

**ІНСТИТУТ КІБЕРНЕТИКИ  
ІМЕНІ В.М. ГЛУШКОВА НАН УКРАЇНИ**

«ЗАТВЕРДЖУЮ»

Директор Інституту кібернетики  
імені В.М. Глушкова НАН України  
академік НАН України



*Г.В. Сергієнко*  
Г.В. Сергієнко  
07 2020 року

**РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ  
ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯ  
(ОНД.08)**

**для здобувачів освітньо-наукового рівня «доктор філософії»**

галузь знань  
спеціальність  
освітній рівень  
освітньо-наукова програма  
вид дисципліни

12 «Інформаційні технології»  
112 «Комп'ютерні науки»  
третій (освітньо-науковий)  
«Комп'ютерні науки»  
обов'язкова

Форма навчання	денна / заочна
Навчальний рік	2020/2021
Рік навчання	2
Кількість кредитів ECTS	3
Мова викладання, навчання та оцінювання	українська
Форма заключного контролю	екзамен

Викладач: **Летичевський Олександр Олександрович**, д.ф.-м.н., с.н.с.

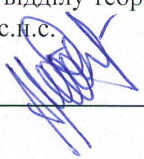
Пролонговано Вченою радою Інституту кібернетики імені В.В. Глушкова НАН України

Навчальні роки продовження	Голова вченої ради	Підпис	№ протоколу	Дата протоколу
20___/20___ р.	_____	_____	_____	_____
20___/20___ р.	_____	_____	_____	_____
20___/20___ р.	_____	_____	_____	_____
20___/20___ р.	_____	_____	_____	_____

**КИЇВ – 2020**

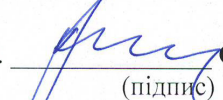
**РОЗРОБНИК:**

Завідувач відділу теорії цифрових автоматів,  
д.ф.-м.н., с.п.с.

  
Летичевський Олександр Олександрович

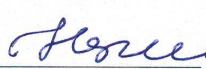
Робочу програму розглянуто та схвалено на засіданні відділу мікропроцесорної техніки

Протокол від "02" 07 20 20 року № 4

Завідувач відділу  
академік НАН України, д.т.н.  О.В. Палагін  
(підпис)

Робочу програму ухвалено науково-методичною радою

Протокол від "15" 07 20 20 року № 3

Голова науково-методичної ради  
академік НАН України  І.В. Сергієнко  
(підпис)

Робочу програму затверджено Вченою радою Інституту кібернетики імені  
В.М. Глушкова НАН України

Протокол від "28" 07 20 20 року № 13

Робочу програму погоджено з гарантом освітньої програми 122 «Комп'ютерні науки»

"15" 07 20 20 року

Гарант освітньої програми  
академік НАН України  О.В. Палагін  
(підпис)

**1. Мета дисципліни** спрямована на оволодіння сучасними методами аналізу та синтезу розподілених багатоагентних систем, а також сучасними методами розробки засобів верифікації вимог та тестування для розподілених, паралельних та недетермінованих систем. На базі набутих знань майбутні фахівці зможуть ефективно використовувати прикладні системи верифікації вимог та специфікацій розподілених взаємодіючих систем. Зокрема для генерації тестів для тестування систем з великою або нескінченною кількістю станів та систем реального часу з досяжністю 100 % покриття програмного коду тестами..

**2. Попередні вимоги до опанування або вибору навчальної дисципліни:**

- 1. Знати:** моделі і методи теорії алгоритмів та математичної логіки, теорії програмування, теорії алгоритмічних систем, теорії формальних мов та графік, методів алгебраїчного програмування.
- 2. Вміти:** виконувати побудову специфікацій вимог для програмних або апаратних систем, проводити аналіз коректності, обирати ефективні моделі залежно від вибраних вимог, застосовувати методи перевірки виконаності формул розв'язних теорій, розробляти паралельні та розподілені алгоритми.

**3. Анотація навчальної дисципліни:**

Дисципліна забезпечує аспіранта знаннями та технологією використання формальних методів в процесі розробки програмного та апаратного забезпечення.

Завдання курсу – ознайомитись із існуючими технологіями використання формальних методів у верифікації та тестуванні, застосовувати методи формальної верифікації та модельного тестування в процес розробки систем, що критичні до безпеки.

