ІНСТИТУТ КІБЕРНЕТИКИ
ІМЕНІ В.М.ГЛУШКОВА НАН УКРАЇНИ

«ЗАТВЕРДЖУЮ»
Директор Інституту кібернетики
імені В.М. Глушкова НАН України
академік НАН України

_____ ІванСЕРГІЄНКО

«29» 09 2025 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
ФОРМАЛЬНА ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
(ДВА.3.1.6)
для здобувачів освітньо-наукового рівня «доктор філософії»

галузь знань	**F «Інформаційні технології»**
спеціальність	**F1«Прикладна математика»**
освітній рівень	**третій (освітньо-науковий)**
освітньо-наукова програма	**«Прикладна математика»**
вид дисципліни	**вибіркова**

Форма навчання	**денна / заочна**
Навчальний рік	**2025/2026**
Рік навчання	2
Кількість кредитів ECTS	2
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**екзамен**

Викладачі: **Летичевський Олександр Олександрович,** д.ф.-м.н., с.н.с.
Колчин Олександр Валентинович, к.ф.-м.н., с.н.с.

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

Навчальні роки пролонгації	Голова вченої ради	Підпис	№ протоколу	Дата протоколу
20____/20____ р.	_____	_____	_____	_____
20____/20____ р.	_____	_____	_____	_____
20____/20____ р.	_____	_____	_____	_____
20____/20____ р.	_____	_____	_____	_____

КИЇВ – 2025

РОЗРОБНИКИ:

Летичевський Олександр Олександрович, д.ф.-м.н., с.н.с., зав. відділу
Колчин Олександр Валентинович, , к.ф.-м.н., с.н.с.

Робочу програму розглянуто та схвалено на засіданні відділу чисельних методів та комп'ютерного моделювання

Протокол від "19" вересня 2025 року № 9

Завідувач відділу
академік НАН України, д.ф.-м.н. _____ **О.М. Хіміч**
(підпис)

Робочу програму ухвалено науково-методичною радою

Протокол від "22" вересня 2025 року № 2

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 "29" вересня 2025 року № 15

Учений секретар _____ **А.І. Куляс**
(підпис)

Робочу програму погоджено з гарантом освітньої програми F1Прикладна математика
"29" вересня 2025 року

Гарант освітньої програми
доктор фізико-математичних наук _____ **В.М. Горбачук**
(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 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 граматик, методів алгебраїчного програмування та методи тестування базовані на моделі.
2. *Вміти*: виконувати побудову специфікацій вимог для програмних або апаратних систем, проводити аналіз коректності, обирати ефективні моделі залежно від вибраних вимог, застосовувати методи перевірки виконуваності формул розв'язних теорій, застосовувати критерії тестового покриття та аналізувати ефективність тестових наборів.

3. Анотація навчальної дисципіни:

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 символьного 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 основні задачі побудови абстракцій,формальної верифікації, та модельного тестування	*Лекції*	*Екзамен, усні відповіді*	*30%*
РН 1.2	Знати основні сучасні формальні методи в області символьного 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 нарізних рівнях абстракції.	*Лекції, самостійна робота*	*Екзамен*	*10%*
РН 2.2	Вміти застосовувати сучасні програмні комплекси, що базуються на формальних методах, для верифікації програмних та апаратних систем, що	*Самостійна робота*	*Активна робота на лекції*	*20%*

	критичні до безпеки.			
РН 2.3	Вміти застосовувати методи модельної розробки та модельного тестування.		*Екзамен*	*20%*
РН3.1	Обґрунтовувати власний погляд на задачу та вміти доносити свою думку до інших	*Лекції*	*усні відповіді на лекціях*	*5%*
РН4.1	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%*
РН4.2	Відповідально ставитися до виконуваних робіт, нести відповідальність за їх якість	*Самостійна робота*		*10%*

