FIȘA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Tehnică din Cluj-Napoca
1.2 Facultatea	Automatică și Calculatoare
1.3 Departamentul	Calculatoare
1.4 Domeniul de studii	Calculatoare și Tehnologia Informaţiei
1.5 Ciclul de studii	Licenţă
1.6 Programul de studii / Calificarea	Calculatoare și Tehnologia Informaţiei / Inginer
1.7 Forma de învăţământ	IF – învăţământ cu frecvenţă

2. Date despre disciplină

2.1 Denumirea disciplinei	***Inginerie software***				Codul disciplinei	**33**
2.2 Titularul de curs	Prof. dr. ing. Eneia Todoran - Eneia.Todoran@cs.utcluj.ro					
2.3 Titularul activităţilor de seminar / laborator / proiect	Conf. dr. inf. Mitrea Paulina - Paulina.Mitrea@cs.utcluj.ro Conf. dr. ing. Mitrea Delia - Delia.Mitrea@cs.utcluj.ro Sl. dr. Anca Iordan - Anca.Iordan@cs.utcluj.ro					
2.4 Anul de studiu	III	2.5 Semestrul	5	2.6 Tipul de evaluare		E
2.7 Regimul disciplinei	Categoria formativă: *DF – fundamentală, DS - de specialitate, DC - complementară*					DF
	Opționalitate: *DOB - obligatorie, DOP - opțională, DFA - facultativă*					DOB

3. Timpul total estimat

3.1 Număr de ore pe săptămână	4	din care:	Curs	2	Seminar	-	Laborator	1	Proiect	1
3.2 Număr de ore pe semestru	56	din care:	Curs	28	Seminar	-	Laborator	14	Proiect	14

3.3 Distribuţia fondului de timp (ore pe semestru) pentru:	
(a) Studiul după manual, suport de curs, bibliografie şi notiţe	20
(b) Documentare suplimentară în bibliotecă, pe platforme electronice de specialitate şi pe teren	17
(c) Pregătire seminarii / laboratoare, teme, referate, portofolii şi eseuri	17
(d) Tutoriat	5
(e) Examinări	10
(f) Alte activităţi:	-

3.4 Total ore studiu individual (suma (3.3(a)...3.3(f)))	69
3.5 Total ore pe semestru (3.2+3.4)	125
3.6 Numărul de credite	5

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	Programare orientata pe obiecte, Tehnici de programare
4.2 de competenţe	Competentele disciplinelor de mai sus

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Blackboard / whiteboard, internet, proiector, calculator
5.2. de desfăşurare a laboratorului / proiectului	Calculator, internet, software specific

6. Competenţele specifice acumulate

6.1 Competenţe profesionale	**C3 -** Soluţionarea problemelor folosind instrumentele ştiinţei şi ingineriei calculatoarelor • **C3.1 -** Identificarea unor clase de probleme şi metode de rezolvare caracteristice sistemelor informatice • **C3.2 -** Utilizarea de cunoştinţe interdisciplinare, a tiparelor de soluţii şi a uneltelor, efectuarea de experimente şi interpretarea rezultatelor lor

	• **C3.3 -** Aplicarea tiparelor de soluţii cu ajutorul uneltelor şi metodelor inginereşti • **C3.4 -** Evaluarea comparativă, inclusiv experimentală, a alternativelor de rezolvare, pentru optimizarea performanţelor. Dezvoltarea şi implementarea de soluţii informatice pentru probleme concrete,
6.2 Competenţe transversale	N/A

7. Rezultatele aşteptate ale învăţării

Cunoştinţe	Descrie, identifică, sumarizează concepte și metode elementare privitoare la ingineria sistemelor software și modul lor de aplicare în probleme concrete.
Aptitudini	Specifică cerinţe, analizează, elaborează, dezvoltă și testează programe în limbaje de programare de uz general (C, etc.) și/sau obiect-orientate (C++, Java, etc.), aplicând elementele specifice ingineriei software.
Responsabilităţi și autonomie:	Își asumă responsabilitatea pentru calitatea și corectitudinea soluţiilor dezvoltate și manifestă autonomie în organizarea activităţilor de proiectare și învăţare, colaborând eficient în echipe și adaptându-se la cerinţe și tehnologii noi.

