УДК 004.75 DOI: 10.20998/2413-3000.2024.8.7

А. А. ПАШНЄВ, М. В. СЛЄПУШКОВ, Д. О. ГУРТ, І. В. ЛЮТЕНКО

ДОСЛІДЖЕННЯ УПРАВЛІННЯ РОЗГОРТАННЯМ ПРОГРАМНОЇ СИСТЕМИ ІЗ ВИКОРИСТАННЯМ РЕСУРСІВ ХМАРНИХ ПРОВАЙДЕРІВ

Проведено аналіз основних етапів процесу розгортання програмної системи із використанням ресурсів хмарних провайдерів. Розроблені моделі можливих варіантів управління розгортанням програмної системи із використанням нотації BPMN. На основі проведеного моделювання визначені переваги та недоліки неавтоматизованого, автоматизованого та автоматичного управління процесом розгортання. Розроблено та декомпозовано модель автоматичного розгортання програмної системи із використанням ресурсів дата центрів хмарних провайдерів в нотації IDEF0, яка дозволила дослідити функціональну взаємодію окремих етапів процесу розгортання програмної системи. На основі розробленої контекстної та декомпозиційної діаграм IDEF0, в рамках моделювання процесу розгортання програмної системи, проведено аналіз функціональної взаємодії етапів Build та Deploy, який показав, що саме ці етапи мають найбільше функціональне навантаження та потребують більш детального дослідження, з метою пошуку шляхів їх оптимізації. З цією метою, було розроблено та проведено аналіз декомпозиційних діаграм IDEF0 та DFD, які моделюють функціональну взаємодію та потоки даних між складовими підпроцесів Build та Deploy. Аналіз функціональної взаємодії та потоків даних, що породжуються та передаються між окремими складовими підпроцесів Build та Deploy дозволив виявити важливий аспект, який полягає в тому, що при кожному виконанні компіляції вихідного коду та створенні койнтейнера потрібно завантажувати дані із зовнішніх бібліотек. Це в свою чергу фактично призводить до суттєвого збільшення зовнішнього трафіку, що впливає на швидкість розгортання програмних систем із використанням ресурсів дата центрів хмарних провайдерів. В якості можливого шляху оптимізації підпроцесів Build та Deploy запропоновано створення кешу для збереження завантажених даних із зовнішніх бібліотек з метою їх повторного використання, а також реалізацію контролю над здійсненням запитів до зовнішніх бібліотек та процесом кешування даних, що дозволить мінімізувати витрати часу на передачу даних із зовнішніх бібліотек під час автоматичного розгортання.

Ключові слова: управління розгортанням, програмна система, дата центр, хмарні провайдери, Build, Deploy.

A. PASHNIEV, M. SLIEPUSHKOV, D. HURT, I. LIUTENKO

RESEARCH OF THE MANAGEMENT OF SOFTWARE SYSTEM DEPLOYMENT USING THE RESOURCES OF CLOUD PROVIDERS

The main stages of the process of deploying a software system using the resources of cloud providers are analyzed. The models of possible options for managing the deployment of a software system using the BPMN notation are developed. Based on the modeling, the advantages and disadvantages of non-automated, automated, and automatic management of the deployment process are identified. A model of automatic deployment of a software system using the resources of cloud providers' data centers in IDEF0 notation has been developed and decomposed, which allowed to study the functional interaction of individual stages of the software system deployment process. Based on the developed contextual and decomposition IDEF0 diagrams, within the framework of modeling the software system deployment process, an analysis of the functional interaction of the Build and Deploy stages was carried out, which showed that these stages have the greatest functional load and require more detailed research in order to find ways to optimize them. With this purpose, were developed and analyzed IDEF0 and DFD decomposition diagrams that model the functional interaction and data flows between the components of the Build and Deploy subprocesses. Analysis of the functional interaction and data flows generated and transferred between the individual components of the Build and Deploy subprocesses revealed an important aspect, which is that each time the source code is compiled, and the container is created, data must be loaded from external libraries. This, in turn, leads to a significant increase in external traffic, which affects the speed of deployment of software systems using the resources of cloud providers' data centers. As a possible way to optimize the Build and Deploy subprocesses, was proposed to create a cache to save the downloaded data from external libraries for reuse, as well as implement control over queries to external libraries and the data caching process, which will minimize the time spent on transferring data from external libraries during automatic deployment.

Keywords: deployment management, software system, data center, cloud providers, Build, Deploy.

Вступ. Дослідження процесу управління розгортанням програмних систем із використанням ресурсів дата центрів хмарних провайдерів є важливим з кількох причин. По перше, хмарні ресурси продовжують відігравати ключову роль на всіх етапах розгортання програмних систем, яке може бути реалізоване різними способами. По друге, кожен із способів реалізації має певні особливості і забезпечує різні ступені автоматизації управління розгортанням програмних систем. По трете, ефективність процесу розгортання залежить від багатьох факторів, що робить надзвичайно важливим дослідження найкращих практик і технологій використання хмарних ресурсів у процесі розгортання. По четверте, масштабованість і гнучкість, які пропонують дата центри хмарних провайдерів, відкривають додаткові можливості для оптимізації процесу розгортання.

Дослідження в цій галузі сприятиме удосконаленню автоматизованих процесів управління розгортанням, що призведе до підвищення ефективності, скорочення часу розгортання та покращення використання ресурсів дата центрів хмарних провайдерів.

Тому, дослідження процесу управління розгортанням програмних систем із використанням ресурсів дата центрів хмарних провайдерів є актуальною науковою задачею.

Аналіз останніх досліджень і публікацій.

