QUANTUM COMPUTING AND TECHNOLOGIES
КВАНТОВІ ОБЧИСЛЕННЯ ТА ТЕХНОЛОГІЇ

https://doi.org/10.15407/intechsys.2025.04.045
УДК 004.4+530.145

Г.Б.МОРОЗ, канд. техн. наук, старш. наук. співроб., пров. наук. співроб.,
Інститут програмних систем НАН України,
просп. Акад. Глушкова, 40, корп. 5, м. Київ, 03187, Україна
https://orcid.org/0000-0001-8666-9503
moroz170@gmail.com

О.Г. МОРОЗ, канд. техн. наук, старш. досл., старш. наук. співроб.,
Інститут інформаційних технологій та систем НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0000-0002-0356-8780
olgahryhmoroz@gmail.com

УСПІХИ ТА ВИКЛИКИ КВАНТОВОЇ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Багато проблем, які неможливо вирішити в межах класичних обчислень, може бути вирішено з допомогою квантових обчислень. Наразі квантові комп'ютери розробляють швидкими темпами з використанням різноманітних технологій, таких як надпровідність, захоплення іонів тощо. Ключовим фактором для створення революційних квантових додатків є квантове програмне забезпечення. Через принципові відмінності між класичним і квантовим обчисленнями застосування методів та інструментів добре розвиненої класичної програмної інженерії для розроблення квантового програмного забезпечення здебільшого є безглуздим. Наразі, існує нагальна потреба у створенні нової фундаментальної дисципліни «Квантова програмна інженерія» із широким залученням до цього процесу як наукових, так і промислових кіл. Перші кроки в цьому напрямі уже зроблено. Є певні успіхи, проте залишається багато невирішених проблем та відкритих питань. У роботі надано основні принципи та теоретичне підґрунтя необхідності розроблення інженерії квантового програмного забезпечення, а також розглянуто наявні проблеми цієї галузі та проаналізовано останні досягнення.

Ключові слова: *інформаційні технології, квантове програмне забезпечення, програмна інженерія, квантові комп'ютери.*

Цитування: Мороз Г.Б., Мороз О.Г. Успіхи та виклики квантової програмної інженерії. *Information Technologies and Systems*. 4 (4). 2025. 45—73. https://doi.org/10.15407/intchsys.2025.04.045

Вступ

Протягом останніх десятиліть квантові комп'ютери стрімко розвиваються і мають потенціал виконувати конкретні завдання набагато швидше, ніж класичні комп'ютери, оскільки для оброблення інформації вони задіють принципи квантової механіки [1, 2]. Існує принаймні два типи застосувань, де квантові комп'ютери, як очікується, перевершать класичні комп'ютери. По перше, це проблеми, які потребують великої кількості паралельних обчислень, такі як оптимізація, шифрування, аналіз великих даних, штучний інтелект, машинне навчання тощо [2, 3]. Іншим застосуванням є проблеми, які потребують ефективного та точного моделювання квантових явищ у таких областях як фізика, хімія, біологія, фізіологія, медицина та матеріалознавство, що є ключовим для виробництва нових матеріалів, підтримки передової аеронавтики та біотехнологій для виготовлення нових вакцин і пошуку методів лікування різних захворювань [4, 5]. Проте, розроблення таких застосувань є дуже складним і потребує як досконалих квантових комп'ютерів, так і відповідного програмного забезпечення, без якого ці комп'ютери перетворяться на майже непотрібний мотлох.

Нині у створенні квантових комп'ютерів досягнуто значних успіхів. І хоча ці комп'ютери все ще належать до так званих «галасливих квантових комп'ютерів проміжного масштабу» [6], тобто вони не мають повної корекції помилок (отже, чутливі до впливу зовнішнього середовища) та необхідної кількісті якісних кубітів для здійснення масштабованих, надійних та відмовостійких обчислень, проте наразі є всі підстави вважати, що ті величезні зусилля та інвестиції, які направляються для розроблення стабільних квантових процесорів (КП) такими компаніями як *IBM*, *D-Wave*, *IonQ*, *Rigetti*, *Microsoft*, *Quantinuum*, *Atom Computing*, *Google* тощо, уможливлять в найближчі роки вихід квантових комп'ютерів за межі квантової ери проміжного масштабу.

Поява таких квантових комп'ютерів відкриває нові, практично необмежені, можливості у всіх сферах людської діяльності і одночасно породжує проблему ефективного використання їхньої величезної обчислювальної потужності. Розв'язання цієї проблеми неможливе без наявності відповідних методів, інструментів та процесів розроблення Квантового Програмного Забезпечення (КПЗ), яке необхідно планувати, проєктувати, кодувати, оцінювати, тестувати, гарантувати якість, тощо [7]. Отже, наразі, одним із пріоритетних напрямів науково-прикладних досліджень повинно стати створення нової наукової дисципліни «Квантова Програмна Інженерія (КПІ)», яка має вивчати концепції, принципи та рекомендації щодо розроблення, підтримки та розвитку квантових програм і бути спрямованою на підвищення їх якості та багаторазового використання шляхом систематичного застосування принципів розроблення програмного забезпечення на всіх етапах життєвого циклу, від початкового аналізу

вимог до виходу з експлуатації. Основними завданнями КПІ є як перенесення або адаптація добре відомих знань, методів, технік і практик класичної програмної інженерії в квантову область, так і розроблення принципово нових методів та технік спеціально для КПЗ з урахуванням всіх уроків, засвоєних під час розроблення класичної інженерії програмного забезпечення [8].

Може виникнути спокуса безпосередньо перенести всі здобутки класичної програмної інженерії в квантову сферу. Проте, оскільки квантове обчислення відрізняється від класичного обчислення на фундаментальному рівні, і архітектура квантових комп'ютерів не є архітектурою фон Неймана, то зробити це неможливо [3, 9]. Переважна більшість звичних прийомів класичного програмування не придатна для квантового програмування, яке вимагає принципово нових методів та інструментів. Наприклад, використання циклів типу *while,* як вони вбудовані в послідовне програмування, не має сенсу, оскільки перевірка змінної для виявлення кінця циклу призведе до згортання обчислень. Крім того, квантові обчислення та принципи, на яких вони засновані, вводять нові поняття, такі як суперпозиція, заплутаність або декогерентність, яких не існує в класичній програмній інженерії.

Останніми роками наукове співтовариство активно працює над виконанням основних завдань КПІ. Так, досягнуто консенсусу щодо гібридної моделі КПЗ, яка поєднуватиме класичні та квантові обчислення [10]. Це потребуватиме технологій, які забезпечують безперебійну взаємодію компонентів у різних обчислювальних середовищах. Використання принципів сервіс-орієнтованих обчислень (СОО) для сумісних інтерфейсів і методологій інженерії сервісів буде важливим для ефективного проєктування та керування квантовими сервісами. Ключовим кроком для створення нових методів і методологій є визначення нових абстракцій для світу квантових обчислень. В цьому напрямку можуть бути корисними напрацювання, отримані в рамках «Модельно-орієнтованої інженерії, МОІ» [11]. Також відомо, що тестування квантових систем значно відрізняється від тестування класичних систем через особливості квантових станів [12]. Потрібні нові спеціальні практики для тестування КПЗ. Інші питання квантової програмної інженерії, такі як розроблення вимог, хоча й менше привертають до себе уваги на поточному етапі через нестачу реальних програмних додатків, також відрізнятимуться від класичного аналога [13].

Метою роботи є дослідити особливості квантових обчислень, архітектури гібридних квантово-класичних обчислювальних систем та базові принципи КПІ, проаналізувати найбільш активні напрями КПІ, наявні успіхи та виклики в цій галузі на близьке майбутнє. Ще однією метою цієї роботи є привернути увагу наукової спільноти та бізнесу до актуальних проблем в різних сферах КПІ, і залучити їх до активної участі в розв'язанні цих проблем.

Деякі особливості квантових обчислень

Розглянемо деякі фундаментальні відмінності між квантовим і класичним обчисленнями, які практично унеможливлюють пряме використання майже всіх напрацювань класичної програмної інженерії в квантовій галузі і спонукають світову науково-бізнесову спільноту до інвестування розвитку КПІ, як нової дисципліни.

Біт проти кубіта

Фундаментальним поняттям класичних обчислень є *біт.* Квантові обчислення побудовані на аналогічному понятті, квантовому біті або, скорочено, кубіті. В залежності від сфери застосування як біт, так і кубіт можуть мати фізичне або математичне тлумачення.

Фізичним носієм біта (або *фізичним бітом*), може бути будь-який з довколишніх об'єктів, у якого інші стани, крім певних двох, неможливі. Нині в класичних комп'ютерах (і не тільки) домінуючим з таких об'єктів є кремнієвий транзистор, розміри якого небажано наближаються до розміру атома.

Фізичним кубітом може бути будь-яка дворівнева квантово-механічна система (наприклад, спін електрона, поляризація фотона тощо).

В квантових обчисленнях кубіти розглядаються як абстрактні математичні об'єкти з певними властивостями. Це надає можливість розвивати загальну теорію квантових обчислень незалежно від фізичної реалізації кубітів.

Між бітом та кубітом існує низка принципових відмінностей, зокрема:

1) ***За кількістю можливих станів.***

Біт — абстрактний об'єкт, який в довільний момент часу може перебувати тільки в одному із *двох можливих станів*, умовно позначених як 0 та 1.

Кубіт — абстрактний об'єкт, який у довільний момент часу може перебувати в одному із *безлічі можливих станів*, які представлені лінійною комбінацією (суперпозицією) $\psi(\alpha, \beta) = \alpha|0\rangle + \beta|1\rangle$ двох так званих *станів обчислювального базиса* $|0\rangle$ та $|1\rangle$, де *амплітуди ймовірностей* α та β — комплексні числа, для яких $|\alpha|^2 + |\beta|^2 = 1$. Стани кубіта $|0\rangle$ та $|1\rangle$, подібні станам 0 і 1 для біта, проте не еквівалентні їм, і є зручною формою запису в позначеннях Дірака двовимірних векторів, зокрема:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Іншими словами, стан кубіта є вектором у двовимірному комплексному векторному просторі (зазвичай званим простором Гільберта), ортонормований базис якого утворюють стани обчислювального базису $|0\rangle$ та $|1\rangle$.

