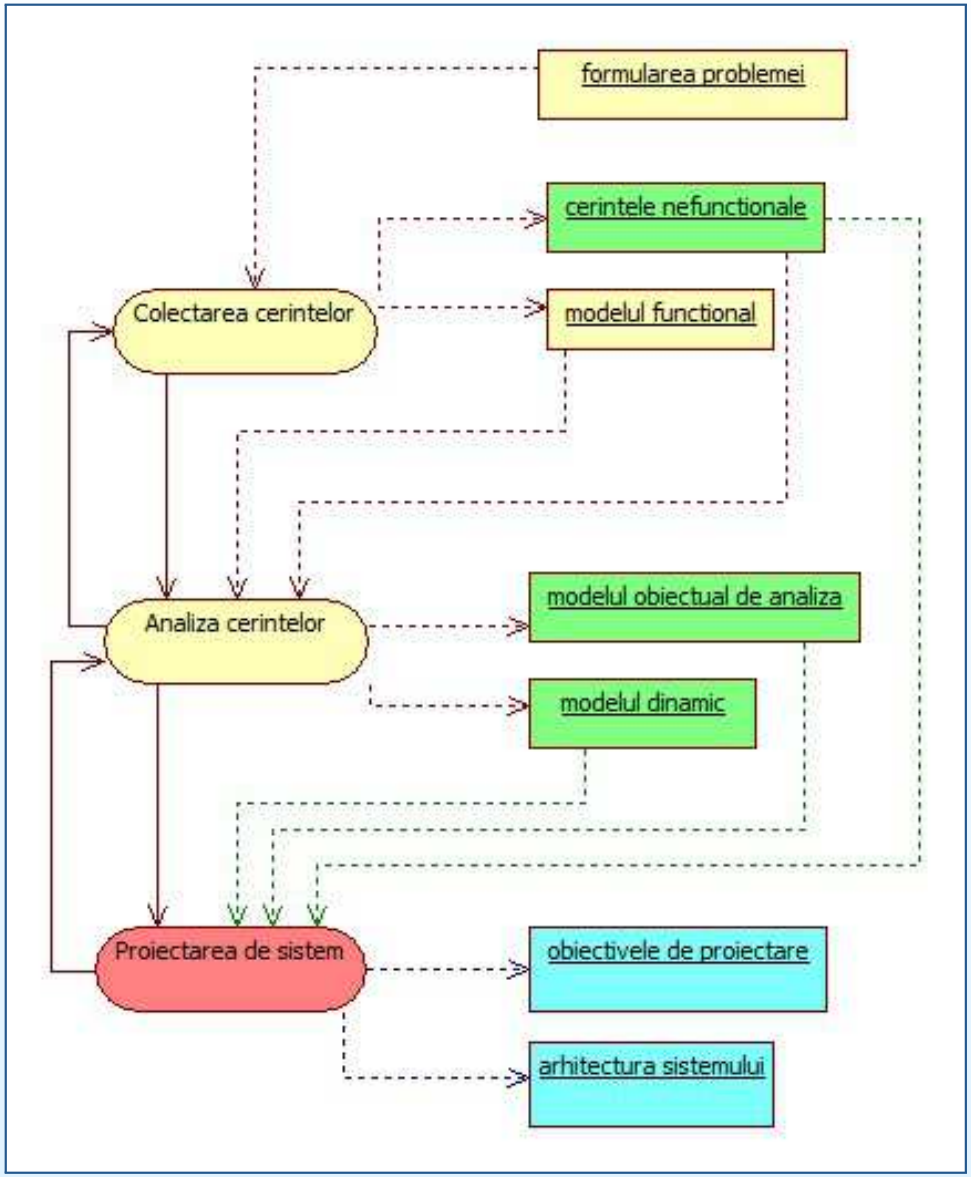
Curs 6

Proiectarea de sistem (II)

Suport de curs bazat pe ***B. Bruegge and A.H. Dutoit***

"Object-Oriented Software Engineering using UML, Patterns, and Java"

Proiectarea de sistem



Proiectarea de sistem (cont.)

- Procesul de transformare a modelului rezultat din ingineria cerinţelor într-un model arhitectural al sistemului
- Produse ale proiectării de sistem
  - *Obiectivele de proiectare* (eng. *design goals*)
    - Calităţi ale sistemului pe care dezvoltatorii trebuie să le optimizeze
    - Derivate din cerinţele nefuncţionale
  - *Arhitectura sistemului*
    - Subsistemele componente (de dimensiuni mai mici, asignabile unei subechipe de dezvoltare)
    - Responsabilităţile subsistemelor şi dependenţele între ele
    - Maparea subsistemelor la hardware
    - Strategii de dezvoltare: strategia de gesionare a datelor cu caracter persistent, politicile de control a accesului, fluxul global de control

Proiectarea de sistem (cont.)

- Activităţi ale proiectării de sistem
  - *Identificarea obiectivelor de proiectare*
    - Identificarea şi stabilirea priorităţilor acelor calităţi ale sistemului pe care dezvoltatorii trebuie să le optimizeze
  - *Descompunerea iniţială a sistemului*
    - Pe baza modelului funcţional şi a modelelor de analiză
    - Bazată pe utilizarea unor stiluri arhitecturale standard
  - *Rafinarea descompunerii iniţiale pentru a răspunde obiectivelor de proiectare*
    - Rafinarea arhitecturii de la pasul anterior până la îndeplinirea tuturor obiectivelor de proiectare
- Analogie cu proiectarea arhitecturală a unei clădiri
  - Componente: camere vs. subsisteme
  - Interfeţe: pereţi/uşi vs. servicii
  - Reproiectare: mutarea pereţilor vs. schimbarea subsistemelor/interfeţelor

Proiectarea de sistem (cont.)

- Subactivităţi în rafinarea descompunerii iniţiale
  - *Maparea hardware-software*
    - Aspecte urmărite: Care este configuraţia hardware a sistemului? Cum sunt distribuite funcţionalităţile pe noduri? Cum se realizează comunicarea dintre noduri? Ce servicii sunt realizate folosind componente existente? Cum sunt aceste componente încapsulate?
    - Această subactivitate conduce adesea la definirea unor subsisteme adiţionale, dedicate transferului de date de la un nod la altul sau gestionării problemelor legate de concurenţă şi fiabilitate
  - *Gestiunea datelor cu caracter persistent*
    - Aspecte urmărite: Care sunt datele cu caracter persistent? Unde ar trebui stocate aceste date? Cum vor fi ele accesate?
    - Această subactivitate conduce adesea la selectarea unui sistem de gestiune a bazelor de date şi la identificarea unor subsisteme adiţionale dedicate gestiunii datelor cu caracter persistent

Proiectarea de sistem (cont.)

- *Definirea politicilor privind controlul accesului*
  - Aspecte urmărite: Cine are acces la date? La care dintre ele? Se pot schimba dinamic drepturile de acces? Cum este specificat şi realizat controlul accesului?
- *Stabilirea fluxului global de control*
  - Aspecte urmărite: Cum sunt secvenţiate operaţiile în sistem? Este sistemul unul dirijat de evenimente? Poate el gestiona mai mulţi utilizatori simultan?
- *Descrierea cazurilor limită* (eng. *boundary conditions*)
  - Aspecte urmărite: Cum este iniţializat şi oprit sistemul? Cum sunt gestionate cazurile excepţionale?

