МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

СПЕЦІАЛІЗОВАНІ АРХІТЕКТУРИ КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ

Методичні вказівки та завдання
до проведення практичних та самостійних робіт
для студентів спеціальностей 123 «Комп'ютерна інженерія»
та 125 «Кібербезпека»

Київ 2023

УДК 681.3.06
С71

Укладач Д.О. Гуменний, канд. техн. наук, доцент

Рецензент О.В. Сєлюков, д-р техн. наук, професор

Відповідальний за випуск Ю.І. Хлапонін, д-р техн. наук, професор

Затверджено на засіданні кафедри кібернетичної безпеки та комп'ютерної інженерії, протокол №8 від 12 квітня. 2023 р.

В авторській редакції.

С71 **Спеціалізовані** архітектури комп'ютеризованих системи: методичні вказівки / уклад.: Гуменний Д.О. – Київ: КНУБА, 2023. – 32 с.

Наведено варіанти завдань для викладання практичних та самостійних робіт із дисципліни "Архітектура комп'ютеризованих систем".

Призначено для студентів, які навчаються за спеціальністю 123 "Комп'ютерна інженерія" та 125 "Кібербезпека".

ЗМІСТ

Вступ

Архітектор програмного забезпечення забезпечує розуміння продукту всіма учасниками проекту розробки цього ПЗ. Його задача полягає в тому, щоб добитися такого стану, коли клієнт, бізнес-рівень, менеджмент, фінанси та виконавці точно розуміють, як саме виглядатиме продукт, які витрати часу і коштів прогнозуватимуться. Архітектор поширює розуміння обмежень, що викладатимуться на продукт та процес його розробки.

Очевидно, що кожен учасник проекту має свою професійну грань розуміння, в рамках якої має бути пояснено суть проекту. Також, розробка архітектури програмного продукту потребує прогнозування того, як саме розвиватиметься цей продукт, який функціонал буде введений до його складу, у якому функціональному оточенні він знаходитиметься і які обмеження діятимуть на нього. Очевидно, що програмний продукт має відображати бачення замовника на його функціонування та вигляд, а відтак і вимоги, які накладаються замовником, мають бути враховані.

Побажання клієнта можуть бути неформальними. Подекуди вони не мають повного опису задачі, середовища використання та умов експлуатації програмного забезпечення. Оскільки нереалістичні вимоги можуть призвести до неможливості виконання проекту або затримок у його розробці, основним завданням архітектора програмного забезпечення є забезпечення реалістичних рішень, які можуть бути виконані у встановлений термін розробки продукту.

Архітектор має забезпечувати дотримання критеріїв неперервності зростання бізнесової цінності продукту, щоб дозволити клієнту фінансово підтримувати процес розробки.

Затверджені архітектором процеси та інструменти розробки повинні враховувати можливості неперервного розширення команди та функціональності продукту, а також враховувати можливість налагодження додаткових обмежень з боку контролюючих підрозділів компанії чи організації - споживача послуг.

Врахування всіх цих факторів є частиною обов'язків архітектора програмного забезпечення.

Цей посібник складений для структурування проведення практичних та самостійних робіт студентів відповідних спеціалізацій та рекомендується для використання у поєднанні з підходом до розробки архітектури програмного забезпечення, заснованому на моделі [2]. З метою мінімізації розпилювання уваги, основою цього посібника є процес побудови програмного забезпечення, характерний для сфери автомобільного виробництва [1]. Рішення про це було прийнято з огляду на високі вимоги нормативних документів, рекомендацій та обмежень, що діють у цій сфері.

Цей посібник, як і практична частина курсу, мають характер обговорення обраної теми з постійним залученням програмного забезпечення, заснованого на моделі [2] та нормативних документів з цієї теми.

Загальні положення

Основна мета дисципліни полягає у створенні умов для вивчення методів розробки архітектури комп'ютеризованих систем з точки зору ієрархічно розподіленої та розгалуженої моделі програмного забезпечення.

Друга мета дисципліни полягає у формуванні розуміння важливості масштабування програмного забезпечення, розподіленості програмного забезпечення та паралелізму у розробці.