Отже, *біт має всього два стани, а у кубіта станів безліч.*

2) ***За результатом вимірювання станів.***

Щоб визначити в якому стані (0 чи 1) знаходиться біт під час виконання програми, його просто потрібно виміряти (зчитати). Важливим є те, що після вимірювання стан біта не змінюється, тобто *вимірювання біта на його стан не впливає*. Саме завдяки цій властивості біта програмісти можуть в реальному часі спостерігати процес виконання класичної програми на моніторі комп'ютера і керувати цим процесом. На відміну від біта, *вимірювання кубіта змінює його стан* (тобто значення амплітуд α і β). В результаті вимірювання стан кубіта $\psi(\alpha, \beta)$ завжди перейде в стан $|0\rangle$ з імовірністю $|\alpha|^2$, або в стан $|1\rangle$ з імовірністю $|\beta|^2$, які за необхідності будуть зчитані як 0 або 1 класичного біта.

Отже, кубіт може існувати в континуумі станів між $|0\rangle$ і $|1\rangle$ поки його не вимірюють (спостерігають). Ця властивість кубіта не дає можливості програмістам відстежувати процес виконання квантової програми в реальному часі аж до моменту завершення обчислень.

3) ***За можливістю створення ідентичної копії (клону).***

Однією з важливих властивостей біта, *стан якого невідомий*, є можливість створення його *ідентичної* копії в будь-який момент виконання класичної програми**.** Це надає можливість переприсвоєння значення одних змінних іншим змінним, що характерно для класичного програмування. Проте, згідно *теореми про заборону клонування* [3], неможливо створити ідентичну копію квантового стану, а отже звичне для класичного програмування переприсвоєння в квантовому програмуванні позбавлене сенсу.

Наведені вище принципові відмінності між бітом та кубітом є підґрунтям для більш вагомих розходжень між класичним та квантовим обчисленнями. Так, керування квантовими обчисленнями здійснюється з допомогою унітарних операторів, інтерференції та заплутаності. Такі поняття взагалі відсутні в класичних обчисленнях. Отже, ми не перекомпілюємо класичну програму, наприклад на *C++*, для роботи на квантовому комп'ютері і навпаки**.**

Все сказане дає змогу зробити основний висновок: квантові обчислення потребують своєї інженерії програмного забезпечення.

Гібридні обчислювальні системи

Сучасна квантова програма, що виконується на квантовому комп'ютері, зазвичай використовує квантовий регістр кубітів для виконання квантових операцій і класичний регістр класичних бітів для запису вимірювань станів кубітів і умовного застосування квантових операторів [14], тобто програмне забезпечення, яке виконується на квантових комп'ютерах, природно є гібридним. Таким чином, КПІ потребує інтеграції нових квантових алгоритмів з наявними класичними інформаційними системами.

Наразі, як правило, квантові комп'ютери віддалено доступні в хмарах. Тому, класичні комп'ютери надсилають до них запити (у вигляді квантових алгоритмів), які потрібно виконати. Потім резуль-

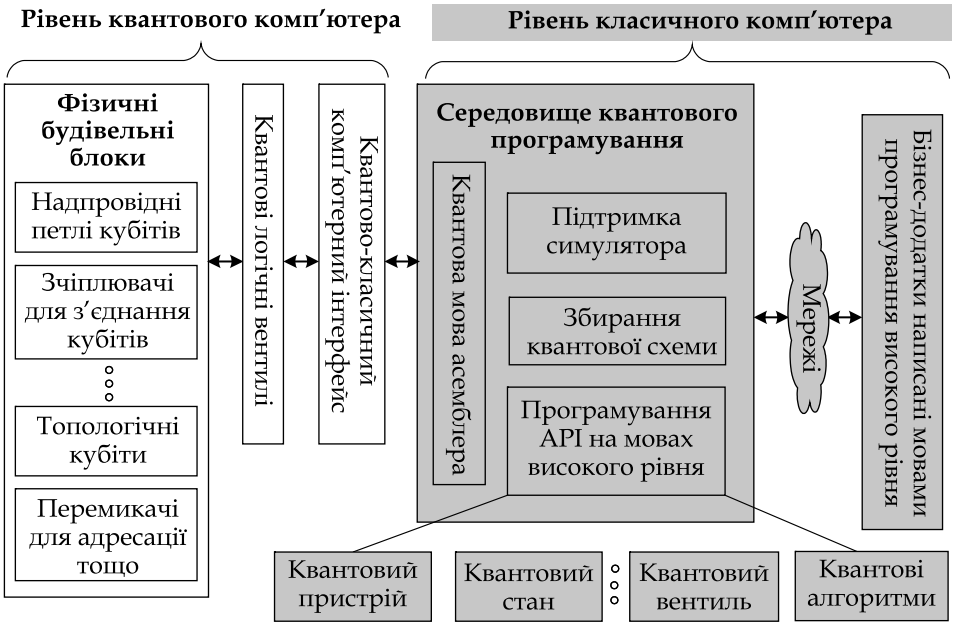

Рис. 1. Архітектура гібридної обчислювальної системи (адаптовано з [15])

тати отримує та інтерпретує класичний комп'ютер, щоб завершити вирішення конкретної проблеми.

У [15] була запропонована загальна архітектура квантової обчислювальної системи, яка складається з двох частин (рис. 1): квантового рівня та класичного рівня. Квантовий рівень містить суто квантове обладнання та схеми, і його можна розглядати як квантовий процесор. Цей рівень містить:

a) *Фізичні будівельні блоки*. Охоплюють квантове обладнання, яке зазвичай використовує надпровідні *контури* для фізичної реалізації кубітів. Крім того, вони також містять фізичну схему з'єднування кубітів та інші елементи, необхідні для адресації кубітів і операцій керування.

b) *Квантові логічні вентилі*. Фізичні схеми, що складають квантові логічні вентилі.

c) *Квантово-класичний комп'ютерний інтерфейс*. Охоплює апаратне та програмне забезпечення, яке здійснює взаємодію між класичним комп'ютером та квантовим процесором.

Класичний рівень складається з класичного обладнання та програмного забезпечення, і містить:

a) *Середовище квантового програмування*. Воно надає такі елементи, як: квантову мову асемблера, необхідну для інструкцій квантовому процесору, абстракції програмування, необхідні для написання квантових програм на мові програмування високого рівня, та підтримку симулятора, а також інтегроване середовище розроблення тощо.

b) *Бізнес-додатки*. Квантові програмні додатки, написані для задоволення потреб бізнесу.

Базові принципи квантової програмної інженерії та їхні наслідки

В результаті обговорення різних точок зору вчених і практиків, які приєдналися до 1-го Міжнародного семінару з розроблення квантового програмного забезпечення та програмування (*QANSWER*'20) було розроблено і зафіксовано у «Маніфесті Талавери» [16] низку принципів і зобов'язань щодо КПІ. Серед них найбільш вагомими є такі:

• КПІ має бути агностичною щодо технологій та мов програмування; вона повинна використовувати методи та процеси, які є зрозумілими, контрольованими та повторюваними широкими спільнотами.

• КПІ має прийняти співіснування класичних і квантових обчислень і сприяти використанню методів реінжинірингу, які дозволяють інтегрувати нові квантові алгоритми з наявними класичними інформаційними системами.

• КПІ має забезпечувати управління проєктами з розроблення КПЗ, створюючи квантове програмне забезпечення, яке задовольняє бізнес-цілям та вимогам, адекватно відповідаючи обмеженням щодо якості, часу та вартості. Необхідно буде розробити методи оцінювання ресурсно-часових зусиль під час розроблення КПЗ.

• КПІ має підтримувати еволюцію квантового програмного забезпечення протягом усього його життєвого циклу.

• КПІ має створювати квантові програми з мінімальнім рівнем дефектів і відповідати за визначення та застосування методів тестування та налагодження квантових програм таким чином, щоб більшість дефектів можна було виявити та усунути до випуску програми.

• КПІ має дбати про якість КПЗ, розробляючи нові метрики для квантових програм і квантових процесів.

• КПІ має сприяти повторному використанню КПЗ, допомагаючи командам розробників обмінюватися, індексувати та знаходити КПЗ, яке можна використовувати повторно, створюючи довідкові бібліотеки та демонстрації додатків.

• КПІ повинна на всіх етапах життєвого циклу КПЗ дбати про забезпечення безпеки та конфіденційності.

• КПІ охоплює управління та менежмент програмного забезпечення. Менеджери повинні бути обізнані з конкретними процесами, організаційними структурами, принципами, політиками, інформацією, навичками та компетенціями, а також послугами, інфраструктурою та додатками, пов'язаними з КПЗ.

Протягом першого місяця після публікації Маніфесту Талавери його підписали сотні провідних дослідників і практиків із 20 різних країн [7]. Цей факт свідчить про важливість викликів КПІ для світової спільноти програмної інженерії та квантових обчислень. Ці виклики мають важливі поточні та майбутні наслідки, як то:

• Фахівці з різних галузей знань повинні визначити, як попередній досвід досліджень програмної інженерії можна перенести в нову сферу квантової програмної інженерії [17].

• Освітні заклади повинні намагатися вирішити проблему нестачі кваліфікованих фахівців у галузі квантових обчислень та КПІ. З цією метою їм доведеться інтегрувати КПІ в навчальні програми в яких має бути чітко вказано, які компетенції та навички потрібні майбутнім інженерам КПЗ.

• Урядові та фінансові органи повинні враховувати принципи квантової програмної інженерії та розглядати квантову програмну інженерію в стратегічних дослідженнях та промислових планах.

• Уряд повинен заохочувати державні органи освіти вносити квантові обчислення та квантову програмну інженерію в навчальні програми, а постачальники квантових технологій повинні бути в курсі останніх тенденцій у квантовій програмній інженерії (як в академічних колах, так і в промисловості).

Активні напрями досліджень квантової програмної інженерії

КПІ — складна, багатогранна галузь, що стрімко розвивається протягом останніх десяти років і містить низку взаємопов'язаних між собою підгалузей. З різних причин рівень розвитку цих підгалузей суттєво відрізняється. Інтенсивність відповідних досліджень з боку науково-бізнесової спільноти є однією з таких причин. Далі представлені результати аналізу загального стану, наявних успіхів та невирішених проблем в тих підгалузях КПІ, які репрезентативно представлені у доступних літературних джерелах.

