

SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	Babeş Bolyai University
1.2 Faculty	Faculty of Mathematics and Computer Science
1.3 Department	Department of Computer Science
1.4 Field of study	Computer Science
1.5 Study cycle	Master
1.6 Study programme / Qualification	Software Engineering

2. Information regarding the discipline

2.1 Name of the discipline		Software design					
2.2 Course coordinator		Prof.PhD. Bazil Parv					
2.3 Seminar coordinator		Prof.PhD. Bazil Parv					
2.4. Year of study	1	2.5 Semester	2	2.6. Type of evaluation	E	2.7 Type of discipline	compulsory

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	3	Of which: 3.2 course	2	3.3 seminar/laboratory	1
3.4 Total hours in the curriculum	42	Of which: 3.5 course	28	3.6 seminar/laboratory	14
Time allotment:					Hours
Learning using manual, course support, bibliography, course notes					30
Additional documentation (in libraries, on electronic platforms, field documentation)					30
Preparation for seminars/labs, homework, papers, portfolios and essays					70
Tutorship					14
Evaluations					14
Other activities:					-
3.7 Total individual study hours			158		
3.8 Total hours per semester			200		
3.9 Number of ECTS credits			8		

4. Prerequisites (if necessary)

4.1. curriculum	<ul style="list-style-type: none"> • Fundamentals of programming • Object-oriented programming • Programming paradigms
4.2. competencies	<ul style="list-style-type: none"> • Average programming skills

5. Conditions (if necessary)

5.1. for the course	<ul style="list-style-type: none"> • Videoprojector, Internet access
5.2. for the seminar /lab activities	<ul style="list-style-type: none"> • Computers, Internet access, UML tool

6. Specific competencies acquired

Professional competencies	<ul style="list-style-type: none"> • Understanding of the software design from the engineering perspective; • Understanding of the software design concepts and principles • Understanding of the software design process and its activities; • Proficient use of tools and languages specific to software systems development • Knowing the specifics of main architectural and design patterns and how to apply them to specific projects.
Transversal competencies	<ul style="list-style-type: none"> • Professional communication skills; concise and precise description, both oral and written, of professional results, • Independent and team work capabilities; able to fulfill different roles • Antepreneurial skills;

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	<ul style="list-style-type: none"> • Know and understand fundamental concepts of software design. • Be able to apply the appropriate architectural and design patterns to different programming projects
7.2 Specific objective of the discipline	<p>At the end of the course, students</p> <ul style="list-style-type: none"> • know the main concepts and principles of software design • have a good understanding of the following terms: software architecture definition(s), architectural styles and models, architecture definition language(s); detailed design; design pattern, construction design; • learn the importance of architectural and detailed design and how to use tools for these tasks; • know several software system types (taken from real-world applications) and the best recommended architectural styles and design patterns.

8. Content

8.1 Course	Teaching methods	Remarks
1. <i>Introduction to software engineering design.</i> Motivation and general design concepts. Overview of the software engineering design	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
2. <i>Software design fundamentals.</i> UML Fundamentals.	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
3. <i>Software design fundamentals.</i> UML structural modeling. UML behavioral modeling	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
4. <i>Fundamentals of software architecture.</i> Fundamentals of requirements engineering. Designing the software architecture	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
5. <i>Architectural styles and patterns.</i> Overview and history of styles and patterns. Data-centered and data-flow systems	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
6. <i>Architectural styles and patterns.</i> Distributed systems. Interactive and hierarchical systems	<ul style="list-style-type: none"> • Interactive exposure • Explanation 	

	<ul style="list-style-type: none"> • Conversation • Didactical demonstration 	
7. <i>Detailed design.</i> Overview of the detailed design. Structural and behavioral design of components. Design principles	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
8. <i>Creational design patterns in the detailed design.</i> Abstract Factory. Factory Method	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
9. <i>Creational design patterns in the detailed design.</i> Builder. Prototype. Singleton.	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
10. <i>Structural design patterns in the detailed design.</i> Adapter. Composite. Façade	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
11. <i>Behavioral design patterns in the detailed design.</i> Iterator. Observer.	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
12. <i>Construction design.</i> Flow-, state-, and table-based construction design. Programming design language, styles, and quality evolution.	<ul style="list-style-type: none"> • Interactive exposure • Explanation • Conversation • Didactical demonstration 	
13. <i>Software design management.</i> Design management framework. Planning	<ul style="list-style-type: none"> • Interactive exposure • Conversation 	
14. <i>Software design management.</i> Implementation and termination. Final review	<ul style="list-style-type: none"> • Interactive exposure • Conversation 	

Bibliography

1. OTERO, C.E.: *Software Engineering Design*, CRC Press, 2012.
site: <http://softwareengineeringdesign.com/Default.htm>
2. BASS, L., CLEMENTS, P., KAZMAN R.: *Software Architecture in Practice*, 2nd ed., Addison-Wesley, 2003
3. KRUCHTEN, PH.: *Architectural Blueprints – The 4+1 View Model of Software Architecture*, IEEE Software 12 (6), 1995, pp. 42-50.
4. SHAW, M.: *The Coming-of-Age of Software Architecture Research*, in Proc. of the 23rd ICSE, IEEE Comp. Soc. 2001, 656, [<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/vit/ftp/pdf/shaw-keynote-rev.pdf>]
5. SHAW, M., GARLAN, D.: *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Administrivia	Conversation, debate, case studies, presentations	Seminar is organized as a total of 14 hours – 2 hours every other week
2. Establishing the target application. First miniproject started	Conversation, debate, case studies, examples	
3. Work on miniproject 1	Exposure, debate, case studies, examples	
4. Miniproject 1 due. Second miniproject started	Exposure, debate, case studies, examples	
5. Work on miniproject 2	Exposure, debate, case studies, examples	
6. Miniproject 2 due. Detailed design issues	Exposure, debate, case	

	studies, examples	
7. Final review and project evaluation	Exposure, live demos	
Bibliography		
Students will search and use software design documentation		
<ul style="list-style-type: none"> • on the department server (win/labor/Romana/master/SED) • on the web, using main CS databases 		

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

<ul style="list-style-type: none"> • This course follows the IEEE and ACM Curricula Recommendations for Software Engineering studies; • Courses with similar content are taught in the major universities in Romania offering similar study programs; • Course content is considered very important by the software companies for improving average software development skills
--

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	<ul style="list-style-type: none"> • knowing the basic concepts of software design • applying different architectural styles and design patterns to different problem domains 	Written exam	40%
10.5 Seminar/lab activities	<ul style="list-style-type: none"> • be able to study and review literature regarding software design • be able to solve a problem using different architectural and design patterns • be able to evaluate a software design 	<ul style="list-style-type: none"> • Miniproject 1 work • Miniproject 2 work • Seminar/lab attendance • Default 	20% 20% 10% 10%
10.6 Minimum performance standards			
<ul style="list-style-type: none"> • At least grade 5 (from a scale of 1 to 10) at written exam and miniproject work. 			

Date

April 30, 2014

Date of approval

.....

Signature of course coordinator

Prof.PhD. Bazil PARV

Signature of seminar coordinator

Prof.PhD. Bazil PARV

Signature of the head of department

Prof.PhD. Bazil PARV