.
- Exemplu: SGBD ca mecanism de proiectare pentru persistenţă

Mecanismele de proiectare vor fi rafinate pentru a deveni mecanisme de implementare

- Exemplu: Oracle SGBD

EXEMPLE DE MECANISME DE ANALIZĂ

•Definirea organizării la nivel înalt a subsistemelor
•Identificarea abstractizărilor cheie
•Definirea generală a instalării
•Identificarea mecanismelor de analiză

- Persistenţă
- Comunicare între procese (IPC şi RPC)
- Transfer mesaje
- Distribuire
- Gestiune tranzacţii
- Control şi sincronizare procese
- Schimb de informaţii, conversii de format
- Securitate
- Detectare/manipulare/raportare erori
- Redundanţă
- Interfaţă cu sistem legacy

EXEMPLE - MECANISME ȘI CARACTERISTICI

- Definirea organizării la nivel înalt a subsistemelor
- Identificarea abstractizărilor cheie
- Definirea generală a instalării
- Identificarea mecanismelor de analiză

Mecanism	Caracteristici
Persistență	**Granularitate**: domeniul de valori al dimensiunii obiectelor persistente **Volum**: numărul de obiecte ce trebuie păstrate **Durata**: durata de păstrare a obiectelor **Mecanismul de acces**: identificarea unică şi extragerea unui obiect **Frecvenţa de acces**: frecvenţa acceselor pentru modificare **Fiabilitatea**: necesitatea de supravieţuire a obiectelor faţă de procese, procesor, sistem.
Comunicare între procese (IPC, RPC)	**Latenţa**: viteza de comunicare între procese **Sincronicitatea**: comunicare sincronă sau asincronă **Dimensiunea mesajului**: spectru de valori pentru dimensiune. **Protocol**, flux de control, buffering, etc.
Interfață cu sistem legacy	**Latenţa** **Durata** **Mecanismul de acces** **Frecvenţa de acces**
Securitate	**Granularitate date**: nivel pachet, nivel clasă, nivel atribut **Granularitate utilizatori**: utilizatori individuali, roluri/grupuri **Reguli de securitate**: bazate pe valorile datelor, pe algoritmi bazaţi pe date, pe algoritmi bazaţi pe utilizator şi date **Tipuri de privilegii**: citire, scriere, creare, ştergere, executarea unei anumite operaţii

PLAN CURS

Analiză arhitecturală

Analiza cazurilor de utilizare

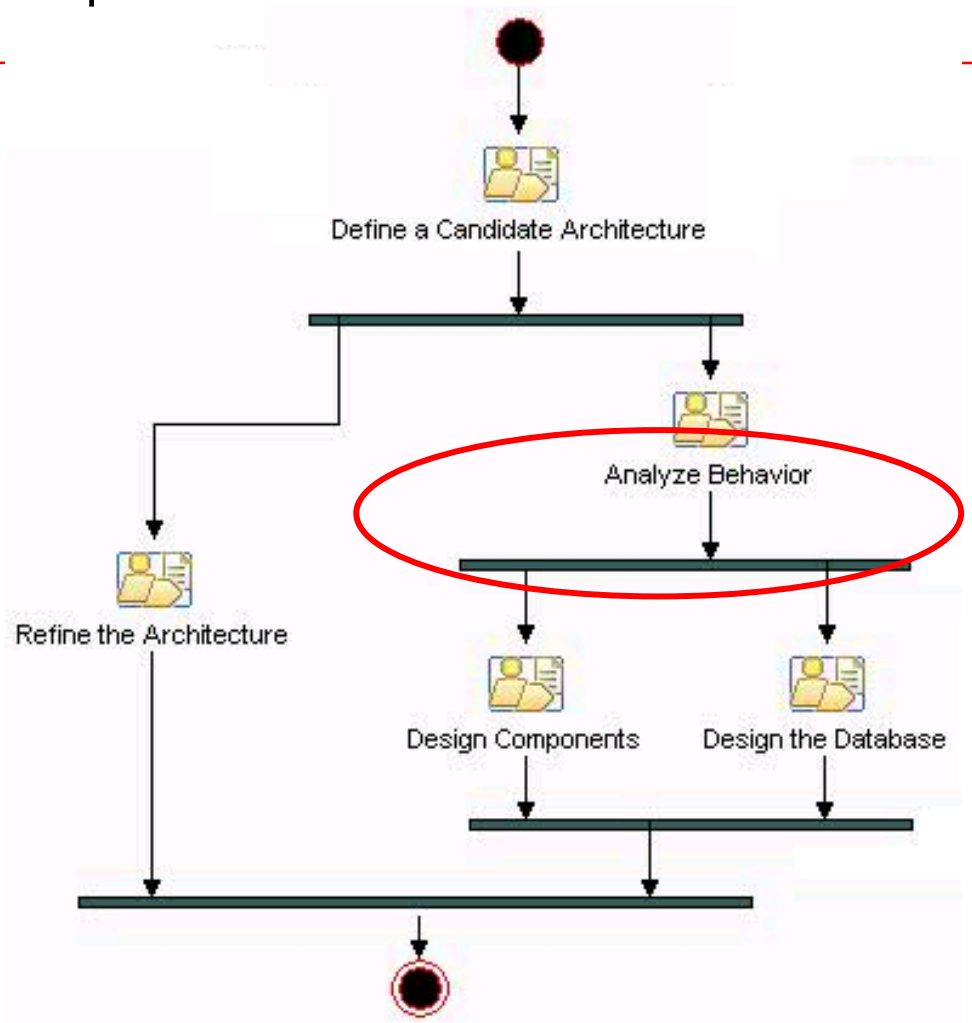
ANALIZĂ BAZATĂ PE COMPONENTE
Etape

Analiză

- Analiză arhitecturală (definire arhitectură candidat)
- Analiza UC (analiză comportament)

Proiectare

- Identificare elemente de proiectare (rafinarea arhitecturii)
- Identificare mecanisme de proiectare (rafinarea arhitecturii)
- Proiectare clase (proiectare componente)
- Proiectare subsisteme (proiectare componente)
- Descrierea arhitecturii la execuţie şi a distribuirii (rafinarea arhitecturii)
- Proiectarea BD



ANALIZA CAZURILOR DE UTILIZARE

Scop

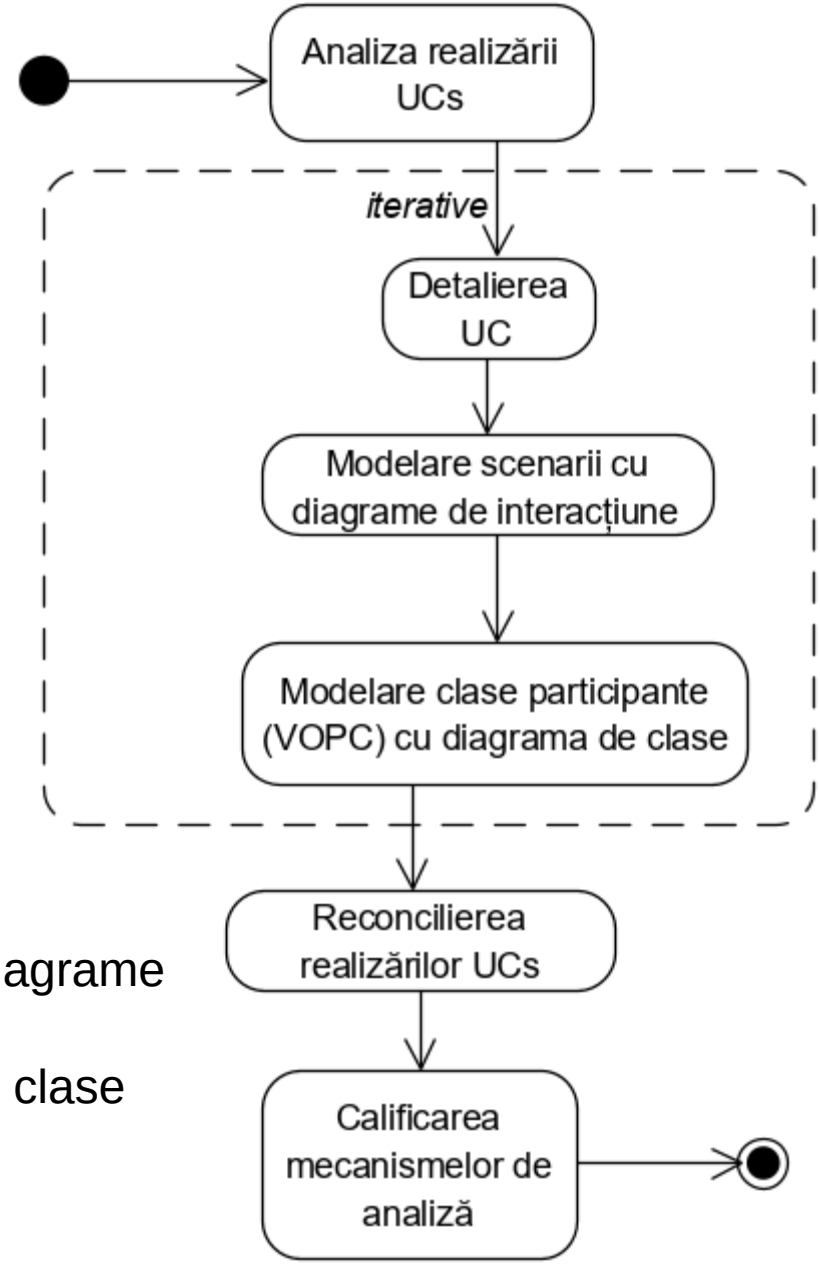
- Identificarea claselor de analiză pentru sistem, incluzând:
  - *Responsabilităţile*, *atributele* şi *relaţiile de asociere* cu alte clase
  - Utilizarea *mecanismelor* de analiză

Rol (responsabil)

