FRI-TEHNOLOGIJA PROGRAMSKE OPREME

UČNI NAČRT PREDMETA/COURSE SYLLABUS

Predmet:	FRI-TEHNOLOGIJA PROGRAMSKE OPREME
Course title:	SOFTWARE ENGINEERING
Članica nosilka/UL Member:	

Študijski programi in stopnja	Študijska smer	Letnik	Semestri	Izbirnost
Upravna informatika, prva stopnja, univerzitetni	Upravna informatika (študijski program)	3. letnik	2. semester	obvezen

Univerzitetna koda predmeta/University course code:	0045568
Koda učne enote na članici/UL Member course code:	1072

Predavanja /Lectures	Seminar /Seminar	Vaje /Tutorials	Klinične vaje /Clinical tutorials	Druge oblike študija /Other forms of study	Samostojno delo /Individual student work	ECTS
45	10	20	0	0	105	6

Nosilec predmeta/Lecturer:	Dejan Lavbič
Vrsta predmeta/Course type:	obvezni/core

Jeziki/Languages:	Predavanja/Lectures:	Slovenščina
	Vaje/Tutorial:	Slovenščina

Pogoji za vključitev v delo oz. za opravljanje študijskih obveznosti:	Prerequisites:

Vsebina:	Content (Syllabus outline):

predavanja:
Temeljne vsebine:
1. Osnovni koncepti
  1. Kaj je tehnologija programske opreme
  2. Faze v razvoju programske opreme
  3. Modeli razvojnega procesa (npr. slapovni, inkrementalni, agilni)
2. Orodja in okolja za razvoj programske opreme
  1. Upravljanje konfiguracije in nadzor različic
  2. Avtomatizirana gradnja in zvezna integracija
  3. Orodja za pomoč pri analizi in načrtovanju (orodja CASE)
3. Opredelitev zahtev
  1. Ugotavljanje zahtev
  2. Dokumentiranje zahtev
  3. Nefunkcionalne zahteve
  4. Ovrednotenje in uporaba specifikacije zahtev
4. Modeliranje
	lectures:
	Core topics:
	1. Basic concepts
	  1. What is Software Engineering
	  2. Phases in the development of software
	  3. Software process models (e.g., waterfall, incremental, agile)
	2. Software engineering tools and environments
	  1. Configuration management and version control
	  2. Automated builds and continuous integration
	  3. Requirements analysis and design modeling tools (CASE tools)
	3. Requirements engineering
	  1. Requirements elicitation
	  2. Requirements documentation
	  3. Non-functional requirements

1. Klasične tehnike (npr. diagrami entite-razmerja, diagrami podatkovnih tokov, diagrami prehajanja stanj)
2. Jezik UML

5. Načrtovanje programske opreme
   1. Principi načrtovanja (abstrakcija, modu-larnost, delitev odgovornosti, skrivanje detajlov, sklopljenost in povezanost, ponovna uporaba)
   2. Klasične metode načrtovanja (funkcio-nalna dekompozicija, strukturirane metode)
   3. Objektno usmerjena analiza in načrtovanje (RUP)
   4. Načrtovalski vzorci
6. Programiranje
   1. Dobre prakse (defenzivno programiranje, varno programiranje, uporaba mehanizmov za lovljenje izjem, prestrukturiranje kode)
   2. Standardi kodiranja
   3. Strategije za integracijo
7. Testiranje programske opreme
   1. Testiranje in življenjski cikel
   2. Strukturirani pregledi
   3. Testiranje enot, integracijsko testiranje, testiranje funkcionalnosti, sistemsko testiranje
   4. Testiranje po metodi črne in bele škatle
   5. Regresijsko testiranje in avtomatizacija testiranja
   6. Testno voden razvoj
8. Vzdrževanje in nadgradnje
   1. Glavni vzroki problemov pri vzdrževanju
   2. Značilnosti programske opreme, ki je primerna za vzdrževanje
   3. Obratno inženirstvo in prestrukturiranje
   4. Ponovna uporaba programske opreme

Izbirne vsebine:

1. Modeli za izboljšanje kakovosti razvojnega procesa
   1. CMMI (Capability Maturity Model Integration) for Development
   2. PSP (Presonal Software Process)
   3. TSP (Team Software Porcess)
2. Agilni in vitki razvoj programske opreme
   1. Scrum
   2. Extreme Programming
   3. Kanban
   4. Uravnoteženje agilnosti in discipline

vaje:

Skupinsko delo na projektih, ki so čim bolj podobni realnim. Projekte lahko predlagajo študenti sami, potrditi pa jih morata nosilec predmeta in asistent. Lahko pa problem definira nosilec predmeta v sodelovanju z ustreznim naročnikom. Velikost skupin je omejena na štiri študente, študenti pa lahko sami izbirajo, s kom bodo delali. Vsaka skupina mora določiti vodjo projekta in ostale vloge, za katere meni, da so potrebne za realizacijo projekta. Študente

4. Evaluation and use of requirements specifications

4. Modeling
   1. Classic modeling techniques (e.g., entity-relationship diagrams, data flow diagrams, finite state machines)
   2. The Unified Modeling Language
5. Software design
   1. Software design principles (abstraction, modularity, separation of concerns, information hiding, coupling and cohesion, reuse)
   2. Classical design methods (functional decomposition, data flow design)
   3. Object-oriented analysis and design (Rational Unified Process)
   4. Design patterns
6. Software construction
   1. Coding practices (defensive coding practices, secure coding practices, use of exception handling mechanisms, refactoring)
   2. Coding standards
   3. Integration strategies
7. Software testing
   1. Testing and the software life cycle
   2. Inspections and walkthroughs
   3. Unit, integration, validation, and system testing
   4. Black-box and white-box testing techniques
   5. Regression testing and test automation
   6. Test-driven development
8. Software maintenance and evolution
   1. Major causes of maintenance problems
   2. Characteristics of maintainable software
   3. Reverse engineering and refactoring
   4. Software reuse

Elective topics:

1. Process improvement models
   1. CMMI (Capability Maturity Model Integration) for Development
   2. PSP (Personal Software Process)
   3. TSP (Team Software Process)
2. Agile and lean software development
   1. Scrum
   2. Extreme Programming
   3. Kanban
   4. Balancing agility and discipline

lab practice:

Team-work on almost real projects. Projects can be proposed by students themselves and approved by instructors or can be defined by teacher in co-operation with real customers. Group sizes are restricted to four and students are given the opportunity to decide who they should work with. Each group is asked to nominate a Project Manager and identify specific roles which they feel will be important in managing the group. Students are encouraged to work on the project outside the

spodbujamo, da delajo na projektu tudi izven ur, ki so po urniku predvidene za izvedbo predmeta. Za spremljanje dela na projektih definiramo mejnike (kontrolne točke), na katerih študenti pokažejo, kaj so že naredili, in dobijo nasvete za nadaljnje delo.	officially scheduled hours. There are several progress meetings (milestones) with the tutor before the final presentation takes place.

Temeljna literatura in viri/Readings:

1. H. van Vliet: Software Engineering, Third Edition, John Wiley & Sons, 2008.
2. I. Jacobson, G. Booch, J. Rumbaugh: The unified software development process, Addison-Wesley, 1999.

Dodatno:

1. M. Seidl, M. Scholz, C. Huemer, G. Kappel: UML @ Classroom, An Introduction to Object-Oriented Modeling, Springer, 2015.
2. CMMI® for Development (CMMI-DEV), Version 1.3 CMU/SEI-2010-TR-033, Software Engine-ering Institute, Carnegie Mellon University, 2010.
3. K. Schwaber: Agile Project Management with Scrum, Microsoft Press, 2004.

Cilji in kompetence:	**Objectives and competences:**
Cilj predmeta je predstaviti pregled aktivnosti v življenjskem ciklu razvoj programske opreme, ki so potrebne ne glede na to, kateri model razvojnega procesa se uporablja. Da bi obvladali te aktivnosti, študentje kombinirajo teoretična znanja s praktičnim delom na (skoraj) realnem projektu. Predvidene kompetence: • sposobnost opredelitve, razumevanja in reševanja kreativnih strokovnih izzivov na področju računalništva in informatike; • sposobnost uporabe pridobljenega znanja pri samostojnem delu za reševanje tehničnih in znanstvenih problemov na področju računalništva in informatike; sposobnost nadgradnje pridobljenega znanja, • sposobnost skupinskega dela v profesionalnem delovnem okolju; vodenje manjše strokovne skupine; • sposobnost administrativnega vodenja procesov, povezanih z raziskovanjem, industrijo, izobraževanjem in drugimi področji; • sposobnost razumevanja in uporabe znanja računalništva in informatike na drugih tehničnih in relevantnih področjih (ekonomija, organizacijske vede ipd.); • praktično znanje in spretnosti, potrebne za uspešno strokovno delo na področju računalništva in informatike; • samostojno reševanje zahtevnih razvojnih, inženirskih in organizacijskih nalog kot tudi povprečno zahtevnih raziskovalnih nalog na področju računalništva in informatike.	The aim of the course is to provide an overview of activities in the software development life cycle that must be preformed regardless of the process model used. In order to master these activities, the students combine theoretical knowledge with practical work on a quasi-real software project. The competences the students gain are: • the ability to define, understand and solve creative professional challenges in computer and information science; • the ability to apply acquired knowledge in independent work for solving technical and scientific problems in computer and information science; the ability to upgrade acquired knowledge. • the ability of teamwork within the professional environment; management of a small professional team; • the ability for administrative management of processes related to research, industry, education and other fields; • the ability to understand and apply computer and information science knowledge to other technical and relevant fields (economics, organisational science, etc); • practical knowledge and skills of computer hardware, software and information technology necessary for successful professional work in computer and information science; • the ability to tackle demanding developmental, engineering, and organisational tasks as well as moderately demanding research tasks in their fields of study.

