

**MINISTERUL EDUCAȚIEI AL REPUBLICII MOLDOVA
UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLȚI
FACULTATEA DE ȘTIINȚE REALE, ECONOMICE ȘI ALE MEDIULUI
CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ**

CURRICULUM

pentru unitatea de curs

„ARHITECTURI PENTRU SISTEME SOFTWARE”

pentru specializarea Administrarea bazelor de date și tehnologii Web

Ciclul II, studii superioare de masterat, învățământ cu frecvență la zi

Titularul disciplinei:

Dr., lector superior Corina Negara

BĂLȚI, 2016

Curriculumul a fost discutat la ședința Catedrei de matematică și informatică

Procesul verbal nr. 7 din 4 februarie 2016

Șeful catedrei dr. conf. univ. E. Plohotniuc _____

Curriculumul a fost aprobat la ședința Consiliului Facultății de Științe Reale, Economice și ale Mediului

Procesul verbal nr. 7 din 23 februarie 2016

Decanul facultății, dr. hab., prof. univ. P. Topală _____

Informații de identificare a disciplinei

Facultatea: Științe Reale, Economice și ale Mediului

Catedra: Matematică și informatică

Domeniul general de studiu: Științe exacte

Denumirea specializării: Administrarea bazelor de date și tehnologii WEB

Administrarea unității de curs:

Codul unității de curs	Credite ECTS	Total ore	Repartizarea orelor				Forma de evaluare	Limba de predare
			Prelegeri	Seminare	Laboratoare	Lucrul individual		
S.02.O.13	10	300	40	–	40	220	examen	Rom/rus

Statutul: Unitate de curs obligatorie

Localizarea sălilor: curs – aula 141, laboratoare – aula 150

Informații referitoare la cadrele didactice

Titularul cursului – *Corina Negara*, dr. în științe pedagogice, lector superior, absolventa Universității de Stat „A. Russo” din Bălți, specialitatea „Matematica și informatica”. A efectuat studiile de master la specializarea „Gestiunea informației”.

E-mail: corina.negara@gmail.com

Orele de consultații - luni: 14.00 -15.30. Consultațiile se oferă atât în regim „față-în-față”, cât și prin utilizarea poștei electronice, Skype. Numele în Skype – [corina.negara](#)

Integrarea cursului în programul de studii (planul de învățământ)

Cursul „Arhitecturi pentru sisteme software” urmează modelul promovat de Software Engineering Institute, Carnegie Mellon University USA, inițiatorii disciplinei arhitectura software. Înțelegerea și proiectarea arhitecturilor software este esențială pentru întreg ciclul de dezvoltare de software, deci este necesară în orice firmă de dezvoltare de software.

În cadrul cursului se introduc și se analizează definițiile și conceptele de vederi, tipuri de vederi și stiluri arhitecturale. Sunt descrise principalele stiluri arhitecturale și proprietățile lor relevante, ulterior, la lecțiile de laborator, studenții utilizează și examinează unele din ele. Este prezentat și exemplificat procesul de proiectare arhitecturală, precum și diverse abordări care asigură securitatea și utilitatea arhitecturilor. Se pune accent pe analiza atributelor de calitate în proiectarea arhitecturilor software. Documentația arhitecturală software și evaluarea sunt prezentate ca activități esențiale în procesul de dezvoltare a arhitecturii pentru sisteme software.

Cursul este destinat studenților de la specializarea „Administrarea bazelor de date și tehnologii WEB” studii superioare de masterat a Facultății de Științe Reale, Economice și ale Mediului. Este o disciplină obligatorie pentru viitorii specialiști din domeniu.

Studierea disciplinei „Arhitecturi pentru sisteme software” se sprijină pe cunoștințele, capacitățile și competențele dezvoltate în cadrul disciplinelor „Java”, „Sisteme distribuite” și „Proiectarea sistemelor software”. Scopurile și conținutul cursului sunt corelate cu scopurile și conținutul cursurilor „Asigurarea calității în sisteme software”.

