

***Curs 4***  
*Analiza cerințelor*

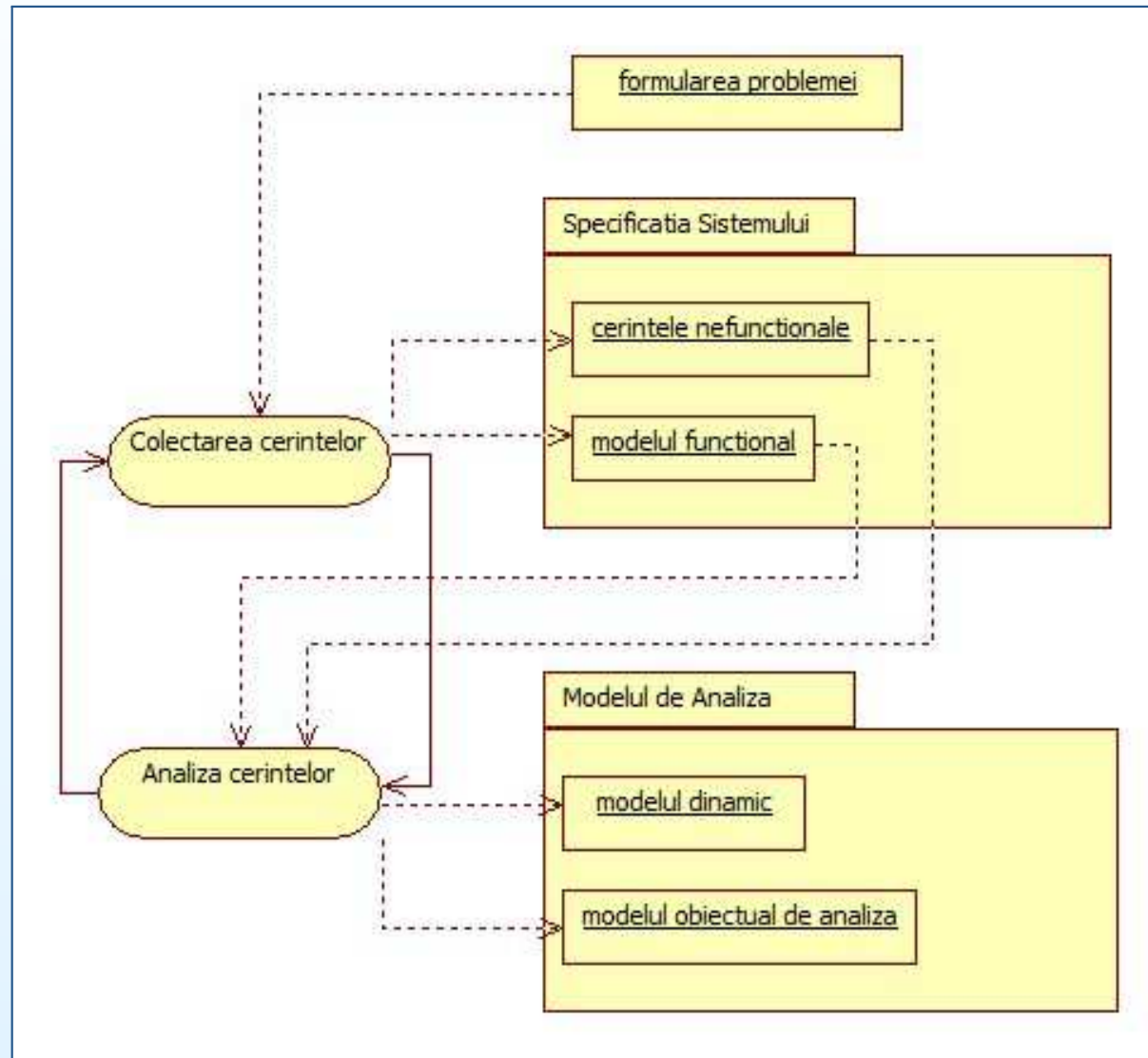
*Suport de curs bazat pe **B. Bruegge and A.H. Dutoit**  
**"Object-Oriented Software Engineering using UML, Patterns, and Java"***

## Sumar Curs 4

---

- Analiza cerințelor - obiective
- Analiza cerințelor - concepte
- Analiza cerințelor - activități tehnice
- Documentul de analiză a cerințelor

# Ingineria cerințelor - activități și modele



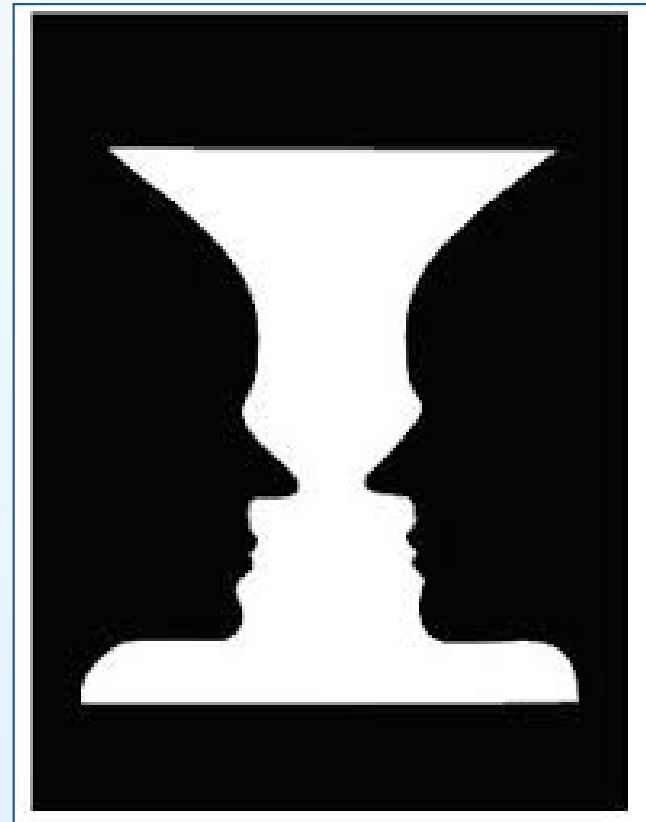
# Analiza cerințelor - obiective

---

- Scopul analizei cerințelor este realizarea unui model al sistemului (*model de analiză*, eng. *analysis model*) corect, complet, consistent și neambiguu
- Accentul este pus pe structurarea și formalizarea cerințelor colectate în etapa anterioară
- Formalizarea permite identificarea de ambiguități, inconsistențe și incompletitudini în descrierea cerințelor, soluționate prin discuții cu clientul/utilizatorii și rezultând în perfectarea specificației sistemului
- Colectarea și analiza cerințelor sunt activități concurente, care se desfășoară iterativ-incremental

## Imagini multistabile

- Ce reprezintă?
- Ambiguitate - dacă ar fi o specificare, ce model am construi?



