УДК 004.046: 004.896: 004.942

ПОРІВНЯННЯ ТИПІВ АРХІТЕКТУРИ СИСТЕМ СЕРВІСІВ

О.О. ПЕТРЕНКО

Розглянуто сучасні архітектури систем сервісів SOA (service-oriented architecture — сервісно-орієнтована архітектура) й EDA (event-driven architecture - подійно-орієнтована архітектура), їх переваги і вади, можливість і доцільність побудови об'єднаної сервісно-орієнтованої архітектури EDSOA (event-driven service-oriented architecture — подійно-керована сервісно-орієнтована архітектура). Показано, що події з'єднують сервіси за допомогою передачі стану бізнес-процесу від одного сервісу, який визначає і публікує події, до інших сервісів, які запускаються конкретними подіями. В свою чергу, обґрунтовано, що сервіси об'єднують події за допомогою передачі даних про перехід одного стану процесу в інший. Особливу увагу в роботі приділено питанням ефективної реалізації запропонованого гібридного рішення архітектури EDSOA до моделювання бізнес-процесів як сервісів.

ВСТУП

Нещодавно з ініціативи фірми IBM було сформовано науку про сервіси, менеджмент і SSME (service science management and engineering — наука про сервіси, менеджмент і інжиніринг), що покликана дослідити основні принципи функціонування складних систем сервісів, шляхи створення, масштабування і вдосконалення таких систем. Варто особливо підкреслити, що розвиток науки про сервіси спирається на два таких відомих технічних нововведення в інформаційних технологіях, як технологія SaaS, коли програмне забезпечення використовується і орендується через інтернет як хмарний сервіс, і SOA (service-oriented architecture — сервісно-орієнтована архітектура) як базовий стиль архітектури у процесі проектування складних розподілених систем.

Масштаб, складність і взаємозалежність сучасних систем сервісів у зв'язку з глобалізацією, демографічними змінами та технологічними розробками досягли безпрецедентного рівня. У найбільш розвинених країнах більше 70% ВВП формується індустрією сервісів, у якій на сьогодні зайнято (за інформацією Міжнародної організація праці) більше половини людства [1, 2]. Сьогоднішня глобальна економіка вимагає від компаній розширення своїх бізнес-процесів за межі організацій та інтеграції бізнесу з партнерами. Необхідність інтеграції та взаємодії додатків у рамках сукупності великої кількості інформаційних систем самого підприємства або кількох підприємств, об'єднаних у цілий партнерський ланцюжок, також справляють істотний вплив на доцільний вибір архітектури систем сервісів.

Поняття «сервіс» і «процес» є взаємозалежними і їх можна використовувати на різних рівнях узагальнення. Наприклад, невеликий процес може бути організований у вигляді окремого сервісу у тому випадку, якщо його можна *типізувати*. У той же час процес може бути розбито на окремі сервіси, що взаємодіють між собою в рамках процесу. Якщо сервісів буде дуже

багато, то не буде досягнуто необхідної типізації, але якщо сервіси будуть занадто високорівневими, то їх застосування буде ускладнено через специфічні відмінності у бізнес-процесах. Відображаючи логіку бізнесу у формі веб-сервісів, можна формувати потоки завдань (workflows), спеціально налаштованих з урахуванням потреб підприємства.

Термін «сервіс» має два основних загально прийнятих значення: економічне, бізнес-орієнтоване значення, наприклад, як у виразі сервісний сектор економіки, IT- орієнтоване значення, веб-сервіс або SOA. У першому випадку акцент робиться на взаємодію у процесі обміну і на нематеріальний характер сервісу, а у другому — на технології програмного забезпечення, яке дозволяє підтримувати сумісність різних програмних модулів або агентів. У сучасних систем сервісів для економіки два значення зливаються, оскільки впровадження сучасної системи сервісів передбачає також інтеграцію інформаційних систем як підсистем організаційної системи систем. У той же час зростає сервісний домен з орієнтацією на сучасні виробничі процеси, наприклад, через «сервітизацію» виробництва [3]. Ці сервісно-орієнтовані процеси також сприяють інтеграції виробничих процесів з інформаційними системами і конкретними IT-технологіями.

Більшість сучасних знань було розроблено в період виробничої економіки, зокрема, в галузі систем управління технічними об’єктами за допомогою IT-технологій. Індустрія послуг потребує створення своєї наукової бази, розроблення методики й інструментарію для розроблення систем сервісів, розгортання підготовки відповідних кадрів, спроможних забезпечити подальше поширення і зміцнення індустрії послуг.

SOA

SOA замість монолітної пропонує блокову систему із взаємодіючих компонентів, у якій різні функціональні модулі додатків (сервіси), призначено для *управління бізнес-процесами*, що пов’язані між собою за допомогою чітко визначених інтерфейсів. Інтерфейси самі по собі незалежні від оточення і платформи, і тому така модель отримала назву моделі «*слабкого зв’язку*». Фактично суть концепції SOA полягає в уніфікації та автоматизації бізнес-процесів за допомогою типових компонентів — сервісів (наприклад, веб-сервіси використовують один стандарт — розширювану мову розмітки XML). При цьому створення, впровадження або зміна бізнес-процесу є компоновкою (оркестровкою) раніше розроблених сервісів, призначених для автоматизації бізнес-функцій.

Для бізнесу SOA означає збільшення задоволеності клієнтів, реальну гнучкість бізнесу, швидкий час виходу на ринок, простоту співробітництва і низьку вартість бізнесу. Для IT-організацій SOA означає більшу продуктивність, зниження витрат на IT за рахунок прискорення розробки додатків, більш м’якого повторного використання сервісів, більш високої якості додатків, і в цілому більш швидкого реагування на запити бізнес-клієнтів для поліпшення і модифікації системи. На додаток до цього існує можливість використання сервісів незалежних постачальників, що забезпечує ще більшу цінність SOA. Слово «гнучкість» часто згадується у ході обговорення переваг SOA і може бути інтерпретовано, з одного боку, як можливість

змінювати бізнес-процеси відповідно до змін вимог ринку і вимог клієнтів і, з другого, як здатність виконувати бізнес-процеси швидше або запускати швидше нові процеси, продукти і сервіси. Гнучкість і швидкість — реальні й відчутні переваги переходу на SOA і багаторазове використання сервісів.

