УДК 681.324

Корень О. М.

ОДИН ПІДХІД ДО ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ ПІДТРИМКИ ЕЛЕКТРОННОЇ ОСВІТИ

У роботі розглянуто базові принципи побудови програмних систем підтримки електронної освіти (ПСПЕО) як розподіленої системи на базі трирівневої клієнт-серверної архітектури. У першій частині проведено загальний аналіз ПСПЕО, розгляд можливих платформ, архітектур їхньої побудови. Далі сформульовано конкретні вимоги до системи та надано їхню орієнтовну реалізацію. Розглянуто суб'єкти та об'єкти дистанційного навчального процесу, їхня взаємодія та протоколи роботи. Визначено інтерфейси взаємодії між окремими об'єктами та базами даних.

Ключові слова: електронна освіта, розподілена система на базі трирівневої клієнт-серверної архітектури.

Вступ

Електронна освіта на сьогодні – важлива та невід'ємна частина навчального процесу в багатьох провідних університетах світу. Ефективна її реалізація насамперед залежить від розвитку засобів дистанційного доступу до продуктів інформаційного забезпечення навчального процесу, тобто комунікаційних технологій і засобів роботи з розподіленими базами зберігання знань. Потреба у такій системі дистанційної освіти визріла і в багатьох вищих навчальних закладах в Україні [1, 2].

Зрозуміло, що впровадження Internet у навчальні заклади позначилося на методиці викладання різних дисциплін. Internet – це не лише засіб отримання додаткової інформації. Сучасні технології дали можливість організувати так зване електронне, або змішане (колаборативне) навчання.

Система дистанційної освіти – це програмний комплекс корпоративного типу. Тобто вимоги до такого комплексу мають бути найвищими, а процес розробки та тестування максимально глибокими та повними. Звичайно, описати розробку такої системи в рамках однієї статті досить важко, якщо взагалі можливо. Тому нашим завданням є опис розробленої підсистеми керування.

Загальні вимоги до великих систем подібного типу поділяються на дві категорії: вимоги до архітектури системи та вимоги до реалізації системи [3].

Будь-яке застосування корпоративного типу перед початком масштабної реалізації потребує побудови моделі. ПСПЕО повинна будуватися таким чином, щоб можна було легко змінювати реалізовані частини системи та додавати нові частини (як окремі функції, так і цілі підсистеми) без повної переробки системи. Зрозуміло, що для можливості ефективної обробки вимог, які постійно змінюються, необхідно постійно синхронізувати систему та її модель.

Необхідною вимогою при побудові великих та ефективних корпоративних систем є повторне використання коду. Повторне використання коду дозволить встановити та постійно вдосконалювати найкращі практики реалізацій, які будуть добре відомі всім учасниками розробки системи.

Кожен клас (або об'єкт) повинен мати чітко визначену роль в ПСПЕО. Це допоможе ефективно забезпечити повторне використання коду, а також чітко організувати процес розробки системи, розподіливши його між максимально можливою кількістю розробників. Яскравим прикладом розділення завдань за набором умінь може слугувати відокремлення програмного коду та графічного дизайну при розробці клієнтських інтерфейсів, що є частиною системи і мають зв'язок із нею по мережі.

Функціональність ПСПЕО повинна постійно відповідати потребам університету, які неодмінно будуть змінюватися. Це означає, що (як було зазначено вище) потрібно забезпечити простий процес додання нової та зміни існуючої функціональності. Лише в цьому випадку буде виконуватися вимога розширюваності системи.

Розбиття структури системи на модулі, які взаємодіють між собою за добре визначеними й описаними інтерфейсами, дозволить розробникам працювати незалежно один від одного. Таке розбиття різко підвищує контрольованість процесу розробки, дозволяє легко тестувати окремі модулі системи, знаходячи помилки та підвищуючи якість реалізації системи.

Система повинна мати можливість гнучкого налаштування. Таке налаштування має звести до мінімуму необхідність втручання в код компо-

нентів системи при невеликих змінах вимог до системи. Такі зміни вимог не повинні викликати змін у коді окремих компонентів. У цьому випадку відповідальна особа змінює налаштування певних підсистем, конфігуруючи необхідні зміни. В той же час інші змінені вимоги до системи можуть призводити до змін у коді. Наприклад, виникла необхідність підтримання кількох мов в інтерфейсах системи (для закордонних студентів), причому користувачі самі обирають для себе мову інтерфейсу. В цьому випадку необхідно вносити зміни у код компонентів системи, розширюючи їхні можливості з підтримки багатомовності, розширювати їхні інтерфейси тощо. Для забезпечення можливості гнучкого налаштування системи, необхідно глибоко продумати архітектурну модель системи, обираючи між багатими можливостями налаштувань і кількістю додаткового коду, який необхідний для цього.

Оскільки система буде підтримувати найрізноманітніші дані, більша частина яких може бути доступна тільки певній категорії користувачів, то є природнім вимагати від ПСПЕО реалізації певних заходів для безпечної обробки даних. До таких заходів належить забезпечення «відокремленості» даних, що відносяться до різних користувачів системи – кожен з них може бачити та редагувати тільки те, що йому дозволено. До цієї категорії також належить можливість гнучкого регулювання дозволами (дані можна групувати за різними критеріями і для кожної такої групи задавати відповідні дозволи), делегація прав користувачів між собою. Кожен, хто має повний доступ до певного об'єкту даних, може керувати правами інших користувачів до цього ж об'єкта. Наприклад, методист кафедри може дати права на редагування певною інформацією своєму помічникові. ПСПЕО повинна надавати можливість включення ведення історії усіх змін та доступу до певної групи даних. Наприклад, історія роботи з електронним підручником або історія входів у систему (тобто доступу до певних методів компоненту аутентифікації).

Сучасні вимоги до розподілених систем корпоративного типу також включають спосіб реалізації системи. До таких вимог відносяться: гетерогенність системи, можливість керування навантаженням серверів і ресурсами, збереження стану та засоби пошуку серверних об'єктів, взаємодія в контексті транзакцій, безпека інформації, історія взаємодії клієнтів і серверів, стандартний вигляд клієнтських інтерфейсів, а також можливість існування кількох їхніх типів. Звичайно, не всі ці вимоги повинні обов'язково бути імплементовані у кожній конкретній ПСПЕО, пілотні чи спрощені проекти можуть не мати засобів пошуку серверних об'єктів. Проте ці вимоги бажані.