- Proiectant

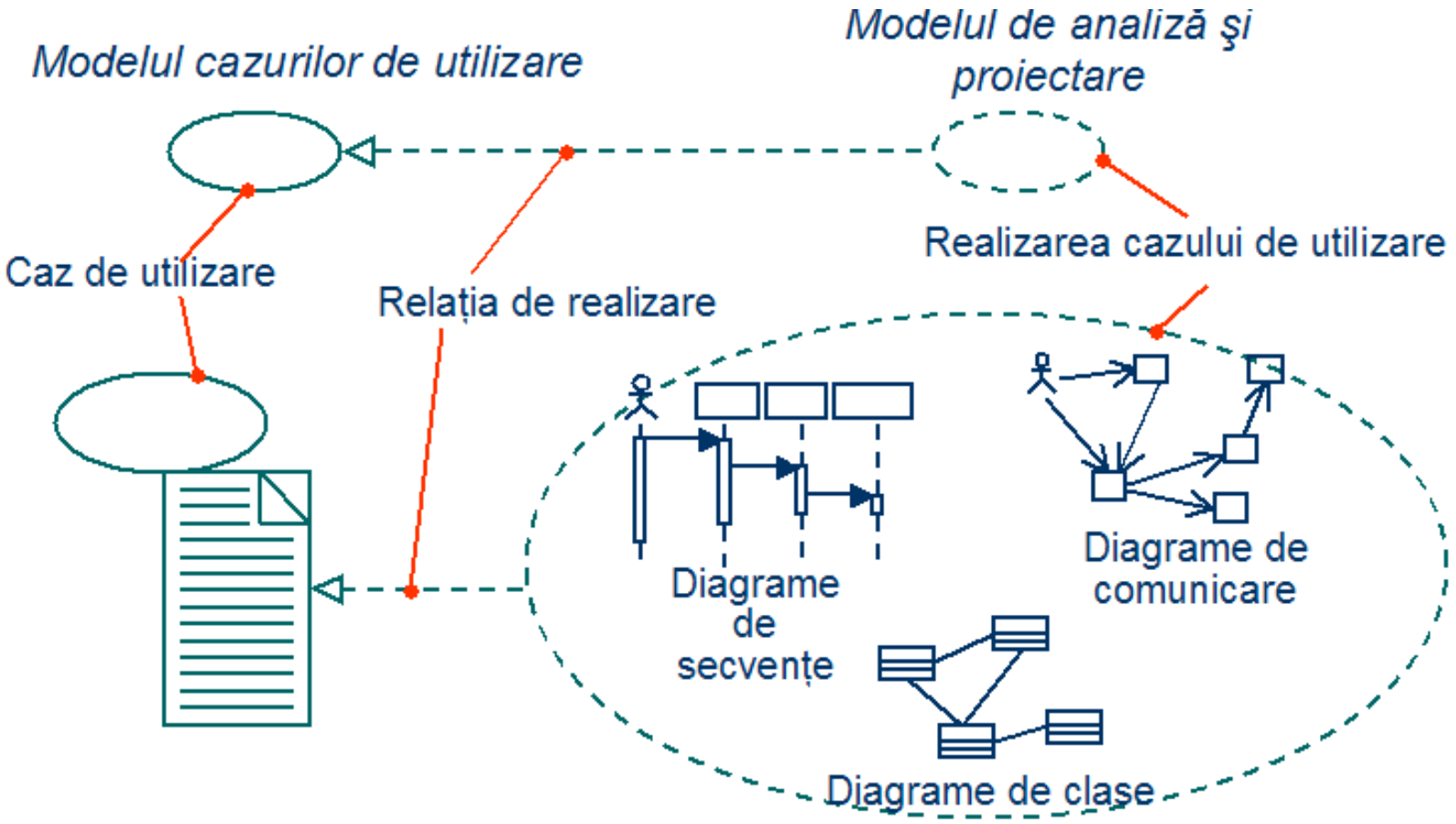
Etape majore

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



REALIZAREA CAZURILOR DE UTILIZARE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



* Colaborare UML = structură formată din elemente (roluri) aflate în colaborare, fiecare realizând o funcţie specializată, pentru a îndeplini împreună o anumită funcţionalitate.

REALIZAREA CAZURILOR DE UTILIZARE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Descrie modul în care un caz de utilizare este *realizat* în cadrul modelelor analiză şi proiect, în termeni de *obiecte ce colaborează prin schimb de mesaje*.

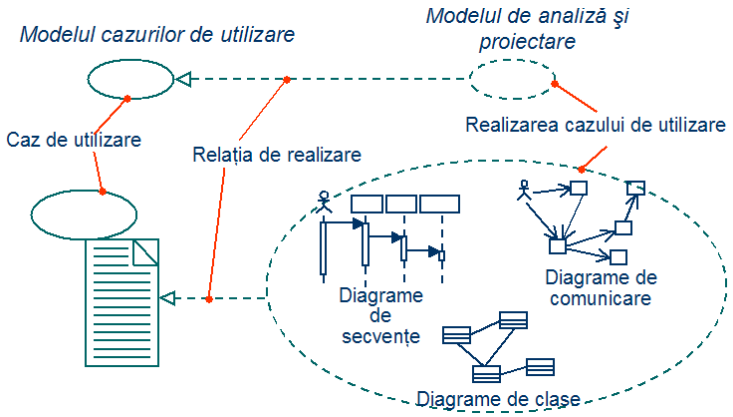
Leagă cazurile de utilizare cu clasele şi relaţiile din modelele analiză şi proiect.

Specifică care sunt clasele ce trebuie construite pentru a implementa fiecare caz de utilizare.

Construcţie în modelele analiză şi proiect care organizează artefactele ce contribuie la realizarea cazului de utilizare.

- În mod tipic conţine diagrame de secvenţe şi diagrame de clase

Reprezentată în UML ca o colaborare* stereotipată cu <<*use-case realization*>> aflată în relaţie de "realizare" cu cazul de utilizare realizat.



* Colaborare UML = structură formată din elemente (roluri) aflate în colaborare, fiecare realizând o funcţie specializată, pentru a îndeplini împreună o anumită funcţionalitate.

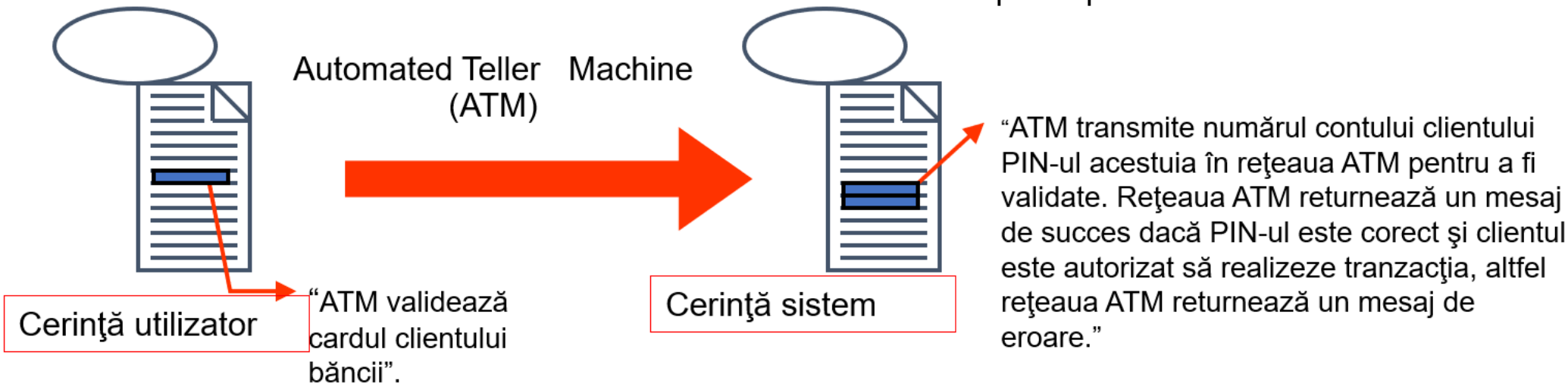
COMPLETAREA DESCRIERII CAZURILOR DE UTILIZARE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Scop: Capturarea de *informaţii suplimentare* necesare pentru înţelegerea *comportamentului intern* solicitat sistemului.

Definire fluxuri de comunicare între actori și sistem reprezentate *textual* sau cu *diagrame de secvențe la nivel de sistem*.

Exemplu: reprezentare textuală.



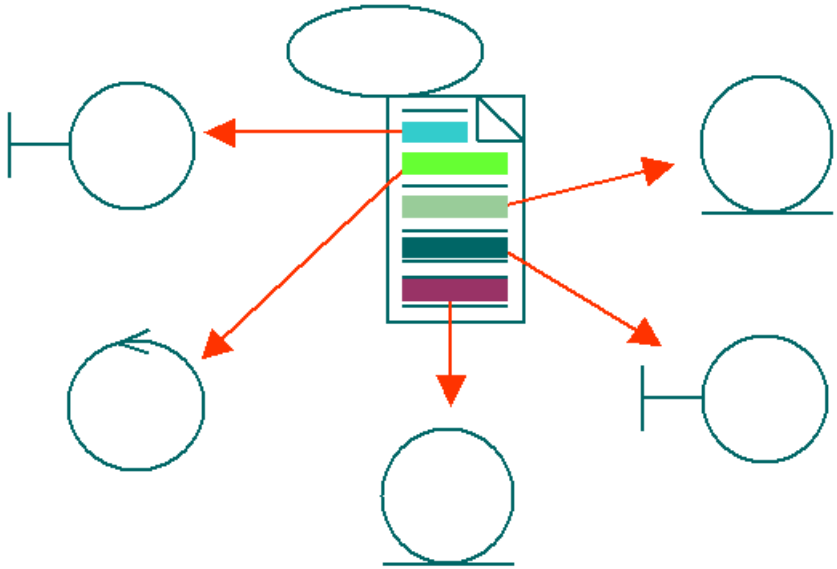
Pentru o mai bună înțelegere a comportamentului intern, acesta se poate modela cu *diagrame de activitate*.

MODELARE SCENARII CU DIAGRAME DE INTERACȚIUNE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Ideea: Identificare clase pe baza comportamentului descris în cazul de utilizare.

Comportamentul unui caz de utilizare trebuie repartizat claselor de analiză.



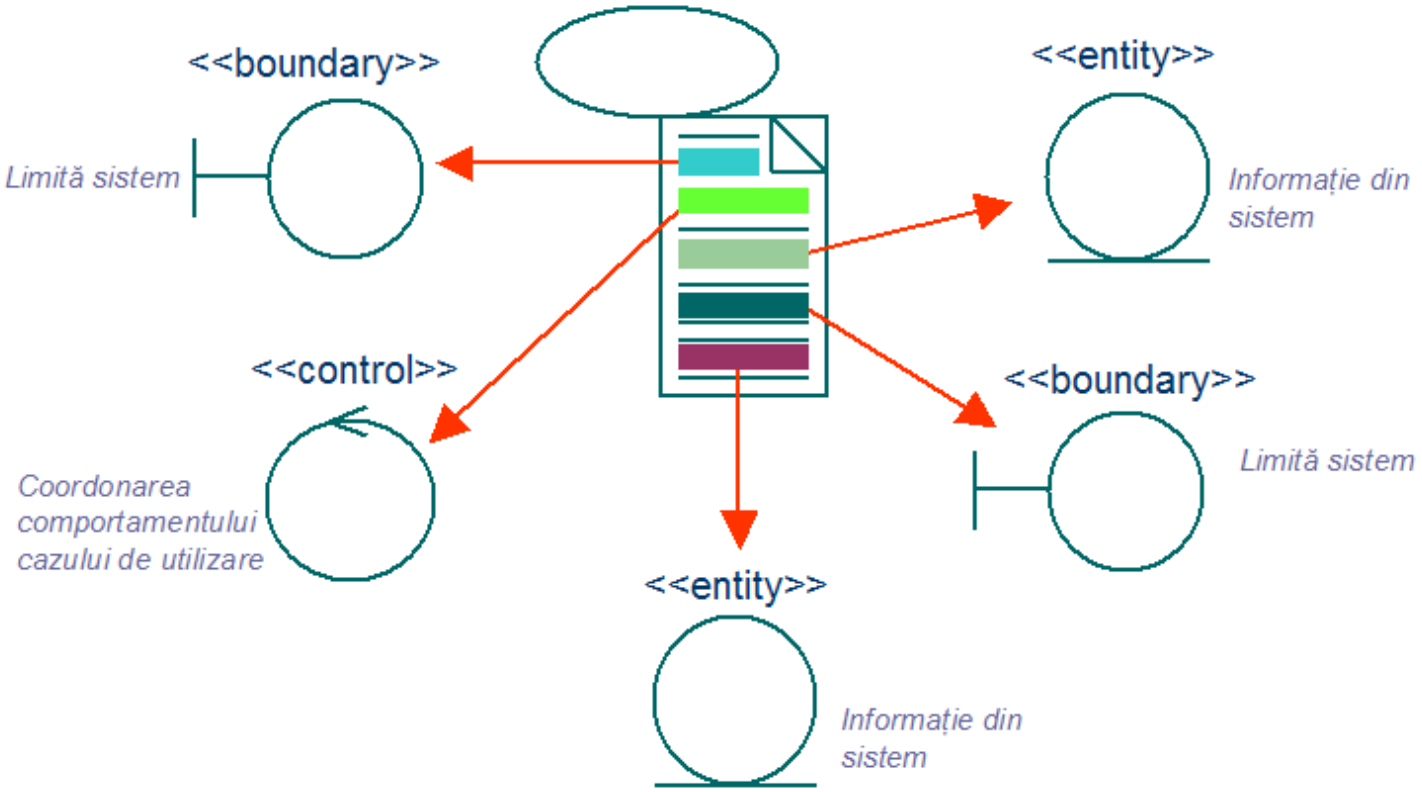
CATEGORII DE CLASE DE ANALIZĂ

•Analiza realizării cazurilor de utilizare
•Completarea descrierilor cazurilor de utilizare
•Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
•Modelarea claselor participante cu diagrame de clase
•Reconcilierea realizărilor cazurilor de utilizare
•Calificarea mecanismelor de analiză

Scop:

Obţinerea independenţei modificărilor prin separarea între

- interfaţa cu exteriorul
- informaţiile utilizate în sistem
- logica sistemului.



