УДК 681.614.8.084

ПІДХІД ДО ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНО-ЕКСПЕРТНОЇ СИСТЕМИ ЕКСПРЕС-ДІАГНОСТИКИ

Т. А. Григорова, Т. М. Веріченко, І. М. Перевернихата
Кременчуцький національний університет імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600, Україна. E-mail: gasta1@yandex.ru; tatiana.verichenko@gmail.com ivan.perevernyhata@gmail.com

Розроблено підхід до проектування та реалізації системи експрес-діагностики імунного стану організму людини. Цей підхід базується на використанні методик компонентно-зорієнтованого програмування. Ідея полягає у виділенні компонентів, що не залежать від типу додатку та платформи розгортання, в ядро інформаційної системи, яке може ініціалізуватися та конфігуруватися клієнтським кодом залежно від потреб цільової платформи. Система включає off-line та on-line додатки, працює з локальною та глобальною базами даних, кожна з яких використовує окрему технологію Sqlite та MSSql. Точкою входу до глобальної бази є web-сервіс, доступ до якого відбувається через медіатор ServiceAdapter, що реалізує інтерфейс сховища даних та інкапсулює в собі логіку роботи з web-сервісом. При проектуванні та реалізації системи використовувалися наступні паттерни: фабрика, стратегія, міст, команда, адаптер, MVVM та Wrapper. Розроблений підхід може використовуватися в проектах, що мають схожу архітектуру.

Ключові слова: підхід до проектування, інформаційна система, шаблони проектування, компоненти системи.

ПОДХОД К ПРОЕКТИРОВАНИЮ И РЕАЛИЗАЦИИ ИНФОРМАЦИОННО-ЭКСПЕРТНОЙ СИСТЕМЫ ЭКСПРЕСС-ДИАГНОСТИКИ

Т. А. Григорова, Т. Н. Вериченко, И. Н. Перевернихата
Кременчугский национальный университет имени Михаила Остроградского
ул. Первомайская, 20, г. Кременчуг, 39600, Украина. E-mail: gasta1@yandex.ru; tatiana.verichenko@gmail.com ivan.perevernyhata@gmail.com

Разработан подход к проектированию и реализации системы экспресс-диагностики иммунного состояния организма. Этот подход базируется на использовании методик компонентно-ориентированного программирования. Идея заключается в выделении компонентов, которые не зависят от типа приложения и платформы развертывания, в ядро информационной системы, которое может инициализироваться и конфигурироваться клиентским кодом в зависимости от потребностей целевой платформы. Система включает off-line и on-line приложения, работает с локальной и глобальной базами данных, каждая из которых использует отдельную технологию Sqlite и MSSql. Точкой входа в глобальную базу является web-сервис, доступ к которому происходит через медиатор ServiceAdapter, который реализует интерфейс хранилища данных и инкапсулирует в себе логику работы с web-сервисом. При проектировании и реализации системы использовались следующие паттерны: фабрика, стратегия, мост, команда, адаптер, MVVM и Wrapper. Разработанный подход может использоваться в проектах, имеющих сходную архитектуру.

Ключевые слова: подход к проектированию, информационная система, шаблоны проектирования, компоненты системы.

АКТУАЛЬНІСТЬ РОБОТИ. В сучасному комп'ютеризованому світі все більшого розмаху набирає рівень розвитку технологій програмування, поширюється поле їх застосування та, звичайно, зростає їх кількість. Перед інженерами постає задача правильного вибору інструменту для розробки програмного забезпечення, а саме: вибір технології та підходів до ефективної розробки ПЗ [1, 2]. Вона стає тим складнішою, чим більш розширюється спеціалізація інструментів для розробки, а розвиток інформаційних технологій веде саме до цього.