Третя мета дисципліни полягає у формуванні чіткого розуміння того, як саме розпочинаються проекти, яким чином вони заводяться у компанію та як компанія перетворює потенційного клієнта на дійсного клієнта.

Критерії виконання робіт

Критерії виконання практичних робіт

Практична робота вважається виконаною з дотриманням наступних умов:

1. Студент подав викладачеві, відповідному до даної практичної роботи, продукти роботи для перевірки.
2. Студент провів доповідь та демонстрацію результатів практичної роботи відповідному викладачеві.
3. Матеріали практичної роботи, демонстрація роботи та доповідь отримали позитивну оцінку відповідного викладача.

Практична робота може мати позитивну оцінку (+1) та нейтральну (0).

Критерії виконання самостійних робіт

Самостійна робота вважається виконаною з дотриманням наступних умов:

1. Студент подав викладачеві, відповідному до даної самостійної роботи, продукти роботи для перевірки.
2. Студент провів доповідь та демонстрацію результатів самостійної роботи відповідному викладачеві.
3. Матеріали практичної роботи, демонстрація роботи та доповідь отримали позитивну оцінку відповідного викладача.

Самостійна робота може мати позитивну оцінку (+1) та нейтральну (0).

Вимоги до програмного забезпечення

Виконання циклу практичних та самостійних робіт передбачає наявність певного програмного забезпечення. Рекомендоване програмне забезпечення наведено у таблиці нижче.

Таблиця 1

Рекомендоване програмне забезпечення

Тип	Розробник	Назва
Operating System	Canonical	Ubuntu 22.04.2 LTS
Main Toolset	Mathworks	MATLAB 2022b
		Simulink
		System Composer
		Requirement toolbox
		Embedded Coder
Office Tools	Google	Google Docs

Виконання робіт у навчальних цілях можливе із застосуванням пробних ліцензій. Так, пробні ліцензії на продукцію Mathworks можна отримати безпосередньо на їхньому сайті [5]. Операційна система та офісний пакет доступні безкоштовно.

Додатковою вимогою є наявність доступу до відеозаписів практичних робіт по цьому курсу. Каталог відеозаписів доступний за посиланням [6].

Практичні роботи

Даний курс складається з п'яти практичних робіт, які є обов'язковими для виконання. Кожну роботу можна оцінити у діапазоні [0-15] балів. Оцінка здійснюється відповідно до повноти виконання роботи та до якості її виконання. Якість виконання оцінюється викладачем, як і повнота.

Максимальний бал за Практичну складову курсу - 75 балів.

Таблиця 2

Оцінювання якості виконання практичних робіт

Назва роботи	Підпункти порядку виконання та практичних робіт для успішного завершення курсу * - Обов’язкові для виконання ^ - Опційні								
Практична робота 1	1*	2*	3*	4^	5^	6^	7^	8^	9^
Практична робота 2	1*	2*	3*	4*	5*	6*	7*	8*	9^
Практична робота 3	1*	2*	3*	4*	5^	6*	7^	8-11*	12^
Практична робота 4	1*	2*	3*	4*	5*	6^	7*	8^	9,10^
Практична робота 5	1*	2*	3*	4*	5*	6^			

Практична робота №1

Тема роботи

“Створення шаблону архітектури програмного забезпечення засобами MATLAB Simulink System Composer”

Опис роботи

Після того, як системні специфікації та системні вимоги вже наявні. Та на основі цього вже створена узагальнена схема архітектури програмного забезпечення виникає необхідність відокремлення і формування програмних компонентів, які входять до компонент

Розробка програмного забезпечення для вбудованих систем керування передбачає наявність взаємодії між різними компонентами програмного забезпечення та взаємодію з стороннім програмним забезпеченням. До якого може належати як програмне забезпечення іншого абстрактного рівня, як абстракції драйверів, ресурсів пам’яті, обробників переривань так і засоби і функції операційної системи, яка може бути частиною програмного рішення, що буде працювати на базі мікроконтролерної системи, в якості вбудованої системи.

