MINISTERUL EDUCAŢIEI AL REPUBLICII MOLDOVA
UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLȚI
FACULTATEA DE ŞTIINŢE 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 şedinţa Catedrei de matematică și informatică

Procesul verbal nr. 7 din 4 februarie 2016

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

Curriculumul a fost aprobat la şedinţa Consiliului Facultăţii de Ştiinţe Reale, Economice și ale Mediului

Procesul verbal nr. 7 din 23 februarie 2016

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

Informaţii de identificare a disciplinei

Facultatea: Ştiinţe Reale, Economice şi ale Mediului

Catedra: Matematică şi informatică

Domeniul general de studiu: Ştiinţe exacte

Denumirea specializării: Administrarea bazelor de date şi tehnologii WEB

Administrarea unităţii de curs:

Codul unităţii 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

Informaţii referitoare la cadrele didactice

Titularul cursului – *Corina Negara*, dr. în ştiinţe pedagogice, lector superior, absolventa Universităţii de Stat „A. Russo" din Bălţi, specialitatea „Matematica şi informatica". A efectuat studiile de master la specializarea „Gestiunea informaţiei".

E-mail:corina.negara@gmail.com

Orele de consultaţii - luni: 14.00 -15.30. Consultaţiile se oferă atât în regim „faţă-în-faţă", cât şi prin utilizarea poştei electronice, Skype. Numele în Skype – corina.negara

Integrarea cursului în programul de studii (planul de învăţământ)

Cursul „Arhitecturi pentru sisteme software" urmează modelul promovat de Software Engineering Institute, Carnegie Mellon University USA, iniţiatorii disciplinei arhitectura software. Înţelegerea şi proiectarea arhitecturilor software este esenţială pentru întreg ciclul de dezvoltare de software, deci este necesară în orice firmă de dezvoltare de software.

În cadrul cursului se introduc şi se analizează definiţiile şi conceptele de vederi, tipuri de vederi şi stiluri arhitecturale. Sunt descrise principalele stiluri arhitecturale şi proprietăţile lor relevante, ulterior, la lecţiile de laborator, studenţii utilizează şi examinează unele din ele. Este prezentat şi exemplificat procesul de proiectare arhitecturală, precum şi diverse abordări care asigură securitatea şi utilitatea arhitecturilor. Se pune accent pe analiza atributelor de calitate în proiectarea arhitecturilor software. Documentaţia arhitecturală software şi evaluarea sunt prezentate ca activităţi esenţiale în procesul de dezvoltare a arhitecturii pentru sisteme software.

Cursul este destinat studenţilor de la specializarea „Administrarea bazelor de date şi tehnologii WEB" studii superioare de masterat a Facultăţii de Ştiinţe Reale, Economice şi ale Mediului. Este o disciplină obligatorie pentru viitorii specialişti din domeniu.

Studierea disciplinei „Arhitecturi pentru sisteme software” se sprijină pe cunoştinţele, capacităţile şi competenţele dezvoltate în cadrul disciplinelor „Java”, „Sisteme distribuite” şi „Proiectarea sistemelor software”. Scopurile şi conţinutul cursului sunt corelate cu scopurile şi conţinutul cursurilor „Asigurarea calităţii în sisteme software”.

Competenţe prealabile:

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

Competențe dezvoltate în cadrul cursului

- Competențe cognitive:
  - de cunoaștere și înțelegere a metodologilor utilizate în proiectarea şi 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 proceul de proiectare a arhitecturilor software.
- Competențe de comunicare:
  - de consultare a clienților referitor la arhitecturile de referință și standardele arhitecturale.