1. Огляд сучасних технологій для застосування в галузі ПСПЕО

Сучасні технології для корпоративних програмних комплексів та розподілених баз даних сягнули дуже далеко з часів перших комерційних застосувань. Навіть короткий огляд цих технологій – це тема окремої праці, а враховуючи динаміку їхнього розвитку – навіть низки таких праць. Тому наведемо тут лише короткий огляд та аналіз.

Усі відомі інтернет-проекти починалися із застосування CGI – Common Gateway Interface. CGI є старим, перевіреним і універсальним способом створення розподілених систем в Інтернеті. Основне його завдання – забезпечення викликів віддалених процедур між клієнтом та сервером із використанням сокетів TCP/IP. Проте технологія CGI сьогодні вважається неефективною, оскільки організація взаємодії клієнта й застосування-сервера є досить складною. Крім того, існує обмеження сесії, дуже обмежена гнучкість таких програм та ін. Тому від цієї простої технології доведеться відмовитися.

Java Server Pages (JSP) є розширенням концепції CGI та узагальненням концепції сервлетів. Розглянемо прилад. Необхідно побудувати HTML-сторінку, частина якої є постійною (статичною), а інша частина залежить від даних, що зберігаються на сервері. При використанні CGI для формування динамічної частини сторінки клієнт повинен формувати додаткові запити до сервера, використовуючи виклики CGI/ISAPI, – статична частина коду сторінки, що завантажилася, може містити код, виконуваний клієнтом. Клієнт, виконуючи цей код, здійснює додаткові виклики до сервера, самостійно будуючи динамічну частину сторінки. Виконуваним кодом на клієнті може бути аплет, або JavaScript. Технологія JSP дозволяє перенести код, що формує динамічну частину сторінки на сервер, звільняючи клієнта від додаткових дій і максимально переносячи навантаження на сервер. На сервері зберігається спеціальний документ, що містить як HTML-теги, так і виконуваний код, який динамічно формує той самий HTML, але у момент виклику. Виконуваним кодом є команди Java, скрипти та компоненти Java Beans/Enterprise Java Beans.

Сторінки JSP перебуває під управлінням так званого контейнера JSP (JSP Container), який забезпечує передачу запита від клієнта й повернення відповіді. Контейнер зобов'язаний підтримувати протокол HTTP, але при цьому може додатково підтримувати й інші протоколи. Сторінки JSP можуть підтримуватись контейнером як у вигляді вихідного тексту, так і у відкомпільованому вигляді. Результат також може бути

сформований у будь-якому вигляді – специфікація не вимагає обов'язкового використання HTML. Найбільш ефективним форматом відповіді, мабуть, можна вважати формат XML.

Існує багато інших альтернативних технологій, серед яких найбільш популярні ASP (розробка Microsoft), PHP, ColdFusion та інші. Всі вони добре розвинені й активно підтримуються своїми розробниками, мають механізми під'єднання до баз даних та багато інших розширень. Немає принципових відмінностей між усіма цими технологіями. Поняття контейнера використовується тільки в JSP, хоча в усіх інших технологіях є частини серверу, які виконують ті самі функції. Контейнер використовується і в інших Java-технологіях, які об'єднані специфікацією J2EE (Java 2 Enterprise Edition). JSP органічно вливається в цю концепцію, що дозволяє ефективно використовувати JSP поруч з іншими корпоративними технологіями, сумісними з J2EE.

Технологія DCOM – одна із сучасних, об'єктно-орієнтованих технологій створення розподілених систем. Сьогодні ця технологія широко використовується поруч із технологіями RMI та CORBA, про які мова піде нижче. DCOM має всі необхідні частини – брокер об'єктних запитів (англ. ORB – Object Request Broker), підтримку статичних і динамічних викликів віддалених методів, мову опису інтерфейсів, свою компонентну модель (VBX/OCX/ActiveX), базу даних інформації про об'єкти (бібліотеки типів), об'єктний монітор транзакцій (Microsoft Transaction Server), стандарт створення складних документів та багато інших сервісів. Компонентна модель довела свою придатність для створення достатньо складних застосувань (наприклад, Microsoft Internet Explorer, починаючи з 5-ї версії повністю побудований з компонентів ActiveX). Усі продукти Microsoft побудовані за COM технологією.

В основу COM (Component Object Model) покладено двійкову структуру об'єктів. Це забезпечило незалежність від мов програмування – для роботи з COM можна використовувати C++, Visual Basic, Visual J++ та Borland Delphi/C++. Об'єкт COM є прикладом класу, що реалізує набір методів, які складають доступні для клієнта інтерфейси.

DCOM визначає три види серверів: in-process (DLL, що знаходиться в адресному просторі клієнта), local (EXE-застосування, що запускається на тій самій машині, що і клієнт) та remote (EXE-застосування, що працюють на віддаленому сервері). В будь-якому випадку серверні об'єкти створюються за допомогою фабрик об'єктів. Фабрики об'єктів у технології COM – це об'єкти, що реалізують стандартні інтерфейси IClassFactory та IClassFactory2.

Технологія COM передбачає власну компонентну модель, що зветься ActiveX. Контейнер, що містить компоненти зветься ActiveLibrary, складний документ – ActiveDoc. ActiveX – це просто COM-об'єкт, який реалізує кілька стандартних інтерфейсів.

Складна компонентна модель повинна передбачати засоби управління компонентами на боці сервера, інакше кажучи, наявність монітора транзакцій. Для DCOM – це Microsoft Transaction Server (MTS). Він забезпечує потрібну функціональність – керування транзакціями, створення пула ресурсів (у тому числі з'єднань із базами даних), створення та керування пулом серверних об'єктів, активацію та деактивацію серверних об'єктів із метою оптимізації ресурсів сервера.