SOA є архітектурним підходом, який дозволяє розкласти функціональність додатків на множину сервісів, розміщених у мережі. Веб-сервіси є технологію реалізації концепції дизайну SOA. Існує велика кількість досліджень, присвячених вимогам надійності в SOA і, більш конкретно, веб-сервісів. Веб-спільнота розробила ряд специфікацій, які підтримують надійний обмін повідомленнями, управлінням транзакціями, реплікаціями і безпекою. Сервіси взаємодіють один з одним за допомогою шаблону синхронного запиту і відповіді, що забезпечує щільне зчеплення між компонентами системи (рис. 1).

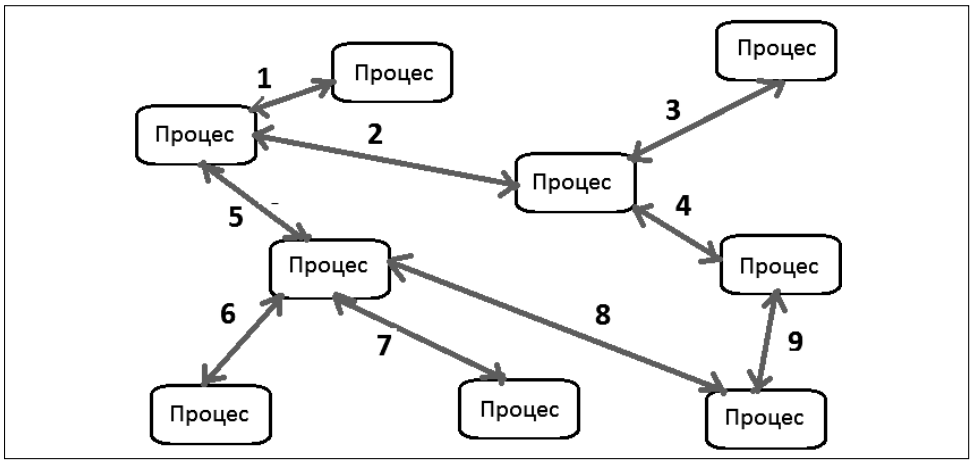

Рис. 1. Архітектура SOA, керована запитами, де 1– 9 — послідовність запитів

До базових принципів SOA треба віднести [4]:

- ***Слабке зв'язування сервісів*** — одна з найбільших переваг SOA. Слабко зв'язані модулі мають кілька добре відомих залежностей, а тісно пов'язані модулі мають багато невідомих залежностей. У суперечність зі слабким зв'язком між сервісами і можливістю їх повторного використання застосування сервісу може бути ґрунтовним і щільним.
- ***Стандартизація*** — стандарти SOA відкриті в тому сенсі, що будь-який виробник програмного забезпечення має право використовувати ці стандарти у процесі розробки архітектури SOA. Крім того, процес створення та перегляду стандартів є більш-менш демократичним, де будь-яка зацікавлена сторона має право брати участь у всіх засіданнях, які призводять до рішень про стандарт.
- ***Модульність*** — SOA реалізує сервіси, що підтримують чітко визначені модульні бізнес-функцій та інформаційні процеси. Ці модулі можуть надалі бути використані повторно і бути частиною бізнес-процесу. Модульність сервісів призводять до поділення додатків на безліч дрібних модулів. Кожен модуль відповідає за одну окрему функцію у додатку.
- ***Скомпонованість*** — здатність ефективно складати сервіси є найважливішою вимогою для досягнення деяких із найбільш фундаментальних ці-

лей сервіс-орієнтованих застосувань. Сервіси, які складаються, здатні брати участь у якості ефективних модулів. Модульна сервісна структура дозволяє створювати нові інформаційні системи, про які розробники сервісів можуть не мати жодного уявлення під час проектування сервісів.

- ***Повторюваність*** — повторне використання сервісів як основна частина аналізу обслуговування та процесів проектування систем сервісів. Згідно з цим принципом сервіс може бути використаний більш ніж за одним сценарієм у різних бізнес-процесах або інформаційних системах.
- ***Відкритість*** — сервіси мають бути легко ідентифіковані й зрозумілі, щоб забезпечити можливість їх повторного використання. Тому у процесі проектування сервісів необхідно враховувати «якість доступу» до сервісів та їх індивідуальні властивості незалежно від того, чи вони зареєстровані в депозитарії, чи ні.
- ***Абстракція*** — цей принцип підкреслює необхідність приховувати так багато з основних деталей сервісів, як це можливо. Це забезпечує безпосередньо описане раніше слабке зв'язування сервісів.
- ***Зв'язок «один-на-один»*** — один із конкретних сервісів викликається у часі тільки одним із споживачів, але зв'язок є двонаправленим.
- ***Синхронність*** — відповіді на запити відправляються назад до споживача в синхронному режимі.
- ***Запуск процесу*** — потік управління ініціюється клієнтом (споживачем послуг).
- ***Зернистість сервісів —*** рівень деталізації обслуговування. Сервіси в SOA є модулями бізнес-логіки досить високого рівня, завдяки чому взаємодія між ними зводиться до обмеженого числа повідомлень за змістом бізнес-логіки замість безлічі низькорівневих викликів, що враховують деталі реалізації сервісів. Такий підхід знижує навантаження на мережу і сприяє більш високій продуктивності системи.
- ***Відсутність стану*** — сервіси проектуються так, щоб залишитися зі станом тільки за необхідності. Сервісам не варто покладатися на тривалий зв'язок між споживачем і постачальником, вони не мають також покладатися на попередні виклики.

