УДК 004.89 DOI: 10.31673/2412-4338.2020.032740

Вишнівський В.В., Гринкевич Г.О., Макаренко А.О., Жебка В.В., Стрельніков В.І., Штіммерман А.М. *Державний університет телекомунікацій*

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АВТОМАТИЗАЦІЇ РОЗРОБКИ ТА РОЗГОРТАННЯ ВІРТУАЛЬНОГО ХМАРНОГО СЕРЕДОВИЩА НА ОСНОВІ БЕЗСЕРВЕРНОЇ АРХІТЕКТУРИ

Анотація: *В статті представлено структуру інформаційної технології автоматизації розробки та розгортання віртуального хмарного середовища на основі безсерверної архітектури. Представлено процес створення інформаційної технології автоматизації розробки та розгортання віртуального хмарного середовища на основі безсерверної архітектури, проведено оцінку представленої метамоделі для прототипового тематичного дослідження та запропоновано можливі майбутні напрямки удосконалення.*

Незважаючи на те, що для розгортання та управління складними хмарними сервісами доступно багато інструментів забезпечення, користувачі часто повинні вказувати рішення IaC за допомогою сценаріїв низького рівня. Натомість відомі рішення вимагають, щоб розробник вказував компоненти програми, такі як веб-додаток, а бажано, щоб фреймворк автоматично перетворював би їх на код, що розгортається. Для вирішення цих проблем та реалізації бажаних можливостей пропонується модельована та масштабована, структура швидкого забезпечення під назвою Інформаційна технологія автоматизації розробки та розгортання віртуального хмарного середовища на основі безсерверної архітектури (ITapрx). ITapрx відповідає вимогам специфікації топології для хмарних додатків, що дозволяє створювати портативні та взаємодіючі шаблони для хмарних сервісів. Використання OpenTOSCA забезпечує стандартизацію розмежування програмних додатків та їх залежностей від специфікацій хмарної платформи.

Основними внесками в цій роботі є те, що представлено ключові елементи моделювання ITapрx, який автоматизує та маскує низькорівневі деталі специфікацій компонентів додатків та специфікацій постачальника хмарних послуг, а натомість пропонує інтуїтивні уявлення високого рівня; запропоновано використання розширеної бази знань та алгоритмів для автоматичного виконання перетворень IaC; зроблено конкретну реалізацію ITapрx та його перевірку в контексті реальних випадків використання.

Ключові слова: *інформаційна технологія, IoT, машинне навчання, хмарні сервіси, віртуальна машина, автоматизація.*

Vyshnivsky V., Grynkevych G., Makarenko A., Zhebka V., Strelnikov V., Shtimmerman A.
State University of Telecommunications, Kyiv

INFORMATION TECHNOLOGY FOR AUTOMATION OF DEVELOPMENT AND DEPLOYMENT VIRTUAL CLOUD ENVIRONMENT BASED ON SERVERLESS ARCHITECTURE

Abstract: *The article presents the structure of information technology to automate the development and deployment of a virtual cloud environment based on a serverless architecture. The process of creating information technology for automation of development and deployment of a virtual cloud environment based on serverless architecture is presented, the presented metamodel for prototype case study is evaluated and possible future directions of improvement are suggested.*

Although many software tools are available for deploying and managing complex cloud services, users often have to specify IaC solutions using low-level scenarios. Instead, known solutions require the developer to specify program components, such as a web application, and preferably the framework would

automatically convert them to expandable code. To solve these problems and implement the desired capabilities, a modeled and scalable, fast-supply structure called Information Technology for Automating the Development and Deployment of a Virtual Cloud Environment Based on a Serverless Architecture (ITappx) is proposed. ITarrch meets the requirements of the topology specification for cloud applications, which allows you to create portable and interoperable templates for cloud services. The use of OpenTOSCA provides standardization of the delimitation of software applications and their dependencies on the specifications of the cloud platform..

The main contributions to this acticle are that the key elements of ITappx modeling are presented, which automates and masks the low-level details of application component specifications and cloud service provider specifications, and instead offers high-level intuitive representations; the use of an extended knowledge base and algorithms for automatic execution of IaC transformations is proposed; made a specific implementation of ITappx and its verification in the context of actual use cases.

Keywords: *information technology, IoT, mathematical learning, cloud services, virtual machine, automation, domain modeling.*

Вишневский В.В., Гринкевич А.А., Макаренко А.А., Жебка В.В., Стрельников В.И., Штиммерман А.Н. *Государственный университет телекоммуникаций, Киев*

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ АВТОМАТИЗАЦИИ РАЗРАБОТКИ И РАЗВЕРТЫВАНИЯ ВИРТУАЛЬНОЙ ОБЛАЧНОЙ СРЕДЫ НА ОСНОВЕ БЕЗСЕРВЕРНОИ АРХИТЕКТУРЫ

Аннотация: *В статье представлена структура информационной технологии автоматизации разработки и развертывания виртуального облачной среды на основе безсерверной архитектуры. Представлены процесс создания информационной технологии автоматизации разработки и развертывания виртуальной облачной среды на основе безсерверноu архитектуры, проведена оценка представленной метамодели для прототипов тематического исследования и предложены возможные будущие направления совершенствования.*

Несмотря на то, что для развертывания и управления сложными облачными сервисами доступно много инструментов обеспечения, пользователи часто должны указывать решение IaC с помощью сценариев низкого уровня. Зато известные решения требуют, чтобы разработчик указывал компоненты программы, такие как веб-приложение, а желательно, чтобы фреймворк автоматически превращал бы их на код, что разворачивается. Для решения этих проблем и реализации желаемых возможностей предлагается моделируемая и масштабируемая структура быстрого обеспечения под названием Информационная технология автоматизации разработки и развертывания виртуальной облачной среды на основе безсерверной архитектуры (ITappx). ITappx соответствует требованиям спецификации топологии для облачных приложений, позволяет создавать портативные и взаимодействующие шаблоны для облачных сервисов. Использование OpenTOSCA обеспечивает стандартизацию разграничения приложений и их зависимостей от спецификаций облачной платформы.

