DALYKO APRAŠAS

Dalyko pavadinimas	Kodas
Programų sistemų architektūra	**ITPSA**

Dėstytojas (-ai)	Padalinys
Koordinuojantis: asist. dr. Andrius Vytautas Misiukas Misiūnas	Informatikos institutas, Vilniaus universitetas

Studijų pakopa	Dalyko tipas
Pirmoji	Privalomasis

Įgyvendinimo forma	Vykdymo laikotarpis	Vykdymo kalba (-os)
Auditorinė	6 semestras	Lietuvių / Anglų

Reikalavimai studijuojančiajam
Išankstiniai reikalavimai: Programavimo ir IT pagrindai, Objektinis programavimas, Duomenų bazių valdymo sistemos, Programų sistemų inžinerija, Interneto technologijos, Žmogaus ir kompiuterio sąveika.

Dalyko apimtis kreditais	Visas studento darbo krūvis	Kontaktinio darbo valandos	Savarankiško darbo valandos
5	144	66	78

Dalyko tikslas: studijų programos ugdomos kompetencijos

Bendrosios kompetencijos:

- įgyti dalykinės srities žinių ir suprasti savo profesiją (*BK2*),
- spręsti problemas (*BK4*),
- organizuoti ir planuoti darbus (*BK6*).

Dalykinės kompetencijos:

- taikyti programų projektavimo bendruosius metodus, formuluoti ir analizuoti programinės įrangos reikalavimus (*DK1*),
- analizuoti uždavinio algoritmo procesą pagal algoritmų bendrąsias savybes (*DK2*),
- kurti ir specifikuoti programinės įrangos (arba IT paslaugos) projektą (*DK3*).

Dalyko studijų siekiniai	Studijų metodai	Vertinimo metodai
Paaiškinti PSA sąvokas, principus, metodus, sudedamąsias dalis ir darbų specifiką gimtąja ar užsienio kalbomis nepriklausomai nuo klausytojo profesijos.	Pavyzdžių nagrinėjimas, literatūros skaitymas ir analizė, savarankiškas darbas, praktinių užduočių atlikimas, probleminis dėstymas.	Egzaminas, praktinių užduočų vertinimas.
Apibrėžti taikomosios srities problemą ir taikyti tinkamus egzistuojančius priimtinus PSA siūlomus sprendimus.		
Įvertinus užduoties sudėtingumą ir atsižvelgus į pradinius bei tarpinius dėstytojo ir/ar komandos vadovo komentarus ir rekomendacijas planuoti savo veiklas.		
Apibrėžti ir apibendrinti kuriamos programų sistemos sąsają, priklausomybę nuo kitos programinės ar techninės įrangos, ir pateikti integracijos specifikaciją.		
Formuluoti uždavinį įvairiuose abstrakcijos lygmenyse, suformuluotam uždaviniui pateikti algoritmą naudojantis schemomis ir aprašymais.		
Rašyti programinės įrangos specifikaciją, vartotojo instrukcijas.		

Detalizuoti ir tikslinti projekto specifikaciją, vertinti projekto specifikacijos atitiktį projekto ir jo kūrimo reikalavimams.		

