УДК 681.513

ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ЕФЕКТИВНОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

О.А. Самойленко
Міжнародний науково-навчальний центр інформаційних технологій та систем НАН України,
soa_pga@mail.ru

Розглядаються технології проектування та реалізації програмного забезпечення, що можуть використовуватись при створенні систем інформаційної підтримки управлінських рішень. Розглянуто головні архітектурні складові програмного забезпечення . Наведено основні можливості та особливості застосування сучасних технологій, що забезпечують ефективність і якість розробки як системи в цілому так і кожної складової окремо.
Ключові слова: система, інформаційна підтримка рішень, OOA, OOD, OOP.

The paper considers technologies of designing and developing software that can be used to create a combined system for information support of managerial decisions. The main architectural parts of the system are considered. The main capabilities and features of the contemporary technologies using that provide effectiveness and quality of the whole system developing as well as every part separately are presented.
Key words: system, informational decision support, OOA, OOD, OOP.

Рассматриваются технологии проектирования и разработки программного обеспечения, которые могут быть использованы при создании комплексной системы информационной поддержки принятия управленческих решений. Рассмотрены основные архитектурные составляющие системы. Приведены основные возможности и особенности использования современных технологий, которые обеспечивают эффективность и качество разработки как системы в целом, так и каждой компоненты отдельно.
Ключевые слова: система, информационная поддержка решений, OOA, OOD, OOP.

Вступ

Для ефективної побудови життєздатного програмного забезпечення необхідно мати знання про існуючі новітні технології як організації процесу конструювання програмного забезпечення (ПО), так і знання про технології, що використовуються на кожному з етапів розробки програмного продукту. В цій статті ми розглянемо основні технології, що застосовуються при розробці сучасних програмних рішень, розглянемо основні архітектурні складові системи інформаційної підтримки управлінських рішень, та технології, що можуть використовуватись як при конструюванні системи в цілому так і для реалізації окремих її складових частин.

1. Постановка задачі

Проаналізувавши вимоги, що висуваються при розробці програмного забезпечення професійного рівня, перелічимо вимоги, які будуть актуальними і для системи інформаційної підтримки управлінських рішень [1].

1. Гнучкість (flexibility). Здатність до адаптації і подальшого розвитку.
2. Розширюваність (extensibility). Можливість додання нових функціональностей без зміни основного коду системи.
3. Компонентна структура. Система повинна складатися з незалежних блоків. Заміна тієї чи іншої складової частини системи не повинна впливати на роботу інших частин.
4. Відокремлення даних від логіки. Дані не повинні змішуватись з їх представленням та бізнес логікою.
5. Кросплатформенність. Здатність системи працювати на різних операційних системах.
6. Можливість працювати з системою одночасно з кількох робочих місць.
7. Можливість доступу до даних і системи за допомогою мережі Інтернет.
8. Забезпечення захисту та безпеки даних.
9. Взаємодія системи з іншими існуючими системами (прийом даних, їх обробка і передача).

Виходячи з перелічених вимог, поставимо за мету розглянути основні технології реалізації і технології організації процесу конструювання програмного забезпечення, що буде задовольняти переліченим вимогам. Розглянемо основні складові системи інформаційної підтримки рішень за шаблоном MVC і технології, що використовуються для реалізації як кожної з компонент так і системи в цілому. В цьому і буде полягати задача нашої роботи.

2. Технології організації процесу конструювання системи та об'єктно-орієнтована методологія

Під технологією організації процесу конструювання програмного забезпечення будемо розуміти систему інженерних принципів для створення економічного програмного забезпечення, яке надійно і ефективно працює в реальних умовах [2]. Процес конструювання ПО зазвичай складається з наступних складових компонентів:

- планування і оцінка проекту;
- аналіз системних і програмних вимог;
- проектування;
- кодування;
- тестування;
- супровід.

