УДК 004.4:658.012.23 **doi: 10.32620/aktt.2025.2.09**

А. В. ЗЕЛЕНКОВ, Л. В. МАНДРІКОВА, Є. Б. БОРОВКОВ

Національний аерокосмічний університет імені М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна

УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА БАЗІ СТАНДАРТУ OMG ESSENCE

Предметом вивчення *в статті є процеси управління життєвим циклом програмного забезпечення.* ***Метою є*** *підвищення ефективності командної розробки програмного забезпечення (ПЗ), а також забезпечення розвитку практик, методів та методологій програмної інженерії.* ***Завдання:*** *аналіз особливостей управління проєктами з розробки ПЗ, дослідження стандарту OMG Essence стосовно опису методів та практик інженерії програмного забезпечення, розробка процедури управління життєвим циклом ПЗ з використанням мови та ядра Essence.* ***Методами дослідження*** *є аналіз науково-технічної документації, системний аналіз, планування та моделювання процесу розробки ПЗ.* ***Результати.*** *Описано порядок управління життєвим циклом ПЗ відповідно до стандарту OMG Essence. Предметно-спеціальна мова Essence, яка має текстовий і графічний синтаксис, використовується для уніфікованого опису методів та практик програмної інженерії. Практики можуть бути адаптовані або заново створені для задоволення конкретних потреб проекту та особливостей організації роботи IT-компанії; з різних практик можуть бути скомпоновані методи. Для моніторингу прогресу та стану ПЗ, а також для аналізу сильних і слабких сторін способу роботи команди пропонується використовувати стани Альф. Альфа (Abstract-Level Progress Health Attribute, ALPHA) - це суттєвий абстрактний елемент проекту в галузі програмної інженерії, за яким потрібно стежити та оцінювати його просування, тому на основі станів Альф пропонується здійснювати планування та управління проєктами розробки ПЗ. Знаючи початковий і бажаний стан набору Альф за допомогою стандарту OMG Essence можна визначити перелік та зміст дій для просування проєкту. На прикладі проєкту розробки ПЗ продемонстровано управління життєвим циклом в Essence WorkBench та досліджено цей інструментальний засіб.* ***Висновки.*** *Отримав розвиток комплексний підхід до управління життєвим циклом ПЗ на основі стандарту OMG Essence. Запропоновано для оцінки стану окремих Альф разом з контрольними списками використовувати показники роботи (KPI) організації – замовника та компанії – розробника ПЗ. Визначено шляхи подальшого розвитку методологій управління життєвим циклом програмного забезпечення на основі OMG Essence, а саме розвиток Essence WorkBench або створення іншого засобу здатного інтегруватися з системами, що автоматизують різні практики програмної інженерії та управління проєктими.*

Ключові слова: *OMG Essence; управління життєвим циклом програмного забезпечення; інженерія програмного забезпечення; управління проєктами; компетенція.*

1. Вступ

1.1. Мотивація дослідження

Програмне забезпечення (ПЗ) в аерокосмічній галузі використовується для управління літальними апаратами, автоматизованого проєктування, управління виробництвом, організаційно-економічного управління підприємствами галузі. Підходи до управління життєвим циклом ПЗ мають враховувати специфіку галузі, що досліджується в роботі [1] на прикладі супутника CubeSat. Аерокосмічне ПЗ вимагає специфічної архітектури та методів кодування, високої якості, дотримання суворих стандартів, ретельного тестування тощо.

Розробка ПЗ для аерокосмічної галузі - це надзвичайно відповідальний процес, в якому застосовуються як універсальні так і спеціальні методи та практики. Кожен метод має свою індивідуальну специфічну структуру та використовує власний стиль і термінологію. Більшість методів є монолітними, тобто вони не розроблені модульним способом, хоча часто мають багато спільного. Все це заважає розвитку та створенню нових методів шляхом комбінації елементів існуючих. Команди розробників повинні мати можливість створювати власні методи, вдосконалюючи та поєднуючи існуючі на нові практики [2].

Розробка ПЗ вимагає наявності кваліфікованого персоналу, який має відповідну освіту, необхідні знання, вміння та досвід. Програмна інженерія включає 15 основних та 8 споріднених областей знань, на

яких базуються процеси розробки ПЗ [3]. Співпраця між розробниками ПЗ та освітянами буде більш ефективною за наявності єдиного способу опису знань в галузі програмної інженерії.

Ефективне управління життєвим циклом застосунків (Application Lifecycle Management, ALM) має забезпечувати досягнення цілей організацій, які займаються їх розробкою, стосовно забезпечення оперативних та якісних випусків програмних продуктів в межах бюджету [4]. Від ПЗ вимагається також висока адаптивність до зміни вимог, які відбуваються протягом його життєвого циклу. Якість програмної системи розуміється як рівень, за якого ця система задовольняє потреби різних зацікавлених сторін і в такий спосіб забезпечує її значущість для них [5]. Процес розробки та експлуатації ПЗ суттєво залежить від замовників та інших стейкхолдерів і має бути адаптивним до зміни їх цілей та потреб.

Актуальним є завдання розробки комплексного спеціалізованого підходу до управління життєвим циклом ПЗ, який можна було застосувати до будь-якого ІТ-проекту незалежно від галузі замовника, чисельності та рівня кваліфікації команди розробки. Враховуючи динамічний розвиток галузі необхідно забезпечити уніфікований та простий спосіб опису, а також розвиток методів та практик програмної інженерії під час виконання проєктів.

1.2. Сучасний стан

Програмна інженерія забезпечує керований комплексний підхід до створення складного програмного продукту колективом інженерів, що визначає всі кроки цієї діяльності - від початкової ідеї до завершення використання продукту клієнтами. ALM включає: керування вимогами, проєктування архітектури, розробку, тестування, обслуговування, керування змінами, підтримку, безперервну інтеграцію, керування проєктами, розгортання, керування випуском. В статті [6] підкреслюється актуальність вдосконалення стандартів і вказівок щодо управління життєвим циклом великих та складних програмних систем. В роботі [7] розглянуто екологічні аспектів в управлінні життєвим циклом ПЗ.

До інженерії ПЗ відносяться як технічні питання розробки ПЗ, так і управління проєктами, накопичення знань, розробка та адаптація спеціальних методів і теорій. Розмаїття методів розробки створює певні проблеми з їх описом, розумінням та вивченням, подолати які може допомогти Essence Theory of Software Engineering [8]. Актуальною як для освіти так і для інженерії програмного забезпечення є задача розвитку логіко-смислових моделей представлення знань [9].

Управління конфігурацією персоналу з врахуванням необхідних компетенцій має важливе значення у будь-яких проєктах, особливо у розробці ПЗ [10]. Організація командної роботи в конкретному контексті гнучкої розробки програмного забезпечення має проблеми з комунікацією, навчанням, визначенням пріоритетів і лідерством [11]. В роботі [12] додатково підкреслюється важливість вдосконалення управління мультикультурними гнучкими командами розробки ПЗ, щоб забезпечити їх ефективність в умовах культурного розмаїття.

Забезпечення високої якості продукту неможливе без забезпечення відповідної якості цих і багатьох інших процесів в компаніях-розробниках, в тому числі не пов’язаних безпосередньо зі створенням ПЗ. В цьому контексті визначають поняття управління життєвим циклом підприємства, у якому необхідно подбати про всі життєві цикли підприємства, будь то життєвий цикл продукту, життєвий цикл проекту чи будь-який інший життєвий цикл [5]. Підходи до управління життєвим циклом ПЗ мають враховувати вплив внутрішнього та зовнішнього середовища ІТ-компанії.

Розвиток штучного інтелекту протягом останніх років робить актуальним як його вдосконалення та використання в системах управління життєвим циклом ПЗ, але поки немає суттєвих практичних напрацювань [13].

1.3. Мета, завдання та структура

Метою є підвищення ефективності командної розробки програмного забезпечення, а також забезпечення розвитку практик, методів та методологій програмної інженерії.

Завдання:

а) аналіз особливостей управління проєктами з розробки ПЗ, враховуючи специфіку ПЗ для аерокосмічної галузі;

б) дослідження стандарту OMG Essence стосовно опису методів та практик інженерії програмного забезпечення;

в) розробка процедури управління життєвим циклом ПЗ з використанням мови та ядра Essence.

Структура роботи: у розділі 2 розглянуто особливості управління проєктами з розробки ПЗ, дослідження стандарту OMG Essence, у розділі 3 наведено результати дослідження, процедуру управління життєвим циклом ПЗ з використанням мови і ядра Essence, приклад реалізації, розділ 4 містить обговорення, а розділ 5 – висновки та перспективи подальших досліджень.