# Analiza cerințelor - concepte

---

- *Modele obiectuale și modele dinamice*
- *Clase entity, boundary și control*
- *Generalizare / specializare*

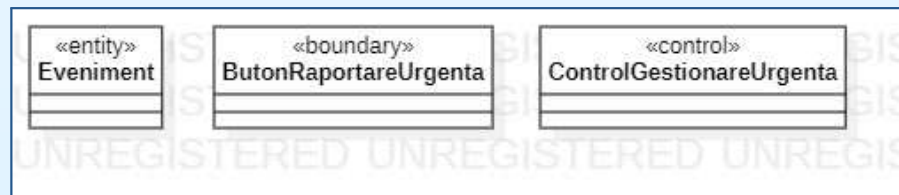
# Modele obiectuale și modele dinamice

---

- *Modelul obiectual de analiză*
  - Surprinde conceptele manipulate de sistem, proprietățile și relațiile acestora
  - Se reprezintă cu ajutorul *diagramei de clase*
  - O clasă din modelul de analiză reprezintă o abstractizare pentru una sau mai multe clase din codul sursă (care, în general, vor conține și un număr mult mai mare de attribute și asocieri)
- *Modelul dinamic*
  - Surprinde comportamentul sistemului
  - Reprezentat cu ajutorul *diagramelor de secvență* și al *diagramelor de tranziție a stărilor*
    - Diagramele de secvență surprind interacțiunile dintr-o mulțime de obiecte, în cadrul unui caz de utilizare
    - O diagrama de tranziție a stărilor surprinde comportamentul unui singur obiect (sau al unui grup de obiecte strâns cuplate)
  - În analiză, modelul dinamic permite identificarea responsabilităților entităților/claselor existente, precum și identificarea de clase/ attribute/ asocieri noi
- **Modelele de analiză surprind doar concepte/ attribute/ relații/ comportamente percepute de utilizatori (*domeniul problemei*)**

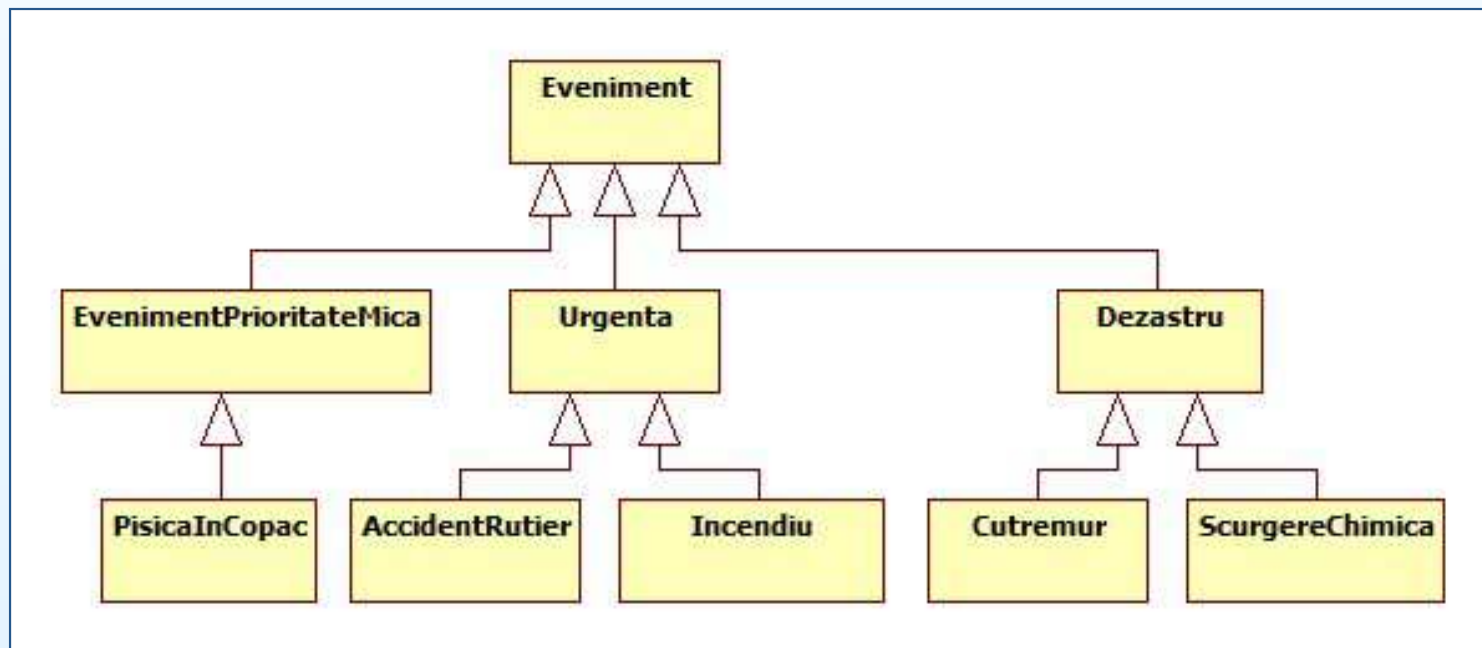
# Clase *entity*, *boundary* și *control*

- Modelul obiectual de analiză constă din clase *entity*, *boundary* și *control*
  - Clase *entity* - responsabile de informația persistentă din sistem
    - Ex.: clasa *RaportUrgență*, clasa *Eveniment*
  - Clase *boundary* - responsabile de interacțiunea actorilor cu sistemul
    - Ex.: clasa *ButonRaportareUrgență*, clasa *FormăRaportareUrgență*
  - Clase *control* - responsabile de realizarea cazurilor de utilizare
    - Ex.: clasa *ControlGestionareUrgență*
- Avantaje ale acestui tip de clasificare
  - Clase / obiecte mai mici, specializate
  - Modele mai ușor de modificat
    - Interfața (obiecte *boundary*) are o probabilitate mai mare de a fi modificată decât funcționalitatea de bază (obiecte *entity* și *control*)
- Notăție: stereotipuri UML + convenții de denumire



# Generalizare / specializare

- Permit organizarea conceptelor în ierarhii
  - *Generalizare* = activitate de modelare având drept scop identificarea unor concepte abstracte, pornind de la unele de nivel jos
  - *Specializare* = activitate de modelare având drept scop identificarea unor concepte specializate, pornind de la unele de nivel înalt





# Analiza cerințelor - activități

---

- Identificarea claselor *entity*
  - Identificarea claselor *boundary*
  - Identificarea claselor *control*
  - Maparea cazurilor de utilizare la obiecte cu diagrame de secvență
  - Identificarea asocierilor
  - Identificarea agregărilor
  - Identificarea atributelor
  - Modelarea comportamentului obiectelor cu diagrame de stări
  - Identificarea ierarhiilor de clase
  - Revizuirea modelului de analiză
- 
- Aceste activități sunt ghidate de euristici
  - Calitatea produselor lor depinde de experiența dezvoltatorilor în aplicarea euristicilor/metodelor aferente