Послідовність та способи реалізації і організації кожної із компонент і будуть визначати технологію організації процесу конструювання програмного забезпечення. На сьогодні існує декілька технологій організації процесу

конструювання. Серед них варто виділити класичний життєвий цикл та ітераційний процес розробки [3]. Класичний життєвий цикл – досить проста технологія, але в той же час є дуже неповороткою і не підходить для розробки програм, вимоги до яких часто змінюються. На відміну від класичного життєвого циклу, ітераційний процес представляється як досить гнучкий і повороткий, кожен цикл розбивається на невеличкі проміжки часу, під час якого аналізується та реалізується невелика частина вимог до програмного продукту, при чому вимоги можуть змінюватись в процесі розробки. Ітеративний процес на сьогоднішній день є одним з найбільш ефективних підходів до успішної розробки сучасних проектів.[2,3]

Конструювання сучасного програмного забезпечення неможливо уявити без використання об'єктно-орієнтованої методології (ООМ), яка складається з наступних частин [3]:

- об'єктно-орієнтований аналіз (ООА);
- об'єктно-орієнтоване проектування (OOD);
- об'єктно-орієнтоване програмування (ООР).
-

3. Основні компоненти системи і технології, пов'язані з розробкою кожної з компонент

Перш ніж приступати до побудови і аналізу архітектури системи, визначимося з мовою програмування. На наш погляд, найбільш ефективною для реалізації системи є мова Java, а саме пакет Java EE (Java Enterprise Edition) [4]. Перелічимо її основні переваги.

1) Незалежність від архітектури ПК і операційної системи, на якій виконується програмний продукт, написаний на Java. Така властивість забезпечується завдяки JVM (Java Virtual Machine). Кожна Java програма взаємодіє безпосередньо не з операційною системою а з JVM, встановленою на комп'ютері користувача. Точніше код програми компілюється в байт код, який інтерпретується тільки JVM, і не залежить від архітектури комп'ютера. JVM входить в пакет JRE, що безкоштовно представлений на офіційному сайті компанії Sun.

2) Безкоштовний пакет JDK (Java Development Kit) для розробників програмного забезпечення, який включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різноманітні утиліти і виконавчу систему Java (JRE).

3) Наявність безкоштовного середовища програмування IDE (Integrated Development Environment). Найбільш відомі IDE: Eclipse, Netbeans, IntelliJ IDEA, Borland JBuilder. Серед названих IDE особливої уваги заслуговує Eclipse. Eclipse є безкоштовним і в той же час дуже потужним і гнучким середовищем програмування. Останні дві властивості забезпечуються завдяки наявності дуже

великої кількості додаткових програмних компонент (plug-in), які дозволяють конфігурувати Eclipse в залежності від потреб розробника.

4) Підтримка великої кількості стандартних бібліотек. Наприклад до пакету Java EE входять наступні стандартні бібліотеки:
   - Ключові бібліотеки, що включають:
     - Collection бібліотеки, які реалізують структури даних, такі як: списки (lists), дерева (trees), карти (maps) і множини (sets)
     - Бібліотеки XML обробки (Parsing, Transforming, Validating)
     - Бібліотеки по забезпеченню захисту та безпеки даних
     - Бібліотеки інтернаціоналізації та локалізації.
     - Логування (logging)
     - Бібліотеки вводу/виводу
     - Бібліотеки для роботи з рядковими та числовими даними
     - Бібліотеки для обробки виключних ситуацій

