doi: 10.32620/oikit.2024.101.10

УДК 629.78:004.3

І. Б. Туркін, О. В. Любімов, І. В. Шевченко

Комбінована модель процесу розроблення апаратно-програмного модуля обробки даних студентського наносупутника CubeSat

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут»

Концепція наносупутників CubeSat змінила правила гри в галузі космічних наукових досліджень і розвитку новітніх космічних технологій. Основні фактори їх успіху: низька вартість, відносна простота виробництва і передбачуваний життєвий цикл. CubeSat дуже важливі для підготовки майбутніх інженерів: бакалаврів і магістрів аерокосмічних університетів. Тому використання CubeSat – це економічно вигідний спосіб дослідження ближнього космічного простору та проведення наукової роботи. Проте існує багато питань, пов'язаних з ефективним розробленням програмного забезпечення, забезпеченням якості програмного забезпечення на системному рівні, обслуговуванням і повторним використанням програмного коду. В роботі розглянуто переваги і недоліки як класичних, так і сучасних гнучких (Agile) моделей життєвого циклу розроблення програмного забезпечення. Для підвищення гнучкості та зменшення складності проєктів CubeSat запропоновано комбіновану модель розроблення апаратно-програмного модуля обробки даних, яка поєднує переваги двох моделей: «водоспадної» (waterfall) моделі для розроблення апаратного забезпечення і гнучкої моделі для розроблення програмного забезпечення. Для всебічної оцінки комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних використано методологію SWOT-аналізу, який є популярним інструментом стратегічного планування. Для його проведення були сформулювані переваги та недоліки, зовнішні можливості та загрози комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних. Запропонована комбінована модель дозволить студентським командам з малим досвідом розробляти програмно-апаратні модулі обробки даних з мінімальними ризиками, забезпечуючи повторне використання коду і збільшуючи привабливість студентських проєктів CubeSat.

Ключові слова: наносупутник, Cubesat, програмне забезпечення, апаратне забезпечення, «waterfall», Agile.

Вступ

CubeSat – це тип наносупутника, який почав свій комерційний і дослідницький шлях у 1999 році, коли в Каліфорнії було розроблено документ «CubeSat Design Specification» [1]. Від самого початку наносупутники CubeSat було започатковані як технологію для наукових досліджень та освітнього процесу. З 1999 року цей тип наносупутників здобув велику популярність як у комерційній та військовій промисловості [2], так і в академічних колах, зокрема в астрономії [3]. Доступність цієї технології для академічних цілей дозволили студентським та науковим групам отримати доступ до досліджень ближнього космосу та Землі з низької навколоземної орбіти (ННО). Типовий CubeSat – це наносупутник розміром від 1U (10x10x10 $см^3$) до 12U, коли декілька одиниць 1U можуть бути зібрані разом (складені або розміщені поруч один з одним). Відносна невелика вартість запуску наносупутників CubeSat пояснюється тим, що зазвичай наносупутники CubeSat доставляються на орбіту як «паразитний», тобто вторинний вантаж. Типовими програмами для таких «паразитних» запусків CubeSat є ініціатива NASA's (CLI - CubeSat Launch Initiative) [4] та програма

Європейського Космічного Агентства (European Space Agency, ESA) під назвою «Fly Your Satellite» (FYS) [5]. Для відділення наносупутників на орбіті від носія використовують спеціальні пристрої для розгортання пікосупутників (P-POD – Poly-Picosatellite Orbital Deployer) [6]. Ключові причини високої популярності [7], а також перспективного майбутнього стандарту та підходу CubeSat [8] полягає у:

1) низькій вартості таких супутників [9, 10];
2) відносно малих термінах на їх конструювання та випробування, що дозволяє побудувати, запрограмувати, протестувати та запустити супутник протягом навчання студентів, наприклад, на магістерському рівні;
3) стандартизації, яка забезпечує повторно використання як окремі частини супутників, так і наземних станцій для отримання телеметрії та управління супутниками на НІО.

Наведемо три типових завдання для наносупутників CubeSat:

1) дистанційне зондування Землі або інших космічних об'єктів;
2) комунікаційна інфраструктура (особливо для сузір'їв CubeSat-супутників, таких як OneWeb і StarLink);
3) дослідження проблем і задач повторного входження в атмосферу Землі.

Спираючись на статистику запусків та орбітальних доставок за останні кілька років [11, 12], [7], можна відмітити швидке зростання популярності та подальшого розвитку цієї галузі космічних технологій (рис. 1).

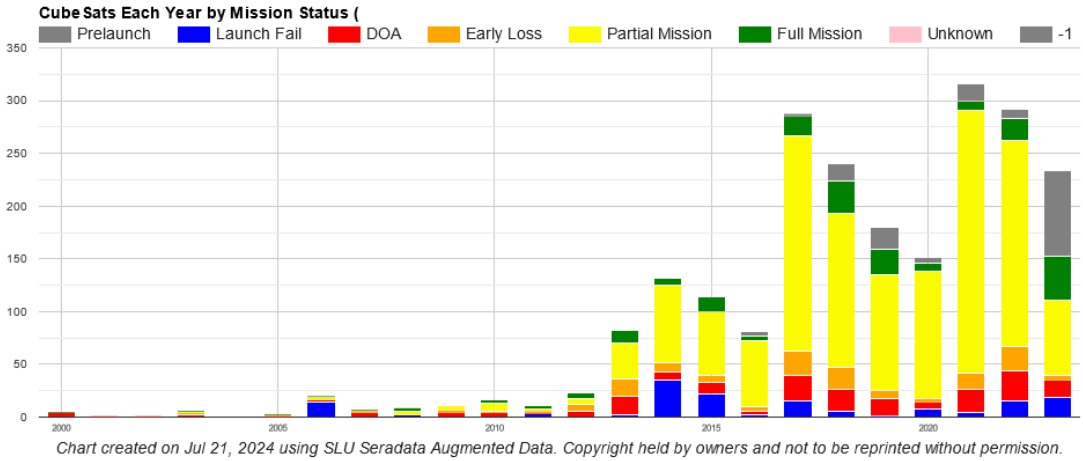

Рис. 1. Статистика запуску CubeSat станом на липень 2024 року [13]

Проєктування та розроблення наносупутників CubeSat є вельми перспективним напрямом розвитку аерокосмічних технологій, до цього завдання долучилось багато команд, що намагаються розробити супутник з нуля. На рис. 1 можна побачити, що лише у 2021 та 2022 роках було запущено майже 500 наносупутників CubeSat. Але також з рис. 1 видно, що на жаль, навіть маючи 20-річний досвід їх запусків і польотів, загальна статистика успішності запусків CubeSat залишається невтішною: приблизно 18,5% супутників у 2022 році були повністю втрачені, і лише 2,9% супутників повністю виконали своє завдання (повну місію). Багато наукових та комерційних команд по всьому світу намагаються мінімізувати ймовірність повної або часткової відмови супутника та максимізувати відсоток успішних стартів. Для цього задіються різноманітні наукові напрями: системний аналіз, теорія надійності, інженерія програмного забезпечення тощо.

Відомо, що розроблення програмного забезпечення займає приблизно 2/3 від усього часу на реалізацію проєкту CubeSat тому, що включає в себе визначення вимог, проєктування, кодування та процеси забезпечення якості (тестування, верифікацію та валідацію), а також наступну експлуатацію та супроводження програмного забезпечення. Завдання ускладняються тим, що необхідно гарантувати інформаційну безпеку: системи CubeSat, включаючи наземні станції, сигнали зв'язку та космічні апарати CubeSat, піддаються різним кібератакам, класифікацію яких надає стандарт ISO/IEC 15408 [14]. Існує безліч статей, які аналізують [15] і пропонують методи аналізу загроз безпеки CubeSat і вирішення цих проблем, наприклад, на основі аналізу дерев атак [16].

