FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Politehnica Timișoara
1.2 Facultatea[1] / Departamentul[2]	Inginerie din Hunedoara/Inginerie Electrică și Informatică Industrială
1.3 Domeniul de studii (denumire/cod [3])	Inginerie electrică / 90
1.4 Ciclul de studii	Licență
1.5 Programul de studii (denumire/cod/calificarea)	Inginerie electrică și calculatoare / 60 / Inginer

2. Date despre disciplină

2.1a Denumirea disciplinei/Categoria formativă[4]			Inginerie software / DS				
2.1b Denumirea disciplinei în limba engleză			Software engineering				
2.2 Titularul activităţilor de curs			Șef lucr. Dr. Ing. Ghiormez Loredana				
2.3 Titularul activităţilor aplicative[5]			Șef lucr. Dr. Ing. Ghiormez Loredana				
2.4 Anul de studii[6]	4	**2.5** Semestrul	2	**2.6** Tipul de evaluare	E	**2.7** Regimul disciplinei[7]	DO

3. Timp total estimat - ore pe semestru: activități didactice directe (asistate integral sau asistate parțial) și activități de pregătire individuală (neasistate) [8]

3.1 Număr de ore asistate integral/săptămână	4 , format din:	**3.2** ore curs	2	**3.3** ore seminar/laborator/proiect	2
3.1* Număr total de ore asistate integral/sem.	56 , format din:	**3.2*** ore curs	28	**3.3*** ore seminar/laborator/proiect	28
3.4 Număr de ore asistate parțial/săptămână	, format din:	**3.5** ore practică		**3.6** ore elaborare proiect de diplomă	
3.4* Număr total de ore asistate parțial/ semestru	, format din:	**3.5*** ore practică		**3.6*** ore elaborare proiect de diplomă	
3.7 Număr de ore activități neasistate/ săptămână	4.93 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			0.93
		ore studiu individual după manual, suport de curs, bibliografie și notițe			2
		ore pregătire seminarii/laboratoare, elaborare teme de casă și referate, portofolii și eseuri			2
3.7* Număr total de ore activități neasistate/ semestru	69 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			13
		ore studiu individual după manual, suport de curs, bibliografie și notițe			28
		ore pregătire seminarii/laboratoare, elaborare teme de casă și referate, portofolii și eseuri			28
3.8 Total ore/săptămână [9]	8.93				
3.8* Total ore/semestru	125				
3.9 Număr de credite	5				

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	• Parcurgerea și/sau promovarea următoarelor discipline: Programarea calculatoarelor și limbaje de programare 1, Programarea calculatoarelor și limbaje de programare 2, Programare orientată pe obiecte / Programare Java, Baze de date.
4.2 de rezultatele învățării	• Cunoștințe despre teoria algoritmilor, reprezentarea acestora, programare orientată pe obiecte, lucrul cu baze de date relaționale;

5. Condiţii (acolo unde este cazul)

5.1 de desfăşurare a cursului	• Cursul se va desfășura într-o sală dotată cu videoproiector, tablă pentru scris și computer cu conexiune la internet.
5.2 de desfăşurare a activităţilor practice	• Sală de laborator echipată cu cu videoproiector, tablă pentru scris și computere cu conexiune la internet.

6. Rezultatele învățării la formarea cărora contribuie disciplina

Cunoștințe	• C3. Studentul/absolventul explică și interpretează desenele care detaliază proiectarea produselor, a instrumentelor și a sistemelor de inginerie electrică.
Abilități	• A3. Studentul/absolventul specifică proprietăți tehnice ale bunurilor, materialelor, metodelor, proceselor, serviciilor, sistemelor, software-ului și funcționalităților, prin identificarea și răspunsul la nevoile particulare care urmează să fie satisfăcute în funcție de cerințele clienților. • • A7. Utilizează eficient instrumente digitale de colaborare (MS Teams, Google Workspace, Moodle etc.) pentru învățare și lucru în echipă.
Responsabilitate și autonomie	• RA2. Studentul/absolventul efectuează căutări bibliografice în literatura de specialitate, consultă și folosește bazele de date științifice și alte surse de informare din domeniul ingineriei electrice. • • RA7. Își evaluează constant propriile nevoi de dezvoltare profesională și adoptă o atitudine proactivă față de învățarea pe tot parcursul vieții.

7. Obiectivele disciplinei (asociate rezultatelor învățării de la punctul 6)

- Scopul acestei discipline îl reprezintă însuşirea principalelor concepte, metode şi tehnici ale ingineriei software, cu accent pe reutilizarea codului sursă atât din punct de vedere al programării orientate pe obiect, cât şi din punct de vedere al programării generice în limbajul de programare Java.

  Obiectivele specifice ale acestei discipline sunt:
  - Cunoaşterea etapelor ciclului de viaţă al unui sistem soft;
  - Înţelegerea conceptelor legate de modelarea softului;
  - Familiarizarea cu limbajul de modelare UML;
- Abilitatea de a utiliza instrumente CASE.