Competențe prealabile:

1. Programare Java.
2. Modelare UML.
3. Cunoașterea fundamentelor ingineriei software.

Competențe dezvoltate în cadrul cursului

- Competențe cognitive:
 - de cunoaștere și înțelegere a metodologiilor utilizate în proiectarea și evaluarea arhitecturii software;
 - de cunoaștere a instrumentelor specifice proiectării arhitecturilor software;
 - de cunoaștere și înțelegere a stilurilor arhitecturale fundamentale pe baza unor studii de caz;
 - de cunoaștere a limbajului de descriere arhitecturală.
- Competențe de analiză:
 - de analiză a arhitecturilor software;
 - de apreciere a caracteristicilor arhitecturilor software;
- Competențe de aplicare:
 - de proiectare a arhitecturii software adecvate cerințelor înaintate;
 - de aplicare a instrumentelor specifice în proiectarea arhitecturală;
 - de aplicare a limbajului de descriere arhitecturală în procesul de proiectare a arhitecturilor software.
- Competențe de comunicare:
 - de consultare a clienților referitor la arhitecturile de referință și standardele arhitecturale.

Finalitățile cursului

La finalizarea studierii cursului studentul va fi capabil:

- Să dezvolte arhitecturi pentru sisteme software cu o complexitate medie;
- Să reprezinte arhitectura sistemului software din diferite perspective folosind un limbaj de descriere arhitecturală și UML;
- Să documenteze din perspective multiple arhitecturi de sisteme software;
- Să utilizeze limbaje de descriere arhitecturală;
- Să selecteze stilul arhitectural corespunzător cerințelor de calitate a sistemului software proiectat;
- Să ofere consultanță în utilizarea de arhitecturi de referință și standarde arhitecturale;

- Să propună câteva arhitecturi pentru un sistem software;
- Să evalueze câteva arhitecturi pentru un sistem software cu scopul de a determina cel mai adecvat cerințelor sistemului software;
- Să determine punctele slabe ale unei arhitecturi și să propună modalități de îmbunătățire a arhitecturii software;
- Să utilizeze instrumente specifice în procesul de proiectare arhitecturală.

Conținuturi Prelegeri

Nr. d/o	Subiectele predate	Nr. de ore
1.	Definiția și rolul arhitecturii software. Arhitectura software în context. Relațiile de influență ale arhitecturii software (Architecture Business Cycle). Rolul și calitățile arhitectului.	2
2.	Analiza definiției arhitecturii. Definirea tipurilor de vederi, stiluri și vederi. Perspectivile arhitecturii software: statică, dinamică, alocării. Introducere în limbajul Acme și instrumentul AcmeStudio.	4
3.	Elementele care dirijează arhitectura și importanța cerințelor referitoare la atributele de calitate. Scenarii pentru exprimarea cerințelor pentru atribute de.	2
4.	Structuri software și șabloane. Familia de stiluri pentru structuri de tip flux de date: batch sequential, pipe-and-filter, control de proces.	4
5.	Familia de stiluri call-return: stilul client-server, stilul peer-to-peer, stilul SOA (service oriented architecture), stilul în trepte specializare pentru client-server, arhitectură aplicații web, arhitectură aplicații Java EE.	4
6.	Stiluri bazate pe evenimente: stiluri cu evenimente explicite și stiluri cu evenimente implicit, implicit invocation. Stiluri cu date partajate: stilul passive repository, stilul blackboard.	2
7.	Evaluarea curentă	2
8.	Perspectiva statică - module views: decomposition, generalization, layered, aspects, data model. Perspectiva alocării: deployment, install, work assignment.	2
9.	Procesul de proiectare arhitecturală: identificarea contextului, descompunere, consolidarea documenta, evaluarea arhitecturii. Exemple.	4
10.	Concepte referitoare la documentație. Tehnici de realizare a documentației. Utilizarea UML pentru reprezentarea arhitecturii. Utilizarea AADL pentru reprezentare arhitecturi de sisteme de timp real și embedded.	2
11.	Proiectare pentru atribute de calitate (ADD). Exemplu de aplicare ADD.	2
12.	Proiectare arhitecturii pentru Securitate. Șabloane arhitecturale suport pentru utilizabilitate (USAP). Ciclul de dezvoltare a arhitecturii securității.	2
13.	Concepte generale în evaluarea arhitecturilor software.	2
14.	SOA și Cloud Computing.	2
15.	Linii de produse software: domeniul, potențial de reutilizare, variabilitate, documentarea și evaluarea arhitecturii. Studiu de caz. Standarde pentru integrare de componente.	2
16.	Evaluarea curentă	2
Total		40