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Результати навчання дисципліни **Програмні результати навчання**	**РН 1.1**	**РН 1.2**	**РН 1.3**	**РН 2.1**	**РН 2.2**	**РН 2.3**	**РН 3.1**	**РН 4.1**	**РН 4.2**
(з опису освітньої програми)									
ПРН-1. 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 напряму, отримання нових знань та/або здійснення інновацій.		+		+					+
ПРН-3. 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.	+				+				
ПРН-4. 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 напрямах.			+					+	
ПРН-6. 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.					+				
ПРН-7. Розробляти та реалізовувати наукові та/або інноваційні інженерні проєкти, 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.				+					+
ПРН-8. 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.	+					+			
ПРН-9. Вивчати, узагальнювати та впроваджувати в навчальний процес інновації комп'ютерних наук.							+		

ПРН-10. 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**
Частина 1. „Формальні методи перевірки властивостей програмних систем"				
1	**Тема 1. Формальні методи та сучасні системи верифікації в розробці програмного забезпечення.** Особливості розробки систем, що критичні до безпеки.Поняття формальної верифікації. Теоретичні основи: трійки Хоара, аксіоми висновку, специфікації властивостей. Різновиди систем та методів верифікації та їх місце в життєвому циклі процесу розробки програмного забезпечення. Адекватність абстракції моделі. Системи верифікації вимог та дизайну. Мови специфікацій формальних вимог. Системи верифікації дизайну. Мови UCM,SySML,Promela.	2		4

	Практична робота: Встановити одну із пробних версій систем верифікації та запустити на існуючих еталонних тестах.Побудувати невелику формальну модель (наприклад, ліфт, кава-машина, FIFO).			
2	**Тема 2.Властивості вимог. Проблеми статичного та динамічного аналізу.** Основні властивості вимог:атомарність, незалежність, повнота, коректність, узгодженість. Збір та аналіз вимог до систем.Статичні та динамічні методи перевірки властивостей. Проблеми великої кількості станів та алгоритмічної нерозв'язності. *Практична робота:*Розробити маленькі приклади для демонстрації властивостейconsistency, completeness, racing.	2		4
3	**Тема 3.Темпоральні логіки. Бінарні діаграми рішень.** Модальні твердження, темпоральні оператори, формули лінійної темпоральної логіки та логіки з розгалудженим часом. Структура Кріпке. Канонічні форми запису, бінарні діаграми прийняття рішень. Метод розкладання Шеннона. *Практична робота:*Для заданих прикладів записати властивості у вигляді LTL-формул та створити BDD діаграми методом розкладання Шеннона.	2		4
4	**Тема 4. Ефект комбінаторного вибуху та методи редукції простору пошуку.** Досяжність у тразиційних системах. Комбінаторне зростання кількості шляхів та станів. Задача досяжності властивості, як задача верифікації. Недетермізізм, інтерлівінг. Методи редукції: абстракції предикатів, симетрії, часткового порядку, символьна перевірка моделі. Уточнення абстракції кероване контр-прикладом.Дедуктивні методи: доведення теорем, інваріанти. Інтерполянт Крейга. Евристики. *Самостійна робота:* Для заданого прикладу знайти інтерполянт; провести експеримент з ефектом комбінаторного вибуху на існуючих інструментальних засобах.	2		4
5	**Тема 5.Формальні методи доведення властивостей.** Символьне моделювання та статична верифікація. Інваріанти циклів. Алгоритм Девіса-Путнема. Метод обмеженої перевірки моделей. Задача виконуваності умови. Машини доведення та системи-розв'язувачі. Автоматичне доведення властивостей в формальних системах. *Практичнаробота:* Інваріанти: написати маленьку програму з циклом, додати анотації та знайти будь-який нетривіальний інваріант циклу. Символьне виконання: написати маленьку програму та описати її стани методом символьного моделювання.	2		4
Частина 2.„Тестування програмних систем на основі формальних моделей"				
6	**Тема 6. Методи тестування в модельній розробці програмних систем.** Методи і різновиди тестування. Тестування базоване на моделі. Критерії тестового покриття. Методи автоматичної генерації тестових даних. Мінімізація тестового набору. Поширені види помилок, вразливостей та мутаційний підхід. Мови TDL, TTCN. Системи автоматичного виконання тестів. *Практична робота:* Для заданого прикладу: (1) забезпечити покриття програми за критерієм MC/DC, (2) створити 3 мутації різних типів та розробити тести для їх викриття, (3) забезпечити покриття граничних значень.	2		6
7	**Тема 7. Методи тестування потоку даних. SSA-форма. Потік управління, залежність даних.** Проблеми ефективності структурних критеріїв покриття. Покриття потоку даних: пари визначення-використання. Форма статичного єдиного призначення. Граф залежності потоку управління. Границі домінування і алгоритми пошуку домінаторів. Прямі і обернені зрізи програм.	2		4