Аналіз останніх публікацій [1-5], які присвячені дослідженню процесу управління розгортанням програмних систем із використанням ресурсів дата центрів хмарних провайдерів, показав, що в них здебільшого розглядаються загальні принципи управління розгортанням програмної системи в хмарному середовищі та описується значення

автоматизації цього процесу, без проведення детального порівняльного аналізу способів реалізації, які забезпечують різні ступені автоматизації розгортання програмних систем.

В публікаціях [6-10] робиться акцент на розкритті деталей практичної реалізації процесу управління розгортанням програмних систем із використанням конкретних технологій, без детального аналізу переваг та недоліків реалізованих процесів, що не дає розуміння можливих шляхів їх удосконалення.

Метою статті є дослідження процесу управління розгортанням програмної системи із використанням ресурсів дата центрів хмарних провайдерів та визначення можливих шляхів його оптимізації.

Виклад основного матеріалу. Процес розгортання програмної системи є ключовим етапом в життєвому циклі розробки програмного забезпечення. Загальний вигляд цього процесу часто представляють у вигляді схеми, наведеної на рис. 1, яка ілюструє послідовність основних етапів розгортання [11].

Рис. 1. Загальний вигляд розгортання програмної системи

Розглянемо більш детально кожний із зображених етапів.

Етап Source представлено як зовнішній процес, який відповідає за структурування та обробку вихідного коду. У сучасних проєктах це реалізується за допомогою розподіленої системи управління версіями Git, або подібних систем [12]. В подальшому вихідний код обробляється в межах окремих етапів процесу розгортання програмної системи, сутність яких буде наведена нижче.

Етап Build містить кілька ключових елементів, кожен з яких відіграє важливу роль в успішному розгортанні програмної системи. Розглянемо ці елементи більш детально:

1. Попередня обробка. На цьому етапі відбувається завантаження необхідних бібліотек, потрібних для компіляції та збірки. Етап попередньої обробки гарантує, що всі необхідні бібліотеки та залежності будуть доступні для наступного етапу процесу збірки.

2. Компіляція. Передбачає перетворення вихідного коду в об'єктний або проміжний код за допомогою компілятора. Вихідний код програми перекладається у представлення нижчого рівня, яке може бути виконане процесором комп'ютера. За потреби додаються статичні файли та посилання на сторонні бібліотеки [2].

Розглядаючи етап Test слід зазначити, що кількість складових елементів цього етапу, при розгортанні програмної системи, може відрізнятися залежно від розміру та складності проєкту. Так, він може бути реалізований виключно за допомогою модульного тестування або мати більш комплексну реалізацію, коли додаток розгортається в тестовому середовищі, подібному до виробничого. В цьому випадку може проводитись широкий спектр тестів, таких як: інтеграційне тестування, функціональне тестування, тестування продуктивності, тестування безпеки, тестування прийнятності для користувача (UAT) та інші види тестувань [3]. Набір тестів може бути розширений або скорочений залежно від розміру та складності проєкту.

Етап Deploy передбачає перенесення розробленого програмного забезпечення із середовища розробки в виробниче середовище системи для його подальшого активного використання. Розглянемо більш детально ключові елементи цього етапу:

1. Створення пакета. Цей етап передбачає створення виробничого середовища для розгортання програмного забезпечення та забезпечення його доступності або створення цільового середовища, що включає сервери, бази даних і сервіси або, за потреби, створення контейнера [13].

2. Розгортання в тестовому середовищі. На цьому етапі відбувається копіювання файлів, інтеграція з даними та налаштування параметрів для тестування. Це дозволяє визначити працездатність програми в контрольованому середовищі.

3. Розгортання у виробничому середовищі. Аналогічно розгортанню в тестовому середовищі, цей етап забезпечує перенесення програмного забезпечення у виробничий режим роботи. Це включає копіювання файлів, інтеграцію з даними та проведення необхідних налаштувань для ефективної роботи виробничого середовища.

4. Масштабування програмної системи в дата центрах хмарних провайдерів включає в себе збільшення або зменшення кількості ресурсів в залежності від навантаження на програмну систему. Масштабування дозволяє ефективно використовувати ресурси хмарних дата-центрів та забезпечує стабільність роботи системи, навіть при зміні обсягу роботи [14].

В залежності від ступеня автоматизації, який визначає роль людського фактору та використання автоматизованих інструментів, управління розгортанням програмної системи може бути: неавтоматизованим, автоматичним та автоматизованим.

Процес неавтоматизованого управління розгортанням (Manual Deployment Management) передбачає, що створення середовища для розгортання, завантаження зі сховища та компіляція коду, створення контейнера для завантаження реалізується вручну. Розгортання в тестовому та виробничому середовищі також відбувається вручну. Кількість тестів ручного тестування зазвичай вибирається невеликою у порівнянні з автоматичним тестуванням [15].

На рис. 2 представлена модель процесу неавтоматизованого управління розгортанням програмної системи з використанням нотації BPMN.

