УДК 681.3.01

В. І. Месюра, к. т. н., доц.;

О. А. Шаригін;

А. В. Козачук, студ.

АРХІТЕКТУРА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ КЕРІВНИКА ЛІКВІДАЦІЇ НАДЗВИЧАЙНИХ СИТУАЦІЙ

Запропоновано архітектуру системи підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій. Проаналізовано задачі, які стоять перед створюваним програмним забезпеченням. Обґрунтовано доцільність використання клієнт-серверної технології, здійснено виділення серверної і клієнтської частин програмного забезпечення. Розроблена архітектура системи базується на шаблонах проектування MVC (модель-вид-контролер), Proxy (Замісник,) Singleton (Одинак).

Система використовується у відділі навчально-бойової і спеціальної підготовки — Центрі підготовки рядового складу воєнізованої охорони Державного територіально-галузевого об'єднання «Південно-Західна залізниця».

Вступ

В [1] і [2] розроблено модель та алгоритми прийняття рішень для задач ліквідації надзвичайних ситуацій в умовах неповної визначеності, що дозволяє розробити відповідне програмне забезпечення. Однією з головних задач в процесі розроблення програмного забезпечення є створення архітектури.

Обґрунтування використання клієнт-серверної технології

Розробка в життєвому циклі програмного забезпечення традиційно складається з таких етапів, як аналіз, проектування, реалізація і тестування [3].

Розглянемо детальніше етап проектування. Існує певна кількість методик, що формалізують процеси проектування і розроблення програмного забезпечення з використанням об'єктно-орієнтованого підходу.

Особливого розповсюдження набули так звані паттерни або шаблони проектування [4]. Шаблони проектування — це архітектурна конструкція багаторазового використання, що є собою розв'язком загальної проблеми проектування в конкретному контексті і описує значення цього розв'язку.

Застосуємо теоретичний апарат цих засобів проектування для створення програмного забезпечення на основі структурної схеми формування рекомендації, що використовує раніше розроблені математичні моделі прийняття рішень [2].

Програмне забезпечення для створюваної інформаційної технології має бути встановлене на кожному об'єкті, на якому потенційно може статись надзвичайна ситуація. Зрозуміло, що встановити на кожний об'єкт високопродуктивний персональний комп'ютер із програмним забезпеченням достатньо витратно з фінансової точки зору. До того ж, немає сенсу в тому, щоб кожний комп'ютер мав великі обчислювальні можливості.

Така проблема вирішується застосуванням клієнт-серверної технології [5]. Найбільш ресурсомістку роботу виконує сервер, клієнт має забезпечити лише відображення інформації та, можливо, валідацію (перевірку введених даних). При цьому на об'єктах можна використовувати «легкі» термінали або портативні пристрої, які є набагато дешевшими за високошвидкісні робочі станції.

Отже, до основних переваг застосування клієнт-серверної технології для створюваного програмного забезпечення слід віднести:

1) потреба лише в одному швидкодійному сервері або кластері серверів, що економить кошти;

2) значне зменшення простоїв технічного забезпечення, оскільки ймовірність одночасних запитів з різних об'єктів у реальних умовах є низькою (надзвичайні ситуації виникають досить рідко) і у більшості випадків всі ресурси будуть витрачатись на оброблення однієї надзвичайної ситуації;

3) можливість постійного поповнення бази знань (у тому числі і за допомогою самої системи), що забезпечує клієнту постійний доступ до найновіших знань.

Виділення клієнтської та серверної частини програмного забезпечення

Якщо повернутись до схеми формування рекомендації, то структурні модулі можна поділити на серверні та клієнтські.

До серверних модулів відносяться: база знань; база даних; елемент виведення; елемент пояснення; модуль критеріїв; модуль формування результуючої рекомендації.

Клієнтською частиною є інтерфейс користувача. Виділення клієнтської частини надає додаткову перевагу — вона може варіюватись. Клієнтською частиною може бути як браузер, так і клієнтське програмне забезпечення. При цьому зв'язок серверу з клієнтом відбувається за допомогою технології веб-сервісу.