2. Управління життєвим циклом ПЗ

2.1. Проєктний та системний погляди на розробку

Розробку та експлуатацію програмного забезпечення можна розглядати як проект або програму. Проєкт - тимчасова діяльність, спрямована на створення унікального продукту, послуги або результату [14]. Проєкти можуть бути як самостійними, так і частиною програми або портфелю. Програма - це сукупність взаємопов'язаних проєктів, а також комплекс організаційних змін, які об'єднані спільними цілями і спрямовані на отримання комерційної вигоди [15].

Програми або проєкти можуть бути ініційовані в будь-який момент життєвого циклу ПЗ для створення або покращення конкретних компонентів, функцій або можливостей. Упродовж життєвого циклу ПЗ програма або проєкт можуть додавати або покращувати певні компоненти, атрибути або можливості, які створюють додаткову цінність як для замовників так і для організації-виконавця.

Життєвий цикл продукту (Product Life Cycle) - серія фаз, які відображають еволюцію продукту від концепції через постачання, зростання, зрілість до виходу з ринку [15]. Життєвий цикл проєкту (Project Life Cycle) - послідовність фаз, через які проходить проєкт від свого початку до завершення [15].

Програма може охоплювати повний життєвий цикл ПЗ для безпосереднього управління вигодами та створення цінності для організації, яка є розробником ПЗ. Контроль за розвитком та вдосконаленням продукту може здійснюватися в межах поточної бізнес-діяльності організації. Таким чином проєкти реалізуються в межах більшої системи, наприклад ІТ-компанії.

Розробку програмних продуктів можна розглядати постачання цінності, але різні зацікавлені особи сприймають цінність по різному. Система постачання цінності є частиною внутрішнього середовища організації (рис. 1). Це внутрішнє середовище існує в більш широкому зовнішньому середовищі, що охоплює економіку, конкурентне середовище, законодавчі обмеження тощо. Цінність для організації-розробника - це отримання прибутку з продажу ПЗ, підвищення якості, розвиток технологій розробки тощо. Клієнти можуть визначити цінність як можливість використання певних функцій ПЗ для досягнення своїх бізнесових цілей, підвищенню ефективності своїх бізнес-процесів за рахунок автоматизації тощо.

Для того, щоб постійно задовольняти або перевищувати очікування учасників проекту, потрібне постійне балансування наступних конкуруючих вимог: вимог предметної області, часу, вартості та якості, а також вимог учасників проекту з їх різними потребами й очікуваннями. Життєвий цикл ПЗ можна розглядати як набір проєктів та управляти їм, використовуючи загальні принципи управління проєктами.

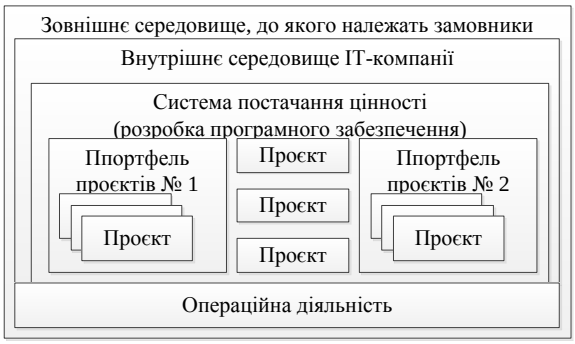


Рис. 1. Проєкти в ІТ-компанії

З точки зору системного підходу, проект може розглядатися як процес переходу з початкового стану в кінцеве - результат за наявністю обмежень і механізмів. Проєкт існує в динамічних умовах, проявляючи характеристики системи. Проєкт виконується в межах різних великих систем, а результат проєкту, у даному випадку програмне забезпечення, може стати частиною більшої системи для реалізації вигід у складі цієї системи.

Загальні підходи до управління проєктами можуть використовуватися для управління розробкою ПЗ, але в сфері програмної інженерії накопичено багато специфічних практик та методів, які доцільно інтегрувати до системи управління життєвим циклом.

2.2. OMG Essence

Міжнародним консорціумом відкритих стандартів OMG розроблено, підтримується та публікується стандарт, який призначений для створення, використання та вдосконалення практик і методів розробки програмного забезпечення [16]. Специфікацію Essence розроблено саме для інженерії програмного забезпечення, хоча загальні принципи можуть бути використані і в інших галузях. Essence використовує просту багаторівневу архітектуру, де методи складаються з практик.

Мова Essence дає змогу виражати практики в простій і стандартній формі, яка дозволяє легко ділитися, розуміти та застосувати практики як окремо, так і в поєднанні з іншими практиками Essence.

Мова Essence (Essence Language) - це предметно-спеціальна мова для визначення методів, практик і ядр (Essence Kernel) [16]. Мова має текстовий і графічний синтаксис, за допомогою якою уніфікується опис методів та практик.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Важливим є відокремлення ядра, що відображає суть розробки ПЗ, від практик, які після об'єднання утворюють метод.

Ядро (Essence Kernel) – це набір елементів, які використовуються для формування спільної основи для опису процесів розробки програмного забезпечення. Ядро Essence охоплює основні елементи розробки програмного забезпечення, які є невід'ємною частиною всіх відповідних методів. Ядро можна розглядати як модель домену, що визначає важливі поняття, які є загальними для всіх, хто працює в області розробки ПЗ. Essence Kernel можна використовувати для моніторингу прогресу та стану будь-якого програмного забезпечення, а також для аналізу сильних і слабких сторін способу роботи команди [16]. Ядро містить альфи, простори діяльності та компетенції (рис.2).

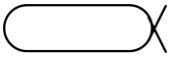
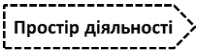
Альфа (Abstract-Level Progress Health Attribute, ALPHA)	
Простір діяльності (Activity Space)	
Компетенція (Competence)	

Рис. 2. Позначення елементів ядра мовою Essence

Щоб зберегти практичну незалежність, ядро не містить екземплярів інших мовних елементів, таких як робочі продукти або дії. Ядро розділено на три області інтересу: споживач (англ. Customer), рішення (англ. Solution) та діяльність (англ. Endeavor). Кожна область інтересу містить: Альфи, простори діяльності та компетенції. Запропоновані стандартом Альфи, їхні зв'язки та області інтересів показано на рис.3.

Альфа (англ. Abstract-Level Progress Health Attribute, ALPHA) – це суттєвий (англ. Essential) елемент проекту в галузі програмної інженерії, за яким потрібно стежити та оцінювати його просування. Альфи - це те, що змінюється у проекті, і зміни чого зацікавлені особи бажають розуміти, відстежувати, забезпечувати, спрямовувати та контролювати.

Альфа – це узагальнений опис того, чим команда керуватиме, вироблятиме та використовуватиме в процесі розробки та підтримки ПЗ. Альфи мають центральне значення для Essence, оскільки вони гарантують, що команда зосереджена на цінних результатах, а не на другорядних проблемах. Альфи не слід сприймати як фізичні продукти, це скоріше критичні показники того, що є найважливішим для моніторингу прогресу. Перелік альф та їх станів викладено в стандарті [16].

Кожна Альфа має невеликий набір попередньо визначених станів, які використовуються під час оцінки прогресу та здоров'я проєкту. Стан визначається шляхом відповіді на питання попередньо визначених для кожного стану контрольних списків (рис.4); стан досягнуто, якщо на всі питання надано позитивну відповідь.

Між станами можливий рух в обох напрямах, можливе повернення до попереднього стану. Можна циклічно повторювати стани кілька разів залежно від обраних практик.

Альфи не залежать від обраних командою практик і методів. Наприклад, зв'язок, показаний на рис. 3, згідно з яким команда виконує та планує роботу, не передбачає жодного конкретного порядку, у якому вони це роблять.