       та інші.
   - Інтеграційні бібліотеки, що дозволяють розробникам взаємодіяти з зовнішніми системами:
     - Інтерфейс JDBC (Java Database Connectivity), що забезпечує організацію доступу до баз даних
     - JNDI (Java Naming and Directory Interface), стандартний програмний інтерфейс до корпоративної служби каталогів
     - RMI і CORBA – технології, для побудови розподілених систем.
   - Бібліотеки побудови і підтримки інтерфейсу користувача:
     - AWT (Abstract Window Toolkit) – набір інструментальних засобів, що дозволяють розробнику отримувати доступ до графічних елементів, наприклад, кнопкам, інтерфейс ним елементам, вікнам і т.д.
     - Swing бібліотека, що побудована на основі AWT і дозволяє більш швидко і зручно розробляти інтерфейс користувача.
   - Бібліотеки, що забезпечують реалізацію веб технологій, таких як:
     - Веб-сервіс
     - JSP (Java Server Pages)
     - EJB (Enterprise JavaBean)
     - J2EE Connector
     - JMS (Java Message Service)
     - JSF (JavaServer Faces)
5) Наявність великої кількості бібліотек класів, представлених сторонніми компаніями. Найбільш відомими з яких є Apache, Oracle, Google.
6) Java повністю підтримує об’єктно-орієнтоване програмування. Це дає змогу використовувати всі переваги цієї технології.

Для проектування системи використаємо архітектурний шаблон MVC (Model-view-controller) [3]. Цей шаблон поділяє систему на три частини: модель даних, представлення даних та керування. Застосовується для відокремлення

даних (модель) від інтерфейсу користувача (представлення) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, що полегшує подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонент програми. Крім того, використання цього шаблону у великих системах приводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Притримуючись принципів цього шаблону, визначимо головні компоненти системи (Рис. 1):

- модель даних та засоби доступу до моделі даних;
- бізнес-логіка;
- представлення даних (інтерфейс користувача).

