Arhitectura sistemelor soft enterprise. Platforma .NET

Conf. dr. Lucian M. Sasu

Facultatea de Matematica si Informatica

Master TMISS, 2023-2024

Organizare

- Curs: 14 saptamani, prezentari
- Laborator:
  - 6 saptamani prezentari de tehnologii (unit testing, logging, enterprise libraries, Entity framework, sabloane enterprise exemplificate etc.)
  - 6 saptamani de lucru la proiect
  - Prezentare si notare in ultima saptamana de ore din semestrul 1
- Incheierea situatiei:
  - Examen scris, grila, 20-40 de intrebari din curs => 50% din nota
  - Nota temei de proiect => 50% din nota
  - Situatia se incheie doar daca fiecare din cele doua note de mai sus e cel putin 5
    - Orice nota ≥ 5 este recunoscuta mai tarziu
  - Cei ce nu au situatia de laborator incheiata in iarna vor primi o noua tema pentru prezentare in sesiunea de restante

Bibliografie

- **Curs: Patterns of Enterprise Application Architecture,** (PoEAA), Martin Fowler et al, Addison Wesley, 2002
- **Laborator: Architecting Microsoft .NET Solutions for the Enterprise,** Dino Esposito, Andrea Saltarello, Microsoft Press, 2nd edition
- **Laborator: Developer's Guide to Microsoft Enterprise Library**, Alex Homer, Microsoft Press; **Enterprise Library 6**, http://msdn.microsoft.com/en-us/library/dn169621.aspx
- **Laborator: Programming Entity Framework**, Julia Lerman & Rowan Miller, O'Reilly – sau o orice alta carte decenta de EF
- **Laborator: Practical Entity Framework: Database Access for Enterprise Applications,** Brian Gorman, Apress, 2020
- **Laborator -** recomandat: **Code Complete, Second Edition,** Steve McConnell, Microsoft Press

Continutul cursului si al laboratorului

- Subiecte:
  - Structurarea pe straturi a aplicatiilor de tip enterprise
  - Structurarea logicii aplicatiei
  - Structurarea interfetei utilizator
  - Asocierea intre obiecte si inregistrari in BD
  - Manipularea starii sesiunii in medii fara stare
  - Principiile de distribuire a aplicatiei
- = sabloane de proiectare pentru aplicatii de mari dimensiuni
- Principiile sunt valabile independent de platforma; exemplificarile sunt in Java si C#
  - Versiunile de Java si C# folosite in carte sunt invechite; actualizarile sunt totusi simplu de facut
- La laborator: implementari folosind elemente specifice platformei .NET: Enterprise library, Entity Framework etc.

Despre ce **NU** se va vorbi la curs

- Comunicare bazata pe mesaje
  - "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions"
  - Message Oriented Middleware (MOM), Message Queuing Telemetry Transport (MQTT)
- Tratarea erorilor
- Securitate
  - Carte: "Writing secure code"
- Integrarea aplicatiilor
  - "Service-Oriented Architecture - Concepts, Technology and Design"
  - "SOA - Principles of Service Design"
  - Microservices Architecture
- Refactoring arhitectural
  - Punct de plecare: "Refactoring to Patterns"
- Interfete utilizator: user experience, internationalization, . . .

Tinta cursului

- Arhitecti de sisteme soft
- Programatori care sunt implicati in dezvoltarea de:
  - Aplicatii enterprise
  - Frameworks
  - Biblioteci
- Integratori de sisteme
- Factori de decizie in alegerea uneltelor de dezvoltare utilizate
- Cei care vor sa poata intelege un vocabular clasic (Remote Façade? Pessimistic lock? Active record?)

