

УДК 004.4

КВЕД К 73.20

№ держреєстрації Р\Р УКРІНТЕІ № 0109U000678

Інв. №

Міністерство освіти й науки України
Національний університет «Києво-Могилянська академія»

04655, м. Київ, вул. Г. Сковороди, 2
тел. (044) 417 84 61

ЗАТВЕРДЖУЮ

Віце-президент із науково-
навчальних студій

доктор філол. наук, професор

_____ В. П. Моренець

“ ____ ” _____ 2010 р.

ЗВІТ

про науково-дослідну роботу

***«Створення програмної платформи підтримки
автоматизованої системи управління навчальним закладом»
(заключний)***

Керівник НДР,
декан ФІ НаУКМА,
завідувач кафедри інформатики,
доктор фізико-математичних наук, професор

М. М. Глибовець

2010

Рукопис закінчено 17.12.2010 р.

Список авторів

1. Керівник проекту: Глибовець Микола Миколайович, декан факультету інформатики, доктор фіз.-мат наук, професор - 1
2. Виконавці:
3. Бублик Володимир Васильович, завідувач кафедри мультимедійних систем факультету інформатики, кандидат фіз.-мат. наук, доцент - 4,5
4. Ляшко Володимир Іванович, старший викладач кафедри інформатики, кандидат фіз.-мат. наук - 4
5. Глибовець Андрій Миколайович, старший викладач, кандидат фіз.-мат. наук - 3
6. Гломозда Дмитро Костянтинович, аспірант факультету інформатики – 2, 3
7. Гороховський Семен Самуїлович, доцент, керівник магістерської програми «Інформаційні управляючі системи та технології», кандидат фіз.-мат. наук, старший науковий співробітник – 1, 2
8. Гулаєва Наталя Михайлівна, старший викладач кафедри інформатики, кандидат фіз.-мат. наук - 3
9. Ковальський Ігор Петрович, доцент кафедри мультимедійних систем, кандидат фіз.-мат. наук, доцент - 1, 2
10. Корень Олександр Миколайович, здобувач кафедри інформатики, 2,5,6
11. Кульчицький Юрій Михайлович, аспірант факультету інформатики 3, 4
12. Олецкий Олексій Віталійович, доцент кафедри мультимедійних систем, кандидат фіз.-мат. наук, доцент - 5
13. Омельченко Віталій Володимирович, завідувач лабораторії інформатики факультету - 4
14. Синявський Олександр Леонідович, доцент кафедри мережних технологій, доктор технічних наук - 1
15. Федорченко Віталій Михайлович, аспірант факультету інформатики 2, 3

РЕФЕРАТ

Звіт про НДР: 153 с., 39 рис., 2 табл., 18 джерел.

Об'єкт дослідження - *автоматизована система управління навчальним закладом.*

Мета роботи – Створення прототипу *програмної платформи підтримки автоматизованої системи управління навчальним закладом.*

Метод дослідження – моделювання на мережах Петрі, імітаційне моделювання, статистика.

Розроблено прототип типової програмної системи управління навчальним закладом (АСУНЗ) апробована в Національному університеті “Києво-Могилянська академія” системи управління, що забезпечує виконання стандартних функцій керування як навчальним процесом та допоміжними структурними підрозділами, так і підсистемою підтримки електронного навчання. Однією з задач досліджень була розробка архітектурного рішення і реалізація платформи, які б забезпечили можливість ефективного виконання всіх функціональностей згідно з вимогами міжнародних стандартів. Така уніфікована комплексна робота, що поєднує дві найважливіші підсистеми університету (Learning Management System, Content Management System) не застосовується в відомих нам системах АСУНЗ України.

Побудована на компонентному підході із застосуванням принципу ІоС система без принципових обмежень може бути розширена до рівня автоматизації будь-якого навчального закладу, або будь-якої установи, де документообіг відіграє важливу роль, без обмеження функціональності. В результаті була розроблена архітектура, що включає останні досягнення у побудові сучасних програмних застосувань. Використання сучасного інструментарію з відкритим кодом для побудови складних програмних комплексів – це основна ідеологія даного проекту.

Результати НДР упроваджені в 2 організаціях.

Прогнозні припущення щодо розвитку об'єкта дослідження – Розповсюдження системи в університетах України.

Умови одержання звіту: за договором. 252171, Київ-171, вул. Горького, 180, УкрІНТЕІ.

Зміст	
Вступ.....	6
1 Колаборативне електронне навчальне середовище. Загальні поняття й принципи	7
1.1 Архітектура колаборативного навчального простору	7
1.2 Інтелектуалізація систем електронної освіти. Поняття навчальної взаємодії.....	9
1.3 Інтелектуалізація процесу навчання.....	12
1.3 Проектування системи електронної освіти.....	13
1.4 Навчальні взаємодії.....	15
1.5 Тестування як реалізація взаємодії „завдання”	18
Висновок до розділу 1	20
Література до розділу 1.....	Помилка! Закладку не визначено.
2 Принципи побудови програмних систем підтримки електронної освіти	21
2.1 Огляд сучасних технологій для застосування в галузі ПСПЕО	23
2.2 Загальні підходи та схеми побудови ПСПЕО.....	26
2.3 Структуризація знань	27
2.4 Розробка загальної схеми керування ПСПЕО	30
2.4.1 Алгоритм навчання	30
2.4.2 Клієнт-серверна архітектура та розробка протоколу	31
2.4.3 Опис баз даних СКНП	32
2.4.4 Протокол взаємодії та клієнтський API.....	33
2.5 Організація взаємодії між автоматизованою системою управління навчальним закладом та системою керування навчанням.....	35
2.5.1 Огляд АСУНЗ MAMS	35
2.5.2 Зв'язок із базою даних СКН Moodle	36
2.5.3 Функція внесення результатів сесії з Moodle.....	37
2.6 Програмна система підтримки колаборативного навчального середовища.....	40
2.6.1 Огляд інструментальних засобів підтримки дистанційного навчання	40
2.6.2 Агенти технології та середовище JADE.....	44
2.6.3 Опис програмної системи підтримки взаємодії LMS і CMS	47
3 Системи управління навчальним процесом та контентом: СУНП (MAMS) та СУНК (Moodle).....	50
3.1 СУНК в інфраструктурі СУНП.....	51
3.2. Порівняльна характеристика СУНП та СУНК	52
3.3 Системи управління навчальним контентом	53
3.4 Елементи систем управління навчальним контентом.....	55
3.5 Структура СУНП.....	57
3.6 Інтеграція СУНП з СУНК.....	58
4 Принципи реалізації	60
4.1 Структура середовища	60
4.1.1 Вибір технології	60
4.1.2 Архітектура.....	62
4.1.3 Нестандартні архітектурні рішення.....	65
4.1.4 Реалізація	67
4.1.5 Нефункціональні та інші вимоги	69
4.1.6 Функціональність системи	71
4.1.7 Нефункціональні характеристики.....	71
4.2 Технічний опис впровадження.....	74
Висновки до розділів 3 і 4.....	79
Література до розділів 3 і 4.....	Помилка! Закладку не визначено.
5 Методи і засоби підтримки автоматизованих практикумів	81
5.1 Формалізація поняття індивідуального робочого простору користувача в інтернет у межах електронної освіти	83
5.2 Можливість повторного використання матеріалів, стандарти та освітні об'єкти....	87
5.3 Формулювання основних принципів концепції індивідуального електронного «робочого зошита» з програмування на віддаленому сервері).....	88
5.4 Властивості електронних Портфоліо.....	88
5.5 Властивості функціональних модулів MOODLE.....	89