Laboratoare

Nr. d/o	Tematica	Nr. de ore
1.	Studiul instrumentului AcmeStudio.	4
2.	Înțelegerea stilului arhitectural pipe-and-filter în baza unei aplicații Java.	4
3.	Studiul instrumentului AcmeStudio. Caracteristici avansate. Creare familii, editare vizualizări, editare proprietăți și reguli și verificare reguli.	4
4.	Înțelegerea familiei de stiluri arhitecturale call-return în baza unei aplicații Java.	6
5.	Înțelegerea arhitecturilor cu invocare implicită în baza unei aplicații Java.	6
6.	Reprezentarea arhitecturilor cu limbajul AADL. Modelarea sistemelor.	8
7.	Reprezentarea arhitecturilor cu limbajul AADL. Modelarea reconfigurării dinamice. Modelarea și analiza fluxurilor abstracte.	8
Total		40

Activități de lucru individual

Sarcina nr. 1 pentru lucrul independent

Analiza sistemului – continuare la temă scrisă la lucrarea de laborator 2.

1. Descrieți în detaliu arhitectura fiecăruia din sistemele A și B.
2. Descrieți modul în care fiecare sistem nou implementează fiecare din funcționalitățile nou cerute.
3. Precizați care au fost deciziile arhitecturale majore și care au fost problemele critice de proiectare pentru modificarea arhitecturii sistemului de bază. Discutați și justificați deciziile de proiectare pe care le-ați luat (utilizând conceptele arhitecturale și vocabularul utilizate la curs).
4. Discutați atributele de calitate pe care le promovează și pe care le inhibă arhitecturile acestor sisteme și deciziile de proiectare pe care le-ați luat cu implicații directe în promovarea sau inhibarea acestor calități.
5. Există alte soluții posibile pe care le-ați fi putut adopta? Ce v-a determinat să alegeți soluția voastră în raport cu aceste soluții posibile?
6. Plecând de la proiectul și implementarea voastră pentru sisteme pipe-and-filter:
 - Se modifică rezultatele calculului dacă se inversează oricare două filtre?
 - S-ar modifica rezultatele dacă filtrele ar fi executate pe calculatoare distribuite cu diferite frecvențe de ceas?
 - În ce măsură implementările voastre diferă de cele ale noțiunii idealizate de arhitectură pipe-and-filter?
 - Evaluați și discutați dificultatea realizării fiecărei modificări, pe baza experienței voastre. Pentru fiecare modificare:
 - evaluați gradul de dificultate în raport cu celelalte modificări;
 - analizați în ce măsură gradul de dificultate este corelat cu faptul că stilul pipe-and-filter permite modificarea respectivă;

- analizați în ce măsură gradul de dificultate este corelat cu faptul că infrastructura a fost mai potrivită pentru modificarea respectivă.

Să presupunem că înregistrările studenților sunt memorate într-o bază de date accesibilă fiecărui filtru. Pentru acest nou sistem intrarea va fi doar ID-ul studentului iar filtrele pot utiliza ID-ul studentului pentru a accesa restul informațiilor din baza de date funcție de necesități.

7. Schițați arhitectura acestui sistem. Descrieți-o astfel încât să poată fi înțeleasă în vederea implementării.
8. Discutați avantajele și dezavantajele modificării sistemului pentru a interacționa cu baza de date.
9. Deviază acest sistem de la stilul pipe-and-filter pur? Explicați și justificați de ce credeți că acest sistem deviază sau nu de la paradigma arhitecturală pipe-and-filter.