Arhitectura

- ***Element subiectiv, o intelegere comun acceptata a designului unui sistem soft***
- Intelegerea se refera la precizarea/acordul asupra ***componentelor majore ale sistemului*** si modul in care ***interactioneaza***
- Este legata de deciziile pe care dezvoltatorii experti din cadrul unui proiect trebuie sa le ia
- In literatura: diferiti autori descriu mai degraba viziunea asupra subiectului; exista definitii formale – dar nu si unanim acceptate

Sisteme enterprise (1)

- Nu exista o definitie riguroasa, unanim acceptata
- Sisteme ce permit “afisarea, manipularea si stocarea de cantitati mari de date, adeseori complexe precum si sprijinirea sau automatizarea proceselor de business” (Fowler)
- Exemple:
  - Sisteme de rezervare
  - Sisteme financiare
  - Sisteme de aprovizionare a lanturilor de magazine
  - Management de documente cu versionare
  - Sisteme de analiza a costurilor
  - Asigurari
  - Sistem bursier

Sisteme enterprise (2)

- Caracteristici generale:
  - cantitate mare de date persistente
  - surse multiple de date
  - acces concurent la date
  - multe ecrane (pot fi cateva sute) de interfete utilizator, eventual in tehnologii si platforme diferite (web, dispozitive mobile, calculatoare clasice, touch screen displays)
  - integrare cu alte aplicatii de tip enterprise
  - numeroase transformari de date intre diferite formate - pentru integrare sau rapoarte

    …o parte din aceste aspecte vor fi detaliate

Sisteme enterprise (3)

- Alte cerinte, des pomenite in conjunctie cu cele de mai sus:
  - Trebuie sa poata raspunde unui numar mare de cereri concurente
  - Multiplatforma
  - Trebuie sa suporte mai multe configuratii – e.g. servere de baze de date: Oracle, MS SQL, PostgreSQL, MySQL etc.
  - Disponibilitate (24/7, 99.9999% uptime)
  - Redundanta, failover clustering, ferme web, posibilitate de mutare in/din cloud etc.
  - Suport tehnic, training
  - Emiterea de pachete de corectie – patch-uri

Sisteme enterprise (4)

- Persistenta datelor
  - Necesara deoarece se lucreaza in sesiuni multiple
  - Persistenta de ordinul anilor sau decadelor
  - Depaseste durata de functionare a dispozitivelor hardware
  - Nu trebuie sa fie afectate de uzura morala a sistemelor de operare sau a sistemelor de gestiune a datelor, sau de limbaje/compilatoare/unelte folosite initial
  - Daca e nevoie, ele sunt migrate la o noua tehnologie – avem subspecializari precum design de baze de date sau unelte de Extract-Transform-Load (ETL)
- Cantitatea de date
  - Terabytes sau petabytes de date
  - Exista sisteme care genereaza 1 GByte/secunda
  - Exemplu: pentru AT&T, numarul de legaturi efectuate este atat de mare, incat nu pot fi complet stocate si analizate; France Telecom avea inainte de 2006 baza de date de 30 TB; Walmart opereaza peste 20 de milioane de tranzactii pe zi

Sisteme enterprise (5)

- Accesul concurent la date
  - aplicatii destinate unei institutii: sute de utilizatori simultan
  - pentru o aplicatie accesata prin Internet: cu cateva ordine de marime mai multe accese concurente
  - accesul concurent nu presupune neaparat izolare perfecta, in anumite cazuri concurenta poate sa existe (citiri simultane, dar acces exclusiv la scriere)
  - izolarea perfecta a acceselor concurente, desi posibila, este dusmanul natural al performantei
    - exista diferite grade de izolare, cu anumite grade de compromis pentru efectele operatiilor simultane asupra datelor

Sisteme enterprise (6)

- Interfata utilizator
  - Multe (= sute, …) ecrane de prezentare a datelor
  - Aplicatii care folosesc tipuri diferite de prezentare: XHTML/XHTML+Ajax/Adobe Flash/HTML 5 sau aplicatii de sine statatoare, e.g. WPF/JavaFX/Mobile GUIs
    - Rich Internet Application (RIA): "a web application that has many of the characteristics of desktop applications" (Sursa: https://en.wikipedia.org/wiki/Rich_Internet_Application)
  - Interfata de acces poate fi pusa la dispozitie sub forma de servicii web SOAP sau REST
  - Logica de prezentare a ecranelor poate fi complexa => automat cu tranzitie de stari
  - Prezentarea ecranelor poate sa fie afectata si de drepturile de vizualizare si editare asignate diferitelor roluri; aceste drepturi fac parte din logica aplicatiei

Sisteme enterprise (7)