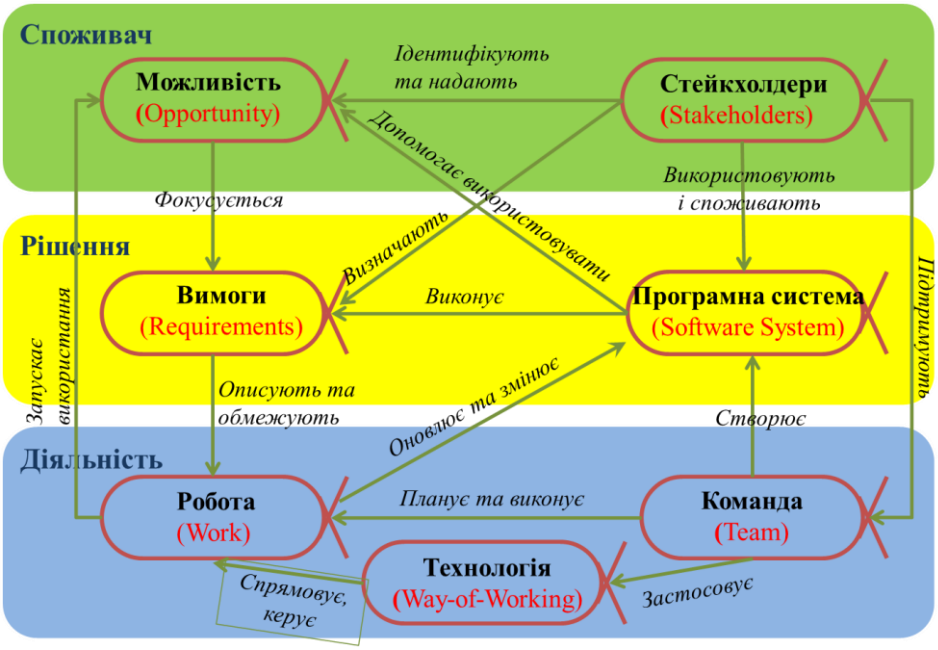


Рис. 3. Зв'язки між Альфами

Рис. 4. Визначення стану Альфи
за контрольними списками

Альфи можуть бути вкладені одна в одну, при цьому виконується важливе правило: просування за станами вкладеної Альфи (частини) означає якесь просування за станами більш загальної Альфи (цілого). Субальфи додають для того, щоб керувати прогресом Альфи, розвиваючи її частини (рис. 5). Стани Альф нижчого рівня сприяють і керують станами більш абстрактних Альф вищого рівня. Специфічні особливості аерокосмічного ПЗ та проєктів з його створення можна описати станами Субальф.

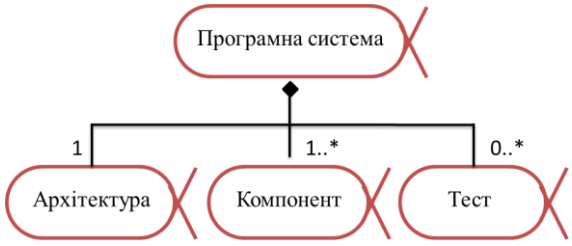


Рис. 5. Приклад Субальф

Основні етапи або віхи розробки програмного забезпечення можна виразити в термінах станів набору Альф. Таким чином, прогрес у альфа-стані означає прогрес у напрямку досягнення цілей розробки програмного забезпечення.

Ядро Essence визначає набір просторів діяльності, які разом визначають типи дій, які потрібно виконати, щоб просувати будь-яке завдання з розробки програмного забезпечення.

Простір діяльності (англ. Activity Space) – опис проблем, з якими стикається команда під час розробки, обслуговування та підтримки програмного забезпечення, а також дій, які команда робитиме, щоб їх вирішити. Ядро надає набір просторів діяльності, які доповнюють альфи, щоб забезпечити опис дій для розробки програмного забезпечення.

Діяльність (англ. Activity) - це виконання певної роботи однією або декількома людьми, наприклад «Задокументувати історії користувачів», «Уточнити беклог» тощо. Діяльність визначає один або кілька підходів до виконання певної роботи та може рекомендувати дії щодо альф та/або робочих продуктів для виконання цієї роботи.

Простір діяльності - це заповнювач (англ. Placeholder) для того, що потрібно зробити в програмній інженерії. (рис. 6). Діяльності, що заповнюють той самий простір діяльності, спільно сприяють досягненню критеріїв завершеності простору діяльності. Після завершення роботи стани альф оновлюються. Коли відбувається зміна стану альфи, це означає що простір діяльності завершено. Критерії завершення простору діяльності, виражаються в термінах, яких станів мають досягти вихідні альфи.

Рис. 6. Діяльність та простір діяльності

Практичні дії можна розміщувати в просторах діяльності, щоб вказати, яких загальних результатів практика та її діяльність допоможуть досягти команді. Простори діяльності ядра, які перелічені у стандарті, показано на рис. 7 [16].

Для просторів діяльності визначають критерії входу та виходу. Критерій - визначає умову, яка повинна бути виконана, щоб увійти в діяльність або простір діяльності; або вважати роботу в діяльності чи просторі діяльності завершеною.

Ядро надає набір компетенцій, які доповнюють альфи та простори діяльності, щоб забезпечити уявлення про ключові здібності або можливості, необхідні для виконання роботи з розробки ПЗ (рис. 8).

Компетенція (Competence) – це представлення ключових можливостей, необхідних для виконання роботи з розробки програмного забезпечення [16]. Компетенції охоплює здібності, досягнення, знання та навички, необхідні для виконання певного виду роботи. Важливо розуміти, з якими компетенціями потрібні спеціалісти для розробки ПЗ.

Кожна компетентність має п’ять рівнів досягнення: «Допомагає (Assists)», «Застосовує (Applies)», «Опановує (Masters)», «Адаптує (Adapts)», «Оновлює (Innovates)». Вони є стандартними для всіх компетенцій ядра. Рівень компетентності визначає, який рівень можливостей має член команди щодо компетенції. Вищі рівні компетенції спираються на нижчі.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

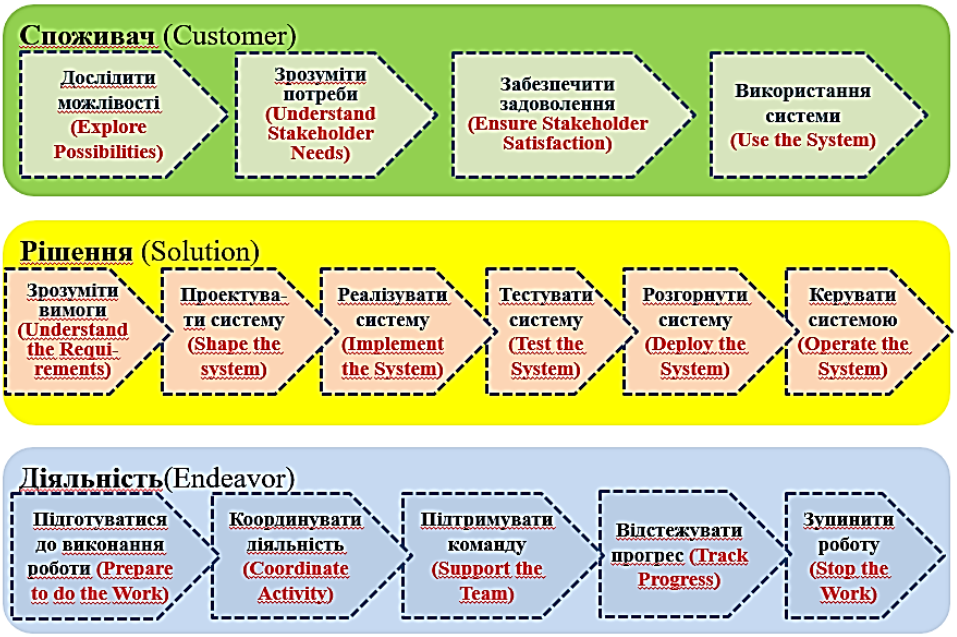


Рис. 7. Простори діяльності

Рис. 8. Компетенції

Альфи, стани Альф, Простори діяльності та Компетентності використовуються для опису вмісту ядра. Вважається достатнім знати ці чотири елементи, щоб мати можливість говорити про стан, прогрес і здоров'я розробки програмного забезпечення.

Альфа часто проявляється у вигляді колекції робочих продуктів, які використовуються для документування та презентації альф.

Робочий продукт - це артефакт, який має цінність і актуальність для розробки програмного забезпечення. Робочі продукти представляють конкретні речі, з якими потрібно працювати, надаючи докази станів альф. Робочим продуктом може бути документ або частина програмного забезпечення, а також інші створені сутності, як-от: створення тестового середовища або проведення навчального курсу тощо. Робочий продукт - це конкретне представлення Альфи, що характеризується ступенем деталізації. Діяльність надає чіткі вказівки щодо того, як створювати або оновлювати робочі продукти, що зрештою призведе до змін стану деяких альф (рис. 9). Маніфест робочого продукту прив'язує робочий продукт до альфи. Це зв'язок від практики до робочого продукту, який використовується для опису альфи. Кілька робочих продуктів можуть бути прив'язані до однієї альфи.