	Самостійна робота: Для заданого прикладу програми: (1) побудувати SSA-форму, (2) побудувати обернений зріз, (3) знайти пари визначень-використань, (4) побудувати тестовий набір для покриття пар визначень-використань.			
8	**Тема 8.Критерії покриття потоку даних.** Поширені критерії тестового покриття на основі аналізу потоку даних. Проблеми існуючих критеріїв. Ланцюги потоку даних та спостережуваність результатів обчислень. Методи автоматизації генерації тестових даних для забезпечення покриття потоку даних. *Самостійна робота:* Для заданого прикладу побудувати тестовий набір для забезпечення покриття ланцюгів обчислень із забезпеченням спостережуваності результатів.	2		4
9	**Тема 9.Алгебраїчний підхід у верифікації та тестуванні.** Агенти та середовища. Алгебраїчна теорія взаємодії та інсерційне моделювання. Алгебра поведінок. Розв'язок поведінкових рівнянь. Алгебраїчна модель генерації тестів. Методи модельного тестування із використанням алгебраїчного підходу. Символьне виконання тестів. *Практична робота:*Для заданого прикладу представити програму в алгебрі поведінок. Показати можливі шляхи при символьному виконанні.	2		6
	Консультація (2 год)			
ВСЬОГО		18		40

Загальний обсяг*60* *годин*, в тому числі:
Лекцій – **18***годин*,
Консультації - **2** *години,*
Самостійна робота – **40** *годин***.**

9. Рекомендовані джерела

основні

1. Теорія алгоритмів [Текст] : навч. посіб. / М. С. Нікітченко, О. С. Шкільняк, С. С. Шкільняк ; Київ. нац. ун-т ім. Тараса Шевченка. - Київ : Київський університет, 2015. - 239 с. - Бібліогр.: с. 235-236. - 100 прим. - ISBN 978-966-439-831-9
2. Математична логіка та теорія алгоритмів [Текст] : підруч. для студ. кіберн. ф-тів вищ. навч. закл. / М. С. Нікітченко, С. С. Шкільняк ; Київ. нац. ун-т ім. Т. Шевченка. - К. : Київський університет, 2008. - 528 с. - Бібліогр.: с. 522-524. - 200 прим. - ISBN 966-439-007-0
3. Скінченні автомати: теорія, алгоритми, складність: підручник для студентів вищих навчальних закладів / Кривий С.Л. під. заг. ред. Палагіна О. В. Київ-Чернівці: "Букрек", 2020. 428 с. ISBN 978-617-7770-45-8
4. Основи дискретної математики, підручник, т.1 -2, Ю.Капітонова,О.Летичевський, С.Кривий, Г.Луцький, М. Печурін, К. 2000, ISBN 966-00-0622-5
5. Летичевський О.О.Алгебраїчне моделювання та його застосування. № 3 (2021): Вісник Національної академії наук України. https://doi.org/10.15407/visn2021.03.059
6. Летичевський О.О. Сучасні наукові проблеми кібербезпеки. Вісник НАН України. 2023. № 2. с. 12—20. https://doi.org/10.15407/visn2023.02.012
7. Лавріщева, К. М. Програмна інженерія : підручник. Академперіодика, Київ.2008. –319 с. ISBN 978–966–02–5052–9
8. Гейко О.О.Забезпечення правильності комп'ютерних моделей через верифікацію та валідацію. Вчені записки ТНУ імені В.І. Вернадського. Т.34. с. 30–37. 2023. https://doi.org/10.32782/2663-5941/2023.4/06
9. Чебанюк О.Agile підхід аналізу вимог із використанням технологій штучного інтелекту. Проблеми програмування. №2–3. С. 140–146. 2024. https://doi.org/10.15407/pp2024.02-03.140
10. Методи реалізації систем інсерційного моделювання [Текст] : автореф. дис. ... д-ра фіз.-мат. наук : 01.05.03 / Песчаненко Володимир Сергійович ; НАН України, Ін-т кібернетики ім. В. М. Глушкова. - Київ, 2015. - 40 с.