Орієнтація на сервіси та SOA може краще використовувати тоді, коли процеси або їх частини є стандартними, коли вони часто повторюються без змін, або коли декільком користувачам потрібно той же компонент процесу для виконання своїх завдань. Виклик (споживання) сервісів у SOA реалізується віддалено за допомогою віддаленого виклику процедури RPC (remote procedure call — віддалений виклик процедур) на вимогу споживача сервісів. SOA добре зарекомендувала себе для побудови великих корпоративних програмних додатків. Ціла низка розробників та інтеграторів пропонують інструменти і рішення на основі SOA (наприклад, платформи Intel SOA Expressway, JBoss SOA Platform, IBM WebSphere, Software AG webMethods, Oracle , BEA Aqualogic, Microsoft Windows Communication Foundation, SAP NetWeaver, TIBCO).

Однак практичне застосування всього потенціалу SOA ускладнюється через часту необхідність використовувати окремі сервіси з несумісними інтерфейсами. Крім того, у бажанні створення підприємств реального часу, які

постійно підключені й завжди доступні в інтернеті, організації стикаються з більш різноманітними бізнес-сценаріями і виявляють потреби в альтернативних шаблонах проектування на доповнення до синхронної керованої запитом SOA.

EDA

EDA (event-driven architecture — подійно-орієнтована архітектура) містить в собі три базові компоненти: генератор події (датчик), оброблювач події і менеджер подій (відповідач). Менеджер подій реєструє всі події, які виникають в системі (зовнішні і внутрішні), відповідним чином ідентифікує і передає оброблювачу подій. Якщо потрібний оброблювач за яких-небудь причин не доступний, подія в залежності від конкретної реалізації або ставиться в чергу, або передається іншому оброблювачу. Завдяки такій схемі система стає максимально гнучкою і чутливою до змін інформаційного середовища — у разі виникнення принципово нової події досить підключити новий оброблювач, не зачіпаючи ядра системи. Подібні системи зазвичай будуються на тригерах, що реагують на певні події та ініціюють відповідні веб-сервіси, рис. 2 [5].

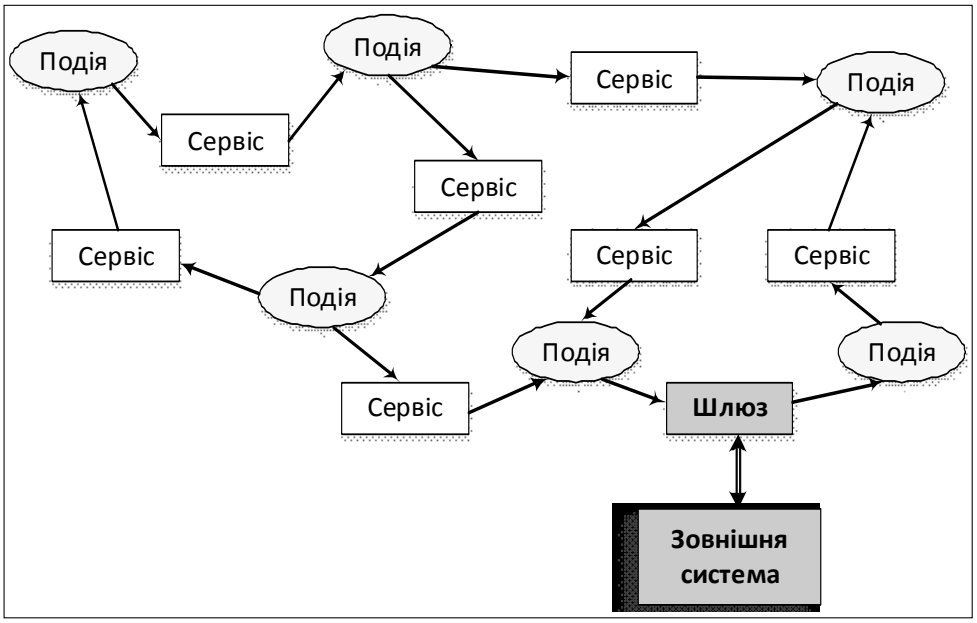

Рис. 2. EDA архітектура з подіями

Обидві нові архітектури SOA й EDA відрізняються від традиційної архітектури тим, що бізнес-процес розбивається на малі й повторно використовувані процедури. Ці процедури формалізуються і реалізуються у вигляді IT-компонентів, які слабо пов'язані і можуть бути інтегровані в динамічний і гнучкий засіб. SOA і EDA намагаються замінити і розбити старі і великі успадковані системи й архітектури на більш гнучкі і багаторазово використовувані бізнес- та IT-компоненти. Співвідношення між SOA й EDA є актуальною темою для обговорення. Споживачі і дослідники дотримуються протилежних думок з приводу того, як сервіси та події мають взаємодіяти на

різних рівнях, включаючи бізнес, інформацію та інформаційні системи. До базових принципів EDA треба віднести:

- ***Роз'єднаність*** — джерело події знає тільки створену подію і не має жодного уявлення щодо її подальшого оброблення заходу або про зацікавлені сторони. Цей принцип використовується у ході інтеграції роз'єднаних додатків і бізнес-процесів. Процес складається з декількох етапів. У EDA етапи не залежать фізично або логічно один від одного, так що кожен може бути змінено, не викликаючи побічних ефектів, на інші, поки повідомлення щодо подій не змінюються.
- ***Повторюваність*** — вирішальним фактором у проектуванні подій є те, що вони мають бути спроектованими таким чином, щоб відповідати не тільки нинішнім вимогам, а й майбутнім потребам і поза сферою використання. Події мають бути розроблені в загальному вигляді, який забезпечує їх повторне використання.
- ***Повідомлення у режимі реального часу*** — технічна інфраструктура має підтримувати створення в режимі реального часу подій і їх доставку. Персонал інфраструктури забезпечує в режимі реального часу перетворення інформації в знання, а потім в інтелектуальні наступні дії.
- ***Свобода діяти —*** повідомлення не вказує, які дії виконуватиме споживач події. Це звіт, а не вимога. Споживачу притаманна логіка, яка визначає, як він буде реагувати. Величезну кількість подій може бути вироблено, але тільки деякі з них будуть споживаними.
- ***Зв'язок «один-до-багатьох»*** — на одну конкретну подію може відгукнутися багато передплатників.
- ***Асинхронність*** — підтримка асинхронних операцій через повідомлення про події.
- ***Запуск процесу*** — потік управління, який визначається одержувачем, базується на розміщених подіях.
- ***Відсутність стану компонентів обробки подій*** — компоненти оброблення подій не мають стану, але сама подія буде мати стан, визначений у ній.