## Cazul de utilizare (CU) *RaporteazăUrgență* a SGA

<b>Nume</b>	<i>RaporteazăUrgență</i>
<b>Participanți</b>	Inițiat de <i>OfițerTeren</i> Comunică cu <i>Dispecerul</i>
<b>Flux de evenimente</b>	<ol style="list-style-type: none"><li>1. <i>Ofițerul</i> activează funcția <i>Raportează urgență</i> a terminalului său.</li><li>2. Sistemul SGA afișează un formular <i>Ofițerului</i>. Formularul include componente privind tipul urgenței (general, incendiu, accident auto), nivelul de alertă, locația, descrierea și resursele solicitate.</li><li>3. <i>Ofițerul</i> completează formularul, inserând cel puțin tipul urgenței și descrierea situației. El poate descrie și soluții viabile la situația de urgență și poate solicita resurse specifice. După completare, <i>Ofițerul</i> trimite formularul.</li><li>4. Sistemul preia informația din formular și notifică <i>Dispecerul</i>.</li><li>5. <i>Dispecerul</i> verifică informația primită și creează un nou <i>Eveniment</i> în baza de date prin invocarea cazului de utilizare <i>DeschideCazNou</i>. Toate informațiile din formularul primit sunt asociate automat evenimentului creat. <i>Dispecerul</i> alege un răspuns prin alocarea de resurse la eveniment (prin cazul de utilizare <i>AlocareResurse</i>) și confirmă primirea formularului printr-un mesaj către ofițer.</li><li>6. <i>Ofițerul</i> primește confirmarea și răspunsul ales.</li></ol>

...

## Identificarea claselor *entity*

- Procesul de identificare pornește de la descrierea cazurilor de utilizare
- *Analiza limbajului natural* (eng. *natural language analysis* [Abbot 1983]) oferă un set util de euristici pentru identificarea claselor/obiectelor, atributelor, operațiilor, relațiilor și constrângerilor
  - Părților de vorbire le corespund tipuri de elemente din model

Parte de vorbire	Componentă din model	Exemple
Substantiv propriu	Obiect	Alice
Substantiv comun	Clasă sau atribut	Ofițer din teren Descrierea evenimentului
Verb "a face"	Operație sau asociere	Creează, trimite
Verb "a fi"	Moștenire	Este un/o
Verb "a avea"	Agregare	Are, constă din
Verb modal	Constrângere	Trebuie să fie

## Identificarea claselor *entity* (cont.)

---

- Avantaje
  - Metodă focusată pe terminologia utilizatorilor
  - Rezultate bune atunci când se dorește identificarea claselor candidat pe baza unor descrieri scurte (fluxul unui scenariu sau al unui caz de utilizare)
- Dezavantaje
  - Calitatea modelului obiectual e dependentă de stilul de specificare al analistului (claritate, consecvență în utilizarea termenilor)
  - Numărul substantivelor este, de regulă, mai mare decât cel al claselor (incluzând sinonime sau attribute ale altor clase)
- Euristici suplimentare pentru identificarea claselor *entity*
  - Termeni pe care dezvoltatorii și utilizatorii trebuie să îi clarifice pentru a înțelege cazul de utilizare
  - Substantive care se repetă în descrierea cazului de utilizare (ex: *Eveniment*)
  - Entități din lumea reală pe care sistemul trebuie să le gestioneze (ex.: *Dispecer*, *Resursă*)
  - Activități din lumea reală pe care sistemul trebuie să le gestioneze (ex.: *PlanDeOperațiiUrgență*)

## CU RaporteazăUrgență - clase entity

Nume clasă	Descriere
<i>OfițerTeren</i>	Ofițer de poliție sau pompieri la datorie. Un <i>OfițerTeren</i> este identificat prin număr de ecuson și nu poate fi alocat la două <i>Evenimente</i> simultan.
<i>RaportUrgență</i>	Raport inițial legat de un <i>Eveniment</i> , trimis de un <i>OfițerTeren</i> unui <i>Dispecer</i> . Primirea unui <i>RaportUrgență</i> determină, de obicei, crearea unui <i>Eveniment</i> de către <i>Dispecer</i> . Un <i>RaportUrgență</i> conține un nivel de urgență, un tip (incendiu, accident rutier, etc.), o locație și o descriere.
<i>Dispecer</i>	Ofițer de poliție care gestionează <i>Evenimentele</i> . Un <i>Dispecer</i> deschide, documentează și închide un <i>Eveniment</i> , ca și răspuns al unui <i>RaportUrgență</i> sau al altui tip de comunicare cu un <i>OfițerTeren</i> . Un <i>Dispecer</i> este identificat prin număr de ecuson.
<i>Eveniment</i>	Situație care necesită intervenția unui <i>OfițerTeren</i> . Un <i>Eveniment</i> poate fi raportat în sistem de către un <i>OfițerTeren</i> , sau de către orice altă entitate externă. Un <i>Eveniment</i> constă dintr-o descriere, un răspuns, o stare (deschis, închis, documentat), o locație și un număr de <i>OfițeriTeren</i> implicați.

# Identificarea claselor *boundary*

---

- Clasele *boundary* reprezintă interfața sistemului cu actorii
  - În cadrul fiecărui caz de utilizare, fiecare actor interacționează cu cel puțin un obiect de tip *boundary*
  - Obiectele *boundary* colectează inputul actorilor și îl transformă într-o formă utilizabilă de către obiectele *entity* și *control*
- Euristici de determinare a claselor *boundary*
  - Identificarea controalelor de care utilizatorii au nevoie pentru a iniția un caz de utilizare (ex.: *ButonRaportareUrgență*)
  - Identificarea formelor de care utilizatorii au nevoie pentru a introduce date în sistem (ex.: *FormaRaportUrgență*)
  - Identificarea notificărilor și mesajelor folosite de către sistem, pentru a răspunde actorilor (ex.: *NotificareConfirmare*)
  - În situația în care un caz de utilizare implică mai mulți actori, identificarea terminalelor acestora, pentru a referi interfața utilizator aferentă (ex.: *StatiaDispecer*)
  - Clasele *boundary* reprezintă elemente de interfață de granularitate mare
  - Elementele de interfață trebuie descrise folosind exclusiv vocabularul utilizatorilor

## CU *RaporteazăUrgență* - clase *boundary*

Nume clasă	Descriere
<i>ButonRaportareUrgență</i>	Buton utilizat de <i>OfițerTeren</i> pentru a iniția cazul de utilizare aferent.
<i>FormăRaportUrgență</i>	Formă deschisă unui <i>OfițerTeren</i> pe o <i>StațieOfițerTeren</i> în momentul selectării funcției <i>RaporteazăUrgență</i> . Forma conține câmpuri aferente tuturor atributelor unui raport de urgență și un buton pentru a fi trimisă.
<i>FormăEveniment</i>	Formă utilizată pentru crearea unui <i>Eveniment</i> . Este prezentată unui <i>Dispecer</i> , pe o <i>StațieDispecer</i> , în momentul primirii unui <i>RaportUrgență</i> . <i>Dispecerul</i> o utilizează și pentru alocarea resurselor și trimiterea confirmării către <i>OfițerTeren</i> .
<i>NotificareConfirmare</i>	Notificare utilizată pentru a afișa confirmarea <i>Dispecerului</i> către <i>OfițerulTeren</i>
<i>StațieDispecer</i>	Calculator utilizat de către <i>Dispecer</i>
<i>StațieOfițerTeren</i>	Calculator portabil utilizat de către <i>OfițerTeren</i>

