PLAN I PROGRAM ZA PREDMET

SE311 PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Napomena: Plan predmeta je važeći od 1.10.2021 do 1.10.2022

Napomena: Ovaj program važi 12 meseci, tj. do početka jesenjeg semestra u školskoj 2022/23.godini, a poslednji ispit se po njemu može polagati u oktobarskom roku 2021.godine. Posle ovih rokova, važi novi Plan za predmet koji se objavljuje oktobra 2022. godine. Studenti koji ne polože ispit do oktobra 2022.godine, rade sve predispitne obaveze ponovo, u skladu sa novim Planom i u skladu sa njim polažu ispit u januarskom roku 2023. godine ili nekom od kasnijih rokova.

PODACI O NASTAVNOM OSOBLJU	
Klasična nastava - Beograd	
Predavanja	Prof. dr Dragan Domazet Dr Borislav Nikolić, docent
Vežbanja	Nebojša Gavrilović
e-mail adresa nastavnika	dragan.domazet@metropolitan.ac.rs
Skype adresa nastavnika	dragan.domazet1
Termini za konsultacije nastavnika preko Skype	Po dogovoru mejlom
e-mail adresa saradnika	nebojsa.gavrilovic@metropolitan.ac.rs
Skype adresa saradnika	n.gavrilovic
Termini za konsultacije saradnika preko Skype	petak od 15 do 17h
Onlajn nastava (preko Interneta)	
Nastavnik/saradnik za konsutacije sa studentima	Nebojša Gavrilović
e-mail adresa nastavnika/saradnika za konsutacije sa studentima	nebojsa.gavrilovic@metropolitan.ac.rs

Skype adresa Nastavnik/saradnik za konsutacije sa studentima	n.gavrilovic
Termini za konsultacije preko Skype	petak od 15 do 17h
Klasična nastava u Centru u Nišu	
Predavanja	dr Milan Tančić, docent
e-mail adresa nastavnika	milan.tancic@metropolitan.ac.rs
Skype adresa nastavnika	milan.tancic89
Termini za konsultacije nastavnika preko Skype	Po dogovoru mejlom
Vežbe i konsultacije u vezi predavanja	Jovana Jović, asistent
e-mail adresa saradnika	jovana.jovic@metropolitan.ac.rs
Skype adresa saradnika	jovana.jovic.metropolitan
Termini za konsultacije saradnika preko Skype	petak od 15 do 17h
PODACI O PREDMETU	
Semestar	5
Preduslovi	SE201
Broj ECTS	8
Broj časova predavanja nedeljno	3
Broj časova grupnih (pokaznih) vežbi nedeljno	1
Broj časova individualnih vežbanja nedeljno	2
Broj časova samostalnog istraživačkog rada nedeljno	0
PODACI O PREDISPITNIM OBAVEZAMA I ISPITU	

Broj domaćih zadataka tokom semestra	15
Maksimalan broj poena za jedan domaći zadatak	1,5
Broj testova tokom semestra	5
Maksimalan broj poena za jedan test	2
Broj kolokvijuma tokom semestra	0
Maksimalan broj poena za jedan kolokvijum	0
Broj projekata tokom semestra	1
Maksimalan broj poena za jedan projekat	25
Broj seminarskih radova tokom semestra	0
Maksimalan broj poena za jedan seminarski rad	0
Maksimalno mogući brojevi poena:	
A) Maksimalan broj poena za zalaganje studenta tokom semestra	**12,5**
B) Maksimalan broj poena za predispitne obaveze	**57,5**
Domaći zadaci (15x 1,5 = 22,5)	22,5
Testovi (5 x 2 = 10)	10
Projekat	25
C) Maksimalan broj poena za ispit	**30**
UKUPAN BROJ POENA (A+B+C):	**100**
Organizacija ispita:	
Vreme trajanja ispita u minutima	180

Forma ispita (obrisati nepotrebne opcije)	Teorijska pitanja i zadaci na računaru
Računarski alati ili pribor koji se koriste na ispitu	PowerDesigner

Literatura (nastavni materijal):

Obavezna literatura :

1. Online nastavni materijal za predmet SE311 Projektovanje i arhitektura softvera, 2020/21, Univerzitet Metropolitan
2. Paul. Clements, Documenting Software Architectures: Views and Beyond, 2nd ed., Pearson Education, 2010. (za lekcije 2-5)
3. C.Lethbrigde, R. Lagariere - Object-Oriented Software Engineering - Practical Software Development using UML and Java - 200511. (za lekciju 6)
4. David Budgen, Software Design, 2nd ed., Addison-Wesley, 2003. (za lekcije 1, 7 i 8)
5. Ian Sommerville, Software Engineering, 9th ed Addison-Wesley, 2011 ili 10th ed., Addison-Wesley, 2011. Pearson 2016 (za lekcije 9-15)

Dodatna literatura :

1. P. Stevens, Using UML – Software Engineering with Objects and Components, Second Edition, Assison-Wesley, Pearson Eduction, 2006
2. R. Pressman, Software Engineering – A Practioner's Approach, Seventh Edition, McGraw Hill Higher Education, 2010
3. Partha Kuchana, Software Architecture Design patterns in Java
4. P.B. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, vol. 12, no. 1995, pp. 42–55.
5. E. Gamma et al., Design Patterns:Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley Professional, 1994.
6. J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.
7. Service-oriened Architecture, Concept, Technology, and Design, autora T. Erl u izdanju Pretince Hall, 2005, ISBN 0-13-185858-0. 8. IEEE. (2003). IEEE Software Engineering Standards Collection on CD-ROM. Los Alamitos, Ca.: IEEEComputer Society Press
9. Ince, D. (1994). ISO 9001 and Software Quality Assurance. London: McGraw-Hill.
10. Software Quality Assurance: From Theory to Implementation. An excellent, up-to-date look at the principles and practice of software quality assurance. It includes a discussion of standards such as ISO 9001. (D. Galin, Addison-Wesley, 2004.)
11. 'Misleading Metrics and Unsound Analyses'. An excellent article by leading metrics researchers that discusses the difficulties of understanding what measurements really mean. (B. Kitchenham, R. Jeffrey and C. Connaughton, IEEE Software, 24 (2), March–April 2007.) http://dx.doi.org/10.1109/MS.2007.49.
12. El-Amam, K. 2001. 'Object-oriented Metrics: A Review of Theory and Practice'. National Research
13. Council of Canada. http://seg.iit.nrc.ca/English/abstracts/NRC44190.html

Veb lokacije :

- http://www.software-engin.com

- http://metcs.bu.edu/~vshtern/673syll.htm
- http://www.uml.org/
- http://www.netobjectives.com/resources/books/design-patterns-explained

Cilj predmeta:

Cilj predmeta je da student bude osposobljen da, na osnovu postavljenih zahteva za razvoj softvera, definiše njegovu arhitekturu i da izvrši njegovo projektovanje pri primeni planom vođenog razvoja objektno-orijentisanog softvera. On treba da kreira detaljnu projektnu dokumentaciju koja je polazna osnova za implementaciju (programiranje) softvera.

Opis predmeta:

Predmet najpre upoznaje student sa osnovama projektovanja softvera, sa arhitekstonskim strukturama, pogledima i stilovima. Izučavaju se stilovi softverskih modula, stilovi povezivanja softverskih komponenata, stilovi alokacije komponenata softvera, kao i hibridni stilovi. Pored navedenog, program predmeta sadrži sledeće nastavne jedinice: Šabloni projektovanja (koji nisu izučavani u okviru predmeta SE201). Strategije i metodi projektovanja softvera. Projektovanje softvera za ponovnu upotrebu. Projektovanje softvera primenom komponenata. Projektovanje distribuiranih softverskih sistema. Projektovanje servisno-orijentisanog softvera. Projektovanje softvera sa mikroservisima. Inženjerstvo softvera u realnom vremenu. Projektovanje pouzdanog softvera.
Analiza i ocena kvaliteta projektnog rešenja softvera.

Ishodi učenja predmeta:

Studenti su osposobljeni za samostalan rad pri projektovanju softverske arhitekture i projektnog rešenja softverskog sistema kroz:

- modelovanje i projektovanje najpoznatijih tipova arhitekture softvera,
- primenu projektnih šablona (patterns),
- makro i mikro okvira (framework) i arhitekturalnih pogleda i stilova,
- dokumentovanje softverske arhitekture i projektnog rešenja,
- analizu i ocenu kvaliteta projektnog rešenja softvera
- upotrebu softverskih komonenata
- planiranje, projektovanje i korišćenje distribuiranih softverskih sistema,
- projektovanje softvera za pružanje usluga,
- projektovanje ugrađenih softvera koji rade u realnm vremenu,
- razumevanje pouzdanosti i dostupnosti softvera, postavljanje pouzdane arhitekture sistema i primenu metrike za merenje pouzdanosti razumevanje procesa upravljanje kvalitetom softvera i merenja atributa kvaliteta softvera.

Deo korpusa znanja (SWEBOK) koji se izučava na predmetu.

Predmet primenjuje korpus znanja (the Body of Knowledge) definisan od strane IEEE Computer Society: SWEBOK V3.0, Guide to the Software Engineering Body of Knowledge, Editores: Pieree Burque i Richard E. Fairley, IEEE Computer Society, 2014 (www.swebok.org).

Kako se predmet bavi projektovanjem softvera, njegove nastavne jedinice realizuju teme navedene u oblasti Software Design i pokrivaju sve navedene jedinice znanja (Knolwedge Units) koje su specificirane u oblasti projektovanja softvera.