11. Символьні методи в тестуванні та верифікації високонадійних програмних систем [Текст] : автореф. дис. ... д-ра фіз.-мат. наук : 01.05.03 / Летичевський Олександр Олександрович ; НАН України, Ін-т кібернетики ім. В. М. Глушкова. - Київ, 2016. - 40 с.
12. Моделі та методи верифікації темпоральних моделей кінцевих автоматів на мовах опису апаратури : дис. ... д-ра філософії : 123 «Комп'ютерна інженерія» / Кирило Юрійович Пшеничний ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2024. – 108 с. – Бібліогр. : с. 99–107.
13. ДорошенкоА.Ю.Формальнітаінтелектуальніметодиавтоматизаціїпроєктуванняпрограмнихсистем. https://doi.org/10.15407/visn2025.04.063
14. А. М. Глибовець, Т. А. Чернова, М. М. Глибовець. Розробка методології імплементації транзакційв розподілених системах з мікросервісною архитектурою. Проблеми програмування.– №1. – 2024. – с.64–76. http://doi.org/10.15407/pp2024.01.064
15. Андон П.І., Дорошенко А.Ю., Акуловський В.Г., Іваненко П.А., Яценко О.А. Формальні та адаптивні методи конструювання високопродуктивних паралельних програм [Текст] : [монографія] / [П. І. Андон та ін. ; ред. В. В. Вероцька] ; НАН України, Ін-т програм. систем ; Проєкт "Наукова книга". - Київ : Наукова думка, 2023. - 310, [1] с. : рис., табл. - (Проєкт "Наукова книга"). - Бібліогр.: с. 298-307. - ISBN 978-966-00-1809-9
16. Колчин О.В., Потієнко С.В. Автоматизований метод перевірки та налагодження тестових сценаріїв наоснові формальної моделі. Control Systems and Computers.–2024, 3, –с.33-44. https://doi.org/10.15407/csc.2024.03.033
17. Генерація багатоцільових формальних моделей із успадкованого коду / С.В. Потієнко, О.В. Колчин // Проблеми програмування. — 2022. — № 3-4. — С. 42-50.https://nasplib.isofts.kiev.ua/handle/123456789/188627
18. Kolchin, A., Potiyenko, S. (2022). Extending Data Flow Coverage to Test Constraint Refinements. In: ter Beek, M.H., Monahan, R. (eds) Integrated Formal Methods. IFM 2022. Lecture Notes in Computer Science, vol 13274. Springer, Cham. https://doi.org/10.1007/978-3-031-07727-2_17
19. Kolchin, A., Potiyenko, S., Weigert, T. (2019). Challenges for Automated, Model-Based Test Scenario Generation. In: Damaševičius, R., Vasiljevienė, G. (eds) Information and Software Technologies. ICIST 2019. Communications in Computer and Information Science, vol 1078. Springer, Cham. https://doi.org/10.1007/978-3-030-30275-7_15
20. C. A. R. Hoare. An axiomatic basis for computer programming. Communications of the ACM, 12(10):576-585, –1969.https://doi.org/10.1145/363235.36325
21. S. Owicki, L. Lamport. Proving Liveness Properties of Concurrent Programs. ACM Trans. Program. Lang. Syst. 4, 3. 1982. 455–495. https://doi.org/10.1145/357172.357178
22. Clarke, Edmund M., et al., eds. Handbook of model checking. Vol. 10. Cham: Springer, 2018.ISBN 978-3-319-10575-8.https://doi.org/10.1007/978-3-319-10575-8
23. Myers, Glenford J. The art of software testing. 2012.https://doi.org/10.1002/9781119202486
24. Elaine J. Weyuker. 1986. Axiomatizing software test data adequacy. IEEE Trans. Softw. Eng. 12, 1 (January 1986), 1128–1138. https://doi.org/10.1109/TSE.1986.6313008
25. Letichevsky, A., Gilbert, D. A Model for Interaction of Agents and Environments. In: Bert, D., Choppy, C., Mosses, P.D. (eds) Recent Trends in Algebraic Development Techniques. WADT 1999. Lecture Notes in Computer Science, vol 1827. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-44616-3_18
26. Letichevsky, A., Gilbert, D. A general theory of action languages. Cybern Syst Anal 34, 12–30 (1998). https://doi.org/10.1007/BF02911258
27. Letichevsky, A.A., Letychevskyi, O.A., Peschanenko, V.S. (2012). Insertion Modeling System. In: Clarke, E., Virbitskaite, I., Voronkov, A. (eds) Perspectives of Systems Informatics. PSI 2011. Lecture Notes in Computer Science, vol 7162. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29709-0_23

