Arhitectura sistemelor soft enterprise. Platforma .NET

Curs 5

Starea sesiunii. Strategii de distribuire.
Rezumat al partii narative

Starea sesiunii (1)

- Bibliografie: cap 6 din PoEAA
- Cum se pastreaza datele care fac parte dintr-o sesiune de business, date care nu sunt inca pregatite sa fie scrise (eng: committed) in BD?
- Nu toate sesiunile de lucru necesita pastrarea starii - *e.g.* vizionarea paginilor unui site de stiri
- In OOP: un obiect = stare (campuri) + comportament (metode)
- *Obiect* fara stare = obiect fara campuri, dar rar intalnit (de regula implementeaza niste servicii ce nu necesita stare: conversii de valori/reprezentare, logging etc.)
- *Server* fara stare = "obiect" fara stare, care raspunde la diferite cereri pentru servicii
  - Serverele Web sunt servere fara stare

Starea sesiunii (2)

- Exemplu: site care prezinta carti; cerere = adresa URL catre un "obiect" (resursa) de tip pagina web statica, servlet, pagina web dinamica etc.
- Daca este pagina web dinamica: posibil ca sa se foloseasca obiecte pentru a formula raspunsul din baza de date, posibil business logic (e.g. prezentarea de recenzii sortate dupa feedbackul primit, prezentare de "ce altceva s-a mai cumparat impreuna cu cartea X" – vezi "association analysis")
- Odata trimisa pagina catre browser, obiectele folosite sunt disponibilizate => server fara stare
- Nici macar cererea repetata a aceleiasi pagini de catre acelasi client nu duce la reutilizarea obiectelor precedente

Starea sesiunii (3)

- Cerinta: se doreste pastrarea listei de produse pe care o persoana le-a vizualizat (in scop de reclama contextuala, sau pentru a permite utilizatorului sa revina la istoricul preferintelor)
- Lista rezultata trebuie sa fie persistata pe server => necesitatea de a folosi un *server cu stare*
- Asta presupune consum de resurse de pe server = memorie pentru mentinerea listelor
- Exemplu: 100 de persoane, 1 cerere/persoana la fiecare 10 sec, procesarea unei cereri pe server = 1 sec; presupunem cereri balansate = distributie uniforma
  - in varianta server cu stare avem nevoie de 100 de liste mentinute simultan, care 90% din timp sunt nefolosite pe server = consum de resurse, utilizare slaba;
  - in varianta fara mentinerea starii => 10 obiecte (pagini de raspuns) sunt folosite, in medie, intr-o secunda

Starea sesiunii (4)

- Daca se mentine starea, atunci la fiecare cerere trebuie folosit obiectul adecvat: fiecare utilizator necesita propriul sau istoric
- HTTP este un protocol ce nu mentine starea (cererea este preluata si i se raspunde; dupa ce se da raspunsul, resursele folosite sunt disponibilizate)
- Insa: ***se poate folosi un server fara stare si totusi sa se mentina starea sesiunii***
- Exemplu clasic de stare: cosul de cumparaturi sau wish list pe un site de e-commerce
- Datele dintr-un cos sunt relevante doar pentru o sesiune; operarea cu cosul este considerata o **tranzactie de business**; cosul poate fi salvat in baza de date si doar atunci se transforma in inregistrari

Starea sesiunii (5)

- Tranzactie de business: trebuie sa satisfaca ACID
  - C = Consistenta; datele trebuie sa fie intr-o stare consistenta la comiterea tranzactiei
  - Consistenta este data de regulile de business = logica ceruta aplicatiei
    - exemplu: numarul de exemplare dintr-o carte cumparata trebuie sa fie un intreg strict pozitiv &&
    - numarul de carti cumparate trebuie sa fie cel mult egal cu numarul de exemplare disponibile
  - Alt exemplu: se editeaza o polita de asigurare, trebuie precizate multe valori; alterarea unei valori pe UI (e.g: judetul) poate insemna ca serverul sa fie solicitat pentru valori dependente (e.g: localitati din judet), pentru a asigura consistenta;
  - Pe toata durata editarii, starea tranzactiei poate fi *neconsistenta*; consistenta trebuie verificata sau asigurata atunci cand se vrea persistarea datelor in BD

Starea sesiunii (6)