Knowledge area	Knowledge unit	Topic	Lekcija	Naziv objekta učenja
SOFTWARE DESIGN	Software Structure and Architecture	Families of Programs and Frameworks	L09	Ponovna upotreba postojećeg softvera
			L09	Radni okviri aplikacijia
			L09	Linije softevrskog proizvoda
			L09	Ponovna upotreba cots softvera
			L09	Zadaci za vežbu
		Architectural Structures and Viewpoints	L13	Inženjerstvo softvera u realnom vremenu
			L13	Projektovanje ugrađenih sistema
			L13	Šabloni projektovanja arhitekture softvera
			L13	Analiza vremena
			L13	Operativni sistemi u realnom vremenu
			L13	Pokazna vežba
			L13	Individualna vežba - zadaci za vežbu
			L02	Softverska arhitektura i atributi kvaliteta
			L02	Sedam pravila za pisanje softverske dokumentacije
			L02	Pregled dokumentacije softverske arhitekture
			L02	Korisnici dokumentacije softverske arhitekture
			L02	Dokumentacija arhitekture i atributi kvaliteta
			L02	Pogledi i izvan pogleda (views and beyond)
			L02	Stilovi i pogledi softverske arhitekture - razlike
			L02	Pogledi na softversku arhitekturu
			L02	Domaći zadatak br.2
			L02	Pokazna vežba-dokumentovanje arhitekture softvera
			L02	Individualna vežba - zadaci
		Architectural Styles	L02	Kategorije stilova softverske arhitekture
			L02	Stilovi (šabloni) softverske arhitekture
			L02	Pokazna vežba-dokumentovanje arhitekture softvera
			L02	Individualna vežba - zadaci
			L03	Stilovi modula softverske arhitekture
			L03	Stil razlaganja softverske arhitekture
			L03	Stil upotrebe softverske arhitekture

			L03	Stil generalizacije softverske arhitekture
			L03	Stil slojeva softverske arhitekture
			L03	Stil aspekta softverske arhitekture
			L03	Model podataka softverske arhitekture
			L03	Pokazna vežba-stilovi softverske arhitekture
			L03	Individualna vežba - zadaci
			L03	Domaći zadatak br.3
			L04	Pogled komponente i konektora
			L04	Različiti stilovi pogleda komponente i konektora
			L04	Stil cevi i filtera
			L04	Stil klijent-server
			L04	Peer-to-peer stil
			L04	Servisno-orijentisani stil
			L04	Stil objavi-pretplati se
			L04	Stil deljenih podataka
			L04	Pokazna vežba - stilovi komponente i konektora
			L04	Individualna vežba - zadaci
			L04	Domaći zadatak br.4
			L05	Stilovi alokacije
			L05	Stil raspoređivanja
			L05	Stil instalacije
			L05	Stil dodeljivanja radnih zadataka
			L05	Ostali stilovi alokacije
			L05	Kombinovanje različitih pogleda
			L05	4+1 kručtenovi pogledi
			L05	Pokazna vežba - stilovi alokacije
			L05	Individualna vežba - zadaci
			L05	Domaći zadatak br.5
			L02	Domaći zadatak br.2
		Design Patterns	L06	Šablon abstraction–occurrence
			L06	Šablon general hierarchy
			L06	Šablon player–role
			L06	Šablon delegation
			L06	Šablon immutable
			L06	Šablon read-only interface
			L06	Šablon proxy
			L06	Objektni klijent-server okvir u projektovanju
			L06	Pokazna vežba-upotreba šablona projektovanja

			L06	Individualna vežba - zadaci
			L06	Domaći zadatak br.6
	Software Design Strategies and Methods	Component-Based Design (CBD)	L10	Softversko inženjerstvo sa komponentima
			L10	Softverska komponente
			L10	Modeli komponenata
			L10	Sastavljanje komponenata
			L10	Pokazna vežba: otakaz i pad rakete ariane 5
			L10	Vežba - zadaci za rad studenata
		Object-Oriented Design	L12	Servisno-orijentisano softversko inženjerstvo
			L12	Servisno-orijentisana arhitektura
			L12	Restful servisi
			L12	Inženjerstvo servisa
			L12	Sastavljanje servisa
			L12	Pokazna vežba
			L12	Vežba za individualni rad
		General Strategies	L07	Metod projektovanja softvera
			L07	Podrška koju metode projektovanja pružaju
			L07	Dizajn virtuelna mašina
			L07	Procesi i strategije projektovanja softvera
			L07	Transformacije u procesu projektovanja softvera
			L07	Pokazna vežba - strategije projektovanja softvera
			L07	Individualna vežba - zadaci
			L07	Domaći zadatak br.7
		Function-Oriented (Structured) Design	L08	Reprezentacija strukturne sistemske analize
			L08	Reprezentacija strukturnog projektovanja
			L08	Ssa/sd proces projektovanja
			L08	Koraci 1 i 2 u transformaciji: ssa
			L08	Korak 3 u transformaciji : analiza transakcije
			L08	Korak 4 u transformaciji : analiza transformacije
			L08	Korak 5 u transformaciji : kompletiranje procesa
			L08	Ssa/sd proces projektovanja - pokazni primer
			L08	Pokazna vežba -metodi projektovanja softvera
			L08	Individualna vežba - zadaci
			L08	Domaći zadatak br.8
		Data-Structure-Centered Design	L08	Jackson strukturno programiranje (jsp)
			L08	Korak 1: crtanje ulazno/izlaznog dijagrama
			L08	Korak 2: kreiranje programskog dijagrama
			L08	Korak 3: nabrajanje operacija i elemenata

			L08	Koraci 4 i 5: konverzija programa u tekst
			L08	Jackson proces razvoja sistema (jsd)
			L08	Jsd proces
			L08	Pokazna vežba -metodi projektovanja softvera
			L08	Individualna vežba - zadaci
			L08	Domaći zadatak br.8
	Key Issues in Software Design	Distribution of Components	L11	Šta su distribuirani sistemi?
			L11	Svojstva distribuiranih sistema
			L11	Modeli interakcije
			L11	Klijent server računarstvo
			L11	Arhitektonski šabloni distribuiranih sistema
			L11	Softver kao servis - saas
			L11	Vežba
		Error and Exception Handling and Fault Tolerance	L14	Greške u softverskom sistemu
			L14	Dostupnost i pouzdanost
			L14	Zahtevi pouzdanosti
			L14	Arhitekture otporne na greške
			L14	Programiranje za pouzdanost
			L14	Merenje pouzdanosti
			L14	Pokazna vežba
			L14	Individualna vežba: zadaci
	Software Design Quality Analysis and Evaluation	Quality Analysis and Evaluation Techniques	L15	Upravljanje kvalitetom softvera
			L15	Kvalitet softvera
			L15	Softverski standardi
			L15	Recenzije i kontrole
			L15	Upravljanje kvalitetom i agilni razvoj
			L15	Pokazna vežba
			L15	Individualna vežba: zadaci
		Measures	L15	Merenje softvera
			L15	Pokazna vežba
			L15	Individualna vežba: zadaci
	Software Design Fundamentals	General Design Concepts	L01	Proces projektovanja
			L01	Projektovanje kao proces rešavanja problema
			L01	Studija slučaja - proces projektovanja selidbe
			L01	Pokazna vežba - projektovanje softvera
			L01	Individualna vežba - zadaci
			L01	Domaći zadatak br.1
		Context of Software Design	L01	Uloge projektnih aktivnosti

			L01	Pokazna vežba - projektovanje softvera
			L01	Individualna vežba - zadaci
			L01	Domaći zadatak br.1
		Software Design Process	L01	Projektovanje u procesu razvoja softvera
			L01	Studija slučaja - bibliotečki sistem
			L01	Pokazna vežba - projektovanje softvera
			L01	Individualna vežba - zadaci
			L01	Domaći zadatak br.1
		Software Design Principles	L01	Proces projektovanja softvera
			L01	Pokazna vežba - projektovanje softvera
			L01	Individualna vežba - zadaci
			L01	Domaći zadatak br.1
SOFTWARE ENGINEERING PROCESS	Process Implementation and Change	Software Process Management Cycle	L09	Studija slučaja: udaljene meteorološke stanice
			L09	Zadaci za vežbu

Način ocenjivanja:

Student se ocenjuje u toku celog semestra. Ocenjuju se njegovi domaći zadaci, rad na projektu, testovi i aktivnost u nastavi. Na kraju u ispitnom roku, ocenjuje se i pismeni ispit.Ocene se daju u poenima. Maksimalni broj poena je 100 (uključujući i pismeni ispit). Na pismenom ispitu student može dobiti do 30 poena, a aktivnosti u toku semestra (predispitne obaveze) mogu mu doneti do 70 poena, po sledećoj strukturi:

- **Domaći zadaci – 22,5 poena:** Posle izučavanja određene nastavne jedinice, odnosno, posle vežbi u okviru jednog predavanja (jedna nedelja) predviđeno je da studenti dobiju zadatak koji treba samostalno da reše. Svaki student dobija različit zadatak i kao preduslov za izlazak na ispit u obavezi je da uradi i pošalje sve zadatke. Predviđeno je ukupno **15 zadataka**, a svaki uspešno rešen zadatak predat u zadatom roku obezbeđuje studentu **1,5 poena**, pod uslovom uspešne odbrane urađenog zadatka. Za studente tradicionalne (u Beogradu) ili hibridne nastave (Centar u Nišu) rok za predaju zadataka je **7 dana nakon izdavanja**, a **posle tog roka umanjuje ostvaren broj poena za 50%.** Krajnji rok za predaju domaćeg zadatka i za studente tradicionalne nastave i za studente onlajn nastave je 10 (deset) dana pre ispitnog roka u kome student polaže ispit.

  Projektovano vreme potrebno za izradu zadataka iznosi:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h	0,5h

- **Projekat - 25 poena:** Svaki student dobija jedan projekt od asistenta kojim treba da pokaže svoju spremnost da primeni stečena znanja u konkretnom primeru. Student bira jednu od ponuđenih tema za projektne zadatke. Procenjeno vreme rada na projektu je 60 sati. Student treba svake nedelje tokom semestra da uradi deo projektnog zadatka prema dobijemnom planu i da ga pošalje asistentu na

uvid, u roku od 7 dana od održane nastave za tu nedelju. **Ukoliko student ne radi projektni zadatak u skladu sa ovim planom, umanjuje mu se poeni za projekat do 30%.** Ako student prilikom ocenjivanja projekta ne dobije najmanje 50% predviđenih poena (12,5 poena), on mora da ga doradi. U suprotnom, dobija 0 poena, tj. ne može dobiti manje od 50% predviđenih poena. **Student koji ne dobije više od 50% predviđenih poena ne može izaći na ispit.** Za studente tradicionalne nastave rok za predaju projektnog zadatka je **15 dana od prestanka nastave u jesenjem semestru, a posle tog roka dobija 0 poena za zalaganje** Krajnji rok za predaju projekta i za studente tradicionalne nastave i za studente onlajn nastave je 10 (deset) dana pre ispitnog roka u kome student polaže ispit.

- **Testovi – 10 poena –** U skladu sa Planom nastave (videti donju tabelu) predviđeno je da student u naznačenim nedeljama treba da uradi ukupno 5 testova, čime može da obezbedi najviše 10 poena, jer svaki test može obezbediti najviše 2 poena. Student mora test da uradi u nedelji datim u Planu nastave (donja tabela), tj. u roku od 7 dana po predavanju. **Za svaki test urađen van datog termina, poen dobijene testom se umanjuju za 50%**, s tim što je krajnji rok za polaganje testa – 10 dana pre ispita. Studenti onlajn nastave testove moraju uraditi najkasnije 10 dana pre ispita ispita (bez umanjenja broja poena).