З огляду на тенденцію глобалізації та зростання створюваних інформаційних систем, їх виведення в web та cloud актуальною є задача розробка підходів до створення багатофункціональних складних інформаційних систем. Саме при проектуванні та реалізації програмного забезпечення, що має працювати на стику технологій та підходів, виникають принципово нові рішення, які розвивають інформаційні технології. Вимоги замовника щодо створення інформаційної системи експрес-діагностики імунного стану людини з підтримкою локального, web та cloud додатків [3], яка зможе вирішити ряд проблем, пов'язаних з можливістю проведення експрес-діагностики, за рахунок відкритого доступу до системи за допомогою Інтернет ресурсів (web та cloud додатки) та відвідування фахівців імунологів (локальний додаток), а також аналіз існуючих підходів, які ґрунтуються на основі клієнт-серверної та сервіс-орієнтованої архітектури [4], виявили необхідність створення нового підходу до розробки комбінованої розподіленої інформаційно-експертної системи.

Метою роботи є пошук рішення щодо проектування та реалізації додатків, які не залежать від платформи та типу додатку у середовищах розробки Microsoft Visual Studio Express та Visual Web Developer Express.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Спроектована система включає off-line та on-line додатки, працює з двома базами даних, а саме, – локальною та глобальною, кожна з яких використовує окрему технологію (Sqlite та MSSql). Точкою входу до глобальної бази є web-сервіс. Доступ до нього відбувається через медіатор ServiceAdapter,

який реалізує інтерфейс сховища досліджень та інкапсулює в собі логіку роботи з web-сервісом. Обидва типи додатків використовують його для відправки та отримання даних із глобального сховища.

Відповідно до вибраного компонентно-орієнтованого підходу в системі було виділено функціонал, який має бути загальним для всіх компонентів. Система включає в себе наступні компоненти: ядро системи, ресурсна бібліотека, локалізаційний модуль, web-сервіс, ліцензійна система, генератор ліцензійних ключів, бібліотека локалізації.

В ядро винесені інтерфейси та загальні базові класи без прив'язки до платформи та типу компонента. В ядрі також знаходяться конфігураційні класи, за допомогою яких можна налаштувати ядро для роботи з локальним або web-додатком.

Для меншої зв'язності системи ядро було розподілене на такі атомарні частини:

- модель даних та інтерфейси;
- система налаштувань;
- система управління створенням документів;
- експертна система;
- система статистики;
- розширяючі функції над .Net Framework [5].

З огляду на те, що ядро підтримує off-line і online додатки, не було можливості використати вбудований механізм налаштувань .Net. Для підтримки такого функціоналу довелося реалізувати механізм налаштувань, який зберігає свій стан між запусками програми.

Розроблена архітектура (рис. 1) – інтерфейсний об'єкт, що приймає в конструктор інтерфейс на сховище налаштувань (Xml сховище для web та реєстр для off-line версії) та використовує його для збереження і завантаження власних властивостей.

В об'єкта присутні події, на які можуть підписатися клієнтські об'єкти, щоб їх сповістили про завантаження або збереження налаштувань, і вони могли коректно відреагувати на їх зміну.

Більшість коду системи покрито unit-тестами. А саме: усі реалізації сховищ (Xml, Sqlite, реєстр), обробка правил експертної системи, допоміжний клас статистики.

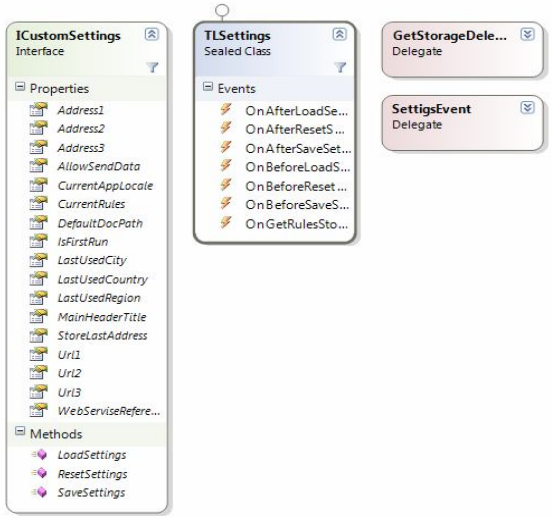

Рисунок 1 – Архітектура налаштувань

Архітектура експертної системи (рис. 2) безпосередньо залежить від моделі даних, так як при проходженні дослідження необхідно оперувати сутностями даних.