- Nu e obligatoriu ca toate datele dintr-o tranzactie de business sa fie persistate in tranzactie sistem
  - Motiv: nu toate datele dintr-o sesiune sunt utile pentru starea sesiunii; unele sunt folosite ca auxiliare (tipul de client, istoricul recent etc.)
- A nu se confunda starea sesiunii - date absolut necesare pentru bunul mers al aplicatiei dpdv al unui utilizator - cu caching-ul datelor - date care imbunatatesc performanta de raspuns a aplicatiei, dar pierderea lor nu afecteaza functionarea ei
- Modalitati de mentinere a sesiunii:
  - Client session state – date stocate pe client
  - Server session state – date stocate in memoria serverului de aplicatie
  - Database session state – date stocate in serverul de baze de date

Starea sesiunii (7)

- Client session state: starea e mentinuta de catre client, prin:
  - Web: cookies (**Tema**: care e dimensiunea maxima a unui cookie? Care e numarul maxim de cookie-uri care se mentin de catre browser? Sunt aceste aspecte dependente de tipul de browser?)
  - Web: URL: query-string-ul "plimbat" intre client si server mentine starea: http://www.google.ro/#**hl=ro&num=100&newwindow=1&q=fowler&start=200&sa=N&fp=adbac09bb9184e4d**
  - Web: Folosirea unui camp ascuns in pagina (<input type= "hidden"… />)
    - Caz particular remarcabil: ViewState mentinut de catre ASP.NET web forms in campul ascuns __VIEWSTATE; poate fi folosit din "code behind" pentru a stoca perechi (string=valoare serializata)
  - Rich client (~~Silverlight~~, ~~Flash~~, HTML5 –> local storage): date mentinute in obiecte gestionate de browser

Starea sesiunii (8)

- Client session state:
  - Pentru pagini HTML: la fiecare cerere, obiectele mentinute de client sunt trimise serverului pentru a putea fi procesate
  - => consum de latime de banda
  - Se poate pune problema compactarii datelor: XML e mai stufos decat JSON pentru serializare
    - XML vs JSON: http://json.org/xml.html
    - XML:

<cos><carte><id>3123</id><cantitate>2</cantitate></carte><carte><id>3124</id><cantitate>1</cantitate></carte></cos>

    - JSON:

{"cos":{"carte":[{"id":"3123","cantitate":"2"},{"id":"3124","cantitate":"1"}]}}

    - Considerati protocol buffers ca alternativa pentru comunicare de date:
      - ” Protocol Buffers (Protobuf) is a free and open-source cross-platform data format used to serialize structured data. It is useful in developing programs to communicate with each other over a network or for storing data. “ Wikipedia, accesat noiembrie 2022

Starea sesiunii (9)

- **Client session state:**
  - Atentie la posibilitatea unui utilizator de a vedea si altera continutul starii (e.g. Fiddler, https://www.telerik.com/fiddler) => uneori se impune criptarea datelor de pe client + validarea la server a datelor primite
- **Server session state:**
  - Starea se mentine de catre server, de regula in memoria RAM
  - Obiectele pot fi de orice tip, nu doar cele ce permit serializare
  - Sesiunile sunt izolate
  - Trebuie mecanism prin care sa se faca recunoasterea sesiunii asociate unui browser
  - Problema la acest tip de sesiune: cluster de calculatoare (web farm): cum asiguri migrarea sesiunii de pe un calculator pe altul?
    - ASP.NET: stocarea sesiunii in server de BD sau pe o masina dedicata – serviciul Windows ASP.NET State Service

Starea sesiunii (10)

