ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2015. Вип. 23. С. 108–118
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 23. P. 108–118

УДК 004.4

ТЕХНОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ТИПОВОЇ ПІДСИСТЕМИ ІНТЕГРОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ УНІВЕРСИТЕТОМ

В. Кухарський, І. Урсул

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000, e-mail: vitaliy.kukharskyy@lnu.edu.ua

Описано процес розробки програмного продукту для підсистеми управління навчальним процесом – найважливішої типової підсистеми інтегрованої інформаційної системи управління університетом. Врахувавши всі особливості бізнес-процесів вищого навчального закладу, представлено процес проектування бази даних, що передбачає інтеграцію та синхронізацію даних з зовнішніми системами; визначено архітектуру майбутнього програмного продукту; сформовано концепцію забезпечення процесу розробки програмного забезпечення з використанням технологій контролю версій, автоматизації процесів, міграцій бази даних, тестування системи, фактично сформувавши екосистему для майбутніх проектів (https://github.com/ifnul).

Ключові слова: інтегрована інформаційна система управління університетом, програмний продукт, база даних, архітектура системи, розробка програмного забезпечення, тестування.

1. ВСТУП

В наш час якість освіти має визначальне значення для успішного розвитку будь-якої країни. Революційні зміни технологій, які опираються на інтелектуальні ресурси, стають найважливішим чинником, який зумовлює розвиток світової економіки. Визначальна роль вищих навчальних закладів саме у розвитку інтелектуальних ресурсів країни, конкурентоспроможних на сучасному ринку праці, тобто спроможних забезпечити реалізацію високоефективних інноваційних проектів у різних секторах економіки. Пошук нових сучасних інструментів, які забезпечать затребуваний сьогодні ринком рівень якості освітньої діяльності, загострився посиленою конкуренцією серед ВНЗ і особливо актуальний для сучасної освіти світового рівня та для української національної системи освіти зокрема, якій інтеграція у Європейську систему освітніх послуг ставить нові вимоги до організації управління навчальним процесом.

Навчальний процес у вищому навчальному закладі – найважливіший і найскладніший вид діяльності (бізнес-процес). Від правильної організації управління навчальним процесом залежить якість підготовки фахівців, а відповідно, і конкурентоспроможність спеціальностей, за якими ведеться підготовка, і навчального закладу в цілому.

На підставі аналізу європейського досвіду, міжнародних стандартів у галузі освіти та з урахуванням специфічних особливостей освітнього середовища в Україні, а також усвідомлення вищими навчальними закладами потреби в активізації зусиль у напрямі гарантії якості освіти стосовно вимог Болонського процесу, виникає гостра потреба розробки та впровадження інструменту для удосконалення університетського менеджменту, який відповідатиме сучасним вимогам і стандартам, оскільки саме

управлінські рішення спроможні змінити всю систему в цілому, а від їхньої правильності та своєчасності залежить ефективність системи освіти. Одним з елементів удосконалення управління навчальним процесом в університеті є його автоматизація, тобто створення та впровадження системи управління навчальним процесом, яка б враховувала тонкощі навчального процесу в університетах, незалежно від їхньої спеціалізації. Брак автоматизованих рішень продовжує тягнути за собою відповідні додаткові штатні ресурси, неминучі помилки і, навіть, збої в навчальному процесі, що суттєво гальмує вихід української системи освіти на міжнародний рівень.

Проведені дослідження та аналіз типових систем управління освітнім процесом у Європейських університетах -партнерах та українських навчальних закладах дають підстави говорити про те, що сьогодні не існує досконалої уніфікованої системи управління навчальним процесом, яка б була вигідною та повнофункціональною для всіх сторін-учасників навчального процесу, відповідала б усім вимогам Болонського процесу та міжнародним освітнім стандартам.

Очевидно, що робота навчальних закладів без використання управлінських інформаційних систем чи з використанням недосконалих рішень стає неефективною, що впливає передусім на якість освіти. Тож створення якісного сучасного програмного продукту, що реалізує автоматизовану систему управління навчального процесу, є надважливим для університетів.

