

Beyond. New Riders, 2010.

[2] M. Kochan. User Stories Applied: For Agile Software Development. Addison-Wesley; Pearson Education, 2004.

[3] A. Cooper, R. Reiman, D. Cronin. About Face 3: The Essentials of Interaction Design. Wiley Publishing, 2007.

[4] В. М. Сеньківський, А. В. Кудряшова, Р. О. Козак. Інформаційна технологія формування якості редакційно-видавничого процесу: Монографія. Львів: Українська академія друкарства, 2019. 272 с.

[5] В. М. Сеньківський, І. В. Піх, А. В. Кудряшова. Теоретичні основи забезпечення якості видавничо-поліграфічних процесів (Частина 2. Синтез моделей пріоритетності дії факторів). Поліграфія і видавнича справа. Львів: УАД, 2016. № 1 (71). С. 20–29.

[6] М. С. Сявавко. Інформаційна система «Нечіткий експерт». Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. 320 с.

[7] В. М. Сеньківський, І. В. Піх. Математичне моделювання процесу ранжування факторів. Моделювання та інформаційні технології. Київ: ППМЕ НА-НУ, 2013. № 69. С. 142–146.

[8] В. Матвеев. Семантичні мережі. URL: matveev.kiev/expert/t5.pdf.

[9] В. М. Сеньківський, І. В. Піх, А. В. Кудряшова, О. В. Литовченко. Удосконалення моделі факторів впливу на композиційне оформлення видання. Поліграфія і видавнича справа. Львів: УАД, 2016. № 2 (72). С. 20–27.

[10] Саати Т. Принятие решений (метод анализа иерархий). Москва: Радио и связь, 1993. 278 с.

[11] Свідоцтво про реєстрацію авторського права на твір № 41832. Україна. Імітаційне моделювання в системному аналізі методом бінарних порівнянь. [Комп'ютерна програма] / Авторські майнові права належать І. В. Гілеті, В. М. Сеньківському, О. В. Мельникову. Зареєстровано 17.01.2012.

ОСОБЛИВОСТІ АНАЛІЗУ ТА ПРОЕКТУВАННЯ ВЗАЄМОДІЙ ІНФОРМАЦІЙНИХ СИСТЕМ В СЕРВІС-ОРІЄНТОВАНИХ АРХІТЕКТУРАХ

Хошаба О.М., Гречанінов В.Ф., Лопушанський А.В.

В даній роботі розглянуті особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах. Відносно визначеної структури сервіс-орієнтованої архітектури визначені особливості роботи кожної з основних компонентів, де надані їх стислі характеристики. Розглянуті особливості проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах, де визначені основні вимоги щодо організації роботи та безпека виконання демонстраційного прикладу, технології його створення та розробки сценарію. Показана важливість визначення вимог щодо роботи інтеграційної шини та створення програмного коду.

This paper considers the features of analysis and design of information system interactions in service-oriented architectures. In relation to the defined structure of the service-oriented architecture, the features of each of the main components are determined, where their brief characteristics are given. Features of designing interactions of information systems in service-oriented architectures are considered, where the basic requirements for the organization of work and safety of execution of a demonstration example, technologies of its creation and development of the scenario are defined. The importance of determining the requirements for the operation of the integration bus and the creation of program code is shown.

Постановка задачі. Існує сервіс-орієнтована архітектура, яка має такі основні компоненти: споживач (користувач), провайдер (постачальник) та брокер (каталог сервісів).

Необхідно: виконати аналіз та визначити необхідні умови реалізації конкретного демонстраційного прикладу взаємодії інформаційних систем в сервіс-орієнтованій архітектурі.

1. Особливості аналізу взаємодій інформаційних систем в сервіс-орієнтованих архітектурах. Метою створення демонстраційного прикладу є визначення аналізу та особливостей реалізації взаємодій інформаційних систем в сервіс-орієнтованій архітектурі (COA), яка являє собою модульний підхід до розробки програмного забезпечення. Даний підхід заснований на використанні розподілених і слабо пов'язаних компонентів. Такі компоненти повинні мати стандартизовані інтерфейси для взаємодії між собою відповідно до різних стандартів і протоколів [1-10].

Основою функціонування COA є робота незалежних сервісів у яких є чітко визначені інтерфейси. Такі сервіси викликаються стандартним способом для вирішення конкретних завдань з обміну даними. В цьому випадку, сервіси можуть не мати інформації щодо додатку яке їх викликає.

COA заснована на взаємодії між трьома компонентами (або ролями) (рис. 1): провайдером, реєстратором (або брокером) і споживачем. Взаємодія між цими ролями включає в себе публікацію інформації про сервіс, пошук доступних сервісів і взаємодії з цими сервісами.