Finalităţile 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 cîteva arhitecturi pentru un sistem software;
- Să evalueze cîteva arhitecturi pentru un sistem software cu scopul de a determina cel mai adecvat cerințelor sistemului software;
- Să determine punctele slabe ale unei arhitecturi şi 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 şi rolul arhitecturii software. Arhitectura software în context. Relațiile de influență ale arhitecturii software (Architecture Business Cycle). Rolul şi calitățile arhitectului.	2
2.	Analiza definiției arhitecturii. Definirea tipurilor de vederi, stiluri şi vederi. Perspectivele arhitecturii software: statică, dinamică, alocării. Introducere în limbajul Acme şi instrumentul AcmeStudio.	4
3.	Elementele care dirijează arhitectura şi importanța cerințelor referitoare la atributele de calitate. Scenarii pentru exprimarea cerințelor pentru atribute de.	2
4.	Structuri software şi şabloane. 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 arhitecture), stilul în trepte specializare pentru client-server, arhitectură aplicații web, arhitectură aplicații Java EE.	4
6.	Stiluri bazate pe evenimente: stiluri cu evenimente explicite şi 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 documentaţie. Tehnici de realizare a documentaţiei. Utilizarea UML pentru reprezentarea arhitecturii. Utilizarea AADL pentru reprezentare arhitecturi de sisteme de timp real şi embedded.	2
11.	Proiectare pentru atribute de calitate (ADD). Exemplu de aplicare ADD.	2
12.	Proiectare arhitecturii pentru Securitate. Şabloane arhitecturale suport pentru utilizabilitate (USAP). Ciclul de dezvoltare a arhitecturii securității.	2
13.	Concepte generale în evaluarea arhitecturilor software.	2
14.	SOA şi Cloud Computing.	2
15.	Linii de produse software: domeniul, potențial de reutilizare, variabilitate, documentarea şi 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.	Înţelegerea 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.	Înţelegerea familiei de stiluri arhitecturale call-return în baza unei aplicații Java.	6
5.	Înţelegerea 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**

Activităţi de lucru individual

Sarcina nr. 1 pentru lucrul independent

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

1. Descrieţi în detaliu arhitectura fiecăruia din sistemele A şi B.
2. Descrieţi modul în care fiecare sistem nou implementează fiecare din funcţionalităţile nou cerute.
3. Precizaţi care au fost deciziile arhitecturale majore şi care au fost problemele critice de proiectare pentru modificarea arhitecturii sistemului de bază. Discutaţi şi justificaţi deciziile de proiectare pe care le-aţi luat (utilizând conceptele arhitecturale şi vocabularul utilizate la curs).
4. Discutaţi atributele de calitate pe care le promovează şi pe care le inhibă arhitecturile acestor sisteme şi deciziile de proiectare pe care le-aţi luat cu implicaţii directe în promovarea sau inhibarea acestor calităţi.
5. Există alte soluţii posibile pe care le-aţi fi putut adopta? Ce v-a determinat să alegeţi soluţia voastră în raport cu aceste soluţii posibile?
6. Plecând de la proiectul şi implementarea voastă 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 frecvenţe de ceas?
   - În ce măsură implementările voastre diferă de cele ale noţiunii idealizate de arhitectură pipe-and-filter?
   - Evaluaţi şi discutaţi dificultatea realizării fiecărei modificări, pe baza experienţei voastre. Pentru fiecare modificare:
     - evaluaţi gradul de dificultate în raport cu celelalte modificări;
     - analizaţi în ce măsură gradul de dificultate este corelat cu faptul că stilul pipe-and-filter permite modificarea respectivă;

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

Să presupunem că înregistrările studenţilor 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 informaţiilor din baza de date funcţie de necesităţi.