Exemplu: sistemul *MyTrip*

- *MyTrip* este un sistem care permite planificarea călătoriilor (traseelor aferente) de către şoferi
- Folosind *MyTrip*, un utilizator îşi poate planifica traseul de efectuat prin intermediul unui calculator personal, prin conectarea la un serviciu Web de planificare (cazul de utilizare *PlanificareTraseu*). Traseul este salvat pe server, în vederea consultării ulterioare. Serviciul de planificare trebuie să suporte diferiţi utilizatori
- Ulterior, utilizatorul/şoferul efectuează călătoria cu maşina personală, în timp ce calculatorul de bord îi oferă instrucţiuni, pe baza informaţiilor legate de traseu salvate pe server şi a poziţiei curente indicate de sistemul GPS încorporat (cazul de utilizare *EfectuareTraseu*)

MyTrip: cazul de utilizare *PlanificareTraseu*

Nume	*PlanificareTraseu*
Flux de evenimente	1. Şoferul îşi deschide calculatorul personal şi se loghează în serviciul Web de planificare. 2. Şoferul introduce o listă de destinaţii la care doreşte să ajungă. 3. Folosind o bază de date cu hărţi, serviciul de planificare calculează cea mai scurtă variantă de a vizita destinaţiile, în ordinea specificată. Rezultatul este dat sub forma unei serii de segmente de drum, ce unesc o serie de intersecţii şi a unei liste de instrucţiuni. 4. Şoferul poate revizui planul, prin inserarea sau ştergerea unor destinaţii. 5. Şoferul salvează traseul sub un nume ales în baza de date a serviciului de planificare, în vederea utilizării ulterioare.

MyTrip: cazul de utilizare *EfectuareTraseu*

Nume	*EfectuareTraseu*
Flux de evenimente	1. Şoferul îşi porneşte maşina personală şi se loghează în sistemul de asistenţă rutieră al calculatorului de bord. 2. După logare, şoferul specifică serviciul Web de planificare şi numele traseului pe care doreşte să îl execute. 3. Sistemul de asistenţă obţine lista destinaţiilor, instrucţiunilor, segmentelor de drum şi intersecţiilor aferente de la serviciul de planificare. 4. Pe baza poziţiei GPS curente, sistemul de asistenţă oferă şoferului următorul set de instrucţiuni. 5. Şoferul ajunge la destinaţie şi opreşte sistemul de asistenţă.

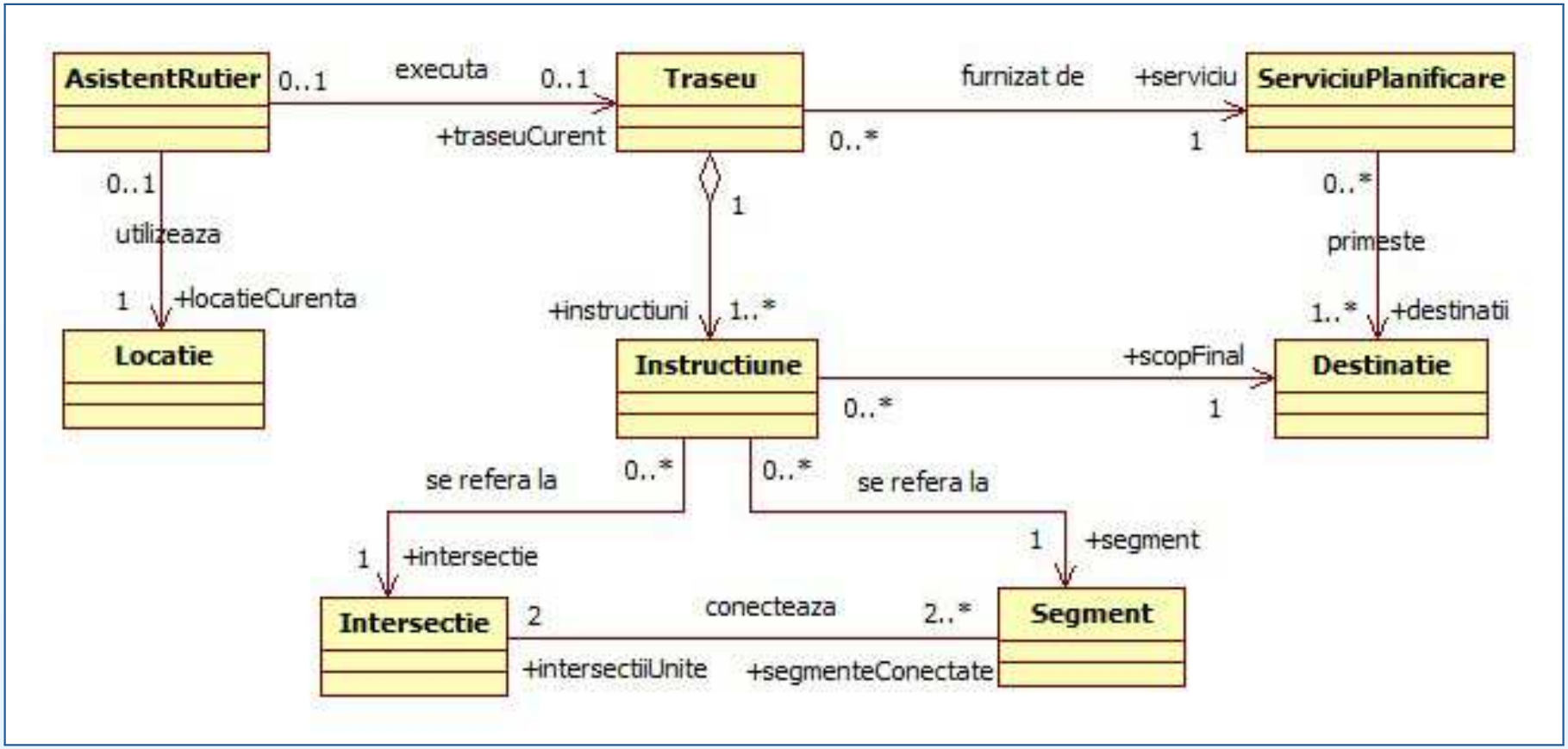
MyTrip: cerinţe nefuncţionale

1. Conexiunea cu serviciul de planificare este realizată prin intermediul unui modem wireless. Se presupune că acesta funcţionează corect la locaţia iniţială.
2. Odată începută călătoria, sistemul *MyTrip* trebuie să furnizeze instrucţiuni corecte, chiar şi în condiţiile pierderii conexiunii cu serviciul web de planificare.
3. Sistemul *MyTrip* trebuie să minimizeze timpul de conectare, pentru a reduce costurile de operare.
4. Replanificarea este posibilă doar în condiţiile în care conectarea la serviciul de planificare este posibilă.
5. Serviciul de planificare trebuie să suporte cel puţin 50 de şoferi diferiţi şi cel puţin 1000 de trasee diferite.

MyTrip: concepte din domeniul problemei

Destinaţie	O *Destinaţie* reprezintă un loc în care şoferul doreşte să ajungă
Intersecţie	O *Intersecţie* reprezintă un punct geografic în care se întâlnesc mai multe *Segmente*
Segment	Un *Segment* reprezintă calea dintre două *Intersecţii*
Instrucţiune	Dată fiind o *Intersecţie* şi un *Segment* adiacent, o *Instrucţiune* descrie, în limbaj natural, modalitatea de a dirija maşina pe respectivul *Segment*
Traseu	Un *Traseu* constă dintr-o succesiune de *Instrucţiuni* între două *Destinaţii*
ServiciuPlanificare	Un *ServiciuPlanificare* este un serviciu Web care poate construi un *Traseu* pe baza unui şir de destinaţii, sub forma unei secvenţe de *Intersecţii* şi *Segmente* asociate
AsistentRutier	Un *AsistentRutier* oferă *Instrucţiuni* şoferului, pe baza *Locaţiei* curente şi a următoarei *Intersecţii*
Locaţie	O *Locaţie* reprezintă o poziţie a maşinii indicată de sistemul GPS încorporat