Temos	**Kontaktinio darbo valandos**							**Savarankiškų studijų laikas ir užduotys**	
	Paskaitos	Konsultacijos	Seminarai	Pratybos	Laboratoriniai darbai	Praktika	**Visas kontaktinis darbas**	**Savarankiškas darbas**	**Užduotys**
1. Įvadinė paskaita: kurso apžvalga, susipažinimas ir tvarkų pasiaiškinimas. Programų sistemų architektūros apibrėžimas.	2			2			**4**	**2**	Temos kuriamai informacinei sistemai formulavimas, 1-3 studentų grupės darbui surinkimas.
2. Sistemų projektavimas naudojant UML. Principai, diagramos, dokumentavimas.	2			2			**4**	**4**	Pradinio kuriamos sistemos plano parengimas.
3. Programų sistemų architektūros diagramos.	2			2			**4**	**3**	
4. Architektūriniai šablonai ir stiliai.	4			4			**8**	**6**	
5. Architektūros dizaino metodika.	2			2			**4**	**4**	Praktinės užduoties suplanuotų etapų atlikimas.
6. Architektūrinio dizaino pagrindai, sluoksniuota struktūra.	2			2			**4**	**4**	
7. Pateikimo sluoksnis ir jo komponentai.	2			2			**4**	**4**	
8. Dalykinės srities sluoksnis ir jo komponentai.	2			2			**4**	**4**	
9. Duomenų sluoksnis ir jo komponentai.	2			2			**4**	**4**	
10. Paslaugų sluoksnis ir jo komponentai.	2			2			**4**	**4**	
11. Kokybiniai atributai.	2			2			**4**	**4**	
12. Specifiniai dizaino objektai.	2			2			**4**	**4**	
13. Komunikacija ir pranešimai.	2			2			**4**	**4**	
14. Fiziniai lygmenys ir realizacija. Debesų aplinka.	2			2			**4**	**4**	
15. Programų sistemų archetipai. Programų sistemų tipo pasirinkimas ir projektavimas.	2			2			**2**	**2**	
Pasiruošimas egzaminui ir egzamino laikymas.		2					**2**	**21**	Savarankiškas literatūros studijavimas. Skaidrių peržvalga. Ruošto projekto peržvalga.
Iš viso	**32**	**2**		**32**			**66**	**78**	

Vertinimo strategija	**Svoris proc.**	**Atsiskaitymo laikas**	**Vertinimo kriterijai**
Pradinio pasirinktos informacinės sistemos plano atsiskaitymas.	0	2-4 semestro savaitę.	Studentas turi gebėti paaiškinti kaip turėtų veikti jo kuriama informacinė sistema pasitelkiant UML ir kitas diagramas, paaiškinti jos funkcionalumą ir naudojimo atvejus. Atsiskaičius pradinę dokumentaciją, leidžiama pradėti įgyvendinimą. Vertinimo kriterijai: pakankama funkcionalumo apimtis (2-3 naudojimo atvejai kiekvienam studentui kiekvienam etapui), teisingi architektūriniai sprendimai bei tinkamas jų paaiškinimas (įskaitant UML bei kitas diagramas). Pradiniame plane numatytas funkcionalumas gali keistis tik suderinus su dėstytoju.
Pasirinktos informacinės sistemos architektūros	40	Keturiais etapais, 4-14	Semestro metu kiekvienas studentas privalo sukurti programų sistemą (pvz., informacinę sistemą, vaizdo žaidimą, mobiliąją

kūrimas ir programavimas.		semestro savaitę.	programėlę ar kt.), pristatyti ir apginti sukurtą sprendimą. Reikalingas teisingas sukurtos sistemos veikimas, išsamus jos veikimo principų paaiškinimas, atsakymai į su užduotimi susijusius klausimus (apie sąvokas, algoritmus). Vertinimo kriterijai: algoritmo veikimo teisingumas, programavimo efektyvumas, teisingos ir išsamios išvados, atsakymų teisingumas, nuoseklumas ir aiškumas, pristatymo savalaikiškumas, gebėjimas ištaisyti pristatymo metu nurodytus trūkumus, gebėjimas esant poreikiui naudotis universiteto suteikta infrastruktūra. Etiškas dirbtinio intelekto metodų naudojimas leidžiamas, bet naudojantiems gali būti didinami užduoties reikalavimai siekiant išlaikyti vienodą užduoties sudėtingumą visiems studentams. Dirbtinio intelekto naudojimas turi būti etiškas ir aiškiai deklaruotas iteracijų pristatymo metu ir galutinėje dokumentacijoje. Maksimalus 1 dalies įvertinimas yra 4 balai; 2 dalis vertinama iki 1 balo.
Galutinės sukurtos informacinės sistemos dokumentacijos atsiskaitymas	10	14-16 semestro savaitė.	
Egzaminas raštu	50	Sesijos metu.	Raštu pateikiami klausimai ir užduotys apie sąvokas, programų architektūrą ir principus, jų taikymo pavyzdžius ir įgyvendinimo aspektus. Vertinimo kriterijai: teisingi ir išsamūs atsakymai į klausimus, aiškus atsakymų pateikimas, tinkamas atsakymų argumentavimas, atsakymas į klausimą. Egzaminą laikyti gali tik studentai, bent iš dalies atsiskaitę pratybų užduotį (surinkę ≥ 0,5 balo iš pratybų).
Papildomi balai	0-20	Semestro ir sesijos metu.	Studentams, ypač gerai atlikusiems užduotį gali būti skiriami papildomi balai už sukurtos sistemos kokybę, įgyvendintą papildomą funkcionalumą, ypač gerai aprašytą dokumentaciją.
Dalyko atsiskaitymas eksternu negalimas.			