Основными взносами в этой работе является, то что представлено ключевые элементы моделирования ITappx, который автоматизирует и маскирует низкоуровневые детали спецификаций компонентов приложений и спецификаций поставщика облачных услуг, а взамен предлагает интуитивные представления высокого уровня; предложено использование расширенной базы знаний и алгоритмов для автоматического выполнения преобразований IaC; сделано конкретную реализацию ITappx и его проверку в контексте реальных случаев использования.

Ключевые слова: *информационная технология, IoT, машинное обучение, облачные сервисы, виртуальная машина, автоматизация.*

1. Вступ

Розгортання та управління додатками самообслуговування необхідно для підприємств, щоб пришвидшити час виходу на ринок своїх хмарних послуг. Це стало необхідним,

оскільки сучасні хмарні сервіси часто будуються як мікросервіси, де кожен із компонентів повинен бути налаштований та розгорнутий на хмарних платформах у певному порядку. Інфраструктура та надання послуг для цих складних випадків реалізації з використанням середовищ сценаріїв низького рівня погіршують продуктивність та негативно впливають на час випуску релізу.

2. Аналіз останніх досліджень і публікацій. Порівнюються існуючі в літературі зусилля щодо управління ресурсами, розгортання та управління додатками самообслуговування для пришвидшити часу редлізу своїх хмарних послуг на основі безсерверної архітектури з даною роботою.

Сьогодні суспільство використовує рішення для оркестрації, такі як CloudFoundry [1], Cloudify [2] тощо, зокрема, у поєднанні з інструментами автоматизації, такими як Ansible, Puppet та Chef.. Однак різні виміри змінних (тобто вирішення питань сумісності програми та несумісних API провайдерів хмарних служб) ускладнюють ручну роботу зі створенням сценаріїв із використанням цих інструментів. У цьому контексті Alien4Cloud [3] пропонує візуальний спосіб генерування моделі топології OpenTOSCA (Topology and Orchestration Specification for Cloud Applications) [4], яка може бути організована Apache Brooklyn. Однак побудова належної топології навіть з використанням підходів, керованих моделями, потребує знань у галузі інформаційних технологій. На відміну від цих підходів, ITapрх абстрагує всі додаткові та специфічні для хмари деталі в метамоделі свого DSML (Domain-Specific Modeling Language) [5] і трансформує бізнес-модель у сумісний з OpenTOSCA IaC формі.

Для зменшення складності розгортання послуг пропонується кілька підходів, заснованих на шаблонах [6]. Аналогічним чином, модельовані шаблони перевірених рішень використовуються для розгортання послуг у хмарних інфраструктурах [7]. Наприклад, MODA-Clouds [8] дозволяє користувачам розробляти та розгортати компоненти додатків для роботи та управління в багатохмарних середовищах за допомогою системи підтримки прийняття рішень. Подібно ITapрх, вони також підтримують повторне використання та ітеративне вдосконалення на основі прописаних правил. Однак для створення плану розгортання бракує засобів перевірки та розширюваності.

Певні можливості надають: ConfigAssure [9] – вирішувач вимог для синтезу конфігурації інфраструктури декларативним способом. Aeolus Blender [10] включає оптимізатор конфігурації Zephyrus [11], спеціальний планувальник Metis та механізм розгортання Arnomic. Zephyrus автоматично генерує абстрактну конфігурацію бажаної системи на основі часткового опису. На відміну від використання бази знань (БЗ- база знань - knowledge base) [12] для трансформації бізнес-моделі використовують вирішувач проблем із обмеженням (Constraint Satisfaction Problems - CSP). Однак вирішення CSP може зайняти значний час для виконання, а для визначення обмежень у конфігураціях потрібна експертиза доменів, яка не потрібна в ITapрх.

Подібно ITapрх, Hirmer et al. [13] слід зосередитися на створенні повної топології, сумісної з TOSCA, із частковою топологією бізнесу, що відповідає бажанням користувачів, за допомогою ланцюга інструментів TOSCA [14]. CELAR [15] поєднує специфікації TOSCA для автоматизації розгортання хмарних додатків, де завершення топології виконується аналізом вимог та можливостей на шаблоні вузла. На відміну від цих зусиль, перетворення моделі в ITapрх базується на запитах до БЗ.

3. Мета дослідження: Побудова належної топології інформаційної технології автоматизації розробки та розгортання віртуального хмарного середовища на основі безсерверної архітектури для підвищення швидкодії системи вцілому, що важливо для сучасних технологій інтернету речей.

4. Результати дослідження.

4.1. Опис проблеми. Розглядається випадок розгортання служби на базі LAMP-стек (Linux, Apache, MySqL, PHP) на хмарній платформі. На рис. 1 показана бажана топологія хмарних додатків, що складається з двох підключених стеків програмного забезпечення, тобто веб-інтерфейсу на базі PHP, та серверної бази даних MySQL. Лівий бічний стек сервісної моделі містить бізнес-логіку, оскільки інтерфейс буде розгорнуто на сервері Ubuntu 16.04. Цей сервер розміщуватиметься на віртуальній машині, а віртуальне середовище буде керуватися за допомогою хмарної платформи OpenStack. У правій частині стеку відображається реляційна база даних, яка використовується для зберігання та запиту даних про товар. База даних – це СУБД MySQL, яка буде розгорнута на певному екземплярі віртуальної машини Google Cloud Platform [16] із сервером Ubuntu 16.04 [17].

