



Introducere

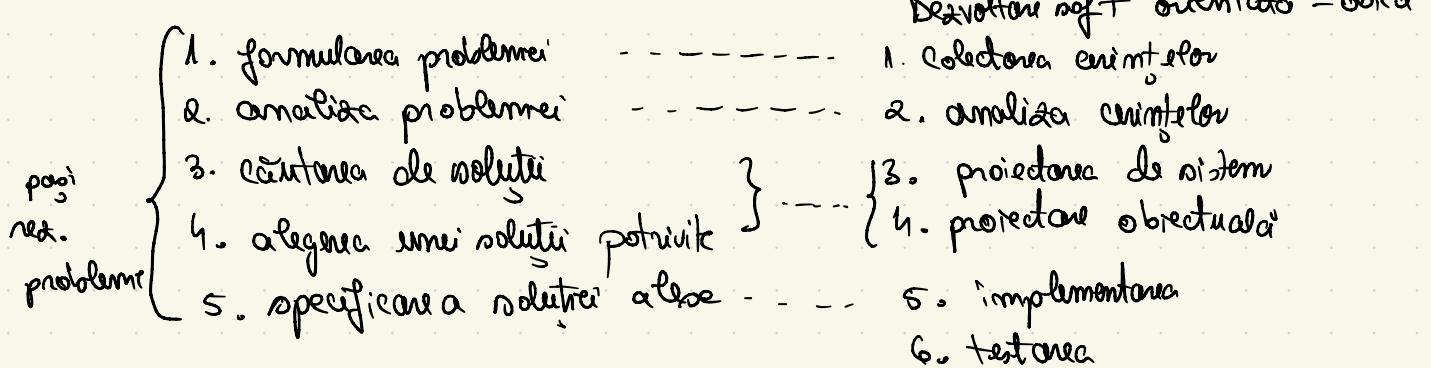
- Ingineria programării = colecție de tehnici, metode și instrumente care ajută la producție unor sisteme soft

Ce preocupe ISS

- modelare

- o reprezentare abstractă a unui sistem
- procesul de reprezentare a elementelor unui sistem

- rezolvare de probleme



- acumulare (recv.) de cunoștințe

- o informatică nouă apărută poate invalida toate modelele anterioare dezvoltate
 - bucket theory of the mind = mintea conținează într-o galăciu în care sunt turnate cunoștințele
- idee pe măsură ce invățăm,
adăugăm cunoștințe în galăciu

- argumentare

- modif. softului necesită înțelegerea componentelor, comportamentului ceea ce este cunoștința națională a rezultatelor din raportele decizilor luate
- orice decizie lucră trebuie doarmentată

Sisteme și modele

- Sistem = ansamblu de părți interconectate (ex: Automat Bălită - sistem)
- Model = orică abstracție a sistemului (ex: Specificație sistem X, model soft)

Participanți și roluri în proiectare

Rol	Responsabilități	Exemplu
client	furnizare cerințe, date liberă, buget, criterii calitate	Companie IT
User	cunoștințe specifice domeniului	client
Manager	- gestiune reacție client - management + motivarea personal	Project Manager
Dezvoltator	- construire sistem (+ specificații) - proiectare, implementare, testare	Analist, Programator, Tester
Specialist HCI	utilizabilitate sistemului	Specialist HCI
Responsabil documentație tehnica	- documentație livrată clientului - discuții cu dezvoltatori, manageri	Analist

- un participant poate îndeplini mai multe roluri

Producere

- artefact realizat în timpul procesului de dezvoltare
- producere de lucru intern — utilizate doar în cadrul proiectului
- producere livrabilă — pt client și specificat în contract

Producere	Tip	Descriere
Specificație	livrabil	- descrie sistemul x din perspectiva userului
Manual de operare	livrabil	- utilizat de utilizatori pt configurație sist. x
Raport de stoc	intern	- descrie pentru o dată cum se situează tabăra - fol. manager
Manual de teste	intern	- register the errors of prototype - made by tester

Activități, sarcini și resurse

- activitate = o mulțime de sarcini realizate cu un anumit scop (+ subactivități)
 - colectare cerințe \Rightarrow definire funcționalități sistem
 - gestiune proiect \Rightarrow monitorizare + control proiect
 - predare la beneficiari \Rightarrow instalarea sistem la o locație
- sarcină = unitate atomică de lucru
- resursă = bun utilizat pt. realizarea sarcini

Cerințe Funcționale / nefuncționale

- funcțională = o funcț. pe care sist. va trebui să o ofere
- nefuncțională = constrângere legată de funcționala sist. (ex. sistemul va oferi feedback într-o sec)

Notări, metode, metodologii

- notări = mulțime de reguli pt. reprezentarea modellilor (ex: UML)
- metode = tehnici cu caracter repetativ, succesiune de pași pt. rezolvarea unei prob.
- metodologie = colecție de metode pt. rezolvarea unei clase de probleme
 - OMT (Object Modeling Technique)
 - analiză
 - proiectare de sistem
 - proiectare obiectuală
 - USD^P
 - colectarea cerințelor
 - analiză
 - proiectare

Dezvoltare soft orientată - obiect

Activitate	Scop	Desenere
Colectarea cerintelor	Definirea de către client și dezvoltator a nevoilor sistemului	<ul style="list-style-type: none"> - modelul funcțional <ul style="list-style-type: none"> - descriere a sistemului în termeni de <u>acționări</u> și <u>casete de utilizare</u> ↓ descrie toate interacțiunile posibile între un actor și sistem - instrumente de reprezentare <ul style="list-style-type: none"> - diagrama UML - dialogue pt descriere textuală a cazurilor - cerințe nefuncționale
Analiza cerintelor	<p>Derivă din descrierea cazurilor un model obiectual al sistemului ⇒</p> <p>→ model complet, consistent, mecanizmuri</p>	<p>model obiectual de analiză</p> <ul style="list-style-type: none"> - model conceptual (diagrama de clase) - model dinamic (diagrama de interacțiuni) - conține concepții caracteristice dom. problemei <ul style="list-style-type: none"> - răspunde la întrebarea : ce?
Proiectarea de sistem	<ul style="list-style-type: none"> - definire obiective - descompunere sist. în subiectoare 	<ul style="list-style-type: none"> - rezultat: <ul style="list-style-type: none"> • descrierea explicită a strategiilor • descompunerea sist. • diagramă de raportare a resurselor (deployment diagram)
Proiectarea obiectuală	<ul style="list-style-type: none"> - def. de clase din domeniul soluție - leg. dintre modelul obiectual de analiză și soft 	<ul style="list-style-type: none"> - descriere a interacțiilor obiectelor și serviciilor - selecția componentelor - restructurare modelului obiectual pt a satisface obiectivele de proiectare - optimizare modelului obiectual pt performanță
Implementarea și testarea	model obiectual → cod sursă de proiect	<ul style="list-style-type: none"> - testare = identificare diferențele între sist. implementat și modelul real, pt identificare defecte → unitară = comparare model obiect de proiect cu fiecare obiect → integrare = se integrează dif. subiecte de sistem → de sistem = comparare sist. în cadrul unei acamide cu modelul corect

Diagramme UML (Unified Modeling Language)

- UML = limbaj standard adoptat de motoare pt reprezentarea modelelor orientate obiect
 - ↳ pt reprezentarea diferitor tipuri de sisteme

Tipuri de modele

- modelul funcțional
 - funcționalitatea sist. din perspectiva utilizator
 - diagrame de cazuri de utilizare
- modelul obiectual (structural)
 - structura sist în termeni de clase, atribute, associeri și operații
 - diagrame de clase
- modelul dinamic
 - comportament intern al Sist.
 - diagrame de interacțiune (seqv. de mesaje între obiecte)
 - diagrame de tranziție a stărilor
 - diagrame de activități (fluxuri de date și control)

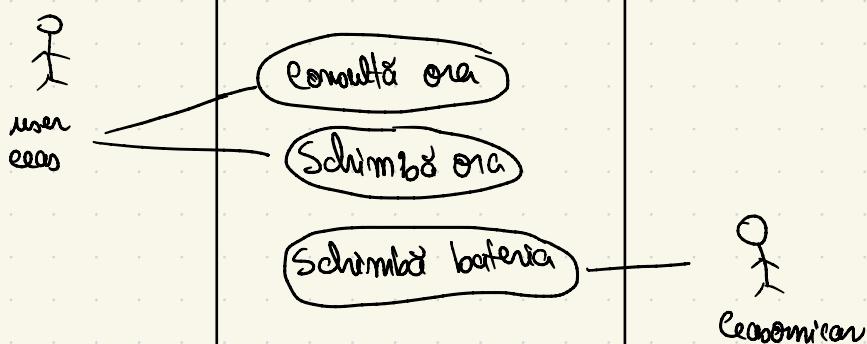
Diagramme de cazuri de utilizare

- folosit în cadrul activităților de colectare și analiză a cerințelor
- actori: factorul extern (user / alte sisteme)