Багато досліджень зосереджено на ретельному підході до верифікації та валідації (V&V) програмного забезпечення [17], використовуючи складні моделі та техніки валідації даних [18]. Інші наукові роботи пропонують моделеорієнтовану розробку програмного забезпечення (Software In the Loop, SIL) та/або апаратного забезпечення (Hardware In the Loop, HIL) [19]. Також різні зусилля зосередженні в області механізмів імітації відмов (Failure Emulation Mechanisms, FEM) [20], а також впровадження різних платформ імітації помилок (Fault Injection Platforms, FIP) [21]. Зазначимо, що всі ці методи дуже типові для «водоспадного» життєвого циклу розроблення програмного забезпечення та загального управління проєктами, а також для строго та ретельно спланованих проєктів.

Однак, строгість та ретельність насправді не є типовими для академічних та дослідницьких проєктів, де зазвичай використовується процес «greenfield exploration», тобто створення нового програмного забезпечення з нуля, без обмежень, які накладають попередні версії чи існуюча інфраструктура. Проєкт з нуля – це низка проблем [22]:

- відсутність попередніх систем або існуючих специфікацій приводить до невизначеності у вимогах та очікуваннях;
- команда не має досвіду з обраними новими технологіями або фреймворками, що уповільнює процес розробки;
- потрібно створювати нову інфраструктуру з нуля, що вимагає значних ресурсів і часу;
- новий проект не має існуючої бази користувачів, тому зворотний зв'язок є обмеженим;
- управління проектом є складнішим через відсутність попередніх шаблонів або стандартів.

До типових методів вирішення проблем проєктування та реалізації складного програмного забезпечення відноситься:

- технічні методи проєктування та реалізації;
- методи покращення планування та організації процесів розроблення, так і покращення комунікації у команді розробників.

Технічні методи включають, але не обмежуються, такими техніками, як декомпозиція, забезпечення мінімальності, відділення коду від даних, належна ідентифікація абстракцій, повторне використання коду тощо. Процеси розроблення та комунікації зазвичай представлені сучасними життєвими циклами розроблення програмного забезпечення, принципами управління людьми та проєктами, а також так званою парадигмою свідомої співпраці - концепцією, яка відображає сучасний підхід до взаємодії між людьми, організаціями або країнами. Згідно з рекомендаціями NASA [23], дуже чітко

зазначено, що команди, які розробляють програмне забезпечення вперше, повинні: «спрощувати», «використовувати знайомі компоненти» та «не займатись безкінечним проєктуванням».

Єдина можливість досягнення успіху в таких умовах є використання комерційно доступних з «коробки» (Commercial-off-the-shelf, COTS) або модифікованих з «коробки» (Modifiable-off-the-shelf, MOTS) компонентів як у програмному, так і в апаратному забезпеченні типового проєкту розроблення CubeSat. Повторне використання компонентів COTS відкриває широкий спектр можливостей застосування не тільки програмного забезпечення з відкритим вихідним кодом, але й пропрієтарного програмного забезпечення [24, 25]. Необхідність повторного використання зазвичай вмотивовано наступним:

- досвідом успішної реалізації подібних проєктів іншими командами;
- обмеженими термінами реалізації проєкту;
- бажанням розробників зосередити зусилля на корисному навантаженні, а не на всій системі CubeSat.

Проєктні роботи навіть з великими об’ємами повторно використаного програмного забезпечення та відкритого вихідного коду залишаються значним викликом для команди розробників ще тому, що студенти бакалаврату або магістратури ще не мають достатньої практики роботи в галузі, не відповідають правилу «10 000 годин досвіду» [26], що призводить до проблем, пов’язаних з помилками в розумінні вимог, проєктуванні, кодуванні, забезпеченні якості тощо. Виконавці недостатньо обізнані з сучасними ефективними інструментами та технологіями створення надійного програмного забезпечення.

Використання програмного забезпечення COTS створює достатньо проблем у процесі розроблення та належного тестування бортового програмного забезпечення CubeSat. Багато модулів програмного забезпечення надаються як зразки або пропонуються до «використання на власний ризик», і тому не відповідають обґрунтованим вимогам до бортового та наземного програмного забезпечення, готового до впровадження. Це означає, що команди розробників вимушені вичерпне тестувати фінальний програмний продукт, щоби забезпечити його якість та готовність до польоту. І останній, але не менш важливий, фактор полягає в небезпечності «сліпого» повторного використання програмного забезпечення з відкритим вихідним кодом (Open-Source software, OSS), що призводить до низької якості та недостатньої зрілості вихідного коду в умовах браку часу на належний огляд всього розробленого програмного забезпечення.

Постановка задачі

Мотивація роботи. Існуючі проблеми створення якісного апаратно-програмного забезпечення студентських наносупутників потребують комплексного вирішення. Одним з напрямів вирішення є запровадження комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних студентського наносупутника CubeSat. В основу комбінованої моделі покладено поєднання двох відомих підходів: класичної «каскадної, або водоспадної» (waterfall) моделі для розроблення апаратного забезпечення і сучасної гнучкої (agile) моделі для розроблення програмного забезпечення. Комбінована модель дозволяє розділити складне завдання на відносно прості складові, з якими майбутні бакалаври і магістри зможуть працювати в межах своїх кваліфікаційних робіт.

Ціль роботи. З використанням SWOT-аналізу виконати всебічну оцінку

комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних.

Вибір мови програмування та операційної системи для реалізації CubeSat

Мова програмування С є вибором номер один і фактичним стандартом на ринку вбудованих (embedded) систем, а отже і для розроблення програмного забезпечення CubeSat, що легко пояснити наступним:

– мова програмування С має мінімальні накладні витрати, що є обов'язковим для систем з обмеженим енергетичним бюджетом;

– дозволяє використовувати як COTS (комерційні готові вироби), так і по-справжньому пропрієтарні апаратні платформи (процесори);

– типове програмне забезпечення CubeSat вимагає багато програмування на рівні апаратного забезпечення, і саме тут мова С є дуже потужною та ефективною;

– вимагає менше часу для початку розробки (у порівнянні з об'єктно-орієнтованими мовами, такими як C++ або Rust).

Надалі автори планують використовувати «Native C», що означає розроблення програмного забезпечення, яке написане на мові С і яке компілюється спеціальним компілятором для конкретної апаратної платформи. Іншими словами, це найбільш ефективно скомпільований вихідний код для конкретної апаратної платформи.

Щодо операційних систем (OS), які використовуються в індустрії CubeSat, то лідером вибору команд розробників є операційна система FreeRTOS, тому автори для використання обрали саме її.

Класичний життєвий цикл розроблення програмного забезпечення CubeSat

Загальноприйнятим життєвим циклом космічного проєкту, а отже і розроблення програмного забезпечення для нього, досі є «водоспадний», який представляє собою лінійно-послідовну модель життєвого циклу. Історично це була перша впроваджена модель життєвого циклу в інженерію програмного забезпечення. Вона дуже проста для розуміння і використання і полягає у розбитті проєктних завдань на лінійно-послідовні етапи, де кожен етап залежить від результатів попереднього і відповідає конкретній спеціалізації.