**4. Завдання (навчальні цілі):** набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень у області формальних методів у верифікації та тестуванні програмних систем, що критичні до безпеки, відповідно до науково-освітньої кваліфікації «Доктор філософії з комп'ютерних наук». Зокрема, розвивати здатність формалізувати, використовувати методи символного моделювання, генерувати та реалізовувати нові конкурентоздатні ідеї в галузі формальних методів, здатність критично переосмислювати наявні методи та відстежувати тенденції їх розвитку.

**5. Результати навчання за дисципліною:**

Результат навчання (РН) (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		Форми (та/або методи і технології) викладання і навчання	Методи оцінювання та пороговий критерій оцінювання (за необхідності)	Відсоток у підсумковій оцінці з дисципліни
Код	Результат навчання			
РН 1.1	Знати та розуміти основних задачі формальної верифікації та модельного тестування	<i>Лекції</i>	<i>Екзамен, усні відповіді</i>	<i>30%</i>
РН 1.2	Знати основні сучасні формальні методи в області символного моделювання, використання автоматичного доведення та програм-розв'язувачів в задачах верифікації та модельного тестування.			
РН 1.3	Знати перелік сучасних програмних комплексів для розв'язання основних задач верифікації та тестування			
РН 2.1	Вміти застосовувати методи формалізації програмних та апаратних специфікацій на різних рівнях абстракції.	<i>Лекції, самостійна робота</i>	<i>Екзамен</i>	<i>10%</i>
РН 2.2	Вміти застосовувати сучасні програмні комплекси, що базуються на формальних методах, для верифікації програмних та апаратних систем, що критичні до безпеки.	<i>Самостійна робота</i>	<i>Активна робота на лекції</i>	<i>20%</i>

PH 2.3	Вміти застосовувати методи модельної розробки та модельного тестування.		Екзамен	20%
PH3.1	Обґрунтовувати власний погляд на задачу та вміти доносити свою думку до інших	Лекції	усні відповіді на лекціях	5%
PH4.1	Демонстрація авторитетності, інноваційність, високий ступінь самостійності, академічна та професійна доброчесність, послідовна відданість розвитку нових ідей або процесів у передових контекстах професійної та наукової діяльності.			5%
PH4.2	Відповідально ставитися до виконуваних робіт, нести відповідальність за їх якість	Самостійна робота		10%

## 6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Програмні результати навчання (з опису освітньої програми)	Результати навчання дисципліни									
	PH 1.1	PH 1.2	PH 1.3	PH 2.1	PH 2.2	PH 2.3	PH 3.1	PH 4.1	PH 4.2	
<b>ПРН-1.</b> Мати передові концептуальні та методологічні знання з комп'ютерних наук і на межі предметних галузей, а також дослідницькі навички, достатні для проведення наукових і прикладних досліджень на рівні останніх світових досягнень з відповідного напрямку, отримання нових знань та/або здійснення інновацій.		+		+						+
<b>ПРН-3.</b> Формулювати і перевіряти гіпотези; використовувати для обґрунтування висновків належні докази, зокрема, результати теоретичного аналізу, експериментальних досліджень (опитувань, спостережень, ...) і математичного та/або комп'ютерного моделювання, наявні літературні дані.	+				+					
<b>ПРН-4.</b> Розробляти та досліджувати концептуальні, математичні і комп'ютерні моделі процесів і систем, ефективно використовувати їх для отримання нових знань та/або створення інноваційних продуктів у комп'ютерній науці та дотичних міждисциплінарних напрямках.			+					+		
<b>ПРН-6.</b> Застосовувати сучасні інструменти і технології пошуку, оброблення та аналізу інформації, зокрема, статистичні методи аналізу даних великого обсягу та/або складної структури, спеціалізовані бази даних та інформаційні системи.					+					
<b>ПРН-7.</b> Розробляти та реалізовувати наукові та/або інноваційні інженерні проекти, які дають можливість переосмислити наявне та створити нове цілісне знання та/або професійну практику і розв'язувати значущі наукові та технологічні проблеми комп'ютерної науки з дотриманням норм академічної етики і врахуванням соціальних, економічних, екологічних та правових аспектів.				+						+
<b>ПРН-8.</b> Глибоко розуміти загальні принципи та методи комп'ютерних наук, а також методологію наукових досліджень, застосувати їх у власних дослідженнях у сфері комп'ютерних наук та у викладацькій практиці.	+					+				
<b>ПРН-9.</b> Вивчати, узагальнювати та впроваджувати в навчальний процес інновації комп'ютерних наук.							+			