рекомендовані

28. A. V. Anisimov and A. Novokshonov. Verifiable Arithmetic Computations Using Additively Homomorphic Tags.In: 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 93-96, https://doi.org/10.1109/ATIT49449.2019.9030485

29. Peled, D.A. (2019). Formal Methods. In: Cha, S., Taylor, R., Kang, K. (eds) Handbook of Software Engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-00262-6_5
30. Kolchyn, O., Potiyenko, S. (2024). Model-Based Test Cases Generation for Extended Data Flow Coverage Criteria. In: Silhavy, R., Silhavy, P. (eds) Software Engineering Methods Design and Application. CSOC 2024. Lecture Notes in Networks and Systems, vol 1118. Springer, Cham. https://doi.org/10.1007/978-3-031-70285-3_43
31. Weigert, T. et al. (2019). Generating Test Suites to Validate Legacy Systems. In: Fonseca i Casas, P., Sancho, MR., Sherratt, E. (eds) System Analysis and Modeling. Languages, Methods, and Tools for Industry 4.0. SAM 2019. Lecture Notes in Computer Science(), vol 11753. Springer, Cham. https://doi.org/10.1007/978-3-030-30690-8_1
32. Floyd, R.W. (1993). Assigning Meanings to Programs. In: Colburn, T.R., Fetzer, J.H., Rankin, T.L. (eds) Program Verification. Studies in Cognitive Systems, vol 14. Springer, Dordrecht. https://doi.org/10.1007/978-94-011-1793-7_4
33. Godefroid, P., Wolper, P. (1992). Using partial orders for the efficient verification of deadlock freedom and safety properties. In: Larsen, K.G., Skou, A. (eds) Computer Aided Verification. CAV 1991. Lecture Notes in Computer Science, vol 575. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-55179-4_32
34. Valmari, A., Clegg, M. (1991). Reduced labelled transition systems save verification effort. In: Baeten, J.C.M., Groote, J.F. (eds) CONCUR '91. CONCUR 1991. Lecture Notes in Computer Science, vol 527. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-54430-5_111
35. M.Y. Vardi and P. Wolper. 1994. Reasoning about Infinite Computations. Inf. Comput. 115, 1 (Nov. 15, 1994), 1–37. https://doi.org/10.1006/inco.1994.1092
36. Bryant R. Graph-Based Algorithms for Boolean Function Manipulation.In IEEE Transactions on Computers, vol. C-35, no. 8, pp. 677-691, Aug. 1986, https://doi.org/10.1109/TC.1986.1676819
37. R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. Journal of Computer and System Sciences. Volume 35, Issue 1. 1987. Pages 59-71.https://doi.org/10.1016/0022-0000(87)90036-5
38. Norris IP, C., Dill, D.L. Better verification through symmetry. Form Method Syst Des 9, 41–75 (1996). https://doi.org/10.1007/BF00625968
39. Park, D. (1981). Concurrency and automata on infinite sequences. In: Deussen, P. (eds) Theoretical Computer Science. Lecture Notes in Computer Science, vol 104. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0017309
40. Ho, A., Smith, S., & Hand, S. (2005). On deadlock, livelock, and forward progress (No. UCAM-CL-TR-633). University of Cambridge, Computer Laboratory.ISSN 1476-2986. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-633.pdf
41. Thomas Lengauer and Robert Endre Tarjan. 1979. A fast algorithm for finding dominators in a flowgraph. ACM Trans. Program. Lang. Syst. 1, 1 .1979. 121–141.https://doi.org/10.1145/357062.357071
42. R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. 1989. An efficient method of computing static single assignment form. In Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '89). Association for Computing Machinery, New York, NY, USA, 25–35. https://doi.org/10.1145/75277.75280
43. Gerard Holzmann.The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley – 2004. - 585 pp.https://dl.acm.org/doi/10.5555/1405716
44. Tim Weilkiens / Systems Engineering with SysML/UML: Modeling, Analysis, Design. The OMG Press. 2008. 320p. https://dl.acm.org/doi/10.5555/1628912
45. Patrice Godefroid, Michael Y. Levin, and David Molnar. 2012. SAGE: whitebox fuzzing for security testing. Commun. ACM 55, 3 (March 2012), 40–44. https://doi.org/10.1145/2093548.2093564