Перелічимо основні етапи водоспадної моделі:

1. Етап аналізу і специфікації вимог – етап, під час якого проєктна команда збирається разом, генерує та записує (специфікує) всі вимоги до проєкту. Етап зазвичай завершується, коли всі члени проєктної команди та замовник або клієнт погоджуються, що всі вимоги є остаточними і повністю визначеними. Після цього етапу вважається, що вимоги ніколи не будуть змінені і повністю готові до процесу проєктування та реалізації.

2. Етап проєктування – етап, під час якого інженерна команда займається дизайном (проєктуванням) складових проєкту, маючи вимоги, зібрані і оформлені на попередньому етапі. Зазвичай така фаза завершується двома документами – документом проєктування програмного забезпечення (Software Design Document, SDD) і документом проєктування апаратного забезпечення (Hardware Design Document, HDD). Якщо деякі з вимог не можуть бути виконані при проєктуванні, то даний етап вважається незавершеним, і проєкт повертається до попереднього

етапу аналізу вимог.

3. Етап реалізації (і модульного тестування) – основний етап, під час якого команда розробників безпосередньо реалізує апаратне і програмне забезпечення. Якщо щось не може бути реалізовано або модульно протестовано, то даний етап вважається незавершеним і проєкт повертається до попередніх етапів для доопрацювання проєктування або аналізу вимог.

4. Етап інтеграційного та системного тестування – етап, під час якого всі складові проєкту об'єднуються (інтегруються) для проведення тестування на системному рівні. Саме тут з'являється більшість непередбачених проблем і конфліктів через помилки на попередніх етапах проєктування або реалізації. Зазвичай цей етап є найважчим і, фактично, «відкриває очі» на дійсний стан проєкту.

5. Етап експлуатації та супроводження – етап, під час якого розроблене програмне забезпечення та/або загальна система створюють корисну цінність для замовника. На цьому етапі зазвичай можуть додаватися нові функції до програмного забезпечення, виправлятися знайдені під час експлуатації помилки та проводитися адаптація розробленого програмного забезпечення до змінюваного середовища.

«Водоспадна» модель життєвого циклу розроблення програмного продукту є спадщиною інженерії програмного забезпечення двадцятого сторіччя і рекомендована багатьма стандартами MIL-STD-499 [27], MIL-STD-1521 [28] і IEEE-15288 [29]. Рекомендований підхід до розроблення CubeSat у «CubeSat 101 Handbook» від NASA [30] також є послідовним, тобто «водоспадним». Це головним чином обумовлено практикою індустрії, а також наявністю фактичного апаратного забезпечення в проєкті CubeSat. Ці фактори приводять до «водоспадного» життєвого циклу розроблення всього проєкту. На рис. 2 наведена типова структура «водоспадної» моделі розроблення програмного забезпечення.

Проблема, яка виникає під час розроблення за «водоспадною» моделлю, полягає не лише у конвеєрному способі структурування завдань, але й у проєктуванні, реалізації, тестуванні та структуруванні вихідного коду.

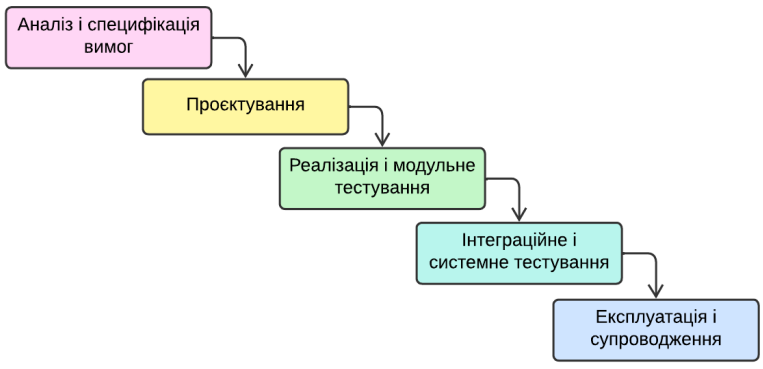

Рис 2. Типова структура «водоспадної» моделі SDLC

Аналіз існуючих відкритих проєктів CubeSat на GitHub показує, що переважно вихідний код є погано структурованим та достатньо примітивним. Навіть у тих проєктах, де використана модульна структуризація коду, програмне

забезпечення для CubeSat є монолітним, тісно пов’язаним, а дані є змішаними з кодом. Часто такий монолітний підхід успадковується від використаного фреймворку як частина стратегії повторного використання. Виходячи з обмежень часу на розроблення, а також обмежень на технічні вміння (hard-skills) у типовій команді розробників програмного забезпечення CubeSat, така монолітна і тісно пов’язана структура залишиться незміненою назавжди.

Подолати цей комплекс проблем можна тільки через вичерпне тестування, на чому й наполягає «CubeSat 101 Handbook» від NASA. Однак тестування та контроль якості вимагають розвинутих навичок, тому в більшості проєктів CubeSat тестування апаратного та програмного забезпечення відкладаються на останній момент і обмежуються лише інтеграційним і системним тестуваннями.

Більш досвідчені команди розробників (головним чином ті, які працюють в промислових компаніях), використовують так звану V-модель, де кожен етап життєвого циклу проєкту повинен бути належно перевірений після свого завершення. V-модель (рис. 3) широко використовується в проєктах, які виконуються відповідно до стандартів IEC 61508 [31] – «Electronic Functional Safety Package», IEC 62304 [32] / ISO 14971 [33] – «Medical Device Software», ISO 26262 [34] – «Automotive Functional Safety» та інші.

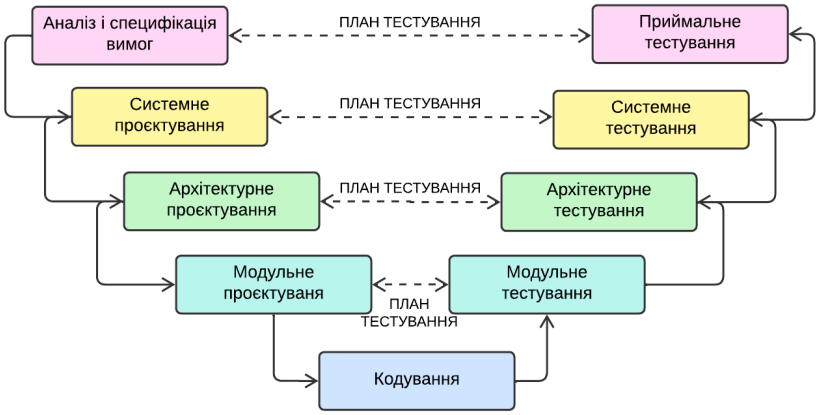

Рис 3. V-модель SDLC

При використанні V-моделі у SDLC команда розробників може краще розділити «Етап проєктування» та «Етап інтеграційного та системного тестування», розглянуті вище, і мати більше часу для проєктування/тестування системи на різних рівнях абстракції. Це, в свою чергу, допомагає краще уявити її функції та краще справлятися зі складністю функцій.

Однак, слід зазначити, що використання V-моделі не змінює того факту, що інтеграція відсувається на самий кінець процесу розробки і все ще потребує значних зусиль з реалізації та тестування на завершальних етапах. Використання такого підходу дає більш детальний огляд процесу верифікації та подальшої валідації програмного забезпечення, але не змінює ні мислення щодо проєктування, ні підходу до проєктування програмного забезпечення CubeSat. Загалом, як «водоспадна», так і V-модель широко критикуються індустрією розроблення програмного забезпечення через їх строгість, негнучкість і лінійність. Важливо підкреслити, що використання V-моделі не покращує швидкість інтеграції чи простоту вихідного коду, вона просто допомагає приділяти більше часу проєктуванню на різних рівнях абстракції.

