АРХІТЕКТУРА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ КЕРІВНИКА ЛІКВІДАЦІЇ НАДЗВИЧАЙНИХ СИТУАЦІЙ

Володимир Месюра[1], *Олександр Шаригін*[2]

[1]Вінницький національний технічний університет
Хмельницьке шосе, 95, Вінниця, 21021, Україна, тел.: (0432) 43-78-80, E-Mail: vimes2009@ yandex.ru
[2] CDM Ukraine, Вінниця, 21050, Україна, вул. Театральна 43, E-Mail: as@cdm.dk

Анотація

В даній роботі побудовано архітектуру системи підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій.

Спочатку авторами аналізуються задачі, які стоять перед програмним забезпеченням, що створюється. На основі аналізу зроблено висновок про доцільність використання клієнт-серверної технології. Після цього проводиться виділення серверної і клієнтської частин програмного забезпечення. Слід відмітити, що в якості клієнтські частини може використовуватись як браузер, так і спеціально створений клієнтський додаток.

Розроблена архітектура системи підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій використовує такі існуючі шаблони проектування як MVC (модель-вид-контролер), Proxy (Замісник) та Singleton (Одинак).

Система використовується у відділі навчально-бойової та спеціальної підготовки – Центрі підготовки рядового складу воєнізованої охорони Державного територіально-галузевого об'єднання "Південно-Західна залізниця".

Вступ

В [1] і [2] було розроблено модель та алгоритми для прийняття рішень для задач ліквідації надзвичайних ситуацій в умовах неповної визначеності. Це дає змогу розробити програмне забезпечення, що базується на моделях і алгоритмах, що розроблено. Однією з головних задач при розробці програмного забезпечення є створення архітектури. Саме цьому й присвячена дана робота.

Основний текст

Розробка в життєвому циклі програмного забезпечення традиційно складається з таких етапів [3]:

- аналіз;
- проектування;
- реалізація;
- тестування.

В даній роботі детально розглянуто етап проектування.

Є певна кількість методик, які формалізують процеси проектування і розробки 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 являє собою розв'язання загальної проблеми проектування в конкретному контексті і описує значення цього рішення.

Спробуємо застосовувати існуючий теоретичний апарат засобів проектування для програмного забезпечення, що створюється. Підгрунтям для проектування програмного забезпечення є структурна схема формування рекомендації, що використовує математичні моделі прийняття рішень, що розроблені в [2].

Розглянемо більш детально задачі програмного забезпечення для інформаційної технології, що створюється. Воно 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 встановленим програмним забезпеченням достатньо витратно з фінансової точки зору. До того ж, немає сенсу в тому, щоб кожний комп'ютер мав великі обчислювальні можливості.

Така задача вирішується шляхом застосування клієнт-серверної технології [5]. Найбільш ресурсоємну роботу виконує сервер, клієнт має забезпечити лише відображення інформаційї та, можливо, валідацію (перевірку введених даних). При цьому на об'єктах можна використовувати "легкі" термінали або портативні пристрої, які коштують набагато дешевше високошвидкісних робочих станцій.

Отже, можна виділити такі переваги застосування клієнт-серверної технології для програмного забезпечення, що створюється:

1) Потрібен лише один швидкодіючий сервер або кластер серверів. Це економить кошти.

2) Ймовірність того, що одночасно будуть запити з різних об'єктів у реальних умовах є низькою (надзвичайні ситуації виникають досить рідко), тому у більшості випадків всі ресурси будуть витрача-

тись на обробку однієї надзвичайної ситуації. Таким чином, буде значно зменшено простій технічного забезпечення.

3) База знань системи може постійно поповнюватись (в тому числі і за допомогою самої системи). Таким чином, клієнт завжди має доступ до найбільш нових знань.

Якщо повернутись до [illegible], то [illegible] "[illegible]" та "[illegible]".

"[illegible]":

- [illegible];
- [illegible];
- елемент виведення;
- елемент пояснення;
- модуль критеріїв;
- модуль формування результуючої рекомендації.

"Клієнтською" частиною є інтерфейс користувача.