7. Schiţaţi arhitectura acestui sistem. Descrieţi-o astfel încât să poată fi înţeleasă în vederea implementării.
8. Discutaţi avantajele şi dezavantajele modificării sistemului pentru a interacţiona cu baza de date.
9. Deviază acest sistem de la stilul pipe-and-filter pur? Explicaţi şi justificaţi de ce credeţi 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. Evaluaţi şi discutaţi dificultatea relativă a realizării fiecărei modificări în raport cu celelalte modificări, pe baza experienţei voastre. Pentru fiecare modificare:
   a. evaluaţi gradul de dificultate în raport cu celelalte modificări;
   b. analizaţi în ce măsură gradul de dificultate este corelat cu faptul că stilul call-return avantajează modificarea respectivă;
   c. analizaţi 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, justificaţi alocarea noii funcţionalităţi la diferite trepte. Specificaţi dacă există alternative rezonabile la soluţia propusă de voi şi precizaţi motivele alegerii pe care aţi făcut-o.
3. Analizaţi relaţia dintre vederea C&C şi vederile modulare asupra arhitecturii sistemului. Există o corespondenţă clară între elementele arhitecturale din perspectivă dinamică şi elementele implementării ? În particular, ce anume ar putea influenţa alegerea locurilor unde se execută treapta logică şi treapta datelor ?
4. Discutaţi avantajele şi dezavantajele utilizării şablonului în trepte pentru această aplicaţie. Discutaţi diferenţa dintre treptele din arhitectura sistemului şi maşinile pe care se execută aceste trepte. Ce ar putea influenţa arhitectul în alocarea unei trepte la o anumită maşină ? În particular, ce nume ar putea influenţa alegerea locurilor unde se execută treapta logică şi treapta datelor ?
   OBS: Pentru următoarele întrebări nu se cere şi implementarea.
5. Fie următorul text, generator al unui set de cerinţe noi:
   „Unii clienţi ar prefera să primească datele returnate în format XML în loc de format textual. De asemenea, se zvoneşte că în viitor va trebui : ca sistemul să suporte clienţi bazaţi pe web”.
   Consideraţi cel puţin două variante pentru obţinerea acestor capabilităţi. Analizaţi comparativ cele două soluţii.

6. Utilizând sistemul modificat ca bază pentru analiză, explicaţi cum îi puteţi îmbunătăţi securitatea. De exemplu, aţi putea considera utilizarea unei comunicări de date criptate între trepte pentru a preveni scurgerile de informaţii şi verificarea contului şi parolei utilizator pentru a preveni accesele nedorite.

   Schiţaţi arhitectura unui astfel de sistem şi răspundeţi la următoarele întrebări:

   a. Ce impact ar avea asupra arhitecturii modificările necesare securizării sistemului?
   b. Discutaţi modificările arhitecturale, motivaţiile şi elementele implicate în securizarea sistemului.

   Dacă credeţi că sistemul nu poate fi făcut mai sigur, justificaţi-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. Discutaţi 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 şi relativ la implementarea acestuia.
2. Descrieţi modul în care modificările făcute au afectat arhitectura sistemului. Discutaţi deciziile de proiectare critice, alternativele de proiectare pe care le-aţi considerat, şi justificările alegerilor pe care le-aţi făcut. Realizaţi această discuţie în termeni de concepte arhitecturale; explicitaţi atributele de calitate ce au ghidat realizarea arhitecturii şi priorităţile lor relative în luarea deciziilor arhitecturale.
3. Pe baza experienţei voastre, ce tipuri de modificări sunt uşor sau dificil de realizat în arhitecturile cu invocare implicită în comparaţie cu alte stiluri studiate în laboratoarele anterioare. Explicaţi de ce, utilizând exemple.
4. Presupuneţi că un sistem cu invocare implicită funcţionează prea lent, astfel încât pentru a creşte performanţa decideţi replicarea uneia dintre componente. Credeţi că pot să a apară condiţii de competiţie? Descrieţi cauzele posibile de apariţie a condiţiilor de competiţie. Descrieţi soluţii potenţiale pentru prevenirea acestei probleme şi comparaţi avantajele lor relative. Este acest tip de problemă inerent sistemelor cu invocare implicită?
5. Descrieţi pe scurt modificările ce ar fi necesare pentru a transforma sistemul nou într-un sistem distribuit, cu invocare implicită. Evaluaţi dificultatea realizării acestor modificări. Credeţi 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 şi evaluarea, din punct de vedere arhitectural, a unuia din standardele următoare.

