Practical seminar on mathematical methods in software engineering

COURSE SYLLABUS

Code and name of specialty	121 Software Engineering	**Institute / faculty**	Faculty of Computer Science and Software Engineering
Program name	“Software Engineering”	**Department**	Software Engineering and Management Information Technologies
Type of program	Educational and Professional	**Language of instruction**	Ukrainian, English

LECTURER

Name and surname, email **Nataliia Khatsko, *nataliia.khatsko@khpi.edu.ua***

Ph.D., associate professor at the Department of Software Engineering and Management Information Technologies of NTU «KhPI», associate professor. Prepared and published more than 40 research papers and textbooks (SCOPUS Author ID https://www.scopus.com/authid/detail.uri?authorId=57200820629; Researcher ID https://app.webofknowledge.com/author/#/record/17252627; GoogleScholar https://scholar.google.com.ua/citations?user=US7Ovx4AAAAJ&hl=uk; ORCID orcid.org/0000-0002-2543-0280)

Basic courses: "Computer Mathematics (parts 1, 2, 3)", "Practical seminar on mathematical methods in software engineering", "Formal methods of software verification", "Formal methods of software systems research"

GENERAL DESCRIPTION OF THE COURSE

Summary	The course "Practical seminar on mathematical methods in software engineering" is a discipline in the cycle of professional training in the specialty 121 "Software Engineering". It is taught in the sixth semester in the amount of 150 hours (5 ECTS credits), in particular: practical classes - 32 hours, independent work - 118 hours. The course provides for the calculation work. The course ends with the report on the results of individual calculation work and credit.
Course objectives	Formation of students' practical knowledge and skills necessary for the application of mathematical methods in the design of software systems. Acquisition of practical skills in the use of formal methods and models of discrete mathematics in the processing of discrete information and the description of discrete processes associated with software development.
Types of classes and control	Mini-lectures, practical classes (workshops), seminars, self-study, calculation work. Final control - credit.
Term	6

Student workload (credits) / Type of course	5 / Mandatory	**Lectures (hours)**	-	**Workshops (hours)**	32	**Self-study (hours)**	118

Program competences	GC01. Ability to abstract thinking, analysis and synthesis. GC 02. Ability to apply knowledge in practical situations. GC 03. Ability to communicate in the state language both orally and in written form. GC05. Ability to learn and master modern knowledge. GC06. Ability to search, process and analyze information from various sources. PC22. Ability to accumulate, process and systematize professional knowledge on the creation and maintenance of software and recognition of the

	importance of lifelong learning. PC26. Ability to algorithmic and logical thinking.

Learning outcomes	Teaching and learning methods	Forms of assessment (continuous assessment CAS, final assessment FAS)
PLO01. Analyze, purposefully search for and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology. PO11. Choose source data for design, guided by formal methods of describing requirements and modelling.	Mini-lectures, practical classes in the form of seminars on relevant topics, individual tasks in the form of course work, self-work with literary sources.	**Continuous assessment CAS**: Student's assessment during practical classes. Interim evaluation of the implementation of the components of the calculation work. **Final assessment FAS**: credit

ASSESSMENT AND GRADING

	core (points) for all types of learning activities	ECTS grading scale	The national grading scale
Ranges of points corresponding to grades	90-100	A	excellent
	82-89	B	good
	74-81	C	
	64-73	D	satisfactory
	60-63	E	
	35-59	FX	Unsatisfactory (with the exam retake option)
	0-34	F	Unsatisfactory (with mandatory repetition of the course)

Allocation of grade points

100% final evaluation
as a result of final credit (30%) and current evaluation 70%).
30% credit:…
70% current rating:
assessment of tasks on practical works

Workshop 1-2	10%
Workshop 2-5	10%
Workshop 6-7	10%
Workshop 8-9	10%
Workshop 10-11	10%
Workshop 12-13	10%
Workshop 14-15	10%

Course policy	Students must attend all classes according to the study schedule and adhere to the norms of academic ethics. Students must work with compulsory and recommended reading, including Internet resources. Students must complete and submit all practice works (workshops) during the semester in which the course is taught, before the examination session. The final assessment is not carried out without the personal presence of students.