↓

Identificare → definirea frontierei = diferențierea între funcțiile realizate de sistem și cele realizate de mediul său

exemplu



gestionare ale
entrărilor externe

cas de utilizare = o abstracție ce acoperă toate scenariile posibile

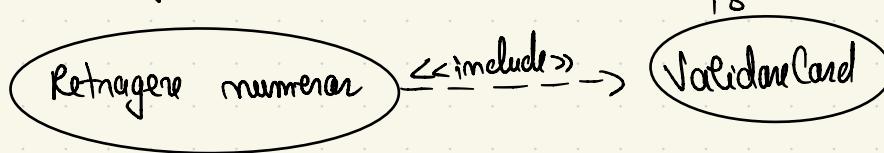
- nume (nume)
- actori participanți (care comunică cu casul)
- flux de evenimente (recv. de interacțiuni dintre actori și sistem)
 - ↳ fluxul normal și fluxurile alternative (enori, cond. speciale) și decesii separate
- condiții de intrare (precondiții)
- condiții de ieșire (postcondiții)
- limite de calitate (constrainte de performanță)

Scenariu = o instanță a unui cas de utilizare, descrivând o secvență de acțiuni

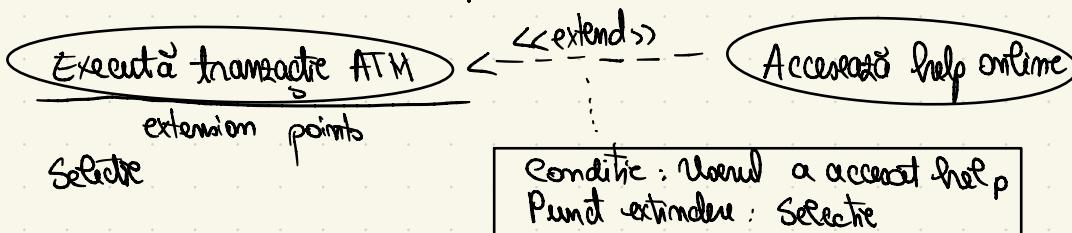
- nume scenariu (obiectivat)
- instanțele actoilor participanți (obiectivat)
- fluxul de evenimente

Relații în diagrame

- relația comunicare = actoare -> casă de utilizare = ⇒ schimb info.
 - se reprezintă ca o asociere UML binară
- relația de inclusiune = dependență între 2 cazuri de utilizare
 - ↳ casul de utilizare inclus nu este optional



- relația de extindere = dependență între 2 cazuri de utilizare
 - = precizată când și unde poate fi impreună comportamentul obținut
 - ↳ nu depinde de cel care extinde



- relația de generalizare = între clasuri / actori

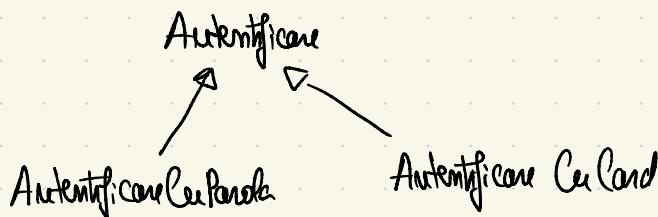
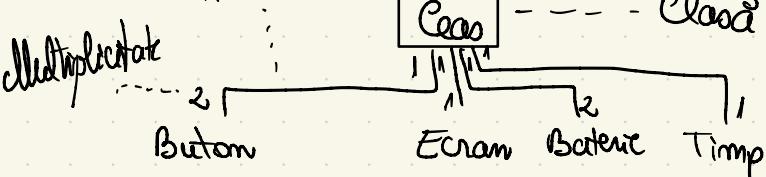


Diagramme de clase / obiecte

- utilizate în descrierea structurii unui sistem

Asocieri



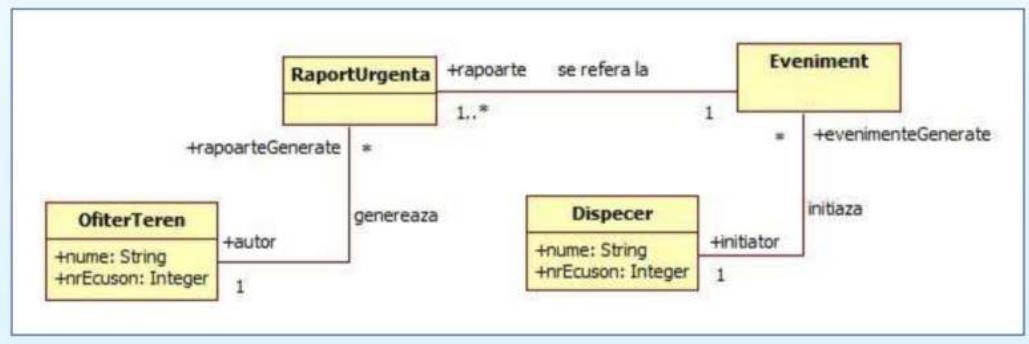
- clasa = abstracție - structura și comportamentul

- obiect = instanță a clasei, creare, modificare și distrugere

are o identitate

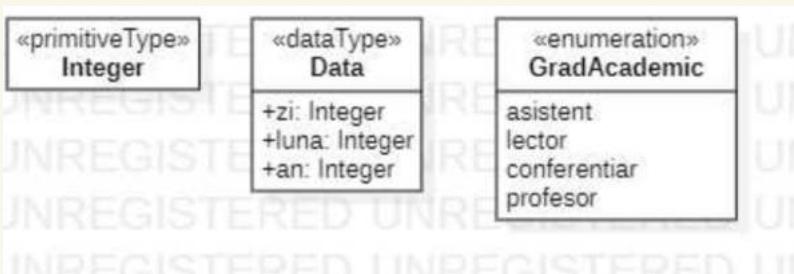
Clase și obiecte

- numărul claselor sunt la singura, ce mijlocul?



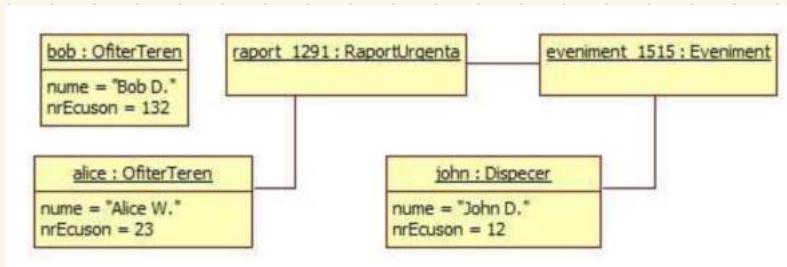
Data Type

- instanțele sunt identificate doar prin valoare (nu au identitate proprie)
- << primitive >> : Integer, Real, String, Boolean
- << enumeration >> : liste de literale
- << DataType >> : cu atribut



Obiecte

- pot primi nume în diagramile de obiecte sau pot fi anonime
- numerele se scriu cu litere mici

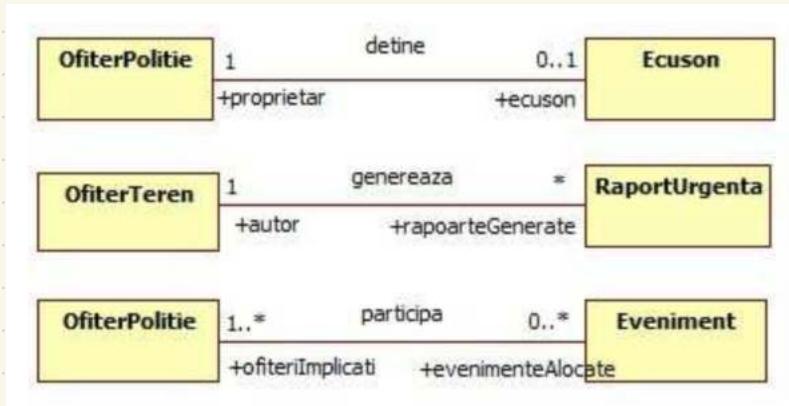


Multiplicitate și roluri

- frecare dintre capetele unei asocieri poate fi etichetată cu un nume de rol

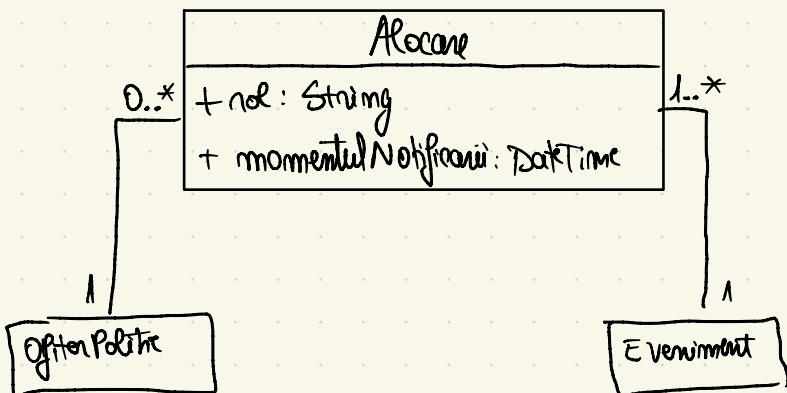
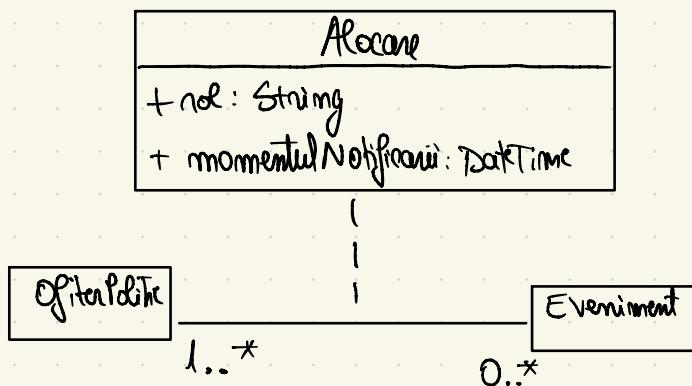
- one - to - one 1 or 0..1
- one - to - many 1..*, 0..* or *
- many - to - many

scopul asocierii și
permite diferențierea între asociere
def. și conexiunea utilizată