5.6	Планування та реалізація функцій профілю «Студент»	93
5.7	Віддалена компіляція c++ проектів: застосування gsc	95
5.8	Оцінювання та робота з часом	95
	Висновки до розділу 5	97
	Література до розділу 5	Помилка! Закладку не визначено.
6	Розробка інтерактивних навчальних матеріалів	99
6.1	Вивчення курсів з програмування в системах електронної освіти	99
6.2	Аналіз засобів створення та супроводу робочого зошита викладача	100
6.3	Поняття веб-порталу та платформи в e-learning	101
6.4	Розробка системи управління навчальними матеріалами в робочому зошиті викладача	102
6.5	Особливості реалізації. Інтеграція з порталом Joint European Projects на основі стандарту CMS E107	108
	Висновки до розділу 6	109
	Література до розділу 6	Помилка! Закладку не визначено.
7	Архітектура та інформаційна модель програмної системи підтримки електронних портфоліо	111
7.1	Базові специфікації	111
7.2	АРХІТЕКТУРА ЕЛЕКТРОННИХ ПОРТФОЛІО	113
7.3	ІНФОРМАЦІЙНА МОДЕЛЬ ОРГАНІЗАЦІЇ ДАНИХ	118
7.4	рекомендації щодо розробки та імплементації	120
	Висновки до розділу 7	120
	Література до розділу 7	Помилка! Закладку не визначено.
8	Математична модель взаємодій в програмній системі підтримки електронного навчання	121
8.1	Основні підходи до побудови моделей взаємодій	121
8.2	Модель взаємодій на основі мереж Петрі	125
8.3	Універсальність моделі	127
8.4	Складність задачі верифікації координаційного механізму	129
8.5	Доведення тотальності алгоритму координації дій при виникненні помилок	135
	Література до розділу 8	140
	Перелік посилань	148

Вступ

Мотивація досліджень

Метою науково-дослідної роботи було створення прототипу колаборативного навчального середовища, яке включає типову автоматизовану систему управління навчальним закладом (АСУНЗ), що забезпечує виконання стандартних функцій керування навчальним процесом та допоміжними структурними підрозділами, інтеграція колаборативного мультимедійного навчального простору через освітні портали з системою управління навчальним процесом, створення прототипів освітніх порталів факультетів з забезпеченням прозорого віддаленого доступу до інформаційно-довідкових та навчальних матеріалів засобами Інтернету із забезпеченням "дружнього" веб-інтерфейсу. Однією з задач досліджень була розробка й перевірка архітектурних рішень і реалізація платформи для можливості реалізації практично довільної функціональності згідно з вимогами, що будуть розроблятися під час розвитку колаборативного навчального середовища.

1 Колаборативне електронне навчальне середовище.

Загальні поняття й принципи

Наразі, більшість вузів України декларують про входження в міжнародну освітню спільноту, що вимагає поряд з традиційними формами надання освітніх послуг і підтримки електронної освіти. Таке поєднання дістало назву колаборативне або змішане навчання, або колаборативне електронне середовище.

1.1 Архітектура колаборативного навчального простору

Електронна освіта все частіше тяжіє до соціоцентричних систем, де як викладач, так і учень мають можливість змістовного спілкування стосовно навчального контенту. Однак таке спілкування є досить складним з точки зору ефективності, коли інформація, що курсує в електронній освіті, має монолітний характер. Звідси маємо необхідність сегментації навчальної інформації разом з повторним використанням. Серед відомих недоліків і проблем електронного навчання також зазначимо складну інтеперабельність навчальних платформ, що заважає обміну навчальними матеріалами між ними, відсутність ефективного *семантичного* пошуку в популярних платформах, що заважає широкому доступу учнів до репозитаріїв знань (цьому заважає також чисто синтаксичний опис контенту), дуже слабкі засоби персоналізації робочого середовища учня, які не дають можливості створити креативне середовище, пристосоване до індивідуальних властивостей і рис учня (викладача) [1].

Ми пропонуємо концепцію та архітектуру колаборативного навчального середовища, в якому будуть не так відчутні традиційні недоліки електронного навчання, але з'являться нові атракції, такі як швидка інтеграція нових користувачів і створення відчуття причетності користувачів до спільноти, яка вчить і вчиться, ефективне повторне використання знань, поліпшення асинхронного спілкування між різними користувачами, поліпшення доступу до контенту відповідно до преференцій користувачів і бажаних форматів доставки навчального матеріалу з перетвореннями текст-мова, мова-текст, аудіо і відеоперехвати тощо.

Для цього в майбутньому пропонується інтеграція в систему інтелектуальних агентів з подальшим їх вдосконаленням для контролю й моніторингу контенту, семантичної індексації й форматування персоналізованої інформації. Такий загальний підхід не завадить виконанню очевидних вимог до навчального середовища:

- Задоволення стандартів (SCORM тощо)
- Обмін навчальними пакетами між платформами;
- Ефективне середовище виконання, адміністрування
- Семантичний навчальний портал;

- Оптимальне виконання запитів учнів;
- Розподіленість середовища (географічна, функціональна, темпоральна).

Таке середовище ми називаємо колаборативним електронним навчальним середовищем (КЕНС) і розглядаємо як інтелектуальну мережу з онтологічною системою.

З точки зору архітектури комп'ютерна система підтримки електронної освіти мусить представляти навколишнє середовище, базоване на використанні сучасних можливостей мережних технологій, забезпечення віддалених занять як в online, так і в offline, з розвиненими засобами автоматизованого тестування знань і навичок, і, мабуть, найголовніше, забезпеченням симетричного діалогового співробітництва між основними компонентами систем. Розподілене КЕНС має ефективно підтримувати як традиційні форми занять, так і електронну освіту.

Мета пропонованої архітектури полягає в реалізації *цілісного* підходу для виправлення недоліків сучасної електронної освіти, а також у поєднанні автоматичного і колаборативного підходів для поліпшення використання навчального контенту і передачі знань на рівні вищому, ніж навчальні платформи.

Досі вважалося, що застосування інтелектуальних агентів досить дороге й складне з точки зору обчислювальних ресурсів. Тому ми пропонуємо комбінацію автоматичних і визначених користувачем підходів для здобування, розвитку й супроводу семантичних ресурсів із широким застосуванням технологій семантичного вебу. Ці функціональні можливості стануть доступними до користувача за допомогою його повної інтеграції в інтерфейсі. Треба розробити повністю конфігурований адаптивний інтелектуальний інтерфейс, здатний до розвитку залежно від функцій, пропонованих користувачем.

