УДК 004.41

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДИНАМІЧНОГО ОНОВЛЕННЯ ПРОГРАМНИХ КОМПЛЕКСІВ НА ОСНОВІ СЕРВІСНО-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ

Тиран С.Р.[1)], Дзига Ю.В.[2)]

[1)] Тернопільський національний економічний університет, магістрант,

[2)] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

І. Постановка проблеми

Впровадження інформаційних систем дозволяє оптимізувати вартісні і часові витрати для досягнення поставлених завдань вже на початкових етапах, а аналіз накопичених системою даних дозволяє розробляти і впроваджувати більш ефективні алгоритми досягнення цих цілей. Слід врахувати, що необхідність використання інформаційних систем відчувають не тільки великі організації і органи державної влади, а й середні організації.

Розробка нової або розвиток існуючої системи з подальшим впровадженням - досить трудомісткий процес як за часом, так і по ресурсах. Тому найбільш гостро стоїть питання розробки архітектури, методів, принципів і стандартів створення ієрархічних розподілених інформаційних систем, що дозволяють забезпечити задані функціональні можливості, певний рівень безпеки, стійке функціонування в розподіленому середовищі при мінімізації витрат на програмування.

Основою більшості ієрархічних розподілених систем є управління потоками документів. На підставі цих документів здійснюється управління реальними об’єктами, характерними для тієї чи іншої організації, але управління документами і об’єктами, як правило, ведеться автономно кожним рівнем ієрархії, що знижує ефективність системи в цілому. Тому виникає задача розробки методів синхронізації потоків документів і об’єктів в ієрархічних системах.

ІІ. Мета роботи

Метою дослідження є розробка методів і засобів створення гнучких ієрархічних розподілених інформаційних систем з закладеними функціями розвитку і супроводу, що дозволяють розширити можливості автоматизованого поширення фрагментів системи і її поновлення при дотриманні умов фрагментарності, безперервності роботи в процесі оновлення і можливості локального відновлення програмного забезпечення до попередньої версії.

ІІІ. Методи та засоби динамічного оновлення програмних комплексів для систем з ієрархічною структурою

У міру розвитку і ускладнення програмного забезпечення зростає роль засобів, що забезпечують його обслуговування в процесі експлуатації. При цьому важливо, щоб серед цих засобів переважали ті, які мало залежали від оператора і виконували б свою роботу в автоматичному режимі [1]. У першу чергу сюди слід віднести процедури розгортання і поновлення [2].

Перш за все, необхідно виходити з того, що розв’язок задачі поновлення повинен забезпечувати масштабованість, тобто має бути придатний, як для відносно малих систем, так і систем, що працюють по всій території країни або навіть декількох країн. Звичайно, при великих масштабах проводити ручне оновлення стає просто неможливим, і тому потрібна автоматизація процесу.

Таким чином, для проведення оновлення систем ієрархічного типу необхідно виконання умов:

- автоматизації процесу;
- фрагментарності поновлення;
- безперервності роботи системи в цілому в процесі оновлення;
- можливості локальних відкатів програмного забезпечення до попередньої версії;
- наявності інструментальних засобів, які повинні забезпечувати підготовку пакетів оновлень і управління власне процесом оновлення.

Необхідно мати на увазі, що всі масиви об‘єктів можуть бути пов‘язаними, тобто зміна будь-якого об‘єкта може вплинути на функціонування інших. Причому зв’язки ці не завжди заздалегідь передбачувані. Тому ідеальним рішенням було б одночасна моментальна зміна всіх об’єктів поновлення або зупинка системи, по крайній мірі, на одному рівні і копіювання під час зупинки всіх об’єктів, або виконання відповідних процедур для сховищ даних.

Для цього пропонується метод, заснований на парадигмі N-версійного програмування [2]. Суть методу полягає в наступному:

1) Програмісти випускають всі об’єкти оновлення з певним номером версії.

2) У разі повного оновлення модулі версії V викликають сервіси версії V, а ті в свою чергу збережені процедури БД також версії V. Структура БД також в цьому випадку буде відповідати версії V. Допускається використання локального оновлення тільки для модулів, коли модуль має версію $V+1$, але викликає сервіс з номером версії V або більш ранньої. Тоді об'єкти версії V для сервера і БД не змінюються. Аналогічно при локальній зміні тільки сервісів, об'єкти для зміни БД відсутні, але відповідно модулі на клієнті повинні бути змінені, так як вони викликають більш пізню версію сервісу.