Рис. 9 пояснює сутність зав'язків між елементами ядра. Стани Альф дають змогу описати поточний та бажаний стан розробки програмного забезпечення. Простори діяльності та дії описують, що необхідно робити, щоб перевести систему розробки ПЗ у бажаний стан. А компетенції описують які спеціалісти для цього потрібні.

Рис. 9. Зв'язок елементів ядра

У якості основного способу управління життєвим циклом ПЗ та прийняття рішень командами розробки у стандарті пропонується використовувати набори карток Alpa State Cards. Підходи до оцінки стану проєку, планування та контролю описано у вигляді карткових ігор з Alpa State Cards, що не зручно в умовах віддаленої роботи.

Для опису методів та практик пропонується використовувати мову Essence. Специфікація мови створена за допомогою комбінації трьох різних методів: метамоделі, формальної мови та природної мови. Метамодель базується на чотирьох мета-рівнях конструкцій:

- рівень 3 - мета-мова: різні конструкції, які використовуються для вираження цієї специфікації, якот «мета-клас»;
- рівень 2 - конструкція: мовні конструкції, тобто різні типи конструкцій, виражені в цій специфікації, наприклад «Альфа» та «Діяльність»;
- рівень 1 - тип: елементи специфікації, тобто елементи, виражені в конкретних ядрах і практиках;
- рівень 0 - екземпляри під час виконання, тобто це представлення елементів реального життя в процесі розробки ПЗ[16].

Всі конструкції мови знаходяться на рівні 2. Інженер методів, який використовує мову Essence для розробки практик і пов'язаних із ними дій, робочих продуктів тощо, працюватиме на рівні 1.

Метод (англ. Method) в OMG Essence є композицією практик, які описані за допомогою ядра Essence та мови Essence. Порядок виконання роботи визначається використаним екземпляром методу. Методи - це не просто описи для читання розробниками, вони динамічні та підтримують повсякденну діяльність. У будь-який момент часу під час розробки програмного забезпечення можна звернутися до методу та отримати поради щодо того, що робити далі. Метод можна адаптувати в будь-який момент часу і отримати альтернативну пораду щодо того, що робити далі в тій самій ситуації [16].

Практика (англ. Practice) - це повторюваний підхід до виконання чогось із певною метою, який можна використовувати самостійно або в поєднанні з іншими практиками. Практика забезпечує систематичний і перевірений спосіб вирішення певного аспекту поточної роботи. Практики можна змішувати та поєднувати для створення конкретних методів розробки програмного забезпечення, адаптованих до конкретних потреб конкретної команди, організації розробників або окремого проєкту з розробки ПЗ. Практика може бути частиною багатьох методів [16].

Метод має дозволяти команді експериментувати та розвивати спосіб роботи, який відповідає потребам команди при виконанні роботи. Метод члени команди можуть постійно перевіряти та адаптувати. Набір практик, які використовує команда, може змінюватися з часом, оскільки їхні програмні системи розвиваються і спосіб роботи вдосконалюється. Використання ядра і мови Essence дозволяє об'єднати практику з іншими практиками для формування методу «вищого рівня».

Методи та практики описані в стандарті мовою Essence, однотипним та зрозумілим способом [16]. У якості ще одного прикладу використання мови Essence можна розглядати інструкцію з опису практики «USE-CASE 3.0» [17].

Композиція та модифікація мовних конструкцій здійснюється за допомогою операцій злиття та розширення в мові Essence. Розширення - це модифікація або налаштування елемента відповідно до нового контексту. Об'єднання (композиція) - це здатність об'єднувати елементи, щоб створювати потужніші елементи з простіших [16].

Є можливість поєднувати та розвивати різноманітні практики, серед яких є і популярні зараз практики гнучкої розробки ПЗ. В статті [18] обговорюються переваги та проблеми гнучкого програмування при використанні у великомасштабній розробці програмного забезпечення. Ядро OMG Essence пропонується використовувати в поєднанні з процесом Agile для своєчасної перевірки стану процесу, інструментів, процедур і ресурсів. Scrum - це найбільш використовувана практика гнучкої розробки ПЗ. У статті [19] представлено приклади і тематичні дослідження того, як використовують Scrum разом з OMG Essence. У статті [20] досліджується взаємодія Scrum і Essence в межах доменної моделі процесів розробки ПЗ, яка може стати спільною основою для методів розробки ПЗ, що складаються з окремих практик.

Команда з розробки ПЗ працює на рівні 0 метамоделі (див.рис.10), використовуючи методи та практики у своїх проєктах. Операційна семантика стосується того, як створюється модель рівня 0, як відстежується стан завдання в моделі та як модель може використовуватися для надання порад стосовно того, як прогресувати стан завдання. Виконання операційної семантики відбувається всередині середовища виконання.

Більша частина детальної інформації рівня 0 (про виконувану діяльність) може не міститися в самому інструменті Essence, а бути об'єднана в набір інструментів і середовищ, наприклад: інструменти планування та управління проєктами, інструменти управління вимогами, інструменти CASE та IDE, системи контроля версій, системи керування вмістом, сховища документів, електронних таблиць тощо.

Поняття примірника (екземпляра) відноситься до сутності всередині середовища виконання, яка представляє деяку сутність поза цим середовищем. Окрім екземплярів, що належать до моделі рівня 0, середовище виконання містить повну копію опису методу (тобто моделі рівня 1), вибраного для конкретного завдання. Будь-яка адаптація, внесена командою в опис методу під час роботи, також стосується лише цієї копії.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Ключова мета Essence полягає в тому, щоб мати можливість підтримувати «адаптацію». Практики та методи можуть бути адаптовані для задоволення конкретних потреб проекту. Необхідно, щоб така адаптація могла відбуватися в ході проекту, і це означає, що повинна бути можливість змінити модель методу рівня 1 в той час, коли екземпляри методу на рівні 0 ще виконуються. Оскільки модель рівня 0 завжди має бути дійсним екземпляром рівня 1, необхідно підтримувати їх в узгодженому стані.

Для автоматизації процесів управління відповідно до стандарту розробниками пропонується хмарний сервіс Essence WorkBench (https://workbench.ivarjacobson.com), який використовується для опису практик та методів, оцінки стану та управління проєктами з розробки програмного забезпечення. В роботі [21] розробляється фреймворк на основі OMG Essence для аналізу та керування розробкою технічних стартапів.

3. Результати

Пропонується використовувати ядро та мову Essence, а також практики, наведені в стандарті OMG Essence, для управління життєвим циклом ПЗ. Практики можуть об’єднуватися в методи та вдосконалюватися, а також можуть розроблятися нові.

На різних етапах розробки програмного забезпечення команда має потребу у підказках стосовно того, що робити далі. Коли загальний стан діяльності визначено, модель може бути використана для отримання таких порад. Для цього необхідно визначити поточний і бажаний стан проєкту в термінах станів Альф.

На рис.10 наведено результати оцінки в Essence WorkBench початкового стану проєкту розробки ПЗ, взятого для прикладу, за допомогою контрольних списків. Програма WorkBench має виключно англійський інтерфейс. На рис.10 ліва частина вікна містить Альфи та їх стани, зображені у вигляді карток. Коли обрано картку стану Альфи, на лівій панелі виводиться контрольний список, де слід обрати питання, на які дано позитивні відповіді. Надавши відповіді на запитання контрольних списків всіх станів всіх Альф, отримаємо оцінку стану проєкту.

Стандарт рекомендує певні дії для досягнення відповідного стану альф, що описано у вигляді таблиці відповідності дій певним станам альф. Частину таблиці прогресу станів (англ. State Progression Table) наведено на рис.11, де строки – це стани Альф, а стовпці – дії [22]. Червоні осередки вказують на дії, які дозволять просунути відповідний стан Альфи, а жовті – на дію, яка вплине на стан Альфи, але не обов’язково забезпечить досягнення.