Predvideni študijski rezultati:	**Intended learning outcomes:**
Po uspešnem zaključku tega predmeta bo študent zmožen: • poznati in razumeti aktivnosti v življenjskem ciklu razvoja programske opreme	After the compeltion of the course a student will be able to: • know and understand activities in the software development life cycle

• poznati in razumeti različne pristope k razvoju programske opreme • poznati najpomembnejše metodologije za razvoj programske opreme • poznati in razumeti kriterije, ki vplivajo na izbor in prilagoditev razvojne metodologije • uporabljati diagramske tehnike za modeliranje novega sistema • razviti sposobnosti za skupinsko delo, vodenje, načrtovanje in organizacijo, pripravo predstavitev in medosebno komunici-ranje, iskanje informacij, pisno in ustno poročanje • uporabiti pridobljeno znanje pri razvoju realnih softverskih projektov.	• know and understand different approaches to software development • know the most important software development methodologies • know and understand the criteria that affect the choice and adaptation of the development methodology • use diagramming techniques for system modelling • develop professional skills like teamwork, management/leadership, planning and organizing, presentation and communication, information search, written and oral reporting • apply acquired knowledge in the context of a real software development project.

Metode poučevanja in učenja:	**Learning and teaching methods:**
• *Predavanja* z aktivno udeležbo študentov (razlaga, diskusija, vprašanja, primeri, reševanje problemov). • *Laboratorijske vaje* (skupinsko delo na večjem projektu razvoja programske opreme, ki služi kot študija primera za ilustracijo potrebnih aktivnosti). • *Konsultacije* (diskusija, dodatna razlaga, obravnava specifičnih vprašanj).	• *Lectures* with active participation on the part of students (explanation, discussion, questions, examples, problem solving). • *Lab practice* (teamwork on an quasi-real software development project that serves as a case study for illustration of required activities). • *Consultations* (discussion, additional explanation, answers to specific questions).

Načini ocenjevanja:	**Delež/Weight**	**Assessment:**
Način (pisni izpit, ustno izpraševanje, naloge, projekt): Sprotno preverjanje: seminarske naloge, delo na projektu.	50,00 %	Type (examination, oral, coursework, project): Continuing assessment: seminal and project work
Končno preverjanje: pisni izpit Ocene: 6-10 pozitivno, 1-5 negativno (v skladu s Statutom UL).	50,00 %	Final assessment: written exam Grading: 6-10 pass, 1-5 fail

Reference nosilca/Lecturer's references:

Pet najpomembnejših del:

1. MAHNIČ, Viljan. A capstone course on agile software development using Scrum. *IEEE trans. ed.*, Feb. 2012, vol. 55, no. 1, str. 99-106.
2. MAHNIČ, Viljan, HOVELJA, Tomaž. On using planning poker for estimating user stories. *J. syst. softw.*. Sep. 2012, vol. 85, no. 9, str. 2086-2095.
3. MAHNIČ, Viljan, ŽABKAR, Nataša. Measuring progress of Scrum-based software projects. *Electronics and Electrical Engineering*. 2012, vol. 18, no. 8, str. 73-76.
4. MAHNIČ Viljan. Teaching Scrum through team-project work: students' perceptions and teacher's observations. *International journal of engineering education*, 2010, vol. 26, no. 1, str. 96-110.
5. MAHNIČ, Viljan, HOVELJA, Tomaž. Teaching user stories within the scope of a software engineering capstone course: analysis of students' opinions. *International journal of engineering education*, 2014, vol. 30, no. 4, str. 901–915.

Celotna bibliografija je dostopna na SICRISu:
http://sicris.izum.si/search/rsr.aspx?lang=slv&id=4767