
# Requirements and Architecture Modeling in Software Engineering Courses

Tihana Galinac Grbac
Željka Car
Marin Vuković

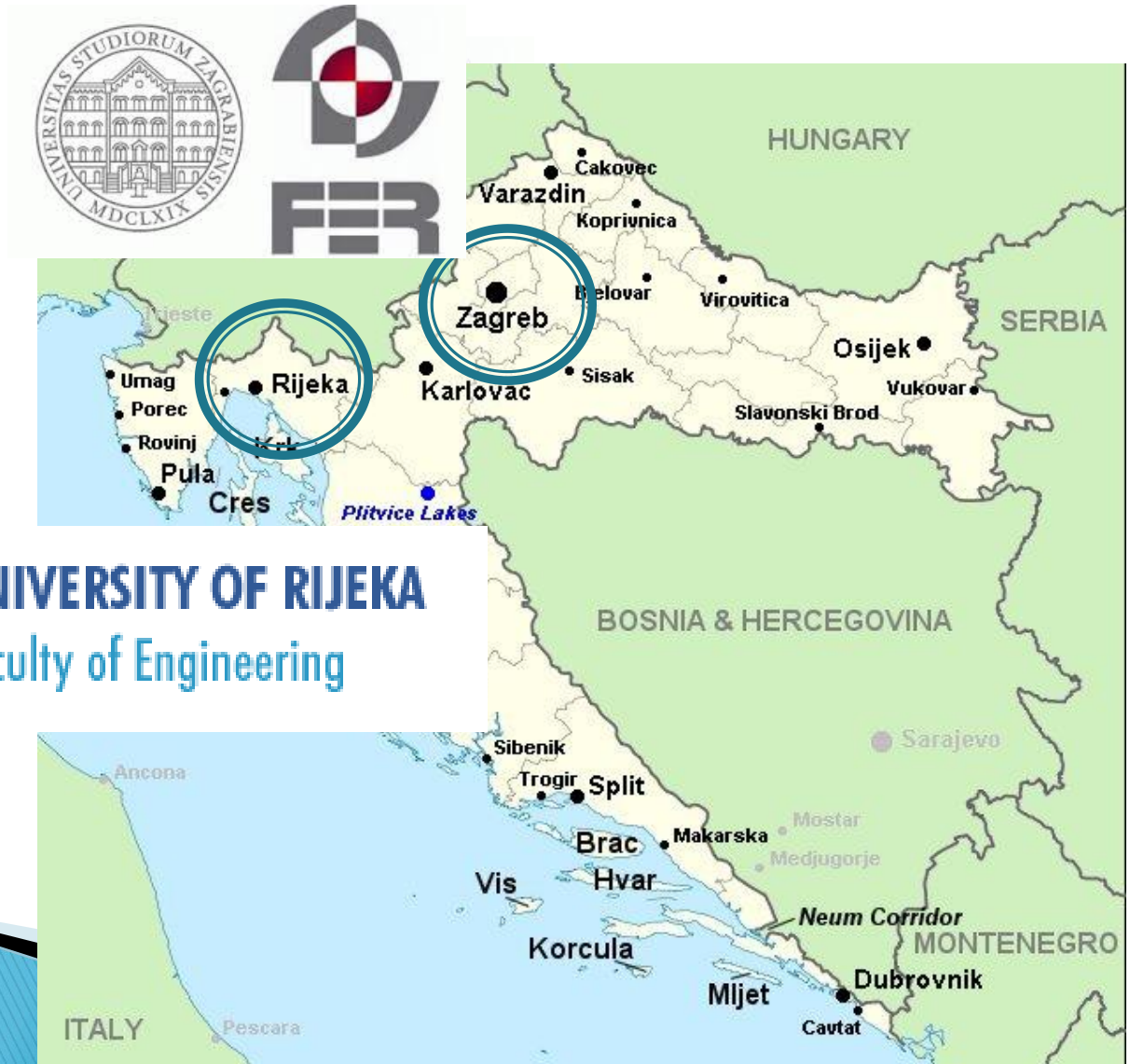
# Motivation

- In Software Engineering important is:
  - Understanding of the SRAM (software requirements and architecture modeling) process and the main purpose of these processes within the software lifecycle.
- When to study SRAM?
  - Majority of study curricula, courses with SRAM learning outcomes are left to final undergraduate level or graduate study programme
- Main problems learning SRAM?
  - students lack abstraction, industrial perspective and critical reasoning abilities

# We aim to

- Discuss learning approaches for requirements and architecture modeling
- Define effective learning strategies aiming to prepare students for real software engineering projects.
- Share best practices

# Case study




UNIVERSITY OF RIJEKA
Faculty of Engineering

# Case study

UNIVERSITY OF RIJEKA
Faculty of Engineering

*Course:*
*Software Engineering*
*Study programme:*
undergraduate *Computing*

*Course:*
*Telecommunication Software Development*
*Study programme:*
*graduate Telecommunications and Informatics*

# Software Engineering Course@RITEH

UNIVERSITY OF RIJEKA
Faculty of Engineering

- 2nd year (2nd semester) undergraduate, 7 ECTS
- Aim of the course:
  - introduces students to software lifecycle and related software engineering processes and activities.
- Course design:
  - 45 lecturing + 15 hour of laboratory work in 15 weeks period.
- Students:
  - On average 50 students attend
  - All are full time students and without previous expertise working in software development
  - low previous experience in programming, although they have basic programming skills in C.