MyTrip: modelul conceptual



Identificarea obiectivelor de proiectare

- *Obiective de proiectare* = criterii de calitate pe care sistemul trebuie să se focuseze
- Trebuie specificate explicit, pentru ca fiecare decizie luată să să se conformeze aceluiaşi set uniform de criterii
- Categorii de obiective de proiectare
  1. *performanţă*: timp de răspuns, puterea de calcul, memorie utilizată
  2. *dependabilitate*: robusteţe, fiabilitate, disponibilitate, toleranţă la erori, securitate, siguranţă
  3. *cost*: costuri de dezvoltare, instalare, întreţinere, administrare
  4. *întreţinere*: extensibilitate, modificabilitate, adaptabilitate, portabilitate, lizibilitate, trasabilitatea cerinţelor
  5. *criterii utilizator*: utilitate, utilizabilitate
- Categoriile 1,2,5 pot fi deduse din cerinţele nefuncţionale şi domeniul problemei, celelalte sunt dictate de client şi furnizor
- Priorităţi şi compromisuri: viteză vs. memorie, deadline vs. funcţionalitate/calitate/personal

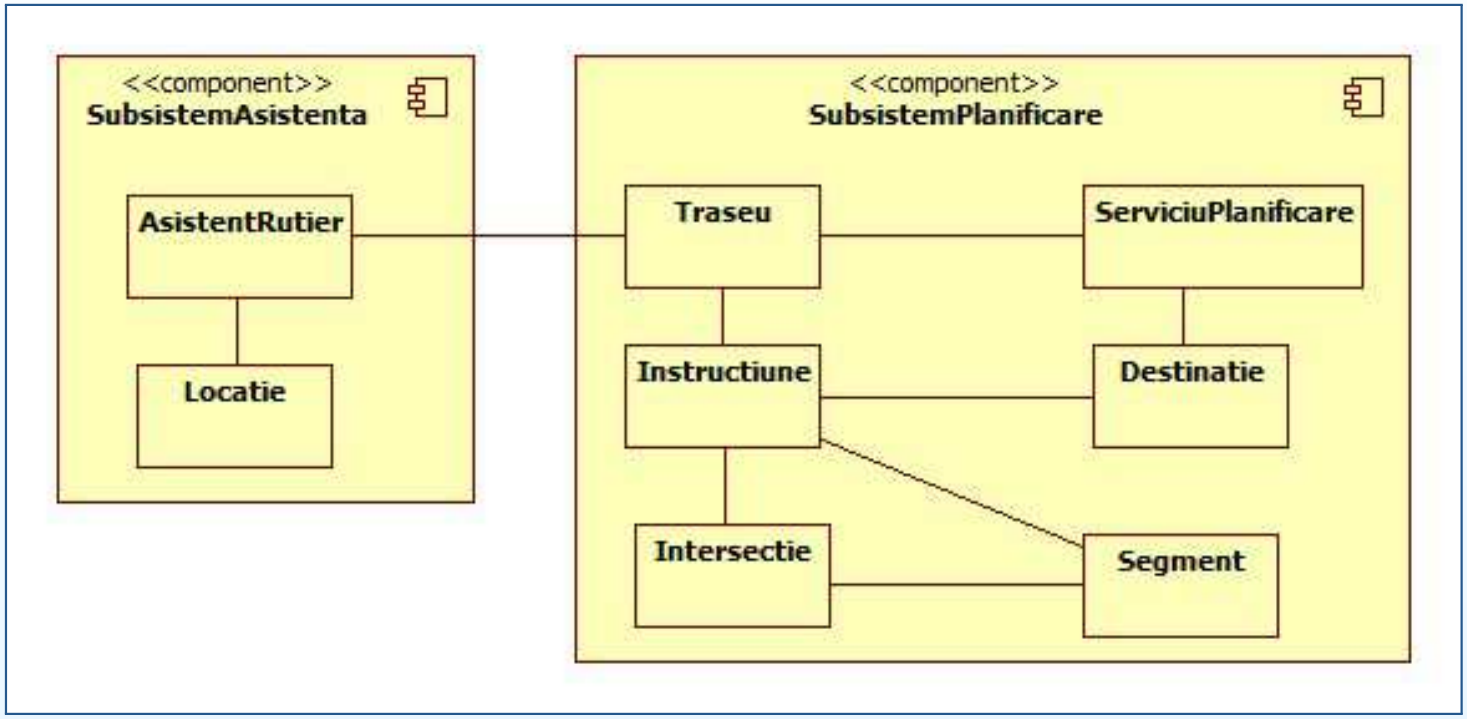
MyTrip: Obiective de proiectare

- *Fiabilitate*: Sistemul *MyTrip* trebuie să fie fiabil [generalizare a cerinţei nefuncţionale 2]
- *Toleranţă la erori*: Sistemul *MyTrip* trebuie să funcţioneze şi în condiţiile pierderii conexiunii între sistemul de asistenţă şi serviciul web [reformulare a cerinţei funcţionale 2]
- *Securitate*: Sistemul *MyTrip* nu trebuie să permită accesul neautorizat la traseele salvate de un şofer [dedusă din domeniul problemei]
- *Modificabilitate*: Sistemul *MyTrip* trebuie să poată fi modificat pentru a utiliza un alt serviciu de planificare [anticipare a schimbării de către dezvoltatori]

Descompunerea inițială a sistemului

- Se realizează pe baza cerinţelor funcţionale, folosind stiluri arhitecturale predefinite
- Sistemul *MyTrip* - două grupuri de obiecte
  - cele implicate în cazul de utilizare *PlanificareTraseu*
  - cele implicate în cazul de utilizare *ExecutareTraseu*
  - Clasele *Traseu*, *Destinaţie*, *Instrucţiune*, *Segment*, *Intersecţie* (strâns cuplate, folosite pentru a reprezenta un traseu) sunt partajate între cele două cazuri de utilizare
- Descompunere *MyTrip* - stil arhitectural *Repository*, subsistemul de planificare este responsabil de crearea structurii de date centralizate
  - Subsistemul *SubsistemPlanificare* = *ServiciuPlanificare* + clasele folosite pentru reprezentarea traseului
  - Subsistemul *SubsistemAsistenta* = restul claselor
  - O singură dependenţă între cele două subsisteme

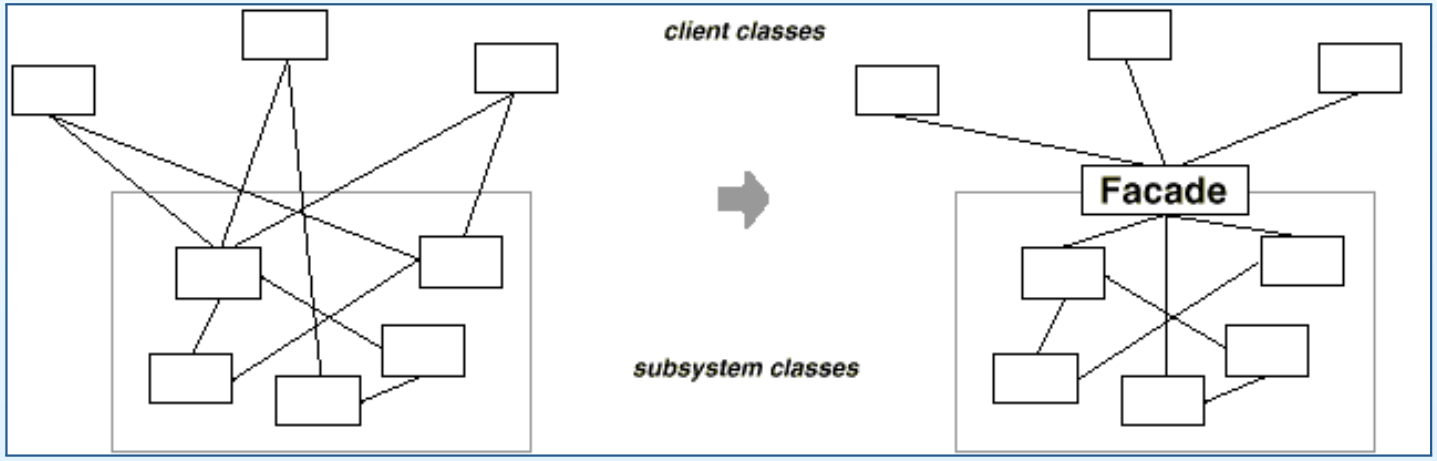
MyTrip: Subsisteme şi responsabilităţi iniţiale



SubsistemPlanificare	Este responsabil de construirea unui *Traseu* ce uneşte o serie de *Destinaţii*. Răspunde la cereri de replanificare din partea *SubsistemuluiAsistenta*.
SubsistemAsistenta	Este responsabil de descărcarea unui *Traseu* din *ServiciulPlanificare* şi de executarea acestuia, prin furnizarea de *Instrucţiuni* şoferului, pe baza *Locaţiei* curente.