8. Obiectivele disciplinei

8.1 Obiectivul general al disciplinei	Obiectivul general al disciplinei constă în studiul și aplicarea de abordari sistematice, disciplinate și cuantificabile în dezvoltarea sistemelor software
8.2 Obiectivele specifice	• Studiul și aplicarea proceselor de dezvoltare software • Înţelegerea activităţilor specifice ingineriei software • Cunoașterea metodelor ingineriei software • Cunoașterea unor instrumente specifice ce asistă inginerul software în procesul de specificare, proiectare şi validare • Cunoașterea unor metode de modelare și analiză performanţa software • Aplicarea proceselor, metodelor și instrumentelor studiate în proiecte software de dimensiuni mici și medii

9. Conţinuturi

9.1 Curs	Nr.ore	Metode de predare	Observaţii
Introducere şi privire de ansamblu asupra domeniului	2	Prezentări Power-Point, exemple, întrebări, discuţii.	
Paradigme de dezvoltare software: paradigme de bază (‘cascada’, prototipizare, componente reutlizabile, metode formale) și paradigme evolutive (dezvoltare incrementală, model spirală, inginerie concurentă)	2		
Procese software moderne: procesul unificat, metode flexibile şi programare extrema	2		
Activităţi de bază in ingineria software (specificare, dezvoltare, validare, evoluţie): concepte și principii	2		
Dezvoltare cerinţe: analiza de domeniu, tipuri de cerinţe, tehnici de obţinere a cerinţelor, captare cerinţe sub forma de cazuri de utilizare	2		
Specificare formala: modelare si analiza formala, introducere in probabilistic model checking	2		
Instrumente pentru analiză si specificare formală; PRISM probabilistic model checker, modelare și analiză performanţa software	2		

Modelare cu clase: diagrame UML de clase, semantica diagramelor UML de clase, procesul de construire a diagramelor de clase, implementarea diagramelor de clase în Java	2		
Utilizare Design Patterns (Abstraction-Occurrence, Composite, Observer, Delegation, Adapter, Façade, Proxy, etc)	2		
Modelare interacţiuni şi comportament: diagrame UML de interacţiune (secvenţiere şi comunicare), diagrame UML de stare	2		
Proiectare si arhitectura software: principii (creşterea gradului de coeziune, reducerea gradului de cuplare), stiluri arhitecturale (Multi-Layer, Pipe-and-Filter, etc.)	2		
Testare software: tehnici de testare (partiţionare în clase de echivalenţă, testarea cailor program) şi strategii de integrare (top - down, bottom-up, bazată pe scenarii de utilizare)	2		
Dezvoltare ghidată de cazurile de utilizare: specificare prin cazuri de utilizare, analiză, proiectare şi implementare pentru realizarea cazurilor de utilizare, testarea cazurilor de utilizare	2		
Specificaţii program: inducţie (well founded induction), pre- şi post-condiţii, prototipizare declarativă	2		

Bibliografie *(bibliografia minimală a disciplinei conținând cel puțin o lucrare bibliografică de referință a disciplinei, care există la dispoziția studenților într-un număr de exemplare corespunzător)*
Bibliography:

1. I. Sommerville, *Software Engineering* (6th, 7th, 8th, 9th, 10th editions), Addison Wesley (2001, 2004, 2006, 2010, 2016).
2. T. Lethbridge, R. Laganiere, *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2nd edition), McGraw-Hill, 2005, http://www.lloseng.com.
3. A. Fox, D. Patterson, *Engineering Software as a Service: An Agile Approach Using Cloud Computing* (1st and 2nd editions), Strawberry Canyon (2014, 2021).
4. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
5. E.M. Clarke, T.A. Henzinger, H. Veith, R. Bloem, editors, *Handbook of Model Checking*, Springer, 2018.
6. M. Odersky, L. Spoon, B. Venners, *Programming in Scala*, (3rd , 4th editions), Artima (2016, 2020).
7. B. Pierce et al, *Software Foundations*, vol. 1: *Logical Foundations*, 2026, https://softwarefoundations.cis.upenn.edu/.
8. E.N. Todoran, *Inginerie software: studii in prototipizare si specificare formala*, Mediamira, Cluj-Napoca, 2006.