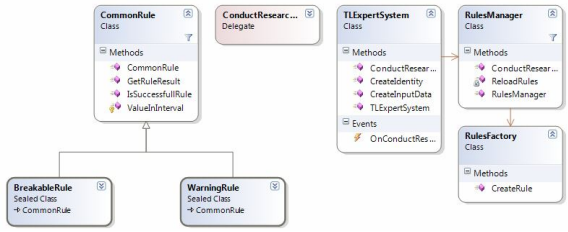

Рисунок 2 – Архітектура експертної системи

Експертна система ініціалізується сховищем правил (Xml сховище для обох версій), менеджер завантажує їх та створює відповідні об'єкти. Після цього експертна система готова до проходження досліджень. Такий механізм дозволяє досягти наступних цілей:

- можливість зміни правил в runtime;
- можливість локалізації правил і, як результат, весь вихідний документ;
- можливість використання unit-тестування для гарантування більшої коректності написання коду.

Для ініціалізації процесу проходження дослідження необхідно передати в об'єкт експертної системи об'єкти класів ResearchInput та Identity, на вихід отримати об'єкт класу ResearchResult. Цей об'єкт генерується менеджером правил експертної системи, який аналізує всі доступні правила в сховищі та динамічно фільтрує їх.

Необхідно зазначити, що деякі правила можуть вести до екстреного завершення дослідження, деякі – до очистки результатів. Діаграма активності, що описує таку поведінку, зображена на рис. 3.

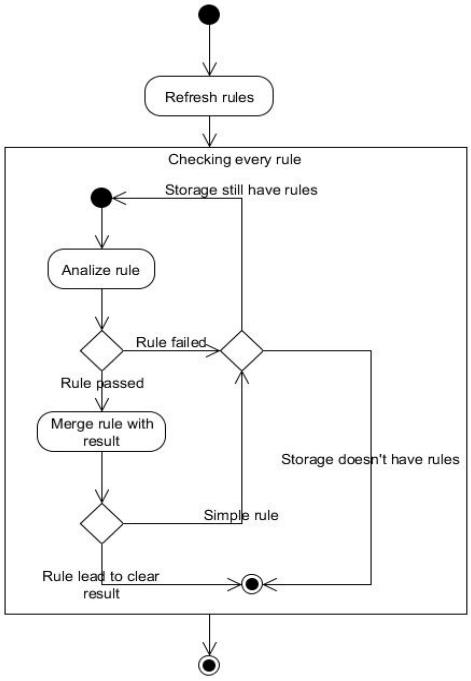

Рисунок 3 – Діаграма активності експертної системи

Під час розробки системи управління вихідними документами було враховано те, що система має можливість демонструвати як результуючі документи дослідження, так і статистичні документи. Також передбачено можливість збереження документів у різних форматах за допомогою уніфікації інтерфейсу документів та механізму роботи з ними. Ієрархіє документів зображена на рис. 4.

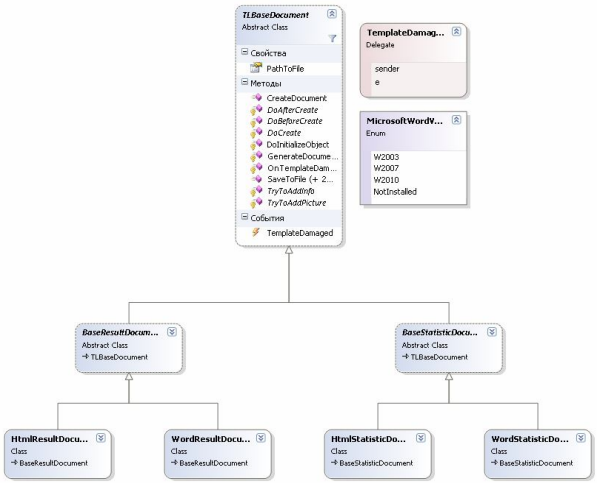

Рисунок 4 – Ієрархія вихідних документів

У базових класах знаходиться основна реалізація функціоналу, в класах-спадкоємцях – лише функціонал, специфічний для конкретного типу документу або його формату.