Для оброблення подій та ініціалізації веб-сервісів можуть бути використані три стратегії: *проста*, *потокова* і *комплексна*. Просте оброблення полягає в ініціалізації веб-сервісів по мірі реєстрації відповідних подій. Основна перевага такої системи — це робота в режимі реального часу. Мінус очевидний — не враховується фактор пріоритету. Потокове оброблення враховує існуючі залежності веб-сервісів і об'єднує кілька сервісів в один загальний потік. Цей підхід виправдовує себе в системах з великими накладними витратами на пошук і отримання інформації з бази даних. І, нарешті, комплексне оброблення — це так зване управління за відхиленнями. Будь-яка подія розцінюється як вихід системи зі стану рівноваги. Ініціалізація веб-сервісів переслідує єдину мету — повернути систему в стан рівноваги (між справою задовольняючи потреби клієнтів системи).

Новітній хмарний механізм оброблення подій CEP (complex event processing — комплексне оброблення подій) знаходиться в експлуатації з 2007 року [6]. У ньому реалізовано такі методи, як виявлення складних візерунків багатьох подій, кореляції подій і абстракції, ієрархії подій і виявлення від-

носин між ними, таких як причинність, підпорядкованість, синхронізація зв'язку з процесами, пов'язаними з подіями. У той час як SOA і оркестрування автоматизують процеси, CEP автоматизує інтелектуальну кореляцію і співвідношення між рішеннями, прийнятими персоналом. Система CEP дозволяє користувачам оброблювати випадкові, не пов'язані події, визначати тенденції і передбачати результати. Заходи можуть бути вжито для запобігання негативного наслідку від виникнення події. Шаблони реакцій також можуть бути проаналізовано, щоб поліпшити основні процеси та інформаційні системи з метою запобігання майбутніх помилок.

Бізнес-аналітики та керівники середньої ланки добре розуміють процеси, керовані подіями, оскільки вони пов'язані з стандартними бізнес-процедурами, такими як відносини замовника і постачальника, операційні процедури, процеси і потоки завдань. Переваги ведення бізнесу в режимі реального часу з EDA зрозуміло:

- Рішення приймаються, а потреби користувачів відразу задовольняються.
- Бізнес-процеси адаптується швидко, щоб задовольнити попит або вирішити проблему.
- CRM (customer relationship management — управління відносинами з клієнтами), підтримка, маркетинг і навіть ланцюжки постачань, можуть бути миттєво змінено.
- Весь бізнес можна розглядати в режимі реального часу для підвищення ефективності підприємства.

ПОРІВНЯННЯ АРХІТЕКТУР EDA Й SOA

EDA й SOA можуть бути реалізовані і функціонувати паралельно без будь-яких суперечностей. Існує три основні передумови для цієї природньої співпраці. Перша — це спільні та визначені бізнес-цілі, друга — використання роз'єднаних компонентів і спільної моделі даних, третя — використання спільної інфраструктури і технологій. EDA й SOA є взаємодоповнюючими у багатьох аспектах. Додавши EDA на вершину SOA архітектури, можна отримати нові можливості. Обидві архітектури SOA й EDA є досить новими концепціями і знаходяться на ранніх стадіях становлення. Багато компаній сьогодні здійснюють впровадження SOA та реалізацію очікуваних користі й цінностей [7]. EDA протягом багатьох років використовувалася як технічний шаблон проектування, який забезпечує масштабованість і високу продуктивність транзакцій. EDA як бізнес-концепція та архітектурний стиль все ще є новою і не підтримується прийнятими платформами і методами проектування. Ще однією серйозною проблемою є управління подіями.

Проведення порівняння EDA й SOA можна сьогодні лише на основі літературних джерел, оскільки бракує інформації щодо практичного досвіду організацій, де EDA експлуатуються. Обидві SOA й EDA націлені на узгодження бізнесу та IT-технології, але за допомогою різних засобів. У той час, як SOA використовує бізнес-сервіси, EDA використовує бізнес-події для подолання розриву між бізнесом та IT. Результати порівняння зведено в таблиці, де виділено три основні напрямки: відображення бізнес-процесів

і бізнес-функцій, використання даних та інфраструктури, здатність до інтеграції.