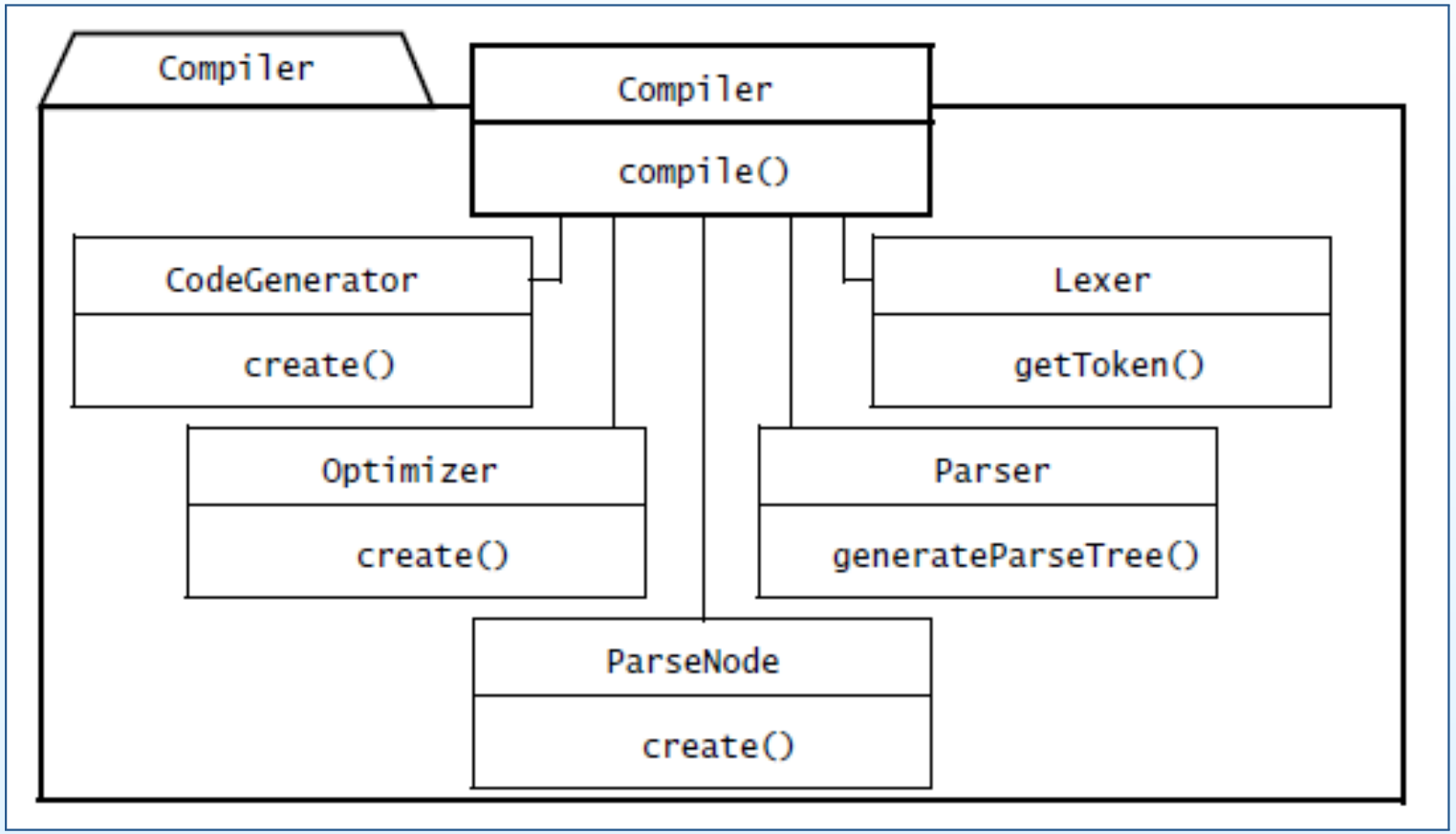
Descompunerea iniţială a sistemului (cont.)

- Euristici pentru descompunerea iniţială a sistemului în subsisteme
  - Asignează clasele identificate la nivelul unui caz de utilizare aceluiaşi subsistem
  - Creează un subsistem dedicat pentru clasele utilizate pentru transportul datelor între subsisteme, sau asignează acele clase subsistemelor responsabile de crearea lor
  - Toate clasele dintr-un subsistem trebuie să fie înrudite funcţional
  - Minimizează numărul de asocieri ce depăşesc graniţele subsistemelor
- Şablonul de proiectare *Facade* (structural) [Gamma et al., 1994]
  - Permite reducerea dependeţelor (cuplării) dintre un subsistem şi clienţii săi



Exemplu de aplicare a şablonului *Facade*

- Clasa *Compiler* reprezintă o faţadă ce ascunde clasele *Lexer*, *Parser*, *ParseNode*, *Optimizer*, *CodeGenerator*



Şablonul *Facade*

- *Definiţie*
  - Oferă o interfaţă unificată, de nivel înalt, pentru un grup de interfeţe ale unui subsistem, care facilitează utilizarea acestuia
- *Aplicabilitate*
  - Oferirea unei interfeţe simple către un subsistem complex
  - Diminuarea numărului de dependenţe între clienţi şi clasele de implementare ale unei abstractizări
  - Stratificarea unui subsistem
- *Participanţi*
  - *Facade* (*Compiler*)
    - Ştie ce clase ale subsistemului sunt responsabile de o cerere
    - Delegă cererile clientului către obiectele corespunzătoare din subsistem
  - Clasele subistemului (*Lexer*, *Parser*, ...)
    - Implementează funcţionalitatea subsistemului
    - Rezolvă sarcinile atribuite de obiectul faţadă
    - Nu ştiu de existenţa faţadei (nu păstrează referinţe spre ea)

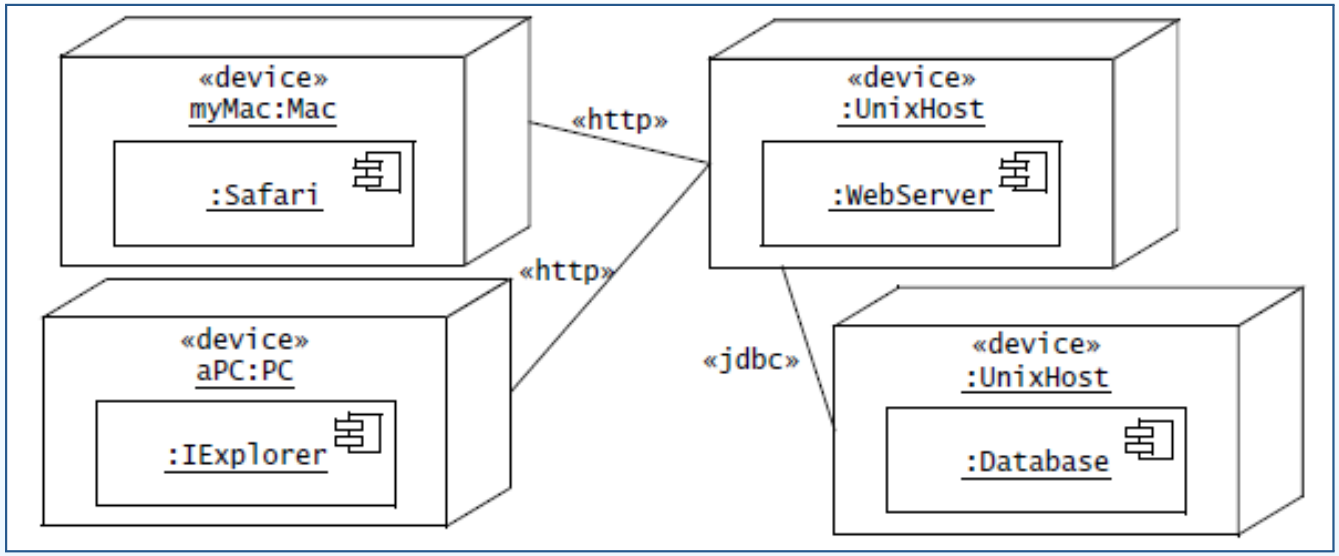
Şablonul *Facade* (cont.)