# Identificarea claselor *control*

---

- Obiectele *control* sunt responsabile de coordonarea obiectelor *boundary* și *entity*
  - Obiectele *control* nu au, de obicei, un corespondent în lumea reală
  - Un obiect *control* este asociat, de regulă, unui caz de utilizare: este creat la începutul cazului de utilizare și distrus la finalizarea acestuia.
  - Ex.: Obiectele *control* gestionează comportamentul legat de succesiunea formelor, cozi istoric/ undo, transmiterea informației într-un mediu distribuit, etc.
- Euristici pentru determinarea obiectelor *control*
  - Identificarea unui control per caz de utilizare
  - Identificarea unui control per actor într-un caz de utilizare
  - Durata de viață a obiectului *control* corespunde duratei de execuție a cazului de utilizare sau duratei unei sesiuni utilizator



## CU *RaporteazăUrgență* - clase control

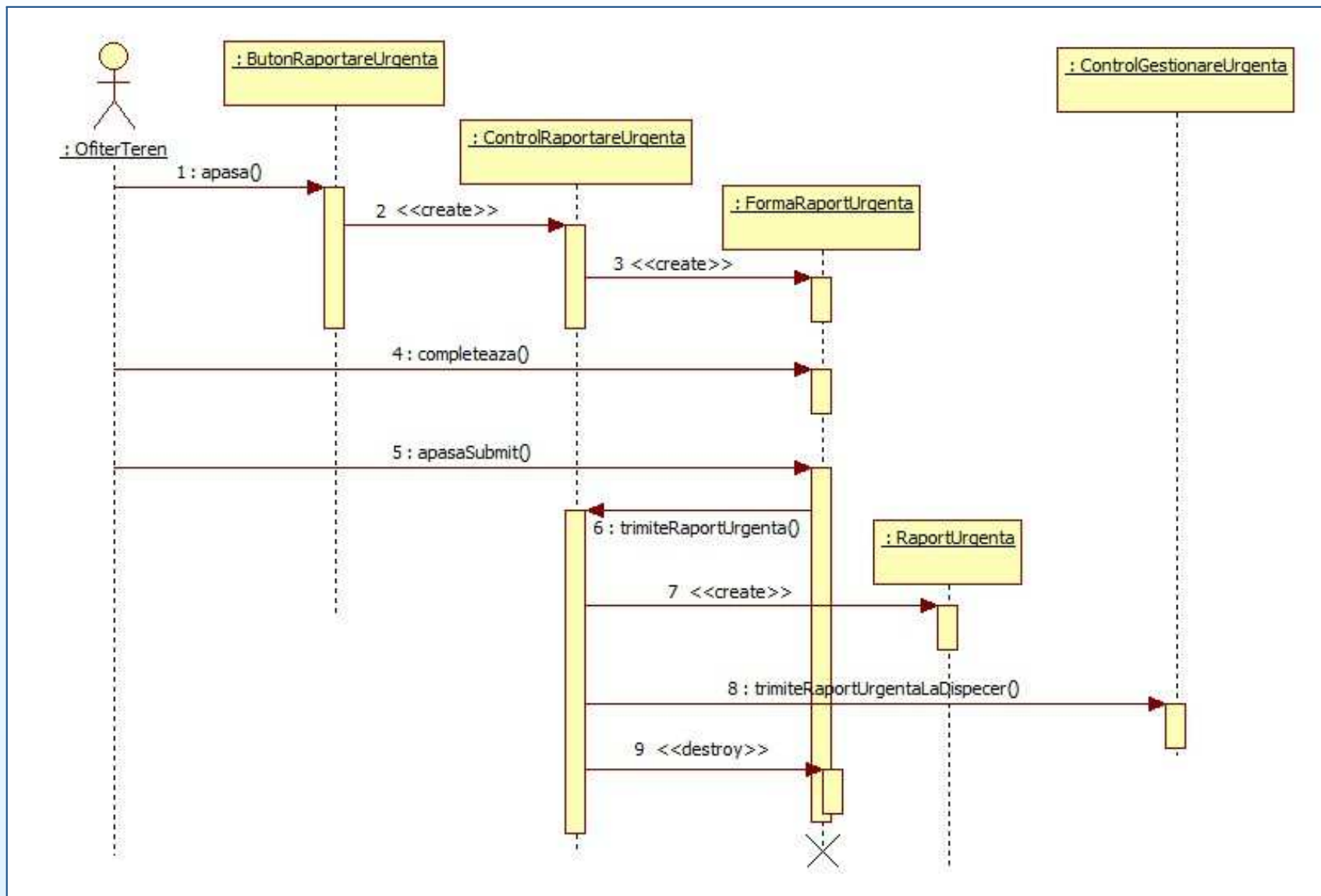
Nume clasă	Descriere
<i>ControlRaportareUrgență</i>	Gestionează raportarea unei urgențe de pe o <i>StațieOfițerTeren</i> . Un obiect de acest tip este creat în momentul în care un <i>OfițerTeren</i> apasă butonul "Raportează Urgență". Ulterior, acesta creează un obiect de tip <i>FormăRaportUrgență</i> , pe care o afișează ofițerului. La trimiterea formularului, acest obiect colectează informația completată, creează un obiect de tip <i>RaportUrgență</i> și îl trimite <i>Dispecerului</i> . Obiectul control așteaptă apoi o confirmare de la stația dispecerului. În momentul primirii confirmării, creează un obiect de tip <i>NotificareConfirmare</i> și o afișează ofițerului.
<i>ControlGestionareUrgență</i>	Gestionează raportarea unei urgențe pe o <i>StațieDispecer</i> . Un obiect de acest tip este creat în momentul primirii unui <i>RaportUrgență</i> . Ulterior, acesta creează o <i>FormăEveniment</i> și o afișează <i>Dispecerului</i> . După ce dispecerul creează un <i>Eveniment</i> , alocă resurse și trimite o confirmare, obiectul control transmite confirmarea <i>OfițeruluiTeren</i> .