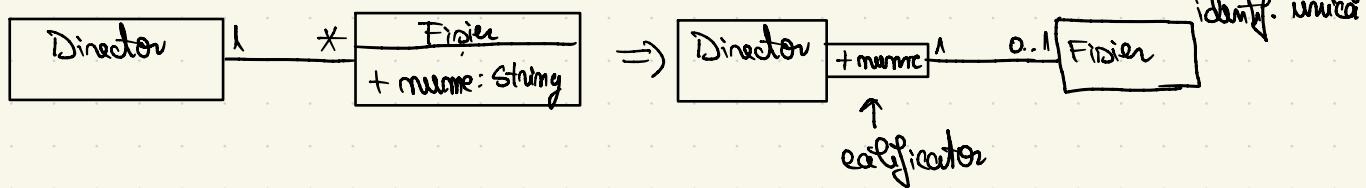
Clase asociate → reprez. relațile dintre 2 sau mai multe clase

- clasa + asociere simplă + constrângere de unicitate.



Asociere eaficata

- eaficare = o metodă de reducere a multiplicărilor, prin utilizarea cheilor

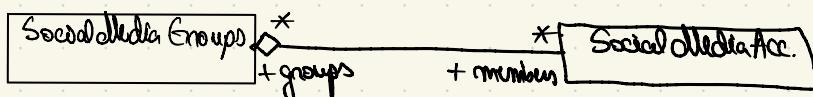


Agregare

- este particular de asociere
- o rel. tip parte - întreg („are”)

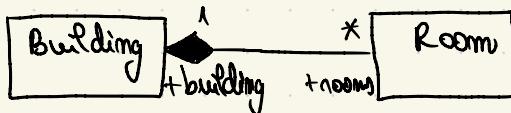
- agregare parțială

- părțile pot exista și independent



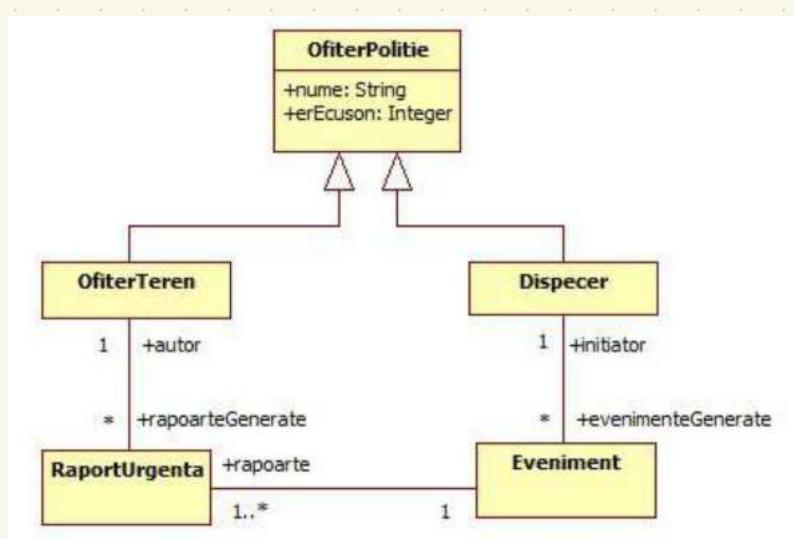
- agregare compunere

- o parte poate fi continuată în cel mult un întreg



Generalizare / Specializare

- generalizare = o relație care permite factorizarea atributelor și operațiilor comune unei mulțimi de clase



- clasele derivate moștenesc atributele și operațiile superclasei
- operatie = specificare comportament
- metodă = impl. comportament

- utilizăm diagramile de clase pt:

- în etapa de analiză a evenimentelor

- formalizare circumstanțe legate de dom.probleme

- clase \Leftrightarrow entități

- tipuri atributilor și semnificații operațiile se pot amâna pt proiectare

- în proiectarea de sistem și obiectuală

- diagramile de clase sunt refinite

- clasele sunt grupate în subsisteme

Diagrama de interacțiune

= descriu secvențe de comunicare între o mulțime de obiecte care interacționează

- obiectele interacționează prin mesaje

- receptiunea unui mesaj declanșează execuția unei metode a obiectului în cauză \Rightarrow
 \Rightarrow mesaje către alte obiecte

- un mesaj trimis poate avea argumente asociate

- tipuri: de securitate / de comunicare (sunt echivalente)

Diagrama de secvență

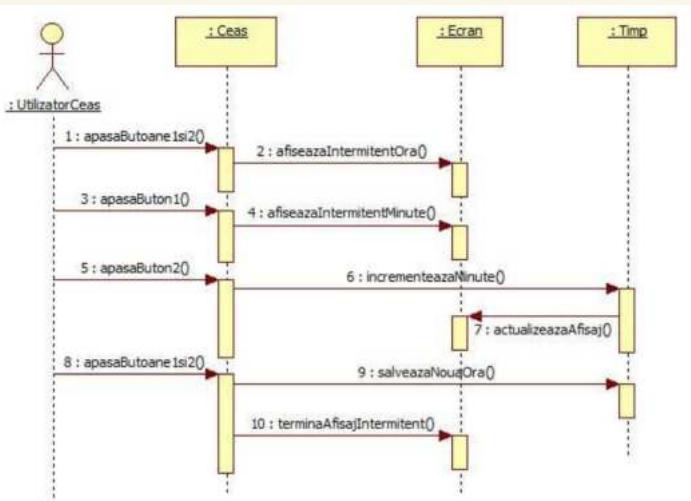
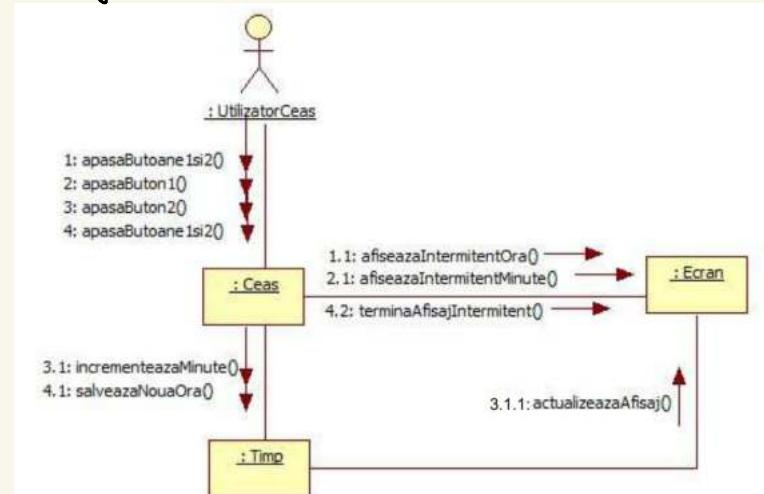