- *Colaborări*
  - Clienţii comunică cu subsistemul trimiţând cereri către obiectul faţadă, care le transmite către obiectele corespunzătoare din subsistem. Deşi obiectele din subsistem efectuează munca reală, faţada ar putea fi obligată să efectueze sarcini proprii, pentru a translata interfaţa sa în interfeţele subsistemului
  - Clienţii care utilizează faţada nu trebuie să acceseze direct obiectele din subsistemul acesteia
- Şablonul nu interzice clienţilor să utilizeze direct clasele subsistemului, dacă acest lucru este necesar

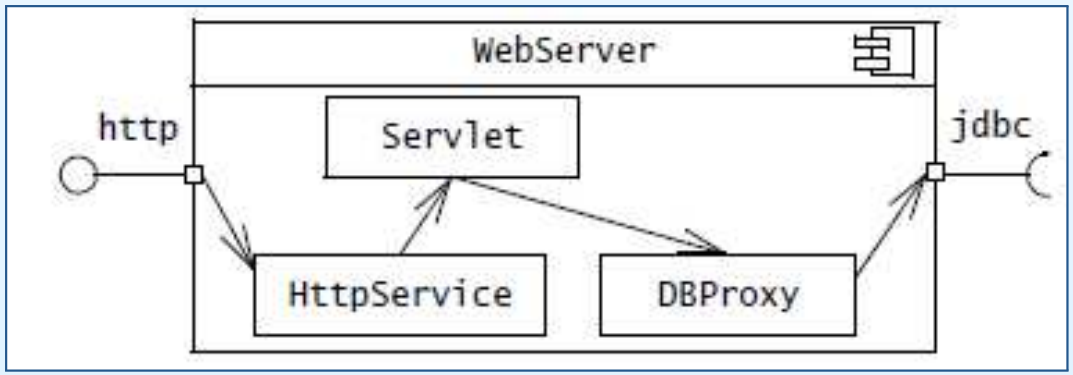
Diagrame de repartiţie a resurselor

- La runtime, un sistem este reprezentat de o mulţime de componente care interacţionează, distribuite la nivelul unor noduri
- O *diagramă de repartiţie UML* (eng. *UML deployment diagram*) ilustrează relaţia dintre componentele runtime şi noduri
  - *Componentă* = unitate autoconţinută, care oferă servicii altor componente sau actorilor
    - Un server web este o componentă care oferă servicii browserelor web
    - Un browser web este o componentă care oferă servicii utilizatorilor
  - *Nod* = dispozitiv fizic (calculator) sau mediu de execuţie pentru componente
    - Un nod poate conţine un alt nod (un dispozitiv conţine un mediu de execuţie)
- Reprezentare
  - Nodurile sunt reprezentate ca şi paralelipipede dreptunghice, ce conţin componente
  - Nodurile pot fi stereotipizate pentru a indica un dispozitiv fizic sau un mediu de execuţie
  - Comunicarea dintre noduri se reprezintă cu o linie continuă, ce poate fi stereotipizată cu un nume de protocol

Exemplu de diagramă UML de repartiţie

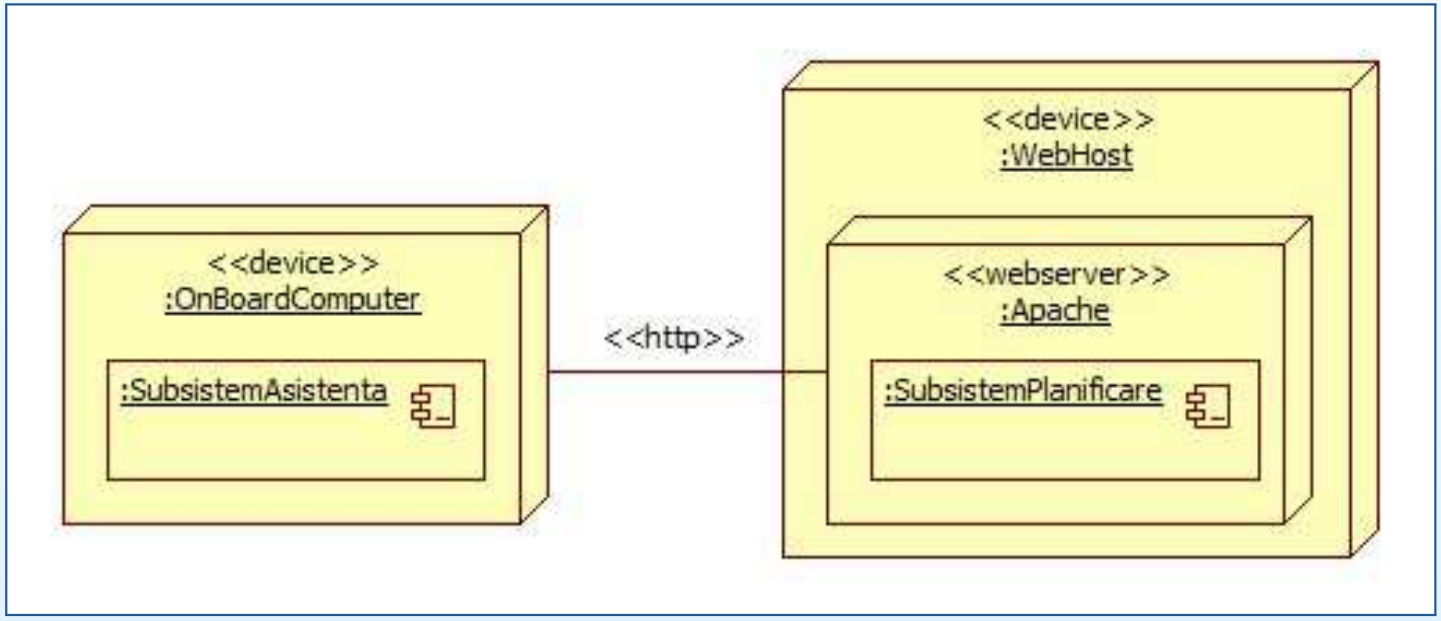


- Componentele pot fi rafinate, pentru a include informaţii relativ la clasele pe care le conţin şi la interfeţele pe care le oferă/solicită



