

FIȘA DISCIPLINEI

Proiectarea sistemelor software

Anul universitar 2026 - 2027

1. Date despre program

1.1. Instituția de învățământ superior	Universitatea Babeș-Bolyai din Cluj-Napoca
1.2. Facultatea	Facultatea de Matematică și Informatică
1.3. Departamentul	Departamentul de Informatică
1.4. Domeniul de studii	Informatică
1.5. Ciclu de studii	Master
1.6. Programul de studii / Calificarea	Inginerie Software
1.7. Forma de învățământ	Cu frecvență

2. Date despre disciplină

2.1. Denumirea disciplinei	Proiectarea sistemelor software			Codul disciplinei	MME8065
2.2. Titularul activităților de curs	Conf. Univ. Dr. Molnar Arthur-Jozsef				
2.3. Titularul activităților de seminar	Conf. Univ. Dr. Molnar Arthur-Jozsef				
2.4. Anul de studiu	1	2.5. Semestrul	2	2.6. Tipul de evaluare	Examen
2.7. Regimul disciplinei	Obligativu		2.8. Tipul disciplinei	Disciplină de specializare (DS)	

3. Timpul total estimat (ore pe semestru al activităților didactice)

3.1. Număr de ore pe săptămână	3	din care: 3.2. curs	2	3.3. seminar/ laborator/ proiect	1
3.4. Total ore din planul de învățământ	42	din care: 3.5. curs	28	3.6 seminar/laborator	14
Distribuția fondului de timp pentru studiul individual (SI) și activități de autoinstruire (AI)					ore
Studiul după manual, suport de curs, bibliografie și notițe (AI)					45
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					50
Pregătire seminare/ laboratoare/ proiecte, teme, referate, portofolii și eseuri					40
Tutoriat (consiliere profesională)					10
Examinări					10
Alte activități					3
3.7. Total ore studiu individual (SI) și activități de autoinstruire (AI)				158	
3.8. Total ore pe semestru				200	
3.9. Numărul de credite				8	

4. Preconții (acolo unde este cazul)

4.1. de curriculum	-
4.2. de competențe	Competențe de programare în cel puțin un limbaj de programare care suportă paradigma orientată obiect; cunoștințe și competențe legate de fazele importante ale ciclului de dezvoltare software.

5. Conții (acolo unde este cazul)

5.1. de desfășurare a cursului	Sală de curs cu video-proiector și acces la Internet
5.2. de desfășurare a seminarului/ laboratorului	Sală de seminar cu video-proiector și acces la Internet

6.1. Competențele dobândite în urma absolvirii programului de studii (se preiau din planul de învățământ)¹

Competențe profesionale	
Codul competenței	Competență
CP2	analiza, proiectarea și implementarea de sisteme software
CP4	modelare și rezolvarea de probleme din lumea reală
Competențe transversale	
Codul competenței	Competență
CT1	capacitatea de analiză și sinteză a informației; comportarea onorabilă, etică, respectarea deontologiei profesionale
CT3	abilități de comunicare profesională: descrierea clară, concisă, verbală și în scris, a rezultatelor profesionale

6.2. Rezultatele învățării specifice programului de studii (se preiau din planul de învățământ)²

Rezultatele învățării vizate prin disciplină		
Codul competenței	Cunoștințe și înțelegere (Knowledge and understanding)	Abilități academice specifice (Specific academic skills)
CP3	Absolventul posedă cunoștințe fundamentale de modelare prin care analizează probleme din viața reală, le transpune în cerințe concrete și elaborează un model software corespunzător. Absolventul este capabil să realizeze cercetări în inginerie software, în special în domeniul gândirii algoritmice și gândirii critice.	Absolventul are abilități de a realiza demersului de educare și pregătire pe diverse teme legate de dezvoltarea sistemelor software. Absolventul este capabil să folosească limbajul de specialitate și terminologia specifică domeniului ingineriei software, astfel încât să poată comunica și interacționa cu membrii unor echipe de lucru.
CP5	Absolventul demonstrează abilități avansate de programare care vor permite acumularea de cunoștințe solide și înțelegerea rapidă a tehnologiilor moderne din domeniu. Absolventul este în măsură să aplice cunoștințe avansate de inginerie software, plecând de la studierea la un nivel ridicat de abstractizare a diferitelor sisteme, fiind capabil să ofere soluții de implementare pentru aplicații la sisteme informatice complexe, integrate.	Absolventul cunoaște și respectă norme și reguli etice și deontologice în cercetarea științifică.