Diagrama de comunicare



- accent pe colaborările între obiectele participante la interacțiune

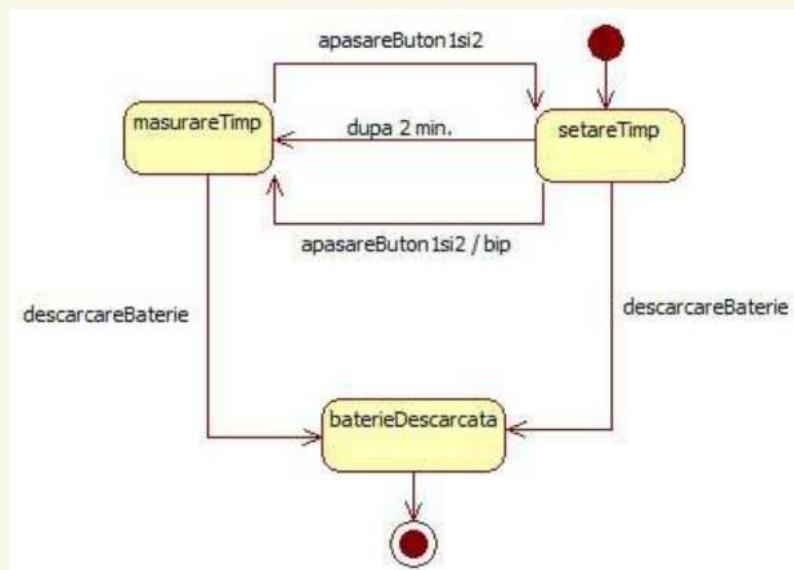
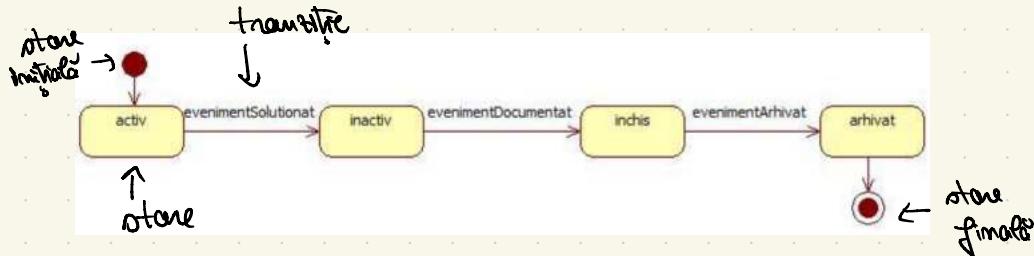
Utilizare

- identif. responsabilităților claselor și identif. de clase noi

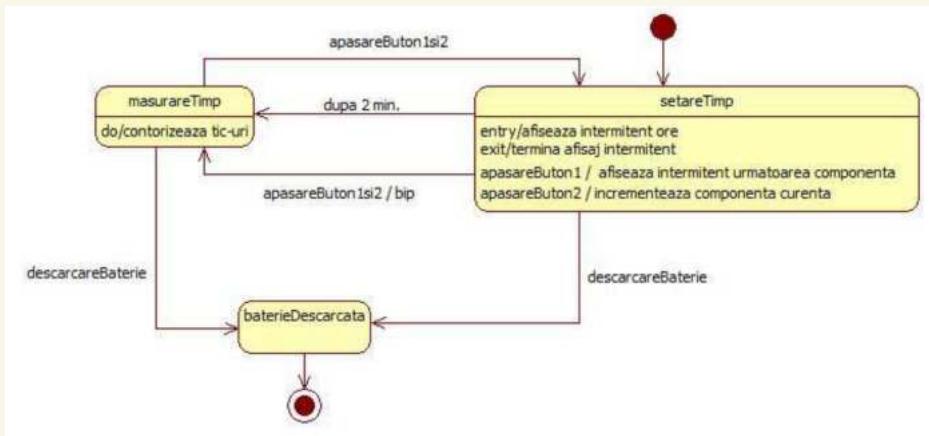
Diagramme de transiție

a stării

- O reprezentare cu stări UML = o not. folositoare pt a descrie succesiunea de stări prin care trăiește un obiect sub acțiunile evenimentelor externe
- stare = o condiție satisfăcătoare de valoare atributelor unui obiect
 - activ : necesită soluționată (ex: un incendiu)
 - inactiv : a fost rezolvată, dar nu a fost înca documentată (ex: incendiu stins, nu s-a făcut pagubă)
 - închis : situație rezolvată și documentată
 - arhivat : evenție închisă, cu documentație arhivată
- transiție = o schimbare de stare, poate fi provocată de
 - declanșarea unor evenimente
 - îndeplinirea condiții
 - trecerea unui interval de timp

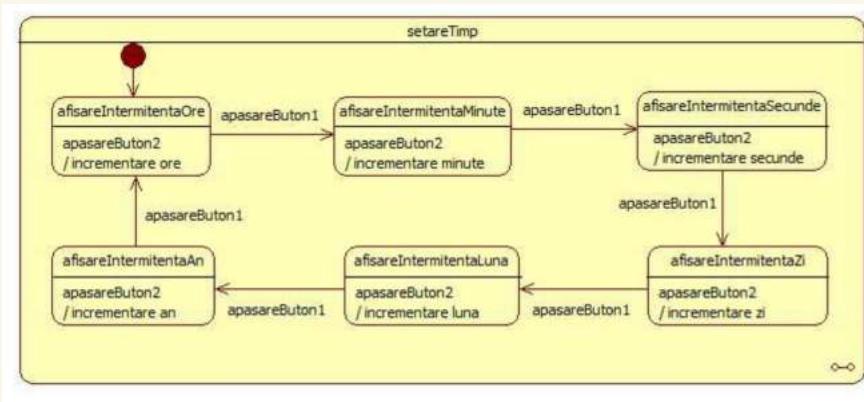


- activum = o unitate fundamentală de procesare
 - ↳ primește impetuș → produce outputuri și, posibil, schimbă starea sistemului
- transiție internă = transiție ce nu determină schimbă stării externe
- activitate = o mulțime coordonată de activuși
 - o stare poate avea o activitate asociată
 - durată mai mult timp nici pot fi întreupte de ieșirea obiectului dintr-o stare
 - introducere primă eticheta /do



Îmbunătățirea magazinilor cu stări

- o alternativă la foloarea transițiilor interne
- crește înțeleptibilitatea, reduce complicitatea diagrameelor

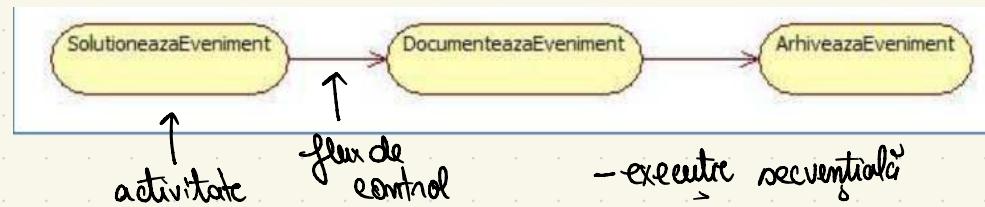


Utilizarea diagrame de transiție a stărilor

- modelare comportamentul obiectelor sau subiectelor
- etapa de analiză: identifică atributelor obiectelor din domeniul problemei
- etapa de proiectare: descrierea obiectelor din domeniul soluției este prezentată comportament complex, dependent de stări.

Diagramme de activități

= descrie model de realizare a unui eveniment
comportament într-o secvență mai multe activități
de activitate și a fluxurilor de obiecte mălăcire

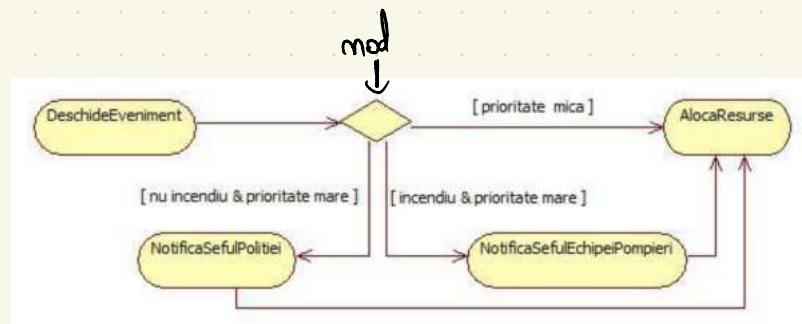


Elemente de control

- permit coordonarea flexibilă de control dintr-o diagramă de activități
- oferă mecanisme de reprezentare a deciziilor, concurenței și dimenziunii
- moduri decizionale
- moduri join
- moduri fork

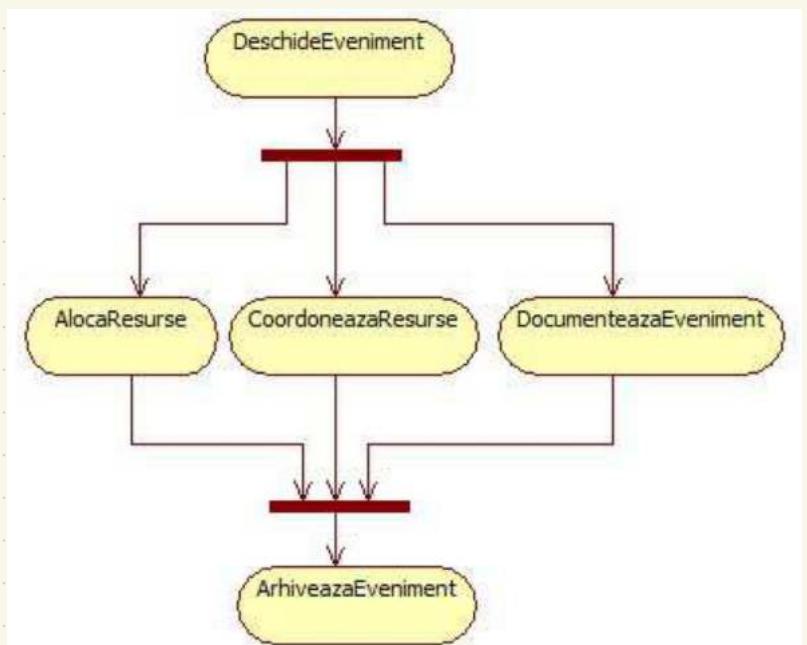
Noduri decizionale

- o ramificație a fluxului de control
- alternative pe baza unei condiții relativ la starea unui obiect sau grup de obiect



