**Study programme(s):** Computer Science

**Level:** academic aster studies

**Course title:** Software engineering in critical systems

**Lecturer:** Zoran D. Budimac

**Status:** Elective

**ECTS:** 4

**Requirements:** none

**Learning objectives**

The course aims to present and critically analyze critical systems as a part of software engineering and information technology systems. Requirements for such systems are to be deeply understood and the role of formal approaches in the life cycle of critical systems is to be deeply understood.

**Learning outcomes**

*Minimum:*

At the of the course it is expected that a successful student will be able to critically appreciate the current classifications of critical systems, including international standards and usage of formal methods in a life cycle of such systems. Also, it is expected that the student will be able to adopt fundamental conclusions of time-dependent systems in the phases of requirements and design, including planning techniques.

*Desirable:*

At the end of a course it is expected that a successful student will show the ability to critically evaluate the usage of temporal logic in engineering and reengineering of critical systems.

**Syllabus**

*Theoretical instruction*

Theoretical basics of critical systems, classifications and analysis, including examples and efforts towards standardizations. Time dependent systems and technical issues in relation to them, the role of formal approaches, software in critical systems and real-time systems, formal approaches in life cycle of critical systems, examples of applications. Models of critical systems, computation calculus, interval-temporal logic, refinement calculus, abstraction calculus, and evolution.

*Practical instruction*

Introduction to formal approaches based on models, logic and process algebra, syntax and semantics of temporal logic with overview of tools such are executable subset of temporal logic TEMPURA, model of temporal agents and algebraic laws with examples, examples of abstract execution of evolution. Examples implemented with tools such is „Ana Tempura“.

**Literature**

*Recommended*

1. Ian Sommerville, 'Software Engineering, 9th edition', 2010 (chapters 16, 17, 18 and 21)
2. Ben Moszkowski, Executing Temporal Logic Programs, Cambridge Univ. Press (http://www.cse.dmu.ac.uk/~cau/papers/tempura-book.pdf)
3. Michael Huth and Mark Ryan, Logic in Computer Science: Modeling and Reasoning about Systems, Cambridge University Press, 2000
4. Anderson, Ross , Security Engineering, Wiley, 2001
5. Boyd, Colin and Mathuriam, Anish, Protocols for Authentication and Key Establishment, Springer, 2003.

| **Weekly teaching load** | | | | |
|---|---|---|---|---|
| Lectures: 1 | Exercises: 2 | Practical Exercises: 0 | Student research: | Other: |

**Teaching methodology**

During lecture classes the classical methods are used with the aid of overhead projector. Exercises are mostly consisting of case study analyses. Assignments are mostly practical, whose aim is to practically apply principles covered during lectures and exercises, using appropriate tools.

**Grading method (maximal number of points 100)**

| **Pre-exam oblications** | **points** | **Final exam** | **points** |
|---|---|---|---|
| Practical assignments | **60** | Oral exam | 40 |