Є кілька причин застосування інтелектуальних мультиагентних систем в електронній освіті. Студенти у віртуальних класах можуть знаходитись на великих відстанях одне від одного, тому статична централізована система не відповідає вимогам. Тут стає в нагоді розподіленість, природна для агентів. Інтереси студентів теж змінюються в часі, навчальні матеріали і методики також. Тут стає в нагоді здатність агентів вчитися та адаптуватися до навколишнього середовища. По-третє, студенти мають різні рівні підготовки, кожен з них особистість, тому методологія навчання мусить адаптуватися до індивідуальності студента. Спілкування, обговорення спільних тем й інтересів, так популярні серед студентів може бути підтримано здатністю агентів до переговорів і наявністю спільної агентної мови, базованої на онтологіях. Нарешті, студенти можуть реєструватися в кількох курсах одночасно. Тому так важлива координація занять, теж природна для агентів. Прикладом спроби створення стандарту освіти на мультиагентних системах може бути система IDEAL [2].

Варто навести переваги поєднання інтелектуальних агентів з навчальним простором:

- Швидка інтеграція нових користувачів і їх залучення до спільноти;
- Ефективне використання контенту та спільних знань;
- Підвищення якості супроводу;
- Поліпшення асинхронної комунікації між користувачами (студентські форуми відомі викладачам);
- Поліпшення доступу до контенту (можна врахувати преференції користувачів).

Наведемо основні компоненти архітектури КЕНС:

- Засоби для авторів (генерація нового контенту, перегляд того, що існує)
- Інтелектуальні семантичні агенти
- Моделі, інтегровані в систему через агентів
- Джерела даних (бази даних, знань, інформаційні потоки)
- Контентні брокери (інтеграція семантичних моделей, зворотній зв'язок)
- Модулі налаштування (персоналізація, сегментація мультимедій, семантичне індексування)

КЕНС дасть змогу поліпшити життєвий цикл контенту і поєднати його з життєвим циклом знань, дозволяючи явне й неявне покращення контенту і його семантики, обмін метазнаннями, в такий спосіб інтегруючи інформаційні джерела в робочі середовища користувачів.

1.2 Інтелектуалізація систем електронної освіти. Поняття навчальної взаємодії

Останні дослідження в області створення сучасних систем електронної освіти (СЕО), привели до загальноприйнятої думки, що таку систему можна розглядати тільки на основі інтерактивних взаємодій як інтелектуальну мережу або деяку онтологічну систему. Тобто СЕО повинна задовольняти такі основні вимоги:

- налаштовуватись на оптимальне виконання запитів користувачів;
- оптимально виконувати основні операції з базами даних і знань по збереженню систематизації та передачі інформації;
- архітектура послуг повинна забезпечувати роботу з різними програмними платформами та орієнтуватися на якісну передачу значного потоку інформації мультимедійного типу.

З функціональної точки зору, враховуючи сутність використання в CEO агентних технологій, для опису взаємодії підсистем характерним є використання “рольового” принципу опису предметної області знань системи та створення систематизованого уявлення про цю область.

Архітектурно, система підтримки електронної освіти повинна представляти собою інтегроване середовище, основане на використанні сучасних можливостей Web-технологій, забезпечення видаленого навчання як в режимі On-line так і в режимі Off-line з розвинутими засобами: автоматизованого тестування знань, забезпечення комутативної інтерактивної взаємодії між базовими компонентами системи (слухач, викладач, адміністратор, агент).

Для формування платформи української системи електронної освіти та гідної її інтеграції в світову систему відкритого навчання потрібно розвивати єдине освітнє навчальне середовище на основі принципу уніфікації рішень, ієрархічної архітектури та використання системи загальновизнаних стандартів. Бажано було б об'єднати зусилля окремих університетів України, а також кращі досягнення окремих національних розробок по створенню комп'ютерних систем підтримки неперервного навчання шляхом створення Відкритого університету електронної освіти України. Про реальність можливості створення такої інституції, яка б працювала з урахуванням національних особливостей, свідчать уже створені українські системи “Академік” Львівського банківського інституту [3] та розподілена система навчання НУ “Львівська політехніка” [4], системи Національного Університету „КПІ”, Національного Університету „КМА” та Національного Київського Університету, що знаходяться в стадії завершення робіт. На жаль, більшість із цих систем ЕО, майже не розглядаються комплексно як складні програмні системи. Тому досить природно постає проблема створення саме такої моделі CEO, яка б давала можливість опису цілісного погляду на процес дистанційного навчання. Прикладом повинен служити міжнародний досвід. Для розв'язку подібних проблем світовою спільнотою був створений комітет по стандартизації IEEE LTSC (P1484 – Learning Technology Standart Committee) розробки архітектури і технологій освітніх систем (Learning Technology Systems Architecture - LTSA) [5] і міжнародний консорціум IMS Global Learning Consortium [6] для розробки специфікацій і рекомендацій по розширенню використання комунікаційних і інформаційних технологій в освіті. А аналогом цілісної моделі може служити модель, що описує колективну роботу в розподілених системах.

У відповідності до ідеології LTSA для розробки технологічних систем електронної освіти бажано виділити такі функціональні складові:

1. розробка електронних варіантів навчальних курсів і посібників з розвинутими засобами самоконтролю засвоєння матеріалу і можливостями інтерактивної взаємодії між основними учасниками навчального процесу (слухач, викладач, адміністратор);
2. керування навчальним контентом, включаючи його доставку;
3. забезпечення простого інтерфейсу отримання знань з бази знань підтримки навчального процесу системи засобами Інтернету;
4. адміністрування процесу навчання;
5. індивідуальне планування навчання, контроль і оцінювання індивідуальних знань слухача;
6. контроль і оцінювання якості викладання; комунікації підтримки процесу навчання і адміністрування.

Для побудови функціональної моделі освітньої навчальної системи конкретного навчального закладу потрібно уточнити п'ять основних рівнів архітектури LTSA.

1. Виділяються два елементи — об'єкт навчання та середовище навчання. Розглядаються питання впливу останньої на слухача в процесі взаємодії з точки зору дослідження процесу передачі знань за допомогою різних засобів. Спільна робота слухачів описується в вигляді внутрішньої взаємодії, аналогічно взаємодії розподілених баз даних при побудові загальної єдиної бази даних.
2. Визначаються особливості розробки, пов'язані з участю в процесі взаємодії людини. Формулюються задачі пов'язані з розробкою інтерфейсу спілкування середовища навчання з слухачем під час процесу взаємодії на природній мові слухача.
3. Визначаються особливості використання системних компонент IEEE 1484.1. Для використання сучасних досягнень в області інформаційних технологій в системі ЕО виділяються: базові процеси (об'єкти), таблиці даних (структури даних) та засоби їх зберігання, основні інформаційні потоки.
4. Фіксується модель організації навчання. Виділяються підсистеми і встановлюється інформаційне навантаження між ними з орієнтацією на особливості вибраної моделі і технології навчання.
5. Визначаються основні вимоги, формуються базові функції, будується концептуальна модель і визначається семантика системи навчання. Забезпечується інтероперабельність освітньої технологічної системи шляхом детального опису основних елементів забезпечення її інтероперабельності: кодів, форматів даних, протоколів взаємодії, рівнів взаємодії і інтерфейсів прикладного програмування (API).

Зрозуміло, що модель організації освітнього середовища буде повністю визначатися організацією алгоритму процесів взаємодії, який складається з таких основних етапів.

