MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
UNIVERSITATEA "TIBISCUS" DIN TIMIȘOARA
FACULTATEA DE CALCULATOARE ȘI INFORMATICĂ APLICATĂ
STR. LASCĂR CATARGIU NR. 6, TIMIȘOARA 300559
TEL/ 0256 220 687
E-mal: fcia@tibiscus.ro www.fcia.tibiscus.ro

FIŞA DISCIPLINEI

1. Date despre program

1.1. Instituţia de învăţământ superior	Universitatea „Tibiscus" din Timişoara
1.2. Facultatea	Calculatoare şi Informatică Aplicată
1.3. Departamentul	Informatică
1.4. Domeniul de studii	Informatică
1.5. Ciclul de studii	Licenţă
1.6. Programul de studii/Calificarea	Informatică

2. Date despre disciplină

2.1. Denumirea/codul disciplinei	**Inginerie Software (IS) – LIN314**						
2.2. Titularul activităţii de curs	Lect.univ.dr. Kristijan Cincar						
2.3. Titularul activităţii de seminar	Lect.univ.dr. Kristijan Cincar						
2.4. Anul de studiu	2	2.5. Semestrul	1	2.6. Tipul de evaluare	E	2.7. Regimul disciplinei	Conținut[1]=DF Obligativitate[2]=DI

3. Timpul total estimat

3.1. Numărul de ore pe săptămână	3	din care **3.2.** curs	1	**3.3.** laborator+proiect	1+1
3.4. Total ore din planul de învăţământ	42	din care **3.5.** curs	14	**3.6.** laborator+proiect	14+14

Distribuţia fondului de timp	**ore**
Studiul după manual, suport de curs, bibliografie şi notiţe	28
Documentare suplimentară în bibliotecă, pe platforme electronice de specialitate	14
Pregătire seminarii/laboratoare, teme, referate, portofolii şi eseuri	28
Tutoriat	8
Examinări	5
Alte activităţi	-
3.7. Total ore studiu individual	83
3.8. Total ore pe semestru	125
3.9. Numărul de credite	5

4. Precondiţii (acolo unde este cazul)

4.1. de curriculum	Algoritmi și structuri de date, Programare orientată pe obiecte, Baze de date
4.2. de competenţe	Abilități de programare, analiză logică, gândire algoritmică, utilizarea mediilor de dezvoltare software

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Sală de curs dotată cu videoproiector, tablă, calculator/laptop, Google Classroom.
5.2. de desfăşurare a seminarului/laboratorului	Sală de laborator cu calculatoare, rețea internet, StarUML, Eclipse, Visual Studio, GitHub.

6. Obiectivele disciplinei - rezultate așteptate ale învățării la formarea cărora contribuie parcurgerea și promovarea disciplinei

Cunoștințe	-Înțelegerea procesului de dezvoltare software și a ciclului de viață al produselor software. - Familiarizarea cu modelele de proces: waterfall, spiral, RUP, Agile, DevOps. - Aplicarea UML pentru modelarea sistemelor software. - Cunoașterea principiilor arhitecturii software și sabloanelor de proiectare. - Integrarea testării automate și a metricilor de calitate.
Abilități	- Modelarea și documentarea sistemelor software. - Aplicarea principiilor SOLID și design pattern-urilor. - Implementarea, testarea și documentarea proiectelor software. - Utilizarea metodologiilor Agile/Scrum în dezvoltare.
Responsabilitate și autonomie	- Lucru eficient în echipe software. - Respectarea standardelor profesionale și etice. - Autoevaluare și învățare continuă.

7. Obiectivele disciplinei (reieşind din grila competenţelor specifice acumulate)

7.1. Obiectivul general al disciplinei	Cunoasterea conceptelor şi problematicii fundamentale ale ingineriei software. Capacitatea de a se familiariza cu noi concepte si de a se adapta rapid la noile tehnologii ce apar in domeniul informaticii
7.2. Obiectivele specifice	Dezvoltarea abilităţilor de abordare a dezvoltării sistemelor software.

8. Conţinuturi

8.1. Curs	**Metode de predare**	**Observații**
1.Introducere în Ingineria Software – concepte, definiții, standarde IEEE, SEBoK	Expunere interactivă, demonstrare, exemplificare, discuții.	Se va stimula participarea activă a studenților. Materialele de studiu vor fi accesibile pe platforma Google Classroom
2. Modele de proces software – waterfall, spiral, V-model, RUP, XP, Agile		
3. Ciclul de viață software – etape, artefacte, managementul cerințelor		
4. UML – cazuri de utilizare, clase, secvență, activitate, stare		
5.OCL și specificațiile formale	Expunere interactivă, demonstrare, exemplificare, discuții.	
6.Proiectare și arhitectură software – stiluri și tipuri		
7. Design patterns – creationale, structurale, comportamentale		Se va stimula participarea activă a studenților. Materialele de studiu vor fi accesibile pe platforma Google Classroom
8.Principii SOLID și codificare standardizată		
9.Testare software – tipuri, TDD, metrici	Expunere interactivă, demonstrare, exemplificare, discuții.	
10.Metodologii Agile și management de proiect		
11. DevOps și CI/CD		
12+13+14.Studii de caz și recapitulare		

Bibliografie:

1. Ian Sommerville, Software Engineering, Addison-Wesley, 10th Edition, 2015.
2. Craig Larman, Applying UML and Patterns, Prentice Hall, 2005.
3. B. Bruegge, A. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java, Pearson, 2010.
4. E. Gamma et al., Design Patterns, Addison-Wesley, 1995.
5. Len Bass et al., Software Architecture in Practice, 4th Edition, 2021.
6. SWEBOK, IEEE Computer Society, 2014.
7. M. Coşulschi, Note de curs – Inginerie Software, UVT, 2022