# Realizarea diagramelor de secvență

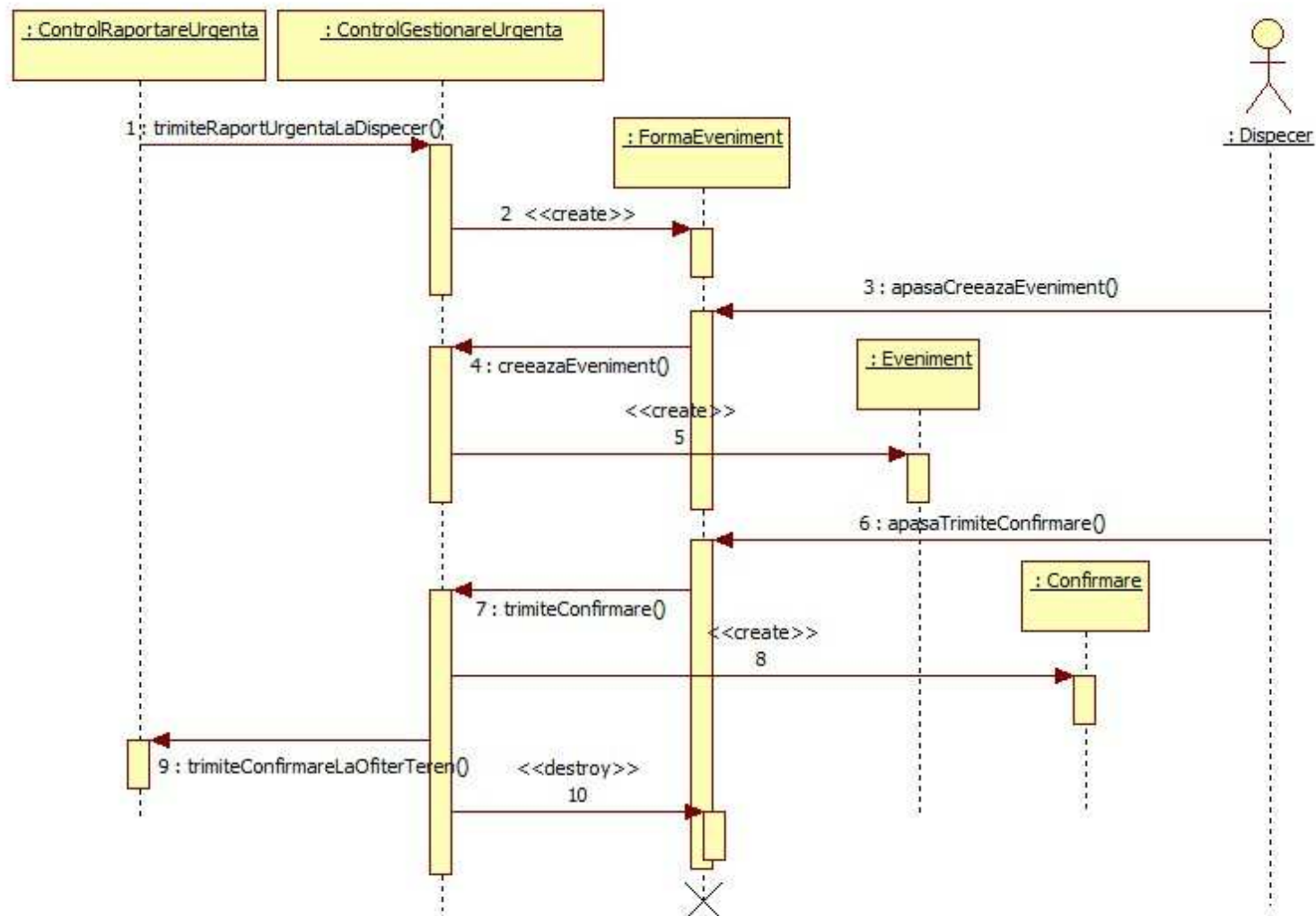
---

- Diagramele de secvență realizează legătura dintre clase/ obiecte și cazurile de utilizare/ scenariile descrise
  - Prezintă modul în care comportamentul descris de un scenariu/ caz de utilizare este distribuit între obiectele participante
  - În etapa de analiză, permit identificarea unor descrieri de comportament ambigue sau a unor clase/ obiecte participante omise
  - Datorită notației utilizate, nu sunt, în general, o metodă de comunicare cu clienții/ utilizatorii la fel de eficientă ca și scenariile/ cazurile de utilizare
- Euristici de realizare a diagramelor de secvență de analiză
  - Prima coloană trebuie să corespundă actorului care a inițiat cazul de utilizare
  - A doua coloană trebuie să corespundă obiectului *boundary* folosit de actor pentru a iniția cazul de utilizare
  - A treia coloană trebuie să corespundă obiectului *control* care gestionează cazul de utilizare
  - Obiectele *control* sunt create de către obiectele *boundary* care inițiază cazurile de utilizare
  - Ulterior, obiectele *boundary* sunt create de către obiecte *control*
  - Obiectele *entity* sunt accesate de către obiectele *boundary* și *control*, dar nu accesează niciodată obiecte *boundary* sau *control*

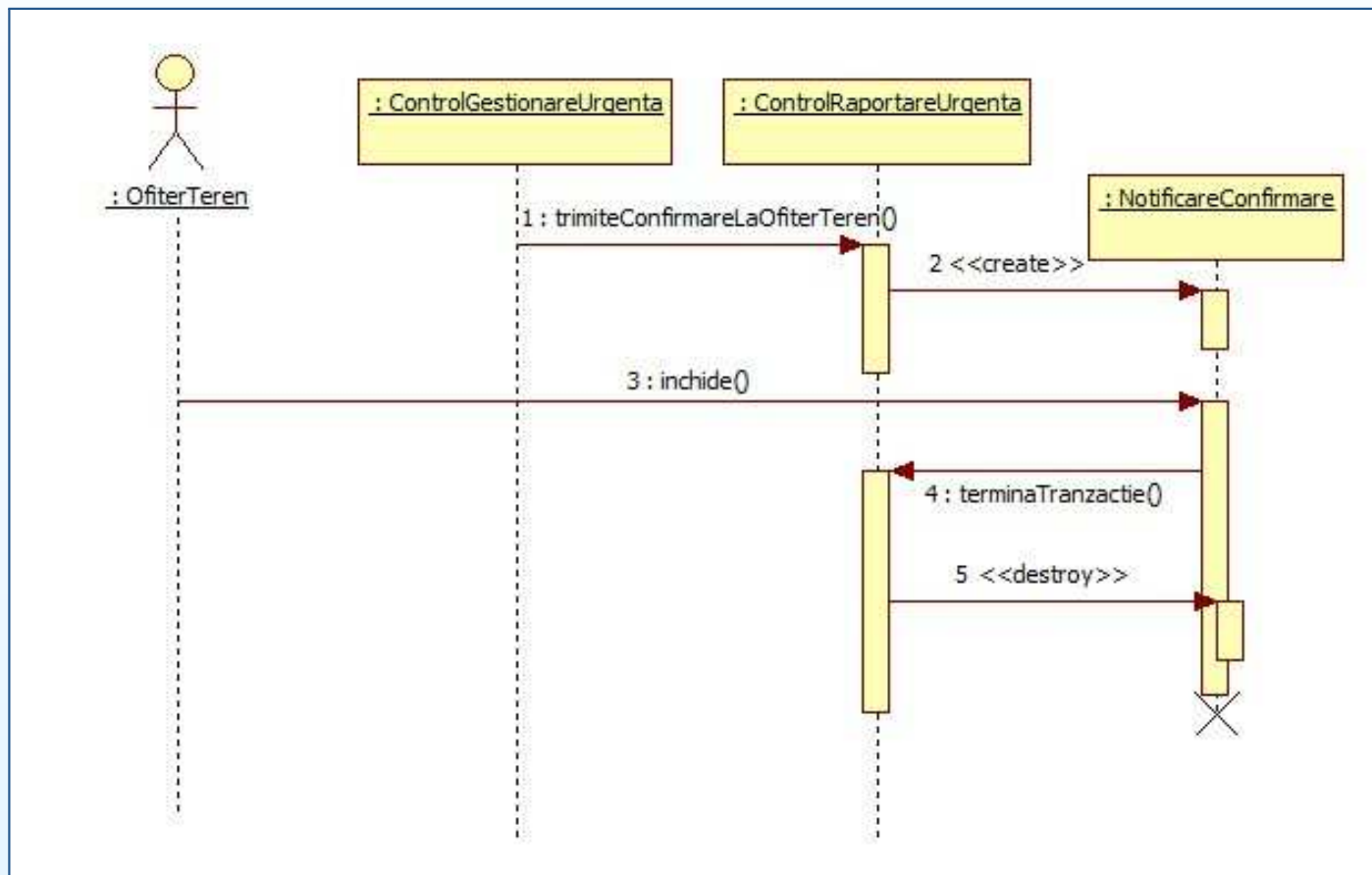
## CU *RaporteazăUrgență* - diagramă de secvență