Таблиця. Порівняльний аналіз EDA і SOA

SOA	EDA
Відображення бізнес-процесів і бізнес-функцій	
Є архітектурним стилем, який визнає сервіси (функціональність), що відображають етапи бізнес-процесу і які створюють більш високий рівень гнучкості бізнесу та IT-середовища	Є архітектурним стилем, що визнає події (повідомлення), що відображають зміни стану процесу, на які бізнес реагує згідно з планом і які забезпечують більш високий рівень гнучкості бізнесу та IT-середовища
Фокусування на декомпозицію бізнес-процесу на сервіси, які слабко пов'язані між собою	Фокусування на виявленні бізнес-подій, які визначаються змінами в стані бізнес-процесу підприємства
Заохочення до повторного використання сервісів (реактивність)	Заохочення до обміну інформацією та використання бізнес-аналітики (активність)
Використання даних	
Дані запитується тими, хто зацікавлений у цій інформації	Дані поширюється в режимі реального часу серед тих, хто зацікавлений у цій інформації і підписався на неї
Сприяє повторному використанню і спільному вживанню даних у різних додатках і для декількох каналів доступу кінцевих користувачів	Сприяє миттєвій ідентифікації та реагуванню на події / інформацію, що управляють бізнес-процесом
Дані як сервіс є архітектурним підходом, що послаблює тісні зв'язки між даними додатками, так що дані можуть контролюватися і поділятися по всьому підприємству	Надмірність даних необхідна для збільшення рівня декомпозиції бізнес-процесу
Інфраструктура і здатність до інтеграції	
Застосовується поетапне розроблення і підтримка великих розподілених додатків	Застосовується поетапне розроблення і підтримка великих розподілених додатків
Застосовується шаблон «Запит / Відповідь»	Застосовується шаблон «Публікація / Підписка»
Сервіси викликаються незалежно від технології їх виготовлення та місця знаходження	Події формують повідомлення, що відправляються незалежним програмним модулям, які абсолютно нічого не знають один про одного
Одночасно лише один із конкретних сервісів викликається одним споживачем (зв'язок «один-до-одного»)	На одну конкретну подію може відгукнутися багато передплатників (зв'язок «один-до-багатьох»)
Сприяє інтеграції шляхом впровадження стандартів і ведення кроків обгортки, нашарування і композиції	Сприяє інтеграції, оскільки на повідомлення про події можуть підписатися всі зацікавлені й потім обирати необхідні з них
Виведення може використовувати синхронні та асинхронні моделі	Використовуються лише асинхронні моделі

Таким чином обидві архітектури SOA й EDA забезпечують перехід організації зі старою архітектурою бізнес-процесів, заснованою на підтримці незалежними монолітними прикладними додатками, до нового типу архітек-

тури, яка базується на незалежних сервісах і подіях, яка є більш динамічною і допускає багаторазове використання компонентів. Обидві парадигми ведуть до довгострокових інвестицій у різних галузях, де вони можуть принести користь, наприклад, шляхом:

- доповнення існуючих інформаційних систем і представлення їх як сервісів, тим самим збільшити рентабельність і продовжити термін використання спадщини, а також знизити витрати на розробку нової функціональності;
- використання існуючих сервісів і подій для підтримки нових бізнес-процесів, які можуть бути розгорнуті на ринку більш швидко і з більш низькою вартістю;
- повторного використання інваріантних міжгалузевих сервісів і подій для декількох бізнес-процесів, підрозділів або підприємств, що може привести до збільшення добутку, тобто кращого повернення від інвестиції.

Подійно-керований шаблон EDA сприяє декомпозиції пов'язаної системи (або бізнес-процесу) у порівнянні з вимогою слабко-пов'язаної системи для SOA. Тобто функціональні процеси в системі відправника, де відбулася бізнес-подія, не залежать від наявності та завершення віддалених процесів в розподілених низькорівневих системах. У той час, як в архітектурі, керованій запитом, система відправника повинна точно знати, які розподілені сервіси відгукуються на виклик, тобто залежить від наявності та завершення цих віддалених сервісів.

Архітектура EDA, керована подіями, має значення тільки тоді, коли події опубліковані, споживається і поширюється ***в режимі реального часу***. Якщо цей підхід застосовується в пасивних сценаріях оброблення подій, наприклад, орієнтованих на фіксовану графіку, то його ефективність не дуже відрізняється від традиційних методів інтеграції даних, що використовуються сьогодні.

EDA притаманна тісна інтеграція з бізнес-процесами, у той час як для SOA завжди слабким місцем були точки входу кінцевих користувачів в силу їх неоднорідності за безліччю параметрів. На думку деяких експертів, як тільки SOA стає тісно зав'язана на бізнес-процесах організації, вона перероджується в подійно-керовану архітектуру і набуває не властиву статичним системам гнучкість й адаптивність. Головна складність у ході розробки подієво-керованих додатків — це гарантування, що всі оброблювачі подій будуть завершуватися досить швидко, щоб не блокувалися системні виклики. Тобто не для всіх додатків ця архітектура підходить, але вона успішно може працювати, якщо в якості оброблювачів подій використовуються веб-сервіси.

ОБ'ЄДНАНА АРХІТЕКТУРА СИСТЕМИ СЕРВІСІВ

З порівняльного розгляду SOA й EDA очевидний тісний зв'язок між двома їх основними компонентами (подіями та сервісами). Дві основні відносини між EDA й SOA теж очевидні. У першому відношенні через події підключаються сервіси за допомогою передачі стану процесу і даних від одного сервісу, який виявляє і публікує події, до інших сервісів, які запускаються

у процесі появи конкретних подій. У другому відношенні сервіси генерують події, переводячи бізнес-процес з одного стану в інший. Іншими словами, ***подія фіксує стан, а сервіс змінює стан***. Взаємодія між SOA й EDA при цьому здебільшого відбувається на двох різних рівнях (рис. 3).