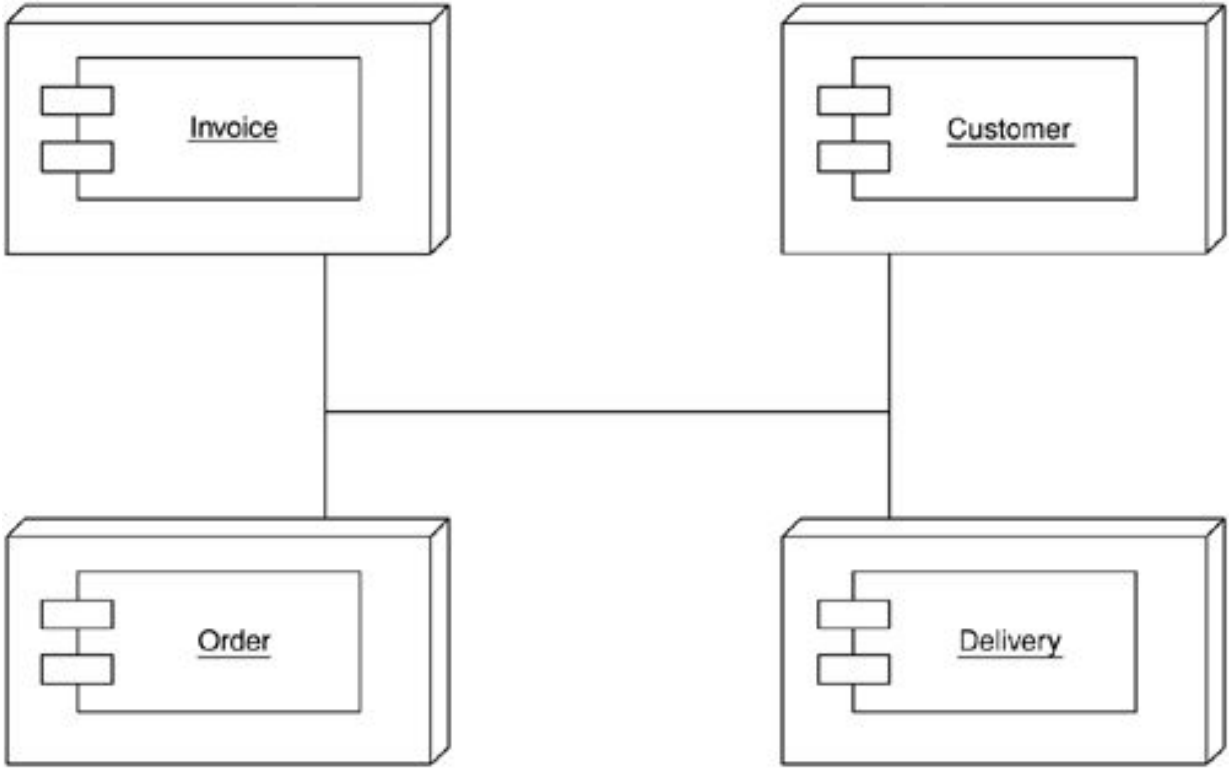
- **Server session state:**
  - Asigurare de incarcare echilibrata (load balancing); domeniu de cercetare, https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=load+balancing
  - Daca starea e mentinuta in afara procesului ce raspunde la cerere, atentie la penalizarile datorate comunicarii cu procesul ce mentine starea – frecvent: comunicare in mediu distribuit
- **Database session state:**
  - Sesiunea se salveaza intr-o unica baza de date
  - Se poate folosi si in conjunctie cu web farm
  - Presupune transformarea din inregistrarile din baza de date in forma obiectuala
  - Atentie la complexitatea implementarii: ASP.NET + SQL Server pentru Database session state = suport built-in; daca se vrea alt server atunci trebuie implementare de provider

Starea sesiunii (11)

- Ce se intampla cand **clientul (browser-ul)** renunta voluntar la sesiune?
  - Client session state: starea se pierde odata cu inchiderea browserului (exceptie: cookie-uri persistente)
  - Server session state: starea expira dupa o anumita perioada (timeout configurabil in aplicatia web)
  - In baza de date: trebuie facuta o curatare periodica
- Ce se intampla cand utilizatorului i se restarteaza browser-ul/se pierde conexiunea/un server se reporneste?
  - Daca vrei sa refaci automat sesiunea -> Database session state sau Server Session state cu server de stare dedicat
  - Daca accepti ca sesiunea e pierduta si se va reface de catre utilizator: oricare din Client Session State sau Server Session State
- Cele trei variante se pot mixa, dar atentie la controlul redundantei datelor de sesiune

Strategii de distribuire (1)

- Bibliografie: cap 7 din PoEAA
- Distribuire = punerea obiectelor rezultate pe mai multe calculatoare in retea
- Prima idee:



Strategii de distribuire (1)