Autorius	**Leidimo metai**	**Pavadinimas**	**Periodinio leidinio Nr. ar leidinio tomas**	**Leidimo vieta ir leidykla ar internetinė nuoroda**
Privalomoji literatūra				
Dr. Andrius Vytautas Misiukas Misiūnas		Paskaitų skaidrės ir kita susijusi medžiaga.		VU e-mokymų sistema
Autorių kolektyvas	2009	NET Application Architecture Guide /Microsoft Patterns & Practices		Redmond, Washington, Microsoft.
Martin Fowler	2003	Patterns of Enterprise Application Architecture		Boston, Addison-Wesley.
Robert C. Martin	2017	Clean Architecture: A Craftsman's Guide to Software Structure and Design		Boston, Pearson/Addison-Wesley.
Papildoma literatūra				
Robert C. Martin	2009	Clean Code: A Handbook of Agile Software Craftsmanship		Upper Saddle River, NJ, Prentice Hall.
Craig Larman	2011	Applying UML and patterns :an introduction to object-oriented analysis and design and iterative development		Upper Saddle River, NJ, Prentice Hall.
Martin Fowler	2003 3rd edition	UML Distilled: A Brief Guide to the Standard Object Modeling Language		Addison-Wesley

COURSE UNIT (MODULE) DESCRIPTION

Course unit (module) title	Code
Software Architecture	**ITPSA**

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 Computer Science

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

Mode of delivery	Period when the course unit (module) is delivered	Language(s) of instruction
face-to-face	6th semester	Lithuanian / English

Requirements for students	
Prerequisites: Programming and IT Fundamentals, Object-Oriented Programming, Database Management Systems, Software Systems Engineering, Internet Technologies, Human-Computer Interaction.	**Additional requirements (if any):**

Course (module) volume in credits	Total student's workload	Contact hours	Self-study hours
5	144	64	78

Purpose of the course unit (module): programme competences to be developed		
General competencies: • acquire knowledge in the subject area and understand one's profession (BK2), • solve problems (BK4), • organize and plan work (BK6). Subject-specific competencies: • apply general methods of program design, formulate and analyse software requirements (DK1), • analyse the process of a problem algorithm according to the general properties of algorithms (DK2), • develop and specify a software (or IT service) project (DK3).		
Learning outcomes of the course unit (module)	**Teaching and learning methods**	**Assessment methods**
Explain software architecture concepts, principles, methods, components, and job specifics in native or foreign languages, regardless of the listener's profession.	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.
Define the problem in the application area and apply appropriate existing acceptable PSA proposed solutions.		
After assessing the complexity of the task and considering the initial and intermediate comments and recommendations of the teacher and/or team leader, plan activities.		
Define and summarize the interface of the application system being developed, its		

dependencies on other software or hardware, and provide an integration specification.		
Formulate a problem at various levels of abstraction, present an algorithm for the formulated problem using diagrams and descriptions.		
Write software specifications, user instructions.		
To detail and refine the project specification, to assess the compliance of the project specification with the requirements of the project and its development.		