<b>ПРН-10.</b> Здійснювати пошук та критичний аналіз інформації, концептуалізацію та реалізацію наукових проєктів з комп'ютерних наук.		+							+	
--	--	---	--	--	--	--	--	--	---	--

## 7. Схема формування оцінки.

### 7.1. Форми оцінювання здобувачів освітньо-наукового ступеня:

#### - оцінювання впродовж навчального періоду:

1. Активна робота на лекції, усні відповіді: РН 1.1, РН 1.2, РН 1.3, РН 2.1 – 10 балів;
2. Робота на практичних заняттях: РН 2.2, РН 2.3 – 30 балів;
3. Виконання завдань, винесених на самостійну роботу: РН 2.2, РН 2.3, РН 4.2 – 20 балів;

#### - підсумкове оцінювання: екзамен.

- максимальна кількість балів, які можуть бути отримані: 40 балів;
- результати навчання, які будуть оцінюватись: РН 1.2, РН 1.3, РН 2.1, РН 2.3;
- форма проведення і види завдань: письмова робота.

Для здобувачів освітньо-наукового ступеня, які набрали сумарно меншу кількість балів, ніж критично-розрахунковий мінімум – 20 балів для одержання іспиту, за рішенням відділу не допустити до складання іспиту.

Рекомендований мінімум – 36 балів.

### 7.2. Організація оцінювання:

У частину 1 входять теми 1-2, у частину 2 – теми 3-6 у частину 3 – теми 7-9. Обов'язковим є виконання завдань, винесених на самостійну роботу.

#### Терміни проведення форм оцінювання:

1. Активна робота на лекції, усні відповіді: протягом навчального періоду;
2. Робота на практичних заняттях: протягом навчального періоду;
3. Виконання завдань, винесених на самостійну роботу: протягом навчального періоду.

### 7.3. Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

## 8. СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ. ТЕМАТИЧНИЙ ПЛАН ЛЕКЦІЙ І ПРАКТИЧНИХ ЗАНЯТЬ

№	Назва лекції	Кількість годин		
		Лекції	Практ.	Самост. робота
<b>Частина 1. „Огляд сучасних методів формальної верифікації та тестування”</b>				
1	<b>Тема 1. Формальні методи та сучасні системи верифікації в розробці програмного забезпечення.</b> Поняття верифікації. Різновиди систем верифікації та їх місце в життєвому циклі процесу розробки програмного забезпечення. Системи верифікації	2	5	2

	<p>вимог та дизайну. Мови специфікацій формальних вимог. Системи верифікації дизайну. Мови UCM, SysML. Система верифікації програмного коду.</p> <p><i>Практична робота:</i> Встановити одну із пробних версій систем верифікації та запустити на існуючих еталонних тестах.</p>			
2	<p><b>Тема 2. Сучасні методи модельного тестування.</b></p> <p>Основні поняття модельного тестування. Різновиди систем модельного тестування. Покриття в модельному тестуванні. Формальні мови для представлення тестових моделей. Методи та системи генерування тестових сценаріїв. Мови TDL, TTCN. Системи автоматичного виконання тестів.</p> <p><i>Практична робота:</i> Встановити одну із пробних версій систем модельного тестування та запустити на існуючих еталонних тестах. Спробувати досягти 100% покриття змінюючи параметри системи.</p>	2	5	2
<b>Частина 2. „Формальні методи верифікації та модельного тестування”</b>				
3	<p><b>Тема 3. Інсерційне моделювання.</b></p> <p>Агенти та середовища та їх взаємодія. Типи агентів та їх атрибутів. Функція занурення. Багаторівневі середовища. Розподілені середовища. Інсерційне моделювання. Система інсерційного моделювання. Приклади задач верифікації в системі інсерційного моделювання.</p> <p><i>Практична робота:</i> Зробити приклад моделі в середовищі інсерційного моделювання.</p>	2	5	3
4	<p><b>Тема 4. Алгебра поведінок</b></p> <p>Поведінка та дії агентів. Алгебра поведінок. Перед-та після умови в діях агентів. Формалізація поведінки агентів. Приклади формалізації поведінки в програмних та апаратних системах. Формалізація властивостей безпеки в системах. Задача досяжності властивості, як задача верифікації.</p> <p><i>Самостійна робота:</i> На прикладі інсерційної моделі формалізувати властивості безпеки та перевірити її досяжність за допомогою системи інсерційного моделювання.</p>	2		8
5	<p><b>Тема 5. Символьне моделювання та статична верифікація.</b></p> <p>Задача виконаності умови. Машини доведення та системи-розв'язувачі. Автоматичне доведення властивостей в формальних системах. Повнота та суперечливість формальних специфікаціях. Предикатні перетворювачі. Пряме та обернене символьне моделювання. Методи генерації інваріантів.</p> <p><i>Практична робота:</i> На прикладі створеної інсерційної моделі довести властивості повноти та суперечливості в рамках системи інсерційного моделювання.</p>	2	5	3