46. Cyrille Arthoet al. The Java Pathfinder Workshop 2019. SIGSOFT Softw. Eng. Notes 45, 2 (April 2020), 20–22. https://doi.org/10.1145/3385678.3385685
47. Păsăreanu, C.S., Visser, W., Bushnell, D. et al. Symbolic PathFinder: integrating symbolic execution with model checking for Java bytecode analysis. Autom Softw Eng 20, 391–425 (2013). https://doi.org/10.1007/s10515-013-0122-2
48. Su, T. et. al. A survey on data-flow testing. ACM Comput. Surv. 50, 35 p. (2017)https://doi.org/10.1145/3020266
49. Beer, I., et al.: Explaining counterexamples using causality. Formal Methods Syst. Des. 40(1), 20–40 (2012). https://doi.org/10.1007/s10703-011-0132-2
50. Neetu, J., Rabins, P.: Automated test data generation applying heuristic approaches—asurvey. Softw. Eng., pp. 699–708 (2019). http://doi.org/10.1007/978-981-10-8848-3_68
51. Barr, E., et al.: The oracle problem in software testing: a survey. IEEE Trans. Softw. Eng. 41,507–525 (2015). https://doi.org/10.1109/TSE.2014.2372785
52. Li, N., Offut J.: An experimental comparison of four unit test criteria: mutation, edge-pair,all-uses and prime path coverage. In: IEEE International Conference on Software Testing,Verification and Validation, pp. 220–229 (2009). http://doi.org/10.1109/ICSTW.2009.30
53. Namin, A., Andrews, J.: The influence of size and coverage on test suite effectiveness. In:Proceedings of International Symposium on Software Testing, pp. 57–68 (2009). http://doi.org/10.1145/1572272.1572280
54. Heimdahl, M., et al.: Test-suite reduction for model based tests: effects on test quality andimplications for testing. In: ASE Conference, pp. 176–185 (2004). http://doi.org/10.1109/ASE.2004.1342735
55. Morell, L.J.: A theory of fault-based testing. IEEE Trans. Softw. Eng. 16(8), 844–857(1990). https://doi.org/10.1109/32.57623
56. Hong, H.S., Ural, H.: Dependence testing: extending data flow testing with controldependence. In: Khendek, F., Dssouli, R. (eds.) TestCom 2005. LNCS, vol. 3502, pp. 23–39. Springer, Heidelberg (2005). https://doi.org/10.1007/11430230_3
57. Ntafos S. 1984. On required element testing. IEEE Trans. Softw.Eng. Vol.10. 795-803. https://doi.org/10.1109/TSE.1984.5010308
58. Cadar, C., Sen, K.: Symbolic execution for software testing: three decades later. Commun.ACM 56(2), 82–90 (2013). https://doi.org/10.1145/2408776.2408795
59. Staats, M., Gay, G., Whalen, M., Heimdahl, M.: On the danger of coverage directed test casegeneration. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 409–424.Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28872-2_28
60. Chilenski, J., Millner, S.: Applicability of modified condition/decision coverage to softwaretesting. Softw. Eng. J. 9, 193–200 (1994)https://doi.org/10.1049/sej.1994.0025
61. Miranda B., Bertolino A. 2020. Testing Relative to Usage Scope: Revisiting SoftwareCoverage Criteria. ACM Trans. Softw. Eng. Methodol. 29, 3, Art. 18.https://doi.org/10.1145/3389126
62. Barani M, Labiche Y and Rollet A. 2023. On factors that impact the relationshipbetween code coverage and test suite effectiveness: a survey. IEEE Int.Conf. on Software Testing, Verification and Validation Workshops (ICSTW).381–388. https://doi.org/10.1109/ICSTW58534.2023.00071
63. Bhatt D., Ren H., Murugesan A., Biatek J., Varadarajan, S., Shankar, N. 2022.Requirements-Driven Model Checking and Test Generation for ComprehensiveVerification. In: Deshmukh, J.V., Havelund, K., Perez, I. (eds) NASA FormalMethods. NFM 2022. Lecture Notes in Computer Science, Vol. 13260. Springer,Cham. https://doi.org/10.1007/978-3-031-06773-0_31
64. Vu, F., Leuschel, M.: Validation of formal models by interactive simulation. In: Glässer, U.,Creissac Campos, J., Méry, D., Palanque, P. (eds.) ABZ 2023. LNCS, vol. 14010, pp. 59–69.Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33163-3_5
65. Y. Meng, G. Gay and M. Whalen, "Ensuring the Observability of Structural Test Obligations," in IEEE Transactions on Software Engineering, vol. 46, no. 7, pp. 748-772, 1 July 2020, https://doi.org/10.1109/TSE.2018.2869146
66. Rastello, F., Tichadou, F.: SSA-Based Compiler Design, p. 382. Springer, Cham (2022).https://doi.org/10.1007/978-3-030-80515-9
67. Odarushchenko O., et al.: Application of formal verification methods in a safety-orientedsoftware development life cycle. In: 13th International Conference on Dependable Systems,Services and Technologies, pp. 1–6 (2023). https://doi.org/10.1109/DESSERT61349.2023.10416448