Так, для мінімізації кількості помилок, що виникатимуть під час розробки програмного забезпечення та його інтеграції з іншими компонентами, необхідно підходити до розробки з абстрактно-вищого рівня, а саме з рівня розробки системних специфікацій, які, в свою чергу мають бути декомпозовані до системних вимог.

Базуючись на системних вимогах та за інформацію із системних специфікацій має бути розроблена узагальнююча схема міжкомпонентних зв’язків, яка описує і показує, які саме сигнати і дані мають передаватися між окремими компонентами. Саме ця схема зв’язків і назв високорівневих абстракцій, з певними додатковими даними, як от частоста виклику, пріоритизації тощо і є предметом даної лабораторної роботи. Вона називається архітектурною схемою програмного забезпечення.

Розробка архітектурної схеми програмного забезпечення - це мета даної роботи.

Вхідні дані для виконання роботи

Для виконання Практична роботи №1 необхідно мати наступні знання та вміння:

- Знання
  - Типи даних для MATLAB Simulink та C(99)
  - Англійська мова на рівні не нижче A1
  - Методи та формати іменування змінних, що застосовуються у MATLAB Simulink та C(99)
- Вміння
  - Запуск MATLAB Simulink та System Composer [8]
  - Підготовка звітної документації відповідно до ДСТУ 3008-95 - “Звіти в області науки і техніки” [7]

Порядок виконання та рекомендації до виконання роботи

Для виконання практичної роботи необхідно досягти наступні пункти:

1. Отримати студентську ліцензію MATLAB Simulink, System Composer, Simulink report у представників MathWorks.
2. Інсталювати та запустити вищеназвані пакети
3. Створити нову локальну папку для роботи з Архітектурою. Назвати папку таким чином: *arch_P1_[прізвище студента]*. Створити поруч з цією папкою файл *README.md* та у текстовому форматі описати в ньому назву роботи, автора та короткий зміст роботи. Додати інструкції щодо запуску програмного архітектури.
4. Створити акаунт у GitHub та створити там репозиторій *Architecture*.
5. Створити SSH ключ на робочій машині, на якій ведуться практична роботи. Приєднати ключ до GitHub аккаунта
6. Створити гілку Practic1 та запушити в неї ініціативний коміт з коментарем Initial.
7. Зайшовши у MATLAB Simulink System Composer створити там нову програмну архітектуру. Зберегти бланк її у локальний фолдер *arch_P1_[прізвище студента] / Arch /* під іменет *Arch.*
8. Залити зміни гілки Practic*1* до репозиторію *Architecture* з відповідними комітами, що якнайкраще відображатимуть зміни проведені у MATLAB Simulink System Composer. При цьому наслідуйте наступну структуру комітів: *[шифр групи] - [текст коміту]*.
9. Створіть нову гілку та назвіть її Practic*1_Docs*. Потім, за допомогою

OpenOffice, Libreoffice, GoogleDoc, тощо створіть файл - системних специфікацій. Оформіть його згідно з ДСТУ 3008-95 та внесіть у нього хід виконання роботи. Додайте ілюстрації та посилання на джерела. Отриманий документ назвіть NOTE-*ARCH-L1-01-[прізвище студента].pdf* та залийте до новоутвореної гілки Practic*1_Docs*.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ викладачеві до репозиторію у якості рев'ювера, або лінк на архів з переліченими у пп. 1-9 артефактами.
- інвайт на захист. Тривалість 15 хв. Назва інайту *[шифр групи]-arch-P1-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.

Практична робота №2

Тема роботи

“Створення програмного компоненту - складової частини програмної архітектури”

Опис роботи

Розробка програмного забезпечення для вбудованих систем керування передбачає наявність взаємодії між різними компонентами програмного забезпечення. Також передбачає опис інтерфейсів кооперації компонентів із зовнішніми програмними частинами, апаратними частина (що переводить програмну архітектуру на системну архітектуру) чи людьми.

До зовнішніх програмних частин може належати і програмне забезпечення іншого абстрактного рівня, як абстракції драйверів, ресурсів пам’яті, обробників переривань, засоби і функції операційної системи, що може бути частиною програмного рішення.