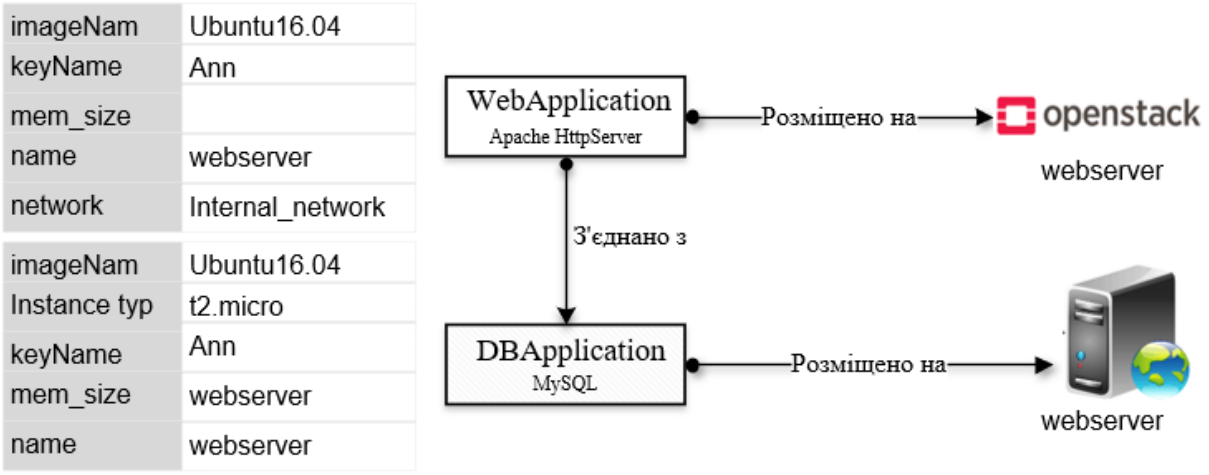

Рис. 1. Бажаний рівень абстракції для бізнес-моделі

Користувачу потрібно забезпечити стек програм на основі PHP та MySQL з двох аспектів. В аспектах надання хмарної інфраструктури топологію програми потрібно встроїти в середовище виконання, яке може бути віртуальними машинами, контейнерами або сторонніми службами. Розробнику потрібно вибрати належний образ, наданий хмарним провайдером, разом із групою безпеки, правилами, мережею, кількістю екземплярів, блоком зберігання, щоб створити нові віртуальні машини на цільовій хмарній платформі.

У аспектах надання послуг потрібно встановити все залежне програмне забезпечення та налаштувати всі обмеження. Наприклад, для інтерфейсу даного прикладу, Apache Httpd потрібно встановити та налаштувати разом з PHP та Java. Подібним чином у серверній системі потрібно встановити та налаштувати MySQL. Крім того, веб-програма вимагає встановлення драйвера підключення до бази даних PHP для доступу до бази даних. Процес встановлення програмного забезпечення залежить від операційних систем, їх версій та менеджерів пакетів. Більше того, служба баз даних повинна запускатися до служби додатків PHP, щоб належним чином запустити WebApp. Рішення IaC для підготовки додатків вимагає всіх деталей встановлення та конфігурації для виконання плану розгортання.

Приклад, що використовується у статті показує, що користувач повинен володіти великими знаннями в сфері доменів, щоб правильно забезпечити навіть простий стек веб-додатків. Незважаючи на те, що для розгортання та управління складеними хмарними службами доступні багато інструментів забезпечення, користувачі часто повинні вказувати рішення IaC за допомогою сценаріїв низького рівня. Натомість бажана можливість вимагала б, щоб по-перше розробник вказував лише компоненти програми, такі як веб-програма та по-друге фреймворк автоматично перетворював би бізнес-модель на артефакти, що розгортаються.

Для вирішення цих проблем та реалізації бажаних можливостей пропонується модельована та масштабована, структуру швидкого забезпечення під назвою ITappx. Модель ITappx відповідає вимогам специфікації топологій та оркестрації для хмарних додатків OpenTOSCA, що дозволяє створювати портативні та взаємодіючі шаблони планів як послуги для хмарних послуг. Використання OpenTOSCA забезпечує стандартизацію розмежування програмних додатків та їх залежностей від специфікацій хмарної платформи.

4.2. Створення та впровадження методу ITappx. Далі розглядається створення методу ITappx. Спочатку сформульовані ключові вимоги, яким повинен відповідати ITappx, а потім розглянуто, як запропонований створення відповідає цим вимогам.

А). Хмарна платформа самообслуговування, така як ITappx, повинна вимагати мінімальних специфікацій від користувача, якому не потрібно володіти глибокими знаннями доменів та максимально автоматизувати процес надання лише компонентів програми. Нижче окреслюються ключові вимоги, яким повинна відповідати ITappx.

Вимога 1: Зменшення деталізації специфікації – ITappx повинен абстрагувати деталі специфікації від користувачів, визначивши спільність стеків резервування, які стають будівельними блоками конвеєру розгортання та управління, які фіксуються як елементи для конкретного домену. Тоді користувач вводить мінімальну кількість вводних данних.

Вимога 2: Автозавершення – ITappx повинен визначити правила трансформації, які перетворюють абстрактні бізнес-моделі в цілісне до побудови завершене, таке, що розгортається сумісне з OpenTOSCA рішення. В тому разі і IaC [18].

Вимога 3: Підтримка постійної інтеграції, доставки та міграції – оскільки можливо, що існуюче розгортання може змінити інфраструктуру (наприклад, змінити хмарну платформу) або змінити додаток (наприклад, замінити технологію сервера баз даних) або обидва ITappx повинна розділити на два етапи та безперебійно підтримувати безперервну інтеграцію, доставку та міграцію.

Б). Моделювання ITappx для доменних моделей DSML: ITappx DSML абстрагується від складності проектування, відокремлюючи програму від технологій розгортання та інфраструктури відповідно до специфікації OpenTOSCA, як описано у вимозі 3 ITappx DSML розроблений із використанням плагіна WebGME [19] та використовує JavaScript, NodeJS та базу даних MySQL.

Метамодель автоматизації моделювання розгортання ITappx була розроблена, використовуючи поєднання:

1) Зворотного проектування.
2) Відображення залежностей між різнорідними хмарами.
3) Відображення залежностей між різними операційними системами та їх версіями.
4) Семантичного відображення.
5) Бізнес стратегії.
6) Створення прототипів.

Характеристики хмарних провайдерів та додатків, вимоги до програмного забезпечення, стратегії та інша інформація, що стосується реалізації послуг та всі інші відомі обмеження, попередньо визначені як будівельні блоки високого рівня в метамоделі.

З цією метою ITappx пропонує різні типи вузлів відповідно до специфікацій OpenTOSCA, які є компонентами програми, та умовами різних постачальників хмарних послуг. Мета полягає в конкретизації абстрактного типу вузла шляхом узгодження бажаної специфікації розробника із заздалегідь визначеними функціональними можливостями, закладеними в метамоделі ITappx та базі знань. Потім конкретні шаблони вузлів поєднуються з певними типами постачальників хмарних служб та їх віртуальними машинами, щоб створити графік залежностей, який повинен бути виконаний для розгортання компонентів програми в певному порядку на бажаній цільовій машині (машинах). Використовуючи запропонований DSML, розробник може легко налаштувати вузол на

визначеній хмарній платформі або конкретній цільовій системі.