Допоміжна інфраструктура створює документ та керує ним, абстрагуючи роботу клієнтського коду від конкретних реалізацій. Вона складається із фабрики, яка створює об'єкти вихідних документів – класів, що інкапсулюють особливості роботи з різними форматами документів.

Під час проектування ліцензійної системи було виділено основні сутності: стан ліцензійної системи, ліцензійний менеджер, атомарна функція програми. Кожна з них відображена в відповідних класах і має універсальний інтерфейс, який дозволяє сховати особливості реалізації від клієнтського коду.

Архітектура ліцензійної системи (рис. 5) базується на наступних паттернах проектування: стратегія, фабрика та міст [6]. Кожна з описаних вище сутностей має абстрактний інтерфейс або клас, який дозволяє приховати конкретні реалізації. Це дозволяє використовувати безкінечну кількість різних станів ліцензійної системи, конфігурувати її як на етапі дизайну, так і у runtime.

Оскільки основний функціонал системи зосереджений в ядрі, то завданням прошарку MVVM є гнучка побудова логіки графічного інтерфейсу користувача.

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки, відомої як модель (можна сказати, що це відокремлення представлення від моделі). Модель представлення являє собою частину, яка відповідає за перетворення даних для їх подальшої підтримки і використання. Патерн команда, який було використано під час реалізації, є аналогом класичних обробників подій, але його можна повторно використовувати в будь-яких місцях, не прив'язуючись до конкретного представлення.

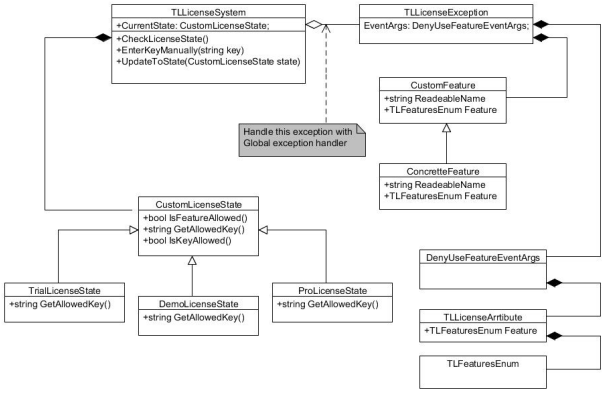

Рисунок 5 – Архітектура ліцензійної системи

Паттерн MVVM складається з двох базових класів для View Model, що відповідає за поля представлення, та Model, що відповідає за логіку представлень. Вони використовуються для уніфікації поведінки MVVM прошарку (наприклад, валідації). Усі їхні нащадки – конкретні реалізації для кожної функції програми (наприклад, налаштування або історія).

Допоміжні класи відіграють роль представлень для перелічувальних типів (наприклад, стать, мова програми, тип результатів досліджень). Було враховано те, що всі представлення, що їх використовують, мають бути локалізовані (рис. 6).

Для off-line версії використовується WPF [7] технологія, яка дозволяє реалізувати MVVM таким чином, що реалізація GUI виконується практично без зв'язності між бізнес-логікою та графічним інтерфейсом користувача.

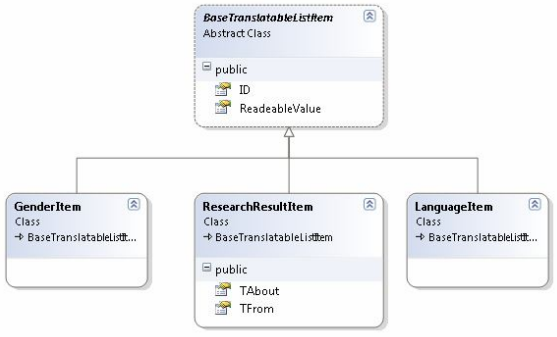

Рисунок 6 – Архітектура класу перелічувальних типів MVVM прошарку

Вся специфічна бізнес-логіка, що має відношення лише до off-line версії, повністю відв'язана від ядра програми або просто розширяє його.

Наприклад, система ліцензування використовується лише в off-line версії. Вона була реалізована для того, щоб вберегти продукт від несанкціонованого використання без відома замовника.