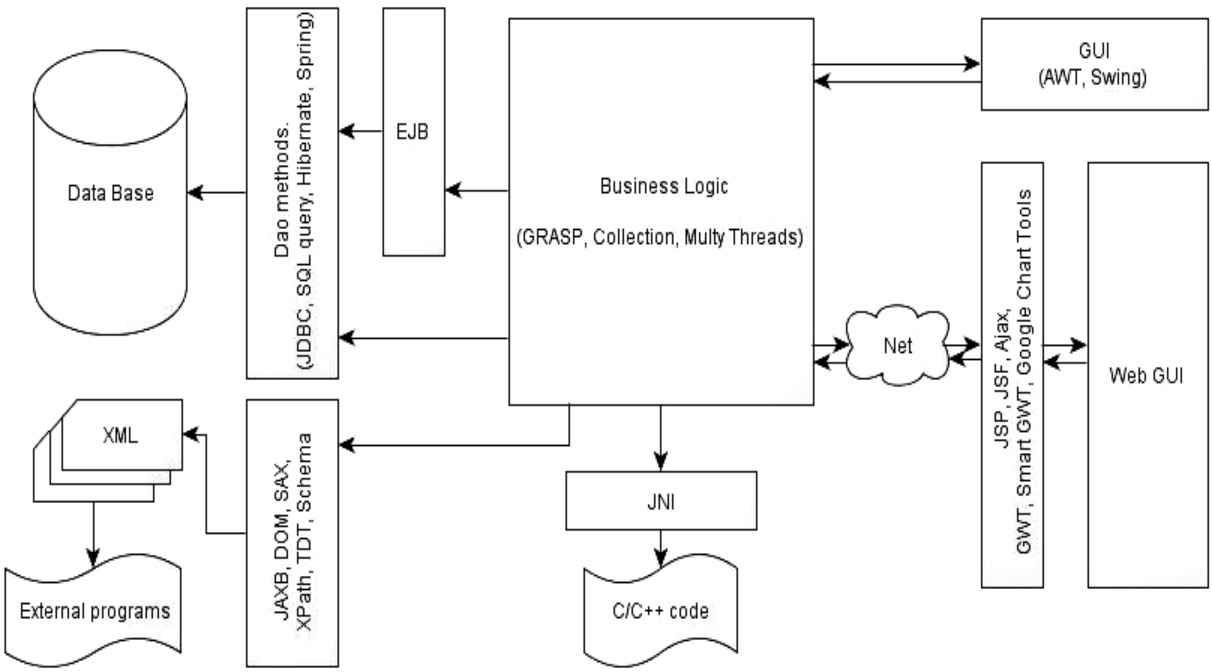

Рис. 1 Основні компоненти системи, представлені за шаблоном MVC

3.1. Модель даних та засоби доступу до моделі даних

Модель даних розділимо на дві частини: база даних та XML файли. Під базою даних (БД) розуміють впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система управління БД (СУБД). Розглянемо дві реалізації СУБД: Oracle і MySQL. СУБД Oracle є дуже потужною реалізацією з великим набором

сервісів, але в той же час вона є повністю комерційною системою. MySQL є більш облегшеною, реалізує менше сервісів і призначена для використання на малих і середніх підприємствах. На відміну від Oracle, MySQL розповсюджується безкоштовно.

Для доступу до бази даних зазвичай пишуться DAO методи, що реалізують певний інтерфейс, через який і взаємодіють з іншою частиною програмного коду.

Для реалізації DAO методів можуть використовуватись звичайні SQL запити, JDBC або технології Hibernate і Spring [5,6].

Під Hibernate розуміють засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Застосування Spring дозволяє, при написанні коду, дотримуватись кращих традицій об’єктно-орієнтованого програмування а також зменшити залежності даних від програмного коду. Бібліотека Spring усуває необхідність у JDBC кодуванні і представляє інтеграційні прошарки для популярних API, включаючи Hibernate. Hibernate і Spring представляють собою вільне програмне забезпечення.

Досить часто для забезпечення загального доступу до даних, або винесення певної функціональності за межі однієї програми використовуються Enterprise JavaBeans (EJB). EJB інкапсулює в собі певну логіку, що може потім використовуватись одночасно кількома програмами.

Для взаємодії з зовнішнім програмним забезпеченням будемо використовувати XML (Extensible Markup Language) формат, як стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними прикладними програмами [4]. Цей стандарт визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Набір тегів, що можуть бути в XML документі, порядок їх слідування, кількість та можливі аргументи визначаються правилами за допомогою DTD (Document Type Definition) або XML Schema. DTD є дуже легким способом описання правил, але має суттєві обмеження, завдяки чому не кожен XML документ може бути описаний DTD правилами. Тому в комерційних проектах використовується XML Schema, яка дозволяє точно описати структуру XML документу. Для обробки XML документів використовуються наступних п’ять технологій:

- Аналізатор SAX.
- Аналізатор DOM.
- Застосування механізму перетворення та фільтра (XSL-FO, XSLT XQuery, XPath).
- Активний аналіз.
- Зв'язування даних.

Об'єктна модель документа DOM (Document Object Model) є програмним інтерфейсом який дозволяє здійснювати обхід цілого документа так, наче він є деревом, вузли якого є об'єкти, що відтворюють зміст документа. Реалізації DOM мають тенденцію до інтенсивного використання пам'яті, оскільки, зазвичай, перед початком роботи документ має бути повністю завантажений, оброблений, та перетворений на дерево об'єктів.

SAX (Simple API for XML) є основаним на подіях інтерфейсом лексичного аналізу. Відповідно до цієї моделі, документ аналізується послідовно, а вміст документа передається на обробники подій аналізатора користувача. SAX є порівняно швидким і не потребує завантаження в пам'ять всього документу, тому при розборі великих документів використовується саме цей аналізатор.

Для швидкої обробки XML документів досить зручно використовувати технологію JAXB (Java Architecture for XML Binding) [7]. Ця технологія дозволяє на основі XML схеми генерувати готові класи, автоматично заповнювати об’єкти цих класів даними з XML документу (unmarshaling), а також представляти об’єкти генерованих класів у вигляді XML (marshaling). При використанні бібліотек JAXB не рекомендується користуватися пакетом java 6, так як стандартна бібліотека цього пакету містить класи, що конфліктують з класами JAXB, тому в такому випадку краще користуватися пакетом java 5.