8. Conţinuturi[10]

8.1 Curs	Număr de ore	Metode de predare[11]
1. Fazele dezvoltării unui sistem soft 1.1. Ciclul de viaţă al softului. Ciclul de viaţă V 1.2. Faza de analiză 1.3. Faza de proiectare 1.4. Faza de implementare 1.5. Faza de testare	2	Studenții au acces la resurse în format electronic, accesând campusul virtual al UPT sau folosind rețeaua locală de intranet a facultății. Se vor utiliza atât prezentări interactive cât și tradiționale. Se vor folosi: problematizarea, studiu de caz, conversația.
2. Limbajul de modelare unificat UML 2.1. Caracteristicile şi avantajele UML 2.2. Tipuri de diagrame 2.3. Instrumente CASE: IBM Rational Rhapsody, Microsoft Visio, ArgoUML, StarUML 2.4. Diagrame specifice fazei de analiză 2.5. Diagrame specifice fazei de proiectare 2.6. Diagrame specifice fazei de implementare	4	
3. Analiza sistemelor soft 3.1. Colectarea şi analiza cerinţelor 3.2. Instrumente utilizate în specificarea cerinţelor. IBM Rational DOORS	4	

3.3. Metode de analiză orientate pe obiecte		
4. Proiectarea sistemelor soft în Java 4.1. Caracteristicile unui proiectării orientate pe obiecte 4.2. Metode de proiectare 4.3. Principii de proiectare 4.3.1. Principiul deschis-închis 4.3.2. Principiul substituţiei 4.3.3. Principiul inversiunii dependenţelor 4.3.4. Principiul responsabilităţii unice 4.3.5. Principiul segregării interfeţelor 4.4. Şabloane arhitecturale 4.5. Şabloane de proiectare 4.5.1. Şabloane creaţionale 4.5.2. Şabloane structurale 4.5.3. Şabloane comportamentale	10	
5. Verificarea şi validarea sistemelor soft 5.1. Testarea programelor 5.1.1. Criterii de alegere a datelor de test 5.1.2. Testarea white-box 5.1.3. Testarea black-box 5.2. Instrumente de testare: JUnit 5.3. Verificarea modelelor, Verificarea specificaţiei şi a corectitudinii algoritmilor 5.4. Testarea modulelor şi a sistemului final 5.5. Verificarea documentaţiei 5.6. Validarea programelor 5.7. Certificarea produselor soft	6	
6. Managementul unui proiect software 6.1. Managementul software 6.2. Managementul configuraţiei 6.3. Managementul echipei 6.4. Instrumente integrate de management al proiectelor soft: JIRA, MKS Integrity 6.5. Estimarea costului unui sistem soft	2	

Bibliografie[12] 1. Ghiormez L., Curs în format electronic - Inginerie software, Campusul virtual al UPT, https://cv.upt.ro/course/view.php?id=2716 sau intranet facultate \\fs\cursuri\INGINERIE SOFTWARE 2025
2.Dorin Bocu, Răzvan Bocu, Provocări și metode de abordare în managementul proiectelor IT, Editura Albastră, Cluj-Napoca,2013
3.Dorin Bocu, Răzvan Bocu, Modelare obiect orientată cu UML, Editura Albastră, Cluj-Napoca, 2006
4.Mircea Cezar Preda, Ana-Maria Mirea, Doina Lavinia Preda, Constantin Teodorescu-Mihai, Introducere în programareaorientată-obiect. Concepte fundamentale din perspectiva ingineriei software, Editura Polirom, Iași, 2010
5.Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Șabloane de proiectare. Elemente de software reutilizabilorientat pe obiect, Editura Teora, 2002
6 Militon Frențiu, Verificarea și validarea sistemelor soft, Presa Universitară Clujană, Cluj-Napoca, 2010
7.Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013
8.Pankaj Jalote, A Concise Introduction to Software Engineering, Springer-Verlag Londra, 2008
9.Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013
10.Tanasa S., Olaru C., Java de la 0 la expert, Editura Polirom, Colectia Calculatoare, 2011.
11. *Head First Design Patterns*, O'Reilly Media, Inc http://www.msquaredweb.com/DesignPatterns/ HeadFirstDesignPatternsInCSharp.zip
12. Data & Object Factory. *Design Patterns* http://www.dofactory.com/Patterns/Patterns.aspx