Технологія CORBA – продукт сумісних зусиль дуже великої кількості виробників програмного забезпечення. У зв'язку з цим було навіть створено спеціальну технологію розробки, обговорення та прийняття рішень. Офіційна назва консорціуму учасників цього процесу – OMG (Object Management Group). Це некомерційна організація, яка займається організаційними питаннями в розробці технології CORBA.

CORBA – це стандарт створення розподілених систем. Універсальним вважається рішення, яке не залежить від мови програмування, апаратної платформи, операційної системи, конкретного мережевого протоколу, будь-яких двійкових стандартів та структур. Для опису специфікацій CORBA було створено спеціальну мову – OMG IDL (Interface Definition Language). Універсальність технології, головним чином, забезпечується саме завдяки базуванню виключно на IDL та використанням стандартів відображення IDL-декларацій на конкретні мови програмування.

Наразі існує тісна взаємодія CORBA із RMI, EJB та всіма іншими Java-орієнтованими технологіями.

Основним недоліком CORBA є відставання реалізації від стрімкого розвитку специфікацій – більшість останніх специфікацій ще не реалізована. Незаперечною перевагою останньої є концептуальність. Якщо якась технологія, що використовується для створення розподіленої системи, є сумісною з CORBA, вона має всі шанси на «довге життя» – всі сучасні технології намагаються забезпечити сумісність із CORBA, що буде підтримуватися і в майбутньому.

Технологія RMI (Remote Method Invocation) – це також сучасна, потужна та ефективна технологія створення розподілених систем. RMI вимагає використання мови Java. Основна задача, яку розв'язує RMI, – забезпечення передачі повідомлення від об'єкта, що існує в контексті однієї віртуальної машини Java (JVM), до об'єкта, що існує в контексті іншої JVM.

RMI виконує стандартні для розподіленої технології дії. На певному етапі розвитку розробники CORBA та RMI зрозуміли, що ці технології можуть взаємодіяти одна з одною. Стандартним протоколом обміну повідомленнями в RMI став не тільки RMP (протокол, що використовувався як основний у RMI), а й IIOP – протокол, що є стандартним для CORBA. Також було внесено багато інших змін. Тісне співробітництво OMG із Sun і JavaSoft дозволило тісно ув'язати технології CORBA, RMI та Enterprise Java Beans.

Компонентна модель EJB (Enterprise Java Beans) орієнтована на створення масштабованих, стійких та надійних серверних застосувань. Оскільки сервери застосувань не розраховані на роботу в інтерактивному режимі та часто не вимагають наявності оператора, то задачі, що розв'язуються за допомогою EJB, концептуально відрізняються від задач, де розумно та зручно використовувати JB.

EJB – це модель створення та керування серверними об'єктами, оптимального використання ресурсів серверів, управління правами доступу та безпекою, транзакціями та взаємодії з базами даних різних типів.

Протокол взаємодії у EJB – стандартний протокол CORBA IIOP. Схема управління транзакціями JTS (Java Transaction Service) – це реалізований на Java сервіс транзакцій CORBA. Існує стандарт відображення EJB на CORBA, що стосується управління транзакціями, безпекою та служби імен (Naming Service). В цьому стандарті вказано, що специфікація EJB 1.1 не вимагає, хоча і вважає бажаною, повну координацію EJB із CORBA. У специфікації Java2 ця вимога вже стає жорсткою. Отже, складовою частиною EJB може бути тільки те, що відповідає стандартам RMI/CORBA.

Як бачимо, більшість технологічних рішень, які орієнтовані на розв'язання певних задач, найбільш ефективно можуть бути використані не окремо, самі по собі, а в режимі тісної взаємодії з іншими технологіями. Кожна з розглянутих технологій підходить для створення Системи керування дистанційною освітою, але не сама по собі. З огляду на складність та різнорідність задач, що поставлені перед ПСПЕО, для кожного класу таких задач треба використовувати таку технологію, яка є найбільш доцільною. Усі ці технології мають добре продуману взаємодію з іншими. Це дозволяє говорити про поняття метатехнології – такої, яка об'єднує в собі потужні та розвинені концепції. З огляду на вимоги до архітектури та реалізації ПСПЕО, що були висунуті у відповідних розділах нашої роботи, найбільш доцільно використовувати симбіоз RMI, CORBA, EJB та XML для створення серверної частини системи. Кожна з цих технологій має продуману розширюваність, а всі разом вони задовольняють усі вимоги до архітектури та реалізації ПСПЕО. Орієнтація на сумісність з технологією CORBA дозволить у подальшому замінювати використовувані технології на альтернативні. Наприклад, частину системи можна буде переписати (або дописати) із використанням технології DCOM від Microsoft, якщо стане очевидною її перевага для реалізації певних компонентів системи. Така можливість стане доступною при завершенні реалізації OMG моста COM-CORBA, який дозволить створювати гетерогенні системи з використанням як COM (або COM+), так і EJB (та інших технологій).

2. Загальні підходи та схеми побудови ПСПЕО

На сьогодні як у світі взагалі, так і в Україні існує величезна кількість програмного забезпечення – систем, серверів, підручників, інтерактивних курсів, розроблених як засоби ПСПЕО [4]. Саме тому починати розробку чогось нового, з одного боку, досить важко, а з другого – легко, як би парадоксально це не звучало. Важко тому, що практично не можливо придумати щось принципово нове, не реалізоване в якійсь системі. Тому спробуємо узагальнити існуючий у світі та в Україні досвід ПСПЕО, виділити головні, критичні частини та побудувати модель, на основі якої можна буде почати розробляти свою ПСПЕО для своїх потреб в університеті.

У цьому розділі ми спробуємо розглянути загальні підходи та схеми, що використовуються при побудові систем дистанційної освіти [4, 5]. На нашу думку, варто одразу відкинути ПСПЕО, які реалізовані як масив статичних веб-сторінок. Можливо, це і виглядає як лекційний матеріал, але під визначення ПСПЕО не підпадає. Отже, в будь-якій системі дистанційної освіти має бути реалізована взаємодія між такими об'єктами: спудей, лектор (тьютор), електронний курс (електронний підручник, електронний репозитарій навчальних матеріалів, електронна навчальна платформа), система керування навчальним процесом (СКНП) освітнього закладу.