1. Слухач вибирає стиль, тактику і стратегію, методи навчання згідно з пропозиціями і можливістю середовища навчання.
2. Проводиться початкове тестування знань слухача і визначення психологічних особливостей слухача з урахуванням його передісторії і мети навчання та з занесенням цієї інформації в відповідні таблиці бази даних.
3. Викладач підбирає навчальні ресурси в базі знань і встановлює режим їх поставки слухачу, включаючи поставку програмного інструментарію підтримки інтерфейсу слухача з системою навчання.
4. Реалізація надання освітніх ресурсів шляхом використання інтерактивних методів мультимедіа
5. Організація як проміжного так і кінцевого тестування для оцінювання отриманих знань слухачем для накопичення інформації в базі даних про успішність слухача.
6. Оцінка рівня сертифікату підтвердження отриманих знань слухачем в системі навчання.

Згідно [15] універсальність LTSA забезпечує її використання в якості еталонної моделі для розробки освітніх технологічних систем довільного типу.

1.3 Інтелектуалізація процесу навчання

Основою довільної системи навчання є побудова ефективної взаємодії між слухачем і середовищем навчання. В більшості випадків ця задача розв'язується шляхом створенням автоматизованої системи навчання на основі як можна ширшої інтелектуалізації процесу навчання. Грицай В.П. [16] виділяє такі концепції навчання.

1. *Когнітивна модель Нормана* [9]. В основу моделі покладено задання знань за допомогою семантичної мережі [17]. Отримання нового знання слухачем проходить за рахунок додавання нової інформації в уже існуючу його семантичну мережу знань. Успіх засвоєння знань в такому процесі повністю залежить від кількості взаємодій з матеріалом навчання і середовищем навчання.
2. *Модель гнучкого навчання* [18]. В цьому підході процес навчання перетворюється в дослідницьку діяльність, основну роль в якій має консультація з викладачем по мірі необхідності.
3. *Гіпертекстова модель* [12]. В основу моделі покладено поняття асоціативності мислення. Задана в явному гіпертекстовому вигляді семантична мережа знань

викладача дає студенту прямий доступ до потрібного матеріалу за допомогою броузингу.

В основу створення більшості автоматизованих систем навчання покладена алгоритмо-евристична теорія навчання Л.Ланди [13], що знайшла розвиток в роботах Атанова Г.А. [14]. В цій теорії вважається, що будь-яка пізнавальна діяльність може бути проаналізована в термінах операцій алгоритмічної, напівалгоритмічної, евристичної, напівевристичної природи. Важливим моментом для такого підходу є побудова адекватної моделі *слухача*. Знання про те, яким ми хочемо бачити слухача після завершення навчання називають нормативною моделлю слухача або стандартом навчання. Згідно [15] останню можна розбити на п'ять компонент: предметну; функціональну; семантичну; процедурну; операційну.

Частину нормативної моделі, що визначає предметні знання, тобто знання навчальних курсів, називають *предметною моделлю слухача*. В термінах інженерії знань такі знання називають експертними знаннями. Для оптимізації процесу формування способу досягнення мети навчання бажаним є структурування знань предметної моделі в вигляді метазнань. Важливим тут є також визначення “рольових” функцій знань, тобто здійснити функціональне структурування шляхом переліку функціональних рубрик. Провівши структурування і зробивши поділ знань на декларативні і процедурні, отримують функціональну предметну модель слухача. Декларативні знання описують твердження (факти) про властивості об'єктів предметної області вивчення і відношень між ними, тобто вони визначають змістовну частину предметних знань, породжуючи семантичну предметну модель слухача. Семантичний факт є найбільш мілкою одиницею знань предметної області. Процедурні знання описують порядок і характер перетворень об'єктів предметної області і грають роль схем орієнтовних основ діяльності слухача в процесі розв'язку задач. Знання, за допомогою яких формуються вміння знаходити розв'язки задач, тобто виробляти спосіб дій для розв'язку певної задачі в процесі навчання, називають *операційними знаннями*.

Узагальнюючи вищезазначене, можна вважати, що інтелектуалізація навчального процесу в автоматизованих системах навчання полягає у виробленні ефективних методів задання знань предметної області як слухача так і середовища навчання шляхом їх поділу на тематичні, функціональні, семантичні, процедурні і операційні та створення на основі цього поділу авторизованих гіпермедійних розподілених баз знань.

1.3 Проектування системи електронної освіти

Подальший розвиток нових інформаційних технологій поступово дозволяє усунути основні гальма розвитку неперервної електронної освіти – формування адаптивної

структури навчання в залежності від індивідуальних особливостей учня, шляхом адекватної підтримки розвитку методик інтерактивного спілкування учителя та учня, які б реалізували: єдність методичного, організаційного, інформаційного, програмного, технічного середовищ; систематизацію інфоресурсів, створення великих розподілених баз знань; побудову стандартів (норм), що полегшили б пошук, обмін та розповсюдження навчальних матеріалів з можливістю інтерактивного спілкування. Відомо, що до таких технологій можна віднести технологію багатоагентних систем.

Система, в якій декілька агентів можуть спілкуватися один з одним, здійснювати обмін поточною інформацією, взаємодіяти між собою, називається багатоагентною (мультиагентною). Такі системи, можна віднести до розподілених систем штучного інтелекту [16]. Традиційно, в системах, для розв'язку однієї задачі використовується декілька інтелектуальних систем. Тобто, задача розбивається на підзадачі, розв'язок яких розподіляється між окремими агентами. В випадку CEO, кожний із учасників процесу навчання може представлятися в вигляді агента. Більш детальніший опис використання агентних технологій в CEO, що демонструє доречність їх використання можна знайти в [17 Глибовець].

Відомо, що процес проектування й моделювання при створенні програмного забезпечення може займати до 80% всіх витрачених ресурсів. Очевидно також, що грамотна й правильно спроектована несуперечлива модель грає критичну роль в усій розробці і помилку на етапі кодування виправити значно простіше, ніж прорахувати в проектуванні. Внутрішньо неузгоджена або неповна модель може призвести до повної невдачі проекту. До того ж, в останні роки з'явилися розвинуті мови підтримки моделювання. Однією з найбільш досконалих серед них є уніфікована мова моделювання UML (Unified Modeling Language). Так версія 1.0 стала розглядатися як галузевий стандарт розширюваної мови об'єктного моделювання. Сьогодні UML – мова візуального моделювання програмного забезпечення і складних систем обробки інформації, що включає в себе певну систему умовних позначень, яка призначена для виразу ідей та рішень, що виконуються на етапі об'єктно-орієнтованого аналізу й проектування. Зазначають, що мову UML доцільно використовувати для створення специфікацій, візуалізації й документування артефактів програмних систем.

Засоби автоматичної кодогенерації дозволяють переводити моделі на UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Нині, існує багато CASE-засобів, що автоматизують процес аналізу й проектування в UML. Прикладом може служити Rational Rose [18].