7. Rezultatele învățării specifice disciplinei

Cunoștințe și înțelegere (Knowledge and understanding)
Absolventul/a are cunoștințe necesare pentru a concepe, modela și proiecta aplicații software complexe.
Absolventul/a posedă cunoștințe fundamentale de modelare prin care analizează probleme din viața reală, le transpune în cerințe concrete și elaborează un model software corespunzător.
Abilități academice specifice (Specific academic skills)

¹ Se vor prelua din Planul de învățământ al programului de studii acele competențe profesionale și/sau transversale la dezvoltarea cărora contribuie disciplina pentru care se elaborează fișa disciplinei. Pentru fiecare competență se va prelua întregul enunț, inclusiv codul competenței, cu formularea care apare în planul de învățământ, fără modificări. Dacă nu se preia nici o competență din oricare din cele două categorii, se șterge linia din tabel aferentă acelei categorii.

² Se menționează rezultatele învățării specifice programului de studiu la dezvoltarea cărora contribuie disciplina pentru care se elaborează fișa. Enunțurile, preluate fără modificări din Planul de învățământ în funcție de tipul disciplinei (DF/DS/DC) se trec în dreptul competenței asociate.

Absolventul/a are abilitatea de a urmări întregul ciclu de viață al dezvoltării unui sistem software.

Absolventul/a este capabil să folosească limbajul de specialitate și terminologia specifică ingineriei software, astfel încât să poată comunica și interacționa cu membrii unor echipe de lucru.

8. Conținuturi

8.1 Curs	Metode de predare - învățare	Observații ³
Introducere Ciclul de dezvoltare software și procesul software; exemple practice – provocările de scalabilitate pentru o platformă care trece de la 1 la 1 milion de utilizatori.	<ul style="list-style-type: none">• Expunere interactivă• Explicație• Conversație• Exemple• Demonstrație didactică	
Provocări în domeniul dezvoltării software Volatilitatea cerințelor, procesul, tehnologia, practici etice și profesionale, administrarea influențelor de design. Exemple practice de la companii și platforme importante.		
Ciclul de dezvoltare software Cerințe, arhitectura software, proiectarea detaliată, proiectarea de construcție, proiectarea interfeței om-calculator, documentarea și administrarea documentației de proiectare.		
Șabloane și stiluri de arhitectură software Arhitectura stratificată, client-server, peer-to-peer, MVC, broker, blackboard, master-slave, arhitecturi orientate pe servicii, microservicii, blockchain și contracte inteligente.		
Stabilirea arhitecturii sistemului		
Stabilirea stivei tehnologice		
Prezentarea studiilor de caz Arhitecturi ale unor sisteme de scară largă, precum cele utilizate pentru streaming media, social media, administrarea documentelor sau platforme de e-mail.		
Proiectarea de construcție și în detaliu Principiile SOLID, principii pentru proiectarea componentelor, șabloane de proiectare.		
Securitatea software Discuție și prezentări ale unor vulnerabilități hardware și software în sisteme software existente, introducere în analiza de risc pentru securitate și protecția datelor cu caracter personal, modelarea amenințărilor, bazele de date MITRE ATT&CK și Common Vulnerabilities and Exposures.		
Calitatea și mentenanța software Standarde și unelte pentru asigurarea calității software, anti-șabloane, code smells, datoria tehnică și refactorizarea.		
Bibliografie <ol style="list-style-type: none">1. E. Gamma, R. Helm, R. Johnson, J. Vlissides – <i>Design Patterns: Elements of Reusable Object-Oriented Software</i>, Addison Wesley, 1995.2. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra - <i>Head First Design Patterns</i>, O'Reilly Media, 2004.3. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - <i>AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis</i>, Wiley, 1998.4. Hohpe Gregor, Woolf Bobby - <i>Enterprise Integration Patterns</i>, Addison-Wesley, 2003 (some resources at https://www.enterpriseintegrationpatterns.com/).5. Martin Fowler - <i>Refactoring. Improving the Design of Existing Code</i>. Addison-Wesley, 1999.6. Carlos Otero - <i>Software Engineering Design - Theory and Practice</i>, CRC Press, Taylor & Francis Group, 2012.7. Alex Xu – <i>System Design Interview – An Insider's Guide</i> (Volumes I and II), ByteByteGo, 2020.8. Martin Kleppmann - <i>Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems</i>, O'Reilly Media, 2017.9. Robert C. Martin - <i>Clean Architecture: A Craftsman's Guide to Software Structure and Design</i>, Pearson, 2017.		

³ De exemplu aspecte organizatorice, recomandări pentru studenți, aspecte specifice legate de curs/seminar cum ar fi invitarea unor practicieni în domeniu etc.