- Motivatie (neinspirata): performanta
- Aparent, framework-urile dedicate distribuirii permit apeluri intre obiecte la distanta in mod transparent (prin proxy design pattern, remote proxy), deci… de ce nu?
- De fapt: distribuirea claselor (1 clasa pe nod) e cea mai proasta alegere posibila
- Motivatie:
  - Apelul de metode merge cel mai repede cand totul se desfasoara intr-un singur proces
  - Comunicare cu alt proces ruland pe aceeasi masina = intarzieri mari, cu cateva ordine de marime mai lent
  - Comunicare prin retea = intarziere mult mai mare - considerati in principal latenta

Strategii de distribuire (2)

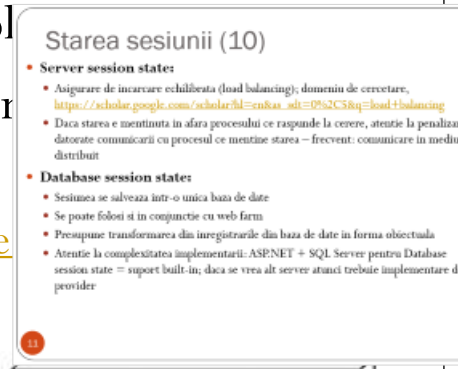
- Uneori, nu ai de ales: daca folosesti aplicatie multi-tier, obiectele trebuie sa fie distribuite
- In acest caz: interfata de comunicare la distanta nu trebuie sa fie la fel de granulara ca la apelurile dintre obiecte din cadrul unui proces
- Interfata locala are granularitate mica: clasa Adresa are
  - get/set oras
  - get/set nume strada
  - get/set cod postal
  - get/set judet
  - get/set numar imobil

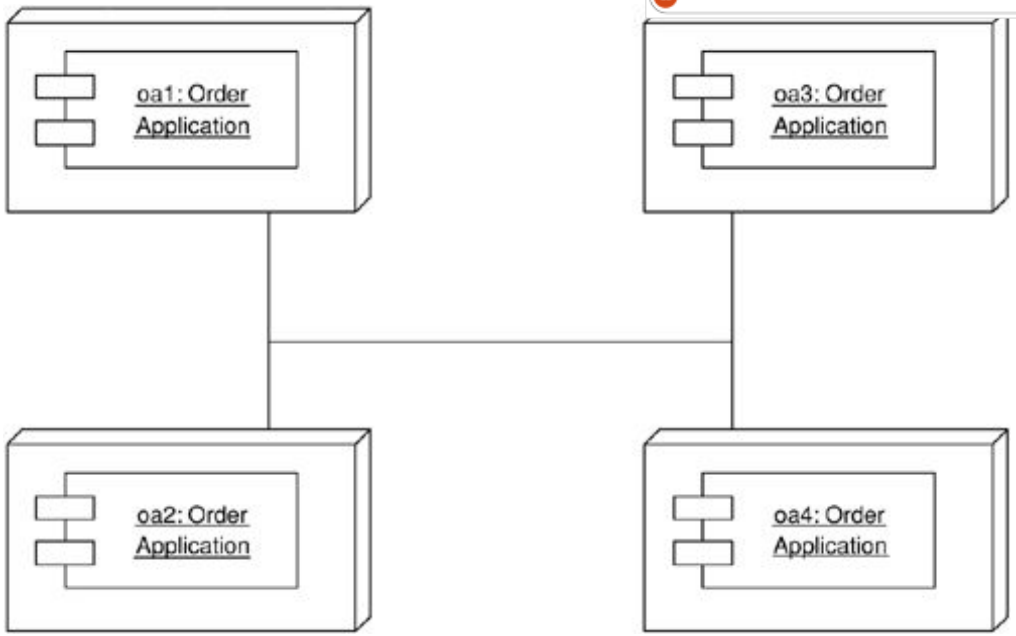
Strategii de distribuire (3)

- Interfata la distanta: apel de 5 metode pentru a obtine datele dintr-o adresa este costisitor
- Preferabil: 1 apel in care se obtin valorile de interes
- Obiecte cu interfata rugoasa (coarse grained interface): o metoda care da intregul obiect, sau seteaza toate valorile intr-un obiect
  - (Artificiu/compromis) Design-ul rugos este dictat de considerentele de performanta, nu de modelarea propriu-zisa
- Chiar si asa, distribuirea de clase pe masini diferite este o idee proasta, deoarece comunicarea devine preponderent bazata pe retea
- **First Law of Distributed Object Design: Don't distribute your objects!**