Сервіс-орієнтоване обчислення

З кінця XX століття СОО стало домінуючою обчислювальною парадигмою, яка змінила спосіб розроблення та використання програмних додатків. У цій парадигмі *вебсервіси* розглядаються як фундаментальні будівельні блоки для підтримки швидкого, гнучкого та економічного розроблення розподілених програм у гетерогенних середовищах [18]. Низка компаній, як-от *IBM*, *Amazon Braket*, *Microsoft* через *Azure Quantum* тощо пропонують в хмарах квантове обладнання та програмне забезпечення, що дає змогу дослідникам отримувати доступ до квантових обчислювальних ресурсів через інтернет без необхідності володіти квантовим обчислювальним обладнанням.

Ці платформи пропонують доступ до різноманітних квантових процесорів та симуляторів, а також до інструментів і середовищ для створення, запуску та тестування квантових алгоритмів. Проте, враховуючи початковий стан квантової апаратної технології, класичні принципи СОО не можна безпосередньо перенести на розробленя КПЗ, а отже необхідно переоцінити, як ці принципи можна адап-

тувати або перевизначити для галузі квантових обчислень. Останніми роками стрімко зростає інтенсивність досліджень у цьому напрямку і наразі можна говорити про низку вагомих результатів.

Так в роботі [19] автори запропонували технологію *TOSCA4QC*, яка представляє два стилі моделювання розгортання на основі стандарту *TOSCA* (*Topology and Orchestration Specification for Cloud Applications*) для автоматизації розгортання та оркестровки квантових додатків. У роботах [20–22] творчим колективом у складі Б. Ведер, Й. Барзен, Ф. Лейманн, М. Бейзел та ін. досліджено додаткові механізмі оркестровки в гібридних квантових програмах порівняно з класичними програмами; розроблено сервісну екосистему для виконання на основі робочих процесів різних квантових алгоритмів; запропоновано метод виявлення частин гібридного робочого процесу, який можна оптимізувати під час виконання тощо.

Загалом ці роботи надають формалізований і досить повний набір інструментів для отримання переваг роботи з робочими процесами для розгортання та виконання гібридних квантових програм.

Інша дослідницька група (Д. Валенсія, М. Мурільо, Е. Могель, Х. Гарсія-Алонсо та ін.) в роботах [23–26] проаналізувала проблеми та труднощі розроблення гібридних квантових додатків з використанням сервіс-орієнтованого підходу; запропонувала механізм адаптації шаблону *API Gateway* для поєднання з квантовим програмним забезпеченням; розширила стандарт *Open API* для розроблення квантових сервісів; розробила підходи до використання методів *DevOps* для забезпечення безперервного розгортання КПЗ; запропонувала техніку розподілу квантових обчислень між різними комп'ютерами шляхом розподілу кадрів, необхідних для даного квантового завдання тощо.

Усі ці роботи забезпечують підхід, який наближає розроблення гібридних квантових додатків до інструментів і методів, які зазвичай використовуються в СОО.

Ці приклади демонструють потенційні переваги застосування принципів сервіс-орієнтованого обчислення до квантової програмної інженерії. Однак, щоб використати весь потенціал сервіс-орієнтованого обчислення, необхідно спочатку вирішити низку проблем. А саме:

Сумісність. Оскільки різні технології та платформи для квантових обчислень продовжують з'являтися, стає зрозумілою потреба в галузевих стандартах або пропозиціях таких стандартів, зроблених дослідниками. Існують деякі приклади, зокрема *OpenQASM*, стандартизований кількома компаніями, які демонструють переваги такої стандартизації. Однак існують ще різні мови рівня асемблера, тому для вирішення цього завдання необхідні зусилля щодо створення якогось проміжного формату (на зразок *QIR*). Подібним чином створення стандартних *API* для взаємодії з квантовими процесорами

значно покращить сумісність та прокладе дослідникам шлях до створення нового покоління квантових сервіс-орієнтованих обчислювальних інструментів і рішень.

Незалежність від платформи. Можливості сучасних КП істотно різняться, так само як і середовища, в яких ці КП пропонуються через хмару, включно з доступністю гібридних середовищ виконання або інших різних покращень продуктивності, пов'язаних із програмним забезпеченням. Це породжує велику різноманітність функцій, доступних для розробників КПЗ. Однак це також створює значну залежність цього програмного забезпечення від платформи, на якій воно буде запущено.

Класичні сервіс-орієнтовані обчислення показали переваги незалежності від платформи, і тому для отримання подібних переваг потрібні додаткові зусилля в квантовій галузі.

Керування попитом та потужністю. Керування попитом та потужністю для підтримки гібридних робочих процесів вимагає координації між класичними та квантовими ресурсами, а також оптимізації передачі даних і зв'язку в різноманітних обчислювальних середовищах. Крім того, квантове обладнання має внутрішні обмеження, включно з часом когерентності, точністю вентилів і підключенням кубітів.

Управління потужністю для квантових сервісів має враховувати ці апаратні обмеження, *а також* обмеження на кількість доступних кубітів і складність квантових схем, які можна виконати. Особливою проблемою, яку слід переглянути, є аналіз потужності в архітектурі мікросервісів, які використовують сторонні сервіси [27], розширюючи їх до сценаріїв, де деякі з цих віддалених сервісів є квантовими.

Навчання робочої сили. Принципи сервіс-орієнтованих обчислень розуміє і використовує велика кількість розробників класичних програм. Однак цих розробників потрібно буде перекваліфікувати, якщо від них очікується розроблення квантового або гібридного програмного забезпечення. Цю проблему можливо вирішити принаймні двома способами. По-перше, шляхом визначення чітких стратегій навчання для спрощення переходу від класичного сервіс-орієнтованого обчислення до квантового сервіс-орієнтованого обчислення. І по-друге, шляхом розроблення методологій та інструментів КПІ, які усувають розрив між класичним і квантовим варіантами сервіс-орієнтованих обчислень.

Модельно-орієнтована інженерія

Модельно-орієнтована інженерія спрямована на вирішення проблеми складності розроблення програмних систем з допомогою мов моделювання, механізмів трансформації та генераторів для підвищення рівня абстракції розроблення програмного забезпечення [10]. В останні роки дослідження в цій галузі охоплюють низку тем, включно з визначенням мов моделювання, створенням процесів керування

моделлю та методів аналізу моделей, а також використанням моделей під час виконання.

Одна з актуальних проблем КПІ полягає в тому, що наявні мови та методології розроблення КПЗ працюють на значно нижчому рівні абстракції порівняно з типовою сферою застосування методів МОІ для класичного програмного забезпечення. Фундаментальні відмінності між класичним і квантовим програмним забезпеченням ускладнюють методи моделювання у таких додатках. Більше того, проєктування гібридного програмного забезпечення рідко моделюється на вищому рівні абстракції, ігноруючи нерелевантні технічні деталі низького рівня та зосереджуючись на архітектурних деталях.

Щоб вирішити ці проблеми, кілька дослідницьких груп почали працювати над керованою моделями квантовою програмною інженерією і вже отримали певні результати. Так, автори роботи [28] дослідили як МОІ можна використовувати для підтримки генерації квантового коду або квантової верифікації та валідації. Гемайнхардт, Віммер та ін. запропонували початкову дорожню карту дослідницьких питань, які слід розглянути, щоб запровадити переваги МОІ в квантове програмне забезпечення [29], а також проаналізували, як керовані моделлю методи оптимізації можуть бути застосовані в контексті КПЗ [30]. Це дає змогу розробникам підвищити рівень абстракції проєктування квантового алгоритму разом із наданим генератором коду.

Використовуючи схожі, керовані моделями принципи, Перес-Кастільо та ін. спрямували свої зусилля з модернізації програмного забезпечення для охоплення квантових технологій на вирішення проблем, пов'язаних з міграцією гібридних програмних систем [31]. На додаток, протягом останніх років поступово удосконалювався процес модернізації програмного забезпечення, який поєднує традиційний реінжиніринг із принципами МОІ.

В результаті було запропоновано методи зворотного проєктування для абстрагування різних мов квантового програмування [32]; було розглянуто реструктуризацію та трансформацію різних подань високого рівня [33] та запропоновано методи генерації коду з проєктів високого рівня гібридного програмного забезпечення [34].

Однією з найважливіших проблем є *абстрактне моделювання квантового/гібридного програмного забезпечення.* У цьому напрямку запропоновано профіль *UML*, який охоплює аналіз та проєктування гібридного програмного забезпечення [35]. Також обговорюються деякі ідеї для отримання нових метамоделей для моделювання квантових програм як розширення *UML* [28]. Окрім *UML*, деякі автори зосереджувалися на інших наявних стандартах. Так у [36] запропоновано підхід до моделювання на основі *BPMN* (*Business Process Model and Notation*), щоб полегшити інтеграцію квантових обчислень із класичними додатками та квантовими схемами, прагнучи спростити

завдання оркестровки та забезпечити портативність. Крім того, деякі згадані роботи використовують розширення метамоделі відкриття знань для підтримки ремонтопридатності КПЗ [32, 33].

На відміну від розширень для наявних стандартів моделювання, в деяких роботах досліджували використання предметно-специфічних мов моделювання (ПСММ) [29]. Такі ПСММ як *SimuQ* [37] і *Quingo* [38] задовольняють конкретні потреби квантових обчислень. Розроблене *Microsoft* [39] квантове проміжне подання *QIR* (*Quantum Intermediate Representation*) служить як ПСММ, побудована на проміжній мові *LLVM*, з метою забезпечення уніфікованого інтерфейсу між мовами та платформами квантового програмування.

Існують й інші подібні бібліотеки та фреймворки, які роблять код незалежним від апаратного забезпечення. Наприклад, програмне забезпечення *Xanadu Pennylane* і, певною мірою, програмне забезпечення *AWS Braket* можуть конвертувати свій код для багатьох постачальників обладнання. Врешті, було запропоновано ПСММ для квантового машинного навчання [40] та для моделювання квантових схем, отриманих із проблем здійсненності [41].

Це лише деякі приклади роботи на перетині між МОІ та КПІ. Проте для вирішення наявних проблем у цьому напрямку все ще потрібні додаткові дослідницькі зусилля, зокрема:

Розроблення високорівневих методологій проєктування гібридних програмних систем. Це має вирішальне значення для подолання розриву між парадигмами класичних та квантових обчислень і передбачає створення абстрактних структур моделювання, які інкапсулюють складність гібридних квантово-класичних взаємодій, забезпечуючи уніфіковане уявлення, яке покращує зрозумілість і полегшує прийняття проєктних рішень. Майбутні дослідження можуть зосередитися на розробці ПСММ, які пропонують інтуїтивно зрозумілі абстракції для квантово-класичної інтеграції, даючи змогу розробникам програмного забезпечення розробляти гібридні програми, не заглиблюючись у низькорівневі технічні аспекти квантових обчислень.