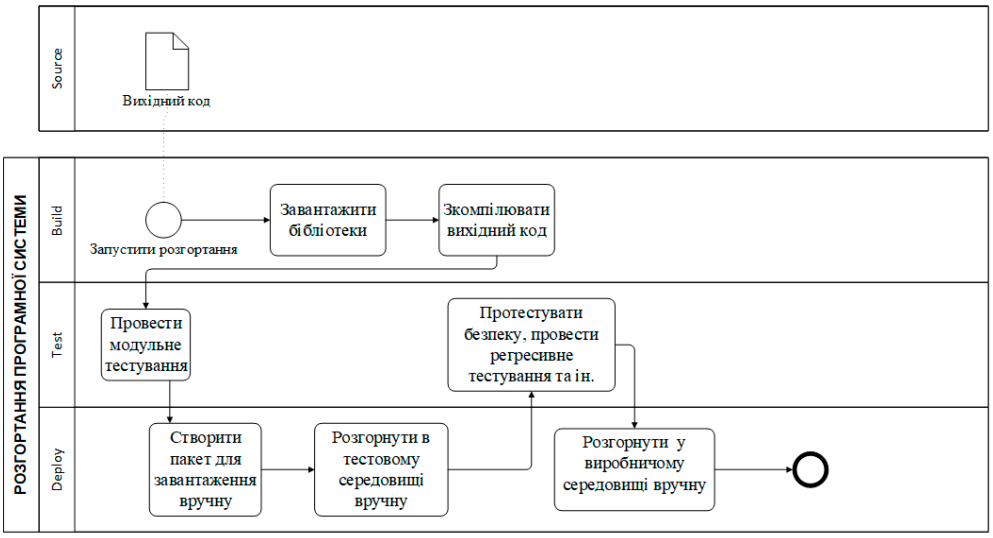

Рис. 2. Модель неавтоматизованого управління розгортанням програмної системи

Проведений аналіз послідовності виконання та реалізації окремих етапів неавтоматизованого розгортання програмної системи дозволив виділити переваги та недоліки такого способу управління розгортанням, які наведені нижче.

Переваги неавтоматизованого управління розгортанням програмної системи:

- неавтоматизоване управління розгортанням програмної системи легше впроваджувати для невеликих проєктів або тестових середовищ;
- оператор або адміністратор системи може вручну контролювати кожний етап розгортання.

Недоліки неавтоматизованого управління розгортання програмної системи:

- зазвичай розгортання програмної системи вимагає більше часу, особливо для великих і складних проєктів;
- неавтоматизоване управління розгортанням може призвести до непередбачуваних помилок через людський фактор;
- складно досягти єдності конфігурації на різних серверах та в різних середовищах, уникаючи ситуацій, коли одна частина системи працює відмінно, а інша - ні.

На противагу неавтоматизованому управлінню розгортанням, процес автоматичного управління розгортанням програмної системи (Automated Deployment Management) передбачає, що всі дії щодо його реалізації відбуваються автоматично. В цьому випадку тригером автоматичного процесу є надсилання коду в репозиторій, який і запускає весь ланцюг подальших подій управління розгортанням. Кінцевим результатом такого процесу буде працююча програмна система [3].

На рис. 3 зображена модель процесу автоматичного управління розгортанням програмної системи з використанням нотації BPMN.

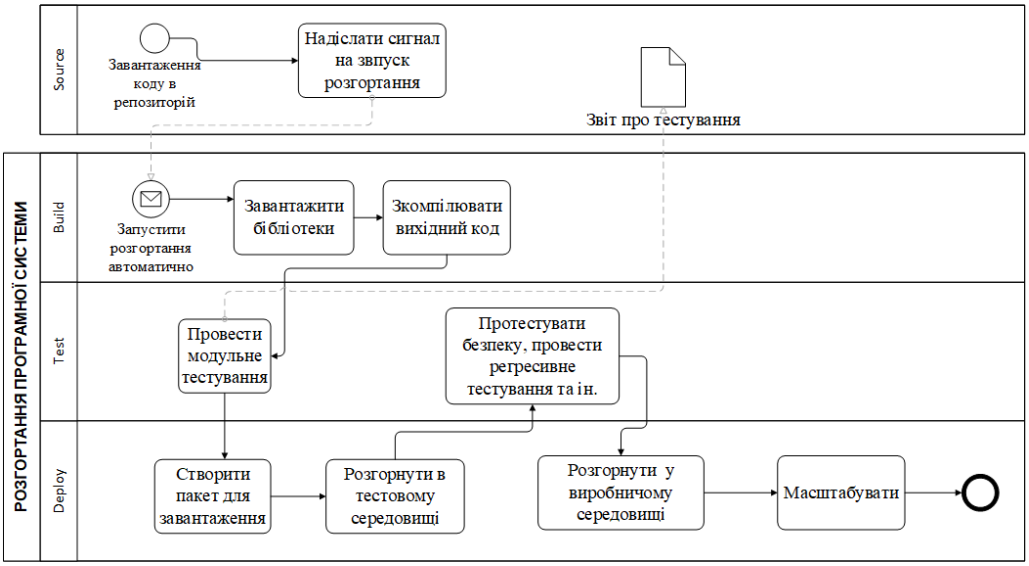

Рис. 3. Модель автоматичного управління розгортанням програмної системи

Аналіз окремих етапів автоматичного управління розгортанням програмної системи, дозволив виділити переваги та недоліки такого процесу.

До переваг процесу автоматичного управління розгортанням програмної системи можна віднести:

-швидке впровадження завдяки автоматизації всіх етапів розгортання;

- зменшення ризику помилок завдяки повторюваним автоматичним процесам;

- зручне використання для масштабування та автоматичного керування великими інфраструктурами;

- легке забезпечення єдності конфігурації в усіх середовищах.

В якості недоліків автоматичного управління розгортанням програмної системи можна зазначити:

- необхідність часу та експертизи для

- встановлення і налаштування автоматичних інструментів;

- вищі витрати на впровадження та підтримку автоматичного середовища у порівнянні з неавтоматизованим розгортанням.

Автоматизоване управління розгортанням (Partially Automated Deployment Management) передбачає автоматизацію основних етапів управління та ручне виконання деяких робіт із розгортання програмної системи. Наприклад, в неавтоматизованому режимі запускаються за потреби тести, які є занадто коштовними або тести, які надто складно автоматизувати (тести, що перевіряють зручність користувацького інтерфейсу програми). Також існують проєкти, де збірки програмної системи в тестовому середовищі розгортаються автоматично, а запуск створення релізу виконується в ручному режимі [8].