З усього сказаного випливає, що на процес створення CEO можна поглянути так: потрібно створити повну експертну систему, основу на розподіленій базі знань, побудова виведення заключень в якій покладено на деяку мультиагентну систему. Тоді для опису проектування такої системи ЕО можна використати систему проектування Rational Rose Professional Edition. Багатоагентна система може бути створена на основі програмної платформи Java Agent Development Environment (JADE). Використання останньої значно спрощує процес розробки мультиагентних систем завдяки використанню FIPA – специфікації та за допомогою ряду інструментаріїв. Перевагою JADE є її платформонезалежність та можливість використання віддаленого GUI-інтерфейсу.

Для створення функціональної моделі мультиагентної системи бажано відстежувати всі можливі взаємодії в системі, логічна структура якої приводиться на Рисунку 1.1

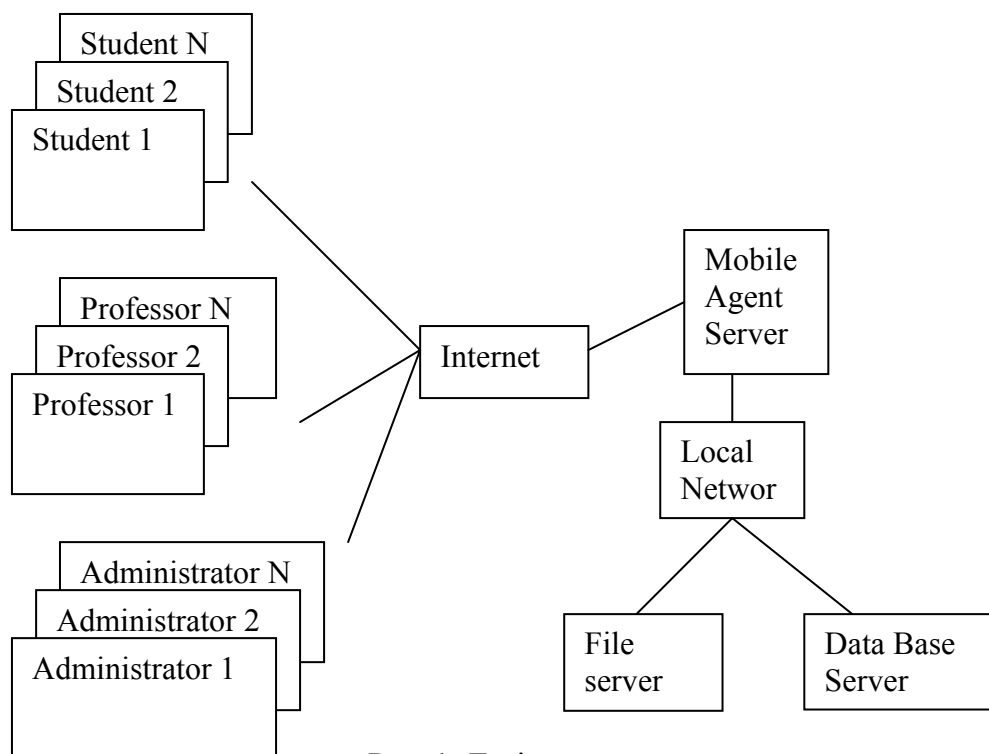


Рис. 1. Логічна структура системи

Зрозуміло, що основу таких взаємодій складає навчальна взаємодія.

1.4 Навчальні взаємодії

Під *навчальною взаємодією* будемо розуміти будь-які дії слухача або середовища, наслідком яких є отримання знань та навичок слухачем. Можна виділити такі види навчальної взаємодії у системах ЕО:

1. *Отримання знань* шляхом перегляду навчальних матеріалів з гіпермедійної бази знань.
2. *Стимуляція навичок* (семінари, практичні заняття, лабораторні роботи)

3. *Спілкування* з викладачами, іншими слухачами, системою

4. *Оцінювання*.

Поняття навчальної взаємодії повинне узагальнювати ці види взаємодій, реалізовані різними способами, зокрема такими, як броузинг, відеоконференція, чат, форум, електронна пошта тощо з урахуванням різних психологічних аспектів. Воно також повинно бути розширюваним, тобто повинно давати змогу адекватно описувати інші типи взаємодій, що ще розробляються, або реалізовані поза системою. Прикладом такої взаємодії є “позакласне” спілкування слухача і викладача, що у реальному житті ніяк не протоколюється і не передбачене програмою, але впливає на результат.

Традиційно, за реалізацію отримання знань у системі відповідає *підсистема отримання знань*. Три останні види взаємодій можуть бути узагальнені за допомогою абстракції *завдання*. Уточнимо ці поняття наступним чином.

Під взаємодією (Interaction) будемо розуміти наступний набір: тип, учасники, комутатор:

Interaction = {Type, Actors, Commuter}.

Тип (**type**) — набір атрибутів взаємодії щодо суттєвих параметрів. Початково тип може складатися з одного поля **TimeSensitivity**—часової чутливості—максимально допустимий час затримки реакції об’єкта на повідомлення, тобто час від кінця отримання об’єктом повідомлення до кінця передачі його повідомлення-відповіді.

Учасник (**actor**)—учасник навчального процесу, що має визначений набір атрибутів та функцій, а також володіє певними правами доступу до системних функцій. Він визначається набором сумісних **інтерфейсів** та часовою чутливістю:

Actor = {Interface1, ..., InterfaceN, TimeSensitivity}.

Комутатор (**Commuter**)— об’єкт що відповідає за проходження і перетворення інформаційних потоків. Він комутує інформаційні потоки. Комутатор визначається набором інтерфейсів, з якими він сумісний, середовищем, і часом затримки:

Commuter={Control, Interfaces, Environment, Type}.

Під управлінням (**Control**) розуміється модель комутації інформаційних потоків. Середовище (**Environment**) — характеризується надійністю передачі, наявністю або відсутністю історії, можливостями передачі різних типів медіа:

Environment={Reliability, History, MediaSet}.

Інтерфейс визначається як набір можливостей проходження медіа у дві сторони:

Інтерфейс= {InCapability, OutCapability},

де

InCapability = {MediaSet, TimeSensitivity},

OutCapability = {MediaSet, TimeSensitivity},

MediaSet = {Text, Sound, Video, Movement, Image, Model} і

Text позначає текст, **Image**—зображення, **Sound**—звук, **Video**—відео, **Movement**—рух, **Model**—інтерактивна модель.

У рамках даного підходу різні типи медіа спеціально не приводяться до гіпертексту, тому що у реальних середовищах не всі вони можуть бути передані.

Схему взаємодії відображає рисунок 1.2.