Обслуговування та еволюція масштабованого КПЗ. Оскільки квантове програмне забезпечення стає все більш складним і поширеним, підтримка та розвиток цих систем створюватиме значні проблеми. Майбутні дослідження можуть вивчити підходи МОІ для прогнозування впливу змін у компонентах КПЗ, забезпечення сумісності та оптимізації продуктивності в різних версіях. Для підтримки масштабованих процесів обслуговування можна розробити такі методи, як регресійне тестування на основі моделі та автоматизовані інструменти рефакторингу, адаптовані до КПЗ.

Інтелектуальна генерація та оркестровка коду. Це має вирішальне значення для підвищення продуктивності та ефективності КПІ. Автоматизуючи генерацію квантового коду з моделей високого

рівня, розробники можуть більше зосередитися на вирішенні проблем, а не на тонкощах мов квантового програмування. Майбутня робота в цій галузі може бути спрямована на розроблення складних механізмів генерації коду, наприклад, на основі оптимізації, керованої моделлю, яка перетворює моделі у виконуваний квантовий код і оптимізує цей код для конкретного квантового обладнання, враховуючи такі фактори, як підключення кубітів і точність вентилів. З іншого боку, оркестровка має на увазі керування виконанням квантових та класичних компонентів у гібридних системах, гарантуючи їх бездоганну спільну роботу для досягнення бажаних результатів. Крім того, дослідження може бути спрямоване на створення інтелектуальних інструментів оркестрування, які динамічно керують виконанням гібридних додатків, оптимізуючи розподіл та виконання ресурсів з метою підвищення продуктивності та надійності.

Тестування та налагодження

Тестування КПЗ полягає в оцінюванні правильності виконання ним своїх передбачуваних функцій, у той час як налагодження полягає у спостереженні за збоями у квантовому програмному забезпеченні, які необхідно діагностувати та усувати, особливо при зіткненні з такими проблемами, як квантовий шум, обмежені ресурси квантового обладнання та симулятора, а також обмежена спостережуваність [42].

Тестування квантових програм є значно складнішим у порівнянні з класичним програмним забезпеченням через властиві їм характеристики, включно з їх імовірнісним характером; обчислення в суперпозиціях; використання розширених функцій, таких як заплутування; труднощі в читанні або оцінці станів квантової програми в суперпозиції; відсутність точних тестових оракулів.

Кілька наукових груп зробили помітний внесок у тестування КПЗ різними способами [43–45]. Було запропоновано нові критерії покриття для оцінювання якості тестових випадків, включно з охопленням входів та виходів квантових програм [46], а також заснованими на розподілі класів еквівалентності [43]. Крім того, кілька передових методів було застосовано для тестування квантових програм, включно з комбінаторним тестуванням [47], тестуванням на основі властивостей [48], а також методами, заснованими на властивості оборотності квантових схем [49].

На додаток, з'явилися методи для тестування платформ квантових обчислень, що базуються на метаморфічному тестуванні та диференційному тестуванні [50]. Для перевірки коректності вихідних даних програми було запропоновано різні типи тверджень, такі як статистичні твердження [51], твердження на основі проєкцій [52] і динамічні твердження [53]. Більше того, оскільки шум квантового комп'ютера ускладнює можливість відрізнити фактичні помилки від збоїв програми, викликаних шумом, останнім часом було зроблено

спроби зменшити шум у квантових обчисленнях для підвищення надійності тестування. Також було розроблено структуру тестування для підтримки як модульного, так і інтеграційного тестування квантових програм [43].

Ефективним підходом до оцінювання якості набору тестів є мутаційне тестування. У різних роботах було запропоновано оператори мутації для квантових програм, наприклад *Muskit* [54] і *QmutPy* [55]. Крім того, проводиться емпіричний аналіз репозиторіїв програмного забезпечення, які містять квантові програми. Це призводить до виявлення шаблонів помилок (деякі з яких є квантовими), що впливають на квантове програмне забезпечення [56]. Ці шаблони можуть використовуватися новоствореними статичними аналізаторами для квантового коду, такими як в [57], для виявлення дефектів на ранніх стадіях процесу розроблення. До того ж, почалося дослідження автоматичного квантового відновлення програм, що демонструє ефективність великих мовних моделей, зокрема *ChatGPT*, у відновленні деяких квантових програм [58].

Налагодження передбачає аналіз програм для виявлення та виправлення помилок з допомогою причинно-наслідкового аналізу між технічними дефектами та виявленими помилками, що має вирішальне значення як для класичного, так і для квантового програмування [59]. Міранський та ін. досліджували тактику налагодження квантової програми, досліджуючи адаптацію класичних стратегій — грубої сили, зворотного відстеження та усунення причини — до квантових контекстів [44]. Метваллі та Метер запропонували структуру налагодження для квантових схем, зосередившись на схемних блоках перестановки амплітуди, фазової модуляції та перерозподілу амплітуди, вирішуючи потребу в спеціалізованих підходах для кожного типу квантової схеми, що сприяє створенню надійних квантових обчислювальних систем [60]. Сато і Кацубе надали чотири міркування щодо пошуку помилок у квантових програмах та ефективний метод пошуку помилок, що поєднує бінарний пошук на основі вартості, раннє визначення, завершення та ретроспективу, підтверджені експериментальними результатами [61].

При налагодженні квантової програми реалізація квантових тверджень часто передбачає дублювання квантових змінних і повторні вимірювання. Хуан та ін. використовували перевірку гіпотези, щоб частково реконструювати інформацію квантового реєстру з вимірювань для тверджень, відзначаючи відсутність підтримки тверджень під час виконання [51]. Щоб подолати це, Лю та ін. запропонували використовувати додаткові кубіти для захоплення інформації цільових кубітів твердження з мінімальною перервою у виконанні [52]. Крім того, Лі та ін. представили, заснований на проекції інструмент *Proq* для налагодження та тестування квантової програми, що використовує проекційні вимірювання для забезпечення

твердження без руйнування квантових змінних [53]. Більш того, Цзінь і Чжао запропонували *ScaffML* [62], мову специфікації поведінкового інтерфейсу для мови квантового програмування *Scaffold* [63], яка визначає попередні та наступні умови для модулів *Scaffold*. Ця мовна інтеграція полегшує налагодження та перевірку, дозволяючи безперебійне включення тверджень у код *Scaffold*.

Хоча всі вищезазначені роботи сприяють покращенню результатів тестування квантового програмного забезпечення, необхідно визначити відкриті проблеми, які потрібно буде вирішити в наступне десятиліття, деякі з яких наведено нижче:

Ефективні тестові оракули. Хоча існують деякі тестові оракули для оцінки проходження та не проходження тестів, більшість із них є дорогими для обчислення [46, 51, 52]. До того ж, іноді тестовий оракул повинен бути згенерований для кожного заданого входу-виходу, що збільшує вартість виконання процесу тестування [64]. Враховуючи дефіцитні та дорогі квантові обчислювальні ресурси, такі тестові оракули не масштабуються. Оракул на основі проєкцій [52] є одним із підходів, але він все ще вимагає читання квантових програм, хоча і спрямований на скорочення часу, необхідного для читання. Відсутня також ефективна підтримка (додавання динамічних тверджень є громіздким) для перевірки станів квантової програми в різних точках під час її виконання. Така перевірка, як і в класичних обчисленнях, важлива для налагодження та ремонту. Таким чином, потрібні більш ефективні тестові оракули.

Найближчі напрями досліджень охоплюють проведення емпіричних досліджень для розроблення вказівок щодо проведення експериментів і вибору відповідних статистичних методів для підтримки балансу необхідних циклів і надійності результатів. Інший напрям досліджень узгоджується з класичною оптимізацією тестування: визначення пріоритетів тестування критичних тестів. У цьому сенсі використання метрик і розроблення моделей якості можуть бути дуже корисними для вдосконалення різних аспектів процесу квантового тестування програмного забезпечення [65].

Тест масштабованості. Більшість наявних підходів до генерації тестових даних базуються на початкових класичних станах квантових програм. *QuraTest* [45] і автори роботи [43], виходять за рамки класичної генерації тестових вхідних даних. Однак поточні роботи не масштабуються до складного КПЗ та мають проблеми з ефективним покриттям високовимірного вхідного простору тесту, який визначається кількістю кубітів, заплутаних та суперпозиційних станів тощо. Таким чином є потреба у вдосконалених методах генерації тестових даних, які призведуть до ефективного виявлення помилок у нетривіальних квантових програмах (наприклад, помилкові стани суперпозиції, неправильно застосовані вентилі), враховуючи практичні обмеження, такі як обмежені квантові обчислювальні

ресурси, а також різноманітність і репрезентативність тестових даних, як ми зазвичай робимо під час класичного тестування програмного забезпечення.

Від симуляторів до справжніх квантових комп'ютерів. Наразі квантові комп'ютери є галасливими, а отже результати тесту ненадійні. Більшість сучасних робіт тестуються на ідеальних тренажерах (наприклад, [45, 47, 50, 66]). Таким чином, методи зменшення шуму доцільно інтегрувати в квантове тестування програмного забезпечення, щоб підвищити надійність результатів тестування. Крім того, через обмежений доступ до реальних квантових комп'ютерів наявні рішення для тестування в основному покладаються на симулятори. Однак у майбутньому, коли реальні квантові комп'ютери стануть більш доступними, вирішальним буде тестування квантових програм на реальних квантових комп'ютерах. Таким чином, для забезпечення економічної ефективності буде потрібна оптимізація тестів. Залежно від масштабу тестування можна розробити підходи до класичної або квантової оптимізації тестів для їх мінімізації або пріоритизації. З цією метою в [66], у разі класичного програмного забезпечення, для оптимізації тестових випадків застосовано алгоритми квантового відпалу та квантової наближеної оптимізації, що однаково застосовне для оптимізації тестів КПЗ.

Подальший розвиток тестування. Стосовно мутаційного тестування залишаються відкритими низка проблем. По-перше, необхідно зрозуміти, чи є штучні несправності справжніми. Також необхідні емпіричні дослідження для виявлення наявних відносин підпорядкування між операторами мутацій через велику кількість мутантів, які можуть бути створені.