Модель автоматизованого управління розгортанням програмної системи наведена на рис. 4.

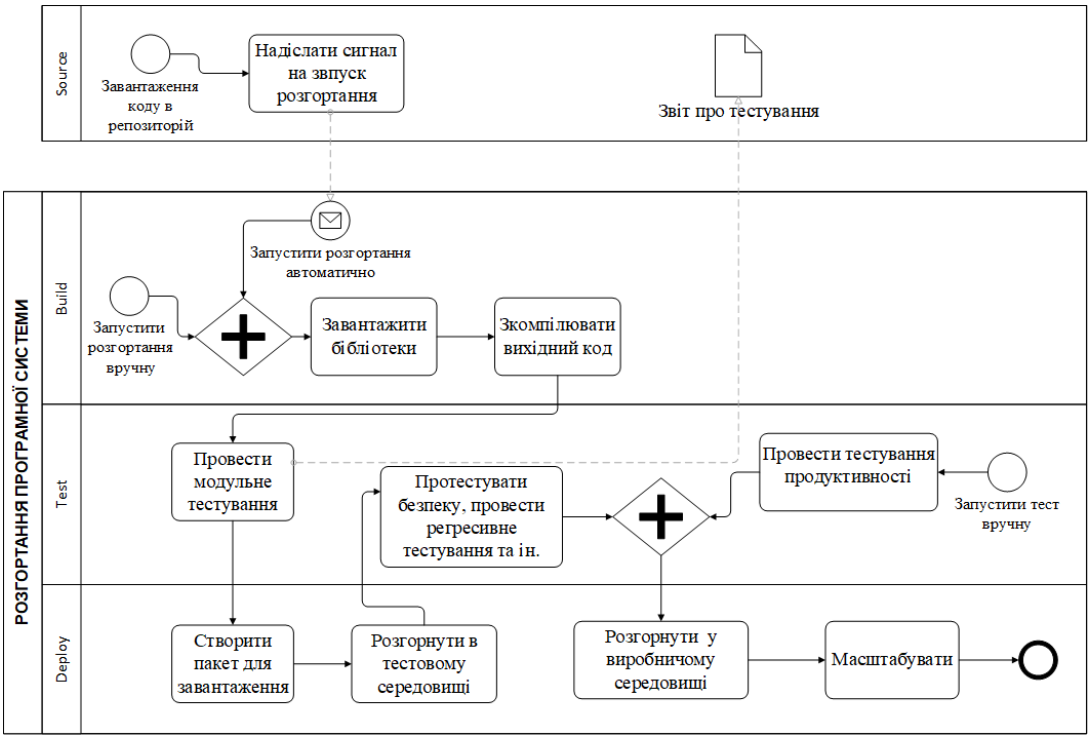

Рис. 4. Модель автоматизованого управління розгортанням програмної системи

В результаті аналізу послідовності виконання та реалізації окремих етапів автоматизованого управління розгортанням програмної системи були виділені наступні переваги та недоліки такого способу управління розгортанням.

Переваги автоматизованого управління розгортанням програмної системи:

- забезпечується поєднання автоматичних та ручних етапів розгортання програмної системи для досягнення кращого результату;

- можливе ручне втручання в процес управління розгортанням на етапах, які вимагають специфічної реакції.

Недоліки автоматизованого управління розгортанням програмної системи:

- вимагається підвищена увага для забезпечення відповідності стандартам та процедурам процесу управління розгортанням;

- успішність всього процесу управління розгортанням залежить від ефективності взаємодії автоматичних та неавтоматизованих етапів виконання.

Як ми бачимо, кожен з цих підходів має свої переваги та недоліки, і вибір між ними залежить від конкретних потреб проєкту, його розміру та вимог до швидкості розгортання і частоти випуску релізів програмної системи. Оскільки при побудові сучасних програмних систем із використанням ресурсів дата центрів хмарних провайдерів існує запит на збільшення кількості автоматичних розгортань, є сенс більш детально дослідити саме цей спосіб управління

розгортанням програмної системи, з метою визначення шляхів його подальшої оптимізації.

З метою дослідження функціональної взаємодії окремих етапів процесу управління розгортанням програмної системи із використанням ресурсів дата центрів хмарних провайдерів, була розроблена відповідна модель у вигляді контекстної та декомпозиційної діаграм в нотації IDEF0, які представлені на рис. 5 - 6.