Також мають виконуватися умови розподіленості та модульності. Наприклад, електронній курс (ЕК) – автономний програмний модуль, завданням якого є провести спудея від початку курсу до його кінця (або відповідно до логічної частини курсу).

Уся робота студента, після замовлення ЕК, полягатиме, звісно, лише у спілкуванні з замовленим ЕК. Тобто фактично не потрібно буде створювати жодних додаткових користувацьких інтерфейсів. Лектор та тьютор переглядає

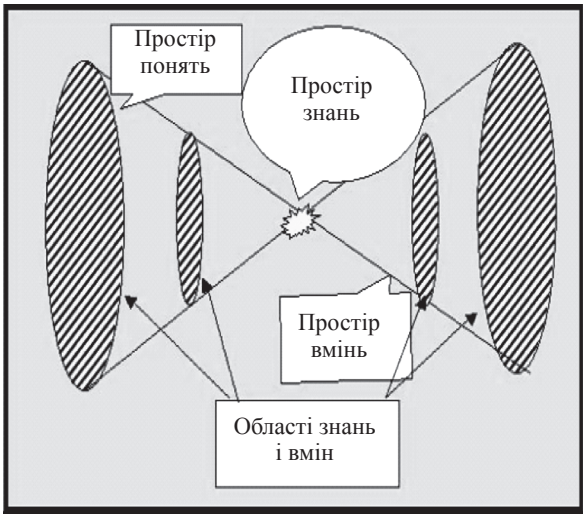
статистику роботи студента з ЕК (яку ЕК автоматично відправляє на сервер СКНП) та надсилає доступними каналами зв'язку свої рекомендації та зауваження. Підсистема контролю та спілкування може бути реалізована як в ЕК (простіший варіант), так і як частина СКНП серверу.

Остання основна підсистема – це система керування навчальним процесом. Залежно від завдань, які ставить перед собою освітній заклад, це може бути проста система, яка лише надає послуги з реєстрації на курсі, отримання ЕК та збір статистики навчання студента, так і складніша, яка включає в себе вбудовані функції розповсюдження мультимедіа контенту, онлайнових лекцій, календарних групових занять і т. д.

3. Структуризація знань

Сукупність знань є нескінченною масою взаємозв'язаних понять. Кожна наука, розділи науки, різні погляди на те чи інше явище – це зріз нескінченного простору знань. Виникає питання: для чого необхідна структуризація знань і оформлення у вигляді особливого формального механізму для електронного навчання з використанням Інтернету? Електронне навчання – принципово нове явище, в якому, окрім лектора та спудея (суб'єкта й об'єкта), є щось нове, не формульоване єдиним терміном. З одного боку, це технічний посередник (комп'ютер), з другого – нескінченне джерело знань, що працює в режимі миттєвого отримання будь-якої інформації (WWW). У зв'язку з цим виникає необхідність (окрім діалогу лектора і спудея, контролю лектора за спудеєм) створити інструмент, що забезпечує не діалог, а спілкування утрьох, коли всі сторони трикутника стають рівноцінними: лектор – WWW – спудей. Структура подання навчального матеріалу кардинально змінюється порівняно з класичною схемою, створюються електронні репозитарії навчальних матеріалів. На сьогодні немає чіткої концепції формування подібних курсів. Тому при їхній організації часто застосовуються універсальні схеми, єдині для різних типів знань. На сьогодні ПСПЕО є замкнутими системами з жорсткими моделями, що не дозволяють адаптувати конкретний тип знань до конкретного формату представлення інформації, але майбутнє – за відкритими системами із загальновизнаними форматами подання знань.

Простір знань є нескінченною кількістю понять, визначень, тверджень, умінь та інших структурних вузлів (СВ), взаємозв'язаних структурними зв'язками різного характеру. Структурні вузли можуть бути утворені і так званими модулями знань (система МРІТО). Цей простір знань може бути розділений на замкнуті області знань, усередині яких існують однорідні. зв'язки однорідного характеру (див. мал. 1).



Мал.1. Простір знань

У загальному просторі знань можна виділити зрізи (області знань), які мають чітку замкнену **структуру** зв'язків. Між цими областями також існують структурні зв'язки, що об'єднують увесь простір знань. Чим більше понять або умінь існує у певній області знань, тим ширший зріз простору знань, тим більше **структурних зв'язків** (СЗ) У вершині двостороннього нескінченного конуса міститься загальне поняття – простір знань. Чим далі від вершини, тим більша кількість понять або умінь (структурних вузлів) є у цьому зрізі (області знань).

Головним завданням при вивченні **структурних** знань, а також зв'язків усередині областей між структурними вузлами, є точна диференціація понять, створення єдиної замкненої логічної схеми. В іншому випадку область знань не можна буде представити у вигляді окремого структурного елемента, і виникнуть труднощі при вивченні структурованих вузлів.

Технологія структурного аналізу інтернет-простору та виділення окремих областей знань є способом складання депозитаріїв навчальних матеріалів (курсів). Зрізи інформаційних ресурсів Інтернету є областями знань, а структура областей знань передбачає організацію дистанційного навчання і складання інформаційних модулів (понять) у логічну структуру курсу.

Одним із важливих критеріїв при структуризації областей знань (ОЗ) і простору знань при використовуванні нових інформаційних технологій – це принцип адекватності, який полягає у створенні таких принципів структуризації, які б органічно вписувались у способи і методи роботи Інтернету та пошукових систем. Тобто області знань повинні бути представлені так, щоб їхня комп'ютерна обробка і способи представлення інформації відповідали вимогам інтернет-технологій. Базовим поняттям тут виступає електронний курс.

Електронні курси зберігаються на навчальному сервері або носіях інформації (CD, МО та ін.). Вони містять дидактичні, методичні й інформаційно-довідкові матеріали з навчальної дисципліні, а також програмне забезпечення, що дозволяє комплексно використовувати їх для самостійного одержання і контролю знань.

Визнаними стандартами для розробки ПСПЕО можуть слугувати системи Прометей-IV та Learning Space 5.0 (Lotus/IBM) [6, 7].