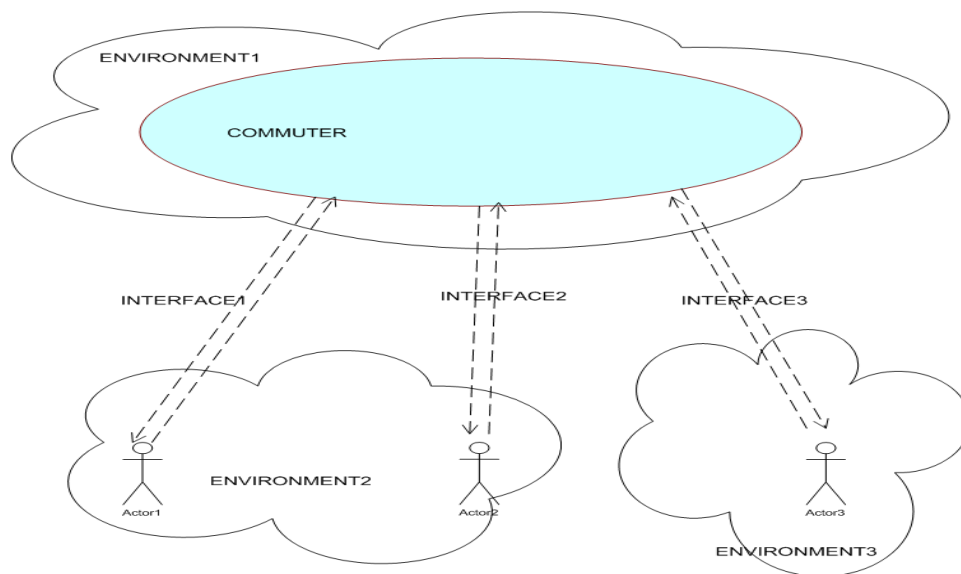


Рисунок 1.2. Схеми взаємодії

Зрозуміло, що однією з основних навчальних взаємодій є – завдання.

Завдання—це комунікативна взаємодія, що проводиться з метою оцінювання рівня знань і умінь слухача. Завдання успадковує властивості взаємодії і узагальнює такі поняття як семінар, практична робота, тест, іспит.

Завдання пропонується описувати наступним чином (вільна нотація):

ЗАВДАННЯ

Мета роботи

Розділи навчального матеріалу що перевіряються або закріплюються

Необхідні для роботи об'єкти

Зміст роботи

Результат на 100%, балів

Додаткові відомості (посилання на навчальні матеріали, необхідні для виконання завдання, методичні вказівки)

ЗАДАЧІ

ЗАДАЧА_1

ТИП

ПОСТАНОВНИК

ОЦІНЮВАЧ

Результат на 100%, балів

Зворотній зв'язок

1.5 Тестування як реалізація взаємодії „завдання”

Одною з головних підсистем будь-якої системи навчання є система контролю рівня засвоєння знань, тобто тести. В сучасних системах тестування застосовуються наступні форми питань:

1. Закрита форма (Multiple Choice)
2. Відкрита форма (Fill-in-the-Blank)
3. Форма на встановлення порядку (Order determining)
4. Форма на встановлення відповідності (Matching)
5. Вільна відповідь

Розглянемо кожну форму детальніше.

Закрита форма пропонує вибрати з декількох (бажано більше двох) відповідей, які пропонуються, правильну. Існує різновид закритої форми, де треба обрати більше однієї правильної відповіді. *Відкрита форма* пропонує доповнити фразу шляхом дописування слова або групи слів. *Форма на встановлення порядку* включає завдання типу “побудуйте правильну послідовність”, пропонуючи з зазначеного набору понять вибрати необхідні і вказати послідовність, за якої вони утворюють формулювання якогось закону, визначення якогось поняття і інше. *Форма на встановлення відповідності* потребує встановлення, яким твердженням з однієї частини відповідають твердження з другої частини. *Вільна відповідь* включає такий підхід. Студент отримує питання і повинен написати відповідь на нього в текстове поле. Вчитель потім має можливість перевірити надруковану інформацію і виставити бали за цю відповідь. Цей тип питання зручно використовувати для ґрунтовної перевірки студентом знань і навичок, але потребує перевірки вчителя. Тест, який містить питання даного типу, перевіряється за стандартними правилами, проте бали за дані питання не виставляються. Вчитель при перегляді статистики складання тестів має можливість продивитися дані відповіді і виставити бали за ці питання.

Так, для іспиту з курсу „Основи дискретної математики”, на якому, згідно програми курсу, студент може заробити 40 балів, і котрий складається з однієї задачі і двох теоретичних питань, модель виглядатиме так:

ЗАВДАННЯ

Мета роботи: **Перевірити остаточні знання студента**

Розділи навчального матеріалу що перевіряються або закріплюються:

Основи дискретної математики

Необхідні для роботи об'єкти: **Аудиторія, система**

Зміст роботи: **Розв'язування задач і відповіді на теоретичні питання**

Результат на 100%, балів: **40**

Додаткові відомості: **див. Програму курсу**

ЗАДАЧА

ТИП: **Текст**

ПОСТАНОВНИК: **Банк задач**

ОЦІНЩИК: **Викладач**

Результат на 100%, балів: **15**

ЗАДАЧА

ТИП: **Текст**

ПОСТАНОВНИК: **Банк Білетів**

ОЦІНЮВАЧ: **Викладач**

Результат на 100%, балів: **15**

ЗАДАЧА

ТИП: **множинний вибір**

ПОСТАНОВНИК: **Тестувальник**

ОЦІНЮВАЧ: **Тестувальник**

Результат на 100%, балів: **10**

При цьому система знає лише насправді суттєві для неї речі: хто ставить завдання, хто його оцінює, скільки балів воно коштує. Для здійснення оцінки викликаються методи відповідних об'єктів.

Якщо постановник—внутрішня системна сутність, така як Тестувальник, то система викличе метод Тестувальника: “Дати завдання слухачеві X з тесту №Y курсу Z”...

Тестувальник запитає у системи потрібну інформацію про стан слухача та умови, візьме зі своєї бази запитань потрібне і запише його для слухача. Система повідомить слухача про отримання завдання.

Якщо постановник—людина, то система повідомить постановнику (наприклад, ел. поштою): „Сьогодні треба дати завдання слухачеві X з тесту Y курсу Z. Для цього потрібно заповнити форму ...”

Якщо оцінювач—системний об’єкт, то після виконання завдання вона викличе відповідний метод оцінювання, базуючись на значенні типу завдання.

Якщо оцінювач—людина, то після виконання завдання вона буде повідомлена „Професоре, оцініть сьогодні слухача X. Для перегляду відповіді відвідайте <http://www.....> , а потім заповніть поле „оцінка””.

В наступних розділах буде наведено конкретніші приклади організації інтерактивної роботи викладача зі студентом в навчальному колаборативному середовищі.

Висновок до розділу 1

При прийнятті запропонованої моделі багато можливостей можна реалізувати поза системою, даючи системі лише необхідну для її роботи інформацію, таку, як значення оцінки, „місце зустрічі” у чаті тощо, що загалом відповідає „принципам проектування” [3] і значно скорочує час розробки.

Наведена модель дозволяє виділити мінімум основних суттєвих для системи властивостей взаємодій, що дозволяє значно простіше реалізувати систему електронної освіти, покладаючи другорядні для системи електронної освіти функції на інші сутності. При такому підході розробка полягатиме значною мірою у інтеграції відомих засобів з об’єктами системи через визначений набір інтерфейсів.

2 Принципи побудови програмних систем підтримки електронної освіти

Електронна освіта на сьогодні – важлива і невід’ємна частина навчального процесу у багатьох провідних університетах світу. Ефективна реалізація цієї можливості насамперед залежить від розвитку засобів дистанційного доступу до продуктів інформаційного забезпечення навчального процесу, тобто, розвитком комунікаційних технологій і засобів роботи з розподіленими базами зберігання знань. Потреба у такій системі дистанційної освіти назріла і в багатьох університетах та вищих закладах освіти в Україні [19,20].