На рис. 1. провайдер розміщує інформацію в каталозі сервісів. Провайдером (постачальником) визначається опис сервісів для різних служб. Інформація про сервіси публікується в реєстрі каталогу сервісів у брокера. Потім споживач використовує реєстр каталогу брокера, щоб знайти опису сервісів, які необхідно використовувати для реалізації необхідної завдання. Отримавши опис сервісів від брокера, споживач звертається до провайдера з допомогою WSDL, щоб отримати дані про роботу сервісів. Іноді, такий результат звернення споживача до провайдера розміщують в каталозі сервісів у брокера. Після цього, споживач виконує запити (SOAP) до провайдера для безпосередньої роботи з сервісами. Згідно технології COA, провайдер вважається власником сервісів, який має XML-опис і конкретну реалізацію.

Також, у провайдера існує досить велика кількість можливостей доступу до сервісів: від технології SOAP за допомогою HTTP до SMTP технології.

Програмно-апаратні комплекси, що розроблені на основі COA, реалізуються як набір веб-служб. Модулі таких комплексів взаємодіють між собою по протоколам SOAP, jini, CORBA, REST і інші.

В якості конкретного демонстраційного прикладу взаємодії інформаційних систем в сервіс-орієнтованій архітектурі в даній роботі наводяться дві системи: C4ISR і DRMIS. Окремі компоненти їх структурної схеми взаємодії на основі інтеграційної шини зображені на рис. 2.

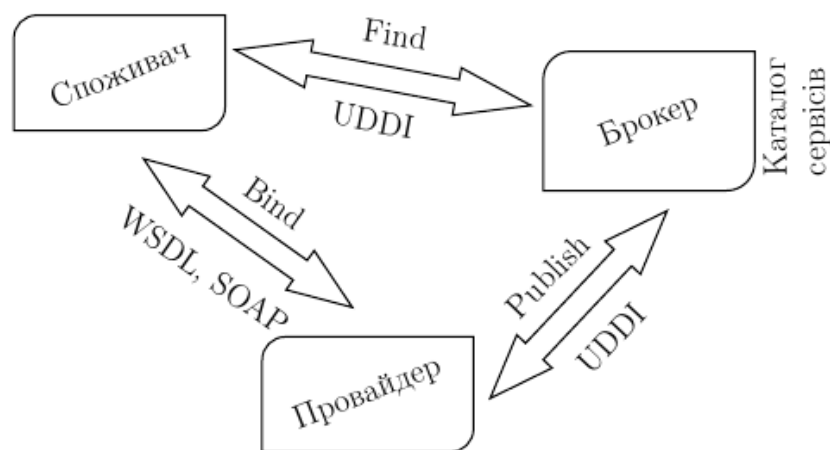


Рис. 1. Схема взаємодії (компонентів) ролей в сервіс-орієнтованій архітектурі

Така схема може являти собою ряд обчислювальних комплексів, які знаходяться на стороні інформаційних систем (рис. 2а) або інтеграційної шини (рис. 2б). Спосіб обміну даними на основі стандартів MIP4 може здійснюватися за допомогою стека протоколів TCP/IP або ґрунтуватися на файловому обміні (XML формат).

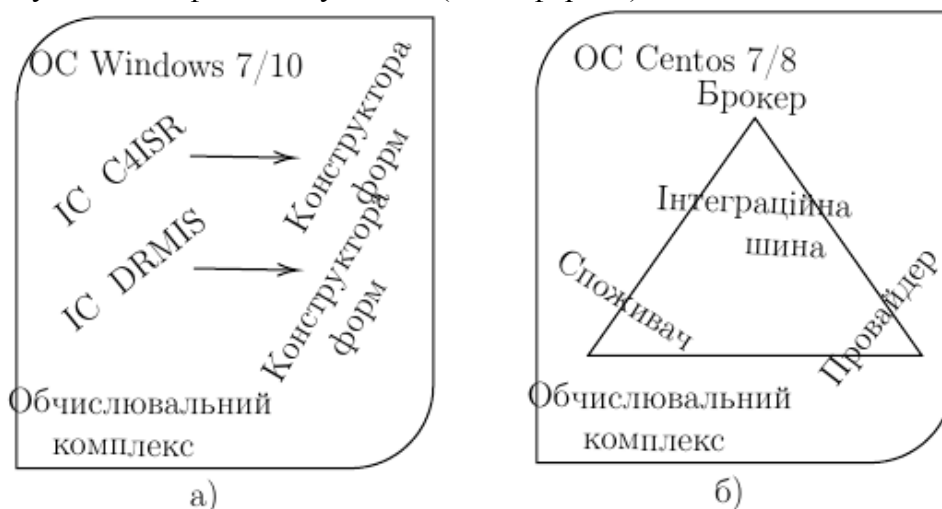


Рис. 2. Один з можливих варіантів структур інтеграційної шини в сервіс-орієнтованій архітектурі

Що стосується цих окремих компоненти структурної схеми взаємодії інтеграційної шини та інформаційних систем C4ISR і DRMIS, то для їх впровадження створюються організаційні та технічні вимоги побудови та експлуатації.