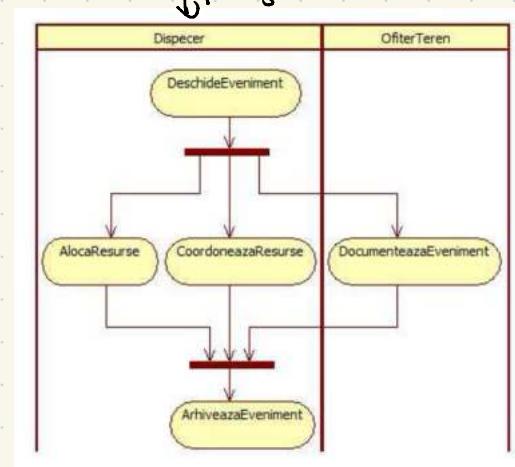
Noduri fork și join

- permit reprezentarea concurenții și a dimenziunii
- fork: diviziarea fluxului de control în threaduri
- join: sincronizarea threadurilor și combinarea fluxului de control între un thread



Partitionarea activităților

- activitățile se grupează în părți
- indică obiectul/subiectul unei acțiuni



Utilizare diagrame

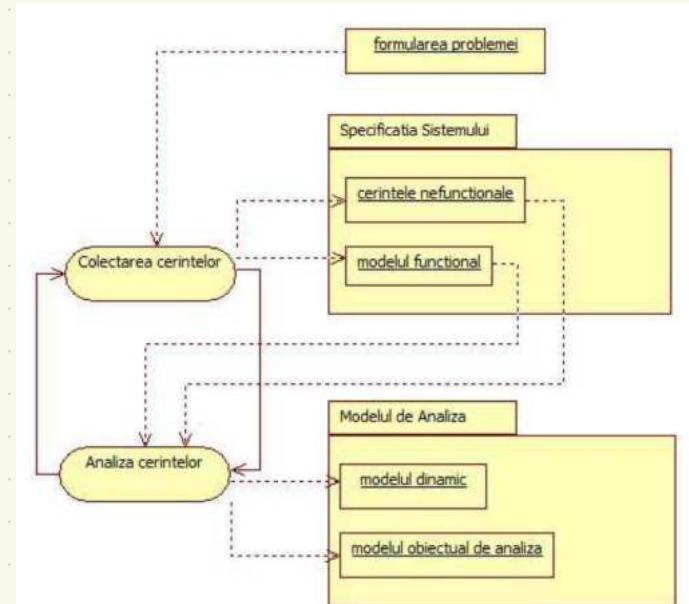
- vedea cimită pe secțiuni ale comp. unei mult. de obiecte
- desenul secțiunii prezintă activități dintr-un grup de obiecte

Colectarea cerintelor

- cerință = un element de funcționalitate pe care sistemul trebuie să îl ofere sau o constrângere pe care trebuie să o îndeplinească
- ingineria cerintelor = un subdomeniu al ingineriei softului, cu scop: definirea cerintelor sist soft ce urmează a fi construite

Activități

- colectare cerințe \Rightarrow specificație sistem (contract între client și dezvoltator)
- analiza cerintelor \Rightarrow modelul de analiză
- specificație: limbaj natural, instrument de comunicare cu client
- modelul de analiză: not. semiformală / formular, instrument de comunicare între dezvoltatori
- activitățile ingineriei cerintelor se bazează pe aspectele externe ale sistemului
- modelurile reprezintă același aspect, cele două activități se desfășoară concomitent și iterativ



- colectarea necesară colaborarea între grupuri de participanți (useri și developers)
- obiectiv: îmbunătățirea comunicării între grupuri (colectare eroare mai târziu ⇒ costisitor)
- dezvoltarea colectată erorile primă observare și gestionarea lor
- tipuri de scenarii
 - initial = procese de lucru curențe
 - ulterior = identificarea featurelor ce urmărește

Activități bazate pe scenariu

- identif. actorilor = diferențe tipuri de useri
- identif. scenariilor = dezvoltarea observă userii în mediul lor
 - = dezvoltă un set de scenarii detaliate
- identif. cazurilor de utilizare = scenarii ⇒ cazuri de utilizare
- reafinare cazuri utilizare
- identif. relațiilor dintre cazurile de utilizare = identif. dependențele (prevenire și consistență)
- identif. cerințelor nefuncționale = performanță, rezurse, securitate

Cerințe funcționale

= descriu interacțiunile dintre sistem și mediul aceluiaș
 ↳ useri + altă lăst cu care interacționează

Cerințe nefuncționale

- = descriu diverse tipuri de constrângeri impuse operației sistemului
- utilizabilitate (uzabilitate de utilizare)
- performanță (temp de răspuns, disponibilitate, etc.)
- fiabilitate (abilitatea sistemului să aibă deplină funcții în cond. stabile, pt. un temp)
- supraviețuirea (rezistență modif. sist. după învățare)
 - adaptabilitate
 - maintenabilitate
 - internationalizare
- cerințe adiționale
 - implementare (hardware)
 - interfață (constrângeri de altă păr.)
 - modul de operare (constrângeri admin. și gesturale)
 - instalare (livrare sistem)
 - legale (licențe, legi)

- **completitudine** = specificarea cunoștințelor se consideră a fi completă dacă surprinde toate aspectele de interes pt user
- **coherență** = fără cunoștințe sunt non-contradictorii
- **claritate** = același cunoștință nu admite interpretări distinții
- **coauditorie** = reprezentă fidel interesele clientului și ale developerilor
- **realism** = împl. cu cunoștințe
- **verificabil** = teste și conformitate
- **transabilitate** = fiecare cunoștință poate fi urmărită
 - ↳ cunoștințe critice
 - utilizator în justificări/argumente
 - urmărită prim referințe într-o documentare, modul și cod

Ingineria Greenfield

- proces dezvoltare de b.o
- cunoștințe furnizate de useri

Re-inginerie

- reproiectare și reimplementare sist.-existant
- funcționalitatea poate fi extinsă

developarii acumulatoare este mai multe cunoștințe
la domeniul problemei (manuale, note/interviu useri, etc)

Ingineria interfețelor

- reproiectare și reimplementare doar a interfeței

Cazuri de utilizare

- nume cauză (Alocă Rezurse)
- nume actori (Dispecer)
- flux de evenimente
- excepții / alternative
- max 2-3 pag/cauză

• Ex.: Cazul de utilizare Raporteză Urgență al SGA

Nume	Raporteză Urgență
Participanți	Inițiat de OfițerTeren Comunică cu Dispecerul
Flux de evenimente (scenariu normal)	<ol style="list-style-type: none"> Ofițerul activează funcția Raporteză urgență a terminalului. Sistemul SGA afișează un formular Ofițerului. Ofițerul completează formularul, inserând nivelul de alertă, tipul, locația și o scurtă descriere a situației. Poate propune și eventuale soluții la situația de urgență. După completare, Ofițerul trimite formularul. Dispecerul consultă informația primită și apelează cazul de utilizare DeschideCazNou. Dispecerul optează pentru una dintre soluțiile propuse și confirmă primirea formularului. Sistemul afișează confirmarea și soluția aleasă Ofițerului.
Condiții de intrare	Ofițerul este logat în sistem.
Condiții de ieșire	Ofițerul a primit confirmarea de la Dispecer SAU o explicație privind motivul eșecului tranzacției.
Cerințe de calitate	Confirmarea Dispecerului ajunge în maxim 30 de sec. după trimitere.

Performanța cazurilor de utilizare

- de identificare - funcționalități mecanice
- de introducere detaliu mecanice
 - drepturi acces
 - cazuri excepție
 - interacțiuni useri - sistem

Nume	Raportează Urgență
Participanți	Initiat de OfițerTeren Comunică cu Dispecerul
Flux de evenimente	<ol style="list-style-type: none"> Ofițerul activează funcția Raportează urgență a terminalului. Sistemul SGA afișează un formular Ofițerului. Formularul include componente privind nivelul de alertă, tipul urgenței (general, incendiu, accident auto), locația, descrierea și resursele solicitate. Ofițerul completează formularul, inserând cel puțin nivelul de alertă și descrierea situației. El poate propune și eventuale soluții la situația de urgență și poate să solicite anumite resurse. După completare, Ofițerul trimite formularul. Sistemul primește formularul și notifică Dispecerul. Dispecerul verifică informația primită și creează un nou Eveniment în baza de date prin invocarea cazului de utilizare DeschideCazNou. Toate informațiile din formularul primit sunt asociate automat evenimentului creat. Dispecerul alege un răspuns prin alocarea de resurse la eveniment (prin cazul de utilizare AlocaResurse) și confirmă primirea formularului printr-un scurt mesaj către ofițer. Sistemul afișează confirmarea și răspunsul ales Ofițerului.