В результаті проведених досліджень отримані наступні результати:

1) При розробці метамоделі для хмарних платформ спостерігався процес розміщення додатків у різних хмарних середовищах та фіксувалися всі спільні характеристики змінних. При цьому використовувалися специфікації для різних хмарних платформ, таких як OpenStack, Amazon AWS, Microsoft Azure тощо для забезпечення віртуальних машин з різними операційними системами (ОС). При цьому розробники можуть вибрати бажані образи ОС для створення віртуальних машин. При цьому розробник може вибрати заздалегідь визначений тип віртуальної машини, доступні мережі, групи безпеки, правила та доступні бібліотеки, які всі визначені як змінні в запропонованій метамоделі. Вони також повинні вказати свій файл середовища, секретний ключ для обраних типів хост-хмар, які є кінцевими точками для прив'язки до певного хмарного провайдера, як показано на рис. 1. За бажанням, попередньо розгорнута машина може бути вказана відповідним IP-адресом та типом ОС. Доступні служби та типи віртуальних машин для хмарних платформ також заздалегідь визначені в метамоделі.

2) Створення метамоделі компонентів додатків. Для служб, розміщених у хмарі, ITapрх надає різні типи вузлів для компонентів програми, таких як веб-програма, програма баз даних, програма DataAnalytics тощо. Наприклад, метамодель дозволяє розробнику вибрати атрибут веб-сервера, мову для коду, атрибут сервера бази даних або атрибути бази даних NoSQL із наданого списку. Розробник повинен вказати атрибути змінної для розгортання бажаного типу компонента програми.

3) Визначення взаємозв'язку між компонентами. Чотири типи відносин зв'язують типи вузлів у метамоделі наступним чином:

– тип взаємозв'язку «hostedOn» означає, що тип вузла-джерела потрібно розгорнути на типі вузла призначення;

– тип відносин «connectsTo» використовується для впорядкування розгортання, щоб пов’язати кінцеву точку типу вихідного вузла з необхідною кінцевою точкою цільового вузла, якщо вони залежать;

– тип підключення «deleteFrom» визначає тип вихідного вузла, який потрібно видалити з типу кінцевого вузла;

– тип підключення «migrateTo» визначає тип вихідного вузла, який потрібно перенести на кінцевий тип вузла (тип відношення «migrateTo» не можна визначити без типу з'єднання «deleteFrom»).

Розширюваність метамоделі: ITapрх – це повноцінний додаток, проте, з великою кількістю свободи дій. Метамодель була розроблена для розширюваності, щоб у майбутньому була можливість додати більше типів вузлів програми. Додавання нового компонента програми вимагає багато часу; однак це одноразові зусилля і їх можна повторно використовувати. Метамоделі ITapрх є розширюваними та багаторазовими, тому нові типи компонентів та платформи можуть бути додані відповідно до вимог у метамоделі ITapрх.

B) Створення БЗ ITapрх. БЗ ITapрх містить базу даних та шаблони типів програм.

1) Створення бази даних БЗ: Діаграма моделі «сутність-зв'язок» бази даних БЗ зображена на рис. 2, де показано набори елементів, що зберігаються в базі знань. Зроблено у вигляді чотирьох таблиць: os_pkg_mgr, os_ dependency, таблиця пакетів даних та sw_dependency для побудови бази даних БЗ. При цьому збережено: всі операційні системи, їх дистрибутиви, менеджер пакетів та версії в таблиці os_pkg_manager; усі доступні типи компонентів додатків, наприклад, веб-додаток на основі PHP, додатки БД на основі MySQL тощо, у таблиці залежностей; всі програмні пакети, необхідні для певного типу додатків, знаходять за допомогою зворотного проектування та зберігаються в таблиці пакетів. Обробляться таблиця пошуку вручну, щоб обробляти всі точки змінних. Зразок розділу структури таблиці бази даних показаний на рис. 3.

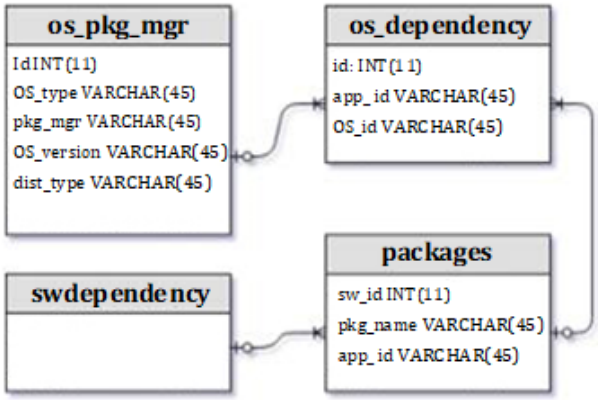

Рис. 2. Модель «сутність-зв'язок» (ER-модель) з базою даних ITappx

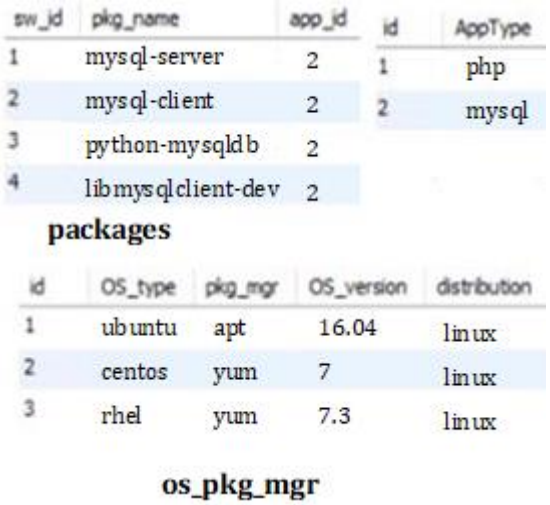

Рис. 3. Приклад частини таблиць бази даних БЗ

2) Розробка шаблону БЗ: Шаблони БЗ розроблені, враховуючи спільність компонентів програми, і містять шаблони, які потрібно заповнити ITappx DSML, запитуючи базу даних БЗ.