Як і для першої роботи, друга робота базується на застосуванні системної специфікації (чи опису архітектури) та на системних вимогах. У ході роботи потрібно утворити програмний компонент, сконфігурувати його згідно із системною специфікацією та інтегрувати до узагальнююча схема міжкомпонентних зв’язків.

Тож, метою даної роботи є створення програмного компоненту і його конфігурація.

Вхідні дані для виконання роботи

Для виконання Практична роботи №2 необхідно мати наступні знання та вміння:

- Знання
    - Типи даних для MATLAB Simulink та C(99)
    - Англійська мова на рівні не нижче A1
    - Методи та формати іменування змінних, що застосовуються у MATLAB Simulink та C(99)
- Вміння
    - Запуск MATLAB Simulink та System Composer [8]
    - Створення програмного компоненту у MATLAB Simulink та System Composer [8]

- Налаштування моделі MATLAB Simulink
- Налаштування таргету у моделі MATLAB Simulink MATLAB Simulink та System Composer
- Запускати на компіляцію модель MATLAB Simulink та схему
- Підготовка звітної документації відповідно до ДСТУ 3008-95 - “Звіти в області науки і техніки” [7]

Порядок виконання та рекомендації до виконання роботи

Для виконання практичної роботи необхідно досягти наступні пункти:

1. На основі результатів виконання Практичної роботи №1 створити Симулінк-поведінку довільного програмного компоненту. Це має бути шаблонна форма поведінки, яка містить інтерфейси та опис.
2. Сформувати вимоги до роботи програмного компоненту. Додати їх до новоствореної в пп.1 моделі. Застосувати для поєднання вимог програмний додаток - Requirement Editor.
3. Відповідно до вимог, що були створену у пп.2, створити симулинк модель програмного компоненту
4. Полінкувати всі вимоги, що були створені у ході робіт над пп.2, до імплементації, яка була сторена у пп.3. За необхідності змінити вимоги та (чи) імплементацію, аби досягти повре покриття вимог, коли всі вимоги мають після провадження в модель, а всі впровадження мають вимоги і не формують так звану “мертву логіку”.
5. Шляхом відлагодження досягти стану, коли модель запускається.
6. Пересвідчитися що налаштування моделі програмного компоненту відповідають системним вимогам. При потребі виправити системні вимоги, попередньо узгодивши ці виправлення із викладачем, або виправити імплементацію.
7. Застосовуючи інструмент Model Anviser здійснити перевірку на відповідність моделі до гайдлайнів, стандартів та кращих практик. Спираючись на наступний перелік критеріїв:
    a. ISO 26262 / ISO 21434
    b. MISRA C (AC GMG)
    c. MAB (MAAB)
    d. Simulink

    у випадку, якщо виникають Error, чи Warning усунути причини таких повідомлень. У випадку протиріч, які виникають із за застосовуваних критеріїв - застосувати Justification. Описавши при цьому природу

проблеми і причини застосування Justification.

8. Забезпечити звіт про виконані роботи, які включали б наступні підпункти:
    a. Опис вимог
    b. Опис системи
    c. Опис імплементації
    d. Результати статичного аналізу
9. Подібно до виконаних пунктів із Практичної роботи 1, провести завантаження змін до Git репозиторію.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ викладачеві до репозиторію у якості рев'ювера, або лінк на архів з переліченими у пп. 1-8 артефактами.
- інвайт на захист. Тривалість 15 хв. Назва інайту *[шифр групи]-arch-P2-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.

Практична робота №3

Тема роботи

“Тестування програмного компоненту”

Опис роботи

Відповідно до ISO-26262, розробка функціонально-безпечного програмного забезпечення потребує наслідування V-моделі. Дана модель передбачає певну послідовність дій при документуванні, аналізу, створенні вимоги, імплементації, проведення статичного аналізу, юніт-тестування, інтеграції тощо.

При тому, проведення будь якого із видів тестування, валідації та верифікації проводиться паралельно до розробки і базується на тих самих вимогах, які застосовуються для розробки. Подекуди тестування випереджу розробку та накладає свої критерії валідності.