1.1 Особливості роботи каталогу сервісів (брокеру). Каталог сервісів демонстраційного прикладу являє собою веб-ресурс, який містить довідкову інформацію про сервіси. Також, каталог сервісів в певному сенсі є сервіс, який має певну URI-адресу, порт, директиви управління і обміну даними. Тому, довідкова інформація про сервіси знаходиться у брокера в каталозі сервісів. Брокер здійснює управління ресурсами каталогу сервісів (сховищами інформації), а провайдер (постачальник) створює довідкову інформацію і вносить зміни.

Можна також відзначити, що каталог сервісів демонстраційного прикладу, що представлений реєстратором (або реєстром) брокера, який включає в себе такі дані:

- бізнес-дані, такі як ім'я, опис і контактну інформацію (дані так званих 'білих сторінок');

- дані, що описують політики, бізнес-процеси і прив'язки програмного забезпечення щодо існуючих сервісів (дані так званих `зелених сторінок').

Брокер зазвичай пропонує розвинені функції пошуку інформації та дані бізнес-класифікації і таксономії (так звані дані `жовтих сторінок'). Згідно з концепцією COA, брокер надає доступний для пошуку реєстр описів сервісів (послуг), що опубліковані провайдерами.

Під час розробки систем, споживачі можуть використовувати інформацію, яка знаходиться в реєстрах для створення статичних прив'язок до сервісів (служб). Надалі, під час роботи системи, додатки можуть підключитися до реєстру (локального або віддаленого) для отримання опису служб і створення динамічних прив'язок до сервісів. Реєстри каталогу сервісів вирішують конкретні завдання: вони дозволяють знаходити сервіси провайдерів і параметри взаємодії з ними.

Каталог сервісів в своїй роботі використовує стандарти UDDI (Universal Description, Discovery, and Integration). UDDI розшифровується як універсальне опис веб-сервісів і застосовується для їх виявлення і інтеграції в єдину систему споживачів (рис. 1).

Робота каталогу сервісів також базується на стандартах UDDI Spec Technical Committee [10, 11, 18], які безпосередньо відносяться до роботи веб-сервісів. Також, існують стандарти каталогу сервісів, розроблені відомими компаніями Apache [18] та Oracle [15]. Такі стандарти пов'язані з роботою додаткових служб стандарти MIP4-IES, що задіяні в роботі COA [1-4].

Необхідно також відмітити, що проект UDDI є галузевої ініціативою, яка спрямована на те, щоб підрозділи, які представляють споживачів, провайдерів і брокерів могли швидко, легко, динамічно знаходити і здійснювати взаємодії один з одним використовуючи веб-сервіси в своїх корпоративних середовищах. Для цього в окремих корпораціях створюється каталог сервісів.

В цьому випадку, каталог сервісів є заповнений реєстр UDDI, який має каталогізувати інформацію про:

- існуючі підрозділи;
- послуги, які пропонують споживачі;
- стандарти зв'язку і інтерфейси, які використовують сервіси для надання послуг.

Створений таким чином каталог сервісів на основі стандарту передачі даних SOAP і UDDI працює як глобальний, незалежний від платформи простір відкритої архітектури, що виконує необхідні функції для використання сервісів.

Підтримку реєстру каталогу сервісів на основі UDDI здійснюють за допомогою використання двох способів:

- на основі веб-сервісів;
- на основі бізнес-реєстрів.

Тому, важливою складовою частиною архітектури COA є каталог сервісів (реєстр). До основного завдання каталогу сервісів належить ведення реєстру сервісів і їх записів. Важливими в роботі інструментами брокерів є реєстри UDDI і ebXML [14].