- **Testovi za samotestiranje u okviru lekcija:** Svaka lekcija ima dva testa za samotestiranje studenta. Student je položio test ako je tačno odgovorio na 5 od 6 postavljenih pitanja (koji se slučano programski generišu iz baze pitanja). Ukoliko u prvom pokušaju student to ne ostvari, može preko mape uma koja je priložena uz test, da pretražuje nastavni materijal da bi našao tačne odgovore, i da nastavi rešavanje otvorenog testa. To može da ponavlja bez ograničenja, sve dok ne dobije bar 5 od 6 tačnih odgovora, tj. sve dok test bude uspešno urađen. Ove godine, neuspešan test ne ograničava da student nastavi čitanje preostalog nastavnog materijala lekcije, ali time gubi mogućnost da dobije 10 poena na zalaganje, koji se dodeljuju studentu koji uspešno uradi testove u svim lekcijama, i koji je učestvovao na forumima svih lekcija, ali i koji nema više od 5 izostanka sa predavanja i vežbi tokom semstra (ako je student klasične nastave). Da bi to ostvario, student mora da uspešno položi ove testove u lekcijama u roku od 7 dana od dana održavanja predavanja po rasporedu nastave. Cilj je da se stimuliše stalni rad studenta na učenju tokom semestra (a ne kampanjsko učenje pred ispit) i njegova bolja priprema za nastavu u okviru sledeće lekcije.
- **Forum:** Cilj foruma je da podstiće proaktivni rad studenata i njihovu saradnju tokom nastave na predmetu. Svaki forum nudi studentima dve teme za njihovo angažovanj. Mogu izabrati jednu od dve teme. **Prva tema foruma** postavlja problem na koji očekuje rešenje ili odgovor studenata. Mogući su različiti tačni odgovri. Student koji da rešenje problema, koje je tačno, a razlikuje se od drugih, prethodnodatih, a isto tačnih rešenja, dobija + za aktivnost a ovom forumu**. Druga tema foruma** je otvorena, jer omogućava da student postavi neki problem ili pitanje, vezano za lekciju, bilo da se problem odnosi na deo predavanja, bilo na deo vežbi, i da potraži pomoć svojih kolega. Za postavljanje problema na forum, studentu se priznaje aktivnost na forumu, tj. dobija +. Isto se priznaje aktivnost na forumu i studentu koji prvi reaguje na ovaj poziv studenta za pomoć, pod uslovom da je njegov savet uspešno rešio problem koji je student postavio. Ovo znači, da forum može imati i onoliko postavljenih problema, koliko ima studenata, jer svaki student može, ako želi da postavi svoj problem ili pitanje za drugi temu foruma. Student koji nije dao na vreme odgovor na prvu temu, može da koristi drugu temu, da stekne + na forumu lekcije. Student koji ima evidentiranu aktivnost (tj. +) na opisan način na forumu svake lekcije, i ako ima uspešno urađene testove u okviru svih lekcija, dobija 10 poena na zalaganje, pod uslovom da nema više od 5 izostanka sa predavanja i vežbi tokom semestra (ako je student klasičnih studija). Detaljniji uslovi su ispod.
- **Zalaganje studenta u nastavi u toku semestra – 12,5 poena:** Poeni za zalaganje stimulišu kontinualni rad studenta tokom semestra**.** Aktivnost u nastavi (određivanje poena za zalaganje) se ocenjuje:

- Kod studenata tradicionalnog oblika nastave primenom sledećih kriterijuma:
    1. Dolaza pripremljen za nastavu, čitanjem nastavnih materijala pre predavanja i vežbi
    2. Redovnost u pohađanju nastave. **Student tradicionalne koji nije, iz bilo kog razloga, ima više od 5 izostanaka (bez obzira na razlog izostanka), dobija automatski 0 poena na zalaganje.** Ovo ograničenje ne važi samo u specijalnim slučajevima kada je student zbog duže bolesti sa ili bez bolničkog tretmana, ali uz potvrdu nadležnog lekara, bio prinuđen da odsustvuje iz nastave više od 5 nedelja.
    3. Redovnost i kvalitet ispunjenja predispitnih obaveza. **Student koji ne uradi sve svoje predispitne obaveze najkasnije 15 dana po završetku nastave na predmetu, dobija nula (0) poena na zalaganje**
    4. **Student koji uradi sa uspehom sve testove u okviru lekcija i uspešno reši problem koji je tema foruma u svim lekcijama dobija maksimalnih 12,5 poena za zalaganje**, pod uslovom da je ispunio uslove pod 2. I 3. Student mora da reši testove u lekcijama i problem u forumu lekcije u roku od 7 dana od dana održavanja predavanja.
    5. Učestvovanja u diskusijama na vežbama i predavanjima, kao i učestalost konsutacija tokom izrade projekta,
- Kod onlajn studenata (studije preko Interneta):
    1. Ranije (u odnosu na ispit) ispunjenje predispitnih obaveza
    2. Konsultacije tokom semestra u vezi rada na predispitnim obavezama
    3. Učešće na forumu predmeta (LAMS)

- **Pismeni ispit – 30 poena:** Pravo polaganja ispita ima student koji je stekao najmanje 35 poena realizacijom svojih predispitnih obaveza, koji je pokušao da uradi sve predviđene testove i koji je dobio najmanje 50% previđenih poena za projekat. Student prijavljuje ispit na kraju semestra, odnosno pre svakog ispitnog roka, u roku koji objavi uprava univerziteta. **Prijavljeni ispit se ne može otkazati**. Taksa za polaganje ispita se naplaćuje i studentu koji je prijavio polaganje ispita, a nije se pojavio na ispitu..

Pravila vezana za predmet:

Nastava: Student na tradicionalnoj nastavi mora da dođe pripremljen na svako predavanje, odn. vežbu time što je unapred pročitao pripremljeno gradivo za dato predavanje preko LAMS-a i pripremio se za diskusiju na predavanju, odn. za rešavanje zadataka na vežbama, kao i postavljanje pitanja u vezi pojedinih nejasnoća sa kojima se suočio tokom proučavanja gradiva.

Konsultacije: Svim studentima nastavnik i asistent su na raspolaganju tokom cele školske godine za konsultacije u prostorijama Univerziteta, na Skype-u u terminu prema dogovoru sa studentima, primenom pričaonice (čata) ili foruma predmeta koje nudi LAMS, ili preko mejlova.

Mentorski rad: Svaki student dobija svog mentora koji mu pomaže tokom studiranja na fakultetu. Svaki student dobija obaveštenje ko mu je mentor i ima mogućnost da kontaktira mentora po bilo kom pitanju koje se tiče njegovog uspešnog rada na fakultetu. Svaki mentor je u obavezi da kontaktira studente za koje je odgovoran kako bi utvrdio eventualne potrebe ili probleme sa kojima se student suočava i kako bi mu pomogao u rešavanju istih.

Domaći zadaci: U toku semestra se rade domaći zadaci. Svaki student dobija od asistenta poseban domaći zadatak. Po uradjenom zadatku, student ga dostavlja na email asistentu u skladu sa oznakom fajla koja je data u postavci zadatka. Asistent može vratiti studentu predat domaći zadatak na doradu, ako smatra da ga nije dobro uradio, sa napomenom razloga zbog kojih mu vraća zadatak Svi zadaci moraju biti rešeni, predati i prihvaćeni od strane asistenta do kraja semestra.. **Asistent zakazuje odbranu predatih domaćih zadataka** studentima tradicionalne nastave. Student na odbrani treba da dokaže samostalnost u rešavanju domaćeg zadatka, pokaže da razume način rešavanja zadatka, da ga izloži i odgovori na dodatna pitanja asistenta. Odbrana domaćeg zadatka onlajn studenata se može obaviti preko Skype-a, ili preko mejla. O načinu odbrane, asistent obaveštava studenta mejlom. Zadaci se ocenjuju poenima na osnovu prikazanog razumevanja problema/teme pri odbrani zadatka, urednosti, sistematičnosti i nivoa detaljnosti prikaza i obrade zadatka. Ukolko se pri odbrani utvrdi da student nije samostalno radio domaći zadatak, student dobija 0 poena za zadatak.

Testovi: Student mora da samostalno uradi svaki od predviđenih testova. Student tradicionalne nastave test mora da uradi u roku od 7 dana od dana predavanja za koji je previđen test (dat u Planu nastave koji je dat dole). Test urađen posle tog roka dobija umanjenje poene za 50%. Student onlajn nastave testove mora d auradi najdalje 10 dana pre ispita. To je poslenji rok i za studente tradicionalne nastave.

Projekat: Projekat je najvažnija i najznačajnija predispitna obaveza studenta. Na početku semestra objavljuje se spisak tema projektnih zadataka. Student može da izabere jednu od ponuđenih tema i da u dogovoru sa asistentom, dobije sve neophodne dodatne informacije i instrukcije za rešavanje izabranog projektnog zadatka. *Svaki student dobija jedinstven i specifični projektni zadatak* njkasnije do 3. nedelje nastave. Uz poseban zahtev i obrazloženje, asistent može studentu odobriti i projektni zadatak na temu koju predloži student, ali asistent je dužan da definiše ostale neophodne podatke i informacije za rad, tj. određuje kontekst u slučaju u kome se rešava prihvaćen projektni zadatak. Projektni zadatak definiše sve aktivnosti koje student treba da uradi da bi došao da krajnjeg rezultata koji se traži. Student je dužan da u Izveštaju u urađenom projektnom zadatku, prvo navede (citira) projektni zadatak, a onda da u svakom poglavlju koje se odnosi na jednu aktivnost projekta, svoj rad i rezultata rada na toj aktivnosti. Pri tome, poštuje redosled aktivnosti definisane projektnom zadatkom. Na kraju projekta, izlaže konačan rezultat i daje zaključak izveštaja. Tekst Izveštaja se piše u trećem licu jednine i mora da bude tehnički, jezički i profesionalno dobro urađen, usklađen za uzorkom dokumenta koji definiše asistent.