CLASĂ <<boundary>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Intermediar între sistem şi actor

Tipuri

- Clase UI
- Clase de interfaţă cu alte sisteme
- Clase de interfaţă cu dispozitive externe

Dependente de context

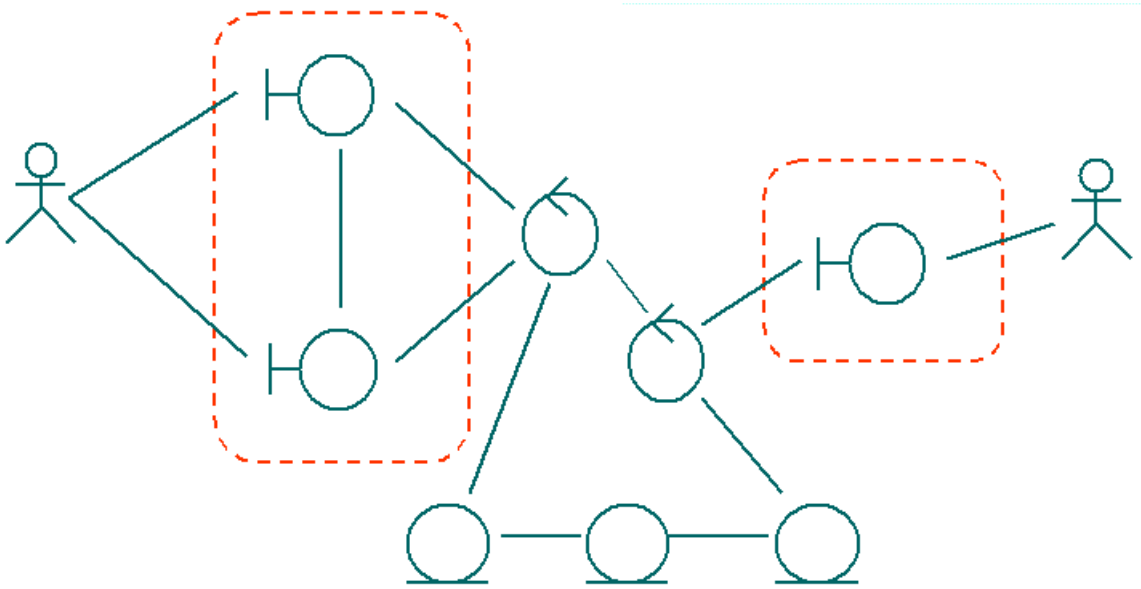
- GUI
- Protocolale de comunicare

ROLUL UNEI CLASE <<boundary>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Modelează ***interacţiunea dintre sistem şi context***

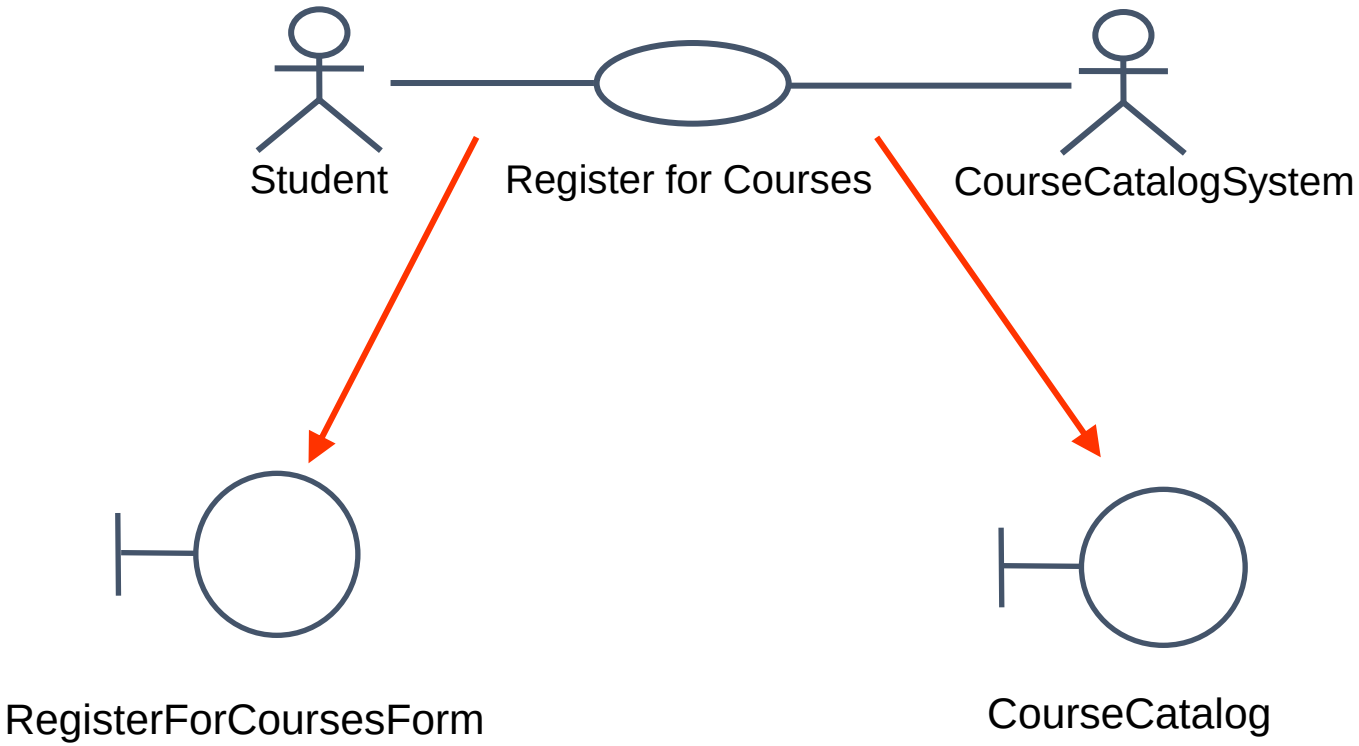
- Transformare şi traducere evenimente
- Notificare de modificări



GĂSIREA CLASELOR <<boundary>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Iniţial – se va considera o clasă <<boundary>> pentru fiecare pereche actor - caz de utilizare.



RECOMANDĂRI: CLASĂ <<boundary>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Clase UI

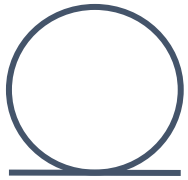
- Concentrare pe informaţia prezentată utilizatorului (în asociere cu prototipul UI)
- Evitare detalii de proiectare a UI

Clase pentru interfaţa cu alte sisteme şi cu dispozitive externe

- Concentrare pe protocoalele ce trebuie definite
- Evitarea detaliilor de implementare a protocoalelor

Concentrare pe ***responsibilităţi***, nu pe detalii!

CLASĂ <<entity>>



- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Reprezintă ***conceptele cheie*** ale sistemului.

Modelează informaţiile ce trebuie memorate

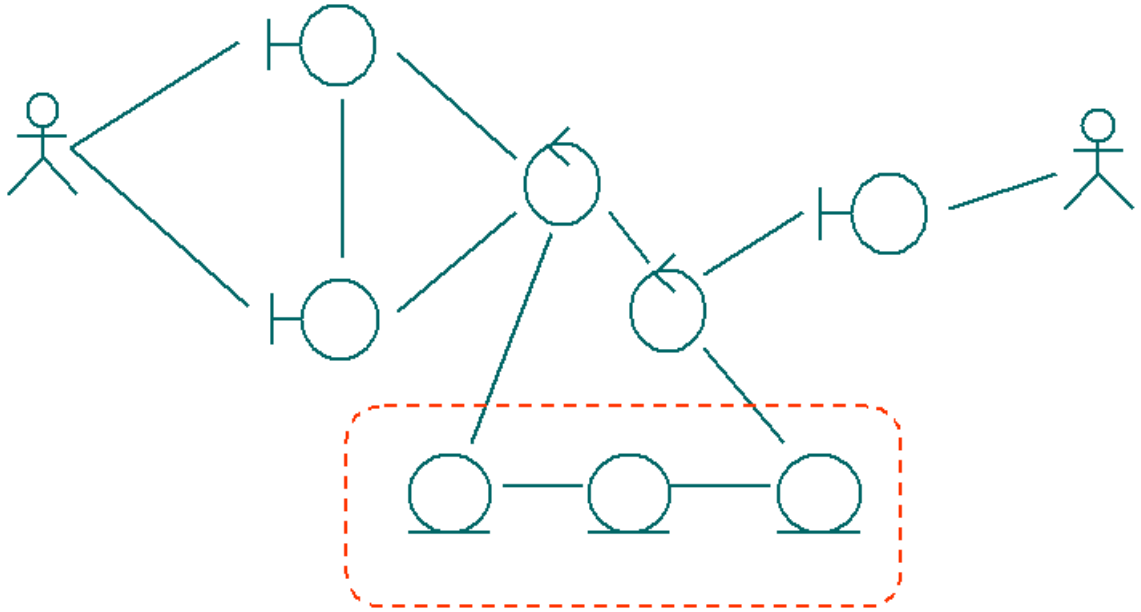
- Persistente (în general)
- Pot avea comportament complex, strâns legat de fenomenul pe care clasa îl reprezintă

Independente de context (de actori)

Nu sunt specifice unui singur caz de utilizare.

ROLUL CLASEI <<entity>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



Memorarea şi ***gestionarea informaţiilor*** din sistem

GĂSIREA CLASELOR <<entity>>

Abstractizările cheie devin, în general, clase <<entity>>

Se pot identifica şi din:

- Fluxul de evenimente al cazului de utilizare
- Glosar
- Modelul domeniului business

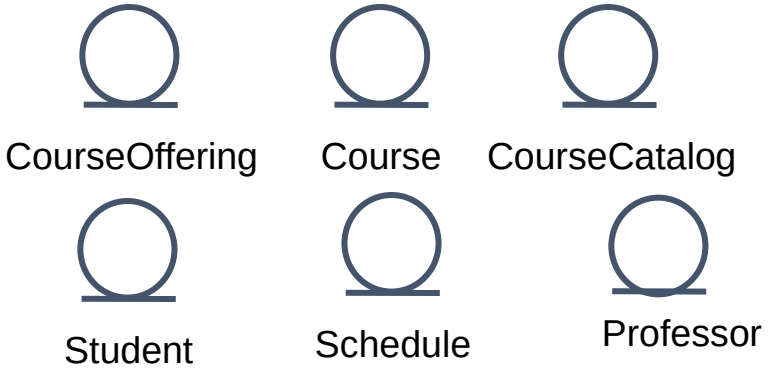
Se vor căuta informaţiile sistem ce trebuie memorate:

- Substantive sau construcţii substantivale care identifică date persistente sunt candidate să devină :
  - Atribute ale unei clase <<entity>>
  - Clase <<entity>>

Exemplu:
Course Registration System

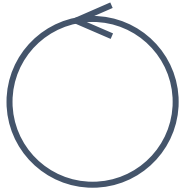
- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Abstractizări cheie



Clase nou identificate

CLASĂ <<control>>



•Analiza realizării cazurilor de utilizare
•Completarea descrierilor cazurilor de utilizare
•Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
•Modelarea claselor participante cu diagrame de clase
•Reconcilierea realizărilor cazurilor de utilizare
•Calificarea mecanismelor de analiză

Realizează ***coordonarea comportamentului*** cazului de utilizare

Obs.