Maparea subsistemelor la hardware

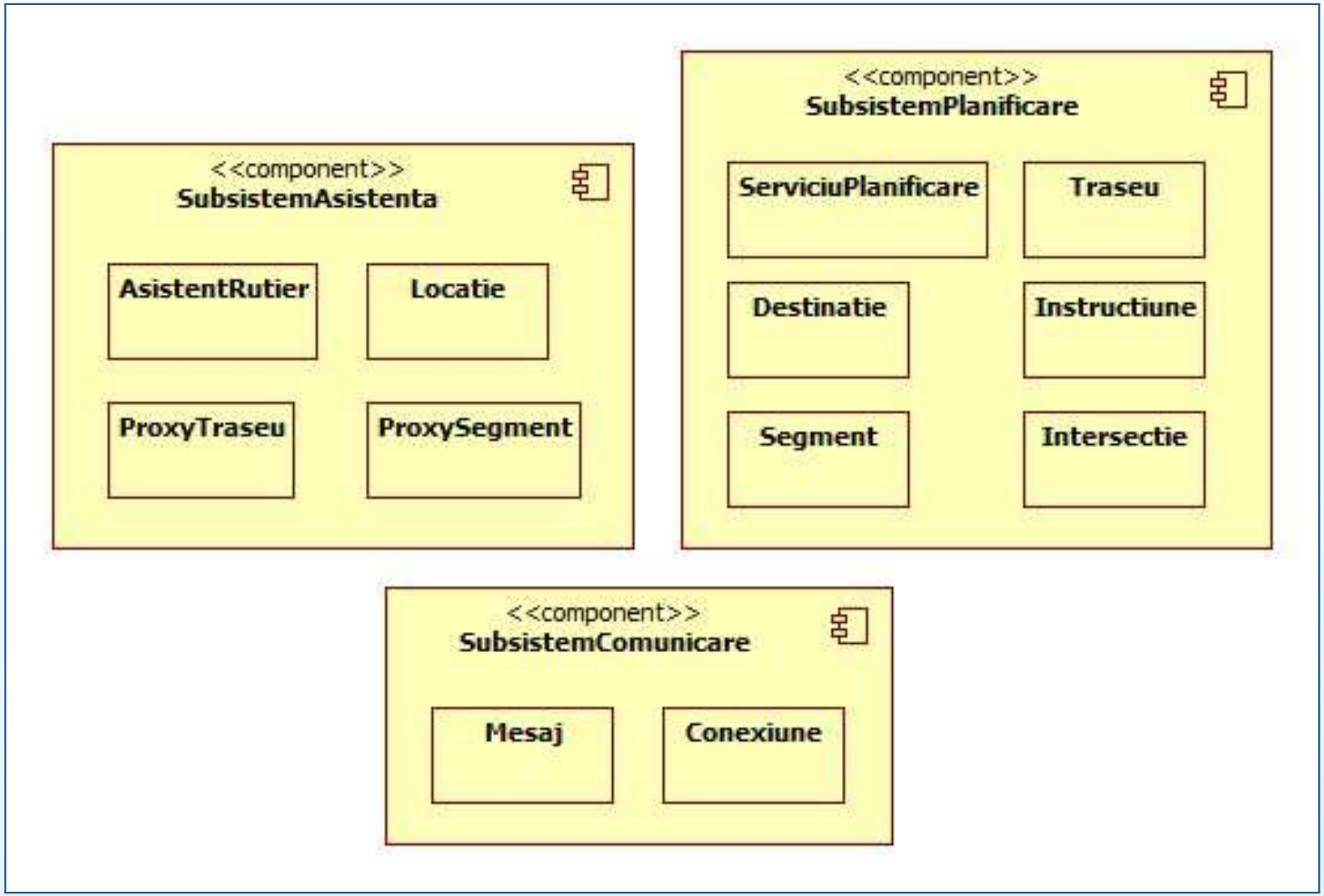
- Strategia de alocare a subsistemelor pe echipamentele hardware şi proiectarea infrastructurii de comunicare între aceste subsisteme influenţează semnificativ performanţele sistemului şi complexitatea acestuia
  - Selectarea configuraţiei hardware include şi selectarea maşinii virtuale pe care va fi construit sistemul
- Alocarea subsistemelor *MyTrip*



Maparea subsistemelor la hardware (cont.)

- Alocarea subsistemelor *MyTrip*
  - Subsistemul *SubsistemPlanificare* rulează pe un server web (*WebHost*), în mediul de execuţie Apache. Maşina virtuală e o maşină Unix
  - Subsistemul *SubsistemAsistenţă* rulează pe computerul de bord al maşinii (*OnBoardComputer*), maşina virtuală fiind un browser web
- Alocarea subsistemelor pe noduri, implică, de obicei, identificarea unor noi clase/subsisteme utilizate pentru transportul datelor între aceste noduri
  - În cazul *MyTrip*, obiectele aferente unui traseu (*Traseu*, *Instructiune*, *Intersecţie*, *Segment*, *Destinaţie*) trebuie transportate de la subsistemul de planificare spre cel de asistenţă
  - Pentru a gestiona comunicarea între aceste două subsisteme, se introduce un subsistem nou *SubsistemComunicare*, ce va fi localizat pe ambele noduri

MyTrip: prima rafinare a descompunerii iniţiale

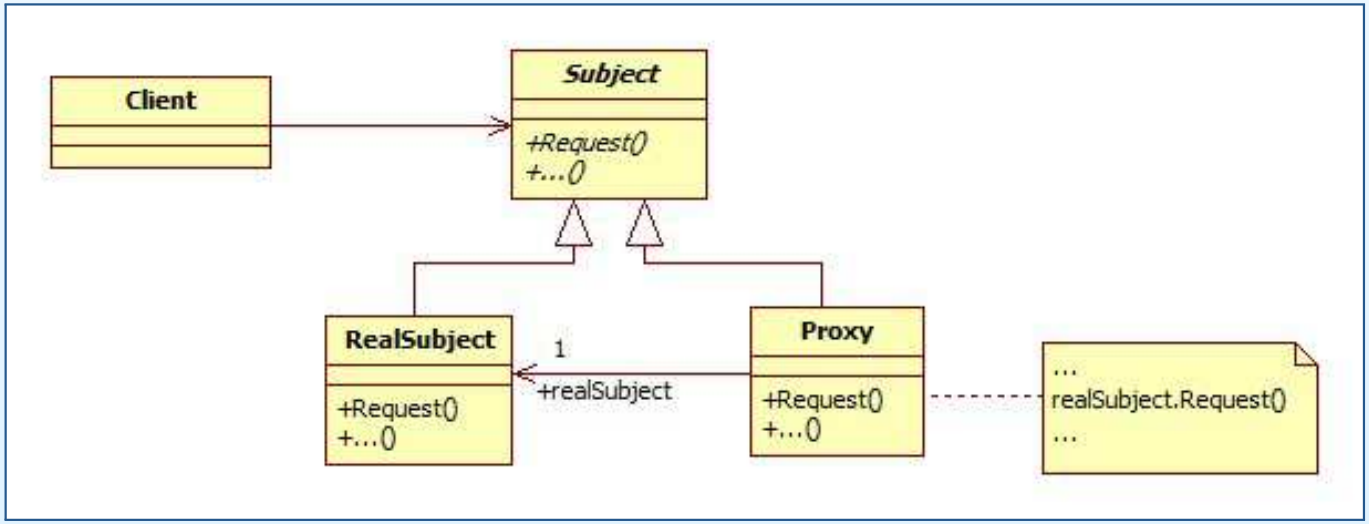


MyTrip: prima rafinare a descompunerii iniţiale (cont)

SubsistemComunicare	Responsabil de transportarea obiectelor de la *SubsistemulPlanificare* la *SubsistemulAsistenta*
Conexiune	O *Conexiune* reprezintă o legătură activă între subsistemul de planificare şi cel de asistenţă. Un obiect *Conexiune* tratează cazurile excepţionale cauzate de pierderea conexiunii în reţea
Mesaj	Un *Mesaj* reprezintă un *Traseu* şi elementele asociate (*Instructiune*, *Destinatie*, *Segment*, *intersecţie*), codificate pentru transport

Şablonul *Proxy* (stuctural) [Gamma et al., 1994]

- *Definiţie*
  - Şablonul asigură, pentru un obiect, un surogat sau un înlocuitor, în scopul controlării accesului la acesta
- *Aplicabilitate*
  - *Proxy la distanţă* - oferă un reprezentant local pentru un obiect dintr-un spaţiu de adresă diferit
  - *Proxy virtual* - creează la cerere obiecte costisitoare
  - ...
- *Structură*