- **Rad na projektu:** Student, po pravilu, kontinualno radi svoj projekat tokom semestra, u skladu sa stečenim novim i potrebnim znanjima. Preporučuje se studentu da u toku rada, konsultuje asistenta, tj. da mu da na uvid urađeni deo projekta, kako bi bio siguran da je dobro shvatio svoj zadatak i primenio pravilan način rešavanja projektnog zadatka. Projekat se ne radi na kraju semestra ili kasnije. On se radi paralelno sa savlađivanjem gradiva, kako bi olakšao studentu savlađivanje novog gradiva i time pomogao u pripremi za savlađivanje narednog gradiva na predmetu.
- **Predaja projekta:** Zahtevan rok za predaju *Izveštaja o urađenom projektnom zadatku* je do kraja 14. nedelje nastave za studente tradicionalne studente nastave, a za studente onliajn nastave – 10 dana pre ispita. Studentima tradicionalne koji predaju Izveštaj o urađenom projektnom zadatku 15 i više dana po završenoj nastavi na predmetu, **umanjuje se broj ostvarenih poena za 30%.** Krajnji rok za predaju projekta, za sve studente, je 10 dana pre ispitnog roka u kome žele da polažu ispit. Studentima onlajn nastave (koji ne prate nastavu u Centru u Nišu) se ne umanjuje broj poena, jer su oni, po pravilu zaposleni, ili sprečeni da redovno studiraju (primenom tradicionalog i hibridng oblika nastave).

- **Odbrana i ocenjivanje projekta:** Ako student prilikom ocenjivanja projekta ne dobije najmanje 50% predviđenih poena (15 poena), on mora da ga doradi. U suprotnom, dobija 0 poena. Student koji ne dobije više od 50% predviđenih poena ne može izaći na ispit. Student je dužan da projekat odbrani kod asistenta. Odbrana projekata studenata tradicionalne nastave i hibridne nastave (u Centru u Nišu) vrši se u 15. nedelji jesenjeg semestra za vreme vežbi, a ako je potrebno, i van vežbi – na dodatnim časovima. Van ovog termina, student može da brani projekat u posebnom terminu koji odredi asistent pred svaki ispitni rok. Cilj odbrane projekta je da asistent ustanovi da li je student samostalno radio projekat i u tom cilju odbrana projekta podrazumeva da student demonstrira znanje koje je iskoristio za izradu projekta. Odbrana projekta za studente internet nastave obavlja se preko Skype-a, a ako je za studenta prihvatljivo, u prostorijama univerziteta.

Na LAMS sistemu se nalazi uputsvo za izradu projekta.

Ispit: Ispit se polaže u okviru ispitnih rokova u računarskoj učionici fakulteta. Za vreme ispita nije dozvoljeno korišćenje mobilnih telefona ili drugih komunikacionih uređaja, kao ni Interneta. Ispit ima teorijski i praktični deo. Teorijski deo čine pitanja na koje student treba da pruži odgovore u pisanom obliku. Praktični deo čine zadaci koje student treba da reši uz primenu softverskog alata navedenog u ispitnom zadatku. Teorijski deo obezbeđuje 10 poena, a praktični deo 20 poena. *Student je položio ispit ako položi oba dela, tj. da ima najmanje 5 poena na teorijskom delu, i najmanje 10 poena na praktičnom delu ispita.* Student koji prepisuje na ispitu, udaljava se sa ispita, i **kažnjava se sa 10 predispitnih poena**. Protiv studenta za koga se utvrdi da je drugom kolegi pomagao u toku ispita, biće pokrenut disciplinski postupak i biće određena adekvatna mera. Ukoliko student pokaže znatno slabije rezultate na pismenom ispitu nego što je pokazao radom na predispitnim obavezama, asistent/nastavnik može da zatraži od studenta da **naknadno brani svoj rad na domaćim zadacima i projektu.** To se može primeniti ako je student na ispitu dobio manje od 20 poena, a na predispitnim obavezama je dobio više od 50 poena. Ukoliko nastavnik posumnja da je student uradio pismeni ispit zadatak prepisujući, može da sa njim održi i **usmeni deo ispita**, kako bi proverio njegovo znanje. Ukoliko student pokaže vidno slabije znanje nego što to pokazuje njegov pismeni deo ispita, nastavnik daje ocenu na ispitu na osnovu ocene njegovog usmenog dela ispita.

Za studente koji ne polože ispit do kraja školske 2021/22. godine, tj. do 30.9.2022. godine, a nisu položili ispit iz SE311, važe sledeća pravila:

- Studenti tradicionalne koji nisu ostvarili uslov za upis naredne godine studija, imaju obavezu ponovnog slušanja nastave i ponovnog rada na svim predispitnim obavezama, jer im se poništavaju svi poeni stečeni u prethodnoj školskoj godini. Ispit mogu da polažu u januarsko-februarskom ispitnom roku ili kasnije, ako su ostvarili uslov za izlazak na ispit, tj. uradili nove predispitne obaveze i stekli najmanje 35 poena.
- Studenti tradicionalne nastave koji su ostvarili uslov za upis naredne godine studija mogu preneti predmet u narednu godinu, bez obaveze ponovnog rada na predispitnim obavezama, ukoliko su prethodno ostvarili uslov za izlazak na ispit. Ako student nije ostvario dovoljan broj poena za izlazak na ispit, radiće predispitne obaveze kao onlajn student (odnosno moćiće da svoje predispitne obaveze pošalje najkasnije 10 dana pre ispitnog roka u kojem student želi da polaže ispit, ali ispit moraju položiti najkasnije u oktobarskom roku 2021. godine. Ako i *posle tog ispitnog roka ne polože ispit, poništavaju im se poeni na svim do tada urađenim predispitnim obavezama, i moraju da rade nove predispitne obaveze i pripremaju ispit po Planu i programu koji važi od 01. oktobra 2021. godine.*

- Studenti onlajn nastave, rade predispitne obaveze i polažu ispit po ovom Planu i programu sve do oktobarskog roka 2022. godine. Predispitne obaveze mogu poslati najkasnije 10 dana pre ispitnog roka u kojem student želi da prijavi ispit. Ako i *posle tog ispitnog roka ne polože ispit, poništavaju im se poeni na svim do tada urađenim predispitnim obavezama, i moraju da rade nove predispitne obaveze po Planu i programu koji važi od 01. oktobra 2022. godine.*
- Studenti koji su odslušali četvrtu godinu i stekli status apsolventa rade predispitne obaveze i polažu ispit po ovom Planu i programu sve do oktobarskog roka 2022. godine. Predispitne obaveze mogu poslati najkasnije 10 dana pre ispitnog roka u kojem student želi da prijavi ispit. Ako i *posle tog ispitnog roka ne polože ispit, poništavaju im se poeni na svim do tada urađenim predispitnim obavezama, i moraju da rade nove predispitne obaveze po Planu i programu koji važi od 01. oktobra 2022. godine.*

Ukupna ocena na predmetu se dobija sabiranjem poena dobijenih radom na predispitnim obavezama (maksimalno do 70) i poena sa ispita (maksimalno do 30) i to na sledeći način (definisano Zakonom o visokom obrazovanju):

- do 50 poena, ocena 5
- od 51 do 60 poena, ocena 6
- od 61 do 70 poena, ocena 7
- od 71 do 80 poena, ocena 8
- od 81 do 90 poena, ocena 9
- od 91 do 100 poena, ocena 10.

PLAN NASTAVE

Nedelja	Čas	Nastavna jedinica	Tematske jedinice (objekti i podobjekti učenja su boldirani)	Ishodi učenja – znanja ili veštine koje student treba da dobije	Vežbe (1 čas pokazne vežbe i 3 časa individualnih vežbi)
1		**L01 - Osnove projektovanja softvera**	1. **Proces projektovanja:** Rešavanje problema u procesu projektovanja. Faze unutar procesa projektovanja. Pokazni primer: Zahtevanje rešenja. Pokazni primer: Model rešenja. Pokazni primer: Evaluacija modela shodno početnim zahtevima i elaboriranje modela. 2. **Uloge projektnih aktivnosti:** Identifikovanje plana projektovanja. Kanali komunikacije projektanta softvera. Planovi projektovanja softvera. Projektni model. Pokazni primer: Projektni model. 3. **Projektovanje kao proces rešavanja problema:** Studija slučaja - Proces projektovanja selidbe. 4. **Studija slučaja - Proces projektovanja selidbe:** Projektovanje procesa selidbe i planiranje prostora. Prilagođavanje procesa projektovanja selidbe. 5. **Projektovanje u procesu razvoja softvera:** Studija slučaja - Bibliotečki sistem. 6. **Studija slučaja - Bibliotečki sistem:** Primena modela vodopada u procesu projektovanja bibliotečkog sistema. 7. **Proces projektovanja softvera:** Softver i isporuka softvera. Modeli razvoja softvera. Inicijalni model projektovanja softvera. Primena različitih koncepata i ograničenja u projektovanju softvera. Prenošenje znanja o projektovanju. Metoda prepoznavanja, restrukturiranja i proceduralnog znanja. Odluke o dokumentovanju procesa projektovanja. Projektovanje unutar projektnog tima. How to Build A Project Team (video). Pokazni primer: Proces restrukturiranja softvera po fazama.	Razumevanje procesa projektovanja softvera; Razumevanje i primenu procesa projektovanja softvera i projektnih aktivnosti; Upoznavanje sa procesom projektovanja softvera, modelom rešenja i kreiranjem incijalnog modela projektovanja softvera; Definisanje i razumevanje dugoročnog plana izmena softvera i procesa restrukturiranja softvera;	Pokazna vežba: Zahtevanje rešenja: Softver za vođenje knjigovodstva. Model rešenja softvera za vođenje knjigovodstva. Projektni model softvera za vođenje knjigovodstva u formi konceptualnog dijagrama. Projektni model softvera za vođenje knjigovodstva u formi dijagrama stanja. Zahtevanje rešenja - sistem za mentorisanje - mentor. Zahtevanje rešenja - sistem za mentorisanje - mentij i učenik. Zahtevanje rešenja - sistem za mentorisanje - saradnik. Model rešenja softvera - sistem za mentorisanje. Projektni model - sistem za mentorisanje - dijagrama stanja. Individualna vežba:

					Zadatak za samostalni rad - proces projektovanja. Zadatak za samostalni rad - uloge projektnih aktivnosti. Zadatak za samostalni rad - projektovanje u procesu razvoja softvera. Zadatak za samostalni rad - proces projektovanja softvera. Zadaci za samostalan rad. Zadaci za diskusiju na vežbi.
2		**L02 - Arhitektonske strukture, pogledi i stilovi**	1. **Softverska arhitektura i atributi kvaliteta:** Arhitektura i atributi kvaliteta. Rešavanje problema projektovanjem i dokumentovanjem softverske arhitekture. What is Software Architecture? (video). 2. **Sedam pravila za pisanje softverske dokumentacije:** Pravilo 1: Pisati dokumentaciju iz tačke gledišta čitaoca. Pravilo 2: Izbegavajte nepotrebno ponavljanje. Pravilo 3: Izbegavajte dvosmislenost. Pravilo 4: Koristite standardnu organizaciju (šablon). Pravilo 5: Snimiti obrazloženje. Pravilo 6: Zadržati dokumentaciju trenutnom, ali ne previše trenutnom. Pravilo 7: Pregledati dokumentaciju i proveriti svrsishodnost. 3. **Pregled dokumentacije softverske arhitekture:** Korisnici dokumentacije softverske arhitekture. Dokumentacija arhitekture i atributi kvaliteta. Pogledi i izvan pogleda (Views and Beyond). 4. **Korisnici dokumentacije softverske arhitekture:** Dokumentacija softverske arhitekture. Korisnici softverske dokumentacije. 5. **Dokumentacija arhitekture i atributi kvaliteta:** Pet načina za prikaz atributa kvaliteta. Isplativnost dokumentacije. 6. **Pogledi i izvan pogleda (Views and Beyond):** Generalni opis metode pogleda i izvan pogleda (Views and Beyond).	Definisanje i razumevanje različite softverske arhitekture; Pregled dokumentacije softverske arhitekture, definisanje krajnjih korisnika softverske arhitekture i atributa kvaliteta; Razumevanje pogleda na softversku arhitekturu, primenu stilova softverske arhitekture i primenu skupa tehnika u cilju detaljnog predstavljanja svakog	Pokazna vežba: Primer popunjavanja dokumenta softverske arhitekture - Sekcija 1 i 2 - softver Personalni trener. Primer popunjavanja dokumenta softverske arhitekture - Sekcija 2D - softver Personalni trener. Prikaz softverske arhitekture kroz stil cevi i filteri - softver Personalni trener. Primena različitih stilova na softversku arhitekturu - softver Personalni trener.

			Metoda pogleda i izvan pogleda u agilnom okruženju. Arhitektura koja se menja brže nego što se dokumentuje. Pokazni primer: Šablon dokumenta za opis arhitekture iz jednog pogleda. 7. **Stilovi i pogledi softverske arhitekture - razlike:** Razlike između stilova softverske arhitekture i pogleda na softversku arhitekturu. 8. **Pogledi na softversku arhitekturu:** Opis pogleda na softversku arhitekturu. Kombinovanje različitih pogleda na softversku arhitekturu. Pokazni primer: Šablon dokumenta za opis arhitekture iz više različitih pogleda. 9. **Kategorije stilova softverske arhitekture:** Tri kategorije stilova softverske arhitekture. Opis tri kategorije stilova softverske arhitekture. Odabir elemenata i relacija za dokumentaciju. Notacije pogleda softverske arhitekture. 10. **Stilovi (šabloni) softverske arhitekture:** Stilovi softverske arhitekture. Primena stilova softverske arhitekture. Kombinovanje različitih stilova softverske arhitekture. Pokazni primer: Stil cevi i filteri. Pokazni primer: primena različitih stilova na softversku arhitekturu. Types of Architectural Styles (video).	dela softverske arhitekture; Razumevanje i primena sedam pravila za pisanje softverske arhitekture;	Individualna vežba: Zadatak za samostalan rad. Zadaci za samostalan rad - sedam pravila za pisanje. Zadaci za samostalan rad - pogledi i izvan pogleda. Zadaci za samostalan rad - pogledi na softversku arhitekturu. Zadaci za samostalan rad - kategorije stilova. Zadaci za samostalan rad - stilovi softverske arhitekture. Zadaci za diskusiju na vežbi.
3		**L03 - Stilovi softverskih modula**	1. **Stilovi modula softverske arhitekture:** Elementi, relacije i svojstva stilova modula. Svojstva stilova modula. Koji stilovi modula se koriste. Notacije za stilove softverskih modula. Architectural Styles (video). 2. **Stil razlaganja softverske arhitekture:** Pregled stila razlaganja (dekompozicija). Elementi, relacije i svojstva stila razlaganja. Upotreba stila razlaganja. Notacija stila razlaganja. Pokazni primer: Stil razlaganja (dekompozicije). Pokazni primer: Stil razlaganja (dekompozicija) - dijagrami arhitekture - najviši nivo stila razlaganja. Pokazni primer: Stil razlaganja (dekompozicija) - dijagrami arhitekture - razlaganje Java modula na serverskoj strani. 3. **Stil upotrebe softverske arhitekture:** Pregled stila upotrebe. Elementi, relacije i svojstva stila upotrebe. Upotreba stila upotrebe. Notacija stila upotrebe. Pokazni primer: Stil upotrebe. 4. **Stil generalizacije softverske arhitekture:** Pregled stila generalizacije. Elementi, relacije i svojstva stila generalizacije. Upotreba stila generalizacije. Pokazni primer: Stil	Razumevanje i primena stilova softverskih modula, pregled elemenata, relacija i svojstva pogleda modula; Kombinovanje pogleda modula sa drugim pogledima na softversku arhitekturu, notacije unutar pogleda modula; Razumevanje i primena stila razlaganja, pregled elemenata, relacija i	Pokazna vežba: Primena stila razlaganja na softversku arhitekturu aplikacije Personalni trener. Primena stila upotrebe na softversku arhitekturu aplikacije Personalni trener. Primena stila slojeva na softversku arhitekturu aplikacije Personalni trener. Primena stila aspekta na softversku arhitekturu aplikacije Personalni

			generalizacije. Generalizations (video). 5. **Stil slojeva softverske arhitekture:** Pregled stila slojeva. Elementi, relacije i svojstva stila slojeva. Notacija stila slojeva. Pokazni primer: Stil slojeva. 6. **Stil aspekta softverske arhitekture:** Pregled stila aspekta. Elementi, relacije i svojstva stila aspekta. Notacija stila aspekta. Pokazni primer: Stil aspekta. Pokazni primer: Stil aspekta - nastavak. 7. **Model podataka softverske arhitekture:** Pregled modela podataka. Elementi, relacije i svojstva modela podataka. Pokazni primer: Model podataka.	svojstva stila razlaganja; Upotreba stila razlaganja na konkretnom primeru i kombinovanje sa drugim stilovima softverske arhitekture; Razumevanje i primena stila upotrebe, stila generalizacije, stila slojeva i stila aspekta; Kombinovanje navedenih stilova softverske arhitekture sa drugim stilovima softverske arhitekture, razumevanje notacija prikazivanja različitih stilova i pregled elemenata, relacija i svojstava; Predstavljanje softverske arhitekture kroz model podataka poštujući definisanu notaciju, elemente i relacije modela;	trener. Primena stila razlaganja na softversku arhitekturu sistema za mentorisanje. Primena stila stila slojeva na softversku arhitekturu sistema za mentorisanje. Individualna vežba: Zadatak za samostalan rad. Zadaci za samostalan rad - stil razlaganja. Zadaci za samostalan rad - stil upotrebe. Zadaci za samostalan rad - stil generalizacije. Zadaci za samostalan rad - stil slojeva. Zadaci za samostalan rad - stil aspekta. Zadaci za samostalan rad - model podataka. Zadaci za diskusiju na vežbi.
4		**L04 - Stilovi povezivanja softverskih komponenata**	1. **Pogled komponente i konektora:** Upotreba pogleda komponente i konektora. Grafičko predstavljanje pogleda komponente i konektora. Elementi, relacije i svojstva pogleda komponente i konektora. Element komponenta u pogledu komponente i konektora. Element konektor u pogledu komponente i konektora. Relacije u pogledu komponente i konektora. Svojstva u pogledu komponente i konektora. Upotreba stila komponente i konektora. Notacije pogleda	Razumevanje stilova povezivanja softverskih komponenata (iz pogleda komponente i konektora); Pregled različitih stilova pogleda	Pokazna vežba: Prikaz softverske arhitekture softvera za vođenje knjigovodstva upotrebom stila komponenata i konektora.

			komponente i konektora - prikaz komponenti. Notacije pogleda komponente i konektora - prikaz portova i konektora. Notacije pogleda komponente i konektora - prikaz konektora kroz UML jezik. Relacije pogleda komponente i konektora sa drugim pogledima na softversku arhitekturu. Uporedni prikaz komponenta i konektor pogleda i pogleda modula softverske arhitekture. 2. **Različiti stilovi pogleda komponente i konektora:** Stilovi toka podataka softverske arhitekture. Stil poziv - povratak softverske arhitekture. Stil peer-to-peer softverske arhitekture. Stil softverske arhitekture zasnovan na događaju. 3. **Stil cevi i filtera:** Pregled stila cevi i filtera. Elementi, relacije i svojstva stila cevi i filtera. Pokazni primer - upotreba stila cevi i filtera. 4. **Stil klijent-server:** Pregled stila poziv-povratak. Elementi, relacije i svojstva stila klijent-server. Pokazni primer - upotreba stila klijent-server. Client-Server Architecture - Software Engineering (video). 5. **Peer-to-peer stil:** Elementi, relacije i svojstva stila peer-to-peer. Pokazni primer - upotreba stila peer-to-peer. 6. **Servisno-orijentisani stil:** Pregled servisno-orijentisanog stila softverske arhitekture. Elementi, relacije i svojstva servisno-orijentisanog stila. Pokazni primer - upotreba servisno-orijentisanog stila. Service-Oriented Architecture (video). 7. **Stil objavi-pretplati se:** Pregled stila objavi-pretplati se. Elementi, relacije i svojstva stila objavi-pretplati se. Pokazni primer - upotreba stila objavi-pretplati se. 8. **Stil deljenih podataka:** Elementi, relacije i svojstva stila deljenih podataka. Pokazni primer - upotreba stila deljenih podataka.	komponenata i konektora; Razumevanje stilova: cevi i filtera, klijent-server, peer-to-peer, servisno-orijentisanog stila, stila-objavi-pretplati se i stila deljenih podataka; Primena navedenih stilova na konkretnim primerima softverske arhitekture i razumevanje načina upotrebe u konkretnim situacijama; Razumevanje razlike u predstavljanju softverske arhitekture različitim stilovima pogleda komponente i konektora;	Prikaz softverske arhitekture aplikacije Personalni trener kroz klijent-server stil. Prikaz softverske arhitekture aplikacije Personalni trener kroz stil deljenih podataka. Prikaz softverske arhitekture aplikacije Personalni trener kroz servisno-orijentisani stil. Prikaz softverske arhitekture sistema za mentorisanje kroz servisno-orijentisani stil. Individualna vežba: Zadatak za samostalan rad. Zadaci za samostalan rad - stil cevi i filtera. Zadaci za samostalan rad - stil klijent-server. Zadaci za samostalan rad - peer to peer stil. Zadaci za samostalan rad - servisno-orijentisani stil. Zadaci za samostalan rad - stil objavi-pretplati se. Zadaci za samostalan rad - stil deljenih podataka. Zadaci za diskusiju na vežbi.