Результати дослідження виводяться у двох форматах – Microsoft Word та Html. Крім того, формуються різні типи документів – результуючий документ дослідження та результат статистичної обробки. Результуючий документ використовує контроль WebBrowser. Оскільки він не вбудований до WPF, довелося імпортувати додаткові інтерфейси Windows для його налаштування – mshtml.

Технологія WPF дозволяє легко створювати сучасний привабливий інтерфейс користувача за допомогою широкого інструментарію, доступного в Xaml розмітці.

Для зручності можуть бути використані інструменти, вбудовані в Expression Blend.

Для уніфікації стилів вікон створено базовий клас для всіх вікон програми. Всі вікна та контроли створюються за допомогою паттерну фабрика.

Більшість із них становлять собою контроли UserControl, що дає можливість комбінувати їх у будь-яких варіантах, що зменшує зв'язність один з одним.

Деякі інтерфейси користувача потребують покрокового вводу даних та відповідей на запитання. З огляду на це доцільно організувати візард, який інкапсулює логіку, що належить до покрокового вводу даних. Під час проектування та реалізації було передбачено необхідність валідації вводу даних, заміну представлень та порядок кожного кроку візарда, а також різну логіку роботи по завершенню.

Таким потребам задовольняє архітектура, що дозволяє в об'єкт візарда (вікна) передавати інтерфейс контролера, який управляє його поведінкою.

Було створено два контролера для візардів першого запуску та проходження дослідження (рис. 7).

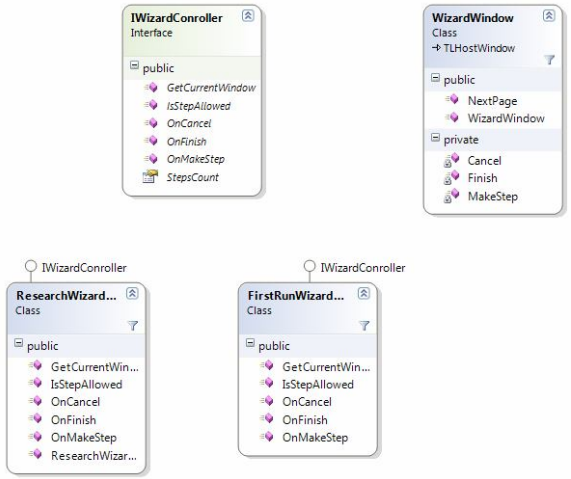

Рисунок 7 – Діаграма класів візардів програми

Off-line додаток підтримує декілька мов за рахунок механізму локалізації.

До особливостей команд в WPF можна віднести:

– незалежність від конкретного елемента управління (кнопки, тощо);

– підтримка швидких комбінацій клавіш;

– зміна стану елемента управління залежно від стану прив'язаної команди.

Off-line додаток підтримує декілька мов за рахунок механізму локалізації.

До особливостей команд в WPF можна віднести:

– незалежність від конкретного елемента управління (кнопки, тощо);

– підтримка швидких комбінацій клавіш;

– зміна стану елемента управління залежно від стану прив'язаної команди.

Основні команди, що не прив'язані до контексту викликів, зберігаються в модулі TLCommands.

Сім глобальних для всього додатку команд: показати історію, налаштування, довідку, локальну статистику, пройти дослідження, роздрукувати допоміжний бланк, вийти з додатку, доступні з будь-якого місця додатку. Інші – лише в тих view-моделях, в яких вони реалізовані (наприклад, команда збереження налаштувань).

On-line версія системи містить модуль проведення досліджень, модуль статистики, валідацію даних, локалізацію програми (російська та англійська мови).

Модуль проведення досліджень, що включає в себе експертну систему з температурних показників та показників аналізу крові, є частиною ядра, спільного для off-line та on-line додатків. Для перевірки коректності введення даних реалізовано механізм клієнтської та серверної валідації.

Для візуалізація інформації розподілу показника імунного стану використовується геоінформаційна система web-сервісу від Яндекс. Для забезпечення можливості виділення окремих регіонів карти були використані полігони областей. Їх було розташовано на сервері у вигляді xml-документів (рис. 8).