68. Liu, S., Nakajima, S.: Automatic test case and test oracle generation based on functionalscenarios in formal specifications for conformance testing. IEEE Trans. Softw. Eng. 48(2),691–712 (2022). https://doi.org/10.1109/TSE.2020.2999884
69. Kolchin, A., Potiyenko, S.,Weigert, T.: Extending data flow coverage with redefinition analysis.In: IEEE International Conference on Information and Digital Technologies (IDT), Zilina,Slovakia, pp. 293–296 (2021). https://doi.org/10.1109/IDT52577.2021.9497535
70. Martin, T., Kosmatov, N., Prevosto, V., Lemerre, M.: Detection of polluting test objectivesfor dataflow criteria. In: Dongol, B., Troubitsyna, E. (eds.) Integrated Formal Methods. IFM2020. Lecture Notes in Computer Science, vol. 12546. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63461-2_18
71. Lange, T., Neuhäußer, M.R., Noll, T., et al.: IC3 software model checking. Int. J. Softw. ToolsTechnol. Transf. 22, 135–161 (2020). https://doi.org/10.1007/s10009-019-00547-x
72. Gallardo, M.M.,Merino, P., Panizo, L.: The role of abstraction in model checking. In: Lopez-Garcia, P., Gallagher, J.P., Giacobazzi, R. (eds.) Analysis, Verification and Transformationfor Declarative Programming and Intelligent Systems. Lecture Notes in Computer Science,vol. 13160. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-31476-6_8
73. Beyer, D., Gulwani, S., Schmidt, D.: Combining Model Checking and Dataflow Analysis.Handbook of Model Checking, pp. 493–540. Springer, 2018. https://doi.org/10.1007/978-3-319-10575-8
74. Chekam T., Papadakis M., Traon Y., Harman M. An Empirical Study on Mutation, Statement and Branch Coverage Fault Revelation That Avoids the Unreliable Clean Program Assumption. In IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, pp. 597-608. (2017). https://doi.org/10.1109/ICSE.2017.61