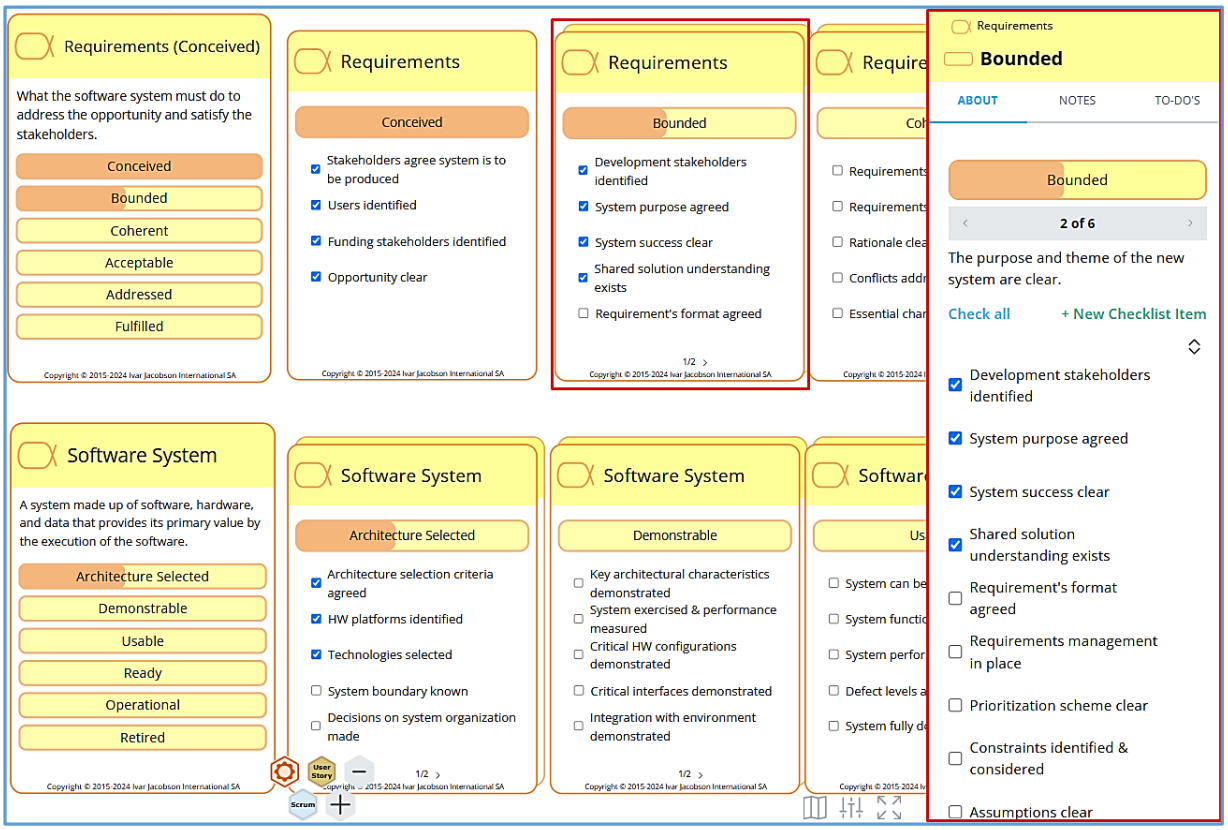


Рис. 10. Оцінка початкового стану проєкту в Essence WorkBench

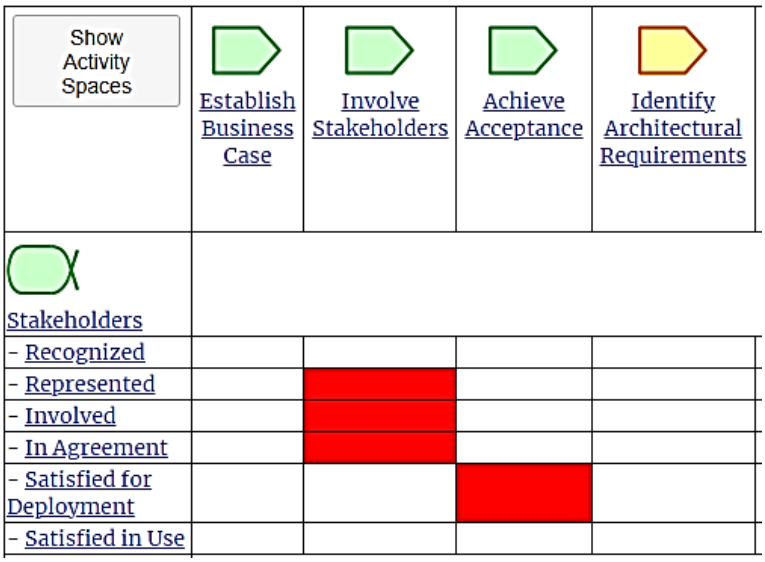


Рис. 11. Частина таблиці прогресу станів

Алгоритм роботи з таблицею:

1. Вибрали альфу та перевірити її стан.

2. Визначити стан, якого необхідно досягти.

3. За допомогою таблиці визначаємо дії, які рекомендує стандарт для переведення альфа у цей стан.

4. У стандарті ці дії описуються у вигляді практик [22].

В термінах станів альф можна описати етапи життєвого циклу ПЗ та використовувати такий опис для планування. Коли обсяг робот у проєкті досить великий доцільно теж у вигляді набору станів Альф визначити деякі проміжні етапи проєкту або віхи, щоб покращити планування та контроль виконання планів. Використовуючи State Progression Table, за необхідності доповнену власними діяльностями та практиками, можна планувати дії, які необхідні для реалізації проєкту.

В Essence WorkBench для розглянутого проєкту додано практики «User Story Essentials» та «Scrum Essentials», що призвело до додавання в модель додаткових Субальф робочих продуктів та дій до просторів діяльності. WorkBench надає можливість відстежувати вплив дій на стан Альф (Субальф) та деталізацію робочих продуктів. На рис. 12 в лівій частини вікна WorkBench зображено дії та їх вплив на відповідні стани Альф та Субальф. На правій панелі вікна надано опис обраної дії, включаючи умови для початку її виконання у вигляді станів Альф та необхідні компетенції виконавців та рівні їх компетентності.

Модель Essence не передбачає механізмів контролю за виконанням рекомендованих заходів, тому модель є розімкненою системою управління.

Життєвий цикл ПЗ має бути інтегрований у систему створення цінності організації, яка займається розробкою ПЗ. Стани альф «Робота (Work)», «Команда (Team)» та «Технологія (Way-of-Working)» сфери інтересів «Діяльність» (див. рис.3) у цьому випадку мають бути узгоджені з показниками ефективності роботи організації, в межах якої працює команда. Команди розробників організації використовують та розвивають спільні технології або підходи до роботи, відповідні інформаційні технології. На стан будь-якої команди впливає корпоративна культура, процеси розвитку та управління персоналом в організації.

З іншого боку, на стани альф «Можливість (Opportunity)» та «Зацікавлені сторони (Stakeholders)» сфери інтересів «Споживач (Customer)» (див. рис.3) мають бути узгоджені зі ступенем досягнення цілей організації – замовника ПЗ.

OMG Essence передбачає визначення станів Альф вручну за допомогою контрольних списків, а також використовуючи інформацію про стани відповідних Субальф. Контрольні списки Альф та Субальф можуть бути доповнені для врахування цілей та задач організації-розробника та організації замовника. Доповненням контрольних списків можна врахувати особливості проєктів та програмного забезпечення, що розробляється, а також використовуваних практик, в останньому випадку це стосується більшою мірою Субальф.

Якщо в організаціях замовника та/або виконавця впроваджено управління за ключовими показниками ефективності KPI або збалансовану систему показників [23], то до контрольних списків відповідних Альф можна додати питання, що базуються на кількісних показниках ефективності замовника або розробника.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Протягом життєвого циклу ПЗ вдосконалюються практики та методи. Модель Essence на першому рівні передбачає накопичення знань та вдосконалених практик для спільного використання різними командами.

Пропонується наступний порядок управління життєвим циклом ПЗ:

1. Визначення поточного стану проєкту з розробки ПЗ в термінах стану Альф.
2. Планування майбутніх етапів та контрольних точок в термінах стану Альф.
3. За допомогою State Progression Table визначення необхідних практик та дій для переведення життєвого циклу ПЗ у потрібні стани відповідно до послідовності етапів та контрольних точок.
4. Додання до опису життєвого циклу необхідних Субальф, визначення їх станів; за потреби доопрацювання практик та методів, або розробка нових з описом мовою Essence.
5. На основі опису практик та методів визначення необхідних ресурсів та витрат часу на виконання робіт.
6. Виконання командою необхідних дій та оцінка результатів в кожній контрольній точці.