Зрозуміло, що упровадження Internet в навчальні заклади мусить якось позначитися на методиці викладання того або іншого предмету. Internet - це не тільки засіб, де учні можуть одержувати додаткову інформацію. 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 роботи досить важко, якщо можливо. Тому завданням роботи є опис розробленої підсистеми керування.

Виходячи з цього, вимоги до системи дистанційної освіти описано саме з цієї точки зору. Загальні вимоги до великих систем подібного типу поділяються на дві категорії: вимоги до архітектури системи та вимоги до реалізації системи [21].

Кожен клас (або об’єкт) повинен грати чітко визначену роль в ПСПЕО. Це допоможе як ефективно забезпечити повторне використання коду, так і чітко організувати процес розробки системи, розподіливши його між максимально можливою кількістю розробників. Яскравим прикладом розділення завдань за набором умінь може слугувати відокремлення програмного коду і 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, що неодмінно будуть змінюватися. Це означає, що (як було зазначено вище) потрібно забезпечити простий процес додавання нової та зміни існуючої функціональності. Лише в цьому випадку буде виконуватися вимога розширюваності системи.

Розбиття структури системи на модулі, які взаємодіють між собою за гарно 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 бути такою, що має можливість гнучкого налаштування. Таке налаштування повинне звести до мінімуму необхідність втручання в код компонентів системи при невеликих змінах вимог до системи. Такі зміни вимог не повинні викликати змін у коді окремих компонентів. В 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 по підтримці багатомовності, розширювати їхні інтерфейси тощо. Для забезпечення можливості гнучкого налаштування системи, необхідно глибоко продумати архітектурну модель системи, обираючи між багатьма можливостями налаштувань і кількістю додаткового коду, який необхідний для цього.

Оскільки системою будуть підтримуватися найрізноманітніші дані, більша частина яких може бути доступна тільки певній категорії користувачів, то є природним вимагати від ПСПЕО реалізації певних заходів для безпечної обробки даних. До таких заходів відноситься забезпечення “відокремленості” даних, що відносяться до різних користувачів системи – кожен з них може бачити й редагувати тільки те, що йому дозволено. До цієї категорії також відноситься 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’єкта 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 декілька таких, які стосуються способу реалізації системи. До таких вимог відносяться: гетерогенність системи, можливість керування навантаженням серверів та ресурсами, збереження стану і 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 їх 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.

2.1 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 ще однієї праці, а враховуючи динаміку їх 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 Інтернет. Основна його задача – 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. Окрім того існує обмеження сесії, дуже обмежена гнучність таких програм та ін. Тому від цієї простої технології прийдеться 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 знаходяться 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 гарно 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 ж самі функції. Контейнер використовується і в інших Java-технологіях, що об'єднані специфікацією J2EE (Java 2 Enterprise Edition). JSP органічно впливається 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, мова 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 застосунків (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) покладена двійкова структура 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 являє собою екземпляр 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 свою 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 стороні сервера – іншими словами, наявність монітора транзакцій. Для DCOM – це Microsoft Transaction Server (MTS). Він забезпечує потрібну функціональність – керування транзакціями, створення пула ресурсів (в тому числі з’єднань з базами даних), створення і керування пулом серверних об’єктів, активацію і 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 була навіть створена спеціальна технологія 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 і дружня 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.

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.

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 різноманітність задач, що поставлені перед ПСПЕО, для кожного класу таких задач треба використовувати таку технологію, яка є найбільш доцільною. Усі ці технології мають гарно 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 даної роботи, найбільш доцільно використовувати симбіоз RMI, CORBA, EJB та XML для створення серверної частини системи. Кожна з цих технологій має гарно продуману розширюваність, а всі вони разом дозволяють задовольнити всі 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 (і інших технологій).

2.2 Загальні підходи та схеми побудови ПСПЕО

На даний момент і у світі взагалі і в Україні зокрема існує просто величезна кількість програмного забезпечення – систем, серверів, підручників, інтерактивних курсів розроблених як засоби ПСПЕО [22]. Саме тому починати розробку чогось нового з одної

сторони досить важко, а з іншої – легко, якби парадоксально це не звучало. Важко тому, що практично не можливо придумати уже щось принципово нове не реалізоване десь у 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.

В цьому розділі ми спробуємо розглянути загальні підходи та схеми, що використовуються при побудові систем дистанційної освіти [23,24]. На наш погляд варто зразу відкинути ПСПЕО, які реалізовані як масив статичних веб-сторінок. Можливо, це і може виглядати, 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, електронний курс (ЕК) – автономний програмний модуль, завданням якого є провести спудея від початку курсу до його кінця (або відповідно логічної частини курсу).

Вся робота студента, після замовлення ЕК буде полягати звісно лише в спілкуванні з ним. Тобто фактично не потрібно буде створювати ніяких 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. В залежності 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 більш складна, яка включає в себе вбудовані функції розповсюдження мультимедіа контенту, онлайн-лекцій, календарних групових занять і т. д.

2.3 Структуризація знань

Сукупність знань є нескінченною масою понять, так чи інакше взаємозв'язаних між собою. Кожна наука, а також розділи науки, чи різні погляди на те або 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 і спудея (суб'єкта і об'єкту) є щось нове, не сформульоване

єдиним терміном. З одного боку – це технічний посередник (комп'ютер), з іншого, – це нескінченне джерело знань, що працює в режимі миттєвого отримання будь-якої інформації (WWW). У зв'язку з цим виникає необхідність, (окрім діалогу лектора і спудей, контролю лектора за спудеєм), створити інструмент, що забезпечує не діалог, а спілкування утрюх, коли всі сторони трикутника стають рівноцінними: лектор – WWW – спудей. Структура подання учбового матеріалу абсолютно міняється в порівнянні з класичною схемою, створюються електронні репозиторії навчальних матеріалів. На сьогоднішній день немає чіткої концепції формування подібних курсів. Тому при їх організації часто застосовуються універсальні схеми, єдині для різних типів знань. На даний момент більшість існуючих ПСПЕО є замкнутими системами з жорсткими моделями, що не дозволяють адаптувати конкретний тип знань до конкретного формату представлення інформації але майбутнє - за відкритими системами з загально визнаними форматами подання знань.

Простір знань є нескінченною кількістю понять, визначень, тверджень, умінь та інших структурних вузлів (СВ), взаємозв'язаних між собою структурними зв'язками різного характеру. Структурні вузли можуть бути утворені і так званими модулями знань (система МРІТО) Даний простір знань може бути розділений на замкнуті області знань, усередині яких існують однорідні зв'язки однорідного характеру (див рисунок 2.1).