Termenul limită (deadline) de prezentare a sarcinii nr. 1 – 28 februarie.

Sarcina nr. 2 pentru lucrul independent

Analiza sistemului – continuare la temă scrisă la lucrarea de laborator 4.

1. Evaluați și discutați dificultatea relativă a realizării fiecărei modificări în raport cu celelalte modificări, pe baza experienței voastre. Pentru fiecare modificare:
 - a. evaluați gradul de dificultate în raport cu celelalte modificări;
 - b. analizați în ce măsură gradul de dificultate este corelat cu faptul că stilul call-return avantajează modificarea respectivă;
 - c. analizați în ce măsură gradul de dificultate este corelat cu faptul că infrastructura a fost mai potrivită pentru modificarea respectivă în raport cu celelalte modificări.
2. Referitor la adăugarea jurnalizării, justificați alocarea noii funcționalități la diferite trepte. Specificați dacă există alternative rezonabile la soluția propusă de voi și precizați motivele alegerii pe care ați făcut-o.
3. Analizați relația dintre vederea C&C și vederile modulare asupra arhitecturii sistemului. Există o corespondență clară între elementele arhitecturale din perspectivă dinamică și elementele implementării? În particular, ce anume ar putea influența alegerea locurilor unde se execută treapta logică și treapta datelor?
4. Discutați avantajele și dezavantajele utilizării șablonului în trepte pentru această aplicație. Discutați diferența dintre treptele din arhitectura sistemului și mașinile pe care se execută aceste trepte. Ce ar putea influența arhitectul în alocarea unei trepte la o anumită mașină? În particular, ce nume ar putea influența alegerea locurilor unde se execută treapta logică și treapta datelor?
OBS: Pentru următoarele întrebări nu se cere și implementarea.
5. Fie următorul text, generator al unui set de cerințe noi:
„Unii clienți ar prefera să primească datele returnate în format XML în loc de format textual. De asemenea, se zvonește că în viitor va trebui : ca sistemul să suporte clienți bazați pe web”.
Considerați cel puțin două variante pentru obținerea acestor capabilități. Analizați comparativ cele două soluții.

6. Utilizând sistemul modificat ca bază pentru analiză, explicați cum îi puteți îmbunătăți securitatea. De exemplu, ați putea considera utilizarea unei comunicări de date criptate între trepte pentru a preveni scurgerile de informații și verificarea contului și parolei utilizator pentru a preveni accesele nedorite.

Schițați arhitectura unui astfel de sistem și răspundeți la următoarele întrebări:

- a. Ce impact ar avea asupra arhitecturii modificările necesare securizării sistemului?
- b. Discutați modificările arhitecturale, motivațiile și elementele implicate în securizarea sistemului.

Dacă credeți că sistemul nu poate fi făcut mai sigur, justificați-vă opinia.

Termenul limită (deadline) de prezentare a sarcinii nr. 2 – 30 martie.

Sarcina nr. 3 pentru lucrul independent

Analiza sistemului – continuare la temă scrisă la lucrarea de laborator 5.

1. Discutați orice deviere de la stilul cu invocare implicită pur, identificată fie în sistemul original fie în cel modificat, atât din punct de vedere al proiectului arhitectural cât și relativ la implementarea acestuia.
2. Descrieți modul în care modificările făcute au afectat arhitectura sistemului. Discutați deciziile de proiectare critice, alternativele de proiectare pe care le-ați considerat, și justificările alegerilor pe care le-ați făcut. Realizați această discuție în termeni de concepte arhitecturale; explicați atributele de calitate ce au ghidat realizarea arhitecturii și prioritățile lor relative în luarea deciziilor arhitecturale.
3. Pe baza experienței voastre, ce tipuri de modificări sunt ușor sau dificil de realizat în arhitecturile cu invocare implicită în comparație cu alte stiluri studiate în laboratoarele anterioare. Explicați de ce, utilizând exemple.
4. Presupuneți că un sistem cu invocare implicită funcționează prea lent, astfel încât pentru a crește performanța decideți replicarea uneia dintre componente. Credeți că pot să apară condiții de competiție? Descrieți cauzele posibile de apariție a condițiilor de competiție. Descrieți soluții potențiale pentru prevenirea acestei probleme și comparați avantajele lor relative. Este acest tip de problemă inerent sistemelor cu invocare implicită?
5. Descrieți pe scurt modificările ce ar fi necesare pentru a transforma sistemul nou într-un sistem distribuit, cu invocare implicită. Evaluați dificultatea realizării acestor modificări. Credeți că se poate păstra stilul cu invocare implicită?