1. Cazurile de utilizare complexe pot necesita mai multe clase <<control>>

   Exemple: Manageri de tranzacţii, coordonatori de resurse, tratare erori

2. Cazurile de utilizare ce realizează simpla manipulare a informaţiei memorate se pot realiza folosind numai obiecte <<boundary>> şi <<entity>>

Îşi poate ***delega*** o serie de responsabilităţi către alte clase.

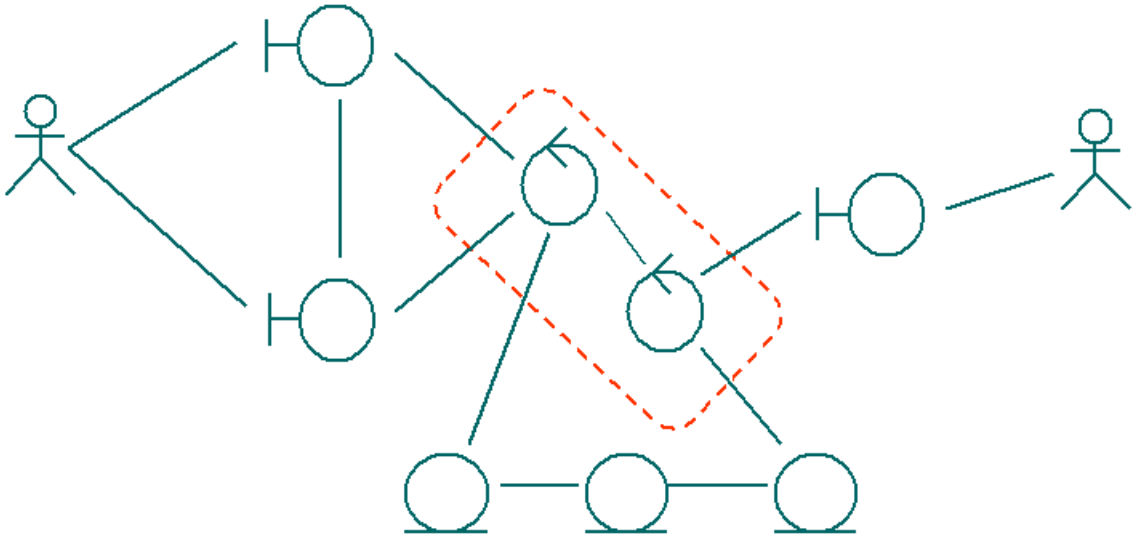
Este *dependentă de cazul de utilizare*, dar *independentă de context*. Poate participa la mai multe cazuri de utilizare strâns corelate.

Caracteristicile comportamentului unei clase <<control>>

- Defineşte logica de control (ordinea evenimentelor) şi tranzacţiile din cadrul unui caz de utilizare.
- Suportă modificări minore dacă se modifică structura internă sau comportamentul claselor <<entity>>.
- Utilizează sau setează conţinutul uneia sau mai multor clase <<entity>>, în consecinţă este implicată în coordonarea comportamentului acestor clase.

ROLUL CLASEI <<control>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



Coordonează comportamentul cazului de utilizare

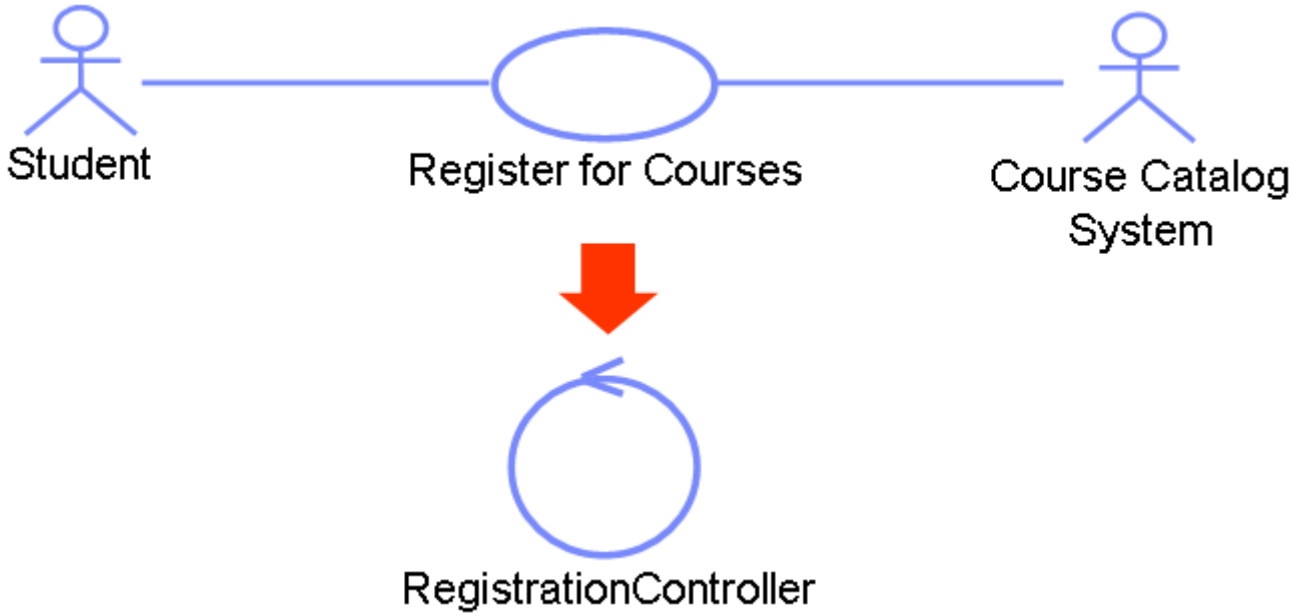
GĂSIREA CLASELOR
<<control>>

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Iniţial se identifică o clasă de control per caz de utilizare.

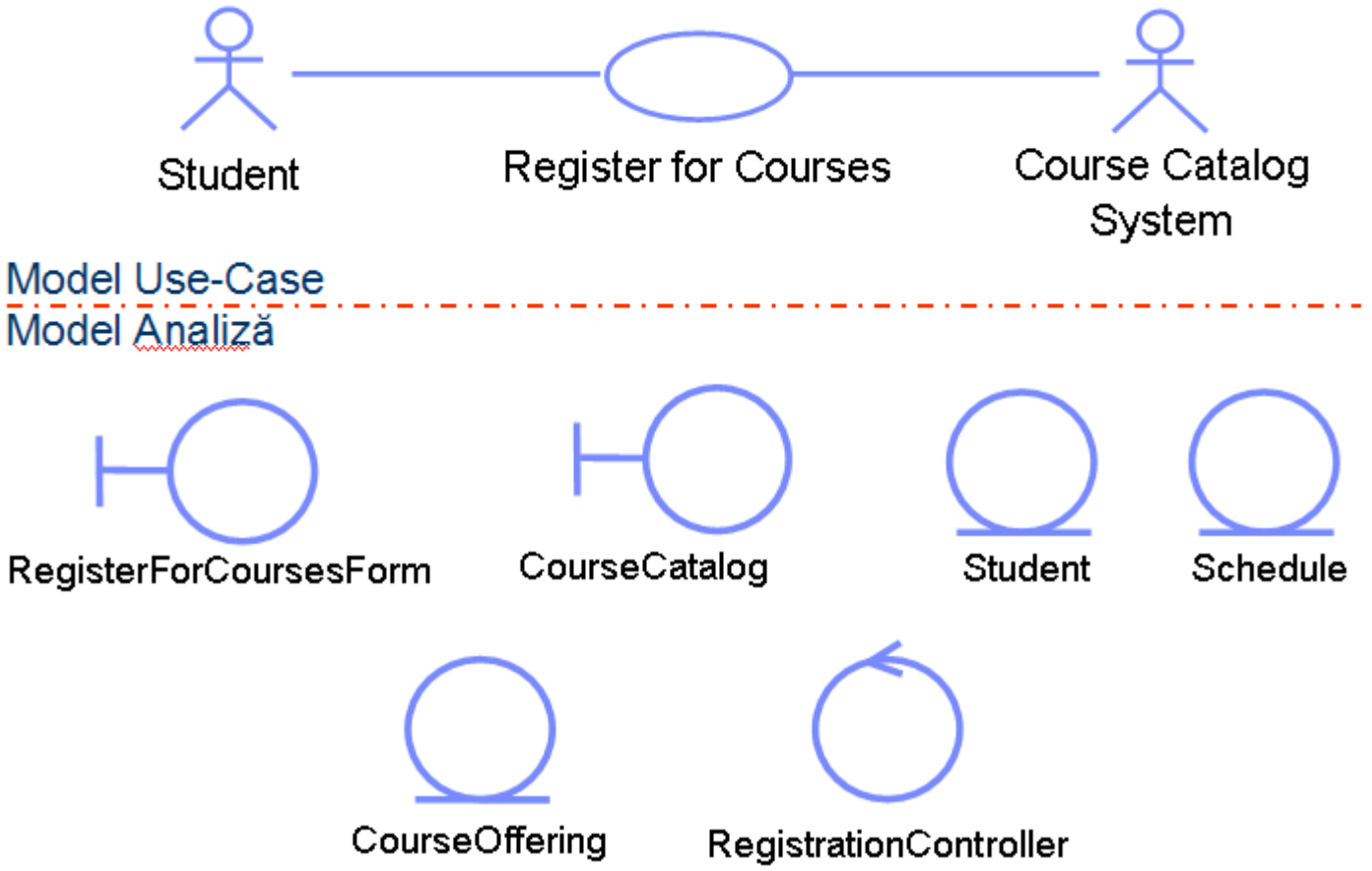
Pe măsură ce analiza evoluează

- o clasă de control complexă a unui caz de utilizare poate evolua în mai multe clase
- o clasă de control poate participa la mai multe cazuri de utilizare
- o parte din responsabilităţile clasei de control pot fi preluate direct de clase <<entity>> sau <<boundary>>



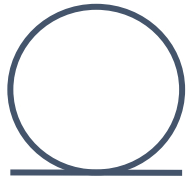
Exemplu:
Course Registration System
UC: Register for Courses

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

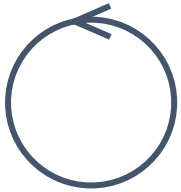


ANALIZA ARHITECTURALĂ
Evaluare formativă

Indicați tipul obiectului reprezentat în figură și ce se modelează cu acest tip de obiect.