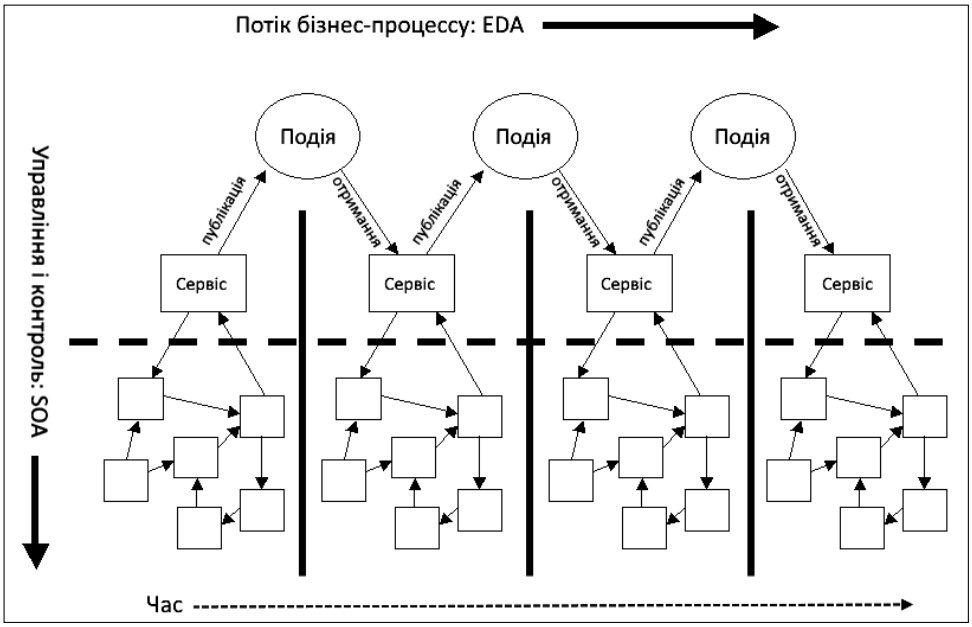

Рис. 3. Взаємодія SOA й EDA

- На першому рівні події з'єднують сервіси за допомогою передачі стану процесу від одного сервісу, який визначає і публікує події, до інших сервісів, які запускаються конкретними подіями.
- На другому рівні сервіси об'єднують події за допомогою передачі даних про перехід одного стану процесу в інший (споживач отримує подію і вживає необхідні заходи, які можуть призвести до виклику нових сервісів і формування нових подій).

Важливо також відзначити, що взаємодія між SOA й EDA також може відбуватися на рівні інформаційної системи, а не тільки на рівні бізнес-процесів. В об'єднаних SOA й EDA сервіси більше не є тільки споживачами подій. Зовнішні / внутрішні інформаційні системи можуть також вживатися і виробляти події.

Об'єднання концепцій SOA й EDA, пропонується називати EDSOA (event-driven service-oriented architecture — подійно-керована сервісно-орієнтована архітектура). Тільки формування бізнес-сервісів і бізнес-подій, а потім їх цільова ув'язка для рішень завдань бізнесу дозволяє домогтися стратегічних переваг для підприємства. При цьому EDA використовується для того, щоб «накинути» на всю програмну інфраструктуру організації «мережу» програмних датчиків і програмних агентів, а також апаратних датчиків, які стежать за подіями у всіх апаратних і програмних компонентах, відстежують значущі для бізнесу події і передають їх у центр прийняття рішень із сигналами, що асоційовані з цими подіями. Це дозволяє технологічно управляти бізнесом не наосліп, а маючи чітку картину всього, що відбувається в конкретний момент на підприємстві.

Як було зазначено раніше, важливо аналізуючи відносини між EDA й SOA враховувати рівень деталізації (зернистості) полій і сервісів. У добре визначеній архітектурі рівень деталізації бізнес-подій збігається з зернистістю бізнес-сервісів. Хоча рівень деталізації обох сервісів та подій має велике значення, поки не зовсім зрозуміло, як обрано потрібний рівень деталізації на етапі визначення сервісів і подій. І навіть менш очевидно поки, як переконатися, що сервіси і події вибрані на одному рівні.

Якщо говорити про SOA й EDA конвергенцію, можна вважати найбільш зручним рішенням цієї задачі використання сервісної шини підприємства ESB (enterprise service bus — сервісна шина підприємства), яка діє в якості проміжного шару (або посередника) для забезпечення комунікації та інтеграції великомасштабних гетерогенних прикладних процесів. ESB підтримує всі можливості: як SOA, так й EDA — оскільки підтримує синхронні, а також асинхронні взаємодії між однією або багатьох зацікавлених сторін. Це не стандарт, ні специфікація, а, швидше, велика кількість відкритих та комерційних ESB, які розроблялися багатьма постачальниками і налаштовані відповідно до їх потреб. До найбільш відомих реалізацій ESB можна віднести Open ESB [8], Apache ServiceMix [9], JBoss ESB [10], IBM WebSphere ESB [11], Celtix IONA й ObjectWeb [12]. Варто також звернути увагу на вільно поширювану шину *Mule* [13].

Як показано на рис. 4, ESB містить бізнес-процеси, сервіси та події, які споживаються або виробляються. Використання ESB в якості проміжного ПЗ забезпечує наступні групи сервісів [14].

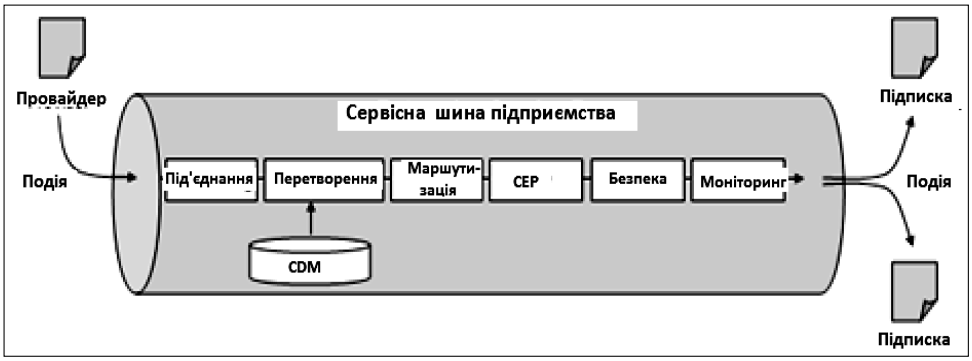

Рис. 4. Среднемесячные числа Вольфа

- ***Транспортування***: транспортний сервіс гарантує доставку повідомлень та їх обмін між різними прикладними процесами. В ESB можна застосувати динамічну маршрутизацію (також на основі змісту) і відправку запитів за декількома напрямками. Це дозволяє ESB балансувати навантаження або механізми відмовостійкості.
- ***Оброблення подій***: сервіс подій дозволяє ESB обробляти події, можливо, аналізувати та контролювати складні серії взаємопов'язаних подій. Для цього, більшість продуктів ESB забезпечує реалізацію специфікацій, присвячених організації подій, таких як WS-Notification й WS-Eventing [15].
- ***Оркестрування***: цей сервіс засновано на використанні BPEL технологій [15] і дозволяє ESB організовувати виконання ряду взаємозв'язаних сервісів. Під час роботи з подіями на кожному етапі бізнес-процесу сервіси вищого рівня організують роботу сервісів нижчого рівня.