- Integrarea cu alte sisteme
  - Pre-exista alte sisteme enterprise cu care trebuie sa se faca legatura uni- sau bi-directionala
    - Exemplu: sistem de tip e-payment, baze de date cu evidenta populatiei, sisteme de taxe si impozite
  - Tehnologii diferite, unele legacy: CORBA, sisteme de mesagerie, tehnologii bazate pe XML, emitere/consumare de JSON
  - Platforme de dezvoltare diferite – Java, .NET, PHP, Python, …
  - Fuziunea unor institutii cere integrare de sisteme
  - Impunerea unor norme sau metodologii
  - Problemele cele mai mari: interpretarea corecta a datelor preluate din alte sisteme; importul datelor

Sisteme enterprise (8)

- Logica aplicatiei (business logic)
  - Cea mai putin logica parte dintr-o aplicatie enterprise
  - Trebuie sa implementeze regulile impuse din lumea reala
  - Materializate ca set de constrangeri care interactioneaza
  - Puternic dependenta de existenta si comunicarea cu expertii in domeniu, sau cu beneficiarul final
  - Susceptibila la schimbari sau augmentari => trebuie asigurata flexibilitate
  - Asigurarea flexibilitatii si a posibilitatii de modificare cu efort minim este o provocare majora
  - ***Capacitatea de testare (unit testing, regression testing, integration testing) este esentiala!***

Pattern-urile arhitecturale

- Vin cu solutii posibile pentru diferite probleme
- ***!!Dar nu impun o anumita rezolvare!!***
- Decizia finala se ia de catre arhitect, in functie de experienta sa si de cerintele aplicatiei
- Deseori, alegerile suporta modificari => nu avem o solutie fixa
- Solutiile posibile trebuie cunoscute si cantarite atent
- In functie de alegeri, se iau decizii si asupra uneltelor folosite - in special framework-uri, biblioteci, componente tip server

Performanta (1)

- Punct de interes **major** intr-o aplicatie enterprise
- Cum o cuantifici? Care sunt dimensiunile cele mai importante? Cum faci optimizarea? **Ce optimizezi** prima data?
- Se impun **masuratori** pe o platforma reala, sau estimarea cat mai precisa a performantei; de multe ori apare dependenta de context
- Ori de cate ori se face o modificare (optimizare/patching), trebuie refacute **masuratorile**
- Insa performanta nu este **singurul** punct de interes; uneori ea este sacrificata in interesul cresterii flexibilitatii solutiei oferite
  - "We should forget about small efficiencies, say about 97% of the time: *premature optimization is the root of all evil…*" (Donald Knuth)

Performanta (2) - dimensiuni

- ***Timpul de raspuns***: timpul necesar sistemului pentru a procesa o cerere din exterior
- ***Viteza de reactie (responsiveness)***: cat de repede sistemul semnaleaza utilizatorului faptul ca a perceput intentia lui: apasarea unui buton poate fi insotita de un mesaj care arata ca s-a trecut la procesare, sau butonul este dezactivat/ascuns /are modificare de text, progress bar etc.
  - Daca sistemul asteapta pana la sfarsitul procesarii, atunci timpul de raspuns este egal cu timpul de reactie
  - O interfata reactiva poate fi benefica pentru modul in care este perceputa si primita aplicatia; alteori poate sa elimine cereri succesive care ar duce la inconsecventa

Performanta (3) - dimensiuni

- ***Latenta*** **–** timpul necesar pentru primirea raspunsului, chiar in lipsa unei procesari de amploare de catre aplicatia enterprise
  - Problema care se manifesta in sistemele distribuite
  - Ca programator nu poti sa intervii asupra latentei; ea e o caracteristica a mediului de comunicare
  - Poti insa sa minimizezi apelurile la distanta, sau sa folosesti caching, sau aplicatii peer to peer
- ***Throughput (debit)***: cat de multe date poti obtine de la aplicatie (octeti pe secunda, tranzactii pe secunda)
  - Evident, aceasta marime este dependenta de complexitatea scenariului, gradul de incarcare a aplicatiei, accese concurente etc.
- ***Capacitatea*****:** debitul maxim
- Performanta: una din ultimele doua, dar de preferat sa se specifice exact care din ele