Дисциплінована гнучка розробка програмного забезпечення CubeSat

На сьогоднішній день більшість так званих «великих» команд розробників програмного забезпечення прагнуть використовувати ітеративні та/або інкрементальні моделі розробки програмного забезпечення такі як SCRUM (рис. 4) або Kanban, що базуються на Agile, тобто гнучких, процесах.

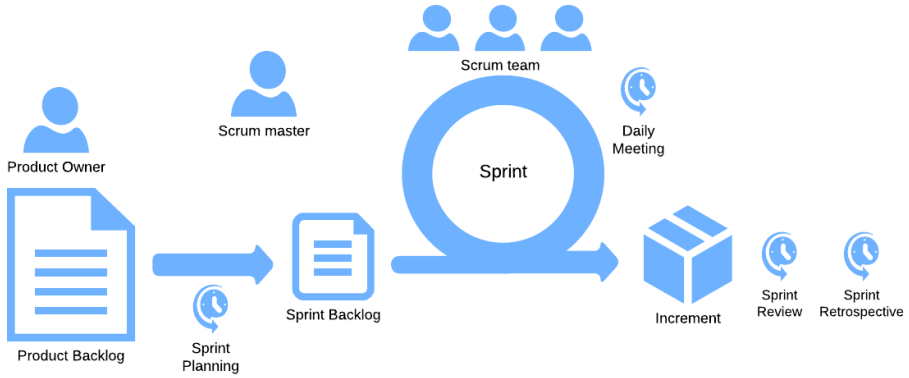

Рис. 4. Типова SCRUM-модель SDLC

При такому підході програмне забезпечення розбивається на окремі функції (features), які готові для реалізації, тестування та демонстрації як окремі автономні одиниці. Просту візуалізацію переваг Agile-моделі в порівнянні з «водоспадною» моделлю показано на рис. 5.

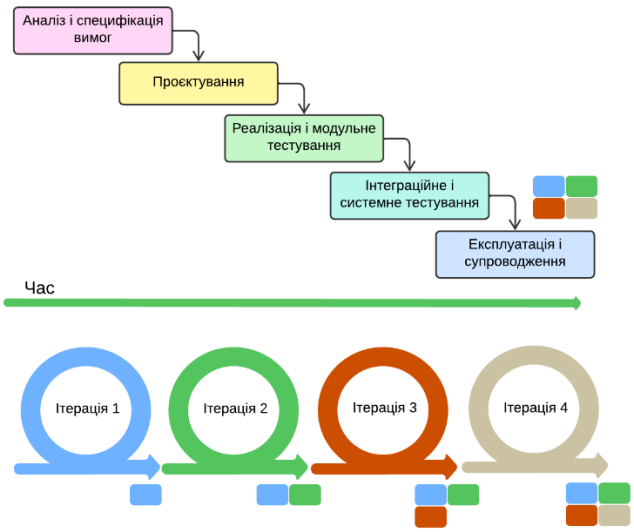

Рис. 5. Порівняння «водоспадної» та Agile- моделей SDLC

Аналізуючи «Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-critical Systems» [35], стає зрозумілим, що існує багато варіацій класичного SCRUM-процесу, які краще підходять для розробки місійно-критичного програмного забезпечення, та, відповідно, можуть бути застосовними для проєктів CubeSat і тому потребують більшого дослідження. Одним із

прикладів такої варіації може виступати фреймворк «Дисциплінований Agile» (Disciplined Agile, DA), який представлений на рис. 6.

З урахуванням виявлених проблем «водоспадної» моделі та переваг Agile-моделей (SCRUM, DA), автори пропонують для розроблення апаратної забезпечення CubeSat використовувати класичну «водоспадну» модель, тоді як для програмного забезпечення та загальної інтеграції наносупутника використовувати Agile-підхід. Таким чином, автори обґрунтовують комбіновану модель, яка стає все більш поширеною при розробленні вбудованих (embedded) електронних пристроїв.

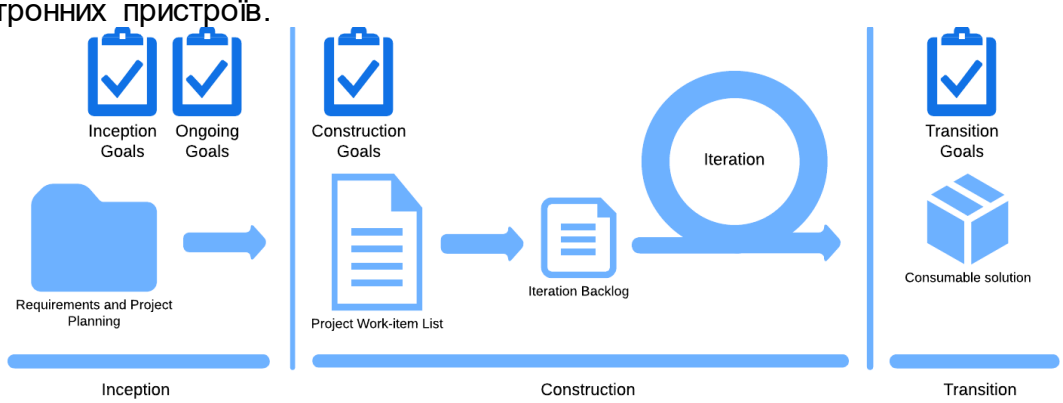

Рис. 6. Дисциплінований Agile для розробки місійно-критичних систем

SWOT-аналіз комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних

Зазвичай SWOT-аналіз використовується як інструмент стратегічного планування для структуризації знань про поточну ситуацію і тенденції, сильні та слабкі сторони, а також можливості і ризики, з якими організація може зіткнутися. SWOT є абревіатурою від англійських слів Strengths (сильні сторони), Weaknesses (слабкі сторони), Opportunities (можливості) та Threats (загрози).

У цій роботі методологію SWOT-аналізу використано для всебічної оцінки комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних. Для проведення SWOT-аналізу переваги та недоліки, зовнішні можливості та загрози комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних необхідно вписати у відповідні комірки SWOT-матриці (табл. 1). Загальною рекомендацією є визначення до десяти пунктів у кожній комірці, що забезпечить фокусування аналізу на найбільш важливих факторах.

Зазначені у табл. 1 можливості спрямовані на подолання розриву між традиційними методами розроблення апаратного забезпечення та гнучкими підходами до розроблення програмного забезпечення.

Виходячи з описаних підходів та загальних практик індустрії перелічимо конкретні інструменти для подолання зазначених ризиків і проблем:

1. Системи управління проєктами для відстеження завдань та прогресу як програмних, так і апаратних команд, наприклад, Jira або Trello,.
2. Інструменти для віртуалізації та емуляції апаратного забезпечення для створення віртуальних середовищ тестування, наприклад, QEMU або VirtualBox.
3. Системи контролю версій для управління кодом та документацією, можливо також із розширеннями для управління версіями апаратних компонентів, наприклад, Git.