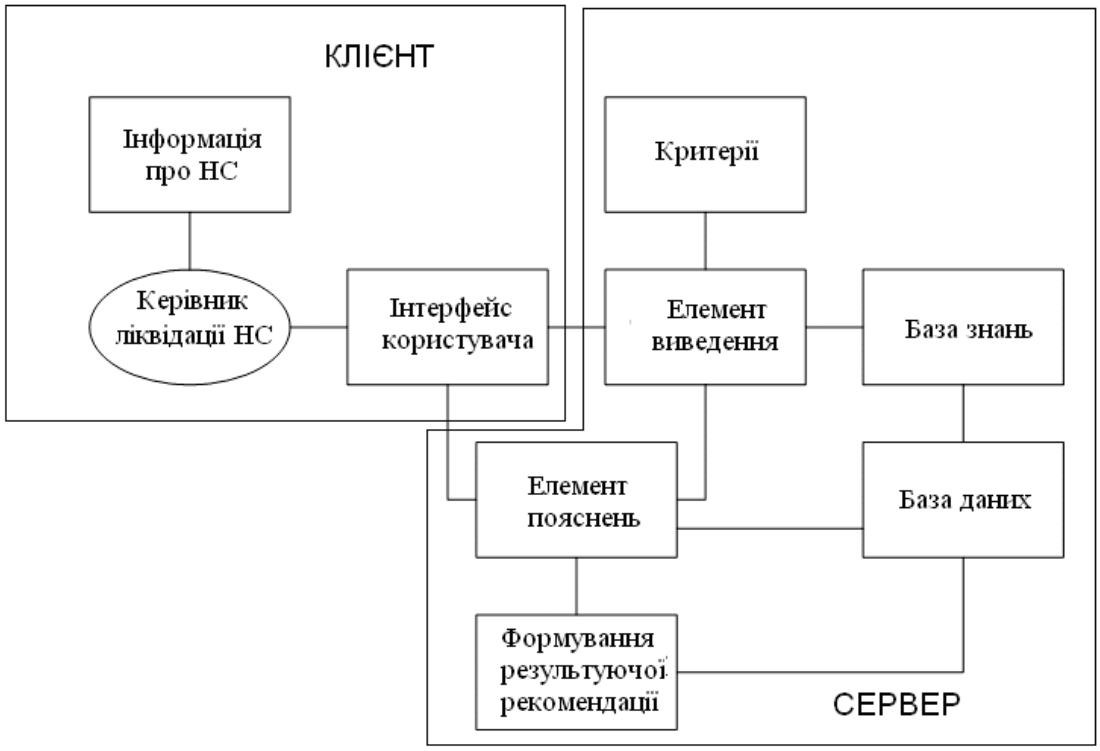


Рис. 1. Виділення серверної і клієнтської частин

На серверній стороні таку функціональність зручно робити за допомогою шаблону проектування MVC (Model-View-Controller). Його призначення — розділити поведінку інтерфейсу користувача на окремі частини, щоб покращити повторне використання програмного коду [4].

Шаблон проектування MVC складається з таких частин [4]:

1) модель — це певне позначення об'єкта-сутності. Вона може бути як простою структурою даних (XML-документ або певний набір даних), так і повноцінною моделлю предметної області;

2) подання — реалізує логіку подання інформації;

3) контролер — зв'язує модель і подання. Контролер отримує повідомлення від користувача, перетворює їх в дії, що виконуються над моделлю, а потім оновлює власне подання.

Якщо повернутись до створюваного програмного забезпечення, то:

— модель має містити майже всі серверні компоненти із структурної схеми формування рекомендації, а саме елемент виведення, елемент пояснення та модуль формування результуючої рекомендації, що використовують базу знань і базу даних з урахуванням модуля критеріїв;

— створимо дві версії подання: веб-подання, яке функціонує в браузері, та клієнтське подання, для якого створюється програмне забезпечення, що функціонує на стороні клієнта. При цьому

зв’язок із серверними компонентами відбувається за допомогою веб-сервісу, який знаходиться на сервері;