В Україні питанням створення типової системи управління навчальним процесом займалися представники багатьох університетів із залученням компаній-розробників спеціалізованого програмного забезпечення вже протягом тривалого часу. Створюючи власні розробки та певні версії подібної системи, українські навчальні заклади намагалися самостійно вирішити цю проблему, однак з огляду на об’єктивні труднощі, брак європейського досвіду та необхідних матеріально-технічних ресурсів такі спроби найчастіше не увінчувалися успіхом, або ж вирішували проблему лише в середині певного університету, або лише на відповідному (окремому) рівні. Багато з таких версій застарілі технологічно та не враховують сучасних вимог в організації освітнього процесу. Негативно впливає і той факт, що програми від різних розробників не можуть ефективно обмінюватись даними між собою.

Зауважимо, що піднята проблема не є новою для європейських та українських навчальних закладів. У законодавчих актах України прописано необхідність впровадження інформаційних та комунікаційних технологій у систему вищої освіти. (Наказ Президента України від 20.10.2005 р. № 1497/2005 “Про першочергові завдання щодо впровадження новітніх інформаційних технологій”, Закон України №75/98-ВР “Про Концепцію Національної програми інформатизації”, Закон України “Про Національну програму інформатизації”; Розпорядження Кабінету Міністрів України від 21.09.2011 р. № 1036-р “Про затвердження плану заходів щодо забезпечення розвитку освіти у сфері інформаційних технологій на період до 2013 року”. В Україні постановою Кабінету міністрів № 752 від 13.07.2011 р. “Про створення Єдиної державної електронної бази з питань освіти” було запроваджено Єдину державну електронну базу з питань освіти (ЄДЕБО), автоматизовану систему накопичення, обробки, зберігання та захисту даних, у тім числі персональних, щодо закладів, які надають освітні послуги. Певні напрацювання мають такі науковці: Л.І. Даниленко, Г.В. Єльникова, В.І. Маслова, В.Ю. Бикова, В.В. Олійник, В.Д. Руденко. І.І. Мусієнко, Г.Ю. Козак, Г.В. Кузнецов, О. Кириченко, Ю. Вигівська.

Однак реальних ефективних, спрямованих на універсальність рішень управлінських дій щодо модернізації та вирішення проблем побудови інформаційних систем управління навчальним процесом розрахованих на широке використання, не відбувалось. Українські навчальні заклади вже намагалися створити аналоги подібної автоматизованої системи управління навчальним процесом: "Деканат", "Сократ", "Автоматизована система управління вищим навчальним закладом", проте, ці системи адаптовані лише для вирішення деяких завдань, які стосуються навчального процесу певного університету.

Європейські системи управління навчальним процесом теж мають багато недоліків. Отож, у підсумку існує потреба у детальному розгляді питань, які пов'язані з розробкою інформаційної системи управління навчальним процесом, що сприятиме підвищенню якості навчання, демократизації управління навчальним закладом, зниження витрат на організацію і управління навчальним процесом, створення регіональних і міжвузівських інформаційних систем, що забезпечують інтенсивніший обмін інформаційними ресурсами в регіоні та галузі на міжнародному рівні, що значно спрощує процес міжнародної академічної мобільності.

Позитивним кроком на шляху дослідження інформаційних систем управління університетом можна вважати результати проекту INURE: 530181-TEMPUS-1-2012-1-DE-TEMPUS-SMGR, метою якого було створення методології розробки інтегрованої інформаційної системи управління університетом для ВНЗ України. Проект "INURE: Інтегрована інформаційна система управління університетом: впровадження досвіду ЄС в країнах СНД" об'єднав зусилля університетів та Міністерства освіти України, Республіки Молдова, Грузії, Республіки Білорусь та європейських партнерів щодо розробки сучасних моделей побудови інтегрованих систем управління університетами і був спрямований передусім на адаптацію сучасної європейської методики управління університетом в ЄС з використанням новітніх комп'ютерних та управлінських технологій. Кінцеві результати проекту:

– створення мережі центрів сучасних технологій інформаційного менеджменту в університетах-учасниках проекту. Центри використовують для вивчення сучасних технологій розробки інтегрованих інформаційних систем управління та досвіду впровадження таких систем у вищих навчальних закладах Європи. Мережа центрів дає змогу надалі розвивати цей напрям, підвищувати кваліфікації адміністративного персоналу університетів та забезпечувати обмін досвідом.