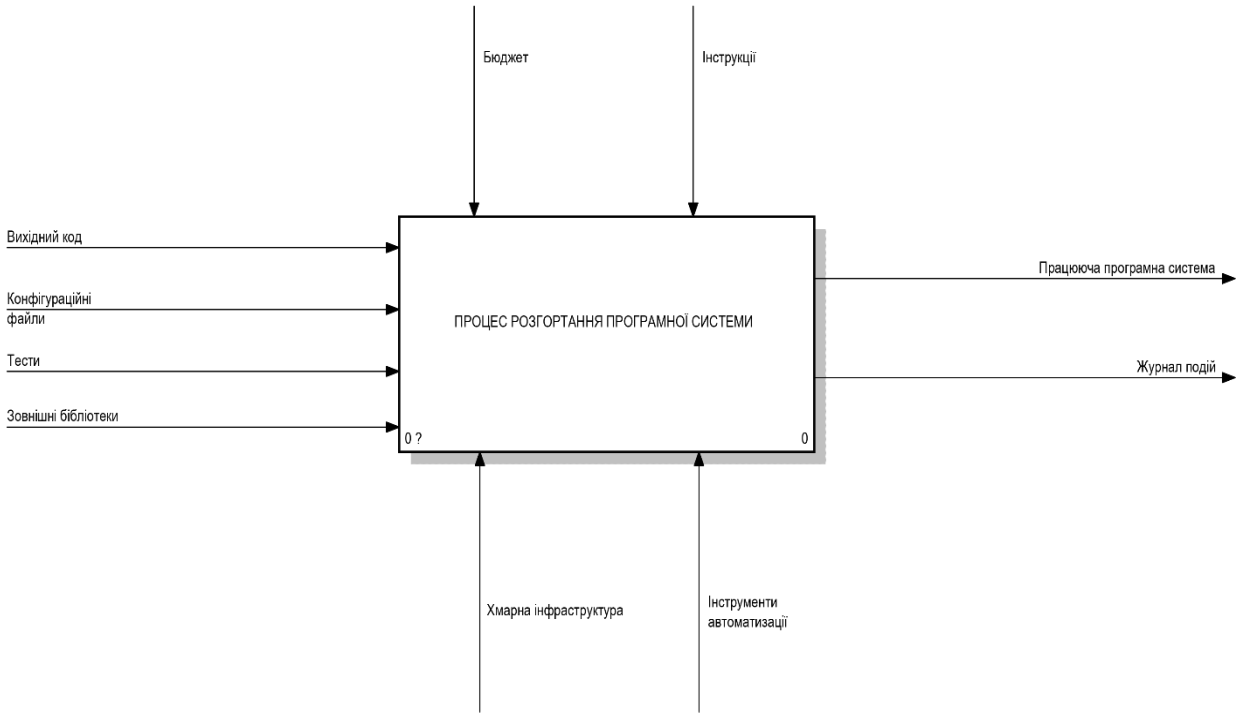

Рис. 5. Контекстна діаграма автоматичного розгортання програмної системи із використанням ресурсів дата центрів хмарних провайдерів в нотації IDEF0

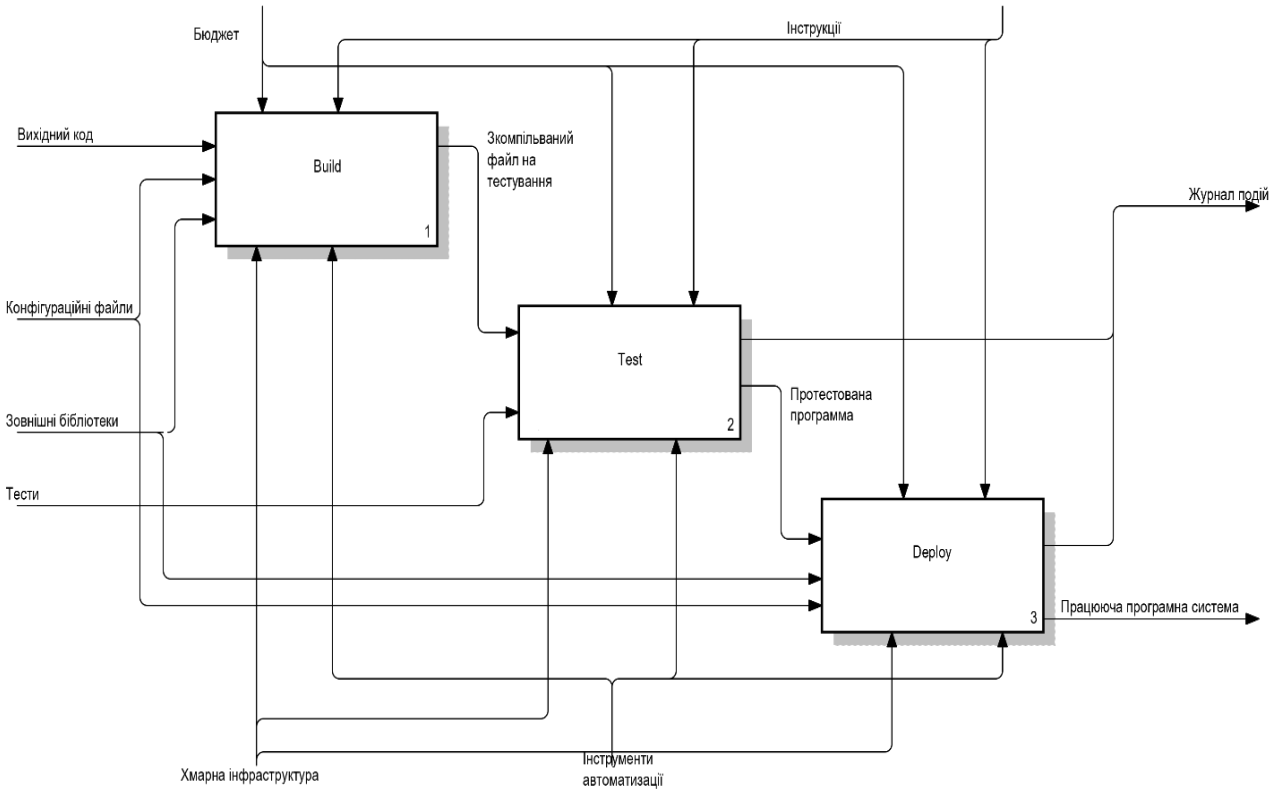

Рис. 6. Декомпозиційна діаграма автоматичного розгортання програмної системи із використанням ресурсів дата центрів хмарних провайдерів в нотації IDEF0

Проведений аналіз функціональної взаємодії етапів Build та Deploy в рамках моделювання процесу розгортання програмної системи із використанням ресурсів дата центрів хмарних провайдерів на основі розробленої контекстної та декомпозиційної діаграм IDEF0, представлених на рис. 5 - 6 показав, що саме ці етапи мають найбільше функціональне навантаження та потребують більш детального дослідження, з метою пошуку шляхів їх оптимізації.