6	<p><b>Тема 6. Формалізація та верифікація розподілених систем.</b></p> <p>Розподілені системи та їх приклади. Формалізація розподілених систем на платформі інсерційного моделювання. Блокчейн, як розподілена система. Формалізація протоколів блокчейн систем. Тестова модель блокчейн системи. Генерація сценаріїв та моделювання блокчейн систем. Тестування розподілених систем, зокрема блокчейн платформ. <i>Практична робота:</i> Зробити формалізацію простого блокчейн протоколу, згенерувати тестові сценарії.</p>	2	2,5	5,5
<b>Частина 3. „Верифікація та тестування в процесі модельної розробки програмного забезпечення”</b>				
7	<p><b>Тема 7. Модельна розробка програмного та апаратного забезпечення.</b></p> <p>Поняття модельної розробки. Загальна схема життєвого циклу модельної розробки. Особливості розробки систем, що критичні до безпеки. Етап збору та верифікації вимог. Формальна верифікація та модельне тестування вимог для високонадійних програмних систем. Автоматична генерація системних тестів та специфікацій дизайну. <i>Самостійна робота:</i> Формалізувати в мові алгебри поведінок систему вимог для алгоритму функціонування ліфту. Сформулювати властивості безпеки.</p>	2		8
8	<p><b>Тема 8. Верифікація програмної архітектури та програмного коду.</b></p> <p>Алгебраїчний підхід у верифікації архітектури. Верифікація мов SysML, UCM. Тестування архітектури та генерація інтеграційних тестів. Автоматична генерація коду. Верифікація програмного коду в імперативних та об'єктно-орієнтовних мовах. Методи пошуку вразливостей в програмному коді. <i>Самостійна робота:</i> Використовуючи пробну версію програми пошуку вразливостей (або існуючу платформу розробки програми) провести пошук вразливості «переповнення буферу».</p>	2		8
9	<p><b>Тема 9. Алгебраїчний підхід в модельному тестуванні програмного коду.</b></p> <p>Алгебраїчна модель генерації тестів. Методи модельного тестування із використанням алгебраїчного підходу. Методи чорної та білої скриньки. Символьне виконання тестів. Символьне тестування онлайн. Інтеграційне та регресивне тестування. Мінімізація тестового набору та особливості модельного тестування розподілених систем. <i>Практична робота:</i> Створити модель для генерації тестів однієї із властивостей телекомунікаційного</p>	2	5	3

	протоколу. Проект з формалізацією системи вимог, верифікацією та генерацією тестових сценаріїв.			
	Консультація (2 год)			
	ВСЬОГО	18	27,5	42,5

**Загальний обсяг 90 годин**, в тому числі:

Лекцій – **18 годин**,

Практичних занять – **27,5 годин**,

Консультації - **2 години**,

Самостійна робота – **42,5 годин**.