Таким чином, до основних складових частини архітектури COA належать (рис. 1): споживачі, постачальники (провайдери) і каталог сервісів, що мають різні цілі, ролі (функціональні можливості) і вимоги до системи (платформі). При цьому, є досить великий практичний досвід використання таких особливостей роботи COA, при якому в різних бізнес-підрозділах таких компаній як IBM і Microsoft [14], що використовують деякі ролі одночасно. У цій же роботі [14] відзначено, що в таких ролях представники архітектури COA були визначені з урахуванням взаємодії між окремими підприємствами. Тому, це не означає, що такий підхід може використовуватися тільки в інформаційних технологіях великого бізнесу. У зв'язку з цим відзначено [14], що в перспективі платформа веб-сервісів на основі архітектури COA буде перетворюватись на повноцінну інтернет-середовище віртуальних розподілених обчислень. При цьому послуги, що надаються постачальником сервісів повинні переслідувати наступні цілі:

- виконувати запити з високою ефективністю;
- мати можливість обробки великої кількості запитів;

- бути високо доступними;
- мати сучасні механізми відмовостійкості та самовідновлення;
- мати якісну підтримку системи (платформи).

1.2 Особливості роботи постачальників сервісів. Організації або підрозділи компаній, які надають послуги в електронному вигляді, ототожнюються з роллю постачальника (провайдера) послуг. Оскільки багато рішень задач технічно можуть бути реалізовані як сервіси, важливо визначити деякі приклади їх використання які є наступними.

Незалежний постачальник програмного забезпечення. Така організація або підрозділ створює і здійснює підтримку програмного забезпечення, яке виконує одну або кілька завдань. Таке програмне забезпечення може бути реалізовано як сукупність сервісів або додаток що розділений на окремі ресурси та призначений для широкого кола кінцевих користувачів (споживачів сервісів).

Корпоративний обчислювальний центр. Корпоративний обчислювальний центр може являти собою підрозділ, який звертається до різноманітних наборів додатків, що забезпечують виконання робочих процесів всередині організації (корпорації). Наприклад, у банку зазвичай є бізнес-процес що забезпечує підтримку виконання операцій з обробки кредитів. Однак, банк може отримати додатковий дохід, пропонуючи такий вид послуги по обробці кредитів зовнішнім організаціям. В цьому випадку, банк міг би надавати свій бізнес-процес по обробці операцій кредитування за допомогою корпоративного обчислювального центру як зовнішній веб-сервіс. Таким чином, він стає постачальником послуг для інших організацій. Тому, розмістивши платформу СОА на базі обчислювального центру, який є провайдером послуг, банк виступає в ролі споживача послуг. Однак, по відношенню до зовнішніх організацій (кредиторам), банк є брокером послуг так як може вказувати на технічні параметри сервісів іншим зацікавленим особам, що необхідні для виконання операцій кредитування корпоративних обчислювальним центром.

Структури, які є агрегаторами веб-сервісів СОА. Такі структури надають посередникам можливість створювати нові послуги за допомогою агрегування інших сервісів і виконують функції брокерів. Наприклад, якщо існує можливість надати сервіс, який перевіряє ряд банківських організацій то існує можливість на перевірку наявності кредитів з метою знаходження найкращої ставки. Даний приклад показує випадок брокера який не надає власних послуг.

Щодо розглянутих прикладів можна відзначити наступне. Постачальник (провайдер) послуг розглядає архітектуру СОА як основу для надання своїх веб-сервісів. У загальному випадку, ці послуги є прикладними програмами, що призначені для вирішення одного аспекту (завдання) загальної проблеми.

Для ефективного вирішення завдань можуть використовуватися наступні вимоги.

Забезпечення доступності сервісу. Сервіс є не досить ефективним, якщо він недоступний протягом деякого часу. Забезпечення того, щоб сервіс гарантовано брав запити на обслуговування від SOAP-маршрутизатора має велике значення внаслідок ефективного вирішення проблем. Використання балансування навантаження і масштабованості дозволяють ефективно вирішувати проблеми доступності сервісів.

Забезпечення безпечного середовища транзакцій. В даний час практично всі організації вже мають в тій чи іншій мірі різну систему безпеки. Проте, архітектура СОА дозволяє здійснювати гнучкі механізми забезпечення безпеки. Завдання реалізації таких механізмів полягає в тому, щоб створити умови, що засновані на досконалих стандартах, протоколах передачі даних, засобів аутентифікації і авторизацією.

Якість обслуговування. Архітектура СОА, яка ґрунтується на веб-сервісах і має ефективні механізми може бути реалізована для гетерогенних розподілених обчислень. В цьому випадку, до одного з найважливіших критеріїв якості обслуговування на основі веб-сервісів належить слідування специфікаціям обміну повідомлень на основі технологій UDDI і ebXML (Electronic Business XML, ebXML). Також, стандарти архітектури СОА

припускають застосування XML-схем для перевірки достовірності повідомлень. Тому, після отримання документа в СОА, веб-сервіс повинен:

- проаналізувати XML-повідомлення і упевнитися в коректності даних;
- визначити якість виконання послуги (Quality-of-Service);
- виконати перевірку політики безпеки або угод торгових партнерів.

