**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 | **Component-Based Programming** |

**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 | • Fundamentals of programming<br>• Object-oriented programming<br>• Programming paradigms |
| 4.2. competencies | • Average programming skills |

**5. Conditions** (if necessary)

| | |
|---|---|
| 5.1. for the course | • Videoprojector, Internet access |
| 5.2. for the seminar /lab activities | • Computers, Internet access, UML tool |

**6. Specific competencies acquired**

| | |
|---|---|
| **Professional competencies** | • Understanding of the software design from the engineering perspective;<br>• Understanding of the software design concepts and principles<br>• Understanding of the software design process and its activities;<br>• Proficient use of tools and languages specific to software systems development<br>• Knowing the specifics of main architectural and design patterns and how to apply them to specific projects. |
| **Transversal competencies** | • Professional communication skills; concise and precise description, both oral and written, of professional results,<br>• Independent and team work capabilities; able to fulfill different roles<br>• Antepreneurial skills; |

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

| | |
|---|---|
| 7.1 General objective of the discipline | • Know and understand fundamental concepts of software design.<br>• Be able to apply the appropriate architectural and design patterns to different programming projects |
| 7.2 Specific objective of the discipline | At the end of the course, students<br>• know the main concepts and principles of software design<br>• 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;<br>• learn the importance of architectural and detailed design and how to use tools for these tasks;<br>• 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. *Introduction to software engineering design*. Motivation and general design concepts. Overview of the software engineering design | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 2. *Software design fundamentals*. UML Fundamentals. | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 3. *Software design fundamentals*. UML structural modeling. UML behavioral modeling | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 4. *Fundamentals of software architecture*. Fundamentals of requirements engineering. Designing the software architecture | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 5. *Architectural styles and patterns*. Overview and history of styles and patterns. Data-centered and data-flow systems | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 6. *Architectural styles and patterns*. Distributed systems. Interactive and hierarchical systems | • Interactive exposure<br>• Explanation | |

| | | |
|---|---|---|
| | • Conversation<br>• Didactical demonstration | |
| 7. *Detailed design*. Overview of the detailed design. Structural and behavioral design of components. Design principles | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 8. *Creational design patterns in the detailed design*. Abstract Factory. Factory Method | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 9. *Creational design patterns in the detailed design*. Builder. Prototype. Singleton. | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 10. *Structural design patterns in the detailed design*. Adapter. Composite. Façade | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 11. *Behavioral design patterns in the detailed design*. Iterator. Observer. | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 12. *Construction design*. Flow-, state-, and table-based construction design. Programming design language, styles, and quality evolution. | • Interactive exposure<br>• Explanation<br>• Conversation<br>• Didactical demonstration | |
| 13. *Software design management*. Design management framework. Planning | • Interactive exposure<br>• Conversation | |
| 14. *Software design management*. Implementation and termination. Final review | • Interactive exposure<br>• 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 serch and use software design documentation

- 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**

- This course follows the IEEE and ACM Curriculla 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 | • knowing the basic concepts of software design<br>• applying different architectural styles and design patterns to different problem domains | Written exam | 40% |
| 10.5 Seminar/lab activities | • be able to study and review literature regarding software design<br>• be able to solve a problem using different architectural and design patterns<br>• be able to evaluate a software design | • Miniproject 1 work<br>• Miniproject 2 work<br>• Seminar/lab attendance<br>• Default | 20%<br>20%<br>10%<br><br>10% |
| 10.6 Minimum performance standards | | | |
| • At least grade 5 (from a scale of 1 to 10) at written exam and miniproject work. | | | |

| Date | Signature of course coordinator | Signature of seminar coordinator |
|---|---|---|
| April 30, 2014 | Prof.PhD. Bazil PARV | Prof.PhD. Bazil PARV |
| Date of approval | | Signature of the head of department |
| ..... | | Prof.PhD. Bazil PARV |