3.2. Бізнес-логіка

До бізнес-логіки будемо відносити програмний код, що забезпечує певні маніпуляції з даними, виконання клієнтських запитів та виконання тих чи інших алгоритмів отримання нових даних на основі існуючих. Компонент бізнес-логіки є основою системи. Саме тут реалізуються основні функціональні можливості системи. А так як потреби до функціональних можливостей можуть досить часто змінюватись (наприклад, можуть додаватися нові алгоритми, нові підходи та методи обробки даних), то ця частина коду повинна обов’язково бути розширюваною, гнучкою та легкою в супроводі. Ці вимоги можуть бути реалізовані за допомогою наступних засобів та технологій:

1) шаблони проектування GoF (Gang of Four) та загальні шаблони розподілу обов’язків GRASP (General Responsibility Assignment Software

Patterns) дозволяють дотримуватись принципів OOD та OOP та ефективно вирішувати задачі загальними способами [8];

2) java doc реалізує ефективну і зручну підтримку документування коду [4];

3) jUnit, jMock, EasyMock та PowerMock технології забезпечують можливість ефективного написання тестів до кожного класу проекту, що дає змогу контролювати правильність реалізації окремої частини програмного коду і подальше коректне його функціонування при проведенні рефакторингу, чи доданні нових функціональних можливостей [9, 10];

4) обробка виключних ситуацій забезпечує надійність програмного продукту і стійкість в ситуаціях, що не є нормальними для основного ходу виконання програми [4];

5) технологія log4j забезпечує ведення протоколу виконання програми (logging), що є ефективним засобом при підтримці програмного продукту і відшуканні його дефектів;

6) JAR технологія забезпечує пакетування класів в один архів (бібліотеку) і використання цієї бібліотеки в інших проектах, завдяки цій технології існує безліч бібліотек, використання яких підвищує надійність та ефективність коду;

7) технологія Ant дозволяє автоматично виконувати компіляцію файлів та їх архівацію в jar-файл, запуск тестів та генерацію репортів аналізу коду, генерацію JAXB класів та збору файлів проекту в один пакет (delivery package);

8) системи управління версіями, такі як SVN (Subversion) чи CVS (Concurrent Versions System) забезпечують контроль над змінами коду та полегшують організацію командної роботи над одним проектом;

9) JNI (Java Native Interface) дозволяє під час виконання програми використовувати код C/C++, скомпільований у вигляді динамічних бібліотек.

3.3. Представлення даних (інтерфейс користувача)

Інтерфейс користувача можна будувати з урахуванням двох можливих варіантів використання системи:

- система використовується одним або кількома користувачами і всі її компоненти знаходяться на одному комп’ютері (standalone);

- компоненти системи знаходяться на різних комп’ютерних системах і віддалені один від одного, взаємодія між компонентами відбувається через локальну мережу чи Інтернет, система може мати кілька інтерфейсів, користувачі можуть мати доступ до системи з будь-якого комп’ютера за допомогою Інтернет.

В першому варіанті для реалізації інтерфейсу користувача достатньо використати AWT і Swing технології.

AWT (Abstract Window Toolkit) - це оригінальний пакет класів мови програмування Java, що слугує для створення графічного інтерфейсу користувача (GUI). AWT визначає базовий набір елементів керування, вікон та діалогів, які підтримують придатний, простий до використання, але обмежений у можливостях графічний інтерфейс. Однією з причин обмеженості AWT є те, що AWT перетворює свої візуальні компоненти у відповідні їм еквіваленти, платформи на якій встановлена віртуальна машина Java. Щоб компенсувати ці обмеження, розроблені класи Swing, як частина бібліотеки базових класів Java (JFC). Swing хоч і надає більше можливостей з роботою з графікою, проте не заміняє їх повністю, тому одночасно можуть використовуватись і AWT, і Swing технології.