Після цього, необхідно виконати ряд робочих операцій, що пов'язані з даним документом [11, 12].

1.3 Взаємодія споживачів і постачальників сервісів. Механізм взаємодії споживача і провайдера (рис. 1) називають зв'язуванням (bind). Цей механізм використовує опис сервісу для створення повідомлення, яке буде відправлено постачальнику служби.

Тому, згідно з концепцією СОА, постачальники надають послуги, що призначені для виконання необхідних та визначених завдань. До постачальників звертаються споживачі сервісів за допомогою спеціальних команд (директив) з метою обміну та управління даними. Більш якісну (або швидку) взаємодію між споживачами і поставщиками сервісів забезпечують брокери, які виконують публікацію або розміщення матеріалу на веб-ресурсах корпоративної (міжнародної) комп'ютерної мережі.

Публікація або розміщення матеріалу полягає в тому, що постачальник налаштовує інформацію (або метадані) про сервіс (або послуги) в реєстрі каталогу сервісів брокера. Постачальники, які взаємодіють з брокерами, можуть бути окремими організаціями з стандартизації, провайдерами програмного забезпечення або розробниками систем. Згідно з документом IBM Web Services Conceptual Architecture [17], постачальниками для публікації описів сервісів використовуються декілька різних механізмів:

- пряма (безпосередня) публікація сервісу. У цьому випадку споживач отримує опис сервісу безпосередньо від власника послуг (постачальника), використовуючи електронну пошту, FTP або дистрибутивний компакт-диск. При цьому, постачальник сервісу надає загальне поняття і одночасно робить її доступною для запитуючої сторони (споживача).

- публікація сервісу на основі HTTP GET запиту. Цей механізм зазвичай використовується в каталозі сервісів (на публічних сховищах веб-сервісів). В цьому випадку, споживач сервісу отримує опис служби безпосередньо від постачальника послуг за допомогою HTTP-запиту GET на каталозі сервісів.

- динамічне обстеження інформаційного оточення. Цей механізм необхідний як для споживачів, так і для постачальників сервісів, які використовують локальні та загальнодоступні реєстри для зберігання і отримання описів послуг. Найбільш частіше використовується технологія UDDI або ebXML. При цьому, постачальником послуг може бути додаток, яке використовує спеціалізований набір API для публікації опису послуги або отримання інформації про сервіс.

До особливостей використання різних механізмів публікації описів сервісів відносяться наступні.

Метод прямої (безпосередньої) публікації сервісів. Слід зазначити, що метод прямої (безпосередньої) публікації є одним з перших способів публікації на каталозі сервісів. Публікація за запитом HTTP GET є важливою серед різних механізмів опису сервісів в зв'язку з її простотою реалізацією. Однак, в деяких роботах [14] розглядається публікація за запитом HTTP GET як тимчасовий або перехідний спосіб, що дозволяє перейти від прямої публікації до динамічного обстеження інформаційного оточення. Тому, динамічне обстеження інформаційного оточення може являється ефективною і універсальною моделлю публікації сервісів.

Динамічне обстеження інформаційного оточення сервісів. Висока ефективність і універсальність динамічного обстеження інформаційного оточення полягає в тому, що споживачі сервісів незалежно один від одного і будь-яких сторонніх чинників звертаються до каталогів сервісів брокерів для пошуку інформації щодо запропонованих послуг. Далі споживач, яким може бути користувач або додаток, використовує спеціалізований набір API функцій до постачальника (провайдера) для реалізації запиту на предмет роботи з сервісами.

Також, динамічне обстеження інформаційного оточення сервісів може включати в себе WSDL механізми для отримання інформації про API-інтерфейсів (методів і параметрів роботи сервісів).

Такі запити формуються на основі стандартного XML формату. Відповіді на запити також передаються з використанням формату обміну повідомленнями XML, такого як SOAP або XML-RPC. Критерії, що використовуються для пошуку сервісу можуть включати в себе якість послуги (швидкість або адекватність відповіді), підтримку протоколів (можливість взаємодії з сервісом) і таксономію (різновид використання) сервісів.

2. Особливості проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах.

