Heorhii Borodkin
Старший викладач кафедри комп'ютерних наук
National University of Life and Environmental Sciences of Ukraine, Kyiv, Ukraine
ORCID ID 0000-0002-6488-6512
george.borodkin@gmail.com
Iryna Borodkina
Кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук
Kyiv National University of Culture and Arts, Kyiv, Ukraine
ORCID ID 0000-0003-3667-3728
borir@ukr.net

СУЧАСНІ ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІДПОВІДНО ДО РЕКОМЕНДАЦІЙ SWEBOK

Анотація. В статті розглядаються сучасні підходи до створення програмних продуктів відповідно до рекомендацій SWEBOK**.** Аналізується досвід розробки програмних продуктів, сформульований у вигляді рекомендацій.

Ключові слова: ядро знань SWEBOK; створення програмного забезпечення; технології створення програмних продуктів.

1. ВСТУП

Постановка проблеми. В кінці 90-х років XX століття розпочався дуже інтенсивний процес впорядкування засобів та технологій створення програмного забезпечення, яке було призначене для продажу – програмних продуктів. Для керування цим процесом на початку XXI століття був створений Міжнародний координаційний комітет з програмної інженерії (Software Engineering Coordinating Committee) при американському об'єднанні комп'ютерних фахівців ACM (Association for Computing Machinery) та Інституті інженерів з електроніки та електротехніки (IEEE Computer Society). Завдяки зусиллям цього комітету у 2004 році з'явилось ядро знань SWEBOK. В ньому систематизовані різнорідні знання в області програмування, планування і управління, сформульовано поняття програмної інженерії та визначено 15 (10 на 2004 рік) областей знань (ОЗ), які відповідають процесам проектування програмного забезпечення (ПЗ) і методам їх підтримки.

Аналіз останніх досліджень і публікацій. Це ядро знань стало настільним довідником для фахівців з інженерії програмного забезпечення, бо вміщує у собі найкращий досвід у створенні програмних продуктів. Воно постійно розвивається і наповнюється новими рекомендаціями, що спираються на новітні засоби, технології і інструменти розробки ПЗ.

Мета публікації. Враховуючи це,метою статті є висвітлення підходів до створення програмних продуктів відповідно до рекомендацій SWEBOK.

РЕЗУЛЬТАТИ ТА ОБГОВОРЕННЯ

Для створення сучасних програмних систем SWEBOK пропонує користуватись технологіями, які показали свою ефективність.

Розробка та використання API. Інтерфейс прикладного програмування (Application Programming Interface – API) – це набір записів, які експортуються та є доступними користувачам бібліотеки або фреймворку для написання своїх власних програм. Окрім записів, API завжди повинен містити твердження про ефекти та/або поведінку програми (тобто її семантику). Розробка API повинна намагатися зробити API: легким для вивчення та запам'ятовування, зручним для створення читабельного коду, важким для

хибного використання, легким для розширювання, повним та здатним підтримувати зворотну сумісність. Оскільки API для широко вживаної бібліотеки або фреймворку, зазвичай, існує довше, ніж реалізація певного проекту, бажано, щоб API був простим і стабільним, таким, що полегшує розробку та обслуговування клієнтських додатків. Використання API включає процеси вибору, навчання, тестування, інтеграції та, можливо, розширення API, що надаються бібліотекою або фреймворком

Особливості об'єктно-орієнтованого підходу до запуску на виконання. Об'єктно-орієнтовані мови підтримують низку механізмів виконання, включаючи поліморфізм та адаптованість. Ці механізми реалізації підвищують гнучкість та адаптованість об'єктно-орієнтованих програм. Поліморфізм – це здатність мови підтримувати загальні операції, не знаючи до моменту виконання, які конкретні об'єкти буде містити програмне забезпечення. Оскільки програма заздалегідь не знає конкретних типів об'єктів, то реальна поведінка визначається під час виконання (так зване динамічне прив'язування). Адаптованість – це здатність програми відслідковувати та модифікувати власну структуру та поведінку під час виконання. Адаптованість дозволяє перевіряти класи, інтерфейси, поля та методи під час виконання, не знаючи їх імен під час компіляції. Це також дозволяє створювати екземпляри під час виконання нових об'єктів та викликати методи обробки за рахунок використання параметризованих імен класів та методів.

Параметризація та шаблони аргументів (дженерики). Параметризовані типи, які також відомі як дженерики (мови Ada, Eiffel) та шаблони аргументів (C ++), дозволяють визначити тип або клас і не вказувати всі інші типи, які параметризовані типи використовують. Невстановлені типи подаються як параметри у місце використання. Параметризовані типи забезпечують третій спосіб (на додаток до наслідування класів та вмісту об'єкта) визначення поведінки в об'єктно-орієнтованому програмуванні.

Твердження, проектування за контрактом та захисне програмування. Твердження (Assertion) (варіанти перекладу терміну: припущення, твердження, пересвідчення) – це виконуваний предикат, який розміщується в програмі, як правило, у вигляді підпрограми або макросу, і дозволяє перевіряти виконання програми. Твердження особливо корисні в програмах з високою надійністю. Вони дозволяють програмістам швидше виправляти невірні допущення інтерфейсу, помилки, які закрадаються під час зміни коду тощо. Твердження, зазвичай, додаються у код під час розробки, а згодом вилучаються з коду, аби вони не погіршували продуктивність.