Потрібні також подальші дослідження, щоб виявити подібності та відмінності між класичними та квантовими програмними артефактами, посібники з розроблення нових інструментів або повторного використання наявних, визначити загальні проблемні точки та розробити найкращі практики з допомогою репозиторіїв програмного забезпечення для майнінгу тощо.

Крім того, для КПЗ відображення методів тестування та діяльності на певних етапах тестування є складним [47]. Було виконано формалізацію цього відображення для модульного та інтеграційного тестування [43], але інші етапи потребують додаткової роботи.

Парадигми програмування

Програмування передбачає кодування алгоритмів (стратегій) для досягнення мети (отримання результату) з допомогою мов програмування. Типовий підхід до формулювання алгоритмів потребує розбиття їх на дедалі простіші кроки, доки не буде досягнуто рівня інструкцій, наданого мовами програмування. Процес розкладання відбувається на різних рівнях абстракції. Таким чином, спосіб

підходу до стратегій залежить від кінцевого набору операцій, доступних для обчислювальної моделі, чи то класичної, чи то квантової.

З 1996 року, коли було запропоновано метамову лямбда-*q*-числення [67], до сьогоднішнього дня було створено багато мов квантового програмування. Більшість із них дотримуються парадигми імперативного програмування. Зокрема це *QASM* [68] на основі *Assembly*, *Ket* на основі *Python* або *Q#* на основі *C#* [69]. Деякі мови є функціональними, як-от *QML* [70] і *Quipper* [71] на основі *Haskell*, або навіть декларативними, як-от *Forest* [72] на основі *Python*.

Усі перераховані мови розроблено для створення квантових схем, які обчислюватимуться на квантовому процессорі. Програмування квантових комп'ютерів на основі схем є серйозною проблемою для класичних програмістів [73]. Хоча цьому сприяє багато факторів, ми зосередимося на двох із них.

З одного боку концепція стратегій для класичних і квантових програм дуже різна. Класична програма кодує *стратегію формування результату*. Кожен крок у класичній програмі змінює стан машини. Зрештою отримуємо результат, і програма вважається правильною, оскільки вона успішно створює результат.

Навпаки, квантова програма кодує *стратегію виявлення результату*. У квантовій програмі обчислення зазвичай починаються з генерації стану суперпозиції, в якому рішення вже існують в одній або кількох конфігураціях стану. Стратегія складається з послідовності кроків для посилення амплітуди цих конфігурацій.

З іншого боку, квантові мови програмування забезпечують низький рівень абстракції. Хоча існує багато мов для квантового програмування, таких як *QASM, Quil, qibo* та *Qiskit*, усі вони обмежують рівень абстракції рівнем, який забезпечується примітивами мови (квантовими вентилями). Ці примітиви діють безпосередньо над фазою та амплітудою кубітів, які представляють квантовий стан. Отже, стратегії квантової програми необхідно розглядати з точки зору маніпуляції фазою та амплітудою. Порівнюючи з класичним програмуванням це схоже на складання стратегій у термінах обертання, зсуву, додавання або перенесення операцій над двійковими регістрами.

В останні роки було розроблено деякі ініціативи, спрямовані на те, щоб зробити квантове програмування більш доступним завдяки підвищенню рівня абстракції та наданню абстракцій, які полегшують розробленя стратегій виявлення. Одним із головних успіхів у досягненні зрозумілості, надійності та простоти класичного програмного забезпечення було впровадження основних типів даних, таких як *Integer, Float* або *Character*, а також простих операцій над ними. Було запропоновано своєрідне квантове програмування на основі *Oracle* [74]. Ідея полягає в тому, щоб розглядати квантові регістри як кодування типу даних. Такі типи доповнюються оракулами, які реалізують над ними основні операції [75]. Здійсненність цього підходу була

досліджена з урахуванням квантових регістрів, що кодують цілі числа. Потім було розроблено набір багаторазово використовуваних і компонованих оракулів, що реалізують прості операції.

З метою надання квантовим програмістам вищого рівня абстракції деякі автори використовують ідею надання квантових типів [76]. Програмістам дозволено визначати власні типи і таким чином створювати нові абстракції, на основі яких будуються їхні алгоритми. Автори робіт [77–79] також зосереджені на наданні різних засобів для кодування конкретних типів і операцій у квантових станах.

Вищевказане виявляє деякі проблеми, які доведеться вирішувати протягом наступних років, а саме:

Складність схем. Оракули продемонстрували свій потенціал для використання шаблону, що підтримує інкапсуляцію складних операцій у вигляді чорних ящиків. Цю ж ідею можна застосувати для інкапсуляції інших частин схем, що кодують цікаві стратегії, наприклад, посилення амплітуди Гровера [80]. Однак реалізація квантових схем зазвичай створює широкі та глибокі схеми, які є викликом доступним квантовим ресурсам. Таким чином, *розроблення методів і процедур для оптимізації кількості кубітів і глибини схем, що реалізують оракули,* залишається відкритим полем досліджень.

Збіркове та багаторазове квантове програмне забезпечення. Розроблення оракулів або будь-якої іншої частини квантових схем часто є дуже складним завданням, яке вимагає глибокого розуміння квантових станів. Щоб зробити більш доступними зусилля, вкладені в розроблення алгоритмів і схем, вони повинні бути якомога придатнішими для повторного використання. Інструменти, доступні для виконання цього завдання повторного використання та композиції, все ще дуже обмежені. Крім того, початкова робота, виконана для забезпечення інструментів для підтримки повторного використання та композиції [81], свідчить про те, що цей процес значно відрізняється від подібних процесів у класичній області програмного забезпечення. Таким чином, *розроблення методів документування, повторного використання та композиції КПЗ є ще одним викликом*, який необхідно вирішити протягом наступного десятиліття.

Абстракції для КПЗ. Кодування типів даних у квантових станах і розроблення операцій над цими станами є хорошою стратегією для початку підвищення рівня абстракції мов квантового програмування. Завдяки цьому можна розробити нові набори операцій для збагачення примітивів і блоків, наданих мовами. Проте типи та операції, які можна знайти в літературі, дуже близькі до тих, які зазвичай використовуються в класичних обчисленнях. Наприклад, у [82] автор визначає квантовий стан, що кодує прості числа. Однак, можливо, це не найкращі абстракції для роботи в квантових станах. Дослідницькій спільноті ще належить розробити абстракції, що підходять для завдань, які повинні виконувати квантові комп'ютери.

Якщо Річард Фейнман [83] розумів, що квантовий комп'ютер підходить для моделювання фізичного процесу, скажімо, хімічної реакції, то, можливо, хорошим типом даних для кодування квантового стану буде молекула, яка за допомогою операцій над квантовим станом може реагувати з іншими молекулами. Цей тип абстракції заслуговує окремого дослідження.

Архітектура програмного забезпечення

Архітектура програмного забезпечення передбачає створення структур, необхідних для розуміння та розроблення програмної системи. Ці структури складаються з елементів програмного забезпечення, їхніх зв'язків і властивостей. Ми можемо використовувати квантові комп'ютери, щоб змусити наше класичне програмне забезпечення вирішувати проблеми, які раніше були недоступні. Отже, квантові системи не повинні працювати незалежно, а мають співіснувати та співпрацювати з класичними системами [22, 34].

Необхідні інструменти та методології для інтеграції квантових рівнів і зацікавлених сторін, подібно до того, що є в класичних інформаційних системах. Мотивацією для інтеграції різних систем має бути не тільки той факт, що вони ґрунтуються на різних обчислювальних парадигмах. Натомість інформаційні системи повинні проєктуватися як єдине ціле і інтегруватися на основі функціональних можливостей, які вони надають, незалежно від апаратної архітектури, в якій вони знаходяться.

Архітектури програмного забезпечення відіграють вирішальну роль у досягненні безшовної інтеграції при проєктуванні систем, які відповідають вимогам бізнесу. Вивченню архітектур програмного забезпечення квантових обчислювальних систем присвячено систематичний огляд [84]. Додаткові проблеми та аспекти, пов'язані з різними архітектурами КПЗ, детально обговорюються в [85], оскільки «архітектура програмного забезпечення квантових обчислювальних систем відіграє ключову роль у визначенні їхнього кінцевого успіху та зручності використання». У [86] автори проводять емпіричне дослідження для вивчення та аналізу архітектурних рішень при створенні квантових програмних систем.

Також були запропоновані деякі конкретні шаблони проєктування [77], які в основному зосереджені на проєктуванні квантових схем. Є попередні дослідження використання на практиці деяких із цих шаблонів проєктування [87]. Запропонована попередня робота щодо розширення визначення шаблонів для гібридних програмних систем [88].

Підсумовуючи, можна сказати, що перспективи досліджень у галузі архітектури квантового програмного забезпечення можуть бути такими:

• **Дослідження факторів, що впливають на архітектурні рішення**, охоплюючи як реалізацію, так і технічний вибір**.** Це дослідження

має заглибитися в багатогранні виміри, які керують цими рішеннями, охоплюючи, крім іншого, міркування продуктивності, сумісність та взаємодію, наслідки витрат, масштабованість, гнучкість тощо.

• **Визначення шаблонів проєктування для побудованих гібридних програмних систем**. Як на низькорівневих компонентах програмного забезпечення, таких як модулі компіляції та функції, щоб забезпечити модульність і багаторазове використання, так і на високому рівні для керування оркестровкою сервісів і робочим процесом.

• **Дослідження спрямовані на отримання більш суттєвих емпіричних даних** щодо застосування шаблонів у промислових сценаріях, що охоплюють як поточні, так і майбутні шаблони проєктування та архітектурні рішення.

Дослідження еволюції архітектури гібридного програмного забезпечення з часом та надання рішення з обслуговування для управління технічним боргом та підтримання чистоти та ефективності архітектур.

Процеси розроблення програмного забезпечення

Акбар та ін. виділяють дві проблеми КПІ: «Інтеграція квантового обчислення з класичним обчисленням» та «Управління проєктами» [89], які свідчать про необхідність конкретних квантових/гібридних процесів розроблення програмного забезпечення. Ключовою проблемою є повна інтеграція цих типів систем в уніфікований життєвий цикл розроблення класично-квантового програмного забезпечення.