- ***Пошуку***: цей сервіс, включений до ESB, сприяє тому, що прикладні процеси виявляють відповідні сервіси, з якими вони взаємодіють [16].
- ***Посередництва***: цей сервіс посередництва є фундаментальним для напрямку бізнес-інтеграції. Він відповідає за: перетворення одного протоколу зв'язку на інший для того, щоб зробити можливим спілкування між гетерогенними середовищами; перетворення змісту будь-якого повідомлення, а також збагачення повідомлення додатковою інформацією для того, щоб будь-які дані, що передаються, були зрозумілими для будь-якого прикладного додатку. Крім перетворень, сервіс є відповідальним за безпеку, яка має вирішальне значення в міжорганізаційних взаємодіях, за вимоги якості QoS (наприклад, вимірювань, кешування, виявлення відмови і наступне відновлення).

Основні обмеження ESB пов'язані з їх застосуванням тільки в суворо контрольованих умовах, де адміністратори можуть правильно конфігурувати, розгортати і, нарешті, тонко налаштовувати проміжне ПЗ, щоб отримати максимальну продуктивність і максимальний рівень надійності. Як тільки взаємодії перетинають кордони контрольованого середовища (яке, можливо, охоплює кілька незалежних адміністративних доменів), можливості цих платформ можуть непередбачувано деградувати — і загальний рівень обслуговування не буде більше легко керованим.

ESB є посередником, який пов'язує сервіси разом. ESB може бути реалізована різними способами: за допомогою класичних повідомлень; EAI (enterprise application integration — інтеграція корпоративних додатків); брокерських технологій; шляхом використання конкретної платформи компонентів, наприклад, інтеграції шин у сервері додатків J2EE (java 2 enterprise edition — обчислювальна корпоративна платформа Java). ESB також може бути поєднанням обох EAI й технології серверів додатків, але реалізація не має впливати на загальну архітектуру.

ESB виступає як інтелектуальний шар для підключення інформаційних систем, різних даних та інших сервісів, які зазвичай розподілено по всьому IT-середовищу підприємства. Вона поєднує в собі можливості синхронного та асинхронного обміну повідомленнями з можливостями інтелектуальних перетворень і маршрутизації. Це гарантує, що повідомлення передаються надійно. ESB дозволяє інтелектуальне оброблення запитів на обслуговування і відповідей, але також подій і повідомлень.

ESB не є новим продуктом, а новою концепцію інтеграції інформаційних систем, координації ресурсів та управління інформацією. На відміну від багатьох попередніх підходів до підключення інформаційних систем, ESB забезпечує підключення програмного забезпечення, яке працює на різних платформах, написаних на різних мовах програмування, і використовує різні технології. ESB є архітектурним шаблоном, який полегшує і спрощує бізнес-інтеграцію через посередницькі й транспортні сервіси. Вона виступає посередником для всіх комунікацій та взаємодій між різнорідними вузлами, як у сервіс-орієнтованої архітектурі (синхронний підхід «один-до-одного»), так у подійно-керованій архітектурі (асинхронний підхід «багато-до-багатьох»). На сьогодні ESB є найбільш ефективним засобом вирішення складних проблем інтеграції та є технічним рішенням, яке забезпечує найбільшу гнучкість бізнесу й ефективне з'єднання між різнорідними додатками.

Варто визнати, що EDSOA дозволяє домогтися поставлених цілей тільки в тому випадку, якщо:

- в інформаційній системі компанії буде структуровано й оформлено програмні компоненти, що реалізують сервіси, важливі для бізнесу (SOA складова);
- інформаційну систему буде насичено програмними датчиками, що формують значущі для бізнесу події (EDA складова);
- оперативне прийняття управлінських рішень буде автоматизовано за схемою, у якій реакція на бізнес-подію викликає старт бізнес-процесу, який ініціює запит на бізнес-сервіси (концепція EDSOA).

На завершення варто навести слова відомого в світі фахівця Юхима Натиса з компанії Gartner Research, який радить «розглядати SOA як довгострокову програмну архітектуру та інженерну практику, що вимагає наявності відповідних кваліфікованих кадрів, але не як спосіб вирішення поточних проблем департаментів інформаційних технологій. Великі вигоди вимагають великих інвестицій. Ухвалення бізнес-семантики для сервісів і додавання інтегрованої підтримки бізнес-подій вимагає великих систематичних зусиль, і ці кроки необхідні для реалізації потенційних можливостей архітектури бізнес-компонентів у всій її повноті» .

ВИСНОВКИ

SOA й EDA є новими поняттями для багатьох організацій і вимагають зміни менталітету і підходів як всередині бізнесу, так і в IT-середовищі. Наприклад, австралійська компанія Springboard Research, опитавши майже 3 тис. IT-керівників в Австралії, Індії, Китаї та Сінгапурі (аж ніяк не найвідсталіший регіон), встановила, що тільки 21% з них розуміють суть концепції SOA [17]. За традицією IT-додатки і процеси орієнтовано на використання передбачуваних і повторюваних подій. Коли ж відбуваються помітні події, які відрізняються винятковістю, то це створює складнощі для значної частини бізнесу. Це робить доцільним впровадження об'єднаної EDSOA архітектури як основної для побудови складних сучасних систем сервісів. Але недосвідченість розробників прикладних додатків у розробці систем, керованих подіями, і труднощі в декомпозиції бізнес-процесів на окремі компоненти стримують у цей час широке застосування парадигми EDSOA. Тим не менш, є також обнадійливі ознаки зацікавленості великих IT-виробників у новій технології (IBM, Microsoft, Google, Oracle тощо), які розпочали дослідження підходів з усунення вказаних труднощів [18–24].