Так, метою даної роботи є створення тестового покриття на рівні юніт тестів для програмного компоненту.

Вхідні дані для виконання роботи

Для виконання Практична роботи №3 необхідно мати наступні знання та вміння:

- Знання
  - Типи даних для MATLAB Simulink та C(99)
  - Англійська мова на рівні не нижче A1
  - Методи та формати іменування змінних, що застосовуються у MATLAB Simulink та C(99)
- Вміння
  - Запуск MATLAB Simulink та System Composer [8]
  - Створення програмного компоненту у MATLAB Simulink та System Composer [8]
  - Налаштування моделі MATLAB Simulink
  - Налаштування таргету у моделі MATLAB Simulink MATLAB Simulink та System Composer
  - Запускати на компіляцію модель MATLAB Simulink та схему
  - Вміння будувати Test Harness
  - Вміння описувати тест кейс, як покрокову інструкцію до виконання на юніті під тестом

- Вміння конфігурувати Test Harness для отримання очікуваних метрик, таких як:
  - Condition
  - Decision
  - Execution
  - Complexity
  - MCDC
- Підготовка звітної документації відповідно до ДСТУ 3008-95 - “Звіти в області науки і техніки” [7]

Порядок виконання та рекомендації до виконання роботи

Для виконання практичної роботи необхідно досягти наступні пункти:

1. Створити Test Harness із програмного компоненту. Утворити їх як окремий файл, що має прив’язку до юніту, що стосуватиметься. Test Harness повинен містити Unit Under Test, Test Sequences, Test Assestments як окремі блоки, що створюються автоматично Simulink Test.
2. Пересвідчитися, що Test Harness має ті самі параметри, що і Unit Under Test. За потреби виправити їх вручні для отриманні відповідності.
3. Додати до Test Harness ті самі вимоги, які були застосовані для імплементації моделі.
4. У Test Sequences побудувати покрококу поведінку окремих тестовий випадів (Test Cases) аби досягти таку поведінки Unit Under Test, яка і очікується від нього. Застосувати When Decomposition метод, аби досягти незалежної поведінки окремих Test Cases. Пересвідчитися, що Test Harness зданий запускатися.
5. Прилінкувати вимоги до Test Cases. Подібно до того, як це виконувалося у Практичній роботі 2.
6. Описати Test Assestments. Зробити це відповідно до вимог.
7. Конфігурувати Test Harness аби отримувати наступні метрики про виконання тестових випадків:
   a. Condition
   b. Decision
   c. Execution
   d. Complexity
   e. MCDC
8. За допомогою інструменту Simulink Test Manager створити і зберегти

Test File. Конфігурувати файл таким чином, щоб у ньому була одна Test Suite, що містить Test Cases у кількості як кількість Test Cases у складі Test Sequences. Забезпечити логіку включення відповідного Test Cases для кожного із запусків Test Sequences.

9. Конфігурувати кожен із Test Cases так, аби він перевіряв вихід verify () функцій, які розташовані у Test Assestments. Ці функції мають містити критерії виконання кожного із Test Cases. Test Manager маю запускати кожен із Test Cases та проводити перевірку (verify()) кожного із критеріїв, що стосуються Test Cases. Результати тестів повинні відображатись як Pass/Fail/Not Tested. При виникненні Not Tested мають бути наведені пояснення.
10. Активувати генерування звіту про тестування. Надати ім'я файлу із звітом у форматі *[шифр групи]-arch-P3-rep-[прізвище студента]*. Вказати ім'я виконавця.
11. Активувати калькуляцію та вивід відсотковості покриття тестами за переліченими вище критеріями. Досягти повного покриття. При неможливості отримання повного покриття розширити кількість вимог. При неможливості досягти повного покриття після додавання вимог - підготувати Justification.
12. Досягти запускаємості тестів. Підготувати звіт про тест статус.
13. Подібно до того, як було виконано у практичних роботах 1 та 2 провести завантаження тестів у Git. для цього створити новий каталог TEST у корені каталогу проекту. Розмістити туди Test Harness, Test File. Файл тестового звіту не додавати до Git.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ викладачеві до репозиторію у якості рев'ювера, або лінк на архів з переліченими у пп. 1-13 артефактами.
- інвайт на захист. Тривалість 15 хв. Назва інвайту *[шифр групи]-arch-P3-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.
- Файл текст звіту має бути додано до інвайту.