5		**L05 - Stilovi alokacije i hibridni stilovi**	1. **Stilovi alokacije:** Tri različita stila alokacije. Elementi, relacije i svojstva stilova alokacije. 2. **Stil raspoređivanja:** Pregled stila raspoređivanja. Elementi stila raspoređivanja. Relacije stila raspoređivanja. Svojstva stila raspoređivanja. Kada se koristi stil raspoređivanja. Pokazni primer: stil raspoređivanja - neformalna notacija. Pokazni primer: stil raspoređivanja - neformalna notacija - slika. Pokazni primer: stil raspoređivanja - UML notacija. Deployment Architectural Perspective (video). 3. **Stil instalacije:** Pregled stila instalacije. Elementi, relacije i svojstva stila instalacije. Kada se koristi stil instalacije. Pokazni primer: stil instalacije - neformalna notacija. Pokazni primer: stil instalacije - UML notacija. 4. **Stil dodeljivanja radnih zadataka:** Pregled stila dodeljivanja radnih zadataka. Elementi, relacije i svojstva stila dodeljivanja radnih zadataka. Pokazni primer: stil dodeljivanja radnih zadataka - neformalna notacija. 5. **Ostali stilovi alokacije:** Upotreba drugih stilova alokacije. Specijalizacija stilova alokacije. 6. **Kombinovanje različitih pogleda:** Kombinovanje pogleda na softversku arhitekturu. Tipovi asocijacija između različitih pogleda. Kombinovani pogledi. Kada kombinovati poglede na softversku arhitekturu. Pokazni primer: Kombinovanje pogleda na softversku arhitekturu. 7. **4+1 Kručtenovi pogledi:** Pregled 4+1 Kručtenovih pogleda. Primena 4+1 Kručtenovih pogleda. 4+1 view into software architeture (video).	Razumevanje stilova alokacije i hibridnih stilova, primena stilova alokacije (stila raspoređivanja, stila instalacije i stila dodeljivanja radnih zadataka); Razumevanje specijalizacije stilova alokacije i primena drugih stilova alokacije koji se mogu koristiti za prikazivanje softverske arhitekture; Razumevanje prednosti kombinovanja različitih stilova u cilju detaljnog predstavljanja softverske arhitekture; Razumevanje i primena 4+1 Kručtenovih pogleda;	Pokazna vežba: Stil raspoređivanja - UML formalna notacija - primer 1. Stil raspoređivanja - UML formalna notacija - primer 2. Stil instalacije - neformalna notacija. Stil dodeljivanja radnih zadataka - neformalna notacija. Stil raspoređivanja - sistem za mentorisanje. Stil instalacije - sistem za mentorisanje. Individualna vežba: Zadatak za samostalan rad. Zadaci za samostalni rad - stil raspoređivanja. Zadaci za samostalni rad - stil instalacije. Zadaci za samostalni rad - stil dodeljivanja radnih zadataka. Zadaci za samostalni rad - kombinovanje različitih pogleda. Zadaci za samostalni rad-4+1 Kručtenovi pogledi. Zadaci za diskusiju na

					vežbi.
6		**L06 - Upotreba šablona projektovanja softvera**	1. **Šablon Abstraction–Occurrence:** Kontekst, problem i ograničenja šablona Abstraction-Occurrence. Rešenje i primer upotrebe šablona Abstraction-Occurrence. Primeri nepravilne upotrebe šablona Abstraction-Occurrence. 2. **Šablon General Hierarchy:** Kontekst, problem i ograničenja šablona General Hierarchy. Rešenje i primer upotrebe šablona General Hierarchy. Primer upotrebe šablona General Hierarchy. 3. **Šablon Player–Role:** Kontekst, problem i ograničenja šablona Player–Role. Rešenje i primer upotrebe šablona Player-Role. Primer upotrebe šablona Player-Role. 4. **Šablon Delegation:** Kontekst, problem i ograničenja šablona Delegation. Rešenje i primer upotrebe šablona Delegation. 5. **Šablon Immutable:** Kontekst, problem i ograničenja šablona Immutable. 6. **Šablon Read-Only Interface:** Kontekst, problem i ograničenja šablona Read-Only Interface. Rešenje i primer upotrebe šablona Read-Only Interface. 7. **Šablon Proxy:** Kontekst, problem i ograničenja šablona Proxy. Rešenje i primer upotrebe šablona Proxy. Proxy pattern (Video). 8. **Objektni klijent-server okvir u projektovanju:** Fabrika klijent konekcije. Primetan sloj. Implementacija Observable sloja. Korišćenje Observable sloja. Poteškoće i rizici prilikom primene šablona projektovanja.	Razumevanje upotrebe šablona Abstraction–Occurrence, General Hierarchy, Player–Role, Delegation, Immutable, Read-Only Interface i Proxy; Razumevanje konkteksta, problema i ograničenja navedenih šablona; Primena šablona i razumevanje primera nepravilne upotrebe navedenih šablona na softverskoj arhitekturi;	Pokazna vežba: Primer upotrebe šablona Abstraction-Occurrence. Primer upotrebe šablona General Hierarchy. Primer upotrebe šablona General Hierarchy - softver za vođenje knjigovodstva. Primer upotrebe šablona Player–Role. Primer sistema za mentorstvo - klasni dijagram. Individualna vežba: Zadatak za samostalni rad. Zadaci za samostalni rad - Abstraction-Occurrence. Zadaci za samostalni rad - General Hierarchy. Zadaci za samostalni rad - Player-Role. Zadaci za samostalni rad - Delegation. Zadaci za samostalni rad - Read-Only Interface. Zadaci za samostalni rad - Proxy. Zadaci za diskusiju na vežbi.

7		**L07 - Strategije i metodi projektovanja softvera**	1. **Metod projektovanja softvera:** Primena metoda projektovanja softvera. Znanje o korišćenju metoda projektovanja softvera. Tri komponente metode projektovanja softvera. 2. **Podrška koju metode projektovanja pružaju:** Standardizacija procesa projektovanja softvera. Tehnički problemi u metodi projektovanja softvera. Problemi iz ugla menadžmenta u metodi projektovanja softvera. Nedostaci u primeni metoda projektovanja softvera. Domen problema i uticaj domena problema. Klasifikacija domena problema. 3. **Dizajn virtuelna mašina:** Dizajn virtuelna mašina (DVM). Karakteristike dizajn virtuelne mašine. Primena dizajn virtuelne mašine. 4. **Procesi i strategije projektovanja softvera:** Opšti proceduralni model procesa projektovanja softvera. Prošireni proceduralni model procesa projektovanja softvera. Koraci u procesu transformacije i elaboracije. Dopuna proceduralnog modelu procesa. Strategije i faktori metode procesa projektovanja softvera. Primena strategije u procesu projektovanja softvera. 5. **Transformacije u procesu projektovanja softvera:** Faza elaboracije. Elaboration Phase (video). Restrukturiranje nakon faze elaboracije. Projektovanje dekompozicije odozgo na dole. The Top Down Approach to Software Development (video). Projektovanje po kompoziciji. Organizacioni uticaji na proces projektovanja. Poznati organizacioni elementi.	Razumevanje strategija i metoda projektovanja softvera, primena metoda projektovanja softvera, korišćenje metoda projektovanja; Razumevanje dela predstavljanja, procesnog dela i skupa heuristika, razumevanje osobina metoda projektovanja softvera; Primena proceduralnog modela procesa projektovanja softvera, razumevanje koraka u procesu transformacije i elaboracije; Primena strategije u procesu projektovanja softvera;	Pokazna vežba: Model sistema pre faze elaboracije. Model sistema nakon faze elaboracije. Model sistema nakon faze transformacije. Pokazni primeri - proceduralno i deklarativno znanje. Individualna vežba: Zadaci za samostalni rad. Zadaci za samostalni rad - metod projektovanja softvera. Zadaci za samostalni rad - dizajn virtuelna mašina. Zadaci za samostalni rad - procesi i strategije projektovanja. Zadaci za samostalni rad - transformacija u procesu projektovanja. Zadaci za diskusiju na vežbi.
8		**L08 - Tradicionalni metodi projektovanja softvera**	1. **Reprezentacija strukturne sistemske analize:** Reprezentacija korišćena za strukturnu sistemsku analizu. P-Spec specifikacija procesa. Rečnik podataka. 2. **Reprezentacija strukturnog projektovanja:** Reprezentacija korišćena za strukturno projektovanje. Drugi oblici opisivanja	Razumevanje tradicionalnih metoda projektovanja softvera, primena reprezentacije	Pokazna vežba: Korak 1 - kreiranje dijagrama konteksta. Korak 2 - kreiranje prvog

			strukturnog projektovanja. 3. **SSA/SD proces projektovanja:** Koraci 1 i 2 u transformaciji: SSA. Korak 3 u transformaciji : Analiza transakcije. Korak 4 u transformaciji : Analiza transformacije. Korak 5 u transformaciji : Kompletiranje procesa. SSA/SD proces projektovanja - pokazni primer. 4. **Koraci 1 i 2 u transformaciji: SSA:** Korak 1 u strukturnoj sistemskoj analizi. Korak 2 u strukturnoj sistemskoj analizi. 5. **Korak 3 u transformaciji : Analiza transakcije:** Analiza transakcije - dijagram toka podataka. Analiza transakcije - opis dijagrama toka podataka. 6. **Korak 4 u transformaciji : Analiza transformacije:** Operacije unutar analize transformacije. Opis analize transformacije. 7. **Korak 5 u transformaciji : Kompletiranje procesa:** Kompletiranje procesa projektovanja. 8. **SSA/SD proces projektovanja - pokazni primer:** Korak 1 - kreiranje dijagrama konteksta. Korak 2 - kreiranje prvog dijagrama toka podataka. Korak 3 - analiza transakcija. Korak 4 - kreiranje dopunjenog dijagrama toka podataka. Korak 5 - kreiranje finalnog dijagrama. 9. **Jackson strukturno programiranje (JSP):** Korak 1: Crtanje ulazno/izlaznog dijagrama. Korak 2: Kreiranje programskog dijagrama. Korak 3: Nabrajanje operacija i elemenata. Koraci 4 i 5: Konverzija programa u tekst. 10. **Korak 1: Crtanje ulazno/izlaznog dijagrama:** Modelovanje strukturnog dijagrama za primer sistema benzinske stanice. Transformacija strukturnog dijagrama sistema benzinske stanice. 11. **Korak 2: Kreiranje programskog dijagrama:** Modelovanje strukturnog programskog dijagrama. 12. **Korak 3: Nabrajanje operacija i elemenata:** Nabrajanje operacija i alociranje elemenata programa. 13. **Koraci 4 i 5: Konverzija programa u tekst:** Konverzija programa u tekst i dodavanje uslova. 14. **Jackson proces razvoja sistema (JSD):** JSD proces. 15. **JSD proces:** Model JSD procesa. Faza modelovanja. Faza umrežavanja. Faza umrežavanja- elaboracija. Faza implementacije.	strukturne sistemske analize i strukturnog projektovanja; Razumevanje načina upotrebe procesa transformacije i koraka unutar procesa transformacije; Razumevanje i primena Jackson strukturnog programiranja (JSP) i procesa razvoja softvera (JSD); Primena koraka unutar strukturnog programiranja i procesa razvoja softvera na realnom primeru softverske arhitekture;	dijagrama toka podataka. Korak 3 - analiza transakcija. Korak 4 - kreiranje dopunjenog dijagrama toka podataka. Korak 5 - kreiranje finalnog dijagrama. Dijagram toka podataka. Primer dijagrama toka podataka - hotel. Primer P-Spec specifikacije procesa. Individualna vežba: Zadatak za samostalni rad. Zadaci za diskusiju na vežbi.
9		**L09 - Ponovna upotreba**	1. **Ponovna upotreba postojećeg softvera:** Zašto koristiti stari softver?. Mogući su i neki problemi pri ponovnoj upotrebi	Razumevanje koristi i problema ponovne	Pokazna vežba:

		softvera	softvera. Načini ponovne upotrebe postojećeg softvera. Koji način primene postojećeg softvera izabrati?. What is Reuse-Oriented Software Engineering? (video). 2. **Radni okviri aplikacija:** Šta je radni okvir?. Radni okviri veb aplikacija. Povratni pozivi. Frameworks - Web Development (video). 3. **Linije softevrskog proizvoda:** Linija softverskog proizvoda. Specijalizovani tipovi softverskog proizvoda. Primer arhitekture linije proizvoda - sistema za slanje vozila. Postupak razvoja novog člana linije proizvoda. Konfigurisanje novog sistema. Konfigurisanje sistema u vreme njegovog raspoređivanja. Product lines Introduction (video). Product lines Definition (video). Product lines Advantages (video). Product line Case Study (video). Product line Case study architecture (video). 4. **Ponovna upotreba COTS softvera:** Prednosti i nedostaci COTS proizvoda. COTS sistemska rešenja i COTS integrisana rešenja. COTS sistemsko rešenje. ERP sistem koprimer COTS sistemkog rešenja. COTS integrisani sistemi. Primena veb servisa za integraciju COTS proizvoda. QNX - When COTS is not SOUP (Video). 5. **Studija slučaja: Udaljene meteorološke stanice:** Opis sistema. Tražena svojstva sistema za upravljanje stanicama. 6. **Pokazna vežba:** Pokazni primer 1 - COTS sistem online kupovine. 7. **Individualna vežba:** Zadaci za samostalan rad. Zadaci za samostalan rad - nastavak. Zadatak za vežbu - studija slučaja meteorološka stanica. Zadaci za diskusiju na vežbi.	upotrebe softvera pri razvoju novih softverskih sistema; Razumevanje koncepta aplikacionog okvira kao skupa objekta namenjenim ponovnoj upotrebi (reuse) i načina kako se aplikacioni okviri mogu da koriste pri razvoju aplikacija; Upoznavanje sa softverskim proizvodnim linijama, koje čini zajednička osnovna arhitektura i ponovno upotrebljive komponente koje se mogu konfigurisati u skladu sa potrebama; Primenjivanje konfiguracija gotovih aplikacionih softverskih sistema (off-the-shelf application software systems) pri razvoju novih sistema;	Pokazni primer 1 - COTS sistem online kupovine. Individualna vežba: Zadaci za samostalan rad. Zadaci za samostalan rad - nastavak. Zadatak za vežbu - studija slučaja meteorološka stanica. Zadaci za diskusiju na vežbi.
10		**L10 - Projektovanje softvera primenom komponenata**	1. **Softversko inženjerstvo sa komponentama:** Zašto koristiti upotrebljene komponente u razvoju softvera?. Principi razvoja softvera baziranog na komponentama. 2. **Softverska komponenta:** Šta je softverska komponenta?. Dve kritične karakteristike softverske komponente. Primer komponente koja prikuplja i isporučuje podatke sa senzora. Component Based Architecture Intro (video). 3. **Modeli komponenata:** Osnovni elementi modela komponenata. Servisi komponente primenjene kao programske	Razumevanje šta je softverska komponenta koje se uključuje u softver kao jedan izvršni element; Razumevanje ključnih elemenata modela softverske	Pokazna vežba: Pokazni primer 2: Otkaz i pad rakete Ariane 5. Individualna vežba:

			jedinice. Procesi projektovanja primenom komponenata (PSPK procesi). PSPK procesi za ponovnu upotrebu komponenata. Izazovi kreiranja komponente za ponovnu upotrebu. PSPK procesi sa upotrebom komponenti. Nalaženje, izbor i validacija komponente. 4. **Sastavljanje komponenata:** Tri načina spajanja komponenata. Spajanje preko nekompatibilnih interfejsa dve komponente. Primer primene komponente adapter. Primer komponenata sistema za biblioteku fotografija. Primer dela deskriptora interfejsa komponente PhotoLibrary. Projektantske odluke pri korišćenju komponenata. OOP vs Component - Handle your Game Objects (Video).	komponente i podrške koju obezbeđuje posrednički softver(midlver) za ove modele, Svest o ključnim aktivnostima procesa softverskog inženjerstva sa primenom ponovno upotrebljivih komponenata; Razumevanje tri različita tipa slaganja komponenata i nekih problema koje treba rešiti radi kreiranja novih komponenata sistema	Zadatak 1 - projektovanje komponenata za sistem generičkog poslovanja. Zadaci za diskusiju. Component and Connectors Views: Definition, Verification, Witnesses (Video).
11		**L11 - Projektovanje distribuiranih softverskih sistema**	1. **Distribuirani sistemi:** Šta je distribuirani sistem i zašto se koristi?. Performanse distribuiranih sistema. What is a distributed system? (video). 2. **Svojstva distribuiranih sistema:** Pitanja na koje projektant treba da odgovori. Proširivost distribuiranih sistema. Bezbednost distribuiranih sistema. Kvalitet servisa distribuiranih sistema. 3. **Modeli interakcije:** Procedurana interakcija i interakcija porukama. Proceduralni poziv (RPC) i poruke. Midlver. What is Middleware? User Experience and Portal (Video). 4. **Klijent server računarstvo:** Mapiranje serverskih i klijentskih procesa na fizičke računare. Slojevita arhitektura distribuiranih klijent-server sistema. The Client Server Model (Video). 5. **Arhitektonski šabloni distribuiranih sistema:** Arhitektonski stilovi distribuiranih sistema. Arhitektura gospodar-rob. Dvoslojna klijent-server ahritektura. Primer primene modela sa debelim klijentom. Višeslojna klijent-server arhitektura. Preporuke za primenu različitih varijanti višeslojne arhitekture. Arhitekture sa distribuiranim komponentama. Korist od primene	Znati ključna pitanja koja se moraju uzeti u obzir prilikom projektovanja i implementacije distribuiranih softverskih sistema; Razumevanje klijent-server kompjuterskog modela i slojevitu arhitekturu klijent-server sistema; Znati uobičajene arhitektonske šablone distribuiranih sistema i znati odabrati odgovarajuću arhitekturu za	Pokazna vežba: Pokazni primer 1 - distribuirani sistem zdravstvene zaštite. Pokazni primer 2. Pokazni primer 3. Individualna vežba: Zadatak 1 - arhitektura sistema bolnica. Zadatak 2 - arhitektura sistema za upravljanje akcijama. Zadatak 3 - arhitektura sistema Nacionalnog

			sistema sa distribuiranim komponentama. Nedostaci primene arhitekture sa distribuitabiranim komponentama. Arhitekture ravnopravnih računara. Polucentralizovana arhitektura sa ravnopravnim računarima. Layered/n-Tier Architectural Pattern (Video). 6. **Softver kao servis - SaaS:** Softver kao servis. SaaS i SOA - koja je razlika?. Dinamičko konfigurisanje servisnog softvera. Proširljivost SaaS sistema. What is Middleware? Service Oriented Architecture Explained (Video). 7. **Pokazna vežba:** Pokazni primer 1 - distribuirani sistem zdravstvene zaštite. Pokazni primer 2. Pokazni primer 3. 8. **Individualna vežba:** Zadatak 1 - arhitektura sistema bolnica. Zadatak 2 - arhitektura sistema za upravljanje akcijama. Zadatak 3 - arhitektura sistema Nacionalnog pozorišta. Zadatak 4 - SaaS Dropbox sistem. Dodatni zadaci za diskusiju.	određenu vrstu distribuiranog sistema; Razumeti pojam softvera kao usluge, pružajući veb pristup daljinskim raspoređenim aplikacionim sistemima. Sposobnost da se distribuira softver na hardveru, razumevanje komunikacije između distribuiranih komponenti i zhati upotrebu posredničkog softvera (midlvera) u slučaju heterogenih softverskih sistema	pozorišta. Zadatak 4 - SaaS Dropbox sistem. Dodatni zadaci za diskusiju.
12		**L12 - Projektovanje servisno-orijentisanog softvera**	1. **Servisno-orijentisano softversko inženjerstvo:** Veb servis. Korist od upotrebe servisno-orijentisanog softverskog inženjerstva. Primer informacionog sistema automobila baziranog na servisima. What is service oriented architecture SOA? (video). 2. **Servisno-orijentisana arhitektura:** Šta je servisno-orijentisana arhitektura?. Standardi veb servisa. Tendencije u razvoju servisno-orijentisanih aplikacija. Tri aspekta veb servisa. WSDL koncepcijski model. Primer WSDL opisa veb servisa. Service Oriented Architecture (video). Introduction to Web Services (video). 3. **RESTful servisi:** Šta je REST?. Primer RESTful servis. Neki problemi sa primenom RESTful servisa. What are RESTful Services (RESTful APIs)? (video). 4. **Inženjerstvo servisa - za zamenu:** Šta je inženjerstvo servisa?. Utvrđivanje mogućih servisa. Kako izabrati potrebne servise?. Studija slučaja - kako izabrati potrebne servise?.	Razumevanje osnovnih pojmove veb servisa, standarda veb servisa, i servisno-orijentisane arhitekture; Razumevanje procesa inženjeringa usluga koji je namenjen proizvodnji veb servisa za višekratnu upotrebu; Upoznavanje sa pojmom sastava servisa kao sredstva razvoja servisno-	Pokazna vežba: Pokazni primer. Pokazni primer - nastavak. Pokazni primer - model poslovnog konteksta. Pokazni primer - konceptualni prikaz servisa. Individualna vežba: Zadatak 1. Dodatni zadaci za