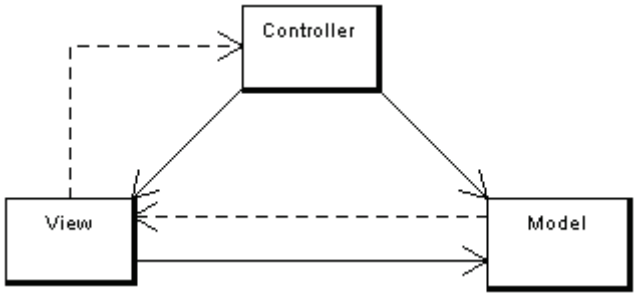


Рис. 2. Структурна схема шаблона проектування MVC

— контролер — це клас (або набір класів), який реалізує логіку взаємодії моделі та подання.

Зв’язок клієнтського програмного забезпечення з веб-сервісом здійснимо з використанням шаблону проектування Proxy (або «Замісник») [4]. Суть шаблону полягає в тому, що створюється сурогат громіздкого об’єкта. «Замісник» зберігає посилання на реальний об’єкт.

Використання веб-сервісу дозволяє зменшити обсяг інформації, яку слід передавати. Між клієнтською і серверною сторонами передається лише необхідна інформація (в форматі XML), чим виключається необхідність передачі повних сторінок з даними. Це дозволяє використовувати відносно нешвидкі засоби забезпечення Internet, наприклад, GPRS.

Для ведення протоколу всіх операцій створимо так званий Logger-клас, який реалізує спільну точку доступу і те, що існує лише один екземпляр цього класу. Logger-клас реалізуємо за допомогою патерна Singleton.

Але, у разі використання клієнт-серверної архітектури слід врахувати і її недоліки, пов'язані з можливими розривами зв'язку, що можуть бути викликані нестабільністю бездротового з'єднання, або проблемами оператора зв'язку. Якщо під час ліквідації надзвичайної ситуації зв'язок з сервером буде втрачено, застосування СППР стане неможливим.

Уникнення подібних ситуацій можна забезпечити використанням резервних каналів зв'язку, оскільки підключення до ще однієї GSM-мережі значно зменшить імовірність розриву зв'язку.

Але вдалішим рішенням може бути розробка оффлайн клієнта, що забезпечить локальне використання копії серверного програмного забезпечення. Такий підхід є прозорішим для користувача, перемикання режимів роботи може здійснюватись автоматично у разі появи та зникнення зв'язку з сервером.

Оффлайн клієнт повинен повністю повторювати функціональність серверу, тобто містити такі модулі, як база знань, база даних, елемент виведення, елемент пояснення, модуль критеріїв, модуль формування результуючої рекомендації. У разі використання локальних серверних модулів клієнтська частина продовжує працювати, змінивши джерело отримання інформації, що здійснюється непомітно для користувача.

За забезпечення своєчасного запуску оффлайн клієнта відповідає сервіс синхронізації. Цей сервіс веде моніторинг з'єднання з сервером і за відсутності з'єднання змінює налаштування клієнтської частини так, щоб остання робила запити до локального серверу.

Для підтримки актуальності знань та даних локального оффлайн серверу сервіс синхронізації повинен періодично оновлювати свої бази знань та даних. Періодичність оновлення залежить від швидкості з'єднання із сервером. Важливою функцією є можливість оновлення за запитом користувача.