Проектування взаємодії направлено на визначення потреб у роботі інтерактивних систем, цифрових і аналогових пристроїв, середовищ, послуг і т.і. Основна увага в проектуванні взаємодії приділяється задоволенню потреб і побажань користувачів в реалізації систем, пристроїв або середовищ. При цьому, важливе місце займає вивчення поведінки існуючих систем, досвід і знання, одержувані в рамках вивчення людино-машинних взаємодій. Загальними вимогами щодо демонстраційного прикладу відносяться наступні.

Загальні вимоги до демонстраційного прикладу можна розділити на такі категорії:

- організація роботи демонстраційного прикладу;
- безпека виконання демонстраційного прикладу;
- технологія створення демонстраційного прикладу.
- вимоги щодо розробки сценарію;
- вимоги щодо роботи інтеграційної шини;
- вимоги щодо програмного коду.

Демонстраційний приклад повинен охоплювати роботу інтеграційної шини по взаємодії інформаційних систем C4ISR і DRMISS. Однак, дані вимоги щодо створення демонстраційного прикладу не охоплюють окремі інформаційні системи C4ISR і DRMISS.

2.1 Організація роботи демонстраційного прикладу. Сценарій демонстраційного прикладу повинен слідувати міжнародним стандартам, обміну та управління даними на основі MIP4-IES та COA. У демонстраційному прикладі повинна бути описана структура основних компонент і їх функціональні особливості. Показані приклади використання протоколів обміну SOAP для взаємодії інформаційних систем на основі XML форматів даних, описані сценарії демонстрації приклади.

У робочій групі з розробки демонстраційного прикладу повинні бути чітко обумовлено строки проведення окремих етапів робіт і артефакти (вихідні документи, звіти, код, лог-файли і т.д.).

З метою демонстрації роботи прикладу для замовників або зацікавлених ІТ-компаній, код разом з покроковою інструкцією використання може бути представлений на інтернет ресурсах в публічних сховищах. Також, необхідно мати можливість розміщення коду на приватних інтернет сховищах доступ до яких здійснюється за допомогою пароля.

Тому, демонстраційна працездатність самого прикладу може здійснюватися на майданчиках розробника, замовника або зацікавлених ІТ-компаній за допомогою таких складових як публічні (або приватні) сховища даних і покрокові інструкції з його використання.

2.2 Безпека виконання демонстраційного прикладу. Так як в подальшому, структури даних будуть містити секретні відомості, то дані повинні бути відокремлені від структур коду. При цьому, програмний код повинен знаходитися окремо від даних і не мати особливої цінності (наприклад, секретної). В цьому випадку, дані можуть знаходитися на захищених розділах локального диска окремого комп'ютера або на знімних носіях (жорстких дисках або флешках). В процесі роботи демонстраційного прикладу вони повинні монтуватися до локальної обчислювальної системи.

Безпосередньо, під час демонстрації прикладу можуть використовуватися апаратні і програмні фаєрволи, засоби криптографії, що необхідні для файлового обміну.

Під час операції обміну даними між інформаційними системами C4ISR і DRMIS слід дотримуватись таких вимог безпеки:

- монтування та демонтування зовнішніх знімних носіїв інформації;
- монтування та демонтування мережевих пристроїв передачі даних на час обробки інформації (крім локального інтерфейсу);
- шифрування та дешифрування, стиснення і розпакування даних при використанні файлового обміну.

2.3 Технологія створення демонстраційного прикладу. Технологія створення демонстраційного прикладу повинна ґрунтуватися на сучасних підходах віртуалізації та контейнеризації за такими етапами виконання робіт:

1. Збірка іміджу (файлового образу) демонстраційного прикладу, що необхідні для роботи з інтеграційною шиною (адаптерів, перетворювачів і т.і.).

2. Створення контейнера основних модулів демонстраційного прикладу, що необхідні для роботи з інтеграційною шиною.

3. Завантаження різних основних версій демонстраційного прикладу з можливістю здійснення відкату до попередніх номерів версій.

4. Розробка сервісних (або серверних) модулів демонстраційного прикладу, що необхідні для роботи з інтеграційною шиною з визначенням їх версійності, класів і методів.

5. Розробка клієнтської частини з можливістю автоматичної генерації вихідного програмного коду демонстраційного прикладу.

6. Розробка засобів тестування (тест-кейсів), збірка і виконання основних модулів демонстраційного прикладу, що необхідні для роботи з інтеграційною шиною.

7. Розробка засобів автоматизованого складання програмного коду.

Архітектура і технології, які будуть використовуватися під час розробки демонстраційного прикладу будуть спрямовані на його можливий та ефективний супровід в майбутньому.