Проектування за контрактом (Design by contract) – це такий підхід до розробки, при якому передумови та післяумови включаються в кожну програму. Коли використовуються передумови та післяумови, кажуть, що кожна програма чи клас укладають контракт з рештою програми. Крім того, контракт надає точну специфікацію семантики програми і, таким чином, допомагає зрозуміти її поведінку. Вважається, що проектування за контрактом покращує якість конструювання програмного забезпечення.

Захисне програмування (Defensive programming) має на меті захистити програму від руйнування через хибні введення (invalid inputs). Поширені способи обробки хибних входжень включають перевірку значень усіх вхідних параметрів та вирішення способу обробки хибного введення. Твердження часто використовуються в захисному програмуванні для перевірки вхідних значень.

Обробка помилок, обробка винятків та толерантність до збоїв. Спосіб обробки помилок впливає на здатність програмного забезпечення відповідати вимогам, які пов'язані з коректністю, надійністю та іншими атрибутами нефункціональних вимог. Для перевірки на помилки використовують широкий набір різних методів обробки Винятки використовуються для виявлення та обробки помилок або виняткових подій. Толерантність до збоїв (Fault tolerance) – це сукупність методів, які підвищують надійність програмного забезпечення шляхом виявлення помилок, а потім відновлення

програмного забезпечення, якщо це можливо, або обмеження наслідків від помилок, якщо відновлення неможливе. Найбільш поширені стратегії стійкості до відмов включають: резервне копіювання і повторна спроба виконання, використання допоміжного коду, використання алгоритмів більшості голосів, заміна помилкового значення на фальшиве значення, яке не матиме згубного ефекту.

Моделі реалізації (Executable Models). Моделі реалізації абстрагуються від деталей конкретних мов програмування та рішень щодо організації програмного забезпечення. На відміну від традиційних програмних моделей, модель, яка побудова¬на на мові моделювання для реалізації, наприклад, xUML (UML для реалізації), може бути розгорнута в різних програмних середовищах без змін. Компілятор моделі реалізації (трансформатор) може перетворити виконувану модель на реалізацію за допомогою набору рішень щодо цільового апаратного та програмного середовища. Таким чином, побудова моделей реалізації може розглядатися як спосіб побудови програмного забезпечення для виконання. Моделі реалізації є однією з основ, що підтримують технології, це спосіб цілком визначити незалежну від платформи модель.

Методи конструювання на основі станів та управління таблицями. Програмування на основі станів (state-based programming), або програмування на основі автоматів, – це технологія програмування, що використовує кінцеві автомати для опису поведінки програм. Графіки переходів автомата стану використовуються на всіх етапах розробки програмного забезпечення. Основна ідея полягає в тому, щоб будувати комп'ютерні програми так само, як це робиться з автоматизацією технологічних процесів. Програмування на основі станів, зазвичай, поєднується з об'єктно-орієнтованим програмуванням, що утворює новий складений підхід, який називається об'єктно-орієнтоване програмування на основі станів.

Конфігурування файлів на виконання та інтернаціоналізація. Для досягнення більшої гнучкості програма часто створюється з можливістю підтримки пізньої часової прив'язки програмою своїх змінних. Конфігурування файлів на виконання – це технологія, яка пов'язує значення змінних та налаштування програми під час запуску програми, як правило, шляхом оновлення та читання файлів конфігурації у режимі он-лайн. Інтернаціоналізація – це технічна діяльність з підготовки програми, як правило, інтерактивного програмного забезпечення для підтримки декількох локальних налагоджувань.

Першочергове програмування тестів (розробка під управлінням тестів – Test-Driven Development – TDD) – це популярний стиль розробки, в якому тестові кейси пишуться перед написанням будь-якого коду. Програмування на основі тестування, зазвичай, може виявити дефекти раніше і виправити їх легше, ніж традиційні стилі програмування. Більше того, написання тестових кейсів спочатку змушує програмістів подумати про вимоги та проектування раніше за кодування, тим самим швидше з'ясовуючи вимоги до ПЗ та проблеми з проектуванням.

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL UNIVERSITY OF LIFE AND ENVIRONMENTAL SCIENCES OF UKRAINE

FACULTY OF INFORMATION TECHNOLOGY

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

PROCEEDINGS

МАТЕРІАЛИ

XI International scientific conference

XI Міжнародної науково-практичної конференції

GLOBAL AND REGIONAL PROBLEMS OF INFORMATIZATION IN SOCIETY AND NATURE USING '2023

ГЛОБАЛЬНІ ТА РЕГІОНАЛЬНІ ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ В СУСПІЛЬСТВІ І ПРИРОДОКОРИСТУВАННІ '2023

15-16 November 2023

Kyiv, NULES of Ukraine

15-16 листопада 2023 року

Київ, НУБіП України

Kyiv 2023

Київ 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАТЕРІАЛИ

XI Міжнародної науково-практичної конференції

ГЛОБАЛЬНІ ТА РЕГІОНАЛЬНІ ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ В СУСПІЛЬСТВІ І ПРИРОДОКОРИСТУВАННІ '2023

15-16 листопада 2023 року

Київ, НУБіП України

Київ 2023

УДК 004

Рекомендовано до друку вченою радою факультету інформаційних технологій Національного університету біоресурсів і природокористування України (протокол № 4 від 20.11.2023)

Укладач: к.е.н., доцент Харченко В.В.

Збірник матеріалів XI Міжнародної науково-практичної конференції "Глобальні та регіональні проблеми інформатизації в суспільстві і природокористуванні '2023", 15-16 листопада 2023 року, НУБіП України, К. НУБіП України, 2023. 117 с.

Відповідальність за зміст публікацій несуть автори.