Strategii de distribuire (4)

- Cum faci uz de faptul ca ai mai multe masini pe care iti poti instala apl
- Toate clasele se pun intr-un singur proces si se distribuie procesul pe r calculatoare; o cerere va fi procesata in intregime de un nod oarecare
  - Cum faci "load balancing"? A se revedea slide cu mentiune de problema de





Strategii de distribuire (5)

- Variante de distribuire care vin ca impuneri:
  - Aplicatie client-server implica diviziune clasica inter-procese
  - Serverul de aplicatii <– >serverul de baze de date
  - Server Web <–> server de aplicatii – uneori cele doua nu se ruleaza in acelasi proces
  - Separarea poate fi datorata producatorilor diferiti; posibil chiar sa avem platforme diferite
- Cand nu ai de ales: la zona de comunicare intre procese se folosesc obiecte si/sau metode rugoase care actioneaza ca fatade pentru obiecte cu interfata fina; apel de metoda setAddress cu parametrii (city, street, county, region, zipcode) inseamna apel de multe metode din obiectele cu granularitate fina

Strategii de distribuire (6)

- Rezultat: remote façade
- Asociat cu remote façade: Data Transfer Object
  - Scop: transferul datelor la distanta
  - Un DTO este doar un container pentru date, nu are comportament/logica de domeniu;
  - DTO e definit la fel la ambele capete la care se face comunicarea
  - Rezulta ca un DTO e indicat sa contina referinte numai la alte DTO sau tipuri de date simple (String, int etc.)
  - Atentie la lazy loading! lazy loading presupune intarzierea apelurilor – care este costul mediu de penalizare? Tradeoff intre cat se transfera pe retea (eager loading) si numarul de apeluri/asteptarea indusa de apeluri la distanta (lazy loading)

Strategii de distribuire (7)

- Interfete pentru apeluri la distanta:
  - Remote procedure calls
  - XML over HTTP – servicii web:
    - SOAP - **Simple Object Access Protocol**
    - REST - **Representational state transfer, e.g. https://www.amazon.com/RESTful-Web-APIs-Services-Changing/dp/1449358063**
  - Serializare binara (e.g. protocol buffers) – mai compacta decat XML, dar ambele capete ale comunicarii trebuie sa agreeze forma binara
- Comunicare asincrona, bazata pe mesaje: Enterprise Integration Patterns – Designing, building and deploying messaging solutions

Rezumat al partii narative (1)

- Tehnici care se impun in dezvoltarea de soft, indiferent de deciziile arhitecturale:
  - Integrare continua (release-uri frecvente)
  - Test driven development
  - Refactoring - vezi de exemplu cartea "**Refactoring to Patterns**"
- Primul pas: Domain Layer – decizie intre:
  - Transaction script
  - Table Module
  - Domain Model
- Alegere: in functie de complexitatea problemei – insa complexitatea nu se poate cuantifica/prevedea exact

Rezumat al partii narative (2)

- Transaction script = model procedural
  - Confortabil pentru multi programatori
  - Usor de folosit cu o BD (ne)relationala, daca nu necesita transformari complexe de date
  - Nu rezista la un business logic complicat
  - Poate avea probleme de duplicare a codului
- Domain model
  - Suporta business logic complex
  - Se poate folosi si pentru probleme simple
  - Cer intelegerea programarii orientate pe obiect + design patterns (ultima fiind un prerequisite consistent, dar ajuta)

Rezumat al partii narative (3)

- Domain model
  - De regula nu exista o potrivire 1:1 intre modelul obiectual si cel relational => complexitate in legarea la baze de date
- Table module
  - Complexitatea este intre cele doua extreme
  - Manipuleaza logica de domeniu mai bine decat Transaction scripts, dar nu poate beneficia de toate mecanismele din OOP
  - Se potriveste bine cu BD relationale
  - Se potriveste cu tipurile de date din .NET (DataSet, DataTable) si mecanismul de data binding; alegerea poate sa fie influentatata si de uneltele de care dispui

Rezumat al partii narative (4)