Performanta (4) - dimensiuni

- ***Incarcarea:*** gradul de stres sub care se afla sistemul la un moment dat
  - Numarul de utilizatori aflati in sistem, numarul de tranzactii sau de procese aflate in desfasurare etc.
  - Corelata cu performanta sistemului: 0.5 secunde pentru 10 utilizatori, 2.5 secunde cu 20 de utilizatori
  - Exemplu: sistem A cu 0.5 secunde pentru 10-20 utilizatori, sistem B cu 0.2 secunde pentru 10 utilizatori si 2 sec pentru 20 de utilizatori => performanta lui B se degradeaza la stres mai repede decat A
  - Frecvent se considera niste praguri dincolo de care performantele sistemului se degradeaza inacceptabil

Performanta (5) - dimensiuni

- ***Eficienta***: performanta impartita la resurse
- ***Scalabilitatea:*** masoara modul in care adaugarea de noi resurse imbunatateste performanta sistemului
  - ***Scalare pe verticala:*** se adauga noi resurse hardware (memorie, procesor)
  - ***Scalare pe orizontala:*** adaugare de masini (reale sau virtuale)
  - Capacitatea softului A este de 20 tranzactii pe secunda (tps) in timp ce pentru B avem 40 tps; care e mai scalabil?
  - Se adauga un server; A lucreaza cu 35 tps iar B cu 50 tps => A scaleaza mai bine decat B pe orizontala, pentru 2 servere
  - De multe ori merita sa concepi sistemul soft pentru scalabilitate decat pentru performanta bruta - hardware-ul e ieftin sau se poate folosi cloud
  - Adaugarea de noi servere e uneori mai ieftina decat refactorizarea aplicatiei

Pattern-uri (1)

- Fiecare pattern descrie o problema care apare des in mediul tau de lucru, apoi descrie solutia la acea problema, astfel incat solutia poate fi reutilizata, chiar in contexte diferite
- Pattern: solutie pentru o problema ce apare frecvent
- Aplicarea nu se face orbeste, ci luand in considerare contextul
- Deseori, pentru o problema pot fi folosite alternativ mai multe pattern-uri similare
- Nu sunt idei originale, ci deriva din experienta din campul muncii
- Important: sa cunosti si sa ***intelegi*** cat mai multe

Pattern-uri (2) - structura

- Numele/abrevierea
  - Favorizeaza comunicarea: MVC, Data Transfer Object
- Intentia
  - Sumarizeaza in 1-2 propozitii la ce se foloseste
- Schita
  - Reprezentare (deseori vizuala, prin diagrame UML) a pattern-ului
- Problema de motivare
  - Un exemplu concret in care pattern-ul devine util
- Solutia
  - Se da rezolvarea, deseori insotita de diagrame UML sau schite de cod
- De regula trebui cautate cat mai multe exemple pentru a lamuri cat mai bine; pot exista perceptii diferite (dar corecte)

Pattern-uri (3) - limitari

- Nu vei gasi o lista completa a pattern-urilor; acestea pot aparea in functie de probleme si context
- Reprezinta un punct de plecare, nu destinatie
- Nu trebuie folosite de dragul lor; folosite in exces, fac sistemul inutil de complex
- Deseori sunt substituibile; prima idee nu e neaparat cea mai buna
- Nu sunt cunoscute de toti cei implicati in dezvoltarea de soft – dar *lumea e educabila*

Ce si cat sa optimizezi: legea lui Amdahl

- Data de Gene Amdahl, ***arhitect*** de calculatoare
- Da o valoare pentru optimizarea maxima ce se obtine din imbunatatirea unei parti a sistemului
- Daca o imbunatatire afecteaza un procent *P* dintr-un sistem, iar imbunatatirea are valoarea *S* (partea *P* merge de *S* ori mai repede), atunci imbunatatirea totala este data de:

$$\frac{1}{(1-P)+\frac{P}{S}}.$$

- Exemplu: se imbunatateste 30% dintr-un sistem (deci portiunea afectata de modificare are masura P=0.3, partea care nu se modifica este 1-P=0.7), iar bucata imbunatatita are viteza dublata (S=2)
- Atunci valoarea imbunatatirii totale este de aproximativ 17.6%:

$$\frac{1}{(1-0.3)+\frac{0.3}{2}} = 1.176470588235294$$

Tema

- Care sunt uneltele de profiling pe care le folositi in practica? mailto:lucian.sasu@yahoo.com