[ ] , [ ]



[ ] , [ ]

[ ] , [ ]

Tip :

(a) Control

(b) Boundary

(c) Entity

Modelează :

(x) interacțiune dintre sistem și context

(y) informații din sistem

(z) coordonarea comportamentului

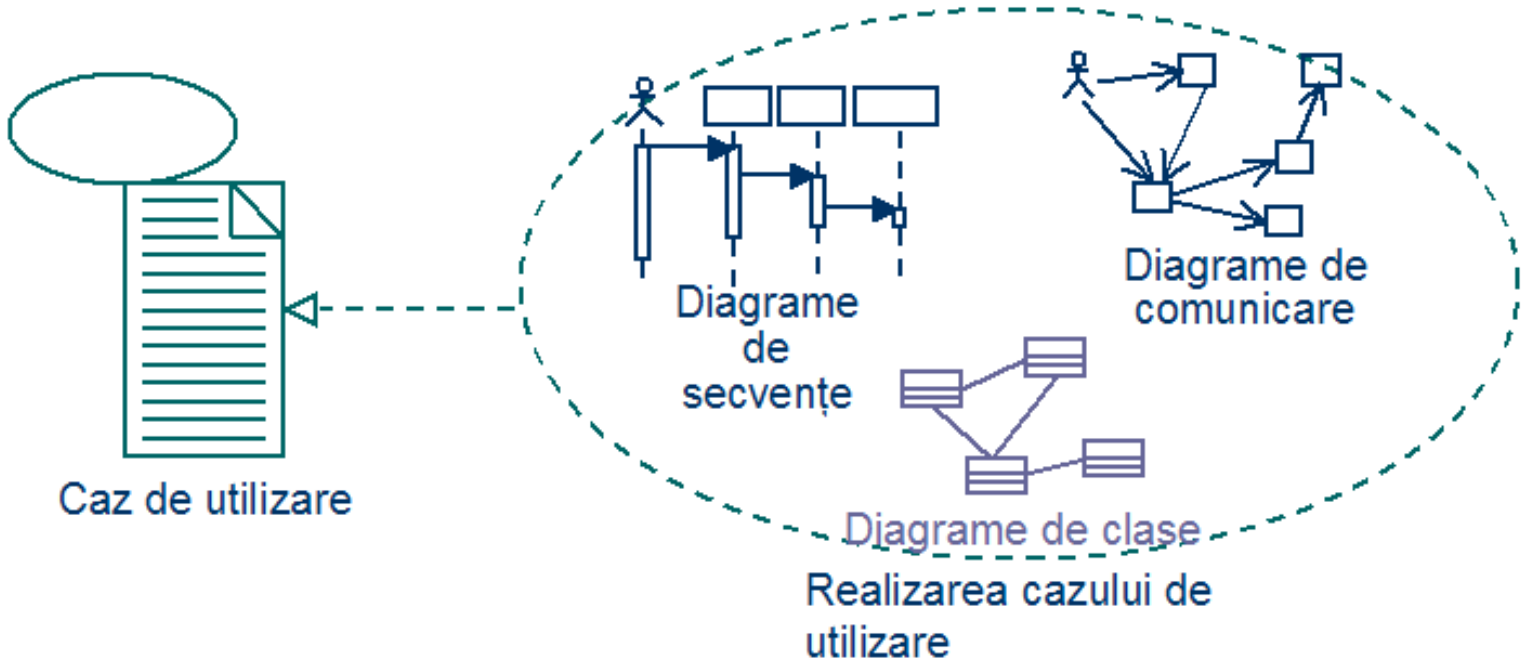
https://forms.gle/AjatSkcLGDfh12Bg7

DISTRIBUIREA COMPORTAMENTULUI CAZULUI DE UTILIZARE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Pentru fluxurile de evenimente ale fiecărui caz de utilizare:

- Modelarea comportamentului de colaborare a obiectelor din sistem, cu una sau mai multe *diagrame de interacţiune* (*secvenţe*, comunicare).
- *Identificarea claselor de analiză* din care se instanțiază obiectele responsabile cu comportamentul solicitat de cazul de utilizare.
- *Alocarea responsabilităţilor* cazului de utilizare la clasele de analiză.



DISTRIBUIREA COMPORTAMENTULUI CAZULUI DE UTILIZARE

Ghid: Diagrame de interacţiune şi cazuri de utilizare

Fiecare diagramă de interacţiune descrie un scenariu al unui caz de utilizare

- Diagramele trebuie să fie numite conform scenariilor cazurilor de utilizare.
- Interacţiunea trebuie să înceapă cu un actor, deoarece întotdeauna un actor invocă cazul de utilizare.

Nu este întotdeauna suficientă o singură diagramă. Pentru cazurile de utilizare complexe pot fi necesare :

- Cel puţin o diagramă pentru fluxul principal
- Cel puţin o diagramă pentru fiecare alternativă non-trivială sau flux excepţional
- Diagrame separate pentru fluxurile complexe

DISTRIBUIREA COMPORTAMENTULUI CAZULUI DE UTILIZARE

•Analiza realizării cazurilor de utilizare
•Completarea descrierilor cazurilor de utilizare
•Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
•Modelarea claselor participante cu diagrame de clase
•Reconcilierea realizărilor cazurilor de utilizare
•Calificarea mecanismelor de analiză

Ghid: Creare obiecte şi clase

Înainte de analizarea cazului de utilizare, plasaţi pe diagrama de secvențe:

- Obiectul actor care iniţiază cazul de utilizare
- Obiectele <<boundary>> corespunzătoare
- Un obiect <<control>> (care poate fi numit cu numele cazului de utilizare urmat de cuvântul Control)
- Obiecte <<entity>> ce trebuie să existe înainte de lansarea cazului de utilizare
  - Exemplu: dacă există o precondiţie ca studentul să fie “logged in”, este probabil că sistemul a extras deja obiectul Student corespunzător, deci acesta va fi plasat pe diagramă.

Asignarea fiecărui obiect din diagrama de secvențe la o clasă existentă sau la o clasă nouă în diagrama de clase:

- La crearea unei clase noi realizaţi imediat şi capturarea semanticilor acesteia:
  - Stereotipul de analiză (<<boundary>>, <<control>> sau <<entity>>)
  - Descriere, atribute, relaţii

Obs: În final clasele vor fi organizate în pachete şi layer-e dar aceasta nu este problema analizei cazurilor de utilizare

DISTRIBUIREA COMPORTAMENTULUI CAZULUI DE UTILIZARE

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Ghid: Alocare responsabilităţi

Comportamentul cazului de utilizare se materializează în obiecte ce comunică prin mesaje.

- La crearea unui mesaj creaţi operația corespunzătoare a clasei
  - Convenţie: începeţi numele operaţiei cu “//”
    - Identificare operaţie a clasei ca fiind o ***responsabilitate de analiză***
    - Exemplu: *// extrage ofertele de curs pentru semestrul curent*
    - În timpul proiectării, responsabilităţile vor fi rafinate în operaţii “reale”

- Identificaţi ce clase deţin datele necesare îndeplinirii responsabilităţii
  - Dacă o clasă deţine datele, responsabilitatea va fi asignată acesteia
  - Dacă datele se află în mai multe clase, există următoarele variante:
    - Plasarea responsabilităţii într-una dintre clase şi adăugarea de relaţii cu celelalte
    - Plasarea responsabilităţii într-o altă clasă (ex. clasă nouă sau clasă de control existentă) şi adăugarea de relaţii cu clasele necesare îndeplinirii responsabilităţii.

DISTRIBUIREA COMPORTAMENTULUI CAZULUI DE UTILIZARE

Ghid: Control centralizat vs. descentralizat

Control centralizat

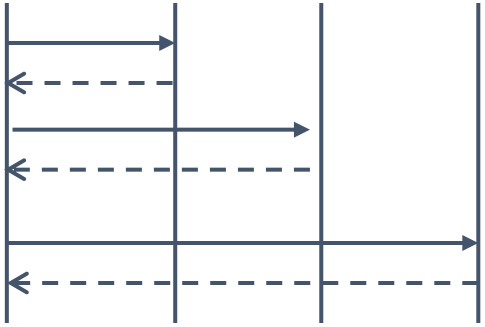
- Un obiect le controlează pe celelalte
- Interacţiunea dintre celelalte obiecte este minimală sau inexistentă

Control descentralizat

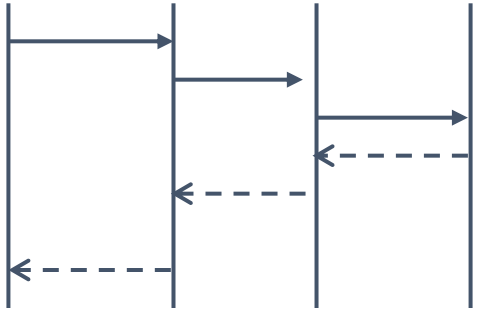
- Nu există un obiect central de control
- Fiecare obiect cunoaşte doar o parte din restul obiectelor
- Mai apropiat de “OO”, dar analizaţi impactul dacă se modifică ordinea operaţiilor.