– розроблено методичні основи проекту "Методологічні основи створення, впровадження та розвитку Інтегрованої Інформаційної Системи Управління Університетом" [1].

Важливим результатом цього проекту є узагальнення досвіду європейських університетів з використанням процесного підходу на підставі аналізу бізнес-процесів університету. У розробленій методології сформульовано загальне бачення структури інформаційної системи управління університетом і сформовано вимоги до кожної з підсистем та рекомендації щодо їхнього створення та впровадження. Деякі з підсистем рекомендовано реалізовувати на базі готових рішень, наприклад фінансові, бібліотечні, управління контентом навчальної діяльності. Щодо підсистеми управління навчальним процесом, то готових рішень не пропонується і рекомендації містять лише абстракті підходи до розробки бази даних, політики ролей користувачів і т. п. Мета нашої праці – ліквідувати цю прогалину і сформувати концептуальні підходи до організації розробки інформаційної системи управління навчальним процесом.

2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Проектуючи систему, основною функцією якої є автоматизація бізнес-процесів університету, треба розуміти роль бази даних у цій системі. Фактично, вона міститиме всю інформацію, яка зберігається в паперовому вигляді. Тому її створення є дуже важливим завданням, а саме проектування визначатиме подальшу якість розробки програмного продукту. Успішне проектування передбачає нормалізовану базу даних, з якою буде зручно працювати.

Важливий момент у проектуванні бази даних – можливість стороннім клієнтам підключатись безпосередньо до сховища даних, виконувати потрібні операції і отримувати певні результати. Наприклад, отримавши доступ до бази даних, користувач матиме змогу одержати інформацію про всі відділи, які існують у системі.

На концептуальному етапі проектування вирішено умовно поділити всі таблиці на чотири типи:

- документи;
- деталі документів;
- об’єкти;
- властивості об’єктів;
- словники;
- технічні таблиці.

Детально розглядаючи кожен тип таблиць, отримаємо. Документи – це таблиці, в яких міститься різноманітна інформація, наприклад, про накази, заяви тощо. Іншими словами, такі таблиці зберігають інформацію про всі паперові документи. До них належать таблиці про звичайні накази, про заяви на вступ до університету, накази на зарахування, прийому на роботу і т. д. Цей тип таблиць було прийнято позначати префіксом “dc” (documents).

Цілком логічно, що разом із документами існує й детальна інформація про них. Отож, описуючи такий вид документів, як заява абітурієнта на вступ до університету, система обов’язково повинна містити таблицю зі статусом такої заяви. Варто зауважити, що кількість таблиць, які можна зачислити до деталей документів, є значно більшою від кількості таблиць-документів. Це зумовлено насамперед складною організацією даних. Наприклад, щоб подати заяву на вступ, треба мати дані про кафедру, конкурсний відбір, потрібні предмети, загальний прохідний бал тощо. Такі таблиці позначають у системі префіксом “dt”.

Об’єкти – такий тип таблиць, який описує фізичні одиниці, а також їхню взаємодію. Скажімо, це може бути таблиця з активами, яка описує все існуюче майно університету, чи таблиця всіх осіб, які колись вчились чи працювали в університеті. Також це може бути таблиця з навчальними дисциплінами, які існують в університеті. Такий тип таблиць позначається префіксом “ob” (objects).

До словників було віднесено інформацію, яка є сталою і дуже рідко змінюється. Прикладом такої інформації може слугувати таблиця про види сімейного статусу чи наукового звання. Варто зауважити, що всі дані для словників було взято з ЄДЄБО (Єдиної Державної Електронної Бази Освіти).