Termenul limită (deadline) de prezentare a sarcinii nr. 3 – 30 aprilie.

Sarcina nr. 4 pentru lucrul independent

Investigarea și evaluarea, din punct de vedere arhitectural, a unuia din standardele următoare.

Standarde propuse pentru investigare.

Cadre pentru componente și servicii distribuite:

- CORBA (OMG);
- DCOM (Microsoft);
- VST (Steinberg);
- DirectX (Microsoft);
- Akka (Typesafe).

Cadre pentru aplicații enterprise:

- .NET (Microsoft);
- Java EE (Oracle);
- RosettaNet.

Descoperirea și coordonarea serviciilor:

- Jini (Apache);
- UPnP (UPnP forum).

Conectivitate la baze de date:

- JDBC (Sun);
- ODBC (Microsoft).

Open Source:

- Apache;
- Eclipse.

ESB (soluții integrare business):

- Mule ESB (MuleSoft);
- Sonic ESB (Progress);
- Microsoft Windows Azure Service Bus;
- WebSphere Enterprise Service Bus (IBM).

Broker mesaje:

- RabbitMQ (Mozilla Public Licence);
- Oracle Message Broker (Oracle);
- Microsoft BizTalk Server;
- WebSphere Message Broker (IBM).

Cadre de dezvoltare aplicații:

- Spring;
- JBoss AOP;
- Ruby on Rails;
- CakePHP.

Conținutul referatului:

1. Tema
2. Partea 1. Introducere (20%):
 - a. Scurt istoric al standardului.

- b. Scurtă descriere a standardului:
 - scopul în care a fost creat;
 - cine l-a creat;
 - cine-l utilizează;
 - standarde cu care se află în relație și principalele diferențe față de acestea.
3. Partea 2. Descrierea standardului (35%):
 - a. Descrierea standardului din punct de vedere arhitectural:
 - descrierea arhitecturii pe care se sprijină standardul (underlying);
 - stilul pe care îl reprezintă sau îl definește standardul;

Obs. Este posibil ca arhitectura standardului să fie o combinație de stiluri pure;

 - descrierea mecanismelor de interacțiune a componentelor și conectorilor.
4. Partea 3. Analiza proprietăților, compromisurilor și aplicabilității (35%):
 - a. Analizați standardul incluzând:
 - atributele de calitate promovate de standard;
 - modul în care standardul oferă suport pentru aceste atribute de calitate;
 - atribute de calitate nepromovate sau inhibitate de standard;
 - de ce standardul promovează aceste atribute de calitate;
 - compromisurile făcute de dezvoltatori în crearea standardului;
 - tipuri de analize sau verificări de consistență pentru care standardul oferă suport;
 - caracteristicile pe care credeți că le-ați putea exploata cu încredere, ca arhitecți, în utilizarea lui practică. Exemple.
5. Partea 4. Documentație (10%):
 - a. Evaluați documentația existentă pentru standard.
 - b. Explicați și dați exemple de moduri în care ar putea fi aceasta îmbunătățită.