Практична робота №4

Тема роботи

“Масштабування проекту”

Опис роботи

Важливою ознакою коректного архітектурного рішення, що побудовано для системи чи комплексу технічних засобів є здатність цього рішення масштабуватися. Таке масштабування характеризується відсутністю потреби змінювати вже впроваджені рішення під час додавання чогось нового. Також, таке рішення характеризується можливістю збільшити кількісні характеристики обробки даних без зміни імплементації, а лише за рахунок конфігураційних дій.

Так, метою даної практичної роботи є масштабування програмної композиції шляхом введення нового програмного компоненту до її складу.

Вхідні дані для виконання роботи

Для виконання Практична роботи №3 необхідно мати наступні знання та вміння:

- Знання
  - Типи даних для MATLAB Simulink та C(99)
  - Англійська мова на рівні не нижче A1
  - Методи та формати іменування змінних, що застосовуються у MATLAB Simulink та C(99)
- Вміння
  - Запуск MATLAB Simulink та System Composer [8]
  - Створення програмного компоненту у MATLAB Simulink та System Composer [8]
  - Налаштування моделі MATLAB Simulink
  - Налаштування таргету у моделі MATLAB Simulink MATLAB Simulink та System Composer
  - Запускати на компіляцію модель MATLAB Simulink та схему
  - Вміння будувати Test Harness
  - Вміння описувати тест кейс, як покрокову інструкцію до виконання на юніті під тестом
  - Вміння конфігурувати Test Harness для отримання очікуваних метрик, таких як:

        - Condition
        - Decision
        - Execution
        - Complexity
        - MCDC
    - Вміння користуватися System View / Architectural View
    - Підготовка звітної документації відповідно до ДСТУ 3008-95 - “Звіти в області науки і техніки” [7]

Порядок виконання та рекомендації до виконання роботи

Для виконання практичної роботи необхідно досягти наступні пункти:

1. Відкрити та перевірити працездатність програмної композиції, яка була сторена на оновлена протягом практичних робіт 1-4.
2. Поруч із вже існуючим програмних компонентом створити ще один програмний компонент. Надати йому робочу назву.
3. Створити самостійно, або у обговорені із викладачем програмні вимоги, які необхідно імплементувати у програмний компонент, що був створений у пп.2. Описати ці вимоги у Requirement Editor та зберегти зміни.
4. Описати інтерфейси до програмного компоненту відповідно до нових вимог. Оновити Системні вимоги, що були внесені до Requirement Editor під час робіт із практичною роботою №1.
5. Перевірити працездатність системи.
6. За допомогою Architecture View генерувати такі схеми огляду архітектури, які б відображали наступні речі:
    a. де саме прилінковані вимоги до імплементації. Зробити це для всієї композиції та для окремих компонентів
    b. показати як саме компоненти взаємодіють один-з-одним
    c. показати часові діаграми налаштування компонентів
7. Імплементувати функціонал, який описаний у програмних вимогах. Імплементувати його до новоутвореного програмного компоненту.
8. Провести статичний аналіз новоствореного програмного компоненту. За наявності помилок - виправити їх. За неможливості виправити помилки - провести Justification. Результати статичного аналізу подати у вигляді звіту.
9. Побудувати Test Harness на основі новоутворених вимог. Описати Test Cases та створити відповідний Test File. провести Verification and

Validation нової імплементації відповідно до нових вимог. Застосувати ті самі Coverage Metric що було застосовано для практичної роботи №3. Згенерувати звіт по тестуванню

10. Всі артефакти та файли проекту мають бути завантажені до Git репозиторію.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ викладачеві до репозиторію у якості рев'ювера, або лінк на архів з переліченими у пп. 1-10 артефактами.
- інвайт на захист. Тривалість 15 хв. Назва інвайту *[шифр групи]-arch-P4-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.
- Файл текст звіту має бути додано до інвайту.