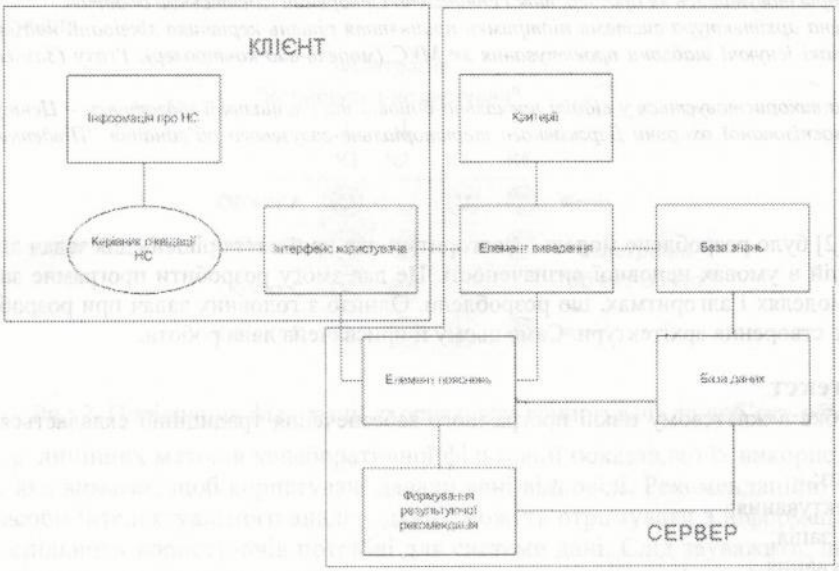

Рис.1. Виділення серверної і клієнтської частин

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

На стороні "серверу" 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 повноцінна модель предметної області;

2) представлення – реалізує логіку представлення інформації;

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

Якщо повернутись до програмного забезпечення, що створюється, то:

1) 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 та враховуючи модуль критеріїв;

2) створимо дві версії представлення – веб-представлення, яке функціонує в браузері, та клієнтське представлення, для якого створюється програмне забезпечення, яке функціонує на стороні клієнта. При цьому зв'язок із серверними компонентами відбувається за допомогою веб-сервіса, який знаходиться на сервера;

3) контролер – це клас (або набір класів), який реалізує логіку взаємодії моделі та представлення.

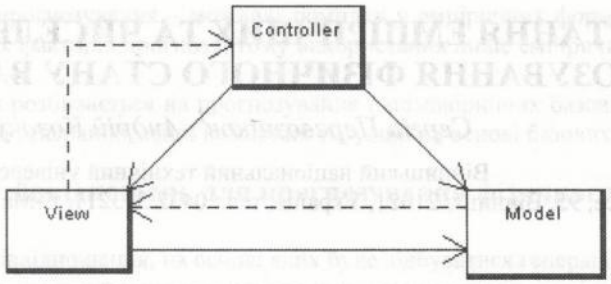

Рис. 2 Структурна схема шаблона проектування MVC.

Для зв'язку клієнтського програмного забезпечення з веб-сервісом застосовують шаблон проектування Proxy (або "Замісник") [4]. Суть шаблону полягає в тому, що створюється сурогат громіздкого об'єкту. "Замісник" зберігає посилання на реальний об'єкт.

Також слід додати, що використання веб-сервісу зменшує об'єм інформації, яку слід передавати. Між клієнтською і серверною стороною передається лише необхідна інформація (в форматі XML), не потрібно передавати повні сторінки з даними. Це дозволяє використовувати відносно нешвидкі засоби забезпечення Internet - наприклад, GPRS.

Додамо також, що в програмному забезпеченні, що створюється, варто реалізувати протокол всіх операцій, що виконуються в системі. Для цього створюють так званий Logger-клас, який реалізує спільну точку доступу і те, що існує лише один екземпляр цього класу. Logger-клас реалізується за допомогою шаблона проектування Singleton.

Висновки

В даній роботі була розроблена архітектура системи підтримки прийняття рішень керівника ліквідації надзвичайних ситуацій. Було виділено серверну і клієнтську частини. При розробці використовувались такі шаблони проектування як MVC, Proxy та Singleton.

Література:

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

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

[3] Брукшир, Дж., Гленн. Введение в компьютерные науки. Общий обзор, 6-е задание. : Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – 688 с.

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

[5] Камерон Р., Михалк Д. ASP.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов. : Пер. с англ.. – М.: ООО "И.Д. Вильямс", 2009. – 608 с.