8.2 Seminar / laborator	Metode de predare - învățare	Observații
Introducere; prezentarea metodei de evaluare și a proiectelor de semestru.	<ul style="list-style-type: none"> • Expunere interactivă • Explicație • Conversație • Exemple • Demonstrație didactică 	
Discuție inițială legată de proiectul de seminar și documentația de arhitectură.		
Lucru pe proiectul de seminar și documentația de arhitectură. Prezentări tehnice susținute de studenți.		
Prezentare legată de procesul de proiectare software în cazul aplicațiilor existente. Prezentări tehnice susținute de studenți.		
Evaluarea primei faze a proiectului de seminar.		
Lucru pe proiectul de seminar și documentația de arhitectură. Prezentări tehnice susținute de studenți.		
Evaluarea documentației de arhitectură.		
Bibliografie <ol style="list-style-type: none"> 1. E. Gamma, R. Helm, R. Johnson, J. Vlissides – <i>Design Patterns: Elements of Reusable Object-Oriented Software</i>, Addison Wesley, 1995. 2. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra - <i>Head First Design Patterns</i>, O'Reilly Media, 2004. 3. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, Thomas J. Mowbray - <i>AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis</i>, Wiley, 1998. 4. Hohpe Gregor, Woolf Bobby - <i>Enterprise Integration Patterns</i>, Addison-Wesley, 2003 (some resources at https://www.enterpriseintegrationpatterns.com/). 5. Martin Fowler - <i>Refactoring. Improving the Design of Existing Code</i>. Addison-Wesley, 1999. 6. Carlos Otero - <i>Software Engineering Design - Theory and Practice</i>, CRC Press, Taylor & Francis Group, 2012. 7. Alex Xu – <i>System Design Interview – An Insider’s Guide</i> (Volumes I and II), ByteByteGo, 2020. 8. Martin Kleppmann - <i>Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems</i>, O’Reilly Media, 2017. 9. Robert C. Martin - <i>Clean Architecture: A Craftsman's Guide to Software Structure and Design</i>, Pearson, 2017. 		






9. Evaluare

Tip activitate	9.1 Criterii de evaluare ⁴	9.2 Metode de evaluare ⁵	9.3 Pondere din nota finală
9.4 Curs	Prezentare în timpul cursului sau seminarului.	Calitatea prezentării și a exemplurilor.	20%
	Crearea documentului de arhitectură pentru o aplicație software de mare complexitate.	Calitatea și nivelul de detaliu al documentației.	40%
	Examen scris în sesiune.	Calitatea răspunsurilor furnizate.	10%
9.5 Seminar/laborator	Analiza arhitecturii și a evoluției unei aplicații complexe cu sursă deschisă.	Calitatea și nivelul de detaliu al documentației.	30%
9.6 Standard minim de promovare			
<ul style="list-style-type: none"> • Studenții vor respecta standardele de integritate academică. • O medie cel puțin egală cu 5.00 trebuie obținută prin susținerea prezentărilor, a proiectelor și a examenului scris. 			

⁴ Criteriile de evaluare trebuie să reflecte direct rezultatele învățării vizate la nivel de program de studii, respectiv la nivel de disciplină. Mai concret, se evaluează achizițiile de învățare menționate în rezultatele anticipate ale învățării.

⁵ Se recomandă stabilirea atât a metodelor de evaluare finală, cât și a strategiei de evaluare pe parcurs.

10. Etichete ODD (Obiective de Dezvoltare Durabilă / Sustainable Development Goals)⁶

	Eticheta generală pentru Dezvoltare durabilă							
								
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
								Nu se aplică nici o etichetă
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Data completării:

...

Semnătura titularului de curs
Conf. Univ. Dr. Molnar Arthur-Jozsef

Semnătura titularului de seminar
Conf. Univ. Dr. Molnar Arthur-Jozsef

Data avizării în departament:

...

Semnătura directorului de departament
Conf.dr. Adrian STERCA

⁶ Selectați o singură etichetă, cea care, în conformitate cu [Procedura de aplicare a etichetelor ODD în procesul academic](#), se potrivește cel mai bine disciplinei. Dacă disciplina tratează tema dezvoltării durabile la modul general (de ex. prin prezentarea/introducerea cadrului general al dezvoltării durabile etc.) atunci se poate alocă eticheta generală de Dezvoltare Durabilă. Dacă niciuna dintre etichete nu descrie disciplina, selectați ultima opțiune: „Nu se aplică nici o etichetă”.