Інтернет ресурси

75. SPIN Homepage, http://spinroot.com/spin/whatispin.html
76. ISO/IEC 13568:2002. Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics. [Електронний ресурс] https://www.iso.org/standard/21573.html
77. ITU-T Recommendation, Z.151, User Requirements Notation (URN) – Language definition. [Електронний ресурс] https://www.itu.int/rec/t-rec-z.151/en
78. ITU-T Recommendation, Z.120, Message Sequence Charts (MSC)[Електронний ресурс] https://www.itu.int/rec/t-rec-z.120
79. Temporal logics LTL, CTL and CTL*. Конспект лекцій [Електронний ресурс] http://people.irisa.fr/Francois.Schwarzentruber/mit2_cvfp_2011/7/poly_logiques_temporelles.pdf
80. Темпоральні логіки та їх застосування. [Електронний ресурс]https://ir.library.knu.ua/handle/123456789/2745
81. Simulation and Bisimulation. Конспект лекцій [Електронний ресурс]http://www.cse.unsw.edu.au/~cs3153/20T1/Week%2004/1Tue/Slides%20Condensed.pdf
82. Counter-example abstraction refinement method. Конспект лекцій [Електронний ресурс]https://www.cs.utexas.edu/~isil/cs389L/cegar-6up.pdf
83. Loopinvariants. Конспект лекцій [Електронний ресурс]https://www.cs.cornell.edu/courses/cs1110/2018sp/lectures/lecture24/presentation-24.pdf
84. Loopinvariants.Конспект лекцій [Електронний ресурс]https://courses.cs.washington.edu/courses/cse507/14au/slides/L4.pdf
85. Loopinvariants. Конспект лекцій [Електронний ресурс]https://www.cs.cmu.edu/~15122-archive/n17/lec/slides/loopinvariants.pdf
86. SAT/SMT: DPLL, теорії та застосування. Конспект лекцій [Електронний ресурс]https://resources.mpi-inf.mpg.de/departments/rg1/conferences/vtsa17/slides/reynolds-vtsa-part1.pdf
87. Symbolic execution. Конспект лекцій [Електронний ресурс]https://courses.cs.washington.edu/courses/cse403/16au/lectures/L16.pdf

88. Cyber vulnerabilities: CommonWeaknessEnumeration storage. [Електронний ресурс]https://cwe.mitre.org/top25/archive/2024/2024_kev_list.html
89. Cyber security: authentication. [Електронний ресурс]https://userpages.umbc.edu/~dgorin1/451/security/dcomm/authentication.htm
90. CyberSecurity: Network Security. Конспект лекцій [Електронний ресурс]https://userpages.umbc.edu/~dgorin1/451/security/dcomm/security_intro.htm
91. Cybersecurity: integrity. Конспект лекцій [Електронний ресурс]https://userpages.umbc.edu/~dgorin1/451/security/dcomm/integrity.htm
92. Тестування програм. Конспект лекцій [Електронний ресурс]http://mmsa.kpi.ua/sites/default/files/disciplines/Розробка%20і%20тестування%20програм/didkovska_m_v_testing_part2_criteria.pdf
93. Dataflowanalysis. Конспект лекцій [Електронний ресурс]https://greg4cr.github.io/courses/spring20dit635/Lectures/Spring20-Lecture9-DataFlow.pdf
94. Single static assignment form. Конспект лекцій [Електронний ресурс]https://groups.seas.harvard.edu/courses/cs153/2018fa/lectures/Lec23-SSA.pdf
95. Single static assignment form description. [Електронний ресурс]https://en.wikipedia.org/wiki/Static_single-assignment_form
96. Definition-usagechains. Конспект лекцій [Електронний ресурс]https://www.slideserve.com/chas/factored-use-def-chains-and-static-single-assignment-forms-powerpoint-ppt-presentation
97. Static Single Assignment: discussions and examples. Конспект лекцій [Електронний ресурс]https://www.cs.cornell.edu/courses/cs6120/2022sp/lesson/6/
98. Compilerdesigncodeoptimization. Конспект лекцій [Електронний ресурс]https://www.tutorialspoint.com/compiler_design/compiler_design_code_optimization.htm
99. Тестування програмного забезпечення систем. Конспект лекцій [Електронний ресурс]https://repositary.knuba.edu.ua/items/c9c554e3-17bc-4860-a870-35db8712bb7f
100. Сучасні технології автоматизованого проектування і верифікації програм. Конспект лекцій [Електронний ресурс]https://ela.kpi.ua/handle/123456789/56781