3) Розширюваність БЗ: БЗ розширюється за задумом. Додавання нових компонентів програми вимагає розробки нових шаблонів (принаймні частково) шляхом зворотного проектування стеку програмного забезпечення. Потрібно визначити спільні характеристики та змінні, і відповідно до цього слід розробити шаблон. Також визначити залежності програмного забезпечення для компонентів програми, які потрібно вставити в таблиці бази даних знань.

4) Підтримка безперервної доставки: ITappx також може обробляти безперервну доставку та додавання/видалення компонентів (Вимога 3), що є лише питанням оновлення моделі з додаванням або видаленням компонента. Наприклад, щоб додати новий сервер баз даних, користувач розширює модель типом вузла DBApplication та налаштовує взаємозв'язки "connectsTo" з веб-сервера на сервер баз даних. ITappx генерує IaC для нещодавно доданого компонента та виконує його для розгортання доданого компонента, не перешкоджаючи доступності існуючого додатка. Оскільки Ansible при повторному використанні дає той же результат, він завжди встановлює однакову конфігурацію в цільовому середовищі незалежно від їх поточного стану.

5) Перевірка обмежень на правильність бізнес-моделей: здійснюється перевірка бізнес-моделей, при цьому визначається наявність порушень обмежень, якщо вони відсутні робиться висновок, що моделі добре сформовані та “правильно зроблені”. Додатково перевіряється правильність конфігурацій кінцевих точок для всіх типів компонентів

програми, всіх типів відносин, всіх типів характерних для хмари тощо, а також моделі зайнятості в цілому перед створенням будь-якого IaC. Робляться також перевірки стосовно інших обмежень на основі правил, для того щоб перевірити сумісність компонентів. Наприклад, пунктом призначення потоку доставки Amazon Kinesis мають бути послуги Amazon (наприклад, Redshift, S3); це не можуть бути служби Azure або OpenStack.

Г) Далі представляємо процес трансформації ITappx, який використовує DSML та Базу знань.

1) Виходячи з вищесказаного на рис. 4 представлено алгоритм розгортання ITappx. Перший блок в алгоритмі відповідає за введення вхідних даних та визначення класів хмарних і програмних моделей.

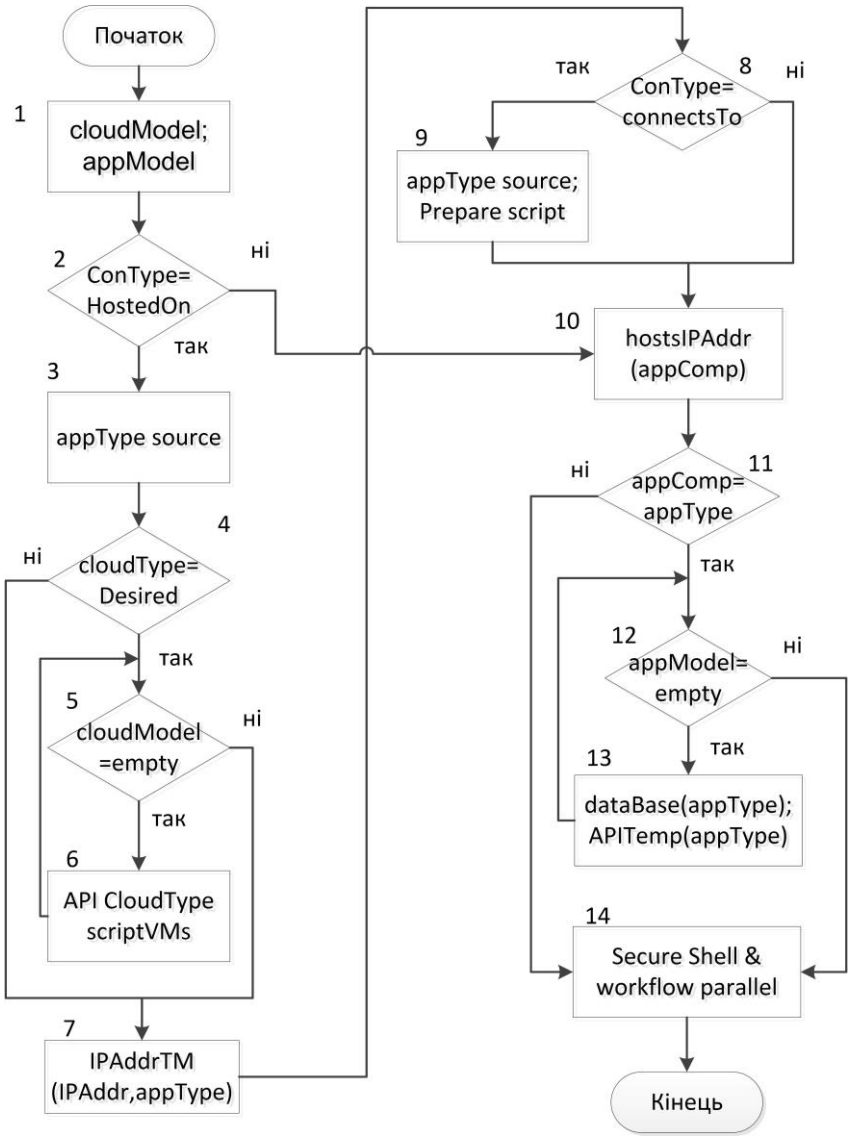

Рис. 4. Алгоритм розгортання ITappx

БЗ для створення рішення розгортання інфраструктури як коду: можливості генерації ITappx увімкнені через плагін WebGME, який користувач викликає після процесу моделювання. Він генерує та виконує IaC, як описано в алгоритмі. Віртуальні машини створюються у зазначеній хмарній платформі на основі призначення з'єднання "HostedOn" [алгоблоки 1-5]. Там, де це можливо, ITappx забезпечить, щоб сценарії, характерні для підготовки, працювали паралельно, для забезпечення більш швидкого розгортання. Після створення віртуальної машини, GenerateConfig () запитує БЗ [алгоблоки 7-10], щоб

заповнити appModel [алгоблок 12] на основі специфікацій користувача. Потім результат запиту заповнює специфічні для програми попередньо визначені шаблони конфігурації та генерує IaC, для автоматизації використовується Ansible, для конкретних компонентів програми [алгоблоки 12-14], використовуючи перетворення на основі шаблону. Подібний підхід застосовується для конфігурування контейнерів для конкретної послуги або для запуску хмарних служб

2) Визначення порядку розгортання та виконання: Сценарій NodeJS у ITapрx створює дерево залежностей для всіх типів програм, визначених у метамоделі, та подає його до механізму робочого циклу оркестрації. Після цього генеруються сценарії для засобів автоматизації (Ansible playbooks) для різних типів компонентів, і ці інструменти, у свою чергу, паралельно відправляють завдання на кілька хостів. Якщо в моделі існує взаємозв'язок «connectsTo», то дозволяється спочатку завершити залежний сценарій, визначаючи ланцюжок залежностей [алгоблоки 7-9].