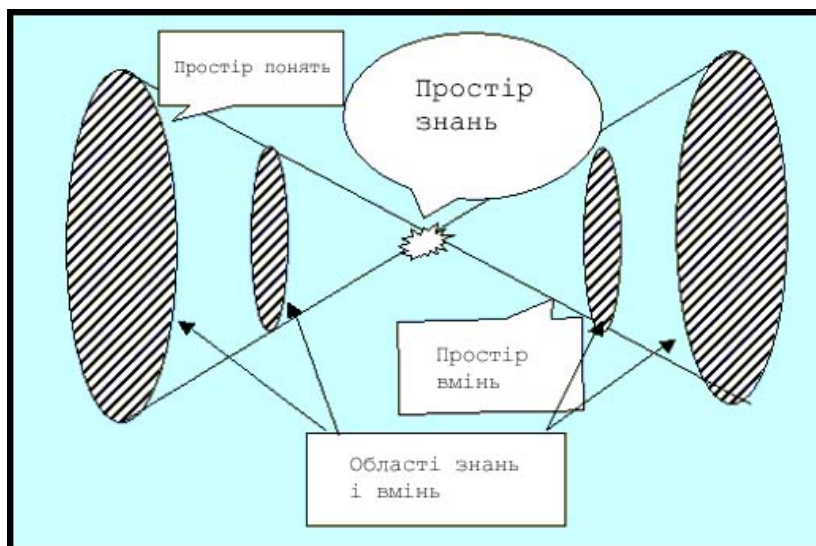


Рисунок 2.1. Простір знань

У загальному просторі знань можна виділити зрізи (області знань), які мають чітку замкнуту **структуру** зв'язків. Між даними областями також існують структурні зв'язки, об'єднуючі весь простір знань. Ніж більше понять, або умінь полягає в даній області знань, тим ширше 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 даному зрізі(області знань)

Головною задачею при вивченні **структурних зв'язків** між областями, а також зв'язків усередині області між структурними вузлами є точна диференціація понять, створення єдиної замкнутої логічної схеми. В супротивному випадку область знань не може бути представлена у вигляді окремого структурного елемента і виникнуть труднощі при вивченні структурованих вузлів. Для цього необхідна технологія.

Технологія структурного аналізу Інтернет-простору і виділення окремих областей знань і служить способом складання депозитаріїв навчальних матеріалів (курсів). Зрізи інформаційних ресурсів ІНТЕРНЕТУ представляють області знань. А структура області знань представляє характер організації дистанційного навчання і складання інформаційних модулів (понять) в логічну структуру курсу

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

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

Робота системи, 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, реалізований через інтерфейс користувача. Функціонування системи ґрунтується на взаємодії об'єктів та на зміні їх властивостей. Базовим об'єктом, що приймає участь у більшості взаємодій є курс. З курсом пов'язані учбові матеріали, календарний план, форуми, групи слухачів та ін.

2.4 Розробка загальної схеми керування ПСПЕО

Отже, після аналізу існуючого досвіду у світі та Україні на ниві побудови ПСПЕО приступимо до головної частини роботи – розробці загальної схеми керування ПСПЕО навчального закладу.

2.4.1 Алгоритм навчання

Спочатку розглянемо два алгоритми за якими буде діяти система – алгоритм вступу на курси ЕО та алгоритм навчання, зображених на рисунку 2.2

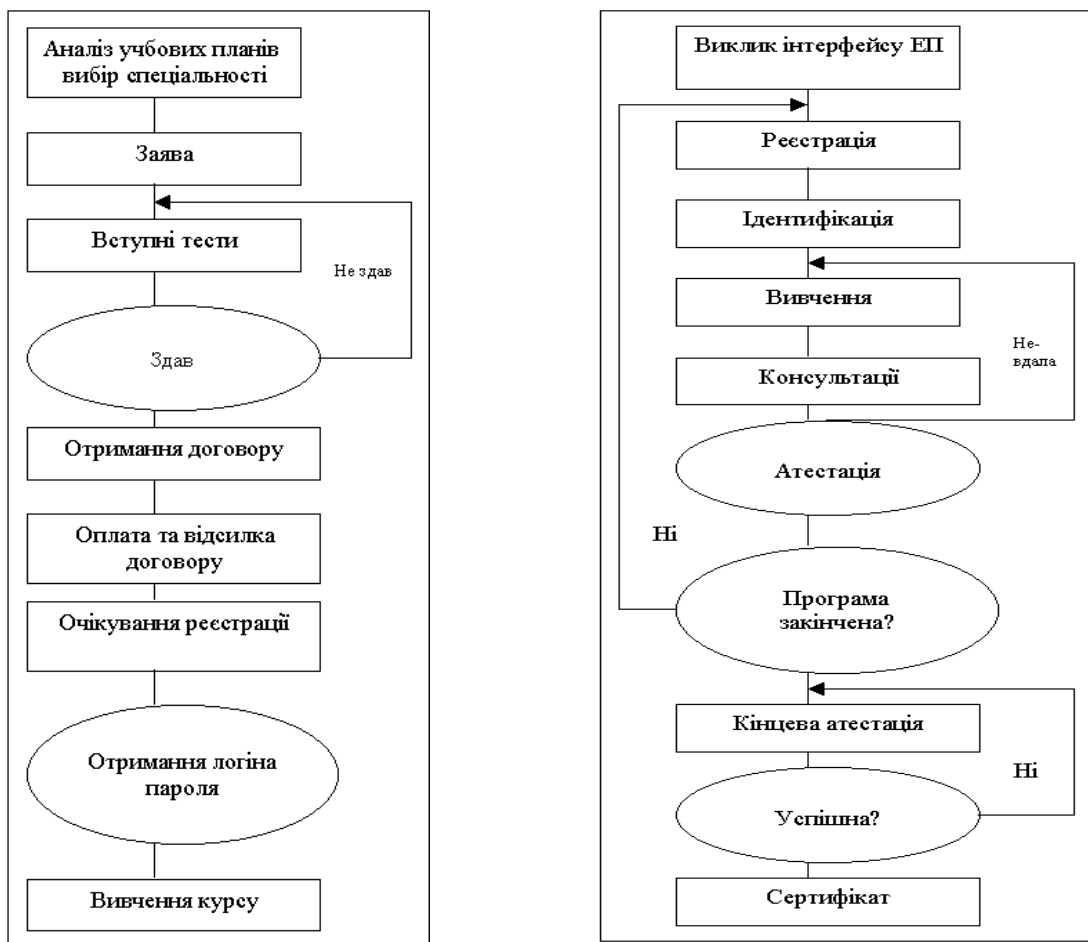


Рисунок 2.2. Алгоритм зарахування та навчання

Перший крок – це вступ до освітнього закладу. Користувач заходить на статичну звичайну сторінку закладу і знайомиться з інформацією про навчання. Ці сторінки є насправді частиною іншої системи – а саме системи керування навчальним процесом освітнього закладу і в загальному випадку 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 чи case-методи чи звичайні методи навчання з відповідними тестами після кожного розділу для закріплення знань. Взагалі, ЕК – це **абсолютно автономна** система, яка використовує свої власні методи та схеми навчання, довільні технології та мови програмування. Основне для нього відповідати клієнт-серверній архітектурі ПСПЕО та працювати за стандартним протоколом, який викладений далі.

2.4.2 Клієнт-серверна архітектура та розробка протоколу

Система керування – це одна з декількох підсистем у комплексі ПСПЕО. Розглянемо загальну схему та виділимо базові підсистеми. На рисунку 2.3 показана така система.