Система «Прометей» призначена для організації повноцінного процесу дистанційної освіти та незалежної перевірки знань, причому розрахована вона переважно на великі потоки слухачів-студентів. Система дозволяє проводити навчання та перевірку знань як у корпоративних мережах, так і в мережі Інтернет, крім того, її можна використовувати і в змішаному навчанні.

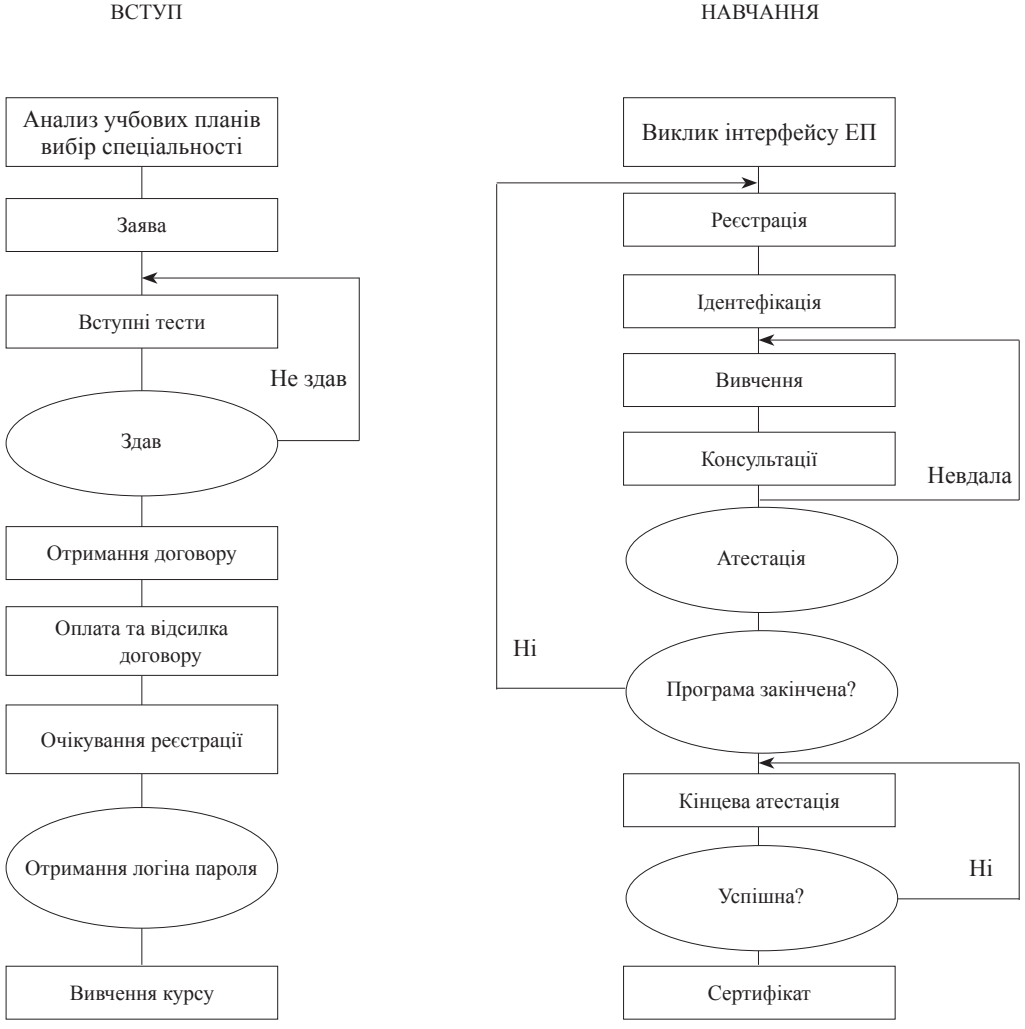
Робота системи «Прометей», як і організація навчального процесу, побудована на об'єктах. Об'єкти використовуються в системах замовлень, платежів, курсів і т. д. Доступ до об'єктів системи, які зберігаються у базі даних, реалізовано через інтерфейс користувача. Функціонування системи базується на взаємодії об'єктів та на зміні їхніх властивостей. Базовим об'єктом, що бере участь у більшості взаємодій, є курс. З курсом пов'язані навчальні матеріали, календарний план, форуми, групи слухачів та ін.

4. Розробка загальної схеми керування ПСПЕО

Отже, після аналізу світового та українського досвіту на ниві побудови ПСПЕО перейдемо до головної частини роботи – розробки загальної схеми керування ПСПЕО навчального закладу.

Алгоритм навчання

Спочатку розглянемо два алгоритми, за якими буде діяти система: алгоритм вступу на курси ЕО та алгоритм навчання. Вони зображені на мал. 1.



Мал. 1. Алгоритм зарахування та навчання

Перший крок – це вступ до освітнього закладу. Користувач заходить на статичну сторінку закладу і знайомиться з інформацією про навчання. Ці сторінки є насправді частиною іншої системи – системи керування навчальним процесом освітнього закладу, і загалом не належать до системи керування ПСПЕО. Проте потрібно передбачити варіант, коли такі сторінки генеруються динамічно з існуючих електронних курсів. Отже, після закінчення вступу спудею видається електронний курс (підручник). Основна навчальна робота студента проходить саме з цим програмним продуктом. Він проводить слухача по всіх темах курсу, застосовуючи або case-методи, або звичайні методи навчання з відповідними тестами після кожного розділу для закріплення знань. Загалом ЕК – це **абсолютно автономна** система, яка використовує власні методи та схеми навчання, довільні технології та мови програмування. Головне, щоб він відповідав клієнт-серверній архітектурі ПСПЕО та працював за стандартним протоколом, який викладений далі.