Таким чином, виникають деякі проблеми, наприклад, інтерпретація результатів класичним аналогом, оскільки квантові обчислення є стохастичними [90]. Крім того, для ефективного розроблення КПЗ будуть необхідні нові архітектурні парадигми та шаблони проєктування [84]. Що стосується управління проєктами, то деякі питання, як-от управління ризиками, конкретно висвітлюються при розробленні гібридних програмних систем. Наприклад, обмежена доступність реального квантового обладнання, що призводить до тестування КПЗ на симуляторах, іноді є причиною отримання різних результатів, коли квантове програмне забезпечення виконується у виробничому середовищі. Інші ризики, які сильно впливають на успіх розроблення КПЗ, це стандартизовані інструменти та фреймворки або масштабованість [91].

Існує кілька попередніх досліджень повного життєвого циклу розроблення КПЗ. Так запропоновано невелику адаптацію моделі водоспаду [92]. Оскільки такі моделі можуть успадкувати всі недоліки класичного життєвого циклу водоспаду, інші пропозиції базуються на ітераційних моделях. Ведер та ін. запропонували ітераційну модель, засновану на «походженні квантових даних», яка обслуговує різні фази життєвого циклу [93]. Перес-Кастільо та ін. запропонували адаптацію спіральної моделі інкрементального зобов'язання, яка

також є ітераційною моделлю, а також управління ризиками під час розроблення програмного забезпечення [94].

Інші пропозиції перевіряють, як сприятливі практики можна інтегрувати в розроблення КПЗ. У цьому контексті з'явилися пропозиції, які стосуються інтеграції парадигми *DevOps* у сферу розроблення КПЗ [28, 95]. У більш аналітичному ключі дослідження на основі інтерв'ю показує найважливіші виклики в розроблення гнучкого КПЗ, такі як стабільне масштабування та потреба в зрілих екосистемах інструментів і стандартних гнучких специфікаціях [96]. Значення цих викликів, ймовірно, буде першорядним у найближчі роки.

Відкритими темами досліджень на наступні роки в цій галузі можуть бути такі:

• **Управління ітераційним розробленням гібридних програмних систем** протягом усього життєвого циклу. Ітераційні моделі розроблення, які довели свою ефективність в управлінні складністю та підвищенні гнучкості в класичних проєктах програмного забезпечення, повинні бути адаптовані до розроблення гібридних програмних систем. Це містить моделі, які можуть пристосуватися до швидких змін у квантових технологіях і забезпечити основу для спільного розвитку класичних і квантових компонентів. Це, у свою чергу, створить унікальні виклики, такі як потреба в квантово-специфічних шаблонах проєктування, інтеграційних тестових структурах та інструментах розроблення, які можуть працювати з квантово-класичними програмними системами.

• **Управління ризиками в спеціалізованій сфері КПЗ** в поєднанні зі стійкими стратегіями масштабування має важливе значення для розроблення все більших і складніших гібридних програмних систем. Таким чином, будуть необхідні адаптивні стратегії управління ризиками, які можуть розвиватися разом із ландшафтом квантових обчислень. Це може включати динамічні моделі розподілу ресурсів, планування на випадок збоїв квантового обладнання та методології для оцінки надійності квантових алгоритмів.

• **Управління проєктами,** особливо зосереджене на операційних аспектах КПЗ, вимагає інтеграції *DevOps* або подібних гнучких парадигм у життєвий цикл розроблення програмного забезпечення. Отже, деякі майбутні дослідницькі напрями можуть бути зосереджені на розробленні або адаптації гнучких інструментальних ланцюжків, включно з системами контролю версій, конвеєрами безперервної інтеграції / безперервного розгортання і гнучкі інструменти управління проєктами, які є квантовими. Цей напрям досліджень міг би прояснити, як квантові обчислення у вигляді сервісу можна інтегрувати в проєкти розроблення програмного забезпечення, даючи змогу гнучким командам ефективно використовувати ресурси квантових обчислень.

Висновки

Наведено деякі фундаментальні відмінності між квантовим і класичним обчисленнями, які практично унеможливлюють пряме використання майже всіх напрацювань класичної програмної інженерії в квантовій галузі і спонукають світову науково-бізнесову спільноту до інвестування розвитку КПІ, як нової дисципліни. Також надано деяку інформацію, яка допоможе краще зрозуміти особливості та завдання КПІ.

Проведений у даній роботі аналіз останніх досягнень в галузі КПІ виявив певні факти, які можна підсумувати таким чином. По-перше, розроблення *квантового програмного забезпечення* вимагає нових методів порівняно з класичним розробленням програмного забезпечення. Ці методи вже почали формувати тіло КПІ. По-друге, квантові комп'ютери досягають стадії розвитку, яка привертає увагу промисловості. Тому емпіричні методи та методи розроблення програмного забезпечення необхідно переглянути, щоб відповідати очікуванням галузі щодо квантових обчислень, роблячи розвиток КПІ пріоритетом.

Щоб досягти цього розвитку, було визначено найбільш важливі виклики КПІ на наступні роки. Успішно вирішуючи ці проблеми, КПІ зможе підтримувати розроблення гібридних програмних систем після завершення ери галасливих квантових комп'ютерів проміжного масштабу.

ЛІТЕРАТУРА / REFERENCES

1. Chong F.T., Franklin D., Martonosi M. Programming languages and compiler design for realistic quantum hardware. *Nature*, 2017, Vol. 549, 180–187. https://doi.org/10.1038/nature23459
2. Martonosi M., Roetteler M. Next steps in quantum computing: computer science's role. *arXiv preprin*t, 2019, arXiv:1903.10541.
3. Nielsen M.A., Chuang I. *Quantum Computation and Quantum Information.* Cambridge University Press, 2010, 710 p.
4. Olson J., Cao Y., Romero J. et al .Quantum information and computation for chemistry. *arXiv preprint*, 2017, arXiv:1706.05413.
5. O'Malley P.J., Babbush R., Ian D Kivlichan S.D. et al. Scalable quantum simulation of molecular energies. *Physical Review,* 2016, X 6, 031007. https://doi.org/10.1103/PhysRevX.6.031007
6. Preskill J. Quantum computing in the NISQ era and beyond. *Quantum* 2, 79, 2018. https://doi.org/10.48550/arXiv.1801.00862
7. Piattini M., Peterssen G., Perez-Castillo R. Quantum computing: A new software engineering golden age. *ACM SIGSOFT Software Engineering Notes*, 2020, Vol. 45 (3), 12–14. https://doi.org/10.1145/3402127.3402131
8. Stepney S., Clark J., Tyrell A., et al. Journeys in non-classical computation: A Grand Challenges in Computing Research. *The International Journal of Parallel, Emergent and Distributed Systems*, 2005, Vol. 20 (1), 5–19. https://doi.org/10.1080/17445760500033291
9. Conte T.M., De Benedictis E.P., Gargini P.A., Track E. Rebooting computing: The road ahead. *Computer*, 2017, Vol. 50, 20–29. https://doi.org/10.1109/MC.2017.8

10. Weigold M., Barzen J., Leymann F., Vietz D. Patterns for hybrid quantum algorithms. *Symposium and Summer School on Service-Oriented Computing,* Springer, 2021, 34–51. https://doi.org/10.1007/978-3-030-87568-8_2
11. Schmidt D.C. et al. Model-driven engineering. *Computer-IEEE Computer Societ,* 2006, Vol. 39 (2), 25–31. https://doi.org/10.1109/MC.2006.58
12. Ali S., Yue T., 2023. Quantum Software Testing: A Brief Introduction. *IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion),* 332–333. https://doi.org/10.1109/ICSE-Companion58688.2023.00093
13. Yue T., Ali S., Arcaini P. Towards Quantum Software Requirements Engineering. *IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2023, Vol. 02, 161–164. https://doi.org/10.1109/QCE57702.2023.10201
14. Cross A.W., Bishop L.S., Smolin J.A., Gambetta J.M.. Open quantum assembly language. *arXiv preprint*, 2017, arXiv:1707.03429
15. Sodhi B. Quality attributes on quantum computing platforms. *arXiv preprint,* 2018, arXiv:1803.07407
16. Piattini M. et al. The Talavera Manifesto for Quantum Software Engineering and Programming, in *QANSWER Quantum Software Engineering & Programming*, Talavera de la Reina, CEUR-WS, 2020, 1–5.
17. Peterssen G. Quantum technology impact: the necessary workforce for developing quantum software. *QANSWER'20 – Quantum Software Engineering & Programming*, Talavera de la Reina, CEUR-WS, 2020, 6–22.
18. Papazoglou M., Traverso P., Dustdar S., Leymann F. Service-oriented computing: State of the art and research challenges. *Computer,*2007, Vol. 40 (11), 38–45. https://doi.org/10.1109/MC.2007.400
19. Wild K., Breitenbucher U., Harzenetter L., Leymann F., Vietz D., Zimmermann M. TOSCA4QC: two modeling styles for TOSCA to automate the deployment and orchestration of quantum applications. *IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC),* IEEE, 2020, 125–134. https://doi.org/10.1109/EDOC49727.2020.00024
20. Weder B., Barzen J., Leymann F., Zimmermann M. Hybrid quantum applications need two orchestrations in superposition: a software architecture perspective. *IEEE International Conference on Web Services (ICWS)*, IEEE, 2021, 1–13. https://doi.org/10.1109/ICWS53863.2021.00015
21. Beisel M., Barzen J., Garhofer S., Leymann F., Truger F., Weder B., Yussupov V. Quokka: a service ecosystem for workflow-based execution of variational quantum algorithms. *International Conference on Service-Oriented Computing,* Springer, 2022, 369–373. https://doi.org/10.1007/978-3-031-26507-5_35
22. Weder B., Barzen J., Beisel M., Leymann F. Provenance-Preserving Analysis and Rewrite of Quantum Workflows for Hybrid Quantum Algorithms. *SN Computer Science,* 2023, Vol. 4, Article 233. https://doi.org/10.1007/s42979-022-01625-9
23. Rojo J., Valencia D., Berrocal J., Moguel E., Garcia-Alonso J., Murillo J. M. Trials and tribulations of developing hybrid quantum-classical microservices systems. *arXiv preprint,* 2021, arXiv:2105.04421
24. Garcia-Alonso J., Rojo J., Valencia D., Moguel E., Berrocal J., Murillo J. M. Quantum software as a service through a quantum API gateway. *IEEE Internet Computing,* 2021, Vol. 26 (1), 34–41. https://doi.org/10.1109/MIC.2021.3132688
25. Romero-Alvarez J., Alvarado-Valiente J., Moguel E., Garcia-Alonso J., Murillo J. M. Using Open API for the Development of Hybrid Classical-Quantum Services. *International Conference on Service-Oriented Computing,* Springer, 2022, 364–368. https://doi.org/10.1007/978-3-031-26507-5_34
26. Romero-Alvarez J., Alvarado-Valiente J., Moguel E., Garcia-Alonso J., Murillo J. M. Enabling continuous deployment techniques for quantum services. *Software: Practice and Experience,* 2024, Vol. 54 (8), 1491–1515. https://doi.org/10.1002/spe.3326
27. Fresno-Aranda R., Fernandez P., Duran A., Ruiz-Cortes A. *S*emi-automated capacity analysis of limitation-aware microservices architectures. *International*

Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, 2022, 75–88. https://doi.org/10.1007/978-3-031-29315-3_7
28. Ali S., Yue T. Modeling Quantum programs: challenges, initial results, and research directions. *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software, (APEQS 2020)*, 14–21. https://doi.org/10.1145/3412451.3428499
29. Gemeinhardt F., Garmendia A., Wimmer M. Towards model-driven quantum software engineering. *IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE),* IEEE, 2021, 13–15. https://doi.org/10.1109/Q-SE52541.2021.00010
30. Gemeinhardt F., Eisenberg M., Klikovits S., Wimmer M. Model-Driven Optimization for Quantum Program Synthesis with MOMoT. *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C),* IEEE, 2023, 614–621. https://doi.org/10.1109/MODELS-C59198.2023.00100
31. Perez-Castillo R., Serrano M.A., Piattini M. *Software modernization to embrace quantum technology*. Advances in Engineering Software, 2021, Vol. 151, Article 102933. https://doi.org/10.1016/j.advengsoft.2020.102933
32. Perez-Castillo R., Jimenez-Navajas L., Piattini M. QRev: migrating quantum code towards hybrid information systems. *Software Quality Journa*l, 2022, Vol. 30 (2), 551–580. https://doi.org/10.1007/s11219-021-09574-x
33. Jimenez-Navajas L., Perez-Castillo R., Piattini M. KDM to UML Model transformation for quantum software modernization. *International Conference on the Quality of Information and Communications Technology,* Springer, 2021, 211–224. https://doi.org/10.1007/978-3-030-85347-1_16
34. Perez-Castillo R., Jimenez-Navajas L., Cantalejo I., Piattini M. Generation of Classical-Quantum Code from UML models. *IEEE International Conference on Quantum Computing and Engineering (QCE),* 2023, 165–168. https://doi.org/10.1109/QCE57702.2023.10202
35. Perez-Castillo R., Jimenez-Navajas L., Piattini M. Modelling Quantum Circuits with UML *IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, 2021, 7–12. https://doi.org/10.1109/Q-SE52541.2021.00009
36. Weder B., Breitenbucher U., Leymann F., Wild K. Integrating quantum computing into workflow modeling and execution. *IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC),* IEEE, 2020, 279–291. https://doi.org/10.1109/UCC48980.2020.00046
37. Peng Y., Young J., Liu P., Wu X. SimuQ: A Framework for Programming Quantum Hamiltonian Simulation with Analog Compilation. *Proc. ACM Program. Lang.,* 2024, Vol. 8, Issue POPL, Article 81, 2425–2455. https://doi.org/10.1145/3632923
38. Fu X., Yu J., Su X. et al. Quingo: A Programming Framework for Heterogeneous Quantum-Classical Computing with NISQ Features. *ACM Transactions on Quantum Computing,* 2021, Vol. 2 (4), Article 19, 1–37. https://doi.org/10.1145/3483528
39. Geller A. Introducing quantum intermediate representation (QIR). *Q# Blog,* Sept. 2020.
40. Moin A., Challenger C., Badii A., Gunnemann S. MOI4QAI: Towards Model-Driven Engineering for Quantum Artificial Intelligence. 2021, CoRR abs/2107.06708, arXiv:2107.06708. https://doi.org/10.48550/arXiv.2107.06708
41. Alonso D., Sanchez P., Sanchez-Rubio F. Engineering the development of quantum programs: Application to the Boolean satisfiability problem. *Advances in Engineering Software,* 2022, Vol. 173 (2), Article 103216. https://doi.org/10.1016/j.advengsoft.2022.103216
42. Shaukat A., Tao Y., Rui A. When software engineering meets quantum computing. *Communications of the ACM,* 2022, Vol. 65 (4), 84–88. https://doi.org/10.1145/3512340
43. Long P., Zhao J. Testing multi-subroutine quantum programs: From unit testing to integration testing. *ACM Transactions on Software Engineering and Methodology,* 2024, Vol. 33 (6), 1–61. https://doi.org/10.1145/3656339

44. Miranskyy A., Zhang L., Doliskani J. On Testing and Debugging Quantum Software. *ArXiv*, 2021, arXiv:2103.09172
45. Ye J., Xia S., Zhang F. et al. QuraTest: Integrating Quantum Specific Features in Quantum Program Testing. *38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2023, 1149–1161. https://doi.org/10.1109/ASE56229.2023.00196
46. Ali S., Arcaini P., Wang X., Yue T. Assessing the effectiveness of input and output coverage criteria for testing quantum programs. 1*4th IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2021, 13–23. https://doi.org/10.1109/ICST49551.2021.00014
47. Wang X., Arcaini P., Yue T., Ali S. Application of Combinatorial Testing to Quantum Programs. *IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 2021, 179–188.
48. Honarvar S., Mousavi M.R., Nagarajan R. 2020. Property-Based Testing of Quantum Programs in Q#. *IEEE/ACM 42nd International Conference on Software Engineering Workshops,* Seoul, Republic of Korea, (ICSEW'20), Association for Computing Machinery, New York, NY, USA, 430–435. https://doi.org/10.1145/3387940.3391459
49. Garcia de la Barrera A., Garcia Rodriguez de Guzman I., Polo P., Piattini M. Quantum software testing: State of the art. *J. Softw. Evol. Process,* 2023, Vol. 35 (4), Article e2419. https://doi.org/10.1002/smr.2419
50. Wang J., Zhang Q., Xu G.H., Kim M. QDiff: Differential Testing of Quantum Software Stacks. *36th IEEE/ACM International Conference on Automated Software Engineering (ASE),* 2021, 692–704. https://doi.org/10.1109/ASE51524.2021.9678792
51. Huang Y., Martonosi M. 2019. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. *46th International Symposium on Computer Architecture (Phoenix, Arizona) (ISCA '19),* Association for Computing Machinery, New York, NY, USA, 541–553. https://doi.org/10.1145/3307650.3322213
52. Liu J., Byrd G.T., Zhou H. 2020. Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation. *Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems,* 1017–1030. https://doi.org/10.1145/3373376.3378488
53. Li G., Zhou L., Yu N. et al. Projection-Based Runtime Assertions for Testing and Debugging Quantum Programs. *Proc. ACM Program. Lang.,* 2020, Vol. 4 (OOPSLA), 1–29. https://doi.org/10.1145/342821
54. Mendiluze E., Ali S., Arcaini P. et. al. *Muskit: A Mutation Analysis Tool for Quantum Software Testing. 36th IEEE/ACMInternational Conference onAutomated Software Engineering (Melbourne, Australia) (ASE '21),* IEEE Press, 2022, 1266–1270. https://doi.org/10.1109/ASE51524.2021.9678563
55. Fortunato D., Campos J., Abreu R. QMutPy: A Mutation Testing Tool for Quantum Algorithms and Applications in Qiskit. *31st ACM SIGSOFT International Symposium on Software Testing and Analysis (Virtual, South Korea) (ISSTA 2022),* 797–800. https://doi.org/10.1145/3533767.3543296
56. Paltenghi M., Pradel M. Bugs in Quantum computing platforms: an empirical study. *ACM on Programming Languages,* 2022, Vol. 6 (OOPSLA1), 1–27. https://doi.org/10.1145/3527330
57. Nayak P.K., Kher K.V., Chandra M.B. et al. Q-PAC: Automated Detection of Quantum Bug-Fix Patterns. *ArXiv*, 2023, arXiv:2311.17705
58. Guo X., Zhao J., Zhao P. On Repairing Quantum Programs Using ChatGPT. *IEEE/ACM 4th International Workshop on Quantum Software Engineering (Q-SE)*, 2024, 9–16. https://doi.org/10.1145/3643667.3648223
59. Chong F.T., Franklin D., Martonosi M. Programming languages and compiler design for realistic quantum hardware. *Nature*, 2017, Vol. 549, 180–187. https://doi.org/10.1038/nature23459
60. Metwalli S.A., Meter R.V. Testing and Debugging Quantum Circuits. *IEEE Transactions on Quantum Engineering*, 2024. 1—15. https://doi.org/10.1109/TQE.2024.3374879