Для реалізації GUI, що забезпечує взаємозв'язок користувача з даними на віддаленому комп'ютері, можуть використовуються такі технології як DHTML та JavaScript. Але в використанні цих двох технологій є певні недоліки: вони не підтримують повністю ООР і не реалізують підхід AJAX (Asynchronous JavaScript And XML) [11], що зараз став стандартом у розробці сучасних web-проектів. Використовуючи DHTML і JavaScript, необхідно самостійно слідкувати за дотриманням принципів ООР і AJAX. На відміну від них технологія JSF (Java Server Faces) підтримує ООР і компонентну модель реалізації GUI. Ця технологія служить для того, щоб полегшувати розробку користувацьких інтерфейсів для Java EE проектів. Для відображення даних JSF використовує JSP (Java Server Pages), технологію, що дозволяє динамічно генерувати HTML, XML та інші web-сторінки. JSP дозволяє вставляти Java-код, в статичний вміст сторінки. Також можуть використовуватись бібліотеки JSP тегів для вставки їх в JSP-сторінки. Сторінки компілюються JSP-компілятором в сервлети, які є Java-класами, і виконуються на сервері.

GWT (Google Web Toolkit) [12] дозволяє створювати AJAX рішення на основі java. GWT технологія реалізує AJAX підхід за допомогою java класів, які в результаті компілюються в відповідний до браузера JavaScript і HTML код. Технологія підтримує MVC підхід, що забезпечує відділення даних від коду. GWT має багату бібліотеку віджетів (примітивів GUI), використання яких дозволяє легко і швидко будувати графічний інтерфейс web-проекту. До того ж існує потужний plug-in для Eclipse - GWT Designer, який візуалізує процес побудови графічного інтерфейсу з використанням GWT технології. Smart GWT має більш багатшу бібліотеку віджетів, але в той же час потребує багато часу при першому завантаженні сторінки web-проекту. Корисними будуть для використання разом з GWT деякі додаткові бібліотеки від Google, наприклад Google Chart Tools, що має велику кількість класів представлення даних у вигляді графіків.

Висновки

Застосування розглянутих технологій дозволяє розробити високо-якісний програмний продукт професійного рівня, що задовольняє вимогам які нині висуваються до сучасного програмного забезпечення. Практично всі перелічені технології і засоби є вільними у використанні, що дає змогу використовувати їх у наукових цілях без додаткових матеріальних затрат. Розглянуто технології, що можуть використовуватись при проектуванні і реалізації систем інформаційної підтримки управлінських рішень.

Література

1. Самойленко О.А., Степашко В.С. Конструювання комплексної системи інформаційної підтримки управлінських рішень // Збірник праць. – Київ: МННЦ ІТС. 2009. — С. 211 - 219.
2. Орлов С.А. Технологии разработки программного обеспечения: Уч. пособ. – Киев: Питер. 2003. — 473 с.
3. Ларман К. Применение UML2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ и итеративную разработку. – Киев: Вильямс. 2007. — 727 с.
4. Хорстманн К.С., Корнелл Г. Core Java. – Киев: Вильямс. 2009. — 2056 с.
5. Bauer C., King G. Java Persistence with Hibernate. – Greenwich: Manning. 2006. — 880 p.
6. «Spring Documentation». – доступний з: http://www.springsource.org/
7. «JAXB». – доступний з: https://jaxb.dev.java.net/
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – Киев: Питер. 2008. — 361 с.
9. «PowerMock». – доступний з: http://code.google.com/p/powermock/
10. «The jMock cookbook». – доступний з: http://www.jmock.org/cookbook.html
11. Mahemoff M. Ajax design patterns – NY: O'Reilly Media. 2006. — 656 p.
12. Geary D., Gordon R. Google Web Toolkit Solutions – Boston: Prentice Hall. 2008. — 408 p.