DALYKO APRAŠAS

Dalyko pavadinimas	Kodas
Architektūriniai programų sistemų stiliai	

Dėstytojas (-ai)	Padalinys
Koordinuojantis: asist. dr. Andrius Vytautas Misiukas Misiūnas	Matematikos ir informatikos fakultetas Duomenų mokslo ir skaitmeninių technologijų institutas

Studijų pakopa	Dalyko tipas
Pirmoji	Pasirenkamasis

Įgyvendinimo forma	Vykdymo laikotarpis	Vykdymo kalba (-os)
Auditorinė	5 semestras	Lietuvių / Anglų

Reikalavimai studijuojančiajam
Išankstiniai reikalavimai: Procedūrinis programavimas, Algoritmai ir duomenų struktūros, Objektinis programavimas, Operacinės sistemos, Duomenų bazių valdymo sistemos, Naudotojo sąsajos kūrimas.

Dalyko apimtis kreditais	Visas studento darbo krūvis	Kontaktinio darbo valandos	Savarankiško darbo valandos
5	135	64	71

Dalyko tikslas: studijų programos ugdomos kompetencijos		
Dalyko tikslas – siekiama, kad studentai supažindintų su aukšto abstrakcijos lygio programiniais stiliais plačiai naudojamais praktikoje, gebėtų aptarti esminius elementus, kurių pagalba galima sukonstruoti abstrakčią struktūrą, tinkančią daugeliui šiandienos informacinių sistemų, ugdytų gebėjimus kurti lengvai modifikuojamas ir pakartotinai panaudojamas sistemas ar jų komponentus.		
Dalyko studijų siekiniai	**Studijų metodai**	**Vertinimo metodai**
Gebės analizuoti užduotis siekiu išskirti tipinius architektūrinius šablonus, sudaryti įgyvendinimo darbų planą, jį vykdyti pasinaudojant grupinėmis programinio kodo dalijimosi sistemomis.	Pavyzdžių nagrinėjimas, literatūros skaitymas ir analizė, savarankiškas darbas, praktinių užduočių atlikimas, probleminis dėstymas.	Egzaminas, praktinių užduočų vertinimas.
Gebės analizuoti programų sistemų architektūrinius stilius, vertins jų tinkamumą ir pritaikomumą sprendžiant konkrečias užduotis.		
Gebės suprojektuoti esamų programinių sprendimų atnaujinimą, pritaikyti optimalų programų sistemų architektūrinį stilių.		
Gebės pasirinkti reikiamą programavimo kalbą, papildomai įgyvendinti naują funkcionalumą keisdamas esamų klasių struktūras ir programos veikimo logiką.		
Gebės apjungti keletą programų sistemų architektūrinių stilių siekiant sukurti veikiantį programinį įgyvendinimą.		
Gebės programų sistemų architektūrinių stilius pritaikyti mašinų virtualizavimo srityje, kuriant ir diegiant debesų kompiuterijos sprendimus.		

Temos	Kontaktinio darbo valandos							Savarankiškų studijų laikas ir užduotys	
	Paskaitos	Konsultacijos	Seminarai	Pratybos	Laboratoriniai darbai	Praktika	**Visas kontaktinis darbas**	**Savarankiškas darbas**	**Užduotys**
1. Įvadas, vertinimo kriterijų aptarimas.	1						**1**	**1**	Temos kuriamai informacinei sistemai formulavimas, 1-3 studentų grupės darbui surinkimas.
2. Kodo dalinimosi ir versijavimo sistemos Git pagrindai.	1			2			**3**	**4**	
3. Programų sistemų architektūros diagramos	2			2			**4**	**4**	Pradinio kuriamos sistemos plano parengimas.
4. Programų sistemų architektūros kūrimo ir dizaino metodika	4			4			**8**	**6**	
5. Sluoksniavimo strategija	3			3			**6**	**4**	Praktinės užduoties suplanuotų etapų atlikimas.
6. Prezentacijos sluoksnis	2			2			**4**	**4**	
7. Logikos sluoksnis	2			2			**4**	**4**	
8. Web servisų sluoksnis	2			2			**4**	**4**	