3) Генерування інфраструктури як коду для міграції. Для міграції компонентів програми на ITapрx визначає, звідки користувач хоче перемістити компоненти програми, та приєднує тип підключення «migrateTo», щоб вказати пункт призначення. Тип відношення «migrateTo» неможливо визначити без типу підключення «delete-From», щоб забезпечити правильність моделі. DSML створить нову віртуальну машину з новою операційною системою для вузла призначення «migrateTo» і згодом видалить поточний вузол.

4.3. Оцінка переваг моделі ITapрx. Далі опишемо результати порівняння часу та зусиль, покладених на розгортання випадків використання додатків, використовуючи

а) ручні зусилля, де розробник повинен увійти в кожну віртуальну машину та ввести команди для встановлення пакетів та розгортання програм,

б) написання сценаріїв для розгортання вручну ці програми,

с) з використанням додатку ITapрx.

Приклад 1: Дослідження розгортання послуг на основі LAMP-стек. Приклад використання: розгортання архітектури мікросервісів на базі Linux, Apache, MySQL та PHP (LAMP-стек), подібний до запропонованого прикладу.

Далі описуються деталі перетворення на основі шаблону, яке відбувається за лаштунками ITapрx DSML. Як зазначено в алгоритмі, DSML проходить дерево бізнес-логіки з рис. 1, яке визначається розробником і збирає всі визначені користувачем атрибути, як показано на рис. 5 та рис. 6. Він заповнює заздалегідь визначений шаблон для конкретного типу програми з визначеними користувачем атрибутами. „Mysql_user“ та „mysql_root_pass“ будуть заповнені із специфікацій, пов’язаних із типом застосування DBA (рис. 5).

Meta type	WebApplication
Attributes	
language	
name	WebApplicantionAn
replication_count	1
src	Https://github.com/Anirban2

Рис. 5. Специфікації, пов’язані з типом WebApplication

Залежності програмного забезпечення компонентів програми збираються шляхом запитів до бази даних БЗ. Наприклад, для встановлення MySQL на машині Ubuntu16.04 потрібні програмні пакети mysql-server та mysql-client. Отже, ITapрx запитує базу даних БЗ і запускає перетворення на основі шаблону, щоб конкретизувати заздалегідь визначений

частковий шаблон. DSML копіює відповідні конфігураційні файли в певні папки, щоб правильно налаштувати MySQL. Таким чином, DSML заповнить заздалегідь визначений файл шаблону з усіма деталями та згенерує IaC, що можна розгорнути. Після генерації всіх специфічних для Ansible файлів ITappx використовує ці файли належним чином для розгортання програми, забезпечуючи хмарну інфраструктуру, як описано в алгоритмі 1.

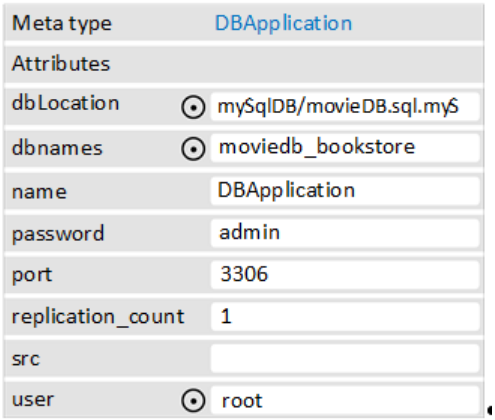

Рис. 6. Специфікації, пов'язані з типом DBAapplication

Нижче описано випадок – використання ручних зусиль. Визначена в результаті невеликого дослідження користувачів хмарних обчислень для тематичного дослідження 1, у якому взяли участь п'ятнадцять команд по три учня в кожній. Користувачам пропонувалось налаштувати файли вручну, створити обробники даних, щоб вказати порядок розгортання на бажаному хості, увійти в кожен хост, де розгортаються компоненти програми, та встановити пакети вручну, налаштувати програмні пакети і, нарешті, запустити різні компоненти. Учасників також попросили написати сценарій у програмному рішенні для віддаленого управління конфігураціями Aansible [20], щоб забезпечити той самий стек програм та інфраструктуру. Вимірювалися час та зусилля для:

а) повністю ручних зусиль;

б) для написання сценаріїв у Ansible та їх виконання вручну;

в) за допомогою середовища ITappx для розгортання сценарію.

Кількісна оцінка на основі дослідження користувача: була створена для проведення дослідження анкета, як показано в таблиці 1. Для кожного питання оціночна шкала становила п.1-10, де одиниця - найпростіше, а десять – найважче рішення.

Таблиця 1.

Кількісна оцінка на основі дослідження користувача

#	Питання
п1	Наскільки легко розгорнути програму PHPMySQL вручну?
п2	Наскільки легко розгорнути PHPMySQL за допомогою інструмента DevOps, такого як Ansible?
п3	Наскільки легко розгорнути PHPMySQL за допомогою ITappx?
п4	Скільки часу та зусиль знадобилося для розгортання програми вручну (за хвилини)?
п5	Скільки часу та зусиль потрібно для розгортання програми за допомогою інструменту DevOps, такого як Ansible (за хвилини)?
п6	Скільки часу та зусиль потрібно для розгортання програми за допомогою ITappx (у хвилинах)?
п7	Яка ймовірність використання платформи ITappx для розгортання програм у майбутньому?

Відповіді на п.1, п.2 та п.3 – простота використання.

Як видно з рис.7, рейтинг "простоти використання" для платформи ITappx набагато вищий порівняно з ручними та сценарійними роботами. Середня складність ручного зусилля оцінюється як 72:2%, а середня складність у виконанні сценаріїв - 71:6%, тоді як середня оцінка складності для використання ITappx становить 30:9%.