Демонстрація прикладу має передбачати його повну автономну роботу. Така робота включає в себе відсутність інтернету або зв'язку з іншими корпоративними мережами і розгортанням (деплоєм) програмного коду на окремій локальній майданчику з наявністю повноцінних файлових сховищ. Також, операції обміну і управління даними між інформаційними системами C4ISR і DRMIS повинні бути максимально автоматизовані і виконуватись за допомогою скриптових мов програмування.

2.4 Вимоги щодо розробки сценарію. Сценарій демонстраційного прикладу повинен показувати процес обміну і управління даними між системами C4ISR і DRMIS за допомогою інтеграційної шини на основі існуючих стандартів MIP4-IES і COA [1-10].

Сценарій за певною структурою демонстраційного прикладу (що показані на рис. 2) може містити наступні вимоги до основних етапів його виконання.

Робота на стороні системи C4ISR. Формування запиту до DRMIS за допомогою конструктора форм. Результат роботи конструктора форм -- файл у форматі json. Виконання операцій стиснення і шифрування вихідного файлу (що містить запит до системи DRMIS).

Демонстрація транспортних функцій. Файловий обмін. Передача запиту від системи C4ISR в інтеграційну шину.

Робота і управління даними на стороні каталогів сервісів інтеграційної шини. Створення каталогів сервісів. Публікація сервісів провайдером. Виконання основних операцій у каталозі сервісів: створення, модифікація і видалення каталогів; створення, модифікація і видалення інформації про сервіси. Публікація сервісів провайдером. Демонстрація роботи запитів від провайдерів до каталогу сервісів за стандартом UDDI. Виконання пошуку сервісів споживачем на платформі jUDDI.

Системне адміністрування і управління даними на стороні провайдера інтеграційної шини. Запуск і зупинка сервісів на контейнерах. Розпакування і розшифровка файлу від

системи C4ISR. Синтаксична і логічна перевірка файлу. Перетворення файлу в формат XML і його імпорт в інтеграційну шину. Виконання CRUD-операцій над даними запиту від системи C4ISR. Виконання операцій взаємодії між сервісами систем C4ISR і DRMIS. Демонстрація роботи запитів за стандартами WSDL і SOAP. Ведення журналу виконання операцій перетворення і обміну даними в інтеграційній шині.

Демонстрація транспортних функцій. Файловий обмін. Обмін запитами в інтеграційній шині між сервісами C4ISR і DRMIS.

Робота на стороні системи DRMIS. Читання запиту від системи C4ISR за допомогою конструктора форм. Формування запиту до C4ISR за допомогою іншого конструктора форм. Результат роботи конструктора форм - файл у форматі json. Виконання операцій стиснення і шифрування вихідного файлу (що містить запит до системи C4ISR).

Повторення операції обміну і управління даними від DRMIS до C4ISR з використанням інтеграційної шини. На стороні системи C4ISR - читання відповіді від DRMIS за допомогою редактора форм.

2.5 Вимоги до інтеграційної шини. В процесі роботи демонстраційного прикладу інтеграційна шина повинна мати такі рівні:

- системний;
- прикладний.

Системний рівень інтеграційної шини передбачає наступні основні компоненти:

- операційна система (Linux CentOS);
- система віртуалізації (KVM);
- віртуальні машини (з використанням будь-яких операційних систем);
- системи управління конфігураціями ресурсів (Ansible).

Прикладний рівень інтеграційної шини передбачає наступні основні компоненти:

- віртуальні технології (Docker, Podman);
- кластера (Kubernetes);
- прикладні програми та сервіси (Java);
- інформаційні технології програмування (JAX-WS, JAX-RS, JAXB і інші);
- збірник проектів (Maven);
- система ведення логів (log4j);
- системи модульного тестування (JUnit5).

2.6 Вимоги щодо програмного коду. Програмний код повинен бути розроблений відповідно до стандартів НАТО щодо взаємодії з інформаційними системами за вказаними:

- протоколам передачі даних;
- запитам до інформаційних систем;
- використанні системних або прикладних бібліотек, мов програмування, технологій і т.і.

Програмний код може бути розроблений за *стандартної версії*. При необхідності може бути розроблена *розширена версія*.

Стандартна версія включає в себе роботу декількох користувачів на основі консольної версії з розробки та взаємодії основних компонент інтеграційної шини для вирішення завдань демонстраційного прикладу. Також, для стандартної версії передбачається розвинені засоби захисту і використання інформаційних ресурсів.

Розширена версія включає в себе роботу користувачів на основі інформаційної екосистеми з підтримкою сучасних фреймворків по роботі з інтеграційною шиною.