9. Duomenų sluoksnis	2			2			**4**	**4**	
10. SOLID principas	1			1			**4**	**4**	
11. Sluoksnių architektūros komponentų dizainas	2			2			**4**	**4**	
12. Bendriniai rūpesčiai	2			2			**4**	**4**	
13. 2, 3 ir N *Tier* architektūra	2			2			**4**	**4**	
14. Į objektus orientuotas stilius, paveldėjimas, kompozicija, agregacija, paveldėjimas, polimorfizmas, perpanaudojimas	2			2			**4**	**4**	
15. Komponentais pagrįstas stilius	2			2			**2**	**2**	
16. *Message bus* stilius	2			2			**2**	**2**	
Pasiruošimas egzaminui.								**12**	Savarankiškas paskaitų medžiagos ir literatūros studijavimas
Iš viso	**32**			**32**			**64**	**71**	

Vertinimo strategija	**Svoris proc.**	**Atsiskaitymo laikas**	**Vertinimo kriterijai**
Pasirinktos informacinės sistemos architektūros kūrimas ir programavimas.	40	Keturiais etapais, 4-14 semestro savaitę.	Per semestrą kiekvienas studentas privalo įgyvendinti informacinę sistemą pasirinkta tema, pateikdamas ir apgindamas sukurtas kompiuterines realizacijas. Reikalaujama, kad studento parašyta kompiuterinė realizacija ne tik duotų teisingus rezultatus, tačiau studentas sugebėtų paaiškinti jos veikimo principus, atsakytų į susijusius su užduotimi klausimus (apie sąvokas, algoritmus). Atsiskaitoma šiais etapais: pradinio plano pristatymas, 4 programavimo iteracijos (iki 4 balų), galutinės dokumentacijos atsiskaitymas (iki 1 balo). Vertinimo kriterijai: algoritmo veikimo teisingumas, atitikimas nefunkciniams reikalavimams, efektyvus programavimas, teisingos ir išsamios darbo išvados, atsakymų į klausimus teisingumas, nuoseklumas ir aiškumas. Studentams leidžiama pasitaisyti programavimo iteracijų atsiskaitymo metu rastas klaidas su kita iteracija. Egzaminą leidžiama laikyti jei atsiskaityta bent viena programavimo iteracija.
Galutinės sukurtos informacinės sistemos dokumentacijos atsiskaitymas	10	14-16 semestro savaitė.	
Egzaminas raštu	50	Sesijos metu	Raštu pateikiami klausimai ir užduotys apie sąvokas, programų architektūros stilius ir principus, jų taikymo pavyzdžius ir įgyvendinimo aspektus. Vertinimo kriterijai: teisingi ir išsamūs atsakymai į klausimus, aiškus atsakymų pateikimas, tinkamas atsakymų argumentavimas.
Egzamino perlaikymas ir eksternas	50-100	Perlaikymo sesijos metu.	Studentai, perlaikantys egzaminą arba laikantys eksternu gali atsiskaityti pratybų užduotį egzamino perlaikymo metu. Perlaikymo metu visa užduotis atsiskaitoma iškart, tai yra nelieka reikalavimo atsiskaityti iteracijomis, tačiau nelieka galimybės pasitaisyti programavimo klaidų kaip semestro metu. Jeigu studentas semestro metu neatsiskaitė pratybų užduoties be pateisinamų priežasčių, už pratybų užduotį, jam skiriami tik gelbėjimosi balai (į juos žiūrima tik priimant sprendimą ar prileisti prie egzamino ir jeigu įprastų balų neužtenka parašyti minimaliam teigiamam balui). Studentai, pratybų užduoties neatsiskaitę laiku dėl pateisinamų priežasčių gali gauti pilną balą perlaikymo metu. Maksimalus balas laikant eksternu arba jei studentas semestro metu neatsiskaitė nė vienos programavimo iteracijos, yra 5.