# Software Engineering Course@RITEH

UNIVERSITY OF RIJEKA
Faculty of Engineering

- **Approach:**
- Lectures mostly follow van Vliet Software Eng. book
  - aiming to learn fundamentals related to software requirements analysis (1 week), and architecture modeling (1 week)
  - Involves lectures on UML (2 weeks)
- Laboratory work is organized in projects
  - 5-6 students in group, develop Android application (their choice)
  - Each student has its own role (project manager, software arhcitect, etc.)
- Methods and Tools
  - Document templates are tailored version of IEEE 830 and 1016 standards (based on years of industrial practice)
  - ArgoUML, Class Responsibility Collaboration Cards, week reviews
- Learnng strategy: lecture+project+reflections

# Software Engineering Course@RITEH

UNIVERSITY OF RIJEKA
Faculty of Engineering

- **Experiences in working with students:**
- Students have problem to understand the difference between software requirements and software design process
- Level of abstraction is very limited
  - requirements rarely according to SMART* definition.
  - functions that could be generalized are often missed
  - Weak use object oriented concepts such as generalization, inheritance, encapsulation
- Do not understand definition of quality attributes
- Student use iterations as reaction to week meetings

*Simple, Measurable, Assignable, Realistic and Time related

# Software Engineering Course@RITEH

UNIVERSITY OF RIJEKA
Faculty of Engineering

- **Recommendations:**
  - Learning fundamentals based on minimal technology use
  - Learning based on examples
  - Frequent discussions, feadbacks with teacher
  - Reflections building links between
    - theory provided in lectures and practice obtained in projects
    - their projects and industrial experience
  - Provide funny and extreme examples with cash flow – show the benefit of these processes
  - Interesting idea would be that students exchange projects between the groups and try to continue development
  - Object oriented course should be before Software Eng.

# *Telecommunication Software Development* @FER

- 1st year (2nd semester) graduate
- Aim of the course:
  - Specialisation knowledge on software product lifecycle processes and software development models.
- Course design:
  - 30 lecturing + 12 hour of laboratory work in 12 weeks period.
- Students:
  - On average 40 students attend
  - All are full time students and without previous expertise working in software development

# Telecommunication Software Development @FER

- Highly structured approach
- Students worked in the group (2 students).
- Reorganized order of lectures: started with the importance of business idea instead of classical lessons about SRAM processes
  - Student project assignment (2 approaches predefined and free project)
  - Documentation – based on RUP
  - Software specification and Requirements management tools
    - introspection, brainstorming, and literature analysis and rarely interviewing
    - IBM Rational Requisite Pro, IBM Rational Rose
- Examination:
  - written and oral exams

# *Telecommunication Software Development* @FER

- **Approach**
- Highly structured approach
- Students worked in the group (2 students).
  - Student project assignment (2 approaches predefined and free project)
  - Documentation – based on RUP
    - Software specification involving description of business idea technical aspects
  - Software specification and Requirements management tools
    - Argumented use of tools: introspection, brainstorming, and literature analysis and rarely interviewing
    - IBM Rational Requisite Pro, IBM Rational Rose
- Examination:
  - written and oral exams

# Telecommunication Software Development @FER

- **Recommendations**
- Specialisation course at higher years teaching SRAM is best practice for better understanding foundations
- Students are better motivated if they develop their own project idea, use and argument the use of their own tools
- Besides software development related topics, the students were taught about presenting their idea to investors, intellectual property, patents and possible software licensing models.
- Use of tehcnologies like RUP in SRAM is highly recommended but in specialiation courses

# Experiences

- Although fundamentals related to SRAM are studied at undergraduate level specialisation courses teaching SRAM in application domains are recommended
- Projects are good practice for teaching SRAM
- Approach to the project
  - Basic courses should be with minimum technology, well specified, in well set up environments, followed with number of examples
  - Specialisation courses shoud provide freadom to choose, engage research approach

# Conclusion

- Key for effective student learning are innovative learning strategies based on
  - student creativity,
  - frequent reflections from teacher and other students
  - Working in teams and
  - engaging communication and frequent discussion.
- We belive that Software requirements and architecture modeling (SRAM) is a complex area and should be adequately implemented within study programme.
  - Firstly, at undergraduate level, after all programming courses the foundations of SRAM should be established.
  - On higher educational levels this fundamentals have to be extended with techniques and principles aligned to business needs.

This work has been supported in part by Croatian Science Foundation's funding of the project UIP-2014-09-7945.

# Questions?