8.2 Activităţi aplicative[13]	Număr de ore	Metode de predare
1. Norme de tehnica securităţii muncii. Elaborarea diagramelor specifice fazei de analiză: diagrama cazurilor de utilizare utilizând StarUML	2	Se va utiliza exercițiul la tablă și implementa-rea programului utilizând computerul.
2. Elaborarea diagramelor specifice fazei de analiză: diagrame de	2	

activităţi utilizând StarUML		
3. Elaborarea diagramelor specifice fazei de proiectare: diagrama de clase, diagrama de pachete utilizând StarUML	2	
4. Elaborarea diagramelor specifice fazei de proiectare: diagrame de stare utilizând StarUML	2	
5. Elaborarea diagramelor specifice fazei de proiectare: diagrame de secvenţă, diagrame de colaborare utilizând StarUML	2	Se va utiliza exercițiul la tablă și implementarea programului utilizând computerul.
6. Elaborarea diagramelor specifice fazei de implementare: diagrama de componente utilizând StarUML	2	Se va utiliza exercițiul la tablă și implementarea programului utilizând computerul.
7. Analiza, proiectarea şi implementarea de aplicaţii în limbajul Java ce utilizează şablonul arhitectural MVP. Testarea aplicațiilor folosind mediul NetBeans IDE.	6	Se va utiliza exercițiul la tablă și implementarea programului utilizând computerul.
8. Analiza, proiectarea şi implementarea de aplicaţii în limbajul Java ce utilizează şablonul arhitectural MVC. Testarea aplicațiilor folosind mediul NetBeans IDE.	8	Se va utiliza exercițiul la tablă și implementarea programului utilizând computerul.
9. Recuperări de laborator.	2	Se va utiliza exercițiul la tablă și implementarea programului utilizând computerul.

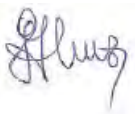
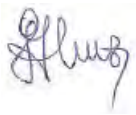
Bibliografie[14]

1. Ghiormez L., Laborator în format electronic - Inginerie software, Campusul virtual al UPT, https://cv.upt.ro/course/view.php?id=2716 sau intranet facultate \\fs\cursuri\INGINERIE SOFTWARE 2025
2. Anca Daniela Ioniță, Modelarea UML în ingineria sistemelor de programare, Editura All, București, 2003
3. Dorin Bocu, Iniţiere în ingineria sistemelor soft, Editura Albastră, Cluj-Napoca, 2002
4. Militon Frențiu, Verificarea și validarea sistemelor soft, Presa Universitară Clujană, Cluj-Napoca, 2010
5. Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013
6. John Hunt, Guide to the Unified Process Featuring UML, Java and Design Patterns, Springer London Ltd, 2014
7. Hamill Paul, Unit Test Frameworks, O'Reilly Media, 2016
8. Tanasa S., Olaru C., Java de la 0 la expert, Editura Polirom, Colectia Calculatoare, 2011.

9. Evaluare

Tip activitate	**9.1** Criterii de evaluare[15]	**9.2** Metode de evaluare	**9.3** Pondere din nota finală
9.4 Curs	Cunoștințe teoretice	Scris - subiecte teoretice și aplicații sau proiect individual / în echipă de două persoane. Tema proiectului se distribuie aleatoriu. Rezolvările în cazului subiectului scris se încarcă pe campusul virtual UPT. Examenul se desfășoară prin intermediul Campusului Virtual UPT sau se primesc subiecte tipărite. În caz de aplicație practică, acesta va fi prezentată oral și va exista legătura între diagramele UML, respectiv aplicația scrisă în limbaj de programare. În caz de proiect în echipă, un student va prezenta diagramele UML, iar celălalt student va prezenta aplicația. În caz de scenariu online se utilizează și aplicația de videoconferință (Zoom)	0.66
9.5 Activităţi aplicative	**S:**		
	L: Abilități în analiza, proiectarea și implementarea aplicațiilor de laborator	Oral – aplicații utilizând computerul In caz de scenariu online testele de control se desfășoară prin intermediul campusului virtual UPT	0.34
	P[16]**:**		
	Pr:		
9.6 Standard minim de performanţă (se prezintă cunoștințele minim necesare pentru promovarea disciplinei și modul în care se verifică stăpânirea lor[17])			
• Activități aplicative: implementarea a cel puțin 4 diagrame UML dezvoltate pe aceeași tematică, dintre care una obligatoriu să fie diagrama de clase.			

- Curs:
- În caz de subiect primit în timpul examenului: implementarea diagramelor UML (cazuri de utilizare și de clase), respectiv implementarea conceptului de model în unul dintre limbajele de programare C#, C++ sau Java sau obținerea a minim jumătate din punctajul total alocat subiectelor. Toate rezolvările se vor realiza pe baza temei subiectului primit.
- În caz de aplicație realizată în prealabil: implementarea tuturor diagramelor UML , respectiv implementarea conceptului de model în unul dintre limbajele de programare C#, C++ sau Java.
- Pentru a promova disciplina trebuie ca ambele activități (curs și laborator) să fie promovate

Data completării	**Titular de curs (semnătura)**	**Titular activităţi aplicative (semnătura)**
10.09.2025		

Director de departament (semnătura)	**Data avizării în Consiliul Facultăţii**[18]	**Decan (semnătura)**
	17.09.2025