З цією метою було розроблено та проведено аналіз декомпозиційних діаграм IDEF0 та DFD, які моделюють функціональну взаємодію складових підпроцесів Build і Deploy, та потоки даних, що породжуються і передаються між ними, відповідно.

На рис. 7 - 8 представлені декомпозиційні діаграми в нотації IDEF0 та DFD підпроцесу Build.

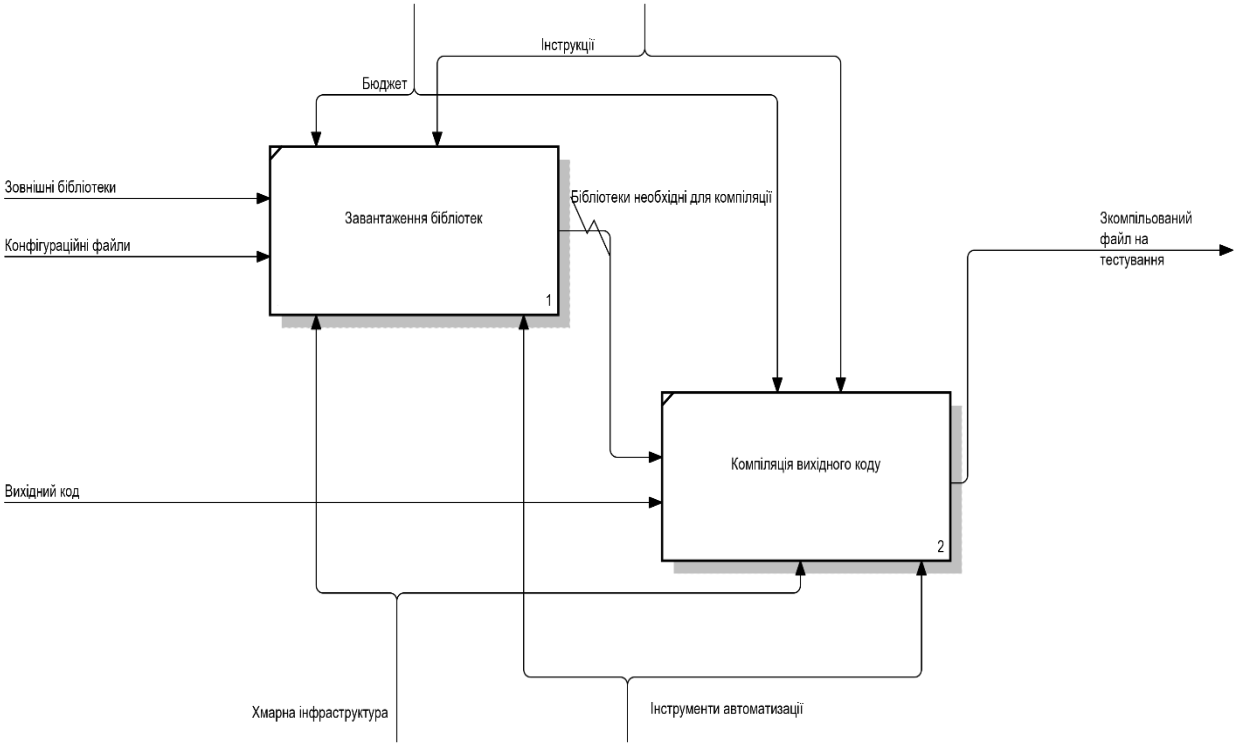

Рис. 7. Декомпозиційна діаграма підпроцесу Build в нотації IDEF0

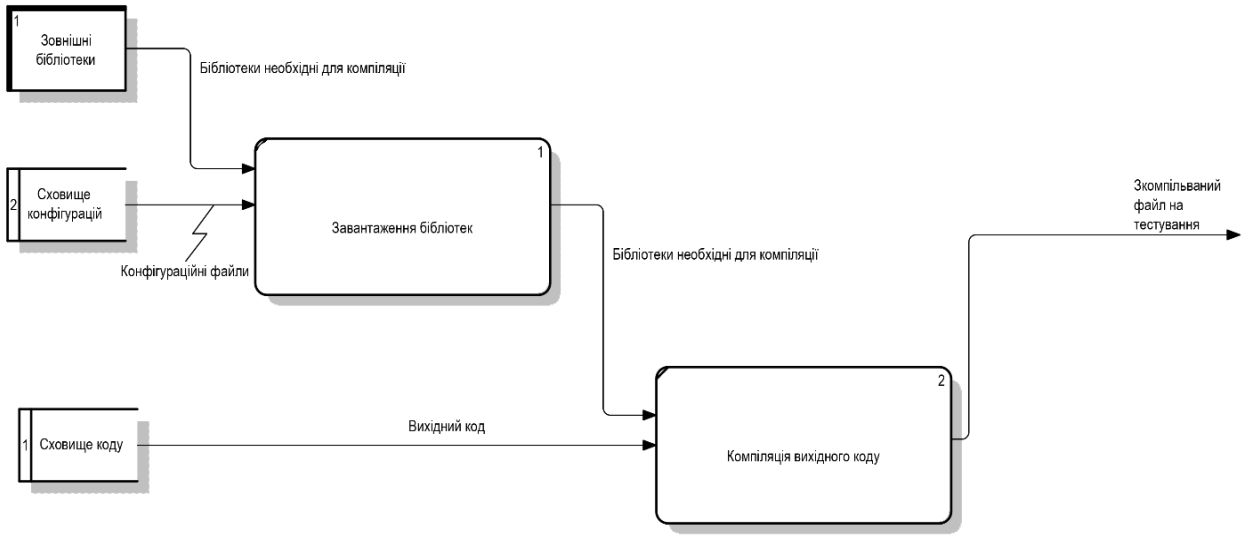

Рис. 8. Декомпозиційна діаграма підпроцесу Build в нотації DFD

Аналіз функціональної взаємодії та потоків даних, які породжуються та передаються між складовими підпроцесу Build показав, що для кожної компіляції вихідного коду відбувається завантаження зовнішніх бібліотек, а це в свою чергу призводить до суттєвого збільшення зовнішнього трафіку.