## CU RaporteazăUrgență - diagramă de secvență (cont.)



## CU *RaporteazăUrgență* - diagramă de secvență (cont.)



## CU *RaporteazăUrgență* - rafinare

- Clasă *entity* nouă, identificată ca urmare a formalizării descrierii cazului de utilizare folosind diagrama de secvență

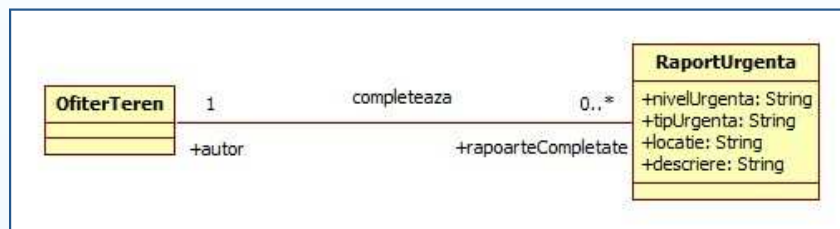
Nume clasă	Descriere
<i>Confirmare</i>	Răspuns al unui <i>Dispecer</i> la un <i>RaportUrgență</i> trimis de către un <i>OfițerTeren</i> . Trimițând o <i>Confirmare</i> , <i>Dispecerul</i> îi comunică ofițerului că a primit raportul, a creat un <i>Eveniment</i> și i-a asignat resurse. <i>Confirmarea</i> include specificarea resurselor alocate și timpul estimat al sosirii lor.

- Rafinarea descrierii cazului de utilizare

Nume	<i>RaporteazăUrgență</i>
...	
Flux de evenimente	<p>...</p> <p>5. ... <i>Confirmarea</i> îi indică ofițerului că raportul a fost primit, evenimentul a fost creat și resursele au fost alocate. <i>Confirmarea</i> include specificarea resurselor (ex. camion pompieri) și momentul estimativ al sosirii lor.</p> <p>...</p>
...	

# Identificarea asocierilor

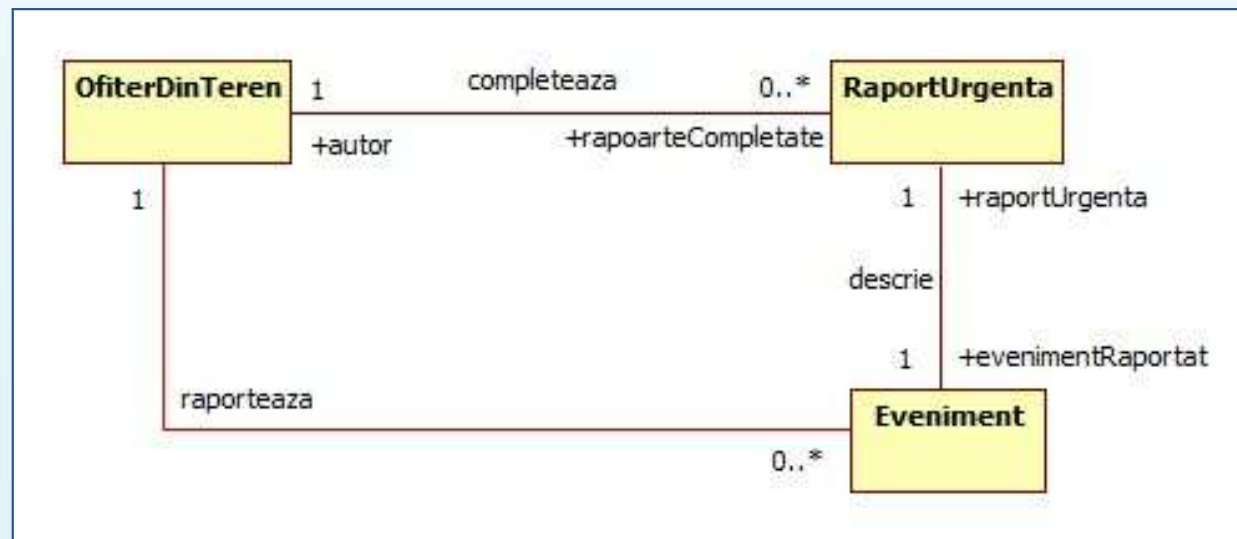
- O *asociere* indică o relație între două sau mai multe clase
  - Ex.: Un ofițer din teren scrie rapoarte de urgență.



- Proprietăți ale unei asocieri
  - nume
  - roluri
  - multiplicități
- Avantaje ale modelării asocierilor în analiză
  - Clarificarea modelului de analiză prin explicitarea modului de relaționare a obiectelor
    - Raportul este scris de către un ofițer din teren și nu de către un dispecer
  - Investigarea cazurilor limită
    - Există rapoarte cu mai mulți autori? Dar anonime?

## Identificarea asocierilor (cont.)

- Euristici pentru reprezentarea asocierilor
  - Examinarea structurilor verbale
  - Numirea explicită a asocierilor și rolurilor
  - Utilizarea calificativelor pentru a identifica spații de nume și attribute cheie
  - Eliminarea tuturor asocierilor redundante (derivabile din alte asocieri)
  - Reprezentarea multiplicităților doar după stabilizarea mulțimii de asocieri
  - Un număr exagerat de asocieri generează redundanță și afectează negativ inteligibilitatea modelului
- Model cu asocieri redundante



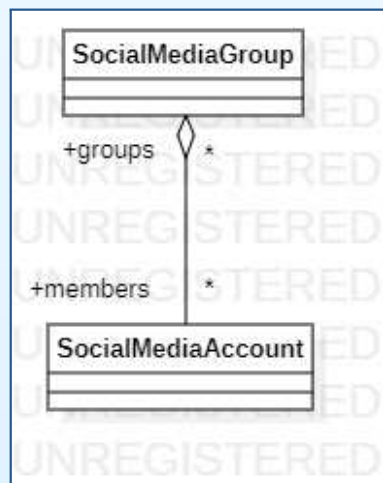


# Identificarea agregărilor

- *Agregarea* (variantea "simplă"/ partajată) reprezintă un tip particular de asociere, care denotă o relație de tip parte-întreg
- *Compunerea* este o agregare mai "puternică", în care partea este, la un moment dat, conținută într-un singur întreg, iar existența părților e condiționată de întreg

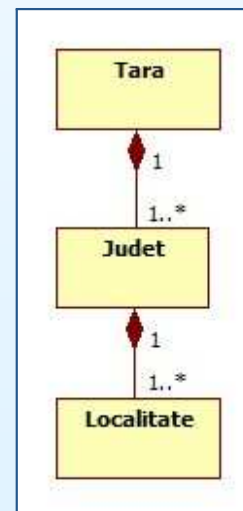
Ex.: Agregare

- Pe rețelele de socializare, o persoană este, în general, înscrisă în mai multe grupuri. După eventuala ștergere a unui astfel de grup, ea rămâne parte a celorlalte.