Дуже важливо правильно виділити частину бізнесу організації, якій буде користь від EDSOA. Цей крок вимагає формування архітектури підприємства, де буде визначено сферу використання задіяних сервісів і подій. Бізнес-сервіси і події мають бути об'єднані таким чином, щоб забезпечити гнучку реконфігурацію процедур оброблення, можливо, шляхом використання декількох реакцій на подію. Усі ці заходи вимагають нової компетенції. Визначені сервіси, події й архітектура інтегруються в складний бізнес-процес.

Конвергенція SOA й EDA потребує розробки нових надійних інфраструктур проміжного ПЗ для реалізації складних сценаріїв прикладних до-

датків (наприклад, ланцюгів постачань, фінансових операцій). Ці підходи, будучи об'єднаними разом, можуть забезпечити використання їх взаємно додаткових характеристик. У цьому сенсі технології ESB можуть бути використані для реалізації конвергенції SOA й EDA. Однак відсутність стандарту й існування безлічі конкретних постачальників ESB реалізацій (чи комерційних, чи відкритих) ставить підприємства перед важким вибором ESB, яка найкраще підходить до конкретних конфігурацій розгортання бізнес-процесу. Це може негативно позначитися на гнучкості, яка забезпечується цими рішеннями і контентом вимог бізнесу.

Крім того, існуючі ESB є складними продуктами, які вимагають великих зусиль для конфігурації, розгортання і налаштування. Це пояснюється, очевидно, великими масштабами впровадження систем сервісів, неоднорідністю і сильною динамікою, яка характеризує намічені сценарії, де проміжне ПЗ має охоплювати кілька незалежно керованих адміністративних доменів. З цього погляду теперішні ESB представляють собою досить статичні рішення, позбавлені гнучкості, яка може бути забезпечена за допомогою автоматичної конфігурації, самооптимізації, самостійної адаптації та самозахисту функціональності. Таким чином, бажано, щоб у майбутньому було здійснено заходи з удосконалення ESBS з метою досягнення нових рівнів гнучкості в складних умовах застосування в індустрії сервісів.

ЛІТЕРАТУРА

1. *Maglio P*. Service systems, service scientists, SSME, and innovation // Communications of ACM. — 2006. — 49, issue 7. — P. 81–85.
2. *Петренко О.О.* Объекты и методы науки о сервисах // Системні дослідження та інформаційні технології. — 2015. — № 2. — С. 75–82.
3. *Succeeding* through service innovation: A service perspective for education, research, business and government. — http://www.ifm.eng.cam.ac.uk/uploads/ Resources/Reports/080428cambridge_ssme_whitepaper.pdf.
4. *Service* Oriented Architecture: SOA Features and Benefits. — https://www. opengroup.org/soa/source-book/soa/soa_features.htm.
5. *A model-driven* ontology approach for manufacturing system interoperability and knowledge sharing. — http://www.researchgate.net/profile/Keith_Case/ publication/257001871_A_model-driven_ontology_approach_for_manufacturing_system _interoperability_and_knowledge_sharing/links/00b49530dc9487310e000000.pdf.
6. *Esper:* Event Processing for Java. — http://www.espertech.com/products/esper.php.
7. *Industrial* SOA. — http://www.oracle.com/technetwork/articles/soa/ind-soa-toc-1934143.html.
8. *Enterprise* Integration EAI vs. SOA vs. ESB. — http://ggatz.com/images/Enterprise_ 20Integration_20-_20SOA_20vs_20EAI_20vs_20ESB.pdf.
9. *Apache* Software Foundation. — http://servicemix.apache.org/home.html.
10. *BossESB*. — http://www.jboss.org/community/docs/DOC-10326.
11. *ESB*: The SOA communication center. — http://www-01.ibm.com/software/ integration/wsesb/.
12. *Celtix:* The Open Source Java Enterprise Service Bus. — http://celtix.objectweb.org.
13. *Event-driven* services in SOA: Design an event-driven and service-oriented platform with Mule. — http://www.javaworld.com/article/2072262/soa/event-driven-services-in-soa.html.

14. *Combining* Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus. — http://www.ibm.com/developerworks/library/ws-soa-eda-esb/.
15. *OASIS* Web Services Notification. — https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn.
16. *OASIS* Web Services Business Process Execution Language. — https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.
17. *EDA* как очередная инкарнация SOA. — http://ecm-journal.ru/post/EDA-kak-ocherednaja-inkarnacija-SOA.aspx.
18. *Advancing* soa with an event-driven architecture. — http://www.intersystems.com/assets/Ensemble_SOA-EDA-6a940c1315b29eec545fa18db122dd47.pdf.
19. *New* to SOA and web services. — http://www.ibm.com/developerworks/ webservices/newto/.
20. *Mixing* Event Driven Computing, SOA, BPM and BI for Instant Responsiveness. — http://www.ebizq.net/blogs/firstlook/2007/10/post_17.php.
21. *Event-Driven* SOA: Events Meet Services. — http://www.oracle.com/technetwork/articles/soa/schmutz-soa-eda-405955.html.
22. *SOA* and BPM Are Better Together. — ftp://public.dhe.ibm.com/software/eg/soa/garbetter.pdf.
23. *Implementing* an Event-driven Service-oriented Architecture in TIP. — http://www.cs.waikato.ac.nz/pubs/wp/2010/uow-cs-wp-2010-02.pdf.
24. *Event-Driven* Architecture and SOA — Allies or Enemies? — http://wis.vsb.cz/ekf/php/tsw/getfile.php?prispevekid=935.

Надійшла 31.08.2015