Під технічними таблицями ми розуміємо ті, які використовують для технічних цілей. Наприклад, у базі даних існують таблиці, які зберігають інформацію про права на читання, запис і модифікацію певних таблиць.

2.1. ІНТЕГРАЦІЯ З ЗОВНІШНІМИ СИСТЕМАМИ

Розробляючи і проектуючи систему для внутрішнього використання в університеті, дуже важливо мати змогу інтегрувати її зі сторонніми програмними продуктами. Великий обсяг накопиченої інформації та висока швидкість надходження нової спричиняють підвищення вимог до сучасних інформаційних систем, у тім числі систем, призначених для інформаційного забезпечення наукових і виробничих процесів [2]. Тому важливо мати єдине сховище даних. Так, проектуючи бази даних для системи "Абітурієнт", яка є важливою частиною організації вступної кампанії в університеті, треба передбачити можливість інтеграції внутрішньої бази даних університету до Єдиної державної електронної бази даних з питань освіти (ЄДЕБО). Постановою Кабінету Міністрів від 13 липня 2011 р. в Україні було оголошено створення ЄДЕБО, автоматизованої системи накопичення, оброблення, зберігання та захисту даних, у тім числі й персональних, щодо освітніх закладів в Україні.

У 2015 р. відповідно до пункту 6.10 Умов прийому до вищих навчальних закладів України, затверджених наказом Міністерства освіти і науки, факт кожної подачі заяви в паперовій формі заноситься уповноваженою особою приймальної комісії до ЄДЕБО безпосередньо під час прийняття заяви абітурієнта до вступу у вищий навчальний заклад [3]. З цього випливає, що всі внутрішні дані університету мають бути занесені до ЄДЄБО. Така необхідність зобов'язує дублювати дані, оскільки оператору треба внести їх в ЄДЄБО, а також і у внутрішню систему. Саме тому проектована система має передбачати спрощення за рахунок інтеграції внутрішньої бази даних з ЄДЕБО.

2.2. СИНХРОНІЗАЦІЯ ІНФОРМАЦІЇ ЗІ СТОРОННІМИ ДЖЕРЕЛАМИ ДАНИХ

Згідно з визначенням синхронізація – це ліквідація відмінності між двома джерелами даних, приведення їх до найновішого стану. Інформаційна система, якою послуговується університет, зобов'язана передавати значну частину інформації в ЄДЕБО. З іншого боку, ЄДЕБО може містити ту інформацію, якої бракує в університетській системі. Саме тому надзвичайно важливо забезпечити процес синхронізації даних між двома цими системами. Зауважимо, що ЄДЄБО як загальна система для всіх університетів, має містити дані зі всіх навчальних закладів, визначивши їх домінуючим джерелом інформації. Відповідно, всі дані, які потрапляють у систему ЄДЄБО, можуть бути збережені та синхронізовані в інформаційній системі університету.

Сама синхронізація відіграє дуже важливу роль у швидкій роботі інформаційної системи, оскільки допомагає скоротити час, який потрібний для роботи з інформацією. Наприклад, певний студент подає заяву на вступ до університету. Перед цим студента треба зареєструвати в системі ЄДЄБО. Якщо ж студент вже існує в системі ЄДЄБО, то реєструвати його не потрібно, варто лише зазначити унікальне значення, яке відповідає цьому студенту. Університет, маючи найактуальніші дані, попередньо провів синхронізацію і вже має інформацію про студента, його унікальні значення, а також інші дані, потрібні для подачі заяви на вступ. Після цього університету потрібно лише подати заяву. Варто зауважити, що у випадку браку даних про студента в інформаційній системі університету, треба виконати додаткові операції, аби визначити, чи існує така особа в системі ЄДЄБО.

3. ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

Коли йдеться про розробку великих програмних продуктів – таких як інформаційна система університету – неможливо не врахувати великі обсяги даних. Сьогодні в інформаційній системі університету існує понад півтори сотні таблиць у базі даних, з якими потрібно проводити всілякі можливі операції. Відповідно, маючи таку кількість таблиць, архітектуру цього програмного забезпечення треба розробляти відповідно до всіх правил і норм проектування. Важливо проектувати її так, щоб можна було легко масштабувати і у разі потреби змінювати певну логіку.

Сьогодні існує два основних типи проектування архітектури великих програмних продуктів – монолітний і мультисервісний [4]. У програмній реалізації системи пропонується монолітна архітектура у зв'язку з дефіцитом людських ресурсів. Неможливість використання мультисервісної архітектури пояснюється браком людських ресурсів, які б супроводжували сервіси. Можливість легко змінювати логіку програмного забезпечення – важливий момент у системах на кшталт "Абітурієнт", оскільки кожного року вимоги до подачі документів й обчислення рейтингу змінюються, тож потрібно мати змогу легко відображати ці зміни в програмному продукті.

В теорії проектування програмного забезпечення побутує думка, що код тоді якісний, коли в нього низька зв'язність між програмними компонентами і висока згуртованість всередині. Тому інформаційна система – це сукупність модульних програмних елементів, поділених за функціональними особливостями. Наприклад, існує модуль, основна функція якого забезпечити зручний доступ до бази даних, а також існує модуль, який призначений для отримання запитів від користувачів і передачу відповідей. Такий підхід до проектування дає змогу у майбутньому легко перевикористовувати вже готові модулі. Розробляючи систему "Абітурієнт", потрібен модуль для роботи з базою даних, який ми і реалізуємо. Згодом, розробляючи систему управління навчальним процесом, знову знадобиться модуль для роботи з базою даних. Проте цього разу вже не потрібно його реалізувати, а можна використати вже існуючий. Система для керування університетом передбачає декілька підсистем, отже, перевикористання модулів дуже доцільне, оскільки допоможе зекономити ресурси для розробки програмного забезпечення.

REST (Representational state transfer) – це стиль архітектури програмного забезпечення для розподілених систем: World Wide Web, який зазвичай використовують для побудови веб-сервісів [5]. Загалом REST виявляється дуже простим інтерфейсом управління інформацією без використання будь-яких додаткових прошарків. Кожну одиницю інформації однозначно визначає глобальний ідентифікатор – URL. Кожна URL має строго заданий формат.

Відсутність додаткових прошарків означає передачу даних у тому вигляді, в якому вони надійшли, тобто ми не змінюємо сам запит в процесі проходження запиту по серверу, як це робить SOAP і XML-RPC, AMF. Кожна одиниця інформації однозначно визначається за її URL, а це означає, що URL, по суті, є первинним ключем для одиниці даних. Наприклад, третя книга з книжкової полички буде мати URL /api/books/3 , а 35 сторінка в цій книзі – /api/books/3/page/35. З цього випливає строго заданий формат. Абсолютно не має значення, в якому форматі надходять дані по URL – /api/books/3/page/35 – це може бути і HTML, і відсканована копія jpeg-файла,і документ Microsoft Word.

Також варто зазначати, що часто виникає необхідність повертати дані в різних форматах (json, xml). Саме для цього існує поняття “MessageConverter” – патерн, який аналізує заголовок HTTP-запиту Content-Type і, залежно від його значення, віддає значення в потрібному форматі. Такий патерн має певні обмеження, оскільки він потребує на вході отримувати об’єкти певного типу.

Управління інформацією сервісу засновано на протоколі передачі даних. Найбільш розповсюджений протокол, звичайно, HTTP. Так-от, для HTTP дії над даними задають за допомогою HTTP методів:

- GET(отримати сутність);
- PUT(оновити сутність);
- POST(додати сутність);
- DELETE(видалити сутність).

Отож, ми маємо CRUD (Create - Read - Update – Delete набір операцій, і REST полягає в тому, що, маючи у веб-сервісі певну сутність, ми зобов’язуємось виконати повністю всі чотири методи, в іншому випадку веб-сервіс не буде RESTful.