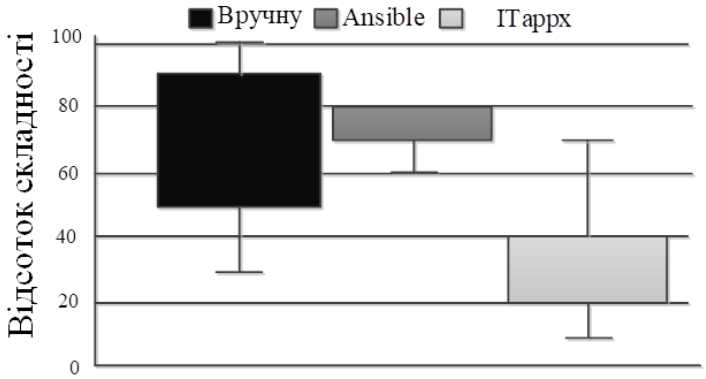

Рис. 7. Порівняння відсотків складності для розгортання служб для різних підходів

Зусилля, докладені користувачем для розгортання моделі LAMP-стек у хмарі, показано в табл. 2: для п.5-п.6, медіана та середнє значення std.dev для часу розгортання, рядки коду, написані для розгортання, час міграції та рядки коду, написані для міграції. При використанні ITappx той самий час розгортання топології становить приблизно 15-20 хвилин для користувачів, які вперше користуються:

Таблиця 2.

Розгортання моделі LAMP-стек у хмарі

	Час розгортання(хв)	Рядки розгортання	Час міграції(хв)	Лінії міграції
медіана	510	300	720	550
значення std.dev	516I244	315I47	653I231	553I142

Відповідь на п7: як показано на рис. 8, 65% респондентів погодились використовувати інструмент ITappx для розгортання хмарних програм у майбутньому, тоді як 30% все ще не впевнені.

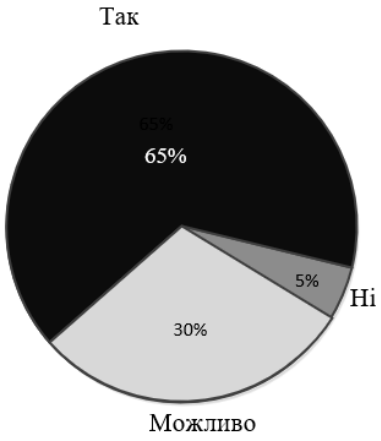

Рис. 8. Ймовірність використання ITappx для майбутнього розгортання хмарних служб

5. Обговорення результатів проведеного дослідження. Наведені результати дослідження переконливо доводять той факт, що ITappx є іноваційним і продуктивним інструментом для розгортання бізнес-додатків.

Платформа ITappx також легко підтримує міграцію компонентів програми, для чого є два типи підключення «deleteFrom» та «migrateTo». Як описано вище, припускаючи, що користувач хоче перенести компонент програми бази даних з однієї машини на іншу машину, яка знаходиться на іншій хмарній платформі OpenStack. Це завдання полягало в міграції служби баз даних MySQL, що відповідає станупереходу з одного вузла на інший вузол, і пропонується додати вузол балансування навантаження, щоб служба стала доступною постійно. ITappx генерує нову структуру робочого циклу на основі змінених специфікацій користувача.

Відповіді на п4, п5 та п6 – час для завершення цілісного налаштування: середній час, який було витрачено на написання сценаріїв для завершення всього процесу міграції, становить 653 хвилини з медіаною 720 хвилин, як показано в таблиці 2. Тоді як приблизні оцінки, які використовують ITappx-міграцію топології, становитимуть лише 10-15 хвилин для користувачів. Середні рядки коду, написані за допомогою ручних зусиль для процесу міграції, складають 553 рядки, згідно з опитуванням, наведеним у табл. 2.

6. Висновки. У роботі представлено модельований підхід до автоматизованої платформи розгортання та управління хмарними додатками. Це допомагає генерувати програми розгортання додатків у моделюванні надання послуг на вищому рівні абстракції та розгортанні коду, не вимагаючи значних знань домену, вимагаючи лише мінімальних зусиль моделювання та відсутність сценаріїв низького рівня. Усі компоненти програми є будівельними блоками в запропонованих модельних середовищах і можуть бути підключені за допомогою відкритих кінцевих точок. DSML генерує рішення IaC «з правильною побудовою» з конвеєра для забезпечення стека додатків у цільовому хмарному середовищі. Використання плагіну WebGME для визначення середовища додатку ITappx дозволяє відокремити його метамодель та БЗ від генеративних аспектів. Запропонований метод ITappx значно підвищує продуктивність та ефективність команд з розгортання та управління додатками.

Список використаної літератури

1. Cloud Foundry. https://www.cloudfoundry.org/community. (дата звернення: 07.03.2020).

2. Multi-Cloud Orchestration. http://getcloudify.org/.(дата звернення: 10.04.2020).

3. Carrasco J., Cubo J., Duran F. and Pimentel E. Bidimensional crosscloud management with tosca and brooklyn, in Cloud Computing (CLOUD), 2016 IEEE 9th International Conference. IEEE, 2016. P. 951 – 956/

4. OpenTOSCA goes Docker Compose https://www.opentosca.org/ (дата звернення: 10.04.2020).

5. Domain-Specific Modeling Language https://www.sciencedirect.com/topics/computer-science/domain-specific-modeling-language (дата звернення: 10.04.2020).

6. Lu H., Shtern M., Simmons B., Smit M., and Litoiu M., Pattern-based deployment service for next generation clouds, Services (SERVICES), 2013 IEEE Ninth World Congress on. IEEE, 2013, P. 464-471.

7. Облачные решения для IT-инфраструктуры https://pirit.biz/reshenija/oblachnye-reshenija (дата звернення: 07.05.2020).

8. Ardagna D., Di Nitto E., Casale G., Petcu D., Mohagheghi P., Mosser S., Matthews P., Gericke A., Ballagny C., D'Andria F. et al., Modaclouds: A model-driven approach for the design

and execution of applications on multiple clouds, Proceedings of the 4th International Workshop on Modeling in Software Engineering. IEEE Press, 2012, P. 50-56.