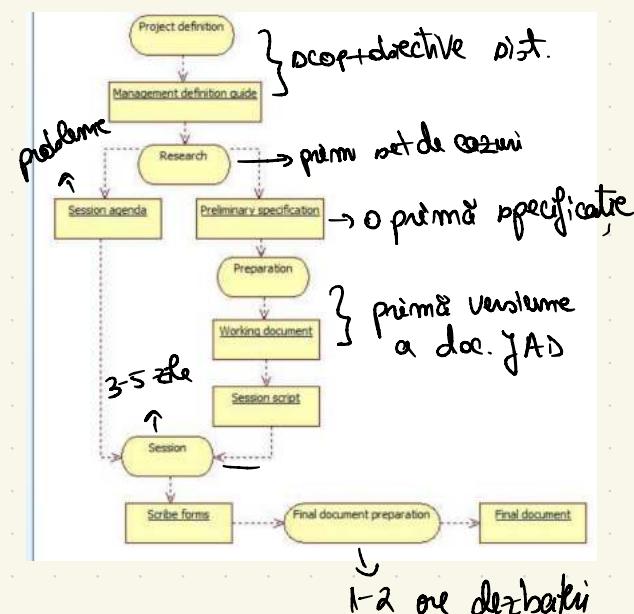
Identificare relații

- reduse complexitate
- eliminarea redundantele / inconsistente
- factorizare comp. comune \Rightarrow relație inclusiune
- modelare comp. optional \Rightarrow relație extensie
- utilizarea judecății a relațiilor
 - \hookrightarrow intelligent, eficace, eficiente

Metoda JAD

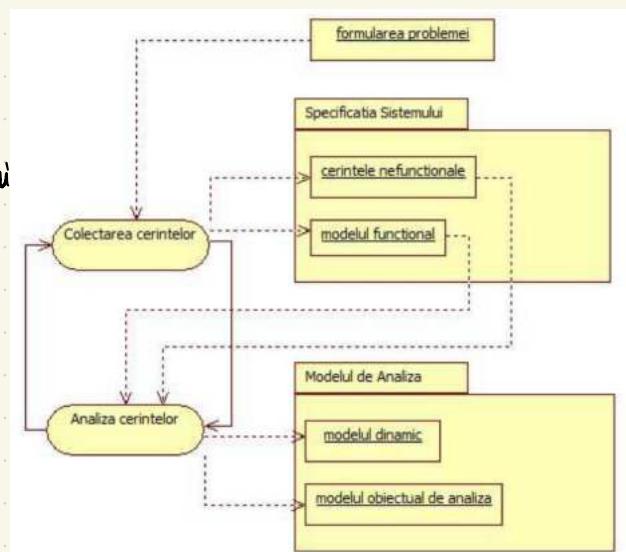
= met. de colectare a cunoștințelor

- activitatea de colectare - între o nr. reziliu de workshop
- \Rightarrow specificație completă sistem
- participare useri, dezvoltatori + moderator specializat
- minimizarea riscului unei modif. ulterioare



Analiza cerintelor

- scop: model al sistemului (de analiză)
- formalizarea \Rightarrow identifică ambiguități, incoherențe, incompletitudini
SOLUȚIE: discuții cu userul
- colectare + analiză = activități consecutive

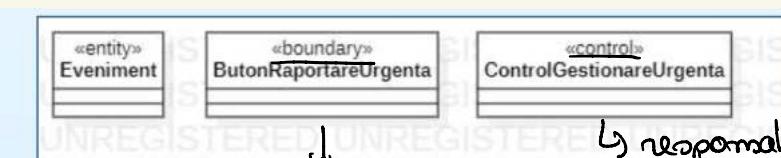


Modelul obiectual de analiză

- diagrama de clase
- concepte manipulabile de sistem, proprietăți
- clasa entity \rightarrow info persistentă în sistem
- clasa boundary \rightarrow interacțiune user
(ex.: Buton Raportare Urgență)
- clasa control \rightarrow realizarea cerințelor de utilizare

Modelul dinamic \rightarrow comp. sistem

- diagrame de sevență și traiectorie
- identifică responsabilitățile entității clase / asociu moj.



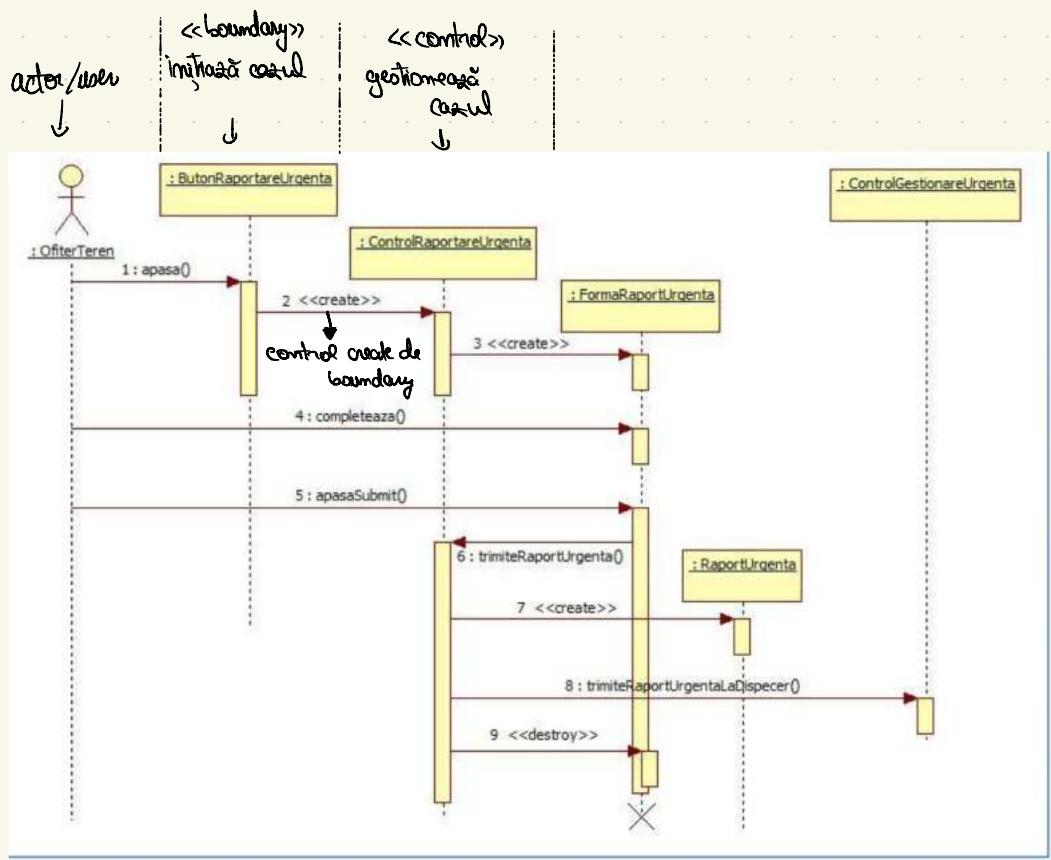
↓
Colectarea inputul
userilor și-l transof.
într-o formă utilizabilă
de clasele «entity» și «control»

↳ responsabilele de
condamarea obiectelor
«entity» și «boundary»
asociate fiecărui un obiect
de utilizare

Clasa Boundary

- bufeante
- forme pt introducere date
- mărfării
- terminale

Diagramme de sequence



Asocieri

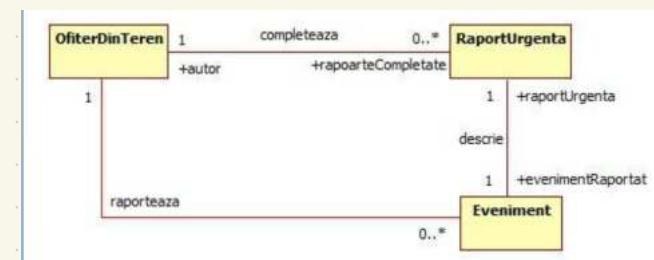
Nlin ofiter reprezentă raportante

actoare

↓

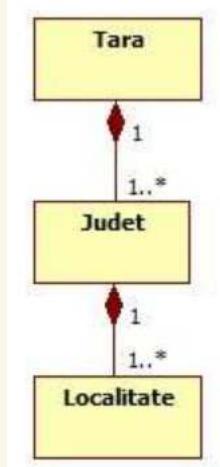
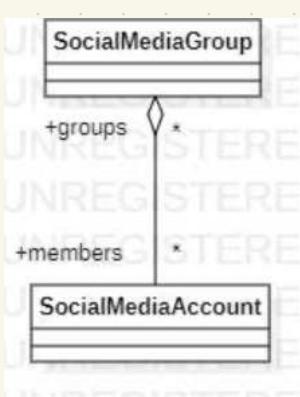


- redundant (ciclu)



Agregație (completă / partajată)