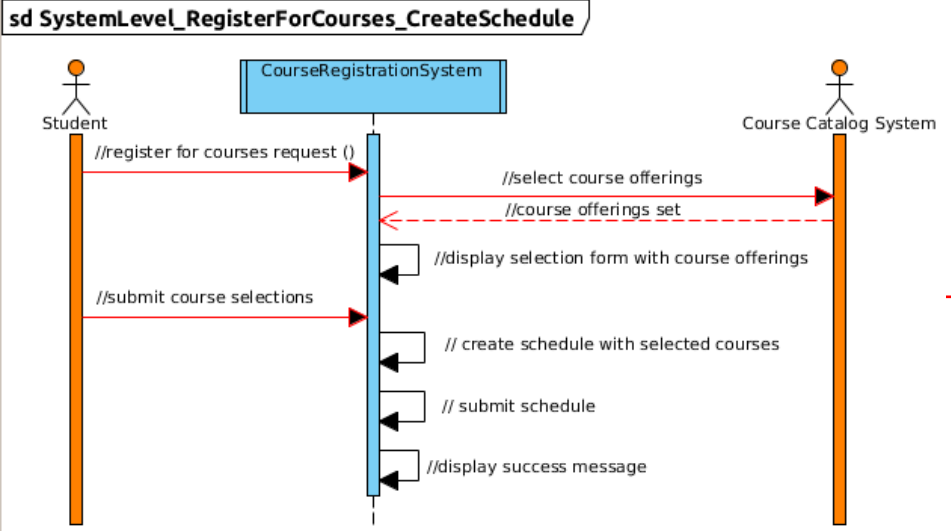
Deseori cele 2 strategii sunt combinate

centralizat

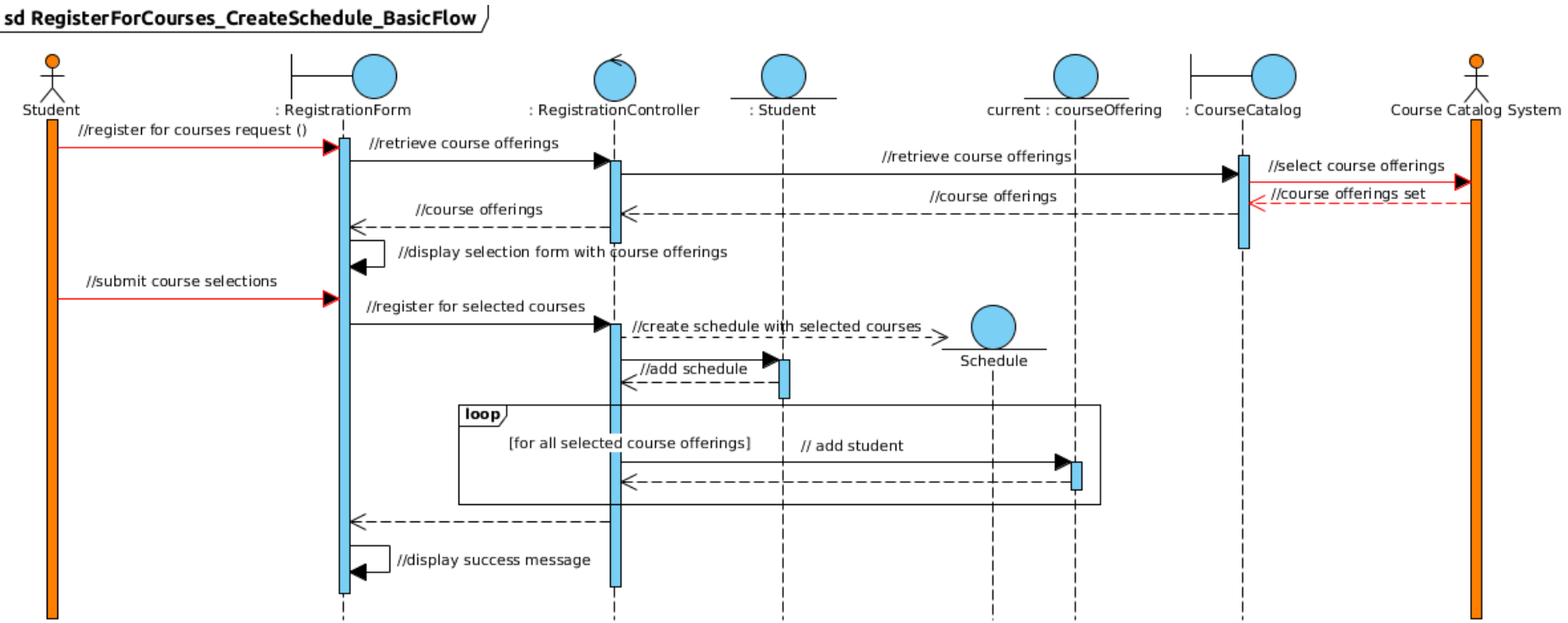


decentralizat



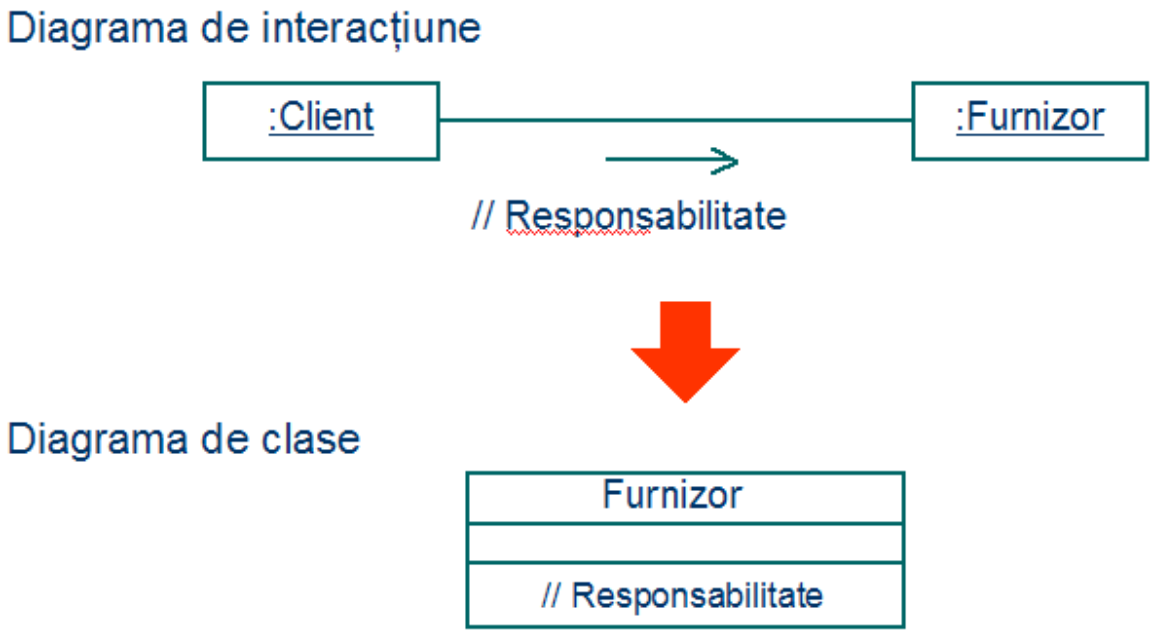


- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



GĂSIRE RESPONSABILITĂȚI

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



OBS:

Atât diagrama de secvenţe cât şi diagrama de comunicare sunt diagrame de interacţiune.

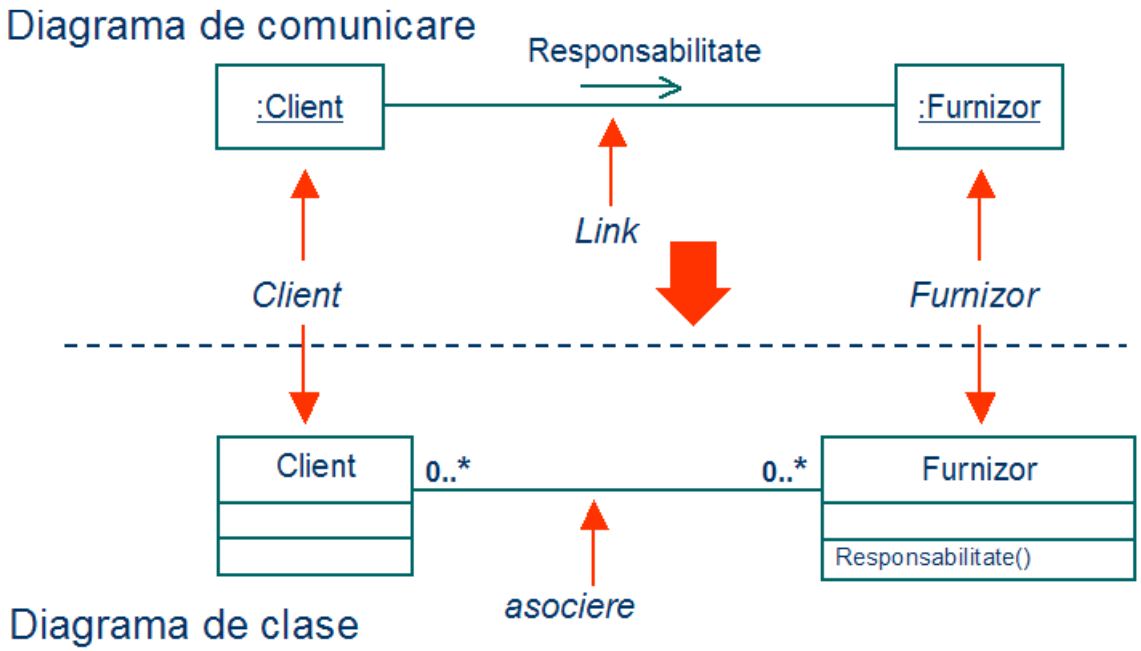
Ambele reprezintă obiecte care colaborează prin schimbul de mesaje dintre acestea.

Pe oricare dintre ele se pot identifica responsabilităţile obiectelor către care sunt trimise mesajele.

GĂSIRE RELAȚII

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

Câte o relaţie de asociere pentru fiecare link!



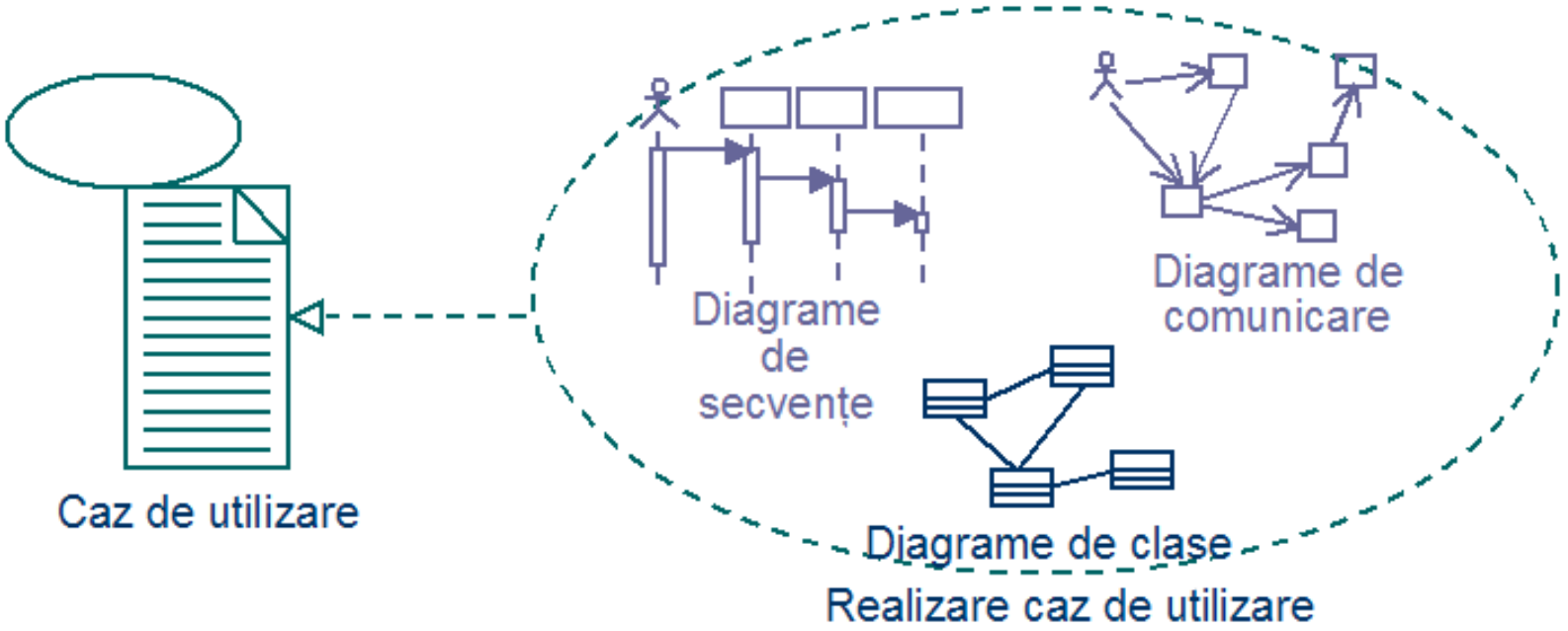
OBS:

Atât pe diagrama de secvenţe cât şi pe diagrama de comunicare se pot identifica legăturile (link) dintre obiectele ce colaborează prin transfer de mesaje.