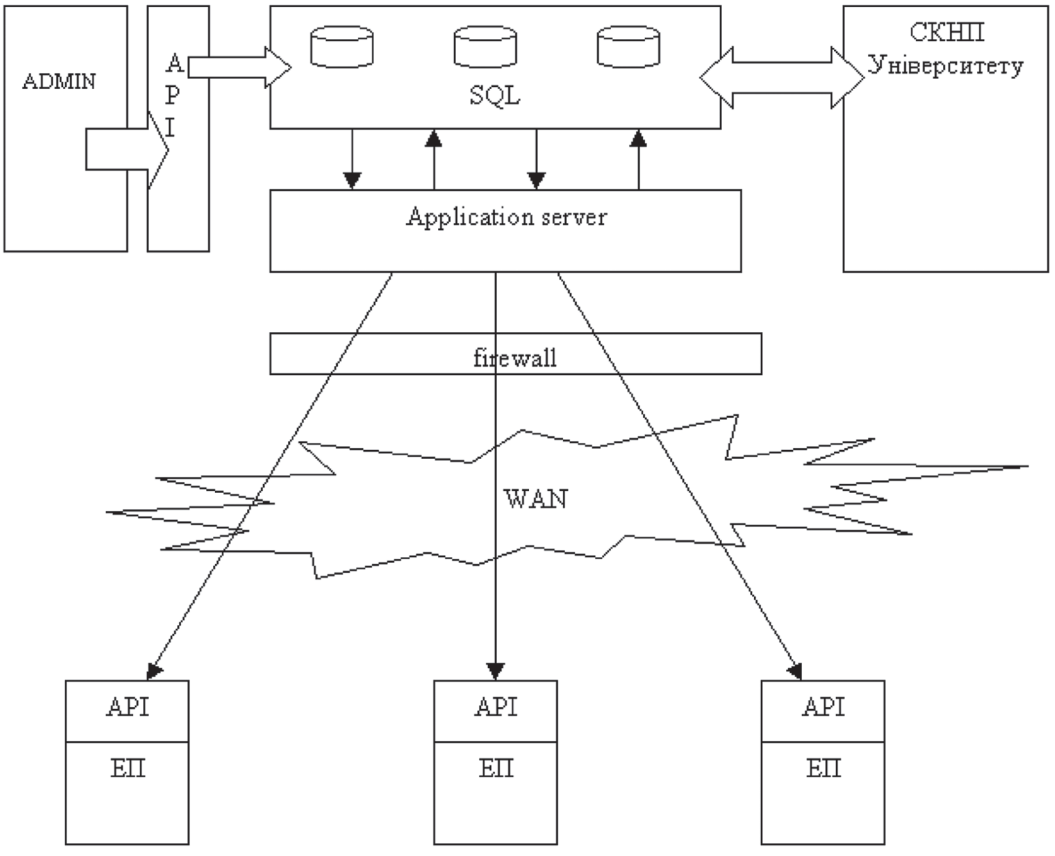
Клієнт-серверна архітектура та розробка протоколу

Система керування – це одна з декількох підсистем у комплексі ПСПЕО. Розглянемо загальну схему та виділимо базові підсистеми. На мал. 2 показано таку систему.

Базове завдання електронного підручника є комплексне навчання спудея та його тестування. З організаційної точки зору це також збір статистики про хід навчання та відправка її на сервер. Оскільки ЕК– незалежна програмна одиниця, її взаємодія можлива або через прямий інтерфейс до бази даних (що не є гарним варіантом), або за допомогою спеціалізованого API. Розробці та специфікації такого API присвячено цей розділ.

ЕК зв'язуються не безпосередньо з сервером баз даних, а зі спеціалізованим сервером, який обробляє транзакції, слідкує за дотриманням прав доступу, може виступати і як збалансовувач навантаження на SQL-сервери. Назвемо такий сервер сервером ПСПЕО, а протокол взаємодії сервера та ЕК – протоколом SDLCP (Simple Distance Learning Communication Protocol).

Наступне завдання системи керування ПСПЕО є робота з базами даних користувачів, електронних курсів, та адміністрування. Тут можливі декілька ситуацій. Якщо в освітньому закладі уже запроваджена СКНП, то такі бази будуть частиною цієї системи. Але якщо такої системи не буде або ПСПЕО буде запроваджено швидше, ніж СКНПР, необхідно буде розробити такі бази даних. Цьому присвячено наступний розділ.



Мал. 2. Структура ПСПЕО

4.1. Опис баз даних СКНП

Як було зазначено у попередньому розділі, бази даних ПСПЕО попередньо можуть бути інтегровані у СКНП освітнього закладу. Проте, за відсутності такої інтеграції чи СКНП можна використовувати викладений нижче варіант.

Головною таблицею тут будуть користувачі системи.

Users DB

№	Назва	Пояснення
1	Id	Ідентифікатор користувача в системі
2	Name1	Ім'я
3	Name2	Прізвище
4	Name3	По батькові
5	Initials	Ініціали
6	Sex	Стать
7	Photo	Фотографія
8	Birth	Дата народження
9	Login	Логін
10	Hash_pass	Хеш-функція паролю
11	Valid	Locked/unlocked
12	Valid_message	Повідомлення
13	Expire	Час
14	Type	Слухач, лектор і т. д.
15	CourseID	Курс

Реалізація на MySQL:

```
create table Users_DB (
  id int primary id key auto_increment,
  name1 varchar(255),
  name2 varchar(255),
  name3 varchar(255),
  initials varchar(2),
  sex char,
  photo varchar(255), //photo is stored as a string on
      file server
  birth date,
  login varchar(18),
  hash_pass varchar(255),
  valid int,
  valid_message varchar(255),
  expire date,
  type int,
  courseID int,
  reserved_string varchar(255),
  reserved_int int
)
```

Місце зберігання ЕК – база даних ЕК (електронний репозитарій). Найкраще реалізовувати його як файловий (FTP) сервер, куди надається доступ для завантаження ЕК, після оплати навчання та отримання відповідних прав доступу. Проте існує досить велика кількість потрібної інформації, яку потрібно зберігати у зовнішній базі даних. Структура такої бази залежатиме від конкретних задач, але загалом може виглядати приблизно так:

Courses_DB

№	Назва	Пояснення
1	Id	Id
2	Id_string	Унікальний id. «Наспорт ЕП»
3	Name	Довільна назва
4	Author	Автор(и)
5	Grade	Мін. рівень користувача
6	Faculty	Комп'ютерні науки
7	Level	Бакалавр комп'ютерних наук
8	Subjects	ПРКС, ПрПр, ОСАОПК
9	Info	Info string
10	Valid_date	
11	Expire_date	
12	Price	Ціна