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. Introductory lecture: course overview, introduction and explanation of procedures. Definition of software architecture.	2			2			**4**	**2**	Topic selection, formation of workgroups (1-3 students)
2. System design using UML. Principles, diagrams, documentation.	2			2			**4**	**4**	
3. Architectural diagrams of software systems.	2			2			**4**	**3**	Preparation of the plan of the system
4. Architectural patterns and styles.	4			4			**8**	**6**	
5. Architectural design methodology.	2			2			**4**	**4**	Execution of system development stages
6. Fundamentals of architectural design, layered structure.	2			2			**4**	**4**	
7. Presentation layer and its components.	2			2			**4**	**4**	
8. Business (logic) layer and its components.	2			2			**4**	**4**	
9. Data layer and its components.	2			2			**4**	**4**	
10. (Web) Service layer and its components.	2			2			**4**	**4**	
11. Qualitative attributes.	2			2			**4**	**4**	
12. Specific design objects.	2			2			**4**	**4**	
13. Communication and notifications.	2			2			**4**	**4**	
14. Physical layers and implementation. Cloud environment.	2			2			**4**	**4**	
15. Archetypes of application systems. Selection and design of application system types.	2			2			**2**	**2**	
Preparation for exam		2					**2**	**21**	
Total	**32**	**2**		**32**			**66**	**78**	Literature analysis.

Assessment strategy	Weight,%	Deadline	Assessment criteria
Submission and defence of the initial plan for the selected information system.	0	2nd – 4th weeks of the semester.	The student must be able to explain how the information system they are developing is intended to operate—using UML and other diagrams—and to describe its functionality and use cases. Implementation may begin once the initial documentation has been submitted and approved. Assessment criteria include: sufficient functional scope (2–3 use cases per student for each stage), sound architectural decisions, and clear explanations of those decisions (including UML and other diagrams). Any changes to the functionality outlined in the initial plan require the lecturer's approval.

Development of the selected information system architecture	40	In four steps, 4th - 14th weeks of the semester.	During the semester, each student must implement a software system (like information system, videogame, mobile app, etc), presenting and defending the developed solution. It is required correct operation of the implemented system, full explanation on the principles of its operation, to answer questions related to the task (about concepts, algorithms). Evaluation criteria: correctness of algorithm operation, programming effectiveness, correct and detailed conclusions, correctness, consistency and clarity of answering, timeliness of the presentation, ability to fix issues pointed out during presentation, ability to use the infrastructure provided by the university if necessary. Maximum grade form 1st part of the task is 4 points; 2nd part is evaluated by up to 1 point. Ethical use of artificial intelligence methods is permitted, but those using them may be subject to increased task requirements to maintain equal task complexity for all students. The use of artificial intelligence must be ethical and clearly declared during the presentation of iterations and in the final documentation.
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, answering the question. Only students who have at least partially completed the practical assignment (having earned ≥ 0.5 points from the practical work) are eligible to take the exam.
Extra credit	0-20	During semester or exam session.	Students who have performed the task particularly well may be awarded additional points for the quality of the system created, the additional functionality implemented, and the particularly well-described documentation.
Extern assessment of this subject is not possible.			

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.
Martin Fowler	2003	Patterns of Enterprise Application Architecture	ISBN-13: 978-0321127426	Boston, Addison-Wesley.
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.
Optional reading				
Robert C. Martin	2009	Clean Code: A Handbook of Agile Software Craftsmanship	ISBN-13: 9780132350884	Upper Saddle River, NJ, Prentice Hall.
Craig Larman	2011	Applying UML and patterns :an introduction to object-oriented analysis and design and		Upper Saddle River, NJ, Prentice Hall.

		iterative development		
Martin Fowler	2003 3rd edition	UML Distilled: A Brief Guide to the Standard Object Modeling Language		Addison-Wesley