Autorius	**Leidimo metai**	**Pavadinimas**	**Periodinio leidinio Nr. ar leidinio tomas**	**Leidimo vieta ir leidykla ar internetinė nuoroda**
Privalomoji literatūra				
Andrius Vytautas Misiukas Misiūnas		Paskaitų skaidrės ir kita susijusi medžiaga.		VU e-mokymų sistema
Autorių kolektyvas	2009	NET Application Architecture Guide /Microsoft Patterns & Practices	ISBN-13: 978-0735627109	Redmond, Washington, Microsoft. http://fizyka.umk.pl/~jacek/docs/net/Application_Architecture_Guide_v2.pdf
Martin Fowler	2003	Patterns of Enterprise Application Architecture	ISBN-13: 978-0321127426	Boston, Addison-Wesley, https://github.com/jjcolumb/awesome-xaf/blob/master/Patterns%20of%20Enterprise%20Application%20Architecture%20-%20Martin%20Fowler.pdf

Papildoma literatūra				
Robert C. Martin	2009	Clean Code: A Handbook of Agile Software Craftsmanship	ISBN-13: 9780132350884	Upper Saddle River, NJ, Prentice Hall, https://github.com/jnguyen095/clean-code/blob/master/Clean.Code.A.Handbook.of.Agile.Software.Craftsmanship.pdf
Robert C. Martin	2017	Clean Architecture: A Craftsman's Guide to Software Structure and Design	ISBN-13: 978-0134494166	Upper Saddle River, NJ, Prentice Hall, https://github.com/yogathanh99/Books/blob/master/Book%20-%20Clean%20Architecture%20-%20Robert%20Cecil%20Martin.pdf

COURSE UNIT (MODULE) DESCRIPTION

Course unit (module) title	Code
Software Systems Architectural Styles	

Lecturer(s)	Department(s) where the course unit (module) is delivered
Coordinator: Assist. Dr. Andrius Vytautas Misiukas Misiūnas	Faculty of Mathematics and Informatics Institute of Data Science and Digital Technologies

Study cycle	Type of the course unit (module)
First	Optional

Mode of delivery	Period when the course unit (module) is delivered	Language(s) of instruction
face-to-face	5th semester	Lithuanian / English

Requirements for students	
Prerequisites: Procedural programming, Algorithms and Data Structures, Object Oriented Programming, Operation Systems, Database management Systems, User Interface Design	**Additional requirements (if any):**

Course (module) volume in credits	Total student's workload	Contact hours	Self-study hours
5	135	64	71

Purpose of the course unit (module): programme competences to be developed		
During this course students will be introduced to high level architectural styles that are used for applications. Architectural style describes a set of principles - a coarse grained pattern that provides an abstract framework for a family of systems. On a high level architectural styles resolve major forces in application structure and enable creation of modifiable and reusable systems. For each style, students will be introduced to an overview, key principles, major benefits, and information that will help to choose the appropriate architectural styles for an application.		
Learning outcomes of the course unit (module)	**Teaching and learning methods**	**Assessment methods**
Will be able to analyse typical software design patterns, and prepare task implementation plans, use code groupware code management systems.	Example analysis, literature reading and analysis, self-study work, tutorials, laboratory work, problem-oriented teaching.	Exam, Test, Assessment of Laboratory work, Assessment of group work.
Ability to search literature for the reference software architecture design patterns.		
Will be able to design solution update, select and apply the most suitable design pattern structure.		
Will be able to select right programming language and modify classes and behavioural solution logic.		
Will be able to combine separate styles to develop single software.		
Will be able to introduce architectural style design knowledge to the cloud computational solutions.		

Content: breakdown of the topics	Contact hours							Self-study work: time and assignments	
	Lectures	Tutorials	Seminars	Exercises	Laboratory work	Internship/work placement	**Contact hours**	**Self-study hours**	**Assignments**
1. Introduction. Evaluation criterion.	1			2			**1**	**1**	Topic selection, formation of workgroups (1-3 students)
2. Source code sharing and version control systems.	1						**3**	**4**	
3. Architectural diagrams of software systems	2			2			**4**	**4**	Preparation of the plan of the system
4. Session Design and development of software systems architecture	4			4			**8**	**6**	