Тобто:

```
create table Users_DB (
  id int primary id key auto_increment,
  idstring varchar(255),
  name varchar(255),
  author varchar(255),
  grade int,
  faculty varchar(255),
  level int,
  subjects varchar(255),
  info varchar(255),
  valid_date date,
  expire_date date,
  price float,
  reserved_string varchar(255),
  reserved_int int
)
```

Потрібні бази для динамічного збору статистики можна генерувати динамічно на основі унікального ідентифікатора користувача та паспорту ЕК. Вони можуть бути простими (лише оцінки з тестування типу склав/не склав) і складними, залежно від типу поточних задач. Примірний код ініціалізації баз виглядає так:

```
<?
//init/update.php
require_once("globals.php");
require_once("common.php");

if($createDB !=0){$stmt="create database
   $databaseName";}
  $stmt1="create table $tableName1(id int primary
      id key auto_increment,
  name1 varchar(255),
  name2 varchar(255),
  name3 varchar(255),
  initials varchar(2),
```

```
  sex char,
  photo varchar(255), //photo is stored as a string on
      file server
  birth date,
  login varchar(18),
  hash_pass varchar(255),
  valid int,
  valid_message varchar(255),
  expire date,
  type int,
  courseID int,
  reserved_string varchar(255),
  reserved_int int
)";
  $stmt2="create table $tableName2(id int primary
      id key auto_increment,
  idstring varchar(255),
  name varchar(255),
  author varchar(255),
  grade int,
  faculty varchar(255),
  level int,
  subjects varchar(255),
  info varchar(255),
  valid_date date,
  expire_date date,
  price float,
  reserved_string varchar(255),
  reserved_int int
)";

$stmt3="insert into $tableName2(name1, name2,
    name3, published, descr, lesson) values
    ('лПНРШАФЕТЩ: уРТБЧПЮОПЕ ТХЛПЧП
    ДУФЧП.- ч ъ-И ФПНБИ.','рПД ТЕД. з.
    иЕМНУБ.','NULL','н.:нЙТ, 1986','book#1','0')";
$stmt4="insert into $tableName2(title, author, isbn,
    published, descr, lesson) values('йОЖПТН
    БФЙЛБ. пУОПЧПРПМБЗБАЭЕЕ ЧЧЕ
    ДЕОЙЕ.','вТПК н.','NULL','дЙБМПЗ - нйжй,
    1996 - 1998.','book#2','0')";

$stmt5="insert into $tableName1(name, description,
    lesson) values('http://www.microsoft.com',
    'valuable site','0')";

//open db  connection
if(!($link=mysql_pconnect($hostName,
    $userName, $password))){
    DisplayErrMsg(sprintf(«Error connecting to
    host %s by user %s", $hostName, $userName));
    exit();
}
if(!mysql_select_db($databaseName, $link)){
    DisplayErrMsg(sprintf("Error in selectig
    database %s», $databaseName));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link), mysql_error($link)));
    exit();
}
if($createDB !=0){
if(!($result=mysql_query($stmt, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement», $stmt));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
        mysql_error($link)));
    exit();
}
}
if(!($result=mysql_query($stmt1, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement", $stmt1));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
    mysql_error($link)));
    exit();
}
if(!($result=mysql_query($stmt2, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement", $stmt2));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
        mysql_error($link)));
    exit();
}
if(!($result=mysql_query($stmt3, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement", $stmt2));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
        mysql_error($link)));
    //exit();
}
if(!($result=mysql_query($stmt4, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement", $stmt2));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
        mysql_error($link)));
    //exit();
}
if(!($result=mysql_query($stmt5, $link))){
    DisplayErrMsg(sprintf("Error in executing %s
    satement", $stmt2));
    DisplayErrMsg(sprintf("Error: %d %s", mysql_
    errno($link),
        mysql_error($link)));
    exit();
}

GenerateHTMLHeader($siteName, "Init completed
    successfully");
    printf("<BR><FORM ACTION=\"index.
    php\"METHOD=post\n>");
    printf("<INPUT TYPE=submit VALUE=
    \"Click\"> to return to Main Page\n");

?>
```

Після створення та ініціалізації баз даних потрібний зручний інтерфейс керування ними.

4.2. Протокол взаємодії та клієнтський API

Природно, що сервер та клієнти будуть обмінюватися стандартизованими повідомленнями за певним протоколом. Такий протокол потрібний і для сервера ПСПЕО та ЕК-клієнтів. Спробуємо визначити головні завдання та базову версію такого протоколу. За обраним нами алгоритмом роботи (див. попередні розділи) користувач уже отримав свій екземпляр ЕК, зареєструвався у базі користувачів (оплатив навчання) та отримав відповідні права доступу (login, password). Протокол спілкування може звичайно існувати у різних версіях. Ми схиляємося до того, що це будуть короткі текстові (ASCII) повідомлення, виконані у вигляді діалогу.

№	String	Description
1	SEND_LOGIN(String l);	Логін
2	SEND_PASSWD(String p);	Пароль
3	IS_VALID_CREDENTIALS(String l, String p);	Логін та пароль
4	SEND_STAT1(String current_Topic);	Поточна тема
5	SEND_STAT2(String current_Topic, Time time_Spend);	Тема та час проведений
6	SEND_STAT3(String Current_Page);	Сторінка
7	SEND_STAT4(String Checkpoint_ name, Int Grade);	Проміжне тестування результат
8	SEND_STAT5(String Exam_name, Int Grade);	Кінцевий результат
9	SEND_INTERR(Int Eroor_code);	
10	RECEIVE_MESS(String mess);	Повідомлення з серверу
11	SESSION_START	
12	SESSION_END	
13	DATA_START	
14	DATA_END	
15	SEND_EP_ID(String id)	

У такому базовому варіанті спілкування між сервером та клієнтами буде проводитися лише з метою віддати для обробки на сервер статистики роботи з ЕК. Для цього призначенні повідомлення SEND_STAT*. Для початку сесії служать повідомлення про аутентифікацію та власне початок сесії.