Розглянемо декомпозиційні діаграми в нотації IDEF0 та DFD підпроцесу Deploy, які представлені на рис. 9 – 10.

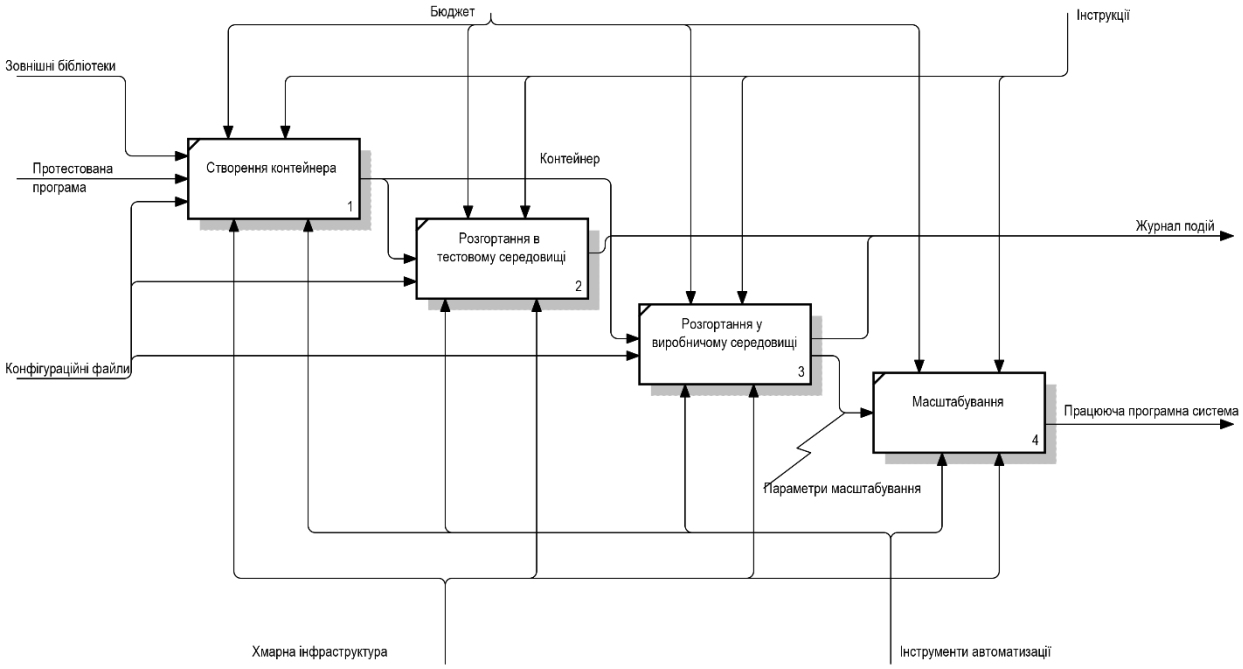

Рис. 9. Декомпозиційна діаграма підпроцесу Deploy в нотації IDEF0

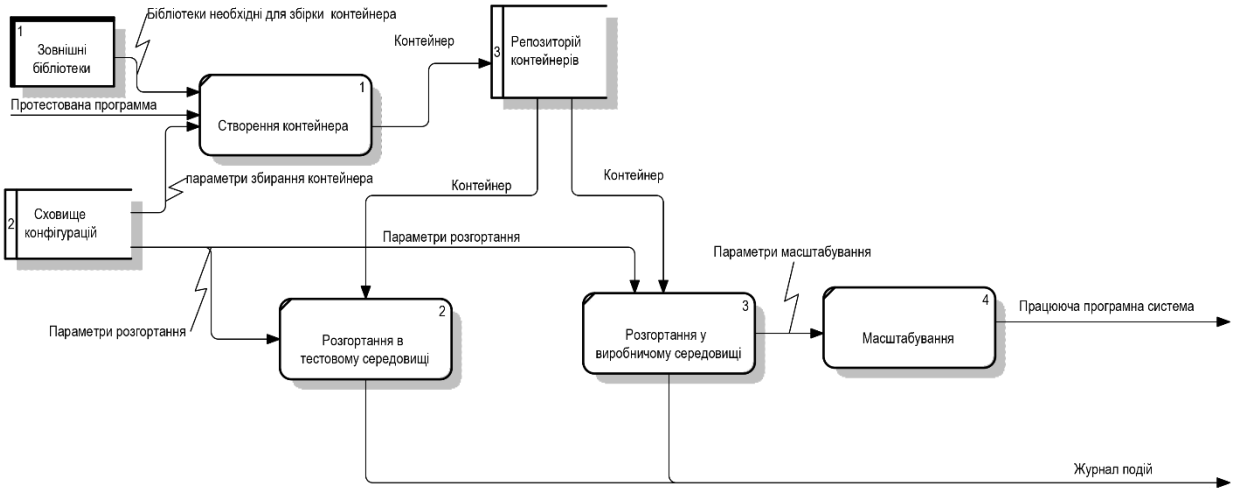

Рис. 10. Декомпозиційна діаграма підпроцесу Deploy в нотації DFD

Проведений аналіз функціональної взаємодії та потоків даних, які породжуються та передаються між складовими підпроцесу Deploy виявив аналогічну особливість, що притаманна підпроцесу Build, а саме - завантаження даних із зовнішніх бібліотек відбувається кожного разу при створенні койнтейнера.