CREAREA UNEI VEDERI CU CLASELE PARTICIPANTE (VOPC) LA REALIZAREA CAZULUI DE UTILIZARE

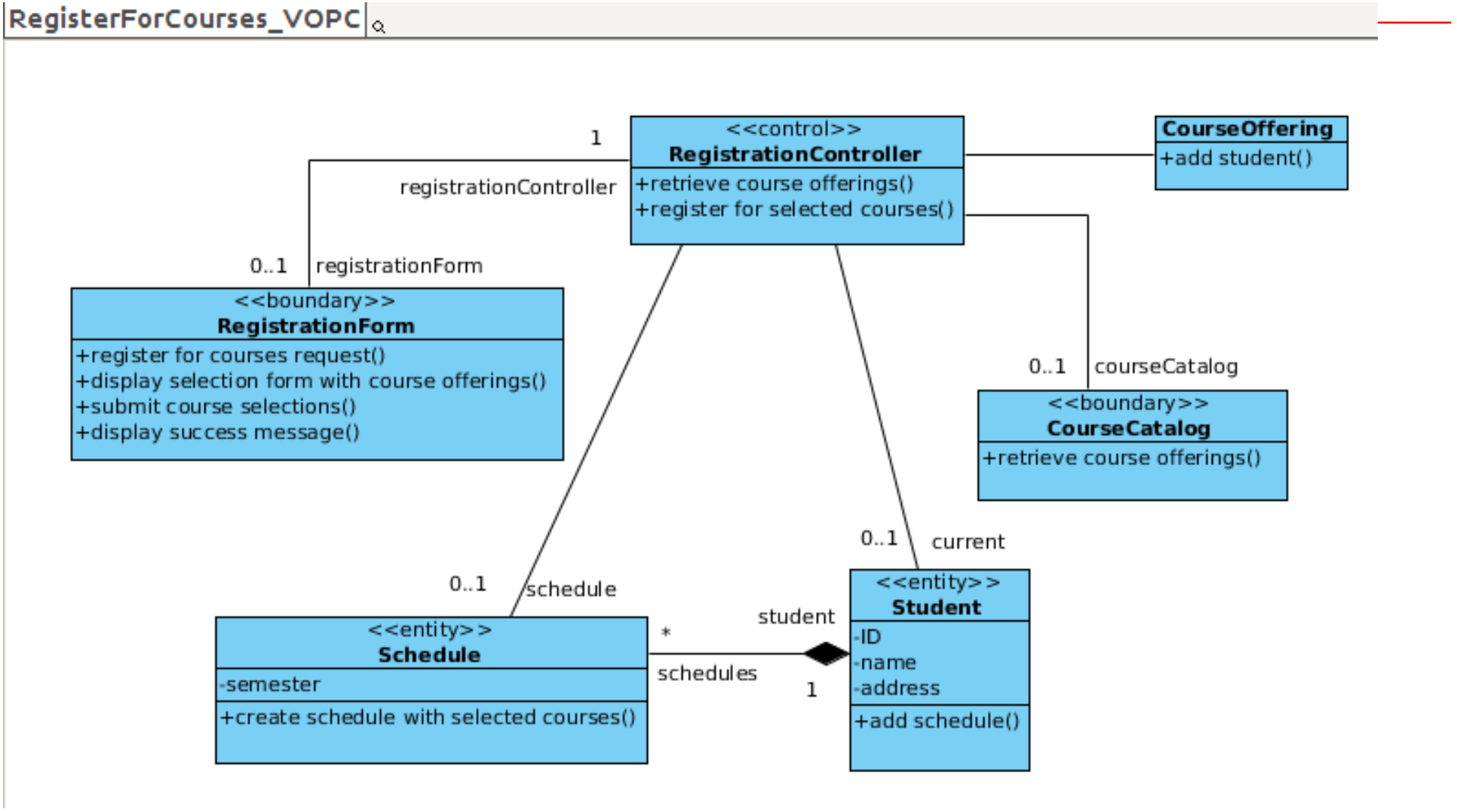
- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

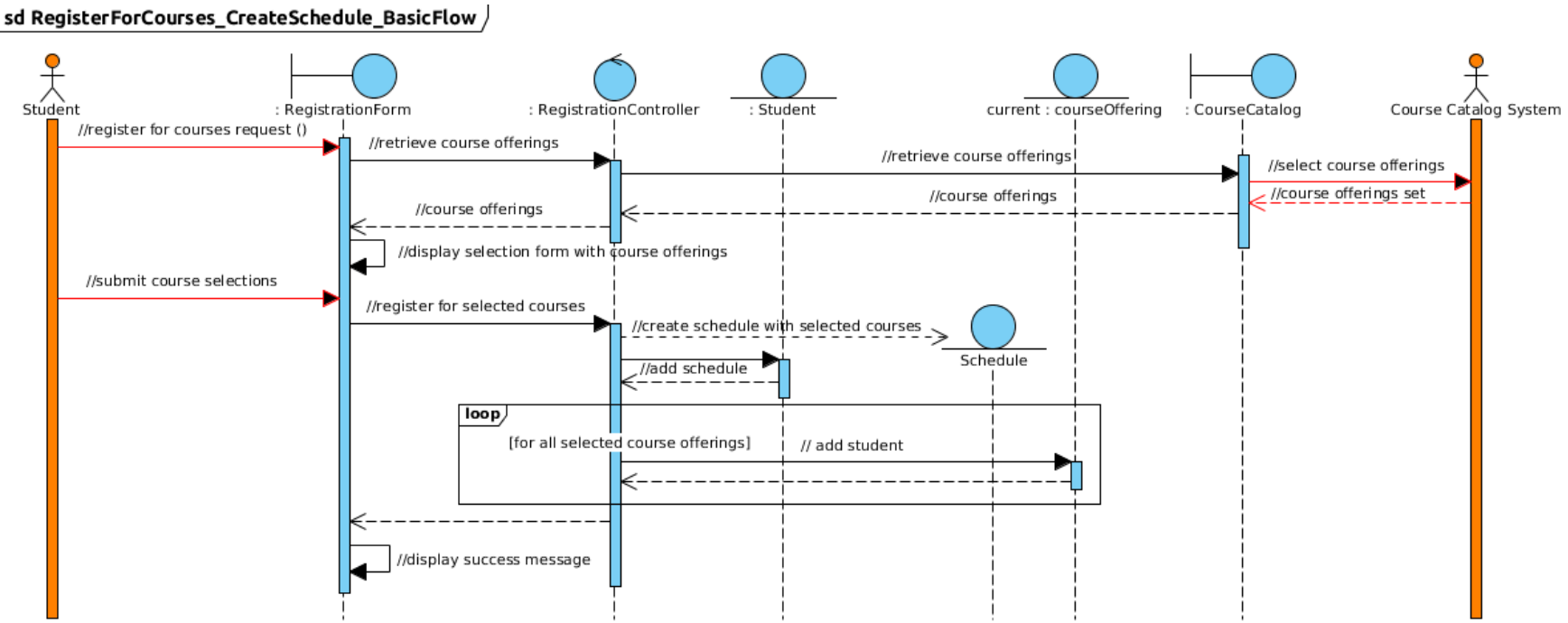
Diagrama de clase VOPC (View Of Participating Classes) conţine clasele ale căror instanţe participă în realizarea cazului de utilizare şi relaţiile necesare pentru a realiza interacţiunile.

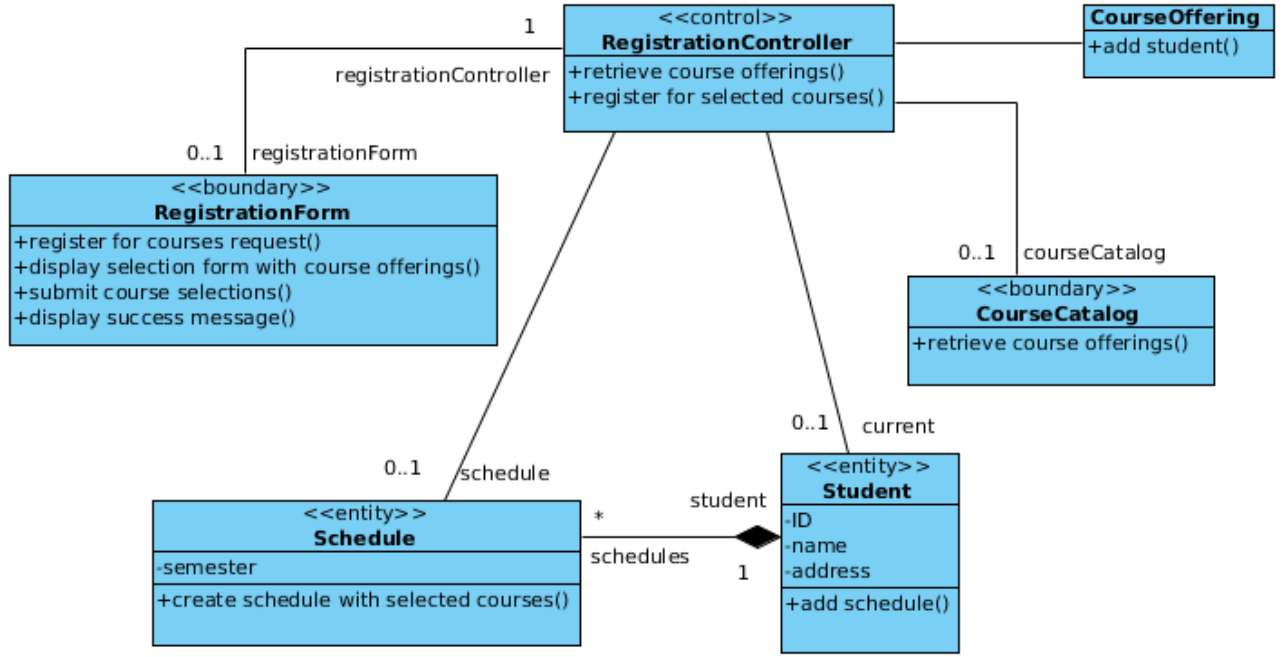


Exemplu:
Course Registration System
UC : Register for Courses

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Modelarea claselor participante cu diagrame de clase
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

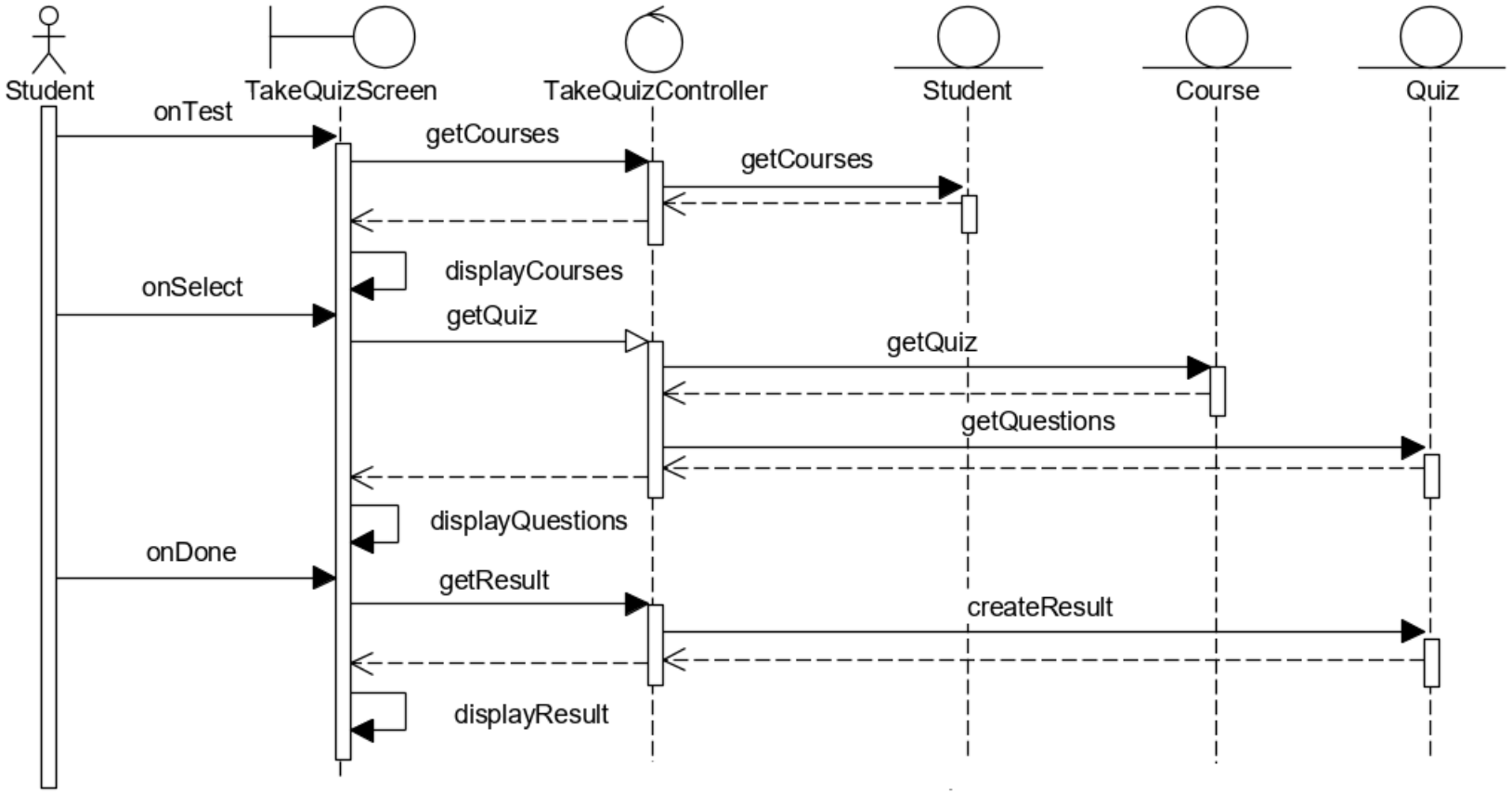



sd RegisterForCourses_CreateSchedule_BasicFlow
Student
: RegistrationForm
: RegistrationController
: Student
current : courseOffering
: CourseCatalog
Course Catalog System
//register for courses request ()
//retrieve course offerings
//retrieve course offerings
//select course offerings
//course offerings
//course offerings set
//course offerings
//display selection form with course offerings
//submit course selections
//register for selected courses
//create schedule with selected courses
Schedule
//add schedule
loop
[for all selected course offerings]
// add student
//display success message