Архітектура в стилі REST складається з клієнтів і серверів. Клієнти ініціюють запити до серверів, а сервери обробляють запити й повертають належні відповіді.

ORM (Object Relational Mapping) – технологія програмування, яка зв’язує бази даних із концепціями об’єктно-орієнтованих мов програмування, створюючи “віртуальну об’єктну базу даних” [6]. Використовуючи один з найбільших стовпів розробки сучасного програмного забезпечення – об’єктно орієнтоване програмування, не можна обійти увагою інший світ – світ реляційних баз даних.

Саме реляційним Системам управління Базами Даних вдалося в 1980 роках звільнити програмістів від непотрібних деталей в організації фізичного зберігання даних, відгородившись від них структурами логічного рівня та стандартизованою мовою SQL для доступу до інформації. Також виявилося, що більшість форматів даних, якими оперують програми, добре лягають на модель двовимірних таблиць і зв’язків між ними. Ці два чинники визначили успіх реляційних СУБД, а як заохочувальну премію спільнота отримала сувору математичну теорію.

На відміну від реляційного світу, ООП розвивали інженери-практики досить стихійно, враховуючи потреби програмістів, саме тому ніякої суворої теорії не було. І реляційна, і об’єктна моделі належать до логічного рівня проектування програмної системи. Вони ортогональні і фактично відображають два погляди на ту саму сутність. Це означає, що можна реалізувати ту саму систему, залишаючись в рамках тільки одного реляційно-процедурного підходу, або ж дотримуючись виключно ООП. Саме тому, використовуючи об’єктно-реляційний підхід до побудови програмного забезпечення в університеті, зручно мати під рукою інструмент, який би виконував всю рутинну роботу з sql-запитами.

Вибрали ORM фреймворк Hibernate (http://hibernate.org/), оскільки він містить вбудовану підтримку на рівні мови програмування Java, добре підтримується і є багатофункціональним.

4. ЗАБЕЗПЕЧЕННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. СИСТЕМА КОНТРОЛЮ ВЕРСІЙ

У контексті розробки програмних систем керування версіями можна розглядати як мистецтво управління паралельними змінами. Зазвичай системи

керування версіями (англ. source code management) працюють як центральні сховища (repository), мають інтерфейс для доступу та спільного використання коду.

Наприклад, якщо двом розробникам треба працювати з тим самим файлом, то вони можуть відкрити і редагувати цей файл, перезаписувати, витираючи зміни, які виконали. Або ж один розробник очікує, поки інший завершить редагування файла. Система керування версіями є ефективним рішенням цієї проблеми. Вона здатна стежити за змінами кожного з файлів сховища, і користувач може бачити не тільки поточний стан файла, а й усі зміни, які відбувались за час його існування. Завдяки цьому можна виконувати відкат (rollback) помилкового коду, забезпечуючи повернення до робочої версії. Певні версії файла можна позначати як кінцеві і, продовжуючи розробку, завжди мати доступ до копій версії, які на цей момент вважають кінцевими.

Окрім того, системи керування версіями дають змогу декільком програмістам одночасно працювати над однією частиною коду. Кожен із них може отримати (checkout) копію коду зі сховища, а після завершення роботи помістити (commit) його назад. У цьому випадку прийнято вважати, що версія прийнята. Звісно ж, дані системи можуть простежити, які зміни використовував користувач.

Існує три класи системи керування версіями:

- локальні (rcs) – зберігають зміни на тому ж комп'ютері, на якому й відбувається робота з файлами;
- централізовані (CVS, Subversion, Perforce) – із сервера копіюють лише ті файли, які треба;
- розподілені (Git, Mercurial, Bazaar) – кожен користувач повністю копіює все сховище, тому більшість операцій із ним відбуваються локально. Це пришвидшує роботу і дає змогу швидко відновити головне сховище у разі збоїв.