Standarde propuse pentru investigare.

Cadre pentru componente şi servicii distribuite:

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

Cadre pentru aplicaţii enterprise:

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

Descoperirea şi coordonarea serviciilor:

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

Conectivitate la baze de date:

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

Open Source:

- Apache;
- Eclipse.

ESB (solutii 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 Mesage Broker (IBM).

Cadre de dezvoltare aplicatii:

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

Conţinutul 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 relaţie şi principalele diferenţe faţă 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 defineşte standardul;
        Obs. Este posibil ca arhitectura standardului să fie o combinaţie de stiluri pure;
      - descrierea mecanismelor de interacţiune a componentelor şi conectorilor.
4. Partea 3. Analiza proprietăţilor, compromisurilor şi aplicabilităţii (35%):
   a. Analizaţi standardul incluzând:
      - atributele de calitate promovate de standard;
      - modul în care standardul oferă suport pentru aceste atribute de calitate;
      - atribute de calitate nepromovate sau inhibate de standard;
      - de ce standardul promovează aceste atribute de calitate;
      - compromisurile făcute de dezvoltatori în crearea standardulu;
      - tipuri de analize sau verificări de consistenţă pentru care standardul oferă suport;
      - caracteristicile pe care credeţi că le-aţi putea exploata cu încredere, ca arhitecţi, în utilizarea lui practică. Exemple.
5. Partea 4. Documentaţie (10%):
   a. Evaluaţi documentaţia existentă pentru standard.
   b. Explicaţi şi daţi exemple de moduri în care ar putea fi aceasta îmbunătăţită.

Indicaţii de redactare a lucrării:

a. Asiguraţi-vă că aţi consultat majoritatea surselor de informaţii. Se va aprecia modul în care organizaţi şi sintetizaţi informaţiile preluate din surse.
b. Adăugaţi la sfârşitul lucrării o secţiune de „Referinţe” în care includeţi sursele studiate. De asemenea, textul lucrării va include citări ale acestor referinţe.
c. Lucrarea nu va avea mai mult de 8 pagini.

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

Evaluare

Cunoştinţele, capacităţile şi competenţele studenţilor 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 x n_c + 0,5 x n_e,

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

N_c = media_teorie * 0,3 + media_laborator * 0,3 + media_s_indepnedent * 0,4

unde n_c – nota curentă, media_teorie – media notelor de la evaluările curente, media_laborator – media notelor pentru lucrările de laborator, media_s_indepnedent – media notelor pentru sarcinile de lucru independent.

Examenul final se susţine în scris.

Principiile de lucru în cadrul disciplinei

1. O parte din sarcinile de învăţare vor fi propuse pentru realizare în grupe mici prin cooperare. Deşi activitatea de învăţare va fi una colectivă, notele pentru realizarea sarcinilor vor fi individuale. Prezentarea sarcinilor realizate va fi însoţită de o evaluare reciproca 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 salutată, iar studenţii care amână frecvent prezentarea sarcinilor îşi formează o imagine nefavorabilă.
3. Nu este salutată întârzierea la ore.
4. Este salutată poziţia activă a studentului, care studiază din propria iniţiativă noi conţinuturi, propune soluţii (aplicaţii, instrumente Web), formulează întrebări în cadrul prelegerilor şi a orelor practice.
5. În cadrul disciplinei o atenţie sporită va fi oferită respectării principiilor *etice*. Prezentarea unor soluţii a sarcinilor, preluate de la colegi sau din alte surse, preluarea informaţiilor din diverse surse, fără a face trimitere la sursă, va fi considerată *plagiat* şi va fi sancţionată prin note de „1” .

Referinţe 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*. [onine] [accesat 21.01.2016]. Disponibil: http://www.sei.cmu.edu/architecture/