Ex.: Compunere

- O localitate e parte a unui singur județ, cel din urmă fiind parte a unei singure țări la un moment dat (dar există posibilitatea schimbării granițelor politice)



# Identificarea atributelor

- *Atributele* reprezintă proprietăți ale obiectelor individuale



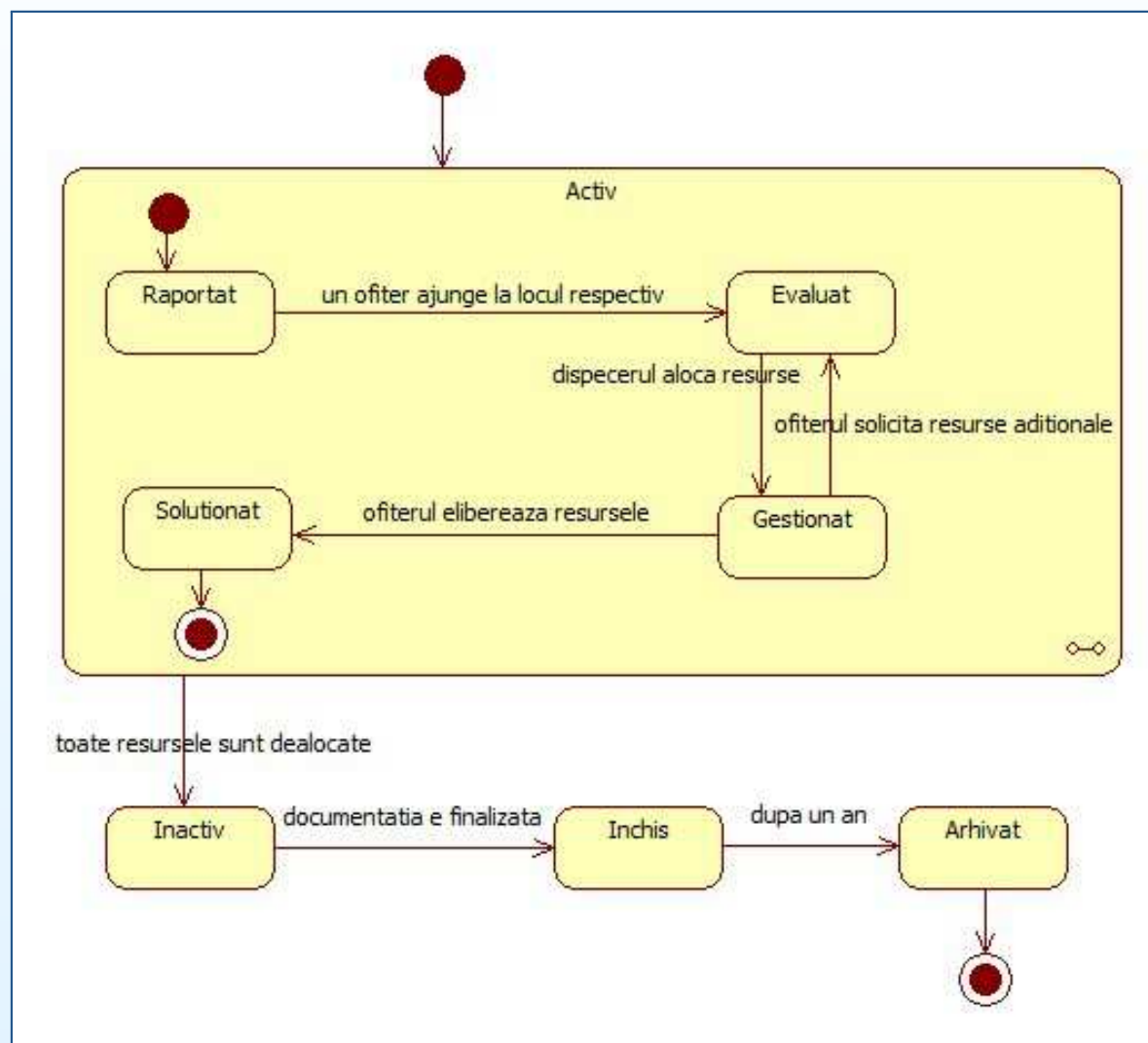
- Euristici pentru identificarea atributelor
  - Examinarea construcțiilor substantive posesive (ex.: descrierea urgenței)
  - Reprezentarea proprietăților persistente ca și attribute
  - Identificarea doar a acelor attribute relevante pentru sistemul în cauză
  - Amânarea identificării atributelor mai puțin relevante din punct de vedere al funcționalității până după stabilizarea modelului obiectual
  - Descrierea sumară a fiecărui atribut
  - Proprietățile de tip obiect/ referință nu se reprezintă ca și attribute, în acest caz se folosesc asocieri!

# Modelarea comportamentului cu diagrame de stări

---

- Modelarea comportamentului
  - Diagrame de secvență - prezintă comportamentul din perspectiva unui singur caz de utilizare
  - Diagrame de tranziție a stărilor - prezintă comportamentul din perspectiva unui singur obiect
- Rolul diagramelor de tranziție a stărilor în etapa de analiză constă în identificarea unor cazuri de utilizare omise sau a unor omisiuni în descrierea cazurilor de utilizare existente
- Nu este necesară construirea unei diagrame de tranziție a stărilor pentru fiecare clasă a modelului obiectual, doar pentru cele ale căror obiecte au durata de viață mare și comportament complex, dependent de stare
  - Cel mai frecvent pentru obiecte *control*, mai puțin frecvent pentru obiecte *entity* și aproape niciodată pentru obiecte *boundary*

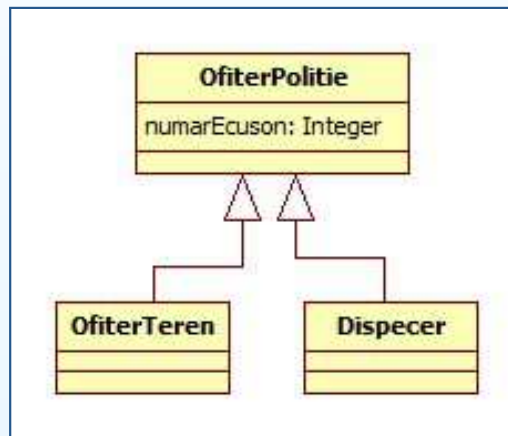
## CU RaporteazăUrgență - diagrama de stări pentru *Eveniment*