= demotează o relație tip parte - întreg



Atribute



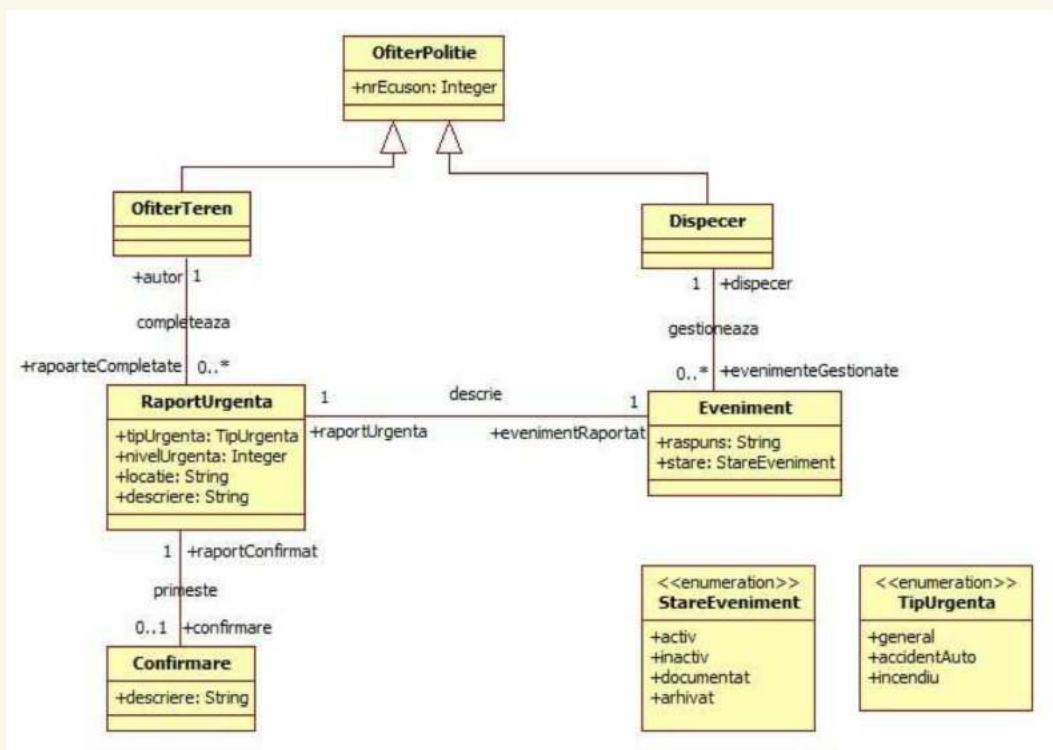
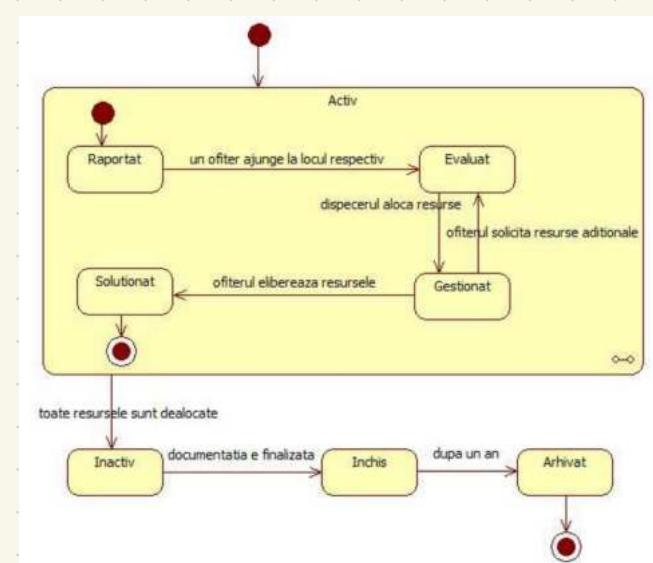
Modelare comportament

- diagramme de sevență

↓
1 cas de utilizare

/ de transiție a stăriilor

↓
1 ng. obiect

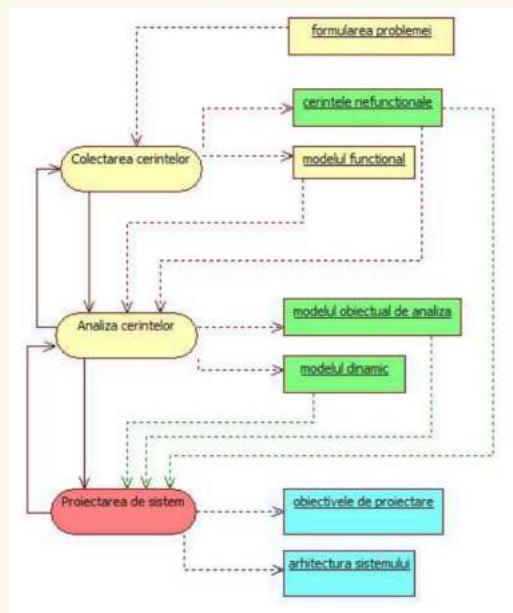


Proiectarea de sistem

= procesul de transf. a modelului rezultat din îngineria cerintelor într-un model arhitectural al sist.

Activități:

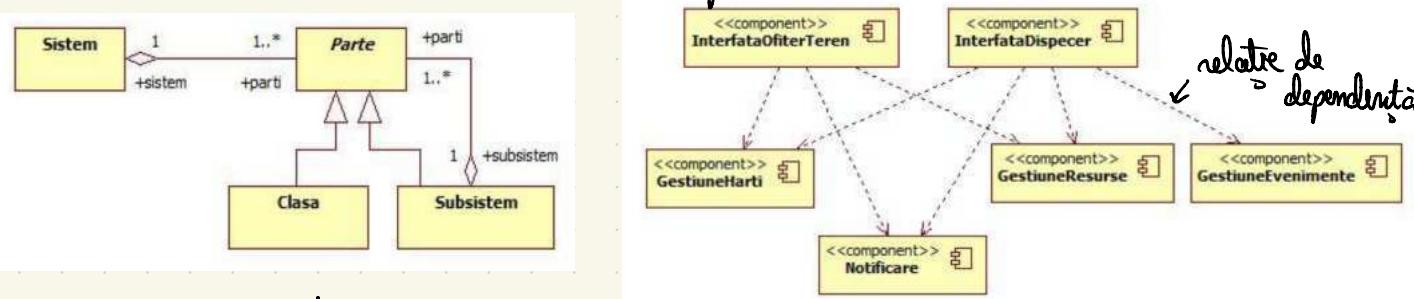
- identif. obiectivelor de proiectare
- descompunerea inițială a sistemului
- reafinarea descompunerii inițiale



Subsisteme și clase

- subsistem = parte înlăturabilă a unui sistem (conține proprii interfețe)
- încapsulează starea și comportamentul claselor componente

exemplu

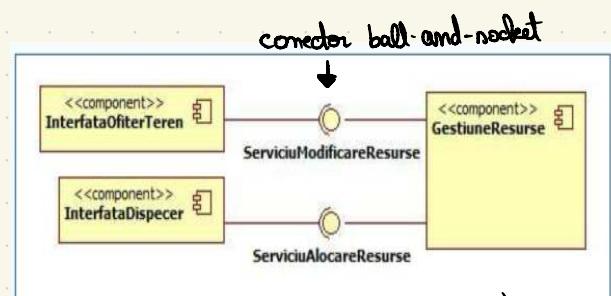


« component »

- logică = subsistem fără reuniune
- fizică = are echivalent reuniune

Servicii, interfețe și API-uri

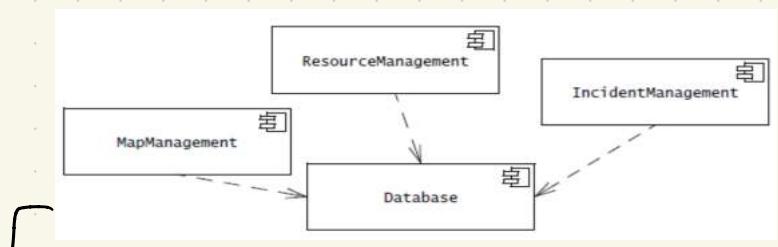
- serviciu = mult. de operații în nuditate
- interfață = mult. de operații UML în nuditate, complet specificate (nume, parametri)
- API (application programming interface)
- = specificare interfețe subsistem într-un limbaj de programare



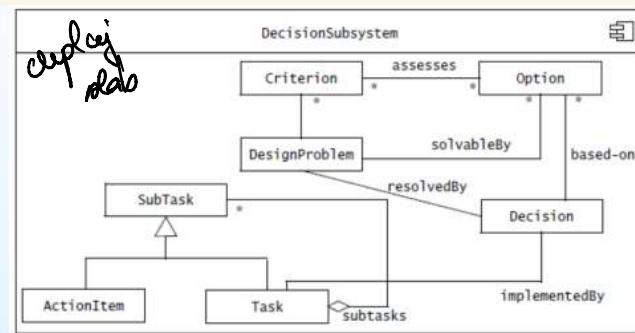
- Ball (lollipop) = interfață oferită
- Socket = interfață solicitată