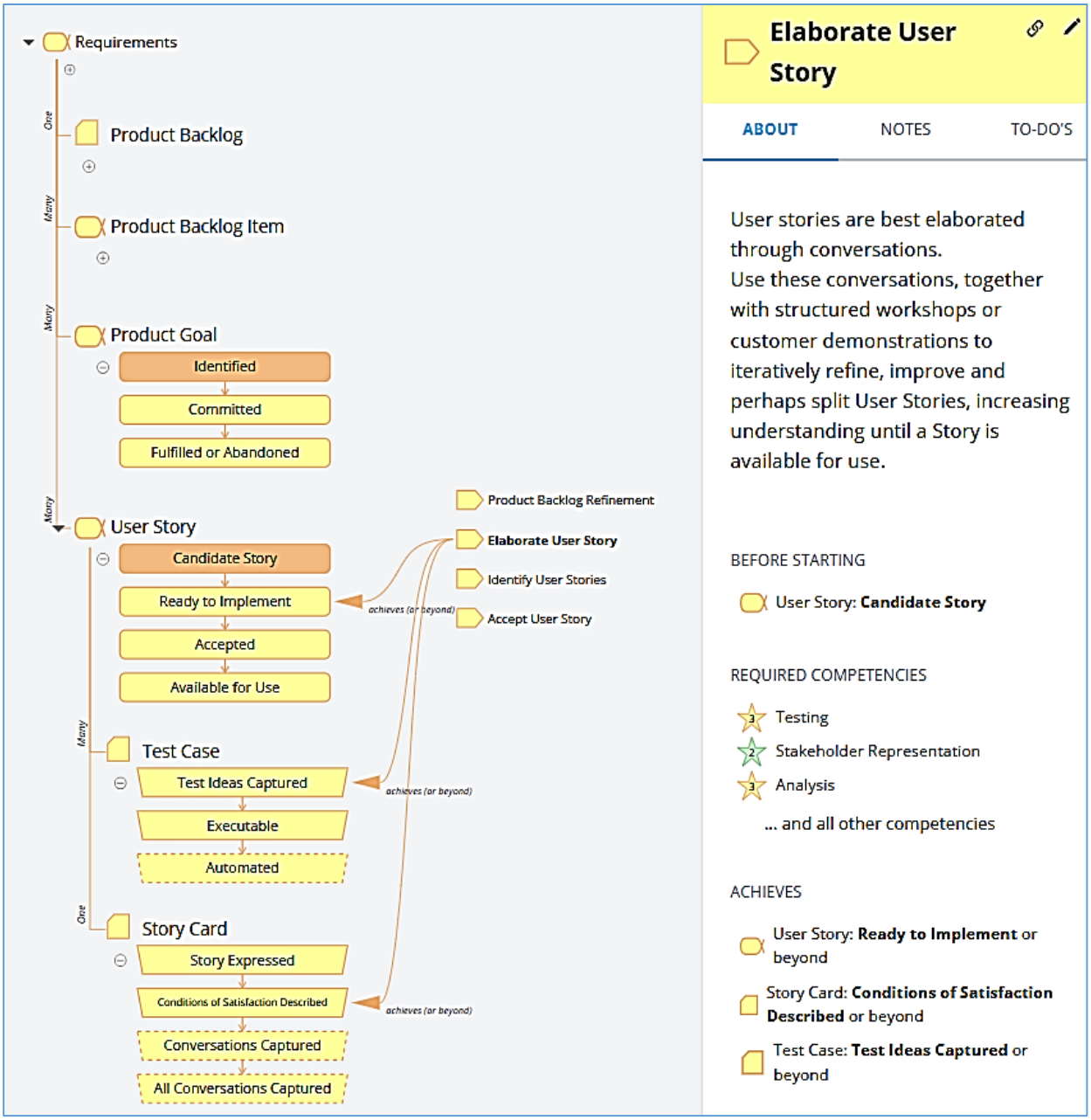


Рис. 12. Частина таблиці прогресу станів

7. Доповнення бази знань компанії модифікованими та новими практиками, якщо досвід їх використання виявився вдалим.

4. Обговорення

Ядро і мова Essence надає можливість описувати практики та методи на першому рівні метамоделі, використовувати екземпляри цих практик та методів на нульовому рівні метамоделі для управління реальними проєктами. Стандарт передбачає уніфікацію опису практик та методів, що робить простішим їх вивчення та використання. Для кожної практики, діяльності або простору діяльності описують вхідні та вихідні умови (стани Альф), що дозволяє визначить, як та чи інша дія змінює стан Альф.

Ядро Essence дозволяє оцінювати стан та здоров'я проєкта, використовуючи стани Альф, а також планувати проєкт, описуючи етапи та віхи також у вигляді зміни станів Альф. Знаючи поточний і бажаний стан Альф можна визначити діяльності та практики, які необхідні для переведення Альф у необхідний стан, який відповідає бажаному прогресу проєкту. Все це буде корисним для команд, що розробляють програмне забезпечення.

Стандарт OMG Essence передбачає розвиток методів та практик, а також створення нових практик під час виконання реальних проєктів з розробки ПЗ.

Розробники стандарту надають у використання хмарний інструмент Essence WorkBench, який дозволяє визначати поточний та бажаний стан проєктів, використовуючи контрольні списки. Є можливість планувати необхідні дії.

Essence WorkBench потребує якоїсь інтеграції з існуючими або перспективними інформаційними системами, які реалізують різні практики та методи, наприклад, системами управління проєктами, багтрекінговими системами тощо. Колективне оцінювання стану проєктів та прийняття рішень з використанням паперових Alpa State Cards наглядне та досить ефективне, коли команда працює в одному місці, і незручне у випадку розділених команд. Картки мають бути віртуальними, що потребує спеціального ПЗ.

5. Висновки

Розроблено комплексний підхід до управління життєвим циклом програмного забезпечення на основі стандарту OMG Essence. Предметно-спеціальна мова Essence, яка має текстовий і графічний синтаксис, використовується для уніфікованого опису методів та практик. Стандарт регламентує уніфікований та відносно простий спосіб опису методів та практик програмної інженерії. OMG Essence надає доступ до великої кількості практик розробки ПЗ, описаних мовою Essence. Практики та методи можуть бути адаптовані для задоволення конкретних потреб проекту. Можна розробити свої власні практики. З різних практик можуть бути скомпоновані методи.

Для моніторингу прогресу та стану програмного забезпечення, а також для аналізу сильних і слабких сторін способу роботи команди використовуються стани Альф ядра Essence. Альфа - це суттєвий абстрактний елемент проекту в галузі програмної інженерії, за яким потрібно стежити та оцінювати його просування. Набор станів Альф описує здоров'я та просування проєкту.

Запропоновано разом з контрольними списками, як це передбачено стандартом, для оцінки стану окремих Альф використовувати показники роботи (KPI) організації – замовника та компанії – розробника ПЗ. Це дозволить врахувати вплив внутрішнього середовища компанії – розробника та поточних цілей та потреб замовника на процес розробки та експлуатації ПЗ.

Планування та контроль виконання планів проєктів також можна здійснювати на основі станів Альф. Знаючи початковий і бажаний стан набору Альф за допомогою стандарту OMG Essence та спеціального хмарного сервісу можна визначити перелік та зміст дій для забезпечення необхідного просування проєкту. Таким чином спрощується управління життєвим циклом ПЗ.

Уніфіковані описи мовою Essence різних практик та методів доцільно використовувати також у навчальному процесі підготовки бакалаврів та магістрів в галузі інженерії програмного забезпечення. Це сприятиме вивченню студентами існуючих практик та методів розробки ПЗ.

У якості напрямів подальших досліджень слід розглянути питання створення спеціального програмного забезпечення для управління життєвим циклом ПЗ за стандартом OMG Essence, здатного інтегруватися з існуючими програмними інструментами, які реалізують практики розробки ПЗ. Слід дослідити можливість використання штучного інтелекту для управління життєвим циклом та вдосконалення методів розробки ПЗ.

Внесок авторів: формулювання проблеми та завдань – **А. В. Зеленков**; аналіз інформаційних джерел – **Л. В. Мандрікова**, **А. В. Зеленков**; підрозділ «Проєктний та системний погляди на розробку ПЗ» – **Л. В. Мандрікова**; підрозділ «OMG Essence» – **А. В. Зеленков**; апробація й аналіз результатів дослідження – **Є. Б. Боровков**; формулювання висновків та плану подальших досліджень – **А. В. Зеленков**.

Конфлікт інтересів

Автори заявляють, що у них немає конфлікту інтересів щодо цього дослідження, фінансового, особистого, авторського чи іншого, який міг би вплинути на дослідження та його результати, представлені в цій статті.

Фінансування

Дослідження проводилося без фінансової підтримки.

Доступність даних

Рукопис не має пов'язаних даних.

Використання штучного інтелекту

Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Усі автори прочитали та погодилися з опублікованою версією рукопису.

Література

1. Agile Software Development Lifecycle and Containerization Technology for CubeSat Command and Data Handling Module Implementation [Electronic resourse] / O. Liubimov, I. Turkin, V. Pavlikov, & L. Volobuyeva // Computation. – 2023. – Vol. 11, Iss. 9. – Article no. 182. DOI: 10.3390/computation11090182.