Звичайно, найкращим класом є останній, адже працюючи з розподіленими системами, розробник може тримати локальну копію всіх змін у себе на робочій машині і відправляти ці зміни на центральне сховище лише наприкінці. Для розробки інформаційної системи управління навчальним процесом університету вибрано систему контролю версій розподіленого класу, оскільки вона допомагає гнучко вести розробку програмного продукту. Зокрема функція "гілкування" дає змогу тримати в системі різні версії програмного продукту. Якщо наш програмний продукт існує в трьох версіях: 1.0 , 1.1 і 1.2 і в найближчий час очікується нова версія – 1.3. Всі версії цього програмного продукту тримаються в паралельних гілках з відповідною назвою. Випускаючи нову версію програмного продукту, існує вірогідність некоректної роботи. Тому готуючи нову версію, в разі несправності якогось функціонала, завжди можна використовувати попередню версію, яка за замовчуванням має працювати злагоджено і не містити помилок.

4.2. АВТОМАТИЗАЦІЯ ПРОЦЕСІВ РОЗРОБКИ

Автоматизація розробки – це процес скриптування або автоматизації широкого набору завдань, які розробники програмного забезпечення виконують повсякденно. Серед них: компіляція коду, пакування коду, виконання автоматизованих тестів, розгортання програмного забезпечення, створення документації.

Останнім часом технологія побудови програмного забезпечення дає змогу значно спростити роботу. Для отримання такого спрощення реалізовано широкий

набір платного та безплатного програмного забезпечення. Його часто використовують для continuous integration – побудови проекту незалежно від розробника. В такій практиці процес побудови проекту може розпочинатись з потрапляння нового функціонала до систем керування версіями, або ж може працювати за графіком.

Серед плюсів можна виділити:

- поліпшення якості програмного забезпечення, оскільки разом із побудовою програмного продукту є змога виконувати перевірки якості;
- економія часу завдяки зменшенню надлишкової роботи.

4.3. МІГРАЦІЇ БАЗ ДАНИХ

Для розробки програмного продукту існують два варіанти роботи з базою даних: централізований, де існує центральна база даних, за якого працюють всі розробники, а також розподілений варіант, де кожен розробник програмного продукту має локальну копію бази даних і веде роботу безпосередньо з нею. Варто зауважити, що при розробці зручнішим є останній варіант, адже він допомагає уникнути колізій при зміні даних, розробнику не потрібно переживати, що певна інформація може бути видалена чи змінена. Однак при розподіленому варіанті, якщо один розробник вносить зміни до бази даних і до програмного продукту, то іншому надалі знадобляться ці зміни. Системи міграції даних – це підхід, який дає змогу тримати всі налаштування бази даних у погоджених файлах. Всі зміни до бази зберігаються в певних файлах. Розробник, вносячи зміни до бази даних, заносить їх і до міграційного файла, а також відправляє до системи контролю версій. Інші розробники, отримавши ці зміни з системи контролю версій, запускають систему міграції даних, яка додає всі потрібні зміни до локальних копій баз даних. Такий підхід дає змогу спростити розробку програмного забезпечення, зробити процес розгортання програмного продукту гнучкішим.

Варто зауважити, що всі зміни тримаються лише в погоджених файлах, це сприймається як правило. Отож, розгортаючи програмний продукт на сервері, і маючи новостворену базу даних, потрібно лише запустити систему міграції бази, після чого з’явиться загальна база даних зі всією структурою, яка необхідна для коректної роботи.

Ще одна вагома функція в системах міграції даних – можливість повертатися до різних версій бази. Наприклад, інформаційна система, яку розробляють для потреб університету, містить декілька версій, кожна з яких працює з різними версіями бази даних. Тому для найстарішого варіанту інформаційної системи потрібно задати найстарішу версію бази даних. Отже, заради такого процесу в системах міграції бази даних була реалізована можливість повертатись до різних її версій.