<table>
<tr><th>8.2. Seminar/laborator</th><th>Metode de predare/învățare</th><th>Observaţii</th></tr>
<tr><td>1.Instalarea și configurarea mediilor de lucru (Eclipse, StarUML)</td><td rowspan="10">Lucrări practice pe calculator, colaborare în echipă, lucru individual.</td><td rowspan="10">Se va stimula participarea activă a studenților.</td></tr>
<tr><td>2.Crearea și documentarea unui proiect software</td></tr>
<tr><td>3+4.Modelarea cerințelor prin diagrame UML</td></tr>
<tr><td>5+6.Proiectare arhitecturală – structură logică</td></tr>
<tr><td>7.Implementarea claselor și interfețelor</td></tr>
<tr><td>8.Aplicarea sabloanelor de proiectare</td></tr>
<tr><td>9.Testarea unităților de cod și analiza metricilor</td></tr>
<tr><td>10+11.Documentarea software și utilizarea GitHub</td></tr>
<tr><td>12.Metodologia Scrum și organizarea sprinturilor</td></tr>
<tr><td>13+14. Prezentarea proiectelor finale</td><td></td><td>Verificarea temelor/proiectelor</td></tr>
<tr><td colspan="3">Bibliografie:
• Aceeași ca la curs</td></tr>
<tr><td colspan="3">8.3. Proiect - Realizarea unui proiect complex
🧩 Etapele realizării unui proiect la Inginerie Software
1. Definirea problemei și a obiectivelor
• Identificarea unei nevoi reale (de exemplu: „gestionarea programărilor într-un cabinet medical”).
• Formularea scopului și obiectivelor aplicației.
2. Analiza cerințelor
• Cerințe funcționale: ce trebuie să facă sistemul (ex: adăugare, modificare, ștergere date).
• Cerințe nefuncționale: performanță, securitate, ușurință în utilizare etc.
• Actorii sistemului: cine îl folosește (admin, utilizator, etc.)
3. Modelarea sistemului (analiză UML)
• Diagrama cazurilor de utilizare (Use Case Diagram)
• Diagrama de clase
• Diagrama de activitate / secvență
• Diagrama componentelor
4. Proiectarea aplicației
• Alegerea tehnologiei (ex: Java + MySQL, Python + Flask, PHP + Laravel, etc.)
• Structura bazelor de date (ERD – Entity Relationship Diagram)
• Arhitectura sistemului (MVC, client-server etc.)
5. Implementarea
• Scrierea codului sursă
• Crearea interfeței utilizator
• Conectarea la baza de date
6. Testarea
• Teste unitare și de integrare
• Teste funcționale (verificarea cerințelor)
• Teste de performanță (dacă e cazul)
7. Documentarea proiectului
• Manual de utilizare
• Manual tehnic
• Concluzii și direcții de dezvoltare
8. Prezentarea / demonstrarea proiectului
• Capturi de ecran, video de prezentare, demo live.</td></tr>
</table>

Bibliografie Aceeaşi ca la curs

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

Cunoasterea problematicii si a procesului de dezvoltare de software este necesara oricarui candidat la angajare intr-o firma de dezvoltare de software. De asemenea, firmele client pentru produse de software pot beneficia de pregatirea unui inginer software.

10. Evaluare

Tip de activitate	Criterii de evaluare	Metode de evaluare	Pondere din nota finală
10.1. Curs	Evaluarea are în vedere următoarele categorii de cunoştinţe: Cunoştinţe generale, evaluate printr-un test cuprinzând întrebări cu variante multiple de răspuns. Cunoştinţe de detaliu, evaluate printr-un test cuprinzând întrebări cu variante multiple de răspuns orientate spre noţiunile cheie predate Cunoașterea conceptelor fundamentale și capacitatea de aplicare.	Examen in sesiunea de examene	40 %
10.2. Proiect/ laborator	Realizarea sarcinilor prevăzute în cadrul activităţii de laborator. Participare activă și corectitudinea temelor de laborator.		20 %
	Aplicarea integrată a cunoștințelor în realizarea unui produs software complet.	Prezentare proiect final	40%
10.3. Standard minim de performanţă			
Examinare scrisă: **Pentru nota 5** este necesar un număr de 5 răspunsuri corecte din 10 posibile la un test de tip grilă. Aceste întrebări acoperă noțiunile de bază ce țin de ciclul de viață software și utilizarea UMLde bază. **Pentru nota 10** este necesar un număr de 10 răspunsuri corecte din 10 posibile la un test de tip grilă unde sunt verificate noțiunile teoretice predate la curs. **Probe practice şi activitate de laborator:** **Pentru nota 5** Studentul predă toate temele de laborator **Pentru nota 10** Realizarea cu succes a unui proiect complex. Utilizarea unui instrument software pentru modelare UML. Capacitatea de a intelege si urma un proces de dezvoltare de software.			

Notă:
1) Regimul disciplinei (conţinut) - pentru nivelul de licenţă se alege una din variantele: DF (disciplină fundamentală) / DS (disciplină de specialitate) / DC (disciplină complementară).
2) Regimul disciplinei (obligativitate) - se alege una din variantele: DI=discipline obligatorii (impuse) / DO=discipline opționale/ DFac (disciplină facultativă).

Data completării
20.09.2025

Titular de disciplină
Lect.univ.dr. Kristijan CINCAR

Data avizării în departament
26.09.2025

Director de departament
Conf univ.dr. Victoria IORDAN