2. Jacobson, I. Escaping Method Prison – On the Road to Real Software Engineering [Electronic resource] / I. Jacobson, & R. Stimson // The Essence of Software Engineering / ed. by V. Gruhn, R. Striemer. – Springer International Publishing, 2018. – P. 51-58. DOI: 10.1007/978-3-319-73897-0.

3. Guide to the Software Engineering Body of Knowledge. Version 3.0 (SWEBOK) [Electronic resourse]. – IEEE, 2024. – 335 p. – Available at: https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOK v3. pdf. (accessed: 23.01.2025).

4. What is Application Lifecycle Management ALM: Definition. Best Tools. Complete Guide. [Electronic resourse]. – Available at: https://visuresolutions.com/blog/alm/#elementor-toc__heading-anchor-0%20 (accessed: 23.01.2025).

5. Грицюк, Ю. І. Система управління якістю програмного забезпечення [Текст] / Ю. І. Грицюк // Український журнал інформаційних технологій. – 2022. – Вип. 4, № 1. – С. 1-20. DOI: 10.23939/ujit2022.01.001.

6. Kozma, D. System of Systems Lifecycle Management – A New Concept Based on Process Engineering Methodologies [Electronic resourse] / D. Kozma, P. Varga, & F. Larrinaga // Applied Sciences. – 2021. – Vol. 11, Iss. 8. – Article no. 3386. DOI: 10.3390/app11083386.

7. Building Up Green Software Life Cycle Model [Electronic resource] / L. Kivimäki , L. Partanen, J. Porras, K. Tarkkanen, A. M .Tuikka, & J. M. Makela // 10th International Conference on ICT for Sustainability (ICT4S), Stockholm, Sweden, 2024. – IEEE, 2024. - P. 20-28. DOI: 10.1109/ICT4S64576.2024.00012.

8. The essence theory of software engineering– large-scale classroom experiences from 450+ software engineering BSc students [Electronic resourse] / K. K. Kemell, A. Nguyen-Duc, X. Wang, J. Risku, & P. Abrahamsso // 19th International Conference on Product-Focused Software Process Improvement (PROFES 2018), Wolfsburg, Germany, November 28 – 30, 2018. – Cham : Springer International Publishing, 2019. – P. 123-138. DOI: 10.1007/978-3-030-03673-7. – Available at: https://arxiv.org/pdf/1809.08827 (accessed: 23.01.2025).

9. Logical-semantic knowledge model for the knowledge base of a lecturer [Text] / S. Dotsenko, O. Banit, D. Nor, & O. Morozova // Radioelectronic and Computer Systems. – 2024. – No. 4. – P. 217-228. DOI: 10.32620/reks.2024.4.18.

10. Modeling the Transformation of Configuration Management Processes in a Multi-Project Environment [Electronic resourse] / N. Dotsenko, I. Chumachenko, A. Galkin, H. Kuchuk, & D. Chumachenko // Sustainability. – 2023. – Vol. 15, Iss. 19. – Article no. 14308. DOI: 10.3390/su151914308.

11. Strode, D. A teamwork effectiveness model for agile software development [Electronic resourse] / D. Strode, T. Dingsoyr, & Y. Lindsjorn // Empirical Software Engineering. – 2022. – Vol. 27. – Article no. 56. DOI: 10.1007/s10664-021-10115-0.

12. Navigating Cultural Diversity: Barriers and Benefits in Multicultural Agile Software Development Teams [Text] / D. Welsch, L. Burk, D. Mötefindt, & M. Neumann // SAC '24: 39th ACM/SIGAPP Symposium on Applied Computing Avila Spain, April 8 - 12, 2024. – Association for Computing Machinery, New York, US, 2024. – P. 818-825. DOI: 10.1145/3605098.3635988.

13. Petrin, N. S. MAISTRO: Towards an Agile Methodology for AI System Development Projects [Electronic resource] / N. S. Petrin, J. C. Néto, & H. C. Mariano // Applied Sciences. – 2025. – Vol. 15, Iss. 5. – Article no. 2628. DOI: 10.3390/app15052628.

14. Стандарт з управління проєктами та Настанова до зводу знань з управління проєктами (Настанова PMBOK) [Текст]. – Інститут з управління проєктами в Україні, 2021. – 275 с. – Режим доступу: https://pmiukraine.org/pmbok7 (дата звернення: 23.01.2025).

15. Довгань, Л. Є. Управління проектами: навчальний посібник [Текст] / Л. Є. Довгань, Г. А. Мохонько, І. П Малик. – К. : КПІ ім. Ігоря Сікорського, 2017. – 420 с.

16. Essence – Kernel and Language for Software Engineering Methods. OMG Document Number: ptc/24-06-07 [Electronic resource]. – OMG, 2024. – 286 с. – Available at: https://www.omg.org/spec/Essence (accessed: 23.01.2025).

17. Jacobson, I. USE-CASE 3.0 The Guide to Succeeding with Use Cases [Electronic resource] / I. Jacobson, I. Spence, & K. Mendonca. – Ivar Jacobson International SA, 2024. – 86 p. – Available at: https://www.ivarjacobson.com/publications/books/use-case-30-ebook (accessed:23.01.2025).

18. Jana, D. ESSENCE Kernel in Overcoming Challenges of Agile Software Development [Electronic resource] / D. Jana, & P. Pinakpani // IEEE 17th India Council International Conference (INDICON) 10-13 December 2020. New Delhi, India. – IEEE, 2020. DOI: 10.1109/INDICON49873.2020.9342375.

19. Sutherland, J. Scrum essentials cards: experiences of scrum teams improving with essence [Electronic resource] / J. Sutherland, I. Jacobson, & B. Kerr // Queue. – 2020. – Vol. 18, Iss. 3. – P. 83-106. DOI: 10.1145/3411757.3418775.

20. Better Scrum through Essence [Electronic resource] / I. Jacobson, J. Sutherland, B. Kerr, & B. Buhnova // Software: Practice and Experience. – 2022. – Vol. 52, Iss. 6. – P. 1531-1540. DOI: 10.1002/spe.3070.

21. Shkirenka, I. Adapting systems engineering to evaluate tech startups: an innovative framework based on OMG Essence [Electronic resource] / I. Shkirenka // The American Journal of Engineering and Technology. - 2024. – Vol. 06, Iss. 11. – P. 54-62. DOI: 10.37547/tajet/Volume06Issue11-07.

22. Practice Library [Electronic resource]. – Available at: https://practicelibrary.ivarjacobson.com (accessed: 23.01.2025).

23. Рєпіна, І. М. Збалансована система показників у системі управління якістю на підприємстві [Текст] / І. М. Рєпіна // Вісник Національного технічного університету "ХПІ" (економічні науки): зб. наук. пр. – Х. : НТУ «ХПІ», 2023. – № 3. – С. 68-72. – Режим доступу: https://repository.kpi.kharkov.ua/items/dc8812de-270d-48bc-b4ea-8e8366ec34c7 (дата звернення: 23.01.2025).

References

1. Liubimov, O., Turkin, I., Pavlikov, V., & Volobuyeva, L. Agile Software Development Lifecycle and Containerization Technology for CubeSat Command and Data Handling Module Implementation. *Computation*, 2023, vol. 11, iss. 9, article no. 182. DOI: 10.3390/computation11090182.

2. Jacobson, I., & Stimson, R. *Escaping Method Prison – On the Road to Real Software Engineering*. The Essence of Software Engineering, Springer International Publishing, 2018, pp. 51-58. DOI: 10.1007/978-3-319-73897-0

3. *Guide to the Software Engineering Body of Knowledge. Version 3.0 (SWEBOK)*. IEEE, 2024. 335 p. Available at: https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf. (accessed: 23.01.2025)

4. *What is Application Lifecycle Management ALM: Definition. Best Tools. Complete Guide*. Available at: https://visuresolutions.com/blog/alm/#elementor-toc__heading-anchor-0%20 (accessed: 23.01.2025).

5. Hrytsyuk, Yu. I. Systema upravlinnya yakistyu prohramnoho zabezpechennya [Software quality management system] *Ukrayins'kyy zhurnal informatsiynykh tekhnolohiy - Ukrainian Journal of Information Technology*, 2022, vol. 4, no. 1, pp.1-20. DOI: 10.23939/ujit2022.01.001. (In Ukrainian).

6. Kozma, D., Varga, P. & Larrinaga, F. System of Systems Lifecycle Management – A New Concept Based on Process Engineering Methodologies. *Applied Sciences*, 2021, vol. 11, iss. 8, article no. 3386. DOI: 10.3390/app11083386.

