FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea Babes-Bolyai**
1.2 Facultatea	**Facultatea de Matematica şi Informatica**
1.3 Departamentul	**Departamentul de Informatica**
1.4 Domeniul de studii	**Informatica**
1.5 Ciclul de studii	**Licenta**
1.6 Programul de studiu / Calificarea	**Informatica**

2. Date despre disciplină

2.1 Denumirea disciplinei (ro) (en)	**Aspecte pragmatice în programare** **(Pragmatic issues in programming)**	

2.2 Titularul activităţilor de curs				**Lect. dr. Radu Lupsa**					
2.3 Titularul activităţilor de seminar				**Lect. dr. Radu Lupsa**					
2.4 Anul de studiu	3	2.5 Semestrul	2	2.6. Tipul de evaluare	C	2.7 Regimul disciplinei	optional		
2.8 Codul disciplinei		MLE5056							

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

3.1 Număr de ore pe săptămână	3	Din care: 3.2 curs	2	3.3 seminar/laborator	1		
3.4 Total ore din planul de învăţământ	36	Din care: 3.5 curs	24	3.6 seminar/laborator	12		

Distribuţia fondului de timp:	ore
Studiul după manual, suport de curs, bibliografie şi notiţe	35
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren	25
Pregătire seminarii/laboratoare, teme, referate, portofolii şi eseuri	60
Tutoriat	5
Examinări	2
Alte activităţi:	

3.7 Total ore studiu individual	127
3.8 Total ore pe semestru	175
3.9 Numărul de credite	7

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	Metode evoluate de programare	
4.2 de competenţe	Deprinderi medii de programare	

5. Condiţii (acolo unde este cazul)

5.1 De desfăşurare a cursului		
5.2 De desfăşurare a seminarului/laboratorului	Laborator cu calculatoare; medii de programare pentru C++, Java, .NET, python	

6. Competenţele specifice acumulate

Competenţe profesionale	C2.1 Identificarea de metodologii adecvate de dezvoltare a sistemelor software C2.3 Utilizarea metodologiilor, mecanismelor de specificare ?i a mediilor de dezvoltare pentru realizarea aplica?i	
Competenţe transversale	CT1 Aplicarea regulilor de munca organizata si eficienta, a unor atitudini responsabile fata de domeniul didactic-stiintific, pentru valorificarea creativa a propriului potential, cu respectarea principiilor si a normelor de etica profesionala CT3 Utilizarea unor metode si tehnici eficiente de învatare, informare, cercetare si	

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	Imbunatatirea generata a eficientei programării Abordarea programării dintr-o perspectiva pragmatica	
7.2 Obiectivele specifice	Imbunatatirea eficientei programării printr-o abordare disciplinata	

	• A avea în vedere sarcinile consumatoare de timp în timpul programaraii, precum și uneltele pentru evitarea lor.

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii	
1. Development speed, long-term versus short-term speed. Complexity as the main asymptotic slow-down factor. The role of a disciplined, systematic approach.	• Interactive exposure • Explanation • Conversation • Didactical demonstration		
2. Programming discipline: Tracking changes and (automated) testing: goals, issues, best practices.	• Interactive exposure • Explanation • Conversation • Didactical demonstration		
3. Programming discipline: *One Responsibility Rule* principle, *Don't Repeat Yourself* principle, Coupling and cohesion. Refactoring.	• Interactive exposure • Explanation • Conversation • Didactical demonstration		
4. Programming discipline: code documentation. Pre/post conditions, border cases, well-chosen identifiers, tools.	• Interactive exposure • Explanation • Conversation • Didactical demonstration		
5. Programming discipline: Undefined behaviour, implementation defined behaviour, premature	• Interactive		

optimization, good optimization.	exposure • Explanation • Conversation • Didactical demonstration	
6. Programming discipline: defensive programming. assert() on pre/post conditions and invariants. Input data validation. Fail fast principle.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
7. Programming discipline: Input data validation, efficient diagnosing of errors, secure code.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
8. Testing and debugging techniques: IDE debugger, assert(), core dumps, regression tests, logging and log filtering.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
9. Patterns and techniques: Classes: value semantic vs. object semantic. Immutable classes.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
10. Patterns and techniques: Constructors, destructors, resources and invariants. RAII.	• Interactive exposure • Explanation	

	• Conversation • Didactical demonstration	
11. Patterns and techniques: exceptions. Exception safety levels.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	
12. Patterns and techniques: multi-threading patterns.	• Interactive exposure • Explanation • Conversation • Didactical demonstration	

Bibliografie

1. Michael Howard and David LeBlanc: *Writing Secure Code*, MicrosoftPress, 2003.
2. Herb Sutter, Andrei Alexandrescu: *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices.* Addison-Wesley, 2010.
3. Martin Fowler and others: *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.
4. Robert C. Martin: *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall.
5. Andrew Hunt , David Thomas: *The Pragmatic Programmer: From Journeyman to Master.* Addison-Wesley, 2000.
6. Marshall P. Cline, Greg Lomow, Mike Girou: *C++ FAQs (2nd Edition)*. Addison-Wesley, 1999.

8.2 Seminar / laborator	Metode de predare	Observaţii
1. Introduction, administrative issues. Code examples. Programming discipline: Tracking changes and (automated) testing.	Dialogue, debate, case study, guided discovery	
2. Programming discipline: One Responsibility Rule principle, Don’t Repeat Yourself principle, Coupling and cohesion. Refactoring. Code	Dialogue, debate, case study, guided discovery	

documentation. Pre/post conditions, border cases, well-chosen identifiers, tools.		
3. Programming discipline: Undefined behaviour, implementation defined behaviour, premature optimization, good optimization. Defensive programming. assert() on pre/post conditions and invariants. Input data validation. Fail fast principle.	Dialogue, debate, case study, guided discovery	
4. Programming discipline: Input data validation, efficient diagnosing of errors, secure code. Testing and debugging techniques: IDE debugger, assert(), core dumps, regression tests, logging and log filtering.	Dialogue, debate, case study, guided discovery	
5. Patterns and techniques: Classes: value semantic vs. object semantic. Immutable classes. Constructors, destructors, resources and invariants. RAII.	Dialogue, debate, case study, guided discovery	
6. Patterns and techniques: exceptions. Exception safety levels. Multi-threading patterns.	Dialogue, debate, case study, guided discovery	

Bibliografie

1. Michael Howard and David LeBlanc: *Writing Secure Code*, MicrosoftPress, 2003.
2. Herb Sutter, Andrei Alexandrescu: *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices.* Addison-Wesley, 2010.
3. Martin Fowler and others: *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.
4. Robert C. Martin: *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall.
5. Andrew Hunt , David Thomas: *The Pragmatic Programmer: From Journeyman to Master.* Addison-Wesley, 2000.
6. Marshall P. Cline, Greg Lomow, Mike Girou: *C++ FAQs (2nd Edition)*. Addison-Wesley, 1999.

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

- Continutul cursului este legat de experiența practica.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 metode de evaluare	10.3 Pondere din nota finală	

10.4 Curs			
10.5 Seminar/laborator	- cunoasterea principiilor de baza discutate la curs și cunoasterea modului de aplicare - recunoasterea punctelor problematice într-un program - găsirea metodelor de evitare a punctelor problematice	Verificarea laboratoarelor	50%
	- demonstrarea intelegerii principiilor prin realizarea unui mini-proiect	Verificarea proiectului	50%
10.6 Standard minim de performanţă			
Media minim 5			

Data completării Semnătura titularului de curs Semnătura titularului de seminar

....

Data avizării în departament Semnătura directorului de departament

..... ….....