COURSE STRUCTURE AND CONTENT

		Workshop 1-2	Analysis of the initial description of the software system planned	**Self-study**	Choose the topic of an individual calculation work, in which mathematical research methods will be used. Research of publications in scientific journals, materials of technical conferences;

			for development. The example of the use of a finite state machine in preparation for software development.		in specialized IT-publics
		Workshop 2-5	Drawing up a protocol of the software system. Mathematical description of a finite automaton.		Analyze whether the previously chosen topic was solved with the help of software implementation. Development of protocols for response of entities to system events. Selection of structural effective entities, definition of system events and their initiators. Creating a mathematical description of a finite automaton for each entity (system decomposition)
		Workshop 6-7	Analysis of the finite automaton of each essence of the system.		Analysis of the subject area of the work in relation to the selected entities. Analysis of the finite automaton of each essence of the system. Making changes to work protocols as needed.
		Workshop 8-9	The automat-product as a description of the system as a whole.		Analysis of the subject area of the work for possible reactions to events in the system. Making a connection of finite automata of individual entities. Analysis of possible joint reactions when considering the automaton-product. Making changes to the protocols as needed.
		Workshop 10-11	Description of the language allowed by each of the received machines.		Identification of possible chains in the work of finite automata of the system. Analysis of compliance of the obtained chains with the system protocols.
		Workshop 12-13	Regular expressions. Tables for constructing regular expressions		Get regular expressions about the language of finite automata using tables to build regular expressions.
		Workshop 14-15	Construction of grammars.		Construction of grammars.
		Workshop 16	Report on the results of individual calculation work		Execution of calculation work.

RECOMMENDED READING

Compulsory	1. Ajit Singh. (2019). Formal Language And Automata Theory. LAMBERT Academic Publishing. 2. Abejide Ade-Ibijola. (2017). New Finite Automata Applications in Novice Program Comprehension. LAP LAMBERT Academic Publishing. 3. Neeru Gupta. (2020). Beginner's Guide-Automata Theory. 4. Ezhilarasu Umadevi Palani. (2019).Finite Automata Problems & Solutions. LAP Lambert Academic Publishing. 5. Stoyan Mihov, Klaus, U., Schulz. (2019). Finite-State Techniques: Automata, Transducers and Bimachines. Cambridge University Press. 6. Hopcroft, John E., Motwani, Rajeev, Ullman, Jeffrey D. (2014). Introduction to automata theory, languages, and computation. Boston: Pearson Education, Inc. 7. Хаггарти, Р. (2017). Дискретная математика для программистов: учебное пособие. Москва: Техносфера.	**Recommended**	1. Michael Sipser. (2006). Introduction to the Theory of Computation. Thomson. 2. Daniel, I. A., Cohen, John Wiley. (1996). Introduction to Computer Theory. 3. John C Martin. Introduction to languages and the Theory of Computation. 4. Lewis, H. P., Papadimition, C. H. Elements of Theory of Computation. Pearson. 5. Mishra, Chandrashekaran. Theory of Computer Science and Automata languages and computation. (2nd ed.). 6. Foster, E. C. Software Engineering : A Methodical Approach. New York: Apress. 7. J. Richard Büchi. (1989). Finite Automata, Their Algebras and Grammars. Towards a Theory of Formal Expressions. 8. Goswami, D., Krishna K. V. (2010). Formal Languages and Automata Theory. 9. Завалишин, Е. П. (2007). Логика : учебное пособие для вузов. Тула: Изд-во ТулГУ. 10. Gerda Ivanickienė. The theoretical material and exercises INFORMATION RESOURCES ON THE INTERNET 1. Goswami, D., Krishna K. V. Formal Languages and Automata Theory. Retrieved from http://www.iitg.ernet.in/dgoswami/Flat-Notes.pdf 2. Introduction to Automata and Complexity Theory. Retrieved from http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html 3. Formal Languages and Automata Theory. Retrieved from http://cs.fit.edu/~dmitra/FormaLang.

Academic integrity

Graduate students are expected to adhere to the Code of Ethics of Academic Relations and Integrity" of NTU "KhPI".

The content of this syllabus is consistent with the course program.