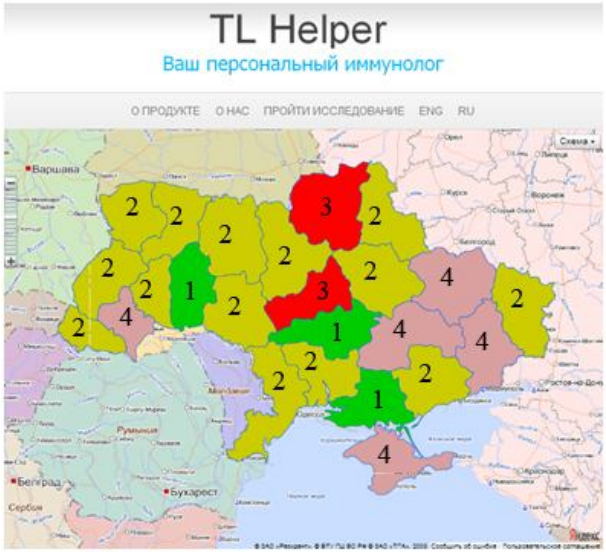

Рисунок 8 – Георозподіл показника імунного стану за регіонами країни (1 – гіперфункція, 2 – норма, 3 – гіпофункція, 4 – границя норми)

Для реалізації web-сервісу [8] було використано базу даних MsSql з огляду на високий рівень її інтеграції з .Net Framework. Серед ОРМ (як медіатора між базою та кодом сервісу) вибрано Ado.Net Entity Framework (EF) [9], що надає можливість взаємодії з об'єктами як за допомогою Linq у вигляді Linq to Entities, так і з використанням Entity SQL.

Проміжний клас TLServiceAdapter, що інкапсулює в собі логіку роботи з web-сервісом, реалізовано з використанням шаблону програмування Wrapper [6]. Він адаптує інтерфейс сервісу у зв'язку з тим, що інтерфейс посилання на сервіс знаходиться в іншому просторі імен. Таким чином, підключивши адаптер до будь-якого проекту, можна надати йому доступ до web-сервісу.

Асинхронність передачі даних у глобальне сховище реалізована за рахунок делегатів, які викликають метод в окремому потоці, імітуючи виконання багатьох задач одночасно.

ВИСНОВКИ. Розроблена система експрес-діагностики імунного стану є легко масштабованою як на локальні додатки, так і на web та cloud. Для її реалізації розроблено підхід до проектування та реалізації багатофункціональних складних інформаційних систем.

Цей підхід базується на використанні методик компонентно-орієнтованого програмування, що забезпечує можливість використання створюваного ПЗ як на базі клієнт-серверної технології, сервіс-орієнтованої архітектури, так і у вигляді локального програмного забезпечення. Ідея полягає у виділенні компонентів, що не залежать від типу додатку та платформи розгортання, в ядро інформаційної системи. Це ядро може ініціалізуватися та конфігуруватися клієнтським кодом залежно від потреб цільової платформи. Розроблений підхід може використовуватися в проектах, що мають схожу архітектуру.

З його використанням було розроблено модель та створено систему, що має гнучку архітектуру та вирішує широкий спектр задач, при цьому маючи мінімальну зв'язність між компонентами.

Розроблений програмний комплекс є експрес-методом самостійної діагностики імунної недостатності, що є доступним кожній людині. Продукт успішно експлуатується в установі практичної охорони здоров'я міста Кременчука, ЛДЦ "Вікомед".

ЛІТЕРАТУРА

1. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем. – СПб.: Питер, 2002. – 464 с.

2. Брауде Э.Дж. Технология разработки программного обеспечения / Пер. с англ.; под ред. Е. Строгановой. − СПб.: Питер, 2004. – 324 с.

3. George Reese. Cloud Application Architectures. – Cambridge: O'Reilly, 2009. – 206 p.

4. Карпов Л.Е. Архитектура распределенных систем программного обеспечения. – М.: МАКС Пресс, 2007. – 130 с.

5. Герберт Шилдт. C# 4.0: полное руководство / Пер. с англ., под ред. И.В. Берштейна − М.: ООО «И.Д. Вильямс», 2011. – 1056 с.

6. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. – Masachusetts: Addison Wesley Longman, 1995. – 368 p.

7. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 3.5 с примерами на C# 2008 для профессионалов / Пер. с англ., под ред. Ю.Н. Артеменко − М.: ООО «И.Д. Вильямс», 2008. – 250 с.

8. Гладцын В.А., Кринкин К.В., Яновский В.В. Сервис-ориентированная архитектура. Стандарты, алгоритмы, протоколы – СПб.: СПбГЭТУ "ЛЭТИ"', 2004. – 108 с.

9. Мэтью Мак-Дональд, Адам Фримен, Марио Шпушта. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов / Пер. с англ.; под ред. Ю.Н. Артеменко − М.: ООО «И.Д. Вильямс», 2011. – 1424 с.

AN APPROACH TO DESIGN AND IMPLEMENTATION OF INFORMATION AND EXPERT SYSTEM FOR RAPID DIAGNOSIS

T. Hryhorova, T. Verichenko, I. Perevernihata
Kremenchuk Mykhailo Ostrohradskyi National University
vul. Pershotravneva, 20, Kremenchuk, 39600, Ukraine. E-mail: gasta1@yandex.ru, tatiana.verichenko@gmail.com, ivan.perevernyhata@gmail.com

The authors have developed a new approach to design and implementation of information and expert system of rapid immune diagnosis. This approach is based on the use of component-oriented programming methods. Its conception consists of deriving the components that are invariant with a type of software or target platform, into the information system core that can be initialized and configured with a client code according to the target. The system consists of web and desktop applications, manipulate local and global databases, which use different Sqlite and MSSql technologies. The entry instruction to the global database is a web-service, the access to which takes place with the help of ServiceAdapter mediator, that implements the data store interface and encapsulates the web-service behavior. For the system development and implementation following patterns were used: Factory, Strategy, Bridge, Team, Adapter, MVVM, and Wrapper [10]. The approach discovered can be used in projects with a similar architecture.

Key words: design approach, information system, design patterns, system components.

REFERENCES

1. Orlov, S.A. (2002), *Tehnologii razrabotki programnogo obespecheniya. Razrabotka slognyh programnyh sistem* [Software development technology. The development of complex software systems], Piter, Saint Petersburg, Russia.

2. Braude, Eric J. (2004), Tehnologiya razrabotki programnogo obespecheniya [Software engering: An Object-Oriented Perspective], Translated by Strogonova, E., Piter, Saint Petersburg, Russia.

3. Reese, George (2009), Cloud Application Architectures, O'Reilly, Cambridge, Britain.

4. Karpov, L.E. (2007), *Arhitektura raspredelennyh sistem programmnogo obespecheniya* [The architecture of distributed software systems.], MAKS Press, Moscow, Russia.

5. Schildt, Herbert (2011), *C# 4.0: polnoe rukovodstvo* [C# 4.0 The complete reference], Translated by Bershteyn, I.V., OOO "I. D. Williams", Moscow, Russia.

6. Gamma, Erich; Richard Helm, Ralph Johnson, and John Vlissides (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Masachusetts, USA.

7. MacDonald, Matthew (2008), *WPF: Windows Presentation Foundation v .NET 3.5 s primerami na C# 2008 dlya professionalov* [Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5], Translated by Artemenko, U.A., OOO "I. D. Williams", .Moscow, Russia.

8. Gladycin, V.A., Krinikin, K.V. and Yankovskiy, V.V. (2004), *Servis-orientirovannaya arhitektura. Standarty, algoritmy, protokoly* [Service-oriented architecture. Standards, algorithms, protocols], SPbGETU "LETI", Saint Petersburg, Russia.

9. MacDonald, Matthew, Freeman, Adam and Szpuszta, Mario (2008), *Microsoft ASP.NET 4 s primerami na C# 2010 dlya professionalov.* [Pro ASP.NET 4 in C# 2010], Translated by Artemenko, U.A., OOO "I. D. Williams", Moscow, Russia.

Стаття надійшла 06.08.2013.