5. Layered style	3			3			**6**	**4**	Execution of system development stages
6. Presentation layer	2			2			**4**	**4**	
7. Logical layer	2			2			**4**	**4**	
8. Web service layer	2			2			**4**	**4**	
9. Data layer	2			2			**4**	**4**	
10. SOLID design	1			1			**4**	**4**	
11. Layered architecture design	2			2			**4**	**4**	
12. Separation of concerns	2			2			**4**	**4**	
13. N-Tier/3-Tear style	2			2			**4**	**4**	
14.Object-oriented style: abstraction, generalization, encapsulation, composition, polymorphism.	2			2			**4**	**4**	
15. Component-based style	2			2			**4**	**2**	
16. Message-Bus style	**2**			**2**			**4**	**2**	
Preparation for exam								**12**	Literature analysis.
Total	**32**			**32**			**64**	**71**	

Assessment strategy	**Weight, %**	**Deadline**	**Assessment criteria**
Development of the selected information system architecture	40	In four steps, 4th - 14th weeks of the semester	During the semester, each student must implement the information system on the chosen topic, presenting and defending the created computer realizations. It is required that the computer implementation written by the student not only gives correct results, but that the student is able to explain the principles of its operation, answer questions related to the task (about concepts, algorithms). Grading is conducted in the following stages: presenting the initial plan, 4 programming iterations (up to 4 points), final documentation presentation (up to 1 point). Evaluation criteria: correctness of algorithm operation, compliance with non-functional requirements, efficient programming, correct and detailed work conclusions, correctness, consistency and clarity of answers to questions. Students are allowed to correct errors found during presenting of programming iterations with the next iteration. A student is allowed to take the exam if at least one programming iteration has been presented.
Assessment of the information system documentation	10	14th-16th week of the semester	
Exam	50	During exam session	The question and tasks are presented on the concepts, application architecture styles and principles, examples of their application and implementation aspects. Evaluation criteria: correct and complete answers to the questions, clear presentation of answers, proper reasoning of the answers.
Exam retake and externship	50-100	During exam retake session	Students retaking an exam or taking an external exam can present the exercise task during the retaking of the exam. During retake of the exam, the entire assignment is presented at once, that is, there is no requirement to present in iterations, but there is no possibility to correct programming errors like during the semester. If a student does not report an exercise assignment during the semester without justifiable reasons, only rescue marks are awarded for the exercise assignment (they are only considered when making a decision whether to allow the student to take the exam and if the normal marks are not enough to write the minimum positive grade). Students who do not report the exercise assignment on time due to justifiable reasons may receive a full mark during the retake. The maximum score for an external student or if the student has not reported any programming iterations during the semester is 5.

Author	**Year of publication**	**Title**	**Issue of a periodical or volume of a publication**	**Publishing place and house or web link**
Compulsory reading				
Andrius Vytautas Misiukas Misiūnas		Lecture notes and slides		Virtual Learning Environment of Vilnius University
	2009	NET Application Architecture Guide Microsoft Patterns & Practices	ISBN-13: 987-0-321-55268-6	Redmond, Washington, Microsoft. http://fizyka.umk.pl/~jacek/docs/net/Application_Architecture_Guide_v2.pdf

Martin Fowler	2003	Patterns of Enterprise Application Architecture	ISBN-13: 978-0321127426	Boston, Addison-Wesley, https://github.com/jjcolumb/awesome-xaf/blob/master/Patterns%20of%20Enterprise%20Application%20Architecture%20-%20Martin%20Fowler.pdf
Optional reading				
Robert C. Martin	2009	Clean Code: A Handbook of Agile Software Craftsmanship	ISBN-13: 9780132350884	Upper Saddle River, NJ, Prentice Hall, https://github.com/jnguyen095/clean-code/blob/master/Clean.Code.A.Handbook.of.Agile.Software.Craftsmanship.pdf
Robert C. Martin	2017	Clean Architecture: A Craftsman's Guide to Software Structure and Design	ISBN-13: 978-0134494166	Upper Saddle River, NJ, Prentice Hall, https://github.com/yogathanh99/Books/blob/master/Book%20-%20Clean%20Architecture%20-%20Robert%20Cecil%20Martin.pdf