Практична робота №5

Тема роботи

“Інтегрування та вбудовування рішення у склад цифрової системи керування”

Опис роботи

По завершенню даного курсу практичних є задача інтегрувати всі програмні компоненти, чи програмну композицію у єдиний програмний код, який придатний до завантаження на цільовий контролер у якості прикладного рівня програмного забезпечення.

Для впровадження композиції до цільового контролера потрібно визначитися із базовими вимогами та обмеженнями, що накладені цим обладнанням. Та, із застосуванням програмного додатку Embedded Coder згенерувати С-код, придатний до подальшого застосування на Middle Layer програмного забезпечення.

Метою цієї практичної роботи є генерування С-коду із програмної композиції.

Вхідні дані для виконання роботи

Для виконання Практична роботи №2 необхідно мати наступні знання та вміння:

- Знання
  - Типи даних для MATLAB Simulink та C(99)
  - Англійська мова на рівні не нижче A1
  - Методи та формати іменування змінних, що застосовуються у MATLAB Simulink та C(99)
- Вміння
  - Запуск MATLAB Simulink та System Composer [8]
  - Створення програмного компоненту у MATLAB Simulink та System Composer [8]
  - Налаштування моделі MATLAB Simulink
  - Налаштування таргету у моделі MATLAB Simulink MATLAB Simulink та System Composer
  - Вміння генерувати програмний код із застосуванням Embedded Coder.
  - Розуміння параметрів оптимізації програмного коду

- Запускати на компіляцію модель MATLAB Simulink та схему
- Підготовка звітної документації відповідно до ДСТУ 3008-95 - “Звіти в області науки і техніки” [7]

Порядок виконання та рекомендації до виконання роботи

Для виконання практичної роботи необхідно досягти наступні пункти:

1. Запустити програмну композицію. Пересвідчитися, що вона працююча.
2. Для кожного із програмних компонент відкрити діалог Параметрів Моделі та у закладні Target обрати ARM; Linux; C 99. Зберегти зміни.
3. У інструменті Embedded Coder згенерувати код із застосуванням оптимізації за пам’яттю. Повторити операцію із оптимізацією по швидкодії. берегти обидві версії порізну.
4. Порівняти результати генерованого коду. Визначити де саме точка входу до програми. Описати де саме кожен із компонентів та де саме кожна підсистема кожного із компонентів. Порівнюючи вказати що змінилося при різних критеріях оптимізації коду. Звернути увагу на ті рішення, які пов’язані із критеріями кібербезпеки.
5. Підготувати звіт про виконану роботу. Зазначивши попунктно результати проведеної роботи.
6. Результати роботи завантажити у резульну гілку Git.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ викладачеві до репозиторію у якості рев’ювера, або лінк на архів з переліченими у пп. 1-6 артефактами.
- інвайт на захист. Тривалість 15 хв. Назва інвайту *[шифр групи]-arch-P5-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.
- Файл текст звіту має бути додано до інвайту.

Самостійні роботи

Даний курс складається з двох самостійних робіт, які є обов'язковими для виконання. Кожну роботу можна оцінити у діапазоні [0-7] балів. Оцінка здійснюється відповідно до повноти виконання роботи та до якості її виконання. Якість виконання оцінюється викладачем, як і повнота.

Максимальний бал за Практичну складову курсу - 14 балів.

Таблиця 3

Оцінювання якості виконання самостійних робіт

Назва роботи	Оцінки по самостійних роботах
Самостійна робота 1	бали:
Самостійна робота 2	бали:

Самостійна робота №1

Тема роботи

“Інтеграційне тестування програмної композиції”

Завдання до самостійної роботи

На основі імплементації, системних вимог та програмних вимог побудувати високорівневий TestHaness який покривав би тестами всю композицію та проводив Validation and Verification системних вимог та інтерфейсів. Для проведення Validation and Verification необхідно застосувати Simulink Test та отримати відповідний Test Report. До результатів тестування має бути включено: Test Result, Test Coverage із критеріями зазначеними у Практичних роботах.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ до Test Report та Test Harness
- інвайт на захист. Тривалість 15 хв. Назва інвайту *[шифр групи]-arch-S1-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.