1
<<control>>
RegistrationController
+retrieve course offerings()
+register for selected courses()
registrationController
CourseOffering
+add student()
0..1
registrationForm
<<boundary>>
RegistrationForm
+register for courses request()
+display selection form with course offerings()
+submit course selections()
+display success message()
0..1
courseCatalog
<<boundary>>
CourseCatalog
+retrieve course offerings()
0..1
current
0..1
schedule
<<entity>>
Schedule
-semester
+create schedule with selected courses()
*
schedules
student
1
<<entity>>
Student
-ID
-name
-address
+add schedule()

ANALIZA ARHITECTURALĂ
Evaluare formativă

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



https://forms.gle/tH7A4ymcyxivfRHZ9

RECONCILIEREA REALIZĂRII CAZURILOR DE UTILIZARE

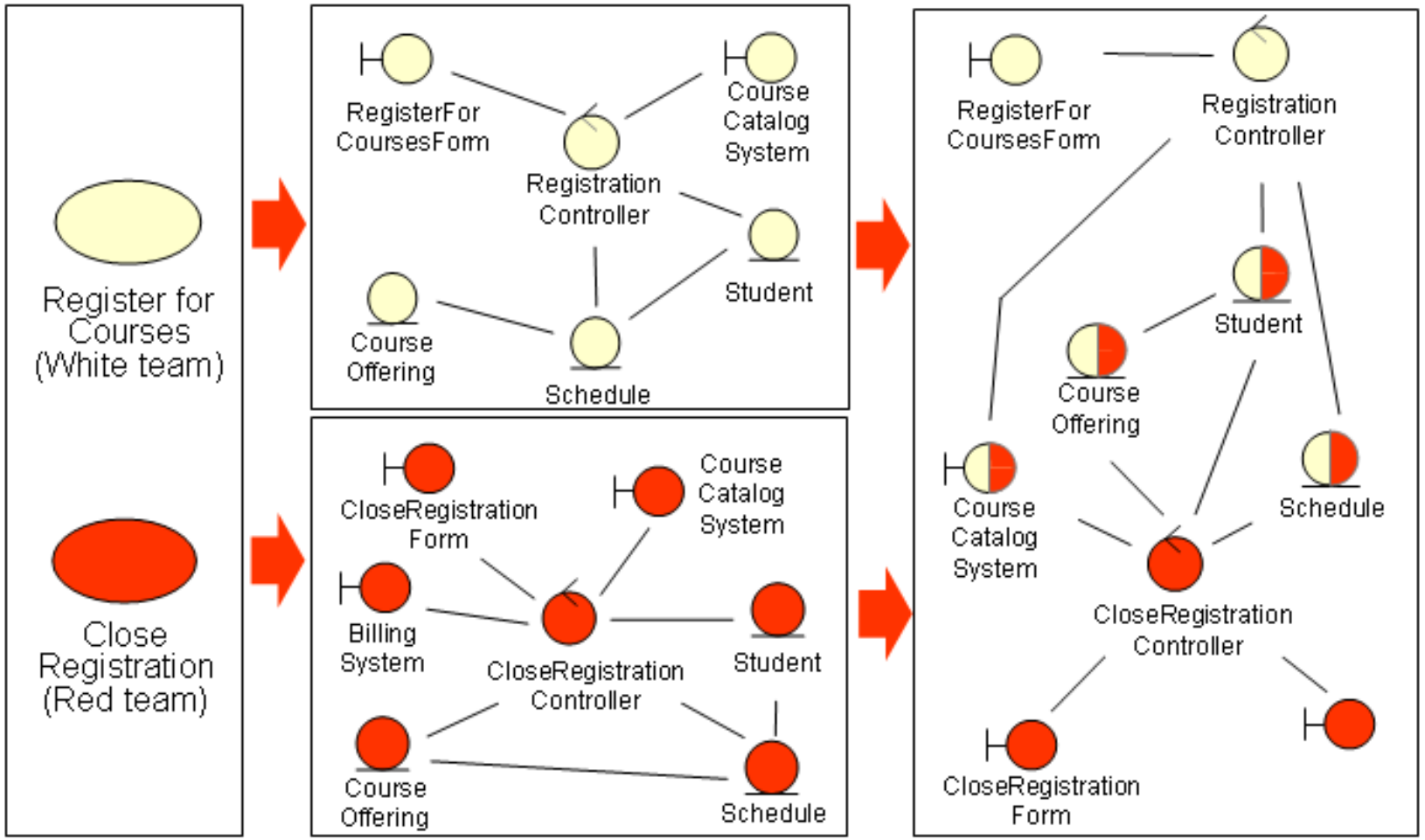
O clasă poate participa la oricâte cazuri de utilizare. De aceea este necesară examinarea consistenţei fiecărei clase în raport cu întregul sistem.

Recomandări:

- Unificarea claselor ce definesc comportamente similare sau reprezintă acelaşi fenomen.
- Unificarea claselor entitate care definesc aceleaşi atribute, chiar dacă au definit comportament diferit; comportamentele vor fi agregate în noua clasă.

EXEMPLU

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză



DESCRIEREA MECANISMELOR DE ANALIZĂ

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

---

1. Colectarea tuturor mecanismelor de analiză într-o listă

Acelaşi mecanism de analiză poate să apară sub diferite nume în realizări ale diferitelor de cazuri de utilizare sau la proiectanţi diferiţi. De exemplu: **memorie**, **persistenţă**, **bază de date** şi **repozitoriu** fac referire la mecanismul de persistenţă. **Comunicare între procese**, **transfer mesaje** sau **invocare la distanţă** se referă la mecanismul de comunicare între procese.

Exemplu :
Course Registration System

- Persistență
- Securitate
- Interfață legacy
- Distribuire

DESCRIEREA MECANISMELOR DE ANALIZĂ

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

1. Colectarea tuturor mecanismelor de analiză într-o listă

2. Realizarea unei corespondenţe între clasele client ale mecanismelor şi mecanismele de analiză

Clasă de analiză	Mecanisme de analiză
Student	Persistență, Securitate
Schedule	Persistență, Securitate
CourseOffering	Persistență, Interfață legacy
Course	Persistență, Interfață legacy
RegistrationController	Distribuire

DESCRIEREA MECANISMELOR DE ANALIZĂ

- Analiza realizării cazurilor de utilizare
- Completarea descrierilor cazurilor de utilizare
- Modelarea scenariilor cazurilor de utilizare cu diagrame de interacţiune
- Reconcilierea realizărilor cazurilor de utilizare
- Calificarea mecanismelor de analiză

1. Colectarea tuturor mecanismelor de analiză într-o listă

2. Realizarea unei corespondenţe între clasele client ale mecanismelor şi mecanismele de analiză

3. Identificarea caracteristicilor mecanismelor de analiză

4. Modelarea utilizând colaborări

Mecanismele identificate şi numite unitar trebuie modelate sub formă de colaborări între clase *ce vor fi definite în procesul de proiectare*.

Unele dintre aceste clase nu oferă direct funcţionalitate a aplicaţiei ci au rolul de a asigura suport pentru aceasta. Deseori aceste clase suport pentru funcţionalitatea aplicaţiei sunt plasate pe nivelele intermediare sau inferioare ale arhitecturii, oferind astfel servicii suport comune pentru toate clasele de la nivelul aplicaţiei.

Obs.

- Nu toate clasele au asociate mecanisme
- E posibil ca o singură clasă să aibă asociate mai multe mecanisme

CONCLUZII
Analiza arhitecturală

Abordare RUP, bazată pe *componente* și *interacțiune* dintre acestea prin *interfețe*.

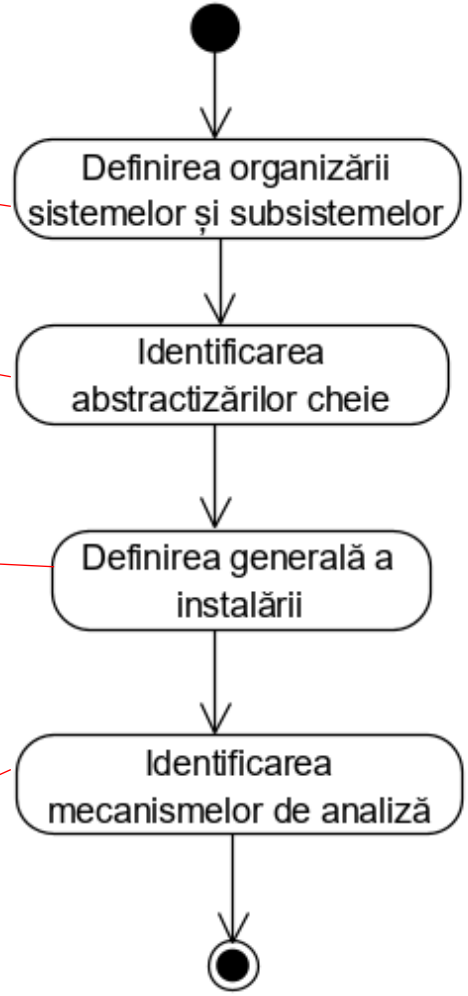
Crearea unei *structuri inițiale* a modelului proiect.
Exemplu : arhitectură multi-nivel.

Abstractizare cheie =concept din domeniul problemei pe care sistemul trebuie să fie capabil să-l manipuleze.
Set *inițial* de clase modelate sub formă de *clase de analiză*.

Schemă generală de instalare a componentelor software pe elementele de infrastructură HW/SW.

Concepte din domeniul IT, de obicei fără legătură cu domeniul problemei.
Exemple : persistență, securitate, distribuire.

Mecanism de analiză → mecanisme de proiectare
Mecanism de proiectare → mecanisme de implementare



CONCLUZII
Analiza cazurilor de utilizare

Abordare RUP, bazată pe *componente* și *interacțiune* dintre acestea prin *interfețe*.

Reprezentare *colaborări* (de obiecte) care *realizează* cazurile de utilizare, în relație de realizare cu cazurile de utilizare respective.

Informaţii suplimentare necesare pentru înţelegerea *comportamentului intern* solicitat sistemului.

Separarea între *interfaţa cu exteriorul* (obiecte <<boundary>>), *informaţiile* utilizate în sistem (obiecte <<entity>>), *logica* sistemului (obiecte <<control>>).
Modelare scenariu de *colaborare între obiecte prin schimb de mesaje*.
Identificare *responsabilități* și *relații*.

Modelarea *claselor* ale căror instanţe participă în realizarea cazului de utilizare şi a *relaţiilor* necesare pentru a realiza interacţiunile.

Crearea diagramei de clase cu *toate clasele de analiză* rezultate din diagramele VOPC și *unificarea responsabilităților și relațiilor* fiecărei clase ce participă la mai multe scenarii.

Asocierea de mecanisme de analiză la (unele) clase și definirea caracteristicilor relevante ale acestora.