3) Всі об'єкти версії $V+1$ розміщуються на клієнті, сервері, і об'єкти $V+1$ виконуються для БД. Таким чином, на клієнті і сервері виявляються розміщеними дві копії програмних засобів (поточна версія і попередня). В БД також розміщуються дві версії збережених процедур. При цьому неможливо (або, принаймні, недоцільно) мати дві синхронно заповнені структури БД, що відрізняються, наприклад, довжиною одного поля в одній таблиці. Якщо структуру все ж потрібно змінити, то при оновленні доцільно вводити розширений варіант, який можна використовувати як в тій, так і в іншій версії програмного забезпечення.

4) У результаті в межах одного фрагмента розміщуються дві версії виконуваного програмного забезпечення: версія V і $V+1$. При цьому продовжує працювати поточна версія V (рисунок 1).

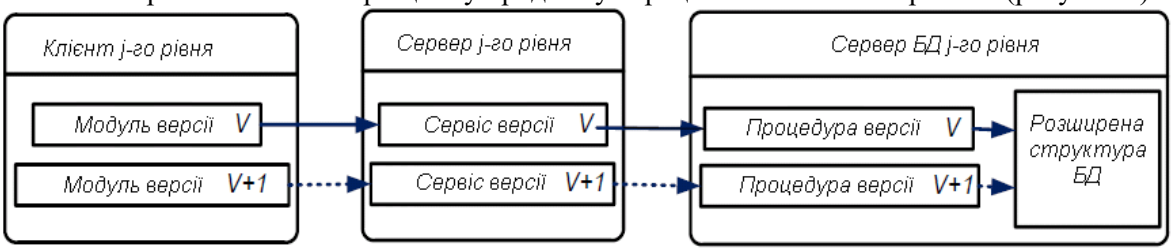

Рисунок 1 – Дві версії виконуваного програмного забезпечення

5) Перехід на нову версію після перезапуску сервіс-браузера та відповідного модуля на одному з клієнтських місць. Тепер для цього конкретного клієнта працює весь ланцюжок за версією $V+1$. Всі інші клієнти продовжують використовувати «стару» версію V, тобто функціонування здійснюється в мультиверсійному режимі.

6) У міру здійснення перезапусків на інших клієнтських місцях всі вони переходять на роботу з «нової» версії.

7) Якщо після першого перезапуску або інших, наступних за першим, виникла помилка, то можна провести відкат на «стару» версію. Для цього адміністратор видає команду «відкат» та ініціює новий перезапуск робочих місць, які в цьому випадку здійснять повернення на «стару» версію. Відзначимо, що в даному випадку відкат відбувається швидко, так як на всіх пристроях присутні всі версії програмного забезпечення і досить просто переключитися на роботу за «старою» версією. Тривала процедура зупинки системи (фрагмента), відновлення файлів, програм і процедур, що зберігаються при цьому відсутня.

8) Якщо «нова» версія «не прижилася», то вона підлягає корекції з подальшим повторним переходом до неї.

9) Якщо нова версія встановлена і експлуатується успішно, то після отримання об'єктів версії вона отримує статус «старої», а версія $V+2$ - статус «нової». Далі все повторюється так, як описано вище.

Відзначимо, що метод, який розглядався ніяк не пов'язаний з конкретним програмним забезпеченням. При їх побудові враховується тільки архітектура системи, можливість використання web-сервісів на серверах додатків, а на серверах баз даних - СУБД реляційного типу з організацією маніпулювання даними на основі мови SQL.

Висновок

Використання запропонованих методів дозволяє проводити розробку інструментальних засобів поновлення, які мінімально залежать від обслуговуваних систем, і можуть застосовуватися для різних програмних комплексів.

Список використаних джерел

1. Coulouris G. Distributed systems. Concepts and Design. Fifth Edition / George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. – Addiso-Wesley. – 2012.
2. Hosek P. Safe software updates via multi-version execution / P. Hosek, C. Cadar // Proceedings of the 2013 International Conference on Software Engineering. – IEEE Computer Society. – 2013. pp. 612-621.