Şablonul *Proxy (cont.)*

- *Participanţi*
  - *Proxy*
    - Păstrează o referinţă care îi permite să accceseze subiectul real
    - Asigură o interfaţă identică celei a clasei *Subject*, astfel încât un obiect proxy să poată înlocui un obiect real
    - Controlează accesul la subiectul real şi poate răspunde de crearea şi ştergerea acestuia
    - Obiectele *proxy la distanţă* răspund de codificarea unei cereri şi a argumentelor acesteia şi de transmiterea cererii codificate către subiectul real, aflat într-un spaţiu de adresă diferit
    - Obiectele *proxy virtuale* pot depozita informaţie suplimentară despre subiectul real, astfel încât să poată amâna accesarea acestuia (ex.: un obiect *ImageProxy* poate stoca dimensiunile unui obiect *RealImage*)
  - *Subject*
    - Defineşte interfaţa comună pentru obiectele *RealSubject* şi *Proxy*, astfel încât un obiect *Proxy* să poată fi utilizat în orice loc în care este aşteptat un obiect *RealSubject*

Şablonul *Proxy (cont.)*

- *RealSubject*
  - Defineşte obiectul real reprezentat de obiectul proxy

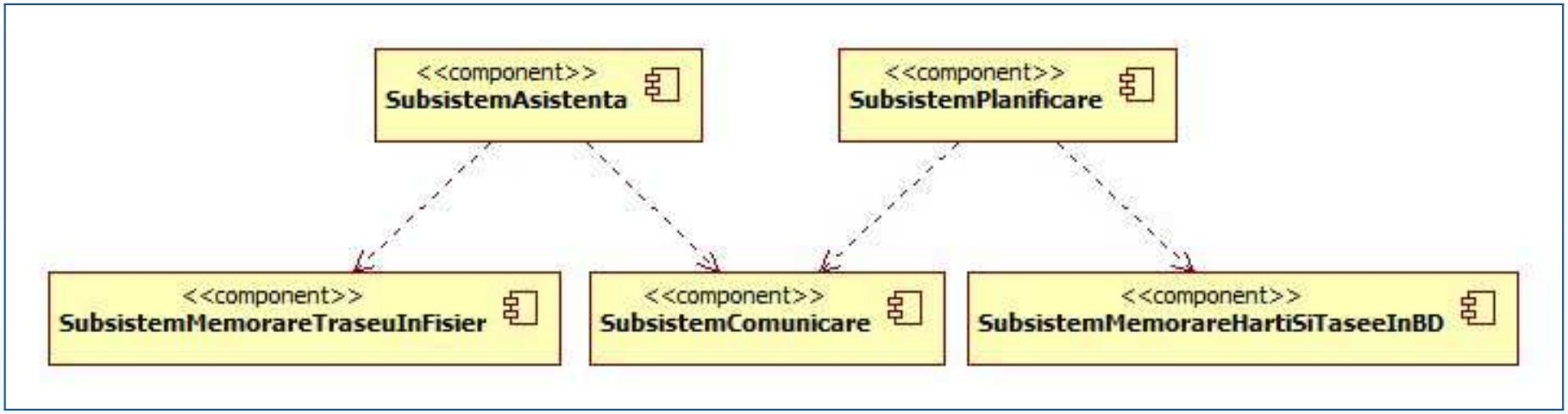
- *Colaborări*
  - Când se impune, obiectul proxy transmite cererile către subiectul real, funcţie de tipul de proxy

- *Consecinţe*
  - Şablonul *Proxy* introduce un grad de "ocolire" la accesul unui obiect. Semnificaţia acestei ocoliri depinde de tipul de proxy: un proxy la distanţă ascunde faptul că un obiect se află într-un alt spaţiu de adresă; un proxy virtual poate efectua optimizări, cum ar fi crearea unui obiect la cerere (un proxy pentru o imagine instanţiază obiectul imagine concret doar atunci când este necesară afişarea imaginii)

Gestiunea datelor persistente

- Unele dintre obiectele ce compun sistemul trebuie să fie *persistente*
  - Valorile atributelor acestora au o durată de viaţă ce o depăşeşte pe cea a unei singure execuţii a sistemului
- Persistenţa obiectelor se poate realiza folosid
  - Fişiere - în cazul în care există mai mulţi cititori, dar un singur scriitor
  - Baze de date - în cazul în care datele sunt accesate de cititori şi scriitori concurenţi
- Strategia de persistenţă pentru sistemul *MyTrip*
  - Traseul curent este salvat într-un fişier, pentru a permite recuperarea acestuia în cazul în care şoferul opreşte maşina înainte de a ajunge la destinaţia finală
  - Toate traseele aferente subsistemului de planificare, precum şi hărţile necesare generării acestora sunt memorate la nivelul unei baze de date
  - Strategia de persistenţă aleasă determină introducerea în sistem a două noi subsisteme

MyTrip: a doua rafinare a descompunerii iniţiale



SubsistemMemorareTraseuInFisier	Responsabil de memorarea traseelor în fişiere pe calculatorul de bord. Deoarece această funcţionalitate este folosită doar pentru a salva trasee atunci când maşina se opreşte, acest subsistem suportă doar memorarea şi încărcarea rapidă a unor trasee complete.
SubsistemMemorareHartiSiTraseeInBD	Responsabil de memorarea hărţilor şi a traseelor într-o bază de date pentru subsistemul de planificare. Acest subsistem suportă accesul concurent al mai multor şoferi şi agenţi de planificare.

Gestiunea datelor persistente: activităţi

- *Identificarea obiectelor persistente*
  - Date candidat pentru persistenţă
    - Entităţile (nu neaparat toate) identificate în etapa de analiză (pentru sistemul *MyTrip*: *Destinaţie*, *Intersecţie*, *Segment*, *Traseu*; *Locaţie* şi *Instrucţiune* se recalculează funcţie de poziţia curentă a maşinii => nu trebuie persistate)
    - Informaţii legate de utilizatori, într-un sistem multi-utilizator (*Şofer*i, spre exemplu), atribute ale unor obiecte *boundary* ce reţin preferinţe ale utilizatorilor, etc.
    - În general, toate clasele ale căror obiecte trebuie să supravieţuiască unei opriri a sistemului, controlată sau neaşteptată
- *Selectarea strategiei de memorare*
  - Decizie complexă, determinată, în general, de cerinţe nefuncţionale: Obiectele trebuie regăsite rapid? Sunt necesare interogări complexe pentru regăsirea acelor obiecte? Obiectele necesită mult spaţiu pentru memorare?
  - În general, 3 opţiuni

Gestiunea datelor persistente: activităţi (cont.)

- *Fişiere*
  + Avantaje: permit o mare varietate de optimizări de viteză şi dimensiune
  - Dezavantaje: lasă pe umerii aplicaţiei gestiunea accesului concurent sau recuperarea datelor în urma unei căderi a sistemului
- *Baze de date relaţionale*
  + Avantaje: tehnologie matură, ce oferă servicii pentru controlul accesului, gestiunea concurenţei şi recuperarea datelor în urma unei căderi a sistemului
  - Dezavantaje: deşi scalabile şi ideale pentru volume mari de date, sunt relativ lente pentru seturi de date mici sau date nestructurate (imagini, text în limbaj natural); în plus, există necesitatea mapării obiect-relaţionale
- *Baze de date orientate obiect*
  + Avantaje: un nivel mai înalt de abstractizare: memorează datele ca şi obiecte, înlăturând necesitatea translatării între obiecte şi entităţile memorate
  - Dezavantaje: mai lente decât bazele de date relaţionale

Euristici de selectare a mecanismului de persistenţă

- Când se folosesc fişiere?
  - Date voluminoase (imagini)
  - Date temporare
  - Densitate mică a informaţiei (log-uri)
- Când se folosesc baze de date?
  - Acces concurent
  - Mai multe platforme sau aplicaţii care accesează aceleaşi date
- Când se folosesc baze de date relaţionale?
  - Interogări complexe
  - Seturi mari de date
- Când se folosesc baze de date orientate obiect?
  - Utilizare masivă a asocierilor pentru regăsirea datelor
  - Seturi de date medii