Самостійна робота №2

Тема роботи

“Аналіз вимог V-Model”

Завдання до самостійної роботи

Згідно із A-SPICE (Фреймворк), ISO-26262 (Функіональна безпека), ISO-21434 (Кібербезпека) загальноприйнятою моделлю розробки програмного забезпечення для системи із наявними вимогами по функціональній безпеці та кібербезпеці є V-Model. Ця модель накладає певні правила на розробку та тестування програмного забезпечення які рекомендується, або строго рекомендуется слідувати.

Завданням цієї самостійної роботи є проаналізувати причини застосування V-моделі. ЇЇ позитивні та негативні аспекти, документацію, розробку та тестування. Надати матеріали у вигляді реферату. Обсяг реферату, його структура можу бути довільна, але має бути вичерпна.

Подання матеріалів на захист роботи

До захисту практичної роботи подаються наступні артефакти:

- доступ до реферату
- інвайт на захист. Тривалість 30 хв. Назва інвайту *[шифр групи]-arch-S2-[прізвище студента]*. Інвайт має бути надісланий відповідному викладачеві та апрувнутний ним.

Заключне слово

У даній методичці ми розглянули основні аспекти архітектури програмного забезпечення для комп'ютеризованих систем. Архітектор програмного забезпечення має забезпечувати розуміння продукту всіма учасниками проекту та враховувати вимоги клієнта. Основні завдання архітектора полягають у реалістичному врахуванні вимог, прогнозуванні розвитку продукту, визначенні функціонального оточення та обмежень, а також забезпеченні неперервного зростання бізнесової цінності продукту. Методичка рекомендує підхід до розробки архітектури, заснований на відповідній моделі, і враховує особливості сфери автомобільного виробництва.

Список рекомендованої літератури

1. Automotive SIG V. Q. W. G. 1. /. Automotive SPICE Process Assessment / Reference Model. A-SPICE. URL: https://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf (date of access: 18.04.2023).
2. What is Model-Based Approach | IGI Global. IGI Global: International Academic Publisher. URL: https://www.igi-global.com/dictionary/formalizing-model-based-multi-objective-reinforcement-learning-with-a-reward-occurrence-probability-vector/35014 (date of access: 18.04.2023).
3. System Composer. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/products/system-composer.html (date of access: 18.04.2023).
4. AUTOSAR Software Components and Compositions- MATLAB & Simulink. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/help/autosar/ug/autosar-software-components-and-compositions.html (date of access: 18.04.2023).
5. Trials. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/campaigns/products/trials.html (date of access: 18.04.2023).
6. Гуменний Д. Відеоматеріали курсу Архітектура комп'ютеризованих систем. Sign in to your account. URL: https://knuba365-my.sharepoint.com/:f:/g/personal/humennyi_do_knuba_edu_ua/EuUUsPoLOtxNu6AXEWPTbOcBXRUghKFT_zbGwkl2USKZTg?e=CxZ77k (дата звернення: 18.04.2023).
7. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки [Текст]. - [Чинний від 1995-07-01]. - К.: Держспоживстандарт України, 1995. - 12 с.
8. MathWorks. Getting Started with System Composer [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/help/systemcomposer/getting-started-with-system-composer.html (дата звернення: 02.05.2023).

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчально-методичне видання

СПЕЦІАЛІЗОВАНІ АРХІТЕКТУРИ КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ

Методичні вказівки та завдання
до проведення практичних та самостійних робіт
для студентів спеціальностей 123 «Комп'ютерна інженерія»
та 125 «Кібербезпека»

Укладач **Гуменний** Дмитро Олександрович

Комп'ютерне верстання *М.М. Власенко*

Підписано до друку 12.04.2023 Формат 60 х 84 1/ 16
Ум. друк. арк. 1,98. Обл.-вид. арк. 0,83.
Електронний документ. Вид № 59/ІІІ-17.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, Україна, 03680
Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.