5. ТЕСТУВАННЯ СИСТЕМИ

Тестування програмного забезпечення, яке використовують для виміру якості розроблюваного програмного забезпечення, зазвичай обмежується такими поняттями: коректність, повнота та безпечність. Але воно також може містити більше технічних вимог, які описані в стандарті ISO 9126 : виявлення інформації про якість продукту стосовно контексту, в якому його треба використовувати. Розробляючи програмне забезпечення для навчального закладу, особливу увагу треба приділяти належному тестуванню. Адже у зв’язку з тією кількістю функціонала, яку потрібно

реалізувати, варто завжди бути впевненим в якості програмного забезпечення. У цьому випадку варто використовувати всі технології тестування.

Модульне тестування полягає в ізольованій перевірці кожного окремого елемента запускаючи тести у штучному середовищі. Для цього потрібно використовувати драйвери та заглушки. Поелементне тестування – найперша змога реалізувати вихідний код. Оцінюючи кожен елемент ізольовано і підтверджуючи коректність його роботи, точно визначивши проблему, значно простіше, ніж, наприклад, якби елемент був частиною системи.

Для інформаційної системи, яку розробляють для потреб університету, дуже важливо бути впевненим у стабільності коду, тому модульне тестування є важливим процесом, який відбувається під час розробки програмного забезпечення.

Інтеграційне тестування – використовують для тестування окремих модулів програм у взаємодії. Інтеграційне тестування виконують після модульного тестування та перед верифікацією та валідацією ПЗ. Якщо розглядати цей процес як систему, то на вході їй надають модулі, які вже пройшли модульне тестування; потім ці модулі групують у більші частини, а далі виконують тести передбачені планом.

Варто зауважити, що запропонований підхід з розробки інформаційної системи для університету дуже зручний для інтеграційного тестування. Нагадаємо, що кожна сутність, яка є в інформаційній системі (заява, деканат, особа), містить чотири базові операції: створення, редагування, читання та видалення. Отже, є змога писати інтеграційні скрипти, проходячи весь цикл. Наприклад, можна створювати довільну сутність, редагувати, читати і видаляти, паралельно перевіряючи успішність операцій.

Список використаної літератури

1. *Чернишенко С.В*. Методологічні основи створення, впровадження і розвитку інтегрованої інформаційної системи управління університетом / С.В. Чернишенко, Ю.І. Воротницький, В.М. Кухарський. – Суми : Сумський державний університет, 2015. – 333 с.
2. *Глоба Л.С*. Онтологія для побудови порталу Єдиної інформаційної системи ВНЗ / Л.С. Глоба, В.В. Кирилков // Вісник Харківського національного університету № 1037. – 2012. – С. 28-42.
3. Наказ Міністерства освіти і науки України “Про затвердження Умов прийому на навчання до вищих навчальних закладів України в 2015 році”. [Електронний ресурс] // Режим доступу: http://zakon4.rada.gov.ua/laws/show/z1390-14#n15
4. Microservice architecture: [Електронний ресурс] // Режим доступу: http://microservices.io/
5. DL.ACM: [Електронний ресурс] // Digital library. – Режим доступу: http://dl.acm.org/citation.cfm?id=932295
6. Object-relational mapping: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/Object-relational_mapping

Стаття: надійшла до редколегії 11.03.2015
доопрацьована 15.04.2015
прийнята до друку 29.04.2015

THE TECHNOLOGY OF SOFTWARE DEVELOPMENT FOR TYPICAL SUBSYSTEM OF UNIVERSITY INTEGRATED MANAGEMENT INFORMATION SYSTEM

V. Kukharskyy, I. Ursul

Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000, e-mail: vitaliy.kukharskyy@lnu.edu.ua

The process of developing software for educational activity management subsystems - one of the major subsystems of standard integrated information systems of the university management is considered. Taking into account all the peculiarities of business processes of the university, a database design has been conducted which provides integration and synchronization with external systems; the architecture of the future of software has been defined; the concept of providing software development process has been formed by using version control technologies, process automation, database migration, testing system, practically forming the ecosystem for future projects (https://github.com/ifnul).

Key words: an integrated university management information system, software , database, system architecture , software development , testing.