9.2 Aplicații - seminar / laborator / proiect	Nr.ore	Metode de predare	Observații
OCSF – framework client-server pentru dezvoltare prin reutilizare	2		
Simple Chat – sistem instant messaging bazat pe OCSF (1)	2		
Simple Chat – sistem instant messaging bazat pe OCSF (2)	2		
Utilizare instrumente CASE de modelare software: diagrame UML de clase, cazuri de utilizare, interacţiune, stare, deployment	2		
Utilizare instrumente CASE de modelare si analiza performanta: PRISM model checker; the Rocq proof assistant	2		
Utilizare Design Patterns	2		
Proiectare cazuri de test, utilizare JUnit	2		
Activitatea de dezvoltare a proiectului implica utilizarea paradigmelor si instrumentelor software prezentate la curs. Este incurajata activitatea in echipa de dezvoltatori software. Studentii utilizeaza paradigme si metode de dezvoltare software prezentate la curs, de exemplu, urmand modelul SaaS (Software as a Service). Studentii pot opta pentru utilizarea unor instrumente de proiectare si verificare formala (model checkers, theorem provers). Exista trei iteratii cu termene predefinite. In cazul abordarii traditionale de tip 'cascada' (modelul linear) termenele corespund activitatilor de specificare a cerintelor, design detaliat si livrare finala. Proiectul se preda in saptamana 13.	14		

Bibliografie *(bibliografia minimală pentru aplicații conținând cel puțin o lucrare bibliografică de referință a disciplinei care există la dispoziţia studenţilor într-un număr de exemplare corespunzător)*

1. T. Lethbridge, R. Laganiere, *Object-Oriented Software Engineering: Practical Software Development using UML and Java* (2nd edition), McGraw-Hill, 2005. http://www.lloseng.com.
2. A. Fox, D. Patterson, *Engineering Software as a Service: An Agile Approach Using Cloud Computing* (1st and 2nd editions), Strawberry Canyon (2014, 2021).
3. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
4. B. Pierce et al, *Software Foundations*, vol. 1: *Logical Foundations*, 2026, https://softwarefoundations.cis.upenn.edu/.
5. PRISM manual, 2026. http://www.prismmodelchecker.org/manual/

**Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

10. Coroborarea conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatorilor reprezentativi din domeniul aferent programului

Ingineria software este o disciplina de baza in domeniul Calculatoare si Tehnologia Informatiei. In cadrul cursului, a lucrarilor practice si a orelor de proiect studentii fac cunostinta cu procese, metode si instrumente specifice, si invata sa aplice abordari sistematice si cuantificabile in dezvoltarea sistemelor software. Continutul disciplinei a fost elaborat in baza interactiunii cu specialisti in domeniul Ingineriei Software din Romania, Europa (UK, Greece), SUA si Canada si a fost evaluat de agentii guvernamentale romanesti (CNEAA si ARACIS).

11. Evaluare

Tip activitate	Criterii de evaluare	Metode de evaluare	Pondere din nota finală
Curs	Abilitati de rezolvare probleme	Examen final	75%
Seminar	-	-	-
Laborator	Abilitati de proiectare si validare in cadrul unui proiect software	Test laborator, evaluare proiect	5%
Proiect			20%

Standard minim de performanţă:
Realizarea unui proiect software de dimensiuni medii utilizand cunostintele predate la cursul de Inginerie Software. Calcul nota disciplina: 5% laborator + 20% proiect + 75% examen final
Conditii de participare la examenul final: Laborator ≥ 5, Proiect ≥ 5
Conditii de promovare: Nota ≥ 5

Data completării:	**Titulari**	**Titlu, prenume / NUME**	**Semnătura**
28.04.2026	Curs	Prof.dr.ing. Eneia TODORAN	
	Aplicații	Conf.dr.info. Paulina MITREA	
		Conf.dr.ing. Delia MITREA	
		Sl. dr. Anca IORDAN	

Data avizării în Consiliul Departamentului Calculatoare	Director Departament, Prof.dr.ing. Rodica Potolea
Data aprobării în Consiliul Facultății de Automatică și Calculatoare	Decan, Prof.dr.ing. Vlad Mureșan