9. Narain S., Levin G., Malik S., and Kaul V., Declarative infrastructure configuration synthesis and debugging, Journal of Network and Systems Management. 2008. Vol. 16, No. 3. P. 235-258,

10. Di Cosmo R., Eiche A., Mauro J., Zacchiroli S., Zavattaro G., and Zwolakowski J., Automatic deployment of services in the cloud with aeolus blender. Service-Oriented Computing. Springer, 2015, P. 397-411.

11. Deployment Automation. https://doc-vu.github.io/DeploymentAutomation. (дата звернення: 25.06.2020).

12. 8 Best Knowledge Base Software in 2020: Features, Pricing, Pros & Cons https://helpcrunch.com/blog/best-knowledge-base-software/ (дата звернення: 10.04.2020).

13. Di Cosmo R., Lienhardt M., Treinen R., Zacchiroli S., Zwolakowski J., Eiche A., and Agahi A. Automated synthesis and deployment of cloud applications. Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM, 2014, P. 211222.

14. Hirmer P., Breitenbucher U., Binz T., Leymann F. et al. Automatic topology completion of tosca-based cloud applications. GI- Jahrestagung, 2014. P. 247-258.

15. Nowak A,, Binz T., Breitenbücher U., Haupt F., Kopp O., Leymann F., Wagner S. Fhttps://www.researchgate.net/publication/258432787_OpenOpenTOSCA_-_a_runtime_for_OpenTOSCA-based_cloud_applications. (дата звернення: 14.05.2020).

16. Solve more with Google Cloud https://cloud.google.com/ (дата звернення: 14.05.2020).

17. Ubuntu 16.04.7 LTS (Xenial Xerus) https://releases.ubuntu.com/16.04/ (дата звернення: 14.05.2020).

18. Giannakopoulos I, Papailiou N., Mantas C., Konstantinou I., Tsoumakos D., and Koziris N., Celar: automated application elasticity platform. Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014, P. 23-25.

19. Topology and Orchestration Specification for Cloud Applications Version 1.0 OASIS Standard 25 November. 2013 OASIS, http://docs.oasis-open.org/tosca/OpenTOSCA/v1.0/OpenTOSCA-v1.0. pdf, (дата звернення: 09.06.2020).

20. Що таке Ansible. Приклад простого playbook-а. https://blog.goodhoster.net/uk/chto-takoe-ansible-primer-prostogo-playbook-a / (дата звернення: 09.06.2020).

References

1. Cloud Foundry. Access mode: https://www.cloudfoundry.org/community. (Date of request: 07.03.2020).

2. Multi-Cloud Orchestration. Access mode: http://getcloudify.org/.(Date of request: 10.04.2020).

3. Carrasco J., Cubo J., Duran F. and Pimentel E. Bidimensional crosscloud management with tosca and brooklyn, in Cloud Computing (CLOUD), 2016 IEEE 9th International Conference. IEEE, 2016. P. 951 – 956/

4. OpenTOSCA goes Docker Compose. Access mode: https://www.opentosca.org/ (Date of request: 10.04.2020).

5. Domain-Specific Modeling Language. Access mode: https://www.sciencedirect.com/topics/computer-science/domain-specific-modeling-language (Date of request: 10.04.2020).

6. Lu H., Shtern M., Simmons B., Smit M., and Litoiu M., Pattern-based deployment service for next generation clouds, Services (SERVICES), 2013 IEEE Ninth World Congress on. IEEE, 2013, P. 464-471.

7. Cloud solutions for IT infrastructure. Access mode: https://pirit.biz/reshenija/oblachnye-

reshenija (Date of request я: 07.05.2020).

8. Ardagna D., Di Nitto E., Casale G., Petcu D., Mohagheghi P., Mosser S., Matthews P., Gericke A., Ballagny C., D'Andria F. et al., Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds, Proceedings of the 4th International Workshop on Modeling in Software Engineering. IEEE Press, 2012, P. 50-56.

9. Narain S., Levin G., Malik S., and Kaul V., Declarative infrastructure configuration synthesis and debugging, Journal of Network and Systems Management. 2008. Vol. 16, No. 3. P. 235-258,

10. Di Cosmo R., Eiche A., Mauro J., Zacchiroli S., Zavattaro G., and Zwolakowski J., Automatic deployment of services in the cloud with aeolus blender. Service-Oriented Computing. Springer, 2015, P. 397-411.

11. Deployment Automation. https://doc-vu.github.io/DeploymentAutomation. (Date of request: 25.06.2020).

12. 8 Best Knowledge Base Software in 2020: Features, Pricing, Pros & Cons. Access mode: https://helpcrunch.com/blog/best-knowledge-base-software/ (Date of request: 10.04.2020).

13. Di Cosmo R., Lienhardt M., Treinen R., Zacchiroli S., Zwolakowski J., Eiche A., and Agahi A. Automated synthesis and deployment of cloud applications. Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM, 2014, P. 211222.

14. Hirmer P., Breitenbucher U., Binz T., Leymann F. et al. Automatic topology completion of tosca-based cloud applications. GI- Jahrestagung, 2014. P. 247-258.

15. Nowak A,, Binz T., Breitenbücher U., Haupt F., Kopp O., Leymann F., Wagner S. F. Access mode: https://www.researchgate.net/publication/258432787_OpenOpenTOSCA_-_a_runtime_for_OpenTOSCA-based_cloud_applications. (Date of request: 14.05.2020).

16. Solve more with Google Cloud. Access mode: https://cloud.google.com/ (Date of request: 14.05.2020).

17. Ubuntu 16.04.7 LTS (Xenial Xerus). Access mode: https://releases.ubuntu.com/16.04/ (Date of request: 14.05.2020).

18. Giannakopoulos I, Papailiou N., Mantas C., Konstantinou I., Tsoumakos D., and Koziris N., Celar: automated application elasticity platform. Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014, P. 23-25.

19. Topology and Orchestration Specification for Cloud Applications Version 1.0 OASIS Standard 25 November. 2013 OASIS. Access mode: http://docs.oasis-open.org/tosca/OpenTOSCA/v1.0/OpenTOSCA-v1.0. pdf, (Date of request: 09.06.2020).

20. What is Ansible. An example of a simple playbook. Access mode: https://blog.goodhoster.net/uk/chto-takoe-ansible-primer-prostogo-playbook-a / (Date of request: 09.06.2020).