61. Sato N., Katsube R. Locating Buggy Segments in Quantum Program Debugging. *arXiv preprint,* 2023, arXiv:2309.04266. https://doi.org/10.1145/3639476.3639761
62. Tiancheng Jin T., Zhao J. ScaffML: A Quantum Behavioral Interface Specification Language for Scaffold. *IEEE International Conference on Quantum Software (QSW),* IEEE, 2023, 128–137. https://doi.org/10.1109/QSW59989.2023.00024
63. Abhari A.J., Faruque A., Dousti M.J. et al. Scaffold: Quantum programming language. Technical Report. Department of Computer Science, Princeton University, 2012.
64. Garcia de la Barrera Amo A., Serrano M.A., Garcia Rodriguez de Guzman I. et. al. Automatic generation of test circuits for the verification of Quantum deterministic algorithms. *1st International Workshop on Quantum Programming for Software Engineering, QP4SE,* 2022, 1–6. https://doi.org/10.1145/3549036.3562055
65. Zhao J. Some size and structure metrics for quantum software. *IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE),* IEEE, 2021, 22–27. https://doi.org/10.1109/Q-SE52541.2021.00012
66. Wang X., Muqeet A., Yue T. et. al. Test Case Minimization with Quantum Annealers. *ArXiv*, 2023, arXiv:2308.05505 [cs.SE]. https://doi.org/10.1145/3680467
67. Maymin P. Extending the Lambda Calculus to Express Randomized and Quantumized Algorithms. *ArXiv,* 1997, arXiv:quant-ph/9612052 [quant-ph]
68. Pakin S. A quantum macro assembler. *IEEE High Performance Extreme Computing Conference (HPEC)*, 2016, 1–8. https://doi.org/10.1109/HPEC.2016.7761637
69. Da Rosa E.C.R., De Santiago R. Ket Quantum Programming. *J. Emerg. Technol. Comput. Syst.,* 2021, Vol. 18 (1), 1–25. https://doi.org/10.1145/3474224
70. Grattage J. An overview of QML with a concrete implementation in Haskell. *Electronic Notes in Theoretical Computer Science*, 2011, Vol. 270 (1), 165–174. https://doi.org/10.1016/j.entcs.2011.01.015
71. Green A.S., Lumsdaine P.L., Ross N.J. et. al. Quipper: a scalable quantum programming language. *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'13,* 2013, 333–342. https://doi.org/10.1145/2491956.2462177
72. Smith R.S., Curtis M.J., Zeng W.J. A Practical Quantum Instruction Set Architecture. *ArXiv*, 2017, arXiv:1608.03355 [quant-ph]
73. Ali S., Yue T. On the Need of Quantum-Oriented Paradigm. *2nd International Workshop on Quantum Programming for Software Engineering (QP4SE)*, 2023, Association for Computing Machinery, New York, USA, 17–20. https://doi.org/10.1145/3617570.3617868
74. Sanchez-Rivero J., Talavan D., Garcia-Alonso J. et. al. Automatic Generation of an Efficient Less-Than Oracle for Quantum Amplitude Amplification. *IEEE/ACM 4th International Workshop on Quantum Software Engineering (Q-SE),* 2023, 26–33. https://doi.org/10.1109/Q-SE59154.2023.00011
75. Leymann F. Towards a Pattern Language for Quantum Algorithms. In *Quantum Technology and Optimization Problems.* Springer International Publishing, Cham, 2019, Vol. 11413, 218–230. https://doi.org/10.1007/978-3-030-14082-3_19
76. Varga T., Aragones-Soria Y., Oriol M. Quantum types: going beyond qubits and quantum gates. *ArXiv*, 2024, arXiv:2401.15073 [quant-ph]. https://doi.org/10.1145/3643667.3648225
77. Haner T., Soeken M., Roetteler M. et. al. Quantum circuits for floating-point arithmetic. *ArXiv*, 2018, arXiv:1807.02023 [quant-ph]. https://doi.org/10.1007/978-3-319-99498-7_11
78. Wiebe N., Kliuchnikov V. Floating point representations in quantum circuit synthesis. *New Journal of Physics*, 2013, Vol. 15, Article 093041. https://doi.org/10.1088/1367-2630/15/9/093041
79. Mayoh B., Tyugu E., Penjam J. *Constraint Programming.* Springer Berlin Heidelberg, 2013.

80. Grover L.K. Quantum Computers Can Search Rapidly by Using Almost Any Transformation. *Physical Review Letters,* 1998, Vol. 80 (19), 4329–4332. https://doi.org/10.1103/PhysRevLett.80.4329
81. Sanchez-Rivero J., Talavan D., Garcia-Alonso J. et. al. Some Initial Guidelines for Building Reusable Quantum Oracles. *Services and Quantum Software – 21st International Conference on Service-Oriented Computing,* 2023, arXiv:2303.14959v1. https://doi.org/10.1007/978-981-97-0989-2-16
82. Garcia-Martin D., Ribas E., Carrazza S. et. al. The Prime state and its quantum relatives. *Quantum*, 2020, Vol. 4, 371. https://doi.org/10.22331/q-2020-12-11-371
83. Feynman R.P. Simulating physics with computers. In *Feynman and computation*. CRC Press, 2018, 133–153. https://doi.org/10.1201/9780429500459-11
84. Khan A.A., Ahmad A., Waseem M. et. al. Software architecture for quantum computing systems. A systematic review. *Journal of Systems and Software,* 2023, Vol. 201, Article 111682. https://doi.org/10.1016/j.jss.2023.111682
85. Yue T., Mauerer W., Ali S., Taibi D. 2023. Challenges and Opportunities. In *Quantum Software Architecture*. Springer Nature Switzerland, Cham, 1–23. https://doi.org/10.1007/978-3-031-36847-9_1
86. Aktar S., Liang P., Waseem M. et. al. 2023. Architecture Decisions in Quantum Software Systems: An Empirical Study on Stack Exchange and GitHub. arXiv: 2312.05421 [cs.SE]
87. Perez-Castillo R., Fernandez-Osuna M., Cruz-Lemus J.A. et. al. A Preliminary Study of the Usage of Design Patterns in Quantum Software. *IEEE/ACM 4nd International Workshop on Quantum Software Engineering (Q-SE),* 2024, In Press. https://doi.org/10.1145/3643667.3648220
88. Weigold M., Barzen J., Leymann F. et. al. Patterns for hybrid quantum algorithms. *Symposium and Summer School on Service-Oriented Computing,* Springer, 2021, 34–51. https://doi.org/10.1007/978-3-030-87568-8_2
89. Akbar M., Khan A., Rafi S. A systematic decision-making framework for tackling quantum software engineering challenges. *Automated Software Engineering*, 2023,Vol. 30, Article 22. https://doi.org/10.1007/s10515-023-00389-7
90. Haghparast M., Mikkonen T., Nurminen J.K. et. al. Quantum Software Engineering Challenges from Developers' Perspective: Mapping Research Challenges to the Proposed Workflow Model. *IEEE International Conference on Quantum Computing and Engineering (QCE),* 2023, 173–176. https://doi.org/10.1109/QCE57702.2023.10204
91. Akbar M.A., Khan A.A., Shameem M. et. al. Genetic model-based success probability prediction of quantum software development projects. *Information and Software Technology,* 2024, Vol. 165, Article 107352.
92. Dey N., Ghosh M., Kundu S.S., Chakrabart A. QDLC – The Quantum Development Life Cycle, 2020, *arXiv*:2010.08053 [cs.ET]
93. Weder W, Barzen J., Leymann F. et. al. Quantum Software Development Lifecycle. *Quantum Software Engineering,* Springer International Publishing, Cham, 2022, 61–83. https://doi.org/10.1007/978-3-031-05324-5_4
94. Perez-Castillo R., Serrano M.A., Cruz-Lemus J.A. et. al.. Guidelines to use the incremental commitment spiral model for developing quantum-classical systems. *Quantum Information and Computation,* 2024, Vol. 24 (1&2), 71–88. https://doi.org/10.26421/QIC24.1-2-4
95. Stirbu V., Haghparast M., Waseem M. et. al. Full- Stack Quantum Software in Practice: Ecosystem, Stakeholders and Challenges. *ArXiv:2307.16345v1,* 2023, 177–180. https://doi.org/10.1109/QCE57702.2023.10205
96. Khan A.A., Akbar M.A., Ahmad A. el al. Agile Practices for Quantum Software Development: Practitioners Perspectives. *arXiv:2210.09825v,* 2022. https://doi.org/10.1109/QSW59989.2023.00012

Отримано / Received 03.02.2025

H.B. MOROZ, PhD (Engineering), Senior Researcher, Leading Researcher,
Institute of Software Systems of the NAS of Ukraine,
40, Hlushkova Akad. ave., Building 5, Kyiv, 03187, Ukraine
https://orcid.org/0000-0001-8666-9503
moroz170@gmail.com

O.H. MOROZ, PhD (Engineering), Senior Researcher,
Institute of Information Technologies and Systems of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0000-0002-0356-8780
olgahryhmoroz@gmail.com

ADVANCES AND CHALLENGES IN QUANTUM SOFTWARE ENGINEERING

Introduction. Quantum computers have been developing rapidly in recent decades, as they use the principles of quantum mechanics to process information and have the potential to perform certain tasks much faster than classical computers. There are at least two groups of problems where quantum computers can outperform classical computers, namely: 1) problems requiring a large amount of parallel computing, such as optimization, encryption, big data analysis, artificial intelligence, machine learning, etc.; 2) problems requiring efficient and accurate modeling of quantum phenomena in fields such as physics, chemistry, biology, physiology, medicine and materials science, etc., which is key to creating new materials, supporting advanced aeronautics and biotechnology, producing new vaccines, and finding treatments for various diseases, etc.

Although for now quantum computers still belong to the so-called "noisy mid-scale quantum computers", there is every reason to believe that the huge efforts and investments directed by the scientific community and business to develop stable quantum processors will allow these computers to go beyond the quantum era of mid-scale computing in the coming years. This opens up new, practically unlimited possibilities for quantum computers in all areas of human activity and at the same time significantly complicates the problem of effective use of their enormous computing power. Solving this problem is impossible without the availability of appropriate specification and verification methods, tools and processes for developing quantum software, which must be planned, designed, coded, evaluated, tested, convenient to use, etc. Since quantum computing differs from classical computing at a fundamental level, and the architecture of quantum computers is not a von Neumann architecture, it is practically impossible to transfer all the achievements of classical software engineering to the quantum sphere. Therefore, in recent years, all developed countries have significantly increased financial and resource investments in the creation of a new scientific direction "Quantum Software Engineering", which should study concepts, principles, processes and develop recommendations for the development, support and advancement of quantum programs and be aimed at improving their quality and reusability through the systematic application of quantum software development principles at all stages of the life cycle, from the initial analysis of requirements to decommissioning.

The purpose of the paper is to study the features of quantum computing, the architecture of hybrid quantum-classical computing systems and the basic principles of quantum software engineering, analyze its most active areas, existing successes and challenges in this field for the near future.

Methods. The analysis of recent achievements in the field of quantum software engineering is carried out and it is shown that the development of quantum software requires, for the most part, fundamentally new methods and approaches compared to classical software development.

Results and conclusions. Due to the fundamental differences between classical and quantum computing, the application of methods and tools of well-developed classical software engineering to the development of quantum software is mostly pointless. Currently, there is an urgent need to create a new fundamental discipline "Quantum Software Engineering" with the broad involvement of both scientific and industrial circles in this process. The first steps in this direction have already been taken. There are some successes, but many unresolved problems and open questions remain. This paper presents the basic principles and theoretical background for the need to develop "Quantum Software Engineering", as well as discusses existing problems and analyzes achievements in this field, which is used to identify necessary breakthroughs and future research directions.

Keywords: *information technology, quantum software, software engineering, quantum computing.*