Indicații de redactare a lucrării:

- a. Asigurați-vă că ați consultat majoritatea surselor de informații. Se va aprecia modul în care organizați și sintetizați informațiile preluate din surse.
- b. Adăugați la sfârșitul lucrării o secțiune de „Referințe” în care includeți sursele studiate. De asemenea, textul lucrării va include citări ale acestor referințe.
- c. Lucrarea nu va avea mai mult de 8 pagini.

Termenul limită (deadline) de prezentare a sarcinii nr. 4 – 22 mai.

Evaluare

Cunoștințele, capacitățile și competențele studenților vor fi evaluate:

- În cadrul lucrărilor de laborator (conform calendarului disciplinei).
- Prin realizarea a 4 sarcini de lucru independent (conform calendarului disciplinei).
- La examenul final (conform orarului întocmit de decanat).

Nota finală la disciplina „Arhitecturi pentru sisteme software” se calculează conform formulei:

$$N_f = 0,5 \times n_c + 0,5 \times n_e,$$

unde N_f – nota finală; n_c – nota curentă, n_e – nota de la examen.

$$N_c = \text{media_teorie} * 0,3 + \text{media_laborator} * 0,3 + \text{media_s_indepedent} * 0,4$$

unde n_c – nota curentă, media_teorie – media notelor de la evaluările curente, media_laborator – media notelor pentru lucrările de laborator, $\text{media_s_indepedent}$ – media notelor pentru sarcinile de lucru independent.

Examenul final se susține în scris.

Principiile de lucru în cadrul disciplinei

1. O parte din sarcinile de învățare vor fi propuse pentru realizare în grupe mici prin cooperare. Deși activitatea de învățare va fi una colectivă, notele pentru realizarea sarcinilor vor fi individuale. Prezentarea sarcinilor realizate va fi însoțită de o evaluare reciprocă a membrilor subgrupului pentru a identifica aportul fiecărui membru în rezultatul final.
2. Calendarul cursului (termenii-limită de prezentare a sarcinilor propuse spre rezolvare, momentele de evaluare etc.) este corelat cu calendarele la alte discipline din semestru. De aceea prezentarea sarcinilor după termenul-limită indicat în calendar nu este salutăată, iar studenții care amână frecvent prezentarea sarcinilor își formează o imagine nefavorabilă.
3. Nu este salutăată întârzierea la ore.
4. Este salutăată poziția activă a studentului, care studiază din propria inițiativă noi conținuturi, propune soluții (aplicații, instrumente Web), formulează întrebări în cadrul prelegerilor și a orelor practice.
5. În cadrul disciplinei o atenție sporită va fi oferită respectării principiilor *etice*. Prezentarea unor soluții a sarcinilor, preluate de la colegi sau din alte surse, preluarea informațiilor din diverse surse, fără a face trimitere la sursă, va fi considerată *plagiat* și va fi sancționată prin note de „1”.

Referințe bibliografice

1. CLEMENTS P., BACHMANN F. et al.. *Documenting Software Architectures, Views and Beyond*, Second Edition. Addison-Wesley Professional, 2010.
2. FAIRBANKS G. *Just Enough Software Architecture, A Risk Driven Approach*. Boulder: Marshall&Brainerd, 2010.
3. BASS L; CLEMENT P., KAZMAN R. *Software Architecture in Practice*. Third Edition. Addison-Wesley Professional, 2012.
4. ALBIN STEPHEN T. *The Art of Software Architecture: Design Methods and Techniques*. New York: John Wiley & Sons, 2003.
5. LATTANZE ANTHONY J. *Architecting Software Intensive Systems: A Practitioners Guide..* Boca Roton: Auerbach, 2008.
6. CLEMENTS P., KAZMAN R., KLEIN M. *Evaluating Software Architectures: Methods and Case Studies*. Adison Wesley Longman, 2002.

7. BUSCHMANN F., MENUIER R. ET AL. *Pattern-Oriented Software Architecture. A System of Patterns*. New York: John Wiley & Sons, 1996.
8. SOFTWARE ENGINEERING INSTITUTE. *Software Architecture*. [online] [accesat 21.01.2016]. Disponibil: <http://www.sei.cmu.edu/architecture/>