			Studija slučaja - kako izabrati potrebne servise?. Projektovanje interfejsa servisa. Projektovanje interfejsa servisa sa izuzecima. Definisanje strukture poruka. Implementacija i raspoređivanje servisa. Service-Oriented Composition (video). 5. **Sastavljanje servisa:** Povezivanje aktivnosti različitih poslovnih procesa. Faze konstruisanja sistema sastavljanjem više servisa. Primer servisa za rezervaciju hotela. Primer povezivanja dva radna toka. Testiranje sastavljenog servisa. Web Service Composition (video). 6. **Projektovanje sistema sa mikroservisima:** Arhitektura mikroservisa. Koristi od mikroservisa. Operativni aspekt primene arhitekture mikroservis. API prolaz. Šablon projektovanja sa API prolazom. Šablon: Magistrala događaja (Event Bus). Šablon: Mreža servisa (Service Mesh). Primer primene servisne mreže u jednoj aplikaciji. Šablon: Backends for Frontends (BFF). Najbolje pprakse: Projektovanje u skladu sa domenom. Najbolje prakse: Decentralizovano upravljanje podacima. Najbolje prakse: Asihrone komunikacije. Najbolje prakse. Primena mikroservisa. Primer: Arhitektura aplikacije za onlajn kupovine. Primer: Miroservisi vođeni događajima sa Vert.x. Primer: Arhitektura sa mikroservisima koja koristi omni kanal u oblaku. Introduction to Microservices (video).	orijentisane aplikacije; Razumevanje kako se modeli poslovnih procesa mogu da da koristi za projektovanje servisno-orijentisanih sistema	diskusiju.
13		**L13 - Projektovanje softvera u realnom vremenu**	1. **Projektovanje softvera u realnom vremenu:** Sistemi za upravljanje u realnom vremenu. Razlika ugrađenih sistema i ostalih softverskih sistema. Introduction to real time software systems (video). 2. **Projektovanje ugrađenih sistema:** Model podsticaja i odgovora. Opšti model ugrađenih sistema u realnom vremenu. Aktivnosti procesa projektovanja sistema u realnom vremenu. Sinhronizacija rada dva procesa pomoću kružnog bafera. Get i Put operacije. Modeliranje sistema u realnom vremenu. Akcije sistema za upravljanje pumpe za gorivo. Programiranje sistema u realnom vremenu. Concepts of Real Time Systems (video). 3. **Šabloni projektovanja arhitekture softvera:** Šabloni za projektovanje arhitekture sistema u realnom vremenu. Osmatraj i reaguj (Observe and react) šablon. Primer sistema za zaštitu protiv provala. Šablon Kontrola okružanja (Environment Control). Primer kočionog sistema protiv proklizavanja. Šablon Procesni kanal (Process pipline). Primer	Razumevanje koncepta ugrađenog softvera (embedded software), koji s ekoristi kod upravljačkih sistema koji reaguju na spoljne događaja u njihovom okruženju. Sticanje znanja o procesu projektovanja sistema u realnom-vremenu (real-time systems), u kojima su softverski sistemi organizovani u vidu	Pokazna vežba: Pokazni primer 1 - sistem za upravljanje vozom. Pokazni primer 2 - utvrđivanje podsticaja. Pokazni primer 3 - arhitektura procesa. Individualna vežba: Zadatak 1 - Arhitektura zadatog sistema. Dodatni zadaci za rad.

			sistema za prikupljanje podataka fluksa neutrona. Architectural patters for real-time systems (video). 4. **Analiza vremena:** Tri ključna faktora projektovanja ugrađenih sistema. Vremenska analiza pada napajanja uređaja. Zahtevi vremena. Raspoređivanje funkcija konkurentnim procesima. Određivanje vremena trajanja svakog izvršenja procesa. Real-Time Software Modeling, Part 1 (video). Real-Time Software Modeling, Part 2 (video). 5. **Operativni sistemi u realnom vremenu:** Osnovne komponente operativnog sistema u realnom vremenu. Upravljanje procesom. Akcije operativnog sistema za upravljanje procesima. Pravila planiranja rada sistema. RTOS Tutorial (1/5) : Why is RTOS required? (video). RTOS Tutorial (2/5) : Task, handler and API (video). RTOS Tutorial (3/5) : Semaphore and event flag (video). RTOS Tutorial (4/5) : Architecture and Performance of RT (video). 6. **Individualna vežba:** Zadatak 1 - Arhitektura zadatog sistema. Dodatni zadaci za rad.	seta kooperišućih procesa. Razumevanje tri arhitektonska šablona koji se često koriste pri projektovanju ugrađenih sistema u realnom vremenu. Razumevanje organizacije operativnih sistema u realnom vremenu i uloge koje oni imaju kod ugrađenih sistema u realnom vremenu	
14		**L14 - Pojektovanje pouzdanog softvera**	1. **Greške u softverskom sistemu:** Vrste grešaka. Pristupi poboljšanju pouzdanosti softvera. Prihvatljivost grešaka. Failure, Fault, and Error - Georgia Tech - Software Development Process (video). 2. **Dostupnost i pouzdanost:** Definicije pouzdanosti i dostupnosti. Uticaj pojedinih ulaza na greške u izlazu. Skup ulaza u sistem. Povezanost pouzdanosti i dostupnosti. Availability and reliability (Ian Sommerville) (video). 3. **Zahtevi pouzdanosti:** Zahtevi pouzdanosti. Metrika pouzdanosti i dostupnosti. Upotreba POFOD, ROCOF i MMTF. Nefunkcionalni zahtevi pouzdanosti. Uputstva za specifikaciju pouzdanosti. Primer mogućih grešaka u radu insulin pumpe. Specifikacije funkcionalne pouzdanosti. Understand 'MTTR', 'MTBF' 'MTTF' (video). Reliability, Availability - Georgia Tech (video). 4. **Arhitekture otporne na greške:** Tolerancija grešaka. Zaštitni sistemi. Samoosmatrajuće arhitekture. Primer arhitekture kontrolnog sistema Airbus 340 aviona. Trostruka modularna redudantnost. Programiranje sa N verzija. Različitost softvera. Pogrešna interpretacija specifikacije. Fault Tolerance Techniques - Georgia Tech (Video). Redundancy, Fault Tolerance, and High Availability (Video).	Razumevanje razlike između pouzdanosti softvera (software reliability) i dostupnosti softvera (software availability) Upoznavanje se metrikom za specifikaciju pouzdanostu i njenom upotrebom pri specifikaciji zahteva za merljivu pouzdanost Razumevanje upotrebe različitih arhitektonskih stilova radi primene pouzdane arhitekture sistema otporne na greške	Pokazna vežba: Pokazni primer 1. Pokazni primer 2. Pokazni primer 2 - nastavak. Individualna vežba: Zadatak 1. Zadaci za individualni rad studenata.

			5. **Programiranje za pouzdanost:** Preporuke za programiranje za minimiziranje javljanja grešaka:1 i 2. Preporuke za programiranje za minimiziranje javljanja grešaka:3. Preporuke za programiranje za minimiziranje javljanja grešaka:4 i 5. Preporuke za programiranje za minimiziranje javljanja grešaka:6 i 7. Preporuke za programiranje za minimiziranje javljanja grešaka: 8. Principles of Programming Languages (Video). 6. **Merenje pouzdanosti:** Potrebni podaci za utvrđivanje pouzdanosti softvera. Faze statističkog testiranja radi merenja pouzdanosti. Generisanje test podataka. Profili rada. 7. **Individualna vežba: Zadaci:** Zadatak 1. Zadaci za individualni rad studenata.	Sticanje znanja o programerskoj praksi u inženjerstvu pouzdanog softvera Razumevanje upotrebe statističkog testiranja radi merenja pouzdanosti softverskog sistema	
15		**L15 - Analiza i ocena kvaliteta projektnog rešenja softvera**	1. **Upravljanje kvalitetom softvera:** Upravljanje kvalitetom, obezbeđenje kvaliteta i kontrola kvaliteta. Plan kvaliteta. Introduction to software quality assurance (video). What is a Quality Management System (QMS)? (video). 2. **Kvalitet softvera:** Ocenjivanje kvaliteta softvera. Kvalitet baziran na procesu. Software quality in software engineering (video). 3. **Softverski standardi:** Standardi poizvoda i standardi procesa. Primena standarda. ISO 9001 standardni okvir. ISO9001 akreditacija i ISO9001 sertifikat. What Is ISO 9001 ? (video). 4. **Recenzije i kontrole:** Svrha recenzije. Proces recenzije. Kontrola softvera. Efektivnost kontrole kvaliteta. Software Peer Reviews: An Executive Overview (video). 5. **Upravljanje kvalitetom i agilni razvoj:** Programiranje u paru. Slučaj velikih sistema. 6. **Merenje softvera:** Metrike softvera. Merenje procesa. Veze spoljnih i unutrašnjih atributa. Merenje sistemskog merenja kvaliteta u kompanijama. Metrika proizvoda. Metrike objektno-orijentisanih softverskih sistema. Analiza komponente softvera. Analitika softvera. Preporuke. Software Quality Metrics You Need To Know (video). Quality assurance tutorial: How to think about quality (video). 7. **Pokazna vežba:** Pokazni primer - merenje dvosmislenosti. Pokazni primer - merenje dvosmislenosti - nastavak. 8. **Individualna vežba:** Zadaci za samostalan rad.	Razumevanje procesa upravljanja kvalitetom softvera i svesti o važnosti planiranja kvaliteta; Razumeti vezu između kvaliteta softvera i procesa razvoja softvera koji se primenjuje; Razumevanje značaja standarda u procesu upravljanja kvalitetom i sticanje znanja kako primeniti standarde radi obezbeđenja kvaliteta; Razumevanje pregleda i kontrole upotrebljenih kao mehanizmi za obezbeđenja kvaliteta softvera; Razumevanje kako merenje može da	Pokazna vežba: Pokazni primer - merenje dvosmislenosti. Pokazni primer - merenje dvosmislenosti - nastavak. Individualna vežba: Zadaci za samostalan rad.

				bude od koristi u oceni atributa kvaliteta softvera i ograničenja u merenju softvera.	