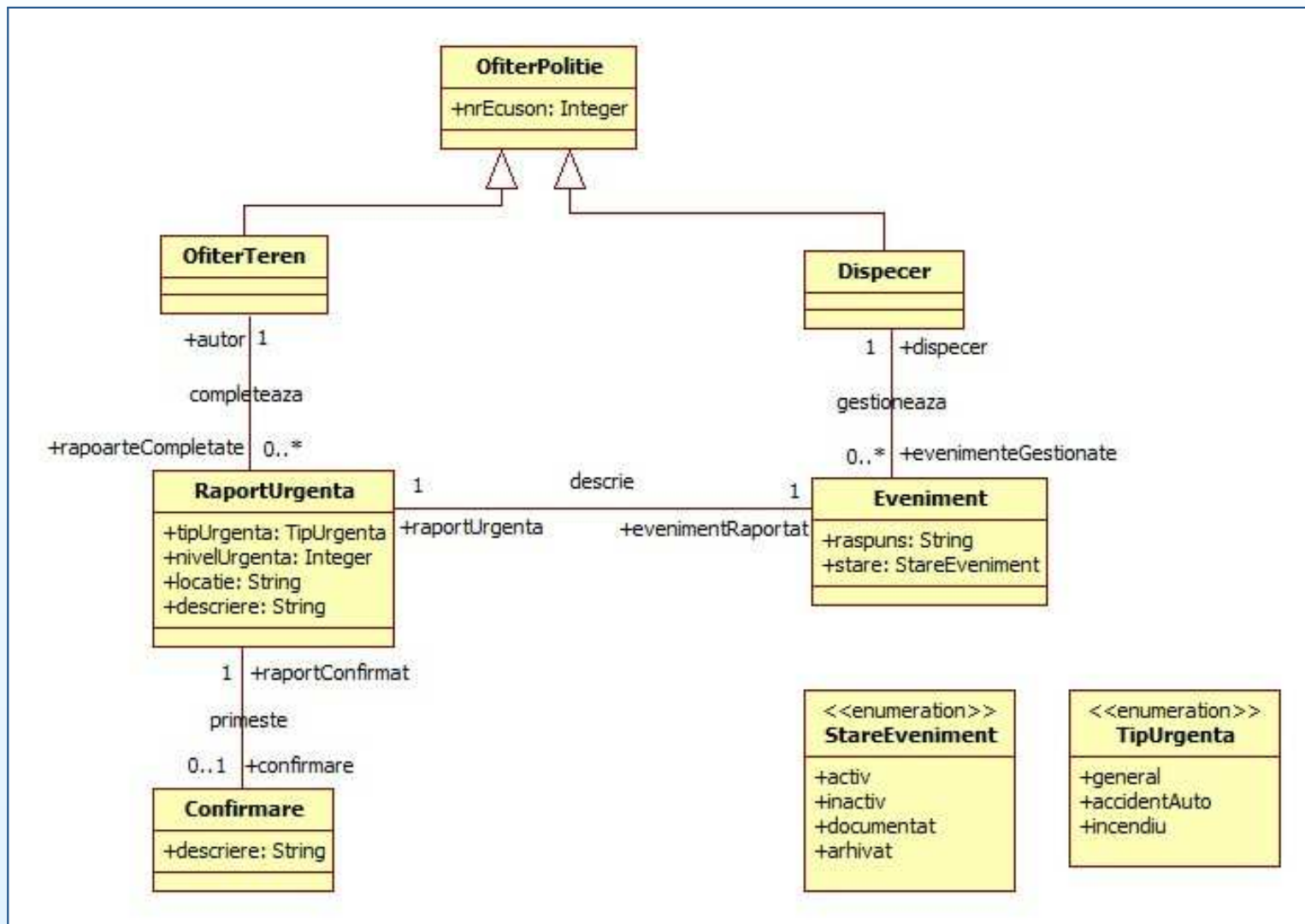
# Identificarea ierarhiilor de clase

---

- Rolul generalizărilor îl constituie eliminarea redundanțelor din modelul obiectual
- Ex.:



## CU RaporteazăUrgență - model al claselor *entity* (conceptual)



# Revizuirea modelului de analiză

---

- Modelul de analiză se dezvoltă iterativ-incremental
- Odata ce devine stabil, urmează revizuirea acestuia
  - Revizuire internă (dezvoltatori) + revizuire dezvoltatori și client
  - E specificarea cerințelor corectă, completă, consistentă și neambiguă? Sunt cerințele realiste și verificabile?
  - Revizuirea e facilitată de existența unor liste de întrebări posibile
- Asigurarea *corectitudinii*
  - Este dicționarul claselor *entity* înțeles de către client?
  - Corespund clasele abstracte unor concepte utilizator?
  - Sunt toate descrierile conforme cu definițiile date de utilizatori?
  - Au toate obiectele *entity* și *boundary* ca și nume construcții substantive sugestive?
  - Au toate cazurile de utilizare și obiectele *control* ca și nume construcții verbale sugestive?
  - Sunt toate cazurile de eroare descrise și gestionate?

## Revizuirea modelului de analiză (cont.)

---

- Asigurarea *completitudinii*
  - Pentru fiecare obiect: Este folosit de vreun caz de utilizare? În cadrul cărui caz de utilizare este creat? modificat? distrus? Poate fi accesat pornind de la un obiect *boundary*?
  - Pentru fiecare atribut: Unde ii este atribuită o valoare? Care este tipul său? Poate reprezenta un calificator?
  - Pentru fiecare asociere: Când este traversată? De ce a fost aleasă respectiva multiplicitate?
  - Pot fi calificate asocierile *one-to-many* și *many-to-many*?
  - Pentru fiecare obiect *control*: Are asocierile necesare accesării tuturor obiectelor din respectivul caz de utilizare?
- Asigurarea *consistenței*
  - Există mai multe clase sau cazuri de utilizare cu același nume?
  - Denotă entitățile cu nume similare concepte înrudite?
  - Există obiecte cu atribute și asocieri similare și care nu fac parte din aceeași ierarhie de moștenire?



## Revizuirea modelului de analiză (cont.)

---

- Asigurarea *realismului*
  - Pot fi îndeplinite cerințele legate de performanță și fiabilitate?
  - Au fost aceste cerințe verificate folosind prototipuri pe platforma hardware aleasă?

# Șablonul documentului de analiză a cerințelor

---

- 1. Introducere
  - 1.1 Scopul sistemului
  - 1.2 Obiectivele și criteriile de succes ale proiectului
  - 1.3 Definiții, acronime și abrevieri
  - 1.4 Referințe
  - 1.5 Sumar
- 2. Sistemul curent
- 3. Sistemul propus
  - 3.1 Sumar
  - 3.2 Cerințe funcționale
  - 3.3 Cerințe nefuncționale
    - 3.3.1 Utilizabilitate
    - 3.3.2 Fiabilitate
    - 3.3.3 Performanță
    - 3.3.4 Suportabilitate
    - 3.3.5 Implementare
    - 3.3.6 Interfață

# Șablonul documentului de analiză a cerințelor (cont.)

---

- 3.3.7 Instalare
- 3.3.8 Cerințe legale
- 3.4 Modele
  - 3.4.1 Scenarii
  - 3.4.2 Modelul cazurilor de utilizare
  - 3.4.3 Modelul obiectual
  - 3.4.4 Modelul dinamic
  - 3.4.5 Prototipul interfeței utilizator
- 4. Glosar

## Referințe

---

- [Abbott, 1983] R. Abbott, *Program design by informal English descriptions*, Communications of the ACM, Vol. 26, No. 11, 1983.