7. Kivimäki, L., Kivimäki, L., Partanen, L., Porras, J., Tarkkanen, K., Tuikka, A. M., & Makela, J. M. Building Up Green Software Life Cycle Model. *Proceedings of the 10th International Conference on ICT for Sustainability (ICT4S)*, Stockholm, Sweden, 2024. IEEE, 2024, pp. 20-28. DOI: 10.1109/ICT4S64576.2024.00012.

8. Kemell, K. K., Nguyen-Duc, A., Wang, X, Risku, J., & Abrahamsso, P. The essence theory of software engineering–large-scale classroom experiences from 450+ software engineering BSc students. *In Proceedings of the 19th International Conference on Product-Focused Software Process Improvement (PROFES 2018)*, Wolfsburg, Germany, November 28–30, 2018. Cham : Springer International Publishing, 2019, pp. 123-138. DOI: 10.1007/978-3-030-03673-7. Available at: https://arxiv.org/pdf/1809.08827 (accessed: 23.01.2025)

9. Dotsenko, S., Banit, O., Nor, D. & Morozova, O. Logical-semantic knowledge model for the knowledge base of a lecturer. *Radioelectronic and Computer Systems.* 2024, no. 4, pp. 217-228. DOI: 10.32620/reks.2024.4.18.

10. Dotsenko, N., Chumachenko, I., Galkin, A., Kuchuk, H., & Chumachenko, D. Modeling the Transformation of Configuration Management Processes in a Multi-Project Environment. *Sustainability*, 2023, vol. 15, iss. 19, article no. 14308. DOI: 10.3390/su151914308.

11. Strode, D., Dingsøyr, T., & Lindsjorn, Y. A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, 2022, vol. 27, article no. 56. DOI: 10.1007/s10664-021-10115-0.

12. Welsch, D., Burk, L., Mötefindt, D., & Neumann, M. Navigating Cultural Diversity: Barriers and Benefits in Multicultural Agile Software Development Teams. In *Proceedings of the SAC '24: 39th ACM/SIGAPP Symposium on Applied Computing Avila*, Spain, April 8 - 12, 2024. Association for Computing Machinery, New York, US, 2024, pp. 818-825. DOI: 10.1145/3605098.3635988.

13. Petrin, N. S., Néto, J. C., & Mariano, H. C. MAISTRO: Towards an Agile Methodology for AI System Development Projects. *Applied Sciences*, vol. 15, iss. 5, article no. 2628. DOI:10.3390/app15052628.

14. *Standart z upravlinnya proyektamy` ta Nastanova do zvodu znan` z upravlinnya proyektamy` (Nastanova PMBOK)* [Project Management Standard and Guide to the Project Management Body of Knowledge (PMBOK Guide)]. Project Management Institute Ukraine,` 2021. 275 p. Available at: https://pmiukraine.org/pmbok7 (accessed: 23.01.2025). (In Ukrainian).

15. Dovgan`, L. Ye., Moxon`ko, G. A., & Maly`k. I. P. *Upravlinnya proektamy`: navchal`ny`j posibny`k. [Project Management: A Tutorial]*. Kyiv, KPI im. Igorya Sikors`kogo Publ., 2017. 420 p. (In Ukrainian).

16. *Essence – Kernel and Language for Software Engineering Methods. OMG Document Number: ptc/24-06-07*. OMG, 2024. 286 p. Available at: https://www.omg.org/spec/Essence (accessed: 23.01.2025).

17. Jacobson, I., Spence, I., & Mendonca, K. *USE-CASE 3.0 The Guide to Succeeding with Use Cases*. Ivar Jacobson International SA, 2024. 86 p. Available at: https://www.ivarjacobson.com/publications/books/use-case-30-ebook (accessed: 23.01.2025).

18. Jana, D., & Pinakpani, P. ESSENCE Kernel in Overcoming Challenges of Agile Software Development. *In Proceedings of the IEEE 17th India Council International Conference (INDICON)*, 10-13 December 2020. New Delhi, India, IEEE, 2020, pp, 1-8. DOI: 10.1109/INDICON49873.2020.9342375.

19. Sutherland, J., Jacobson, I., & Kerr, B. Scrum essentials cards: experiences of scrum teams improving with essence. *Queue*, 2020, vol. 18, iss. 3, pp. 83-106. DOI: 10.1145/3411757.3418775.

20. Jacobson, I., Sutherland, J., Kerr B., & Buhnova, B. Better Scrum through Essence. *Software: Practice and Experience*, 2022, vol. 52, iss. 6, pp.1531-1540. DOI: 10.1002/spe.3070.

21. Shkirenka, I. Adapting systems engineering to evaluate tech startups: an innovative framework based on OMG Essence. *The American Journal of Engineering and Technology*, 2024, vol. 06, iss. 11, pp. 54-62. DOI: 10.37547/tajet/Volume06Issue11-07.

22. *Practice Library*. Available at: https://practicelibrary.ivarjacobson.com (accessed:23.01.2025).

23. Ryepina, I. M. Zbalansovana systema pokaznykiv u systemi upravlinnya yakistyu na pidpryyemstvi [Balanced scorecard in quality management system at the enterprise]. *Visnyk Natsional'noho tekhnichnoho universytetu "KhPI" (ekonomichni nauky): zb. nauk. pr. - Bulletin of the National Technical University "KhPI" (economic sciences).* Kharkiv, NTU «KhPI», 2023, no. 3, pp. 68-72. Available at: https://repository.kpi.kharkov.ua/items/dc8812de-270d-48bc-b4ea-8e8366ec34c7 (accessed: 23.01.2025). (In Ukrainian).

Надійшла до редакції 05.01.2025, розглянута на редколегії 17.03.2025

SOFTWARE LIFECYCLE MANAGEMENT BASED ON THE OMG ESSENCE STANDARD

Andrii Zelenkov, Ludmila Mandrikova, Yevhen Borovkov

Subject of Study: The article focuses on the processes of managing the software development lifecycle. The **goal** is to enhance the efficiency of team-based software development and to foster the advancement of practices, methods, and methodologies in software engineering. **Objectives:** to analyze the specifics of project management in software development, to study the OMG Essence standard regarding the description of software engineering methods and practices, and to develop a procedure for managing the software lifecycle using the Essence language and kernel. **Research Methods:** the study employs scientific and technical documentation analysis, system analysis, and the planning and modeling of the software development process. **Results.** A comprehensive approach to software lifecycle management has been developed based on the OMG *Essence* standard. The domain-specific *Essence* language, featuring both textual and graphical syntax, is used to describe software engineering methods and practices. Practices can be adapted or newly created to meet the specific needs of a project and the organizational structure of an IT company; methods can be composed of various practices. To monitor software progress and status, as well as to analyze the

strengths and weaknesses of a team's workflow, the use of *Alpha* states is proposed. An *Alpha* (Abstract-Level Progress Health Attribute, ALPHA) is a key abstract element of a software engineering project that must be tracked and assessed. It is suggested to plan and manage software development projects, based on *Alpha* states. By knowing the initial and desired states of a set of *Alphas*, the OMG *Essence* standard helps determine the list and content of actions needed to move the project forward. The software lifecycle management process was demonstrated using a software development project in *Essence WorkBench*, and this tool was thoroughly examined. **Conclusions.** A software lifecycle management procedure has been developed in accordance with the OMG *Essence* standard. It is proposed to use the performance indicators (KPIs) of both the client organization and the software development company alongside *Alpha* checklists to assess the state of individual *Alphas*. The study identifies ways to further develop software lifecycle management methodologies based on OMG *Essence*, including enhancing *Essence WorkBench* or creating a new tool capable of integrating with systems that automate various software engineering practices and project management processes.

Keywords: OMG Essence; software lifecycle management; software engineering; project management; competency.

Зеленков Андрій Вікторович – канд. техн. наук, доц., доц. каф. інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Мандрикова Людмила Василівна – канд. техн. наук, доц., доц. каф. інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Боровков Євген Борисович – асп. каф. інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Andrii Zelenkov – Candidate of Technical Sciences, Associate Professor, Associate Professor at the Department of Software Engineering, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: a.zelenkov@khai.edu, ORCID: 0000-0002-2163-4497, Scopus Author ID: 58615826600.

Ludmila Mandrikova – Candidate of Technical Sciences, Associate Professor, Associate Professor at the Department of Software Engineering, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: l.mandrikova@khai.edu, ORCID: 0000-0003-1781-1662.

Yevhen Borovkov – PhD Student at the Department of Software Engineering, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: bara26111986@ukr.net, ORCID: 0009-0000-9721-2189.