Definirea politicilor privind controlul accesului

- În sistemele multi-utilizator, actori diferiţi au, de obicei, drepturi de acces diferite la diferite funcţionalităţi şi date
- Cum modelăm accesul?
  - În timpul analizei - prin asocierea diferitor cazuri de utilizare la diferiţi actori
  - În timpul proiectării - prin determinarea acelor obiecte partajate între actori, precum şi a modului în care actorii pot controla accesul. Funcţie de cerinţele de securitate ale sistemului, definim şi modul în care actorii se autentifică în sistem, precum şi modalitatea de criptare a anumitor date
- Sistemul *MyTrip*
  - Se introduce o clasă *Şofer*, asociată clasei *Traseu*, pentru a asigura trimiterea traseelor doar la şoferii care le-au creat
  - Subsistemul de planificare devine responsabil de autentificarea şoferilor, înainte de a le trimite traseele solicitate
  - Subsistemul de comunicare criptează traficul dintre subsistemul de asistenţă şi cel de planificare

MyTrip - o nouă rafinare

Şofer	*Un Şofer reprezintă un utilizator autentificat. Este utilizat de către subsistemul de comunicare, pentru a memora cheia utilizată pentru criptare şi de către subsistemul de planificare, pentru a asocia Traseele cu utilizatorii.*
SubsistemPlanificare	Este responsabil de construirea unui *Traseu* ce uneşte o serie de *Destinaţii*. Răspunde la cereri de replanificare din partea *SubsistemuluiAsistenta*. *Înainte de a procesa o cerere, subsistemul de planificare autentifică Şoferul de la subsistemul de asistenţă. Şoferul autentificat este utilizat pentru a determina acele Trasee care pot fi trimise subsistemului de asistenţă*.
SubsistemComunicare	Responsabil de transferul obiectelor de la *SubsistemulPlanificare* la *SubsistemulAsistenta*. *Acesta utilizează Şoferul asociat Traseului care se transferă pentru a selecta o cheie şi a cripta traficul.*

Matrici de acces

- Accesul la clase poate fi modelat folosind *matrici de acces*
  - Liniile reprezintă actorii din sistem
  - Coloanele reprezintă clasele ale căror drepturi de acces dorim să fie modelate
  - O intrare în matrice listează operaţiile care pot fi executate pe o instanţă a clasei de pe coloană de către actorul de pe linie
- Variaţii de implementare
  - *Tabel de acces global* - reprezintă explicit fiecare triplet (actor, clasă, operaţie)
  - *Liste de control a accesului* - asociază o listă de perechi (actor, operaţie) fiecărei clase. De fiecare dată când o instanţă a clasei este accesată, se verifică existenţa perechii (actor, operaţie) în listă
  - *Capabilităţi* - asociază perechi (*clasă, operaţie*) unui actor

Stabilirea fluxului global de control

- *Flux de control* = secvenţierea acţiunilor într-un sistem (ce operaţii se execută şi în ce ordine)
- Mecanisme
  - *Flux de control procedural*
    - Operaţiile aşteaptă input ori de câte ori au nevoi de date de la un actor
    - Utilizat mai mult în limbajele procedurale
    - Util pentru testarea subsistemelor
  - *Flux de control dirijat de evenimente*
    - O buclă aşteaptă evenimente externe. Ori de câte ori un astfel de eveniment apare, este direcţionat către obiectul aferent, pe baza unor informaţii asociate evenimentului
    - Avantaje: abordare matură, structură simplă, centralizarea input-urilor
  - *Thread-uri*
    - Varianta concurentă a controlului procedural
    - Sistemul poate crea un număr arbitrar de thread-uri, fiecare răspunzând la un eveniment diferit. Dacă un thread are nevoie de date suplimentare, aşteaptă input de la un actor
    - Avantaje: mecanism intuitiv; dezavantaje: dificil de testat şi depanat

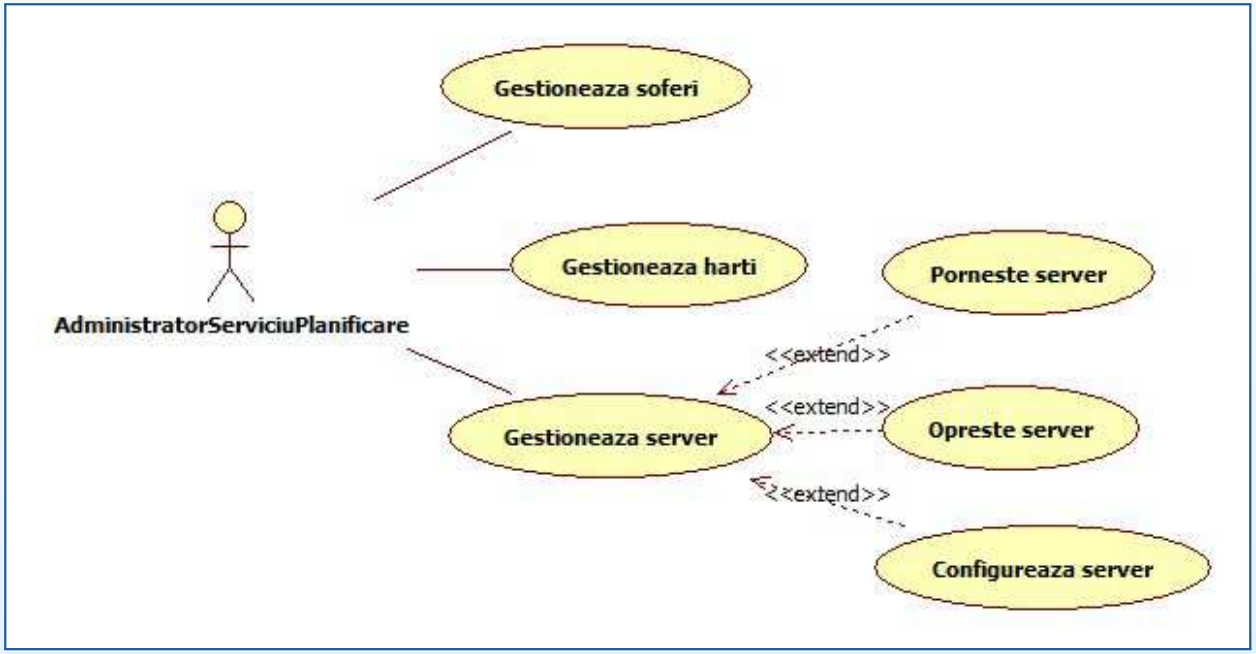
Descrierea cazurilor limită

- Examinarea condiţiilor limită
  - Cum este sistemul pornit, iniţializat şi oprit?
  - Cum sunt gestionate datele corupte sau căderile de reţea?
  - Cazurile de utilizare care gestionează aceste condiţii se numesc *cazuri de utilizare limită* (eng. *boundary use cases*)
- Sistemul *MyTrip*
  - Cum sunt încărcate hărţile în serviciul de planificare?
  - Cum este instalat sistemul într-o maşină?
  - De unde cunoaşte sistemul *MyTrip* serviciul de planificare la care trebuie să se conecteze?
  - Cum sunt adăugaţi şoferii în serviciul de planificare?
- Euristici pentru determinarea cazurilor limită
  - *Configurare*: Pentru fiecare obiect persistent, se identifică acele cazuri de utilizare în care obiectul este creat/ distrus. Pentru obiectele care nu sunt create/distruse în cazurile de utilizare existente (ex.: hărţile în sistemul *MyTrip*), se adaugă un caz de utilizare gestionat de un administrator al sistemului (ex.: *GestionareHarti*)

Descrierea cazurilor limită (cont.)

- *Pornire/oprire*: Pentru fiecare componentă, se adaugă trei cazuri de utilizare pentru a porni, opri şi configura componenta
- *Gestiunea excepţiilor*: Pentru fiecare tip de eşec (ex.: cădere de reţea), se creează un caz de utilizare excepţional, care extinde un caz de utilizare existent (alternativa a utilizării scenariilor de excepţie)

- Cazuri de utilizare limită pentru *MyTrip*



Referinţe

- [Gamma et al., 1994] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, 1994.