Таблиця 1

Результати SWOT-аналізу

	Позитивний вплив	Негативний вплив
Внутрішнє середовище	**Сильні сторони** (Strengths) 1. Скорочення часу виходу на ринок за рахунок того, що Agile-методології дозволяють поступово нарощувати важливу для клієнтів функціональність. 2. Підвищення якості продукту завдяки постійним інтеграції та тестуванню. 3. Покращення комунікації між командами розробників, а також із замовниками. 4. Можливість адаптуватися до змін у проєкті на ранніх етапах. 5. Раннє виявлення проблем особливо на стику апаратного та програмного забезпечення. 6. Підвищення точності оцінок завдяки коротким ітераціям. 7. Постійна інтеграція та рефакторинг допомагають підтримувати якість коду. 8. Покращення управління ризиками завдяки ранньому виявленню проблем та постійному зворотному зв'язку. 9. Підвищення задоволеності клієнтів через можливість швидше отримувати робочі прототипи та вносити зміни. 10. Оптимізація ресурсів за рахунок більш ефективного використання часу та зусиль команди.	**Слабкі сторони** (Weaknesses) 1. Різні темпи розробки та оновлення складових: 2-4 тижні для програмної складової та до 6 місяців апаратної. 2. Вичерпне тестування програмного забезпечення може бути виконано тільки після завершення апаратного та вимагає відтворення реальних умов експлуатації 3. Апаратне забезпечення повинно відповідати строгим процесам розроблення та стандартам. 4. Пізні зміни, які вносяться в апаратне забезпечення, можуть бути дуже дорогими або взагалі неможливими. 5. Існує складність ідентифікації джерела проблеми при відпрацюванні: це проблема апаратного або програмного забезпечення. 6. У регульованих галузях, до яких відноситься і космічна, вимоги до документації зазвичай суперечать Agile-принципам. 7. Через різні підходи до визначення «готовності» маємо ситуацію, коли критерії завершення роботи для апаратного та програмного забезпечення можуть суттєво відрізнятися.
Зовнішнє середовище	**Можливості** (Opportunities) 1. Використання різних Agile-підходів, наприклад, застосування Scrum-моделі для розроблення програмного забезпечення і Commitment-Based Project Management (CBPM) для апаратного забезпечення [36]. 2. Використання модульної архітектури апаратного забезпечення для швидшого розроблення та тестування. 3. Створення абстрактного шару між програмним та апаратним забезпеченням для зменшення залежностей. 4. Одночасне розроблення програмного та апаратного забезпечення з використанням емуляторів або симуляторів для раннього тестування програмного забезпечення. 5. Постійна інтеграція та тестування за рахунок впровадження автоматизованого регресійного тестування. 6. Покращена комунікація команд розробників програмного та апаратного забезпечення через впровадження регулярних зустрічей. 7. Фокус на створенні мінімального життєздатного апаратного продукту (MVP) для забезпечення ранньої інтеграції з програмним забезпеченням. 8. Адаптивне планування за рахунок використання Constraint-based Planning для визначення критичних етапів та залежностей між проєктами. 9. Гнучкість у визначенні критеріїв «готовності», тобто адаптація критеріїв завершення робіт для різних частин проєкту.	**Ризики / загрози** (Threats) 1. Проблеми із синхронізацією розробки апаратної та програмної частини, а значить можливі затримки і проблеми несумісності. 2. Висока складність і ціна змін апаратного забезпечення, що може сповільнювати ітеративний процес. 3. Проблеми з якістю і надійністю продукту через зобов'язаність частих релізів, а значить можливого недостатньо тестування апаратної частини. 4. Неправильне розуміння вимог і цілей через відсутність ефективної комунікації між командами розроблення апаратного і програмного забезпечення. 5. Необхідність в адаптації та вдосконаленні інструментів і методологій, що добре працюють при розробленні програмного забезпечення і можуть не підходити для розроблення апаратного забезпечення. 6. Неповнота/незрозумілість вимог може призвести до розроблення функціональностей, які не відповідають потребам клієнта, що особливо критично для апаратного забезпечення. 7. Складність інтеграції у гнучкий процес розроблення регуляторних стандартів і вимог щодо апаратного забезпечення. 8. Складність відстеження змін і забезпечення їхньої сумісності через керування різними версіями апаратного та програмного забезпечення. 9. Неправильна оцінка часу для розроблення і тестування апаратного забезпечення, що може призвести до затримок і перевищення бюджету.

4. Інструменти для безперервної інтеграції та автоматизованого тестування, наприклад, Jenkins.

5. Інструменти для моделювання та проєктування апаратного забезпечення з можливістю інтеграції з системами управління проєктами, наприклад, CAD-системи.

6. Колаборaційні платформи для покращення комунікації між командами, наприклад, Slack або Microsoft Teams.

7. Інструменти для візуалізації процесів створення та спільного редагування діаграм та мав проєктів, наприклад, Miro.

8. Інструменти для аналізу даних та звітності, наприклад, Microsoft Power BI або Tableau.

9. Спеціалізовані Agile-інструменти для управління Agile-процесами. наприклад, VersionOne або ClickUp.

10. Системи документообігу для управління технічною документацією, особливо в регульованих галузях.

Висновки

Наносупутники CubeSat змінили правила гри в галузі наукових досліджень і розвитку новітніх космічних технологій. Основні фактори їх успіху: низька вартість, відносна простота виробництва і передбачуваний життєвий цикл. CubeSat дуже важливі для підготовки майбутніх інженерів: бакалаврів і магістрів аерокосмічних університетів. Тому використання CubeSat – це економічно вигідний спосіб дослідження ближнього космічного простору та проведення наукової роботи. Проте існує багато питань, пов’язаних з ефективним розробленням програмного забезпечення, забезпеченням якості програмного забезпечення на системному рівні, обслуговуванням і повторним використанням програмного коду. Для підвищення гнучкості та зменшення складності проєктів CubeSat запропонована комбінована модель розроблення апаратно-програмного модуля обробки даних, яка поєднує переваги двох моделей: «водоспадної» (waterfall) моделі для розроблення апаратного забезпечення і гнучкої (agile) моделі для розроблення програмного забезпечення.

Запропонована комбінована модель дозволить студентським командам з малим досвідом розробляти програмно-апаратні модулі обробки даних з мінімальними ризиками, забезпечуючи повторне використання коду і збільшуючи привабливість студентських проєктів CubeSat. Виконаний SWOT-аналіз комбінованої моделі процесу розроблення апаратно-програмного модуля обробки даних дозволяє отримати всебічну оцінки комбінованої моделі.

Розвитком роботи є перехід до кількісних нечітких моделей оцінювання ризиків проєкту.

Фінансування

Дослідження проведено в рамках проєкту «Експериментальне відпрацювання бортового обчислювача безпілотного літального апарата подвійного призначення» за фінансової підтримки Національного фонду досліджень України.

Список літератури

1. CubeSat.org. Cubesat Design Specification Rev 14.1 (by the CubeSat

Program) [Електронний ресурс]. – 2022. – Режим доступу: https://www.cubesat.org/cubesatinfo. – Назва з екрана (дата звернення: 15.07.2024).

2. Cappelletti, C. 2-CubeSat missions and applications / C. Cappelletti, D. Robson // Cubesat Handbook. – Academic Press, 2021. – С. 53–65. – Режим доступу: https://doi.org/10.1016/B978-0-12-817884-3.00002-3.

3. Shkolnik, E. L. On the verge of an astronomy CubeSat revolution / E. L. Shkolnik // Nature Astronomy. – 2018. – № 2. – С. 374–378. – Режим доступу: https://doi.org/10.1038/s41550-018-0438-8.

4. Crusan, J. NASA's CubeSat Launch Initiative: Enabling broad access to space / J. Crusan, C. Galica // Acta Astronautica. – 2019. – № 157. – С. 51–60. – Режим доступу: https://doi.org/10.1016/j.actaastro.2018.08.048.

5. ESA. European Space Agency. Fly Your Satellite Program Intro [Електронний ресурс]. – 2018. – Режим доступу: https://www.esa.int/Education/CubeSats_-_Fly_Your_Satellite/Fly_Your_Satellite!_programme. – Назва з екрана (дата звернення: 22.07.2024).