## 9. Рекомендовані джерела

1. Математична логіка та теорія алгоритмів, підручник, Нікітченко М.С., Шкільняк С.С. К.- Київський університет. – 2008
2. Скінчені автомати: теорія алгоритми складність, підручник, Кривий С.Л. К. -2020
3. Основи дискретної математики, підручник, т.1 -2, Ю.Капітонова,О.Летичевський, С.Кривий, Г.Луцький, М. Печурін, К. 2000,
4. Paulk Mark C. / (February 1993) / Capability Maturity Model for Software (Version 1.1) / Weber Charles V., Curtis Bill, Chrissis Mary Beth // Technical Report (Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University)
5. ISO/IEC 13568:2002". Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics // ISO. – 2002. – 196 pp.
6. Bjørner Dines / The Vienna Development Method: The Meta-Language / Cliff B. Jones // Lecture Notes in Computer Science 61. Berlin, Heidelberg, New York: Springer. – 1978.
7. [Gerard Holzmann](#) / The SPIN Model Checker: Primer and Reference Manual // Addison-Wesley – 2004. – 585 pp.
8. Крэг Ларман / Применение UML 2.0 и шаблонов проектирования (Applying UML 2.0 and Patterns) An Introduction to Object-Oriented Analysis and Design and Iterative Development // — 3-е изд. — М.: Вильямс, 2006. — 736 с.
9. Tim Weilkiens / Systems Engineering with SysML/UML: Modeling, Analysis, Design // The OMG Press. –2008.
10. ITU-T Recommendation, Z.151, User Requirements Notation (URN) – Language definition
11. ITU-T Recommendation, Z.120, Message Sequence Charts (MSC)
12. SPIN Homepage, <http://spinroot.com/spin/whatispin.html>
13. Rik Eshuis / Symbolic model checking of UML activity diagrams // ACM Trans. Softw. Eng. Methodol, №15(1). – January 2006. – pp. 1-38.
14. Islam Abdelhalim / Formal verification of tokeneer behaviours modelled in fUML using CSP/ James Sharp, Steve Schneider, and Helen Treharne // In JinSong Dong and Huibiao Zhu, editors, Formal Methods and Software Engineering, volume 6447 of LNCS, Springer Berlin Heidelberg. – 2010. – pp. 371-387.
15. Helle Hvid Hansen / Automated verification of executable UML models / Jeroen Ketema, Bas Luttik, MohammadReza Mousavi, Jaco van de Pol, and Osmar Marchi dos Santos / Proceedings of the 9th International Conference on Formal Methods for Components and Objects, FMCO'10, Berlin, Heidelberg, Springer-Verlag. – 2011. – pp. 225-250.
16. Stefan Blom / Distributed and Symbolic Reachability / Jaco Van De Pol, Michael Weber // Proceedings Computer Aided Verification (CAV2). –2010. –pp. 354-359.

17. Yael Meller / Verifying Behavior UML Systems via CEGAR / Orna Grunberg, Karen Yorav // Proceedings Integrated Formal Methods (IFM). – 2014. – pp.139-154.
18. J. Peleska / Industrial-Strength Model-Based Testing - State of the Art and Current Challenges // Proceedings Eighth Workshop on Model-Based Testing, DOI: 10.4204/EPTCS.111.1. – 2013. – pp. 3-28.
19. Patrice Godefroid / SAGE: Whitebox Fuzzing for Security Testing / Michael Y. Levin, David Molnar // Magazine Queue – Networks, Vol. 10, Issue 1, Jan. – 2012.
20. Willem Visser / Model Checking Programs with Java PathFinder / Peter Mehlitz // Lecture Notes in Computer Science, Volume 3639. – 2005. – p. 27.
21. Guodong Li / KLOVER: A Symbolic Execution and Automatic Test Generation Tool for C++ Programs / Indradeep Ghosh, Sreeranga P. Rajan // Lecture Notes in Computer Science, Volume 6806. – 2011. – pp. 609-615.
22. S. Ganov / Test Generation for Graphical User Interfaces Based on Symbolic Execution / C. Killmar, S. Khurzid, Dewayne E Perry // Proceedings ISCE Workshop. – 2008.
23. Alexandre Petrenko / Integration Testing of Communication Systems with Unknown Components / Ronald Groz, Keqin Li // Annuals of Communications, vol. 70 , №3-4. – pp 107-125.
24. Gilbert D. / A general theory of action languages / Letichevsky A. // Cybernetics and System Analysis. – 1998. – № 1. – pp. 16–37.
25. Gilbert D. / A Model for Interaction of Agents and Environments / Letichevsky A. // Lecture Notes in Computer Science. – 1999. – № 1827. – pp. 311–328.
26. Letychevskiy O. / Predicate transformers and system verification / Letichevsky A. // Proc. Third International Workshop on Symbolic Computation in Software Science (SCSS 2010), Hagenberg, Austria. – 2010. A. Voronkov, L. Kovacs and N. Bjorner (eds). EPiC Series, Vol.1. –2012. –P. 148–149.
27. Letychevskiy O. / Insertion Modeling System / Letichevsky A., Peschanenko V. // Lecture Notes in Computer Science : [Revised selected papers / S.D.J. Brabosa etc]. – Heidelberg : Springer, 2011. – № 7162. – P. 262–274