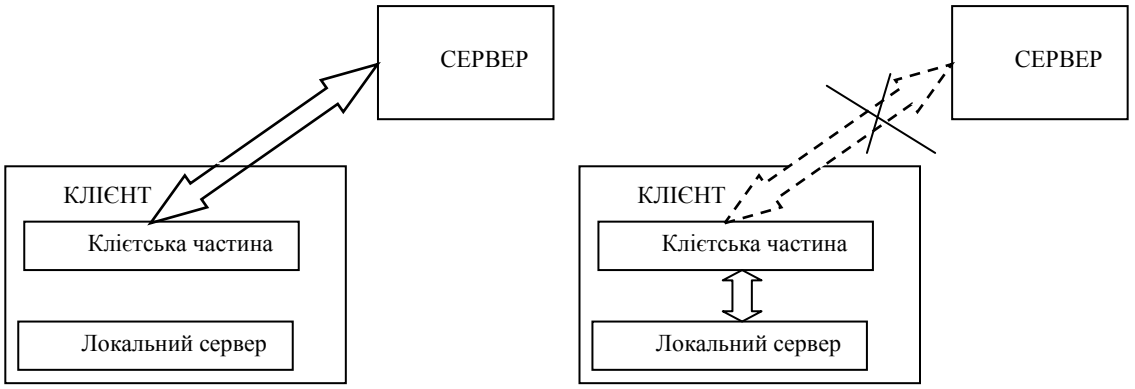
Недоліком використання локального серверу є зменшення швидкості відклику системи (через слабшу апаратну платформу), відсутність останніх оновлень бази знань та бази даних, збільшення апаратних вимог до платформи, на якій встановлено клієнт. Отже, використання локального серверу є недоцільним на смартфонах та планшетних комп'ютерах.

Перехід між режимами займає певний проміжок часу, під час якого система залишається не активною. При цьому перехід до оффлайн режиму триває довше через необхідність ініціалізації локальних серверних модулів.

Якщо з'єднання з сервером відновиться під час роботи системи, можливі два варіанти дій: передача отриманих оффлайн сервером тимчасових результатів на основний сервер з подальшим його використанням або продовження роботи з локальним сервером. Перший варіант слід застосовувати, якщо використання локального серверу суттєво зменшує швидкодію, але при цьому знадобиться час для передачі поточного стану системи по потенційно повільному та нестабільному каналу передачі даних. У разі використання другого варіанта передача даних буде виконана сервісом синхронізації після завершення роботи системи.

Отже, використання оффлайн клієнта збільшує надійність та відмовостійкість СППР, але на-

кладає додаткові вимоги на апаратне забезпечення клієнтських пристроїв.



а б

Рис. 3. Робота системи: а — у штатному онлайн режимі;
б — у аварійному оффлайн режимі

Висновки

Розроблена архітектура системи підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій. Виділено серверну і клієнтську частини. При розробці використовувались такі шаблони проектування як MVC, Proxy та Singleton.

Система використовується у відділі навчально-бойової і спеціальної підготовки — Центрі підготовки рядового складу воєнізованої охорони Державного територіально-галузевого об'єднання «Південно-Західна залізниця».

СПИСОК ЛІТЕРАТУРИ

1. Юхимчук С. В. Модель прийняття рішень для задач ліквідації надзвичайних ситуацій в умовах неповної визначеності / С. В. Юхимчук, О. А. Шаригін // Вісник ХНТУ. — 2008. — № 1(30). — С. 34—38.

2. Юхимчук С. В. Механізм виведення в системах підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій при нечітких вхідних даних / С. В. Юхимчук, О. А. Шаригін // Автоматика. Автоматизация. Электротехнические комплексы и системы. — 2005. — № 1(15). — С. 95—98.

3. Брукшир Дж. Гленн. Введение в компьютерные науки. Общий обзор. / Брукшир Дж. Гленн.; пер. с англ. — [6-е изд.] — М.: Издательский дом «Вильямс», 2001. — 688 с.

4. Sherif M. Yacoub. Pattern — Oriented Analysis and Design: Composing Pattern to Design Software Systems / Sherif M. Yacoub, Hany H. Ammar. — Addison Wesley, 2003. — 416 p.

5. Камерон Р. ASP.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов / Р. Камерон, Д. Михалк ; пер. с англ. — М. : ООО «И. Д. Вильямс», 2009. — 608 с.

Рекомендована кафедрою комп'ютерних наук

Надійшла до редакції 17.11.10
Рекомендована до друку 25.11.10

Месюра Володимир Іванович — професор кафедри комп'ютерних наук, ***Козачук Андрій Валерійович*** — студент Інституту магістратури, аспірантури та докторантури;

Вінницький національний технічний університет;

Шаригін Олександр Анатолійович — начальник відділу розробки, CDM Ukraine, м. Вінниця