Corespunzătoare

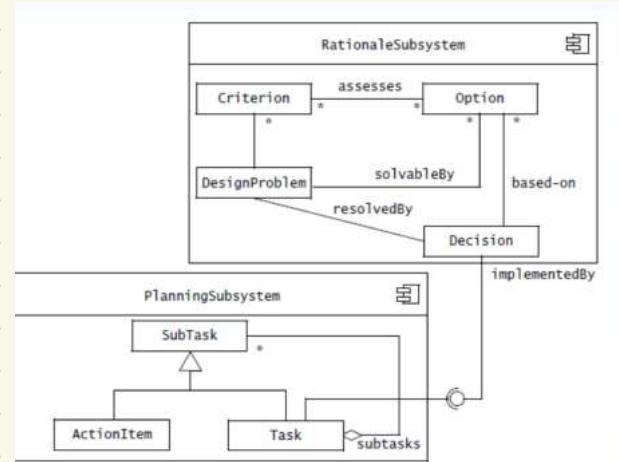
- corespunzătoare = dependență între 2 sist.
 - strânsă = m. mare de dependență
 - slabă = m. mică de dependență
- coesivitate = măsură a dependențelor unui subsistem
 - slăbită = m. mare de către puternice relații cu servicii similare
 - slabă = m. mare de către menșele orice



reducere corespunzătoare



creștere coesivitate



Stratificare și partitionare

- stratificare = descompunere hierarhică a unui sist.
 - generașă o mulțime ordonată de straturi
 - strat = grup de subsist. cu servicii înrudite
 - ↳ sunt ordonate
- partitionare = generașă grup de subsist. la același nivel

Anhitectură \Rightarrow închisă = acces la servicii din stratul immed. inferior

deschisă = acces la servicii dintr-o stratură inf.

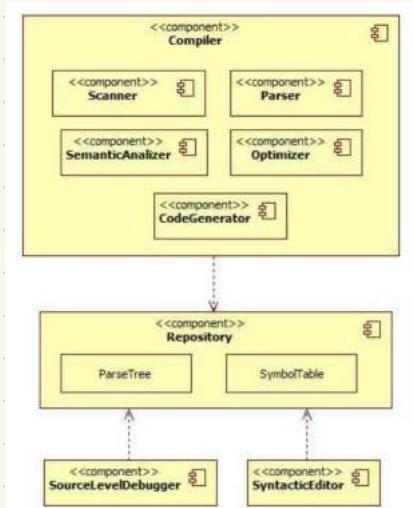
strucție arhitecturală = subiectul descompunerii subiect.

arhitectură software = instanță a unei strucții arhitecturale (ex. Repository)

Repository (SGBD, IDE)

Avantaje

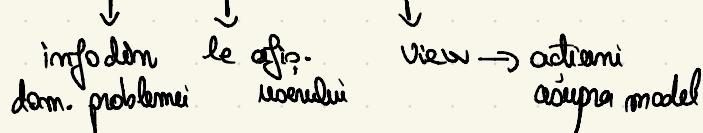
- ușoară în procese complexe, în continuu schimbabile
- servicii mai puțin dep. de noi subiect.



Dezavantaje

- subiectul reprezentă o strucție complexă \Rightarrow modif repos afectează subiect.
- performanță deosebită

Model-View-Controller (MVC)



! subiect tip model sunt independente de view/controller

pe inter. astăzi de motif,
pe baza Observer

- view + controller predă/pun la schimbare

- un tip particular de Repository (model = structura Repo, controller = flux de control Repo)

Client - Server

- server oferă servicii către client

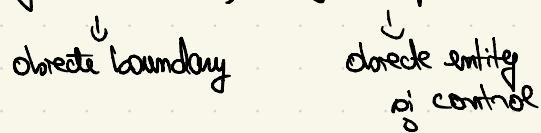
- particular Repository, structura de date gestionată de un proces

Peer-to-peer

- generalizare a client-server → un subiect poate juca atât rol de client, cât și de server

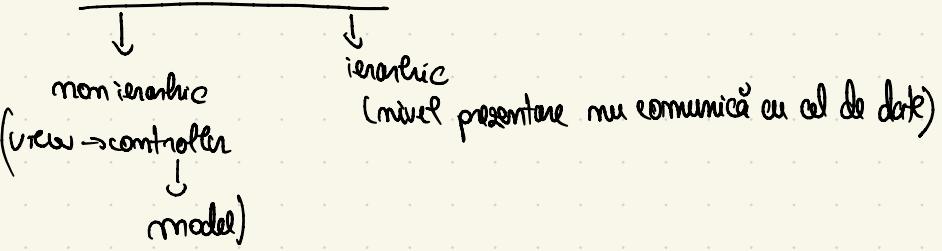
Three-tier architecture

- 3 straturi: interfață user, logica aplicației, acces la date



↳ gestionarea și oferă acces
la datele cu caracter permanent

MVC vs. Three-tier



! MVC nu acopără prob.
persistenței datelor

Four-tier architecture

- o variație a Three-tier

- niv. interfață devine: prezentare server, prezentare client

pot exista dif. direcții

↳ parte din directele
boundary ⇒ reutilizabile

Pipes and Filters

- pipeline = lanțuri emisate procesare (threaduri, ...)

- outputul uneia este inputul celeilalte

- conține 2 tipuri de subiect: pipes și filters → one um comul de intrare și

conexiunea
dintre 2 pasi

umul de ieșire
efect, um pas
de procesare

Subactivități în rezolvarea descompunerii amănără

- mapearea hardware - software
- gestionarea datelor cu caracter persistent
- definirea politicilor privind controlul accesului
- stabilirea fluxului global de control
- dezervarea cerințelor limită

Identificare obiective de proiectare

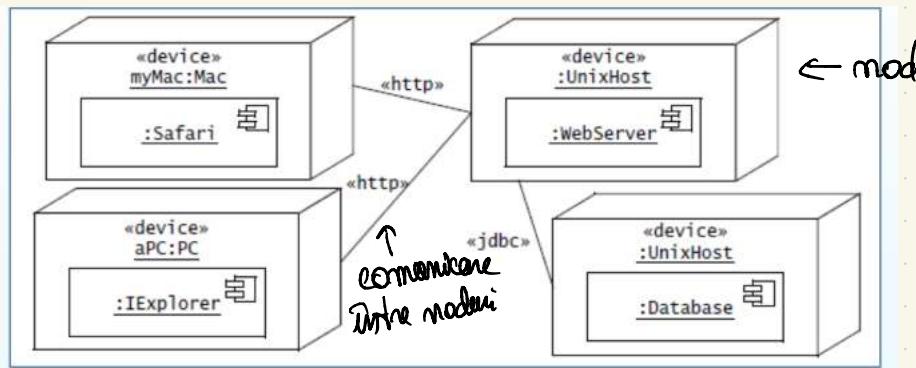
- obiective de proiectare = criterii de calitate pe care trebuie să le formeze
- performanță, dependabilitate, cost, întreținere, criterii user

Schimbul Facade

- = o interfață unificată (de nivel front) pt un grup de interfețe
- dim. nr. de dependențe între client și clasa de implementare
- strategia unui subsystem

Diagrama de repartizare a reacțorilor

- = ilustrarea relației dintre componentele aceluiași modul
- nod = dispozitiv fizic (computer)



Schimbul proxy

- = adaugare pt un client un inter.
- proxy la distanță = oferă un reprezentant local dintr-un spatiu de adrese def.
- proxy virtual = creează la cerere obiecte costisitoare

Gestioneaza datele persistente

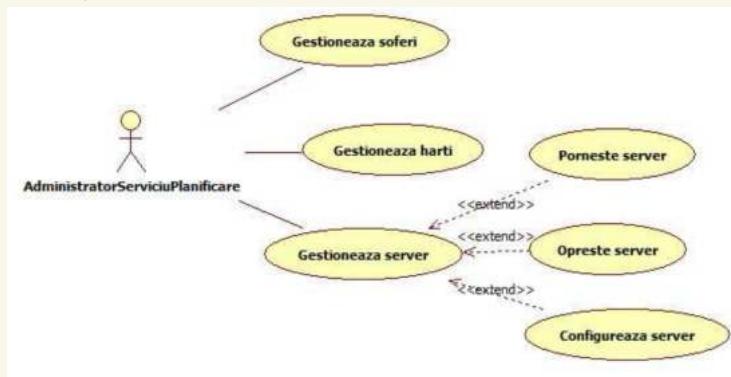
- unele obiecte ce conținuri trebuie să fie persistente
- de realizare prim:
 - ✓ fizice (1 scriitor, n cititori)
 - ✓ BD (cititori și scriitori concurenți)

Proces

- identifică obiectele persistente (toate clasele care au obiecte fizice sau nu sunt urmăriți unei opini)
- selectare strategie de memorare (Fizică, BD relațional, BD orientată - obiect)
 - ↓
 - viteză
 - ↓
 - gestiune concurență
 - ↓
 - mai lentă, abstractizare

Motrici de acces = accesul la clase

Cazuri de utilizare limită



Reutilizare