Таким чином, ми маємо ситуацію, коли при кожному виконанні компіляції вихідного коду та створенні койнтейнера необхідно завантажувати зовнішні бібліотеки, що фактично призводить до суттєвого збільшення зовнішнього трафіку.

Шляхом оптимізації цих підпроцесів може бути створення кешу, де завантажені дані з зовнішніх бібліотек мали б можливість зберігатись для повторного використання, а також забезпечення контролю над здійсненням запитів до зовнішніх бібліотек та процесом кешування даних.

Висновки. Таким чином, основним отриманим науковим і практичним результатом даного дослідження є розроблена модель процесу управління розгортанням програмної системи із використанням ресурсів дата центрів хмарних провайдерів, яка шляхом аналізу функціональної взаємодії та потоків даних, що породжуються та передаються між окремими складовими процесу розгортання програмної системи, дозволила виявити важливий аспект підпроцесів Build та Deploy, який полягає в тому, що при кожному виконанні компіляції вихідного коду та створенні койнтейнера потрібно завантажувати данні із зовнішніх бібліотек. Це в свою чергу фактично призводить до суттєвого збільшення зовнішнього трафіку, що впливає на швидкість розгортання програмних систем із використанням ресурсів дата центрів хмарних провайдерів.

Як можливий шлях оптимізації підпроцесів Build та Deploy запропоновано створення кешу, для збереження завантажених даних із зовнішніх бібліотек з метою їх повторного використання, а також реалізацію контролю над здійсненням запитів до зовнішніх бібліотек та процесом кешування даних, що дозволить мінімізувати витрати часу на передачу даних із зовнішніх бібліотек під час автоматичного розгортання.

Результати цього дослідження можуть слугувати основою для подальшої розробки способів оптимізації процесу управління розгортанням програмних систем в хмарних середовищах, сприяючи подальшому розвитку та впровадженню сучасних підходів у галузі інформаційних технологій.

Список літератури

1. Chandrasekara, C. *Beginning Build and Release Management with TFS 2017 and VSTS*., 2017, Available at: http://dx.doi.org/10.1007/978-1-4842-2811-1.
2. Fowler, M. and Beck, K. *Continuous Integration: Improving Software Quality and Reducing Risk*., Pearson Education, Boston, US. 2006.
3. Humble, J. and Farley, D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, Boston, US. 2010.
4. Coupaye, T. and Estublier, J. *Foundations of enterprise software deployment*, 2000, Available at: https://doi.org/10.1109/CSMR.2000.827313.
5. National Institute for the Software Industry (NISI) *Continuous Delivery 3.0 Maturity Model (CD3M)*, 2019, Available at: https://nisi.nl/continuousdelivery/articles/maturity-model.
6. Grandhi, M. *Optimizing Your Modernization Journey with AWS*. Packt Publishing, Birminham, UK. 2023.
7. Vaughan, D. *Cloud Native Development with Google Cloud: Building Applications at Speed and Scale*, 2023, Available at: https://www.oreilly.com/catalog/errata.csp?isbn=0636920829249.
8. Rawat, S. *CI CD Pipeline with Docker and Jenkins*, BPB Online, London, UK. 2023.
9. Saini, K. *Build Process*, 2022, Available at: https://iq.opengenus.org/build-process/.
10. Gurung G. Cloud Deployment Models: A Comparative Analysis For Optimal Performance, Scalability, And Cost Efficiency. Conference: *6th National Conference NATCOM (AICGPT-2023*), 2023, Available at: https://www.researchgate.net/publication/374447308_Cloud_Deployment_Models_A_Comparative_Analysis_For_Optimal_Performance_Scalability_And_Cost_Efficiency.
11. Van Merode, H. *Continuous Integration (CI) and Continuous Delivery (CD)*, Leeuwarden, NL. 2023, pp. 1-9. Available at: https://doi.org/10.1007/978-1-4842-9228-02023.
12. Rani, M. *What is software build- All you need to know!* 2019 Available at: https://medium.com/webgentle/what-is-the-software-build-all-you-need-to-know-4046b0e674bb.
13. Shah, Jay & Dubaria, Dushyant. Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform. *in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, Available at: https://doi.org/10.1109/ccwc.2019.8666479.
14. Zhai, H., Wang, J. Automatic deployment system of computer program application based on cloud computing. *Int J Syst Assur Eng Manag* 12, 2021, pp.731–740. Available at: https://doi.org/10.1007/s13198-021-01068-0.
15. Mustafa, O., *AWS Deployment Strategies. In: A Complete Guide to DevOps with AWS*. Apress, Berkeley, CA. 2023 pp 183–196 Available at: https://doi.org/10.1007/978-1-4842-9303-4_5.

Надійшла (received) 24.01.2024

Відомості про авторів / Сведения об авторах / About the Authors

Пашнєв Андрій Анатолійович (Pashniev Andrii) – кандидат технічних наук, старший науковий співробітник, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри інформаційних систем та технологій; e-mail: pashniev@email.ua. ORCID: https://orcid.org/0000-0002-9150-6108.

Слєпушков Микола Васильович (Sliepushkov Mykola) – Національний технічний університет «Харківський політехнічний інститут», студент; e-mail: m.sliepushkov@gmail.com. ORCID: https://orcid.org/0009-0001-0004-2820.

Гурт Денис Олександрович (Hurt Denys) – фізична особа підприємець; e-mail: denys.hurt@gmail.com. ORCID: https://orcid.org/0009-0000-3880-9081.

Лютенко Ірина Вікторівна (Liutenko Iryna) – кандидат технічних наук (PhD), доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри програмної інженерії та інтелектуальних технологій управління; м. Харків, Україна, e-mail: cherliv68@gmail.com, ORCID: https://orcid.org/0000-0003-4357-1826.