- Data source layer
  - Pentru transaction scripts: row data gateway, table data gateway
  - Row data gateway: fiecare inregistrare produce un obiect => cod de scris, mult boilerplate code, se pot utiliza generatoare automate de cod
  - Table data gateway: mai putin cod; accesarea se face pe baza numelui de tabel/view/procedura stocata si a subsetului de date ce trebuie procesate
  - Pentru ambele variante de gateway: Nu e nevoie de unelte de transformare obiectual-relationala (O/RM)
  - Se poate folosi un Unit of Work, dar un script e de regula suficient de simplu pentru a pastra singur starea inregistrarilor

Rezumat al partii narative (5)

- Pentru Table Module:
  - Natural: Table data gateway
  - Util de adaugat mecanism de Unit of Work
- Domain Model
  - Active record, daca structura tabelelor se potriveste peste modelul obiectual
  - Daca domeniul este complex: data mapper => independenta domain model fata de reprezentarea datelor
  - Data mapper: scrierea de la zero e o sarcina dificila; e utila o unealta de O/RM

Rezumat al partii narative (6)

- Stratul de prezentare
  - Rich UI/HTML
  - Rich Internet Application: ~~Silverlight~~/~~Adobe Flash~~/JavaFX/Google Web Toolkit/Vaadin/Telerik/Vue.js/...
  - HTML 5 ("It was finalized, and published, on 28 October 2014 by the World Wide Web Consortium (W3C).", Wikipedia)
    - The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash and Microsoft Silverlight.
  - Indiferent de varianta: Model-View-Controller
  - XHTML: Template View, Page Controller (mix de pagini statice si dinamice), Transform View (e.g. XML+XSLT => continut tip text)

Rezumat al partii narative (7)

- Sfaturi tehnice
  - Java: de citit din bibliografie
  - .NET: familiarizare cu uneltele puse la dispozitie de ADO.NET, in special pentru implementare de Table Data Gateway/data binding
  - Familiarizare cu bibliotecile de tip O/RM
    - .NET: Entity Framework, NHibernate;
    - Java: Java Persistence API
  - Pentru diverse chestiuni tehnice, e.g.. proceduri stocate: pro si contra – trebuie cunoscute motivele
  - Intelegerea utilitatii view-urilor = ascunderea complexitatii de design
  - Familiarizarea cu serviciile web, cu transmiterea de mesaje asincrone
    - **Message broker –** Apache Active MQ, utilizabil in conjunctie cu Java, C++, .NET, Python, PHP
    - REST vs SOAP over HTTP? avantaje/dezavantaje?

Rezumat al partii narative (7)

- Exista si alte scheme de stratificare, dar nu esential diferite:

Table 8.1. Brown Layers	
Brown	**Fowler**
Presentation	Presentation
Controller/mediator	Presentation (Application Controller (379))
Domain	Domain
Data mapping	Data source (Data Mapper (165))
Data source	Data source

Rezumat al partii narative (9)

Table 8.4. Marinescu Layers

Marinescu	Fowler
Presentation	Presentation
Application	Presentation (Application Controller (379))
Services	Domain (Service Layer (133))
Domain	Domain (Domain Model (116))
Persistence	Data source

86

Table 8.5. Nilsson Layers

Nilsson	Fowler
Consumer	Presentation
Consumer helper	Presentation (Application Controller (379)
Application	Domain (Service Layer (133))
Domain	Domain (Domain Model (116))
Persistence access	Data source
Public stored procedures	Data source (may include some domain)
Private stored procedures	Data source (may include some domain)

Rezumat al partii narative (10)

- Resurse web:
  - The ServerSide: Java (www.theserverside.com), "Your Enterprise Java Community"
  - Blogs: https://www.joelonsoftware.com/, https://blog.fedecarg.com/ , https://medium.com/nick-tune-tech-strategy-blog/tagged/software-architecture
  - http://www.opengroup.org/
  - Universitar: https://www.ics.uci.edu/~yuzok/software-architecture.html
  - Reddit: https://www.reddit.com/r/softwarearchitecture/
  - Overview: "How Do Open Source Communities Document Software Architecture: An Exploratory Survey", http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6923128
  - Cunoasteti alte comunitati web legate de arhitectura enterprise? mailto:lmsasu@yahoo.com?subject=Comunitati%20dedicate%20Arhitecturarii%20enterprise