Наведемо найпростіший сценарій спілкування, де клієнт відсилає повідомлення про початок роботи над новим розділом і успішне складання попереднього проміжного іспиту:

SESSION_START
SEND_LOGIN(arcanoid11)
SEND_PASSWD(Q4kj8!kjhjk&j)
SEND_EP_ID(QWER-ADSJ-KJGH-BMNV-DGFG-ASDF)
DATA_START
SEND_STAT4(Checkpoint1, 5)
SEND_STAT2("Робота з таблицями у Word", 1)
DATA_END
RECEIVE_MESS
SESSION_END

Під клієнтським API будемо розуміти набір платформонезалежних класів і зовнішніх програм для зв'язку розрізнених типів ЕК з платформою – сервером ПСПЕО. Клієнтський API складатиметься з набору таких програм та протоколу взаємодії, описаного у попередньому розділі. Зазвичай головним завданням такого API буде аутентифікація користувачів та курсів на сервері, їхня реєстрація та збір статистики навчального процесу віддаленого автономного студента. Цим функціональність API, звичайно, не обмежується, проте в цій роботі розглядається лише ця базова функціональність.

При такій реалізації протоколу серйозного значення набувають питання безпеки. Оскільки практично вся взаємодія між сервером та клієнтами – це відкриті текстові повідомлення, то така схема потребує насамперед захищеного каналу зв'язку. На сьогодні це не є проблемою. Весь ip-трафік пропонується пропускати лише по ssl-тунелю, використовуючи для цього засоби stunel [1]. Така організація є досить звичною для сучасного системного адміністратора та не потребує жодних додаткових зусиль. Проте, якщо така схема буде неможливою, потрібно використовувати допоміжну зовнішню утиліту шифрування. Можна використовувати готові безкоштовні продукти (Mypgp).

Висновок

У роботі розглянуто базові принципи побудови ПСПЕО як розподіленої системи на базі трирівневої клієнт-серверної архітектури. У першій частині проведено загальний аналіз ПСПЕО, розгляд можливих платформ, архітектур їхніх побудови. Далі сформовано конкретні вимоги до системи та надана їхню орієнтовну реалізацію. Наведено базовий алгоритм роботи ПСПЕО, розглянуто ролі та інтерфейси, запропоновано структуру баз даних та інтерфейси (протоколи) взаємодії частин системи. Наведено код програмних засобів, які можуть бути використані як клієнтське API – утиліти відсилки повідомлень на сервери реєстрації ЕК, утиліти та системи шифрування повідомлень, а також інтерфейсу користувача та адміністратора до баз даних інформації.

Розроблена система керування ПСПЕО може стати основою для створення та запровадження дієвої системи підтримки електронного навчання в освітньому закладі.

[1] Див. http://www.stunel.org

1. Глибовець М. М. Один із підходів до організації дистанційного навчання / Глибовець М. М. // Проблемы программирования. – 2000. – №1–2. – С. 672–677.
2. Дмитренко П. В. Дистанційна освіта / Дмитренко П. В., Пасічник Ю. А. – К. : НПУ, 1999. – 25 с.
3. Федон П. І. Моделі, методи та технології паралельного програмування / П. І. Федон, А. Ю. Дорошенко, О. А. Летичевський, О. Л. Перевозчикова та ін. // Стан та перспективи розвитку інформатики в Україні. – К. : Наук. думка, 2010. – С. 242–258. – ISBN 978-966-00-0972-0.
4. Дистанционное образование [Електронний ресурс]. – Режим доступу : URL : http://www.informika.ru. – Назва з екрану.
5. Гриценко В. І. Дистанційне навчання: основні визначення [Електронний ресурс] / Гриценко В. І., Кудрявцева С. П. – Режим доступу : URL : http://www.dlab.kiev.ua/TLLL2001/. – Назва з екрану.
6. СДО «Прометей» [Електронний ресурс]. – Режим доступу : URL : http://www.prometeus.ru/. – Назва з екрану.
7. IBM Collaboration software [Electronic Resource]. – Mode of access : URL : http://www-01.ibm.com/software/lotus/ – Title from the screen.

O. Koren

ONE APPROACH FOR CREATION OF PROGRAM SYSTEMS SUPPORTING ELECTRONIC EDUCATION

Work describes basic principles for creation of PSSEE (program systems supporting electronic education) as distributed system on base of three level client-server architecture. First part contains general analysis of PSSEE, observation of possible platforms and its creation architectures. Further, concrete requirements for system are formulated and oriented realization provided. Subjects and objects of distance learning process, their interaction and work protocols are observed. Interfaces of interaction between separate objects and databases were defined.

Keywords: program systems supporting electronic education (PSSEE), module, electronic course, distributed system, distance learning.

УДК 681.3

Костюк О. О.

КОНЦЕПЦІЯ ПОБУДОВИ МАТЕМАТИЧНОЇ МОДЕЛІ ДОКУМЕНТООБІГУ ВІРТУАЛЬНОГО ПІДПРИЄМСТВА

У статті розглянутий підхід до створення моделі документообігу віртуального підприємства (ВП) на основі апарата ланцюгів Маркова. Досліджений процес руху документів у ВП і сформульовані специфічні характеристики, властиві саме віртуальним підприємствам. Описані методи прогнозування кількості документів для розробленої моделі. Положення цієї статті можуть бути використані для подальшого розвитку теорії й практики електронного документообігу й створення на їхній основі прикладного програмного забезпечення.

Ключові слова: електронний документообіг, процесне керування, ланцюги Маркова, модель документообігу.

Загальна постановка проблеми

Відтоді, як керування бізнесом стало наукою, компанії впроваджували різні організаційні моделі для забезпечення максимальної ефективності своєї роботи та розширення бізнесу. Вони використовували горизонтальний, вертикальний та інші проміжні варіанти організаційних моделей, а також централізовані, децентралізовані, матричні й мережні моделі. Кожна з них має свої переваги, але все ж існує потреба у новій організаційній концепції керування, яка б мала більші перспективи. На нашу думку, управлінська модель майбутнього має забезпечити гармонічне