Розширена версія повинна мати підтримку великої кількості користувачів, використання розвинених механізмів управління та захисту інформації, ідентифікації користувачів, призначення ролей, моніторингових функцій всередині інтеграційної шини і т.і.

Розробка програмного коду має здійснюватися згідно основних принципів побудови систем (життєвого циклу) і включати: аналіз, проектування, розробку, тестування і впровадження макету (системи).

Організаційні вимоги до розробки сервісних адаптерів повинні відповідати трьом складовим:

- вхідної інформації, яка надається від конструктора форм інформаційних систем (адаптерів першого типу);
- вихідної інформації;
- алгоритму обробки (CRUDS операції) і управління даними.

Вхідна і вихідна інформація може бути представлена у вигляді однієї або декількох таблиць, в якій будуть знаходитися службова та операційні частини, обов'язкові ключові поля. Обов'язкові поля таблиць описують їх назви (аббревіатури) англійською мовою і змістом, що повинні відповідати різним типам даних (цілі, речові, довгі цілі і т.і.).

До типу даних може ставитися окрема таблиця. При наявності цих трьох складових буде виконуватися проектування діаграм класів і розробка програмного коду.

2.7 Завдання демонстраційного прикладу. Основними завданнями демонстраційного прикладу є:

- визначення структури та функціональних особливостей роботи інтеграційної шини;
- проектування структури даних і команд взаємодії сервісів інтеграційної шини;
- розробка функціональних обов'язків брокера, постачальника і споживача сервісів.

Висновки. В роботі згідно постановці задачі, визначено особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах, де зроблено наголос на структурі та функціонуванні основних компонентів, організації роботи розглянутих інформаційних систем, вимог і технології створення демонстраційного прикладу.

СПИСОК ЛІТЕРАТУРИ

1. MIP4 Information Exchange Specification (MIP4-IES). Overview. P. 13.
2. MIP4 Information Exchange Specification (MIP4-IES). Exchange Mechanism Overview. P. 28.
3. MIP4 Information Exchange Specification (MIP4-IES). Request/Response Exchange Pattern. P. 43.
4. MIP4 Information Exchange Specification (MIP4-IES). Publish/Subscribe Exchange Pattern. P. 49.
5. MIP4 Information Exchange Specification (MIP4-IES). File Exchange Pattern. P. 22.
6. MIP4 Information Exchange Specification (MIP4-IES). Information Definition Overview. P. 89.
7. Reference Architecture for Service Oriented Architecture Version 1.0. P. 104.
8. Web Services (WS-Topics) Topics 1.3 P. 40.
9. Software life cycle processes 12207-2008.} ISO/IEC/IEEE International Standard. Systems and software engineering. : URL: <https://standards.ieee.org/standard/12207-2008.html> (дата звернення: 14.08.2020)
10. UDDI Spec Technical Committee Draft, UDDI Version 3.0.2. P. 420.
11. Ньюкомер Э. Веб-сервисы для профессионалов XML, WSDL, SOAP and UDDI. -- СПб.: Питер, 2003. -- 256 с.
12. Веб-сервисы. XML, WSDL, SOAP и UDDI. Для профессионалов.} Отрывок из книги, глава 1. Введение в веб-сервисы. : URL: <http://citforum.ru/book/xml-wsdl/otryvok.shtml> (дата звернення: 02.03.2020)
13. Backus J.W., Bauer F. L., Green J., et al.} Revised report on the algorithmic language ALGOL60. December 1962, Volume 4, Issue 1, P.420-453.
14. David Chappell, Tyler Jewell.} Java Web Services. Publisher: O'Reilly. First Edition March 2002. P. 276.
15. Luc Clement (Systinet) Andrew Hately (IBM), Claus von Riegen (SAP AG), Tony Rogers (Computer Associates). UDDI Spec TC. UDDI Version 3.0.2. UDDI Spec Technical Committee Draft, Dated 20041019. P. 420.

16. Oracle Fusion Middleware. Programming Advanced Features of JAX-WS Web Services for Oracle WebLogic Server. 11g Release 1 (10.3.1). May 2009. P. 88.
17. IBM Web Services Conceptual Architecture} : URL: https://www.researchgate.net/publication/235720479\Web\Services_Conceptual_Architecture_WSCA_10 (дата звернення: 25.02.2020)
18. Kurt T. Stam, Alex O'Ree.} Apache jUDDI Guide. http://juddi.apache.org/docs/3.3/juddi-guide/html_single/ (дата обращения: 03.08.2020)
19. Wirth syntax notation : URL: https://en.wikipedia.org/wiki/Wirth_syntax_notation (дата звернення: 23.08.2020)