6. CalPoly. P-POD User Guide / California Polytechnic State University [Електронний ресурс]. – 2014. – Режим доступу: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/5806854d6b8f5b8eb57b83bd/1476822350599/P-POD_MkIIIRevE_UserGuide_CP-PPODUG-1.0-1_Rev1.pdf. – Назва з екрана (дата звернення: 24.07.2024).

7. Brycetech. Smallsats by the Numbers 2023 [Електронний ресурс]. – 2023. – Режим доступу: https://brycetech.com/reports/report-documents/Bryce_Smallsats_2023.pdf. – Назва з екрана (дата звернення: 25.07.2024).

8. Kang, J. Creating Future Space Technology Workforce Utilizing CubeSat Platforms: Challenges, Good Practices, and Lessons Learned / J. Kang, J. Gregory, S. Temkin, M. Sanders, J. King // Proceedings of the AIAA Scitech 2021 Forum. – 2021. – С. 1–12. – Режим доступу: https://doi.org/10.2514/6.2021-1437.

9. EXA. Cubesat Market. KRATOS 1U Platform [Електронний ресурс]. – 2021. – Режим доступу: https://www.cubesat.market/kratos1uplatform. – Назва з екрана (дата звернення: 26.07.2024).

10. Reznik, S. Comparison of geostationary and low-orbit «round dance» satellite communication systems / S. Reznik, D. Reut, M. Shustilova // IOP Conference Series: Materials Science and Engineering. – 2020. – № 971. – Режим доступу: https://doi.org/10.1088/1757-899X/971/5/052045.

11. Swartwout, M. Cubesat Database / M. Swartwout // Sant Louis University [Електронний ресурс]. – 2022. – Режим доступу: https://sites.google.com/a/slu.edu/swartwout/cubesat-database. – Назва з екрана (дата звернення: 25.07.2024).

12. Kulu, E. Nanosats Database / E. Kulu // NewSpace Index [Електронний ресурс]. – 2022. – Режим доступу: https://www.nanosats.eu/database. – Назва з екрана (дата звернення: 25.07.2024).

13. Swartwout, M. CubeSat Database / M. Swartwout [Електронний ресурс]. – Режим доступу: https://sites.google.com/a/slu.edu/swartwout/cubesat-database/census. – Назва з екрана (дата звернення: 21.07.2024).

14. ISO15408. ISO/IEC 15408-1:2022 Information security, cybersecurity and privacy protection – Evaluation criteria for IT security – Part 1: Introduction and general model [Електронний ресурс]. – 2022. – Режим доступу: https://www.iso.org/standard/72891.html. – Назва з екрана (дата звернення: 25.07.2024).

15. Potii, O. Advanced Security Assurance Case Based on ISO/IEC 15408 /

O. Potii, O. Illiashenko, D. Komin // Proceedings of the Theory and Engineering of Complex Systems and Dependability. – Springer International Publishing, 2015. – С. 391–401. – Режим доступу: https://doi.org/10.1007/978-3-319-19216-1_37.

16. Falco, G. CubeSat Security Attack Tree Analysis / G. Falco, A. Viswanathan, A. Santangelo // Proceedings of the 2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT). – 2021. – С. 68–76. – Режим доступу: https://doi.org/10.1109/SMC-IT51442.2021.00016.

17. Paiva, D. Enhanced software development process for CubeSats to cope with space radiation faults / D. Paiva, R. Lima, M. Carvalho, F. Mattiello-Francisco, H. Madeira // Proceedings of the 2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC). – 2022. – С. 78–88. – Режим доступу: https://doi.org/10.1109/PRDC55274.2022.00022.

18. Liubimov, O. Data Model and Methods for Ensuring the Reliability and Relevance of Data for the CubeSat Projects / O. Liubimov, I. Turkin // Proceedings of the 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). – 2022. – С. 1–7. – Режим доступу: https://doi.org/10.1109/DESSERT58054.2022.10018658.

19. Goyal, T. Simulator for Functional Verification and Validation of a Nanosatellite / T. Goyal, K. Aggarwal // Proceedings of the 2019 IEEE Aerospace Conference. – 2019. – С. 1–8. – Режим доступу: https://doi.org/10.1109/AERO.2019.8741886.

20. Batista, C.L.G. On the use of a failure emulator mechanism at nanosatellite subsystems integration tests / C.L.G. Batista, E. Martins, M. de Fatima Mattiello-Francisco // Proceedings of the 2018 IEEE 19th Latin-American Test Symposium (LATS). – 2018. – С. 1–6. – Режим доступу: https://doi.org/10.1109/LATW.2018.8347242.

21. Paiva, D. Fault injection platform for affordable verification and validation of CubeSats software / D. Paiva, J.M. Duarte, R. Lima, M. Carvalho, F. Mattiello-Francisco, H. Madeira // Proceedings of the 2021 10th Latin-American Symposium on Dependable Computing (LADC). – 2021. – С. 1–11. – Режим доступу: https://doi.org/10.1109/LADC53747.2021.9672584.

22. Axehill, J. From Brownfield to Greenfield Development - Understanding and Managing the Transition / J. Axehill, E. Herzog, J. Tingström, M. Bengtsson // 31st Annual INCOSE International Symposium. – 2021. – Режим доступу: https://www.researchgate.net/publication/352292737_From_Brownfield_to_Greenfield_Development_-_Understanding_and_Managing_the_Transition

23. NASA. NASA CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers [Електронний ресурс]. – 2018. – Режим доступу: https://smdepo.org/post/10058. – Назва з екрана (дата звернення: 25.07.2024).

24. El Allam, A.K. Highly Modular Software Framework for Reducing Software Development Time of Nanosatellites / A. K. El Allam, A.H.M. Jallad, M. Awad, M. Takruri, P.R. Marpu // IEEE Access. – 2021. – № 9. – С. 107791–107803. – Режим доступу: https://doi.org/10.1109/ACCESS.2021.3097537.

25. Jr., R.L.B. F Prime: An Open-Source Framework for Small-Scale Flight Software Systems / R.L.B. Jr., T.K. Canham, G.J. Watney, L.J. Reder, J.W. Levison // Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites. – 2018. – С. 110–119. – Режим доступу: https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/CL18-2993.pdf

26. Гладвелл, М. Генії і аутсайдери: Історія успіху / М. Гладвелл. – Київ :

Наш формат, 2019. – 320 с.

27. MIL-STD-499, Engineering Management [Електронний ресурс]. – Department of Defense, 1969. – Режим доступу: https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=36018. – Назва з екрана (дата звернення: 21.07.2024).

28. MIL-STD-1521, Technical Reviews and Audits for Systems, Equipments, and Computer Software [Електронний ресурс]. – Department of Defense, 1985. – Режим доступу: https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=36020. – Назва з екрана (дата звернення: 21.07.2024).

29. IEEE Std 15288, Systems and Software Engineering – System Life Cycle Processes [Електронний ресурс]. – Institute of Electrical and Electronics Engineers, 2015. – Режим доступу: https://standards.ieee.org/ieee/15288/5673/. – Назва з екрана (дата звернення: 21.07.2024).

30. NASA. NASA CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers [Електронний ресурс]. – 2018. – Режим доступу: https://www.nasa.gov/wp-content/uploads/2017/03/nasa_csli_cubesat_101_508.pdf?emrc=05d3e2. – Назва з екрана (дата звернення: 25.07.2024).

31. IEC-61508. IEC 61508 Ed. 2.0 en:2010 CMV, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems – Parts 1 To 7 Together With A Commented Version (See Functional Safety And IEC 61508) [Електронний ресурс]. – 2021. – Режим доступу: https://webstore.ansi.org/standards/iec/iec61508eden2010cmv. – Назва з екрана (дата звернення: 25.07.2024).

32. IEC-62304. IEC 62304 Ed. 1.1 b:2015, Medical Device Software – Software Life Cycle Processes [Електронний ресурс]. – 2020. – Режим доступу: https://webstore.ansi.org/standards/iec/iec62304ed2015. – Назва з екрана (дата звернення: 25.07.2024).

33. ISO-14971. ISO 14971:2019, Medical devices – Application of risk management to medical devices [Електронний ресурс]. – 2019. – Режим доступу: https://www.iso.org/standard/72704.html. – Назва з екрана (дата звернення: 25.07.2024).

34. ISO-26262. ISO 26262-6:2018, Road vehicles – Functional safety – Part 6: Product development at the software level [Електронний ресурс]. – 2018. – Режим доступу: https://www.iso.org/standard/68388.html. – Назва з екрана (дата звернення: 25.07.2024).

35. Edmunds, A. Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-critical Systems / A. Edmunds, M. Olszewska, M. Walden // Proceedings of the Second International Conference on Advances and Trends in Software Engineering, SOFTENG 2016. – 2016. – Режим доступу: https://eprints.soton.ac.uk/397196/1/inpEdOlWa16a.full.pdf

36. Reiff, J. Hybrid project management – a systematic literature review / J. Reiff, D. Schlegel // International Journal of Information Systems and Project Management. – 2022. – № 10. – С. 45-63. – Режим доступу: https://doi.org/10.12821/ijispm100203

References

1. CubeSat.org. Cubesat Design Specification Rev 14.1 (by the CubeSat Program) [Electronic resource]. – 2022. – Mode of access: https://www.cubesat.org/cubesatinfo. – Title from the screen (accessed: 15.07.2024).

2. Cappelletti, C. 2-CubeSat missions and applications / C. Cappelletti, D. Robson // Cubesat Handbook. – Academic Press, 2021. – С. 53–65. – Mode of access: https://doi.org/10.1016/B978-0-12-817884-3.00002-3.

3. Shkolnik, E. L. On the verge of an astronomy CubeSat revolution / E. L. Shkolnik // Nature Astronomy. – 2018. – № 2. – С. 374–378. – Mode of access: https://doi.org/10.1038/s41550-018-0438-8.

4. Crusan, J. NASA's CubeSat Launch Initiative: Enabling broad access to space / J. Crusan, C. Galica // Acta Astronautica. – 2019. – № 157. – С. 51–60. – Mode of access: https://doi.org/10.1016/j.actaastro.2018.08.048.

5. ESA. European Space Agency. Fly Your Satellite Program Intro [Electronic resource]. – 2018. – Mode of access: https://www.esa.int/Education/CubeSats_-_Fly_Your_Satellite/Fly_Your_Satellite!_programme. – Title from the screen (accessed: 22.07.2024).

6. CalPoly. P-POD User Guide / California Polytechnic State University [Electronic resource]. – 2014. – Mode of access: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/5806854d6b8f5b8eb57b83bd/1476822350599/P-POD_MkIIIRevE_UserGuide_CP-PPODUG-1.0-1_Rev1.pdf. – Title from the screen (accessed: 24.07.2024).

7. Brycetech. Smallsats by the Numbers 2023 [Електронний ресурс]. – 2023. – Mode of access: https://brycetech.com/reports/report-documents/Bryce_Smallsats_2023.pdf. – Title from the screen (accessed: 25.07.2024).

8. Kang, J. Creating Future Space Technology Workforce Utilizing CubeSat Platforms: Challenges, Good Practices, and Lessons Learned / J. Kang, J. Gregory, S. Temkin, M. Sanders, J. King // Proceedings of the AIAA Scitech 2021 Forum. – 2021. – С. 1–12. – Mode of access: https://doi.org/10.2514/6.2021-1437.

9. EXA. Cubesat Market. KRATOS 1U Platform [Electronic resource]. – 2021. – Mode of access: https://www.cubesat.market/kratos1uplatform. – Title from the screen (accessed: 26.07.2024).

10. Reznik, S. Comparison of geostationary and low-orbit «round dance» satellite communication systems / S. Reznik, D. Reut, M. Shustilova // IOP Conference Series: Materials Science and Engineering. – 2020. – № 971. – Mode of access: https://doi.org/10.1088/1757-899X/971/5/052045.

11. Swartwout, M. Cubesat Database / M. Swartwout // Sant Louis University [Electronic resource]. – 2022. – Mode of access: https://sites.google.com/a/slu.edu/swartwout/cubesat-database. – Title from the screen (accessed: 25.07.2024).

12. Kulu, E. Nanosats Database / E. Kulu // NewSpace Index [Electronic resource]. – 2022. – Mode of access: https://www.nanosats.eu/database. – Title from the screen (accessed: 25.07.2024).

13. Swartwout, M. CubeSat Database / M. Swartwout [Electronic resource]. – Mode of access: https://sites.google.com/a/slu.edu/swartwout/cubesat-database/census/. – Title from the screen (accessed: 21.07.2024).

14. ISO15408. ISO/IEC 15408-1:2022 Information security, cybersecurity and privacy protection – Evaluation criteria for IT security – Part 1: Introduction and general model [Electronic resource]. – 2022. – Mode of access: https://www.iso.org/standard/72891.html. – Title from the screen (accessed: 25.07.2024).

15. Potii, O. Advanced Security Assurance Case Based on ISO/IEC 15408 / O. Potii, O. Illiashenko, D. Komin // Proceedings of the Theory and Engineering of Complex Systems and Dependability. – Springer International Publishing, 2015. – С. 391–401. – Mode of access: https://doi.org/10.1007/978-3-319-19216-1_37.

16. Falco, G. CubeSat Security Attack Tree Analysis / G. Falco, A. Viswanathan, A. Santangelo // Proceedings of the 2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT). – 2021. – С. 68–76. – Mode of access: https://doi.org/10.1109/SMC-IT51442.2021.00016.

17. Paiva, D. Enhanced software development process for CubeSats to cope with space radiation faults / D. Paiva, R. Lima, M. Carvalho, F. Mattiello-Francisco, H. Madeira // Proceedings of the 2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC). – 2022. – С. 78–88. – Mode of access: https://doi.org/10.1109/PRDC55274.2022.00022.

18. Liubimov, O. Data Model and Methods for Ensuring the Reliability and Relevance of Data for the CubeSat Projects / O. Liubimov, I. Turkin // Proceedings of the 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). – 2022. – С. 1–7. – Mode of access: https://doi.org/10.1109/DESSERT58054.2022.10018658.

19. Goyal, T. Simulator for Functional Verification and Validation of a Nanosatellite / T. Goyal, K. Aggarwal // Proceedings of the 2019 IEEE Aerospace Conference. – 2019. – С. 1–8. – Mode of access: https://doi.org/10.1109/AERO.2019.8741886.

20. Batista, C.L.G. On the use of a failure emulator mechanism at nanosatellite subsystems integration tests / C.L.G. Batista, E. Martins, M. de Fatima Mattiello-Francisco // Proceedings of the 2018 IEEE 19th Latin-American Test Symposium (LATS). – 2018. – С. 1–6. – Mode of access: https://doi.org/10.1109/LATW.2018.8347242.

21. Paiva, D. Fault injection platform for affordable verification and validation of CubeSats software / D. Paiva, J.M. Duarte, R. Lima, M. Carvalho, F. Mattiello-Francisco, H. Madeira // Proceedings of the 2021 10th Latin-American Symposium on Dependable Computing (LADC). – 2021. – С. 1–11. – Mode of access: https://doi.org/10.1109/LADC53747.2021.9672584.

22. Axehill, J. From Brownfield to Greenfield Development - Understanding and Managing the Transition / J. Axehill, E. Herzog, J. Tingström, M. Bengtsson // 31st Annual INCOSE International Symposium. – 2021. – Mode of access: https://www.researchgate.net/publication/352292737_From_Brownfield_to_Greenfield_Development_-_Understanding_and_Managing_the_Transition

23. NASA. NASA CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers [Electronic resource]. – 2018. – Mode of access: https://smdepo.org/post/10058. – Title from the screen (accessed: 25.07.2024).

24. El Allam, A. K. Highly Modular Software Framework for Reducing Software Development Time of Nanosatellites / A.K. El Allam, A.H.M. Jallad, M. Awad, M. Takruri, P. R. Marpu // IEEE Access. – 2021. – № 9. – С. 107791–107803. – Mode of access: https://doi.org/10.1109/ACCESS.2021.3097537.

25. Jr., R.L.B. F Prime: An Open-Source Framework for Small-Scale Flight Software Systems / R.L.B. Jr., T.K. Canham, G.J. Watney, L.J. Reder, J.W. Levison // Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites. – 2018. – С. 110–119. – Mode of access: https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/CL18-2993.pdf

26. Gladvell, M. Geniyi i autsajdery: Istoriya uspixu / M. Gladvell. – Kyiv : Nash format, 2019. – 320 s.

27. MIL-STD-499, Engineering Management [Electronic resource]. – Department of Defense, 1969. – Mode of access: https://quicksearch.dla.mil/qsDoc

Details.aspx? ident_number=36018. – Title from the screen (accessed: 21.07.2024).

28. MIL-STD-1521, Technical Reviews and Audits for Systems, Equipments, and Computer Software [Electronic resource]. – Department of Defense, 1985. – Mode of access: https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=36020. – Title from the screen (accessed: 21.07.2024).

29. IEEE Std 15288, Systems and Software Engineering – System Life Cycle Processes [Electronic resource]. – Institute of Electrical and Electronics Engineers, 2015. – Mode of access: https://standards.ieee.org/ieee/15288/5673/. – Title from the screen (accessed: 21.07.2024).

30. NASA. NASA CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers [Electronic resource]. – 2018. – Mode of access: https://www.nasa.gov/wp-content/uploads/2017/03/nasa_csli_cubesat_101_508.pdf?emrc=05d3e2. – Title from the screen (accessed: 25.07.2024).

31. IEC-61508. IEC 61508 Ed. 2.0 en:2010 CMV, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems – Parts 1 To 7 Together With A Commented Version (See Functional Safety And IEC 61508) [Електронний ресурс]. – 2021. – Mode of access: https://webstore.ansi.org/standards/iec/iec61508eden2010cmv. – Title from the screen (accessed: 25.07.2024).

32. IEC-62304. IEC 62304 Ed. 1.1 b:2015, Medical Device Software – Software Life Cycle Processes [Electronic resource]. – 2020. – Mode of access: https://webstore.ansi.org/standards/iec/iec62304ed2015. – Title from the screen (accessed: 25.07.2024).

33. ISO-14971. ISO 14971:2019, Medical devices – Application of risk management to medical devices [Electronic resource]. – 2019. – Mode of access: https://www.iso.org/standard/72704.html. – Title from the screen (accessed: 25.07.2024).

34. ISO-26262. ISO 26262-6:2018, Road vehicles – Functional safety – Part 6: Product development at the software level [Electronic resource]. – 2018. – Mode of access: https://www.iso.org/standard/68388.html. – Title from the screen (accessed: 25.07.2024).

35. Edmunds, A. Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-critical Systems / A. Edmunds, M. Olszewska, M. Walden // Proceedings of the Second International Conference on Advances and Trends in Software Engineering, SOFTENG 2016. – 2016. – Mode of access: https://eprints.soton.ac.uk/397196/1/inpEdOlWa16a.full.pdf

36. Reiff, J. Hybrid project management – a systematic literature review / J. Reiff, D. Schlegel // International Journal of Information Systems and Project Management. – 2022. – № 10. – С. 45-63. – Mode of access: https://doi.org/10.12821/ijispm100203

Надійшла до редакції 16.10.2024, розглянута на редколегії 16.10.2024

Combined model of the hardware-software development process for a student CubeSat nanosatellite data processing module

The CubeSat nanosatellite concept has been a game-changer in the field of space science research and the development of cutting-edge space technologies. The primary factors for their success are low cost, relative simplicity of production, and predictable lifecycle. CubeSats are highly important for preparing future engineers:

bachelor's and master's students of aerospace universities. Therefore, the use of CubeSats is a cost-effective way to explore near space and conduct scientific work. However, there are many issues related to the effective development of software, ensuring software quality at the system level, maintenance, and reuse of software code. The advantages and disadvantages of both classical and modern agile (Agile) software development lifecycle models are considered. To enhance flexibility and reduce the complexity of CubeSat projects, a combined model for the development of a hardware-software data processing module is proposed, which combines the advantages of two models: the waterfall model for hardware development and the agile model for software development. For a comprehensive assessment of the combined model of the data processing hardware-software module development process, the SWOT analysis methodology is used, which is a popular strategic planning tool. For its implementation, the strengths and weaknesses, external opportunities, and threats of the combined model of the data processing hardware-software module development process were formulated. The proposed combined model will allow student teams with limited experience to develop hardware-software data processing modules with minimal risks, ensuring code reuse and increasing the attractiveness of student CubeSat projects.

Keywords: nanosatellite, CubeSat, software, hardware, waterfall, Agile.

Відомості про авторів:

Туркін Ігор Борисович – професор, доктор техн. наук, завідувач кафедри інженерії програмного забезпечення Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна, ел. пошта: i.turkin@khai.edu , ORCID 0000-0002-3986-4186.

Любімов Олександр Вікторович – аспірант кафедри інженерії програмного забезпечення Національного аерокосмічного університету "Харківський авіаційний інститут", Харків, Україна, ел. пошта: o.v.liubimov@khai.edu , ORCID 0000-0003-3636-6939.

Шевченко Ілона Володимірівна – доцент, канд. техн. наук, доцент кафедри інженерії програмного забезпечення Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна, ел. пошта: ilona.shevchenko@gmail.com , ORCID 0000-0003-0100-0726.

About the Autors:

Turkin Ihor– professor, doctor of technology sciences, head of the software engineering department of the National Aerospace University named after M. E. Zhukovsky "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: i.turkin@khai.edu , ORCID 0000-0002-3986-4186.

Liubimov Oleksandr – PhD student at the Department of Software Engineering, National Aerospace University "Kharkiv Aviation Institute" named after M. E. Zhukovsky "Kharkiv Aviation Institute", Kharkiv, Ukraine, email: o.v.liubimov@khai.edu , ORCID 0000-0003-3636-6939.

Shevchenko Ilona– associate professor, Candidate of Technical Sciences, Associate Professor of the Department of Software Engineering of the National Aerospace University named after M. E. Zhukovsky "Kharkiv Aviation Institute", Kharkiv, Ukraine, e-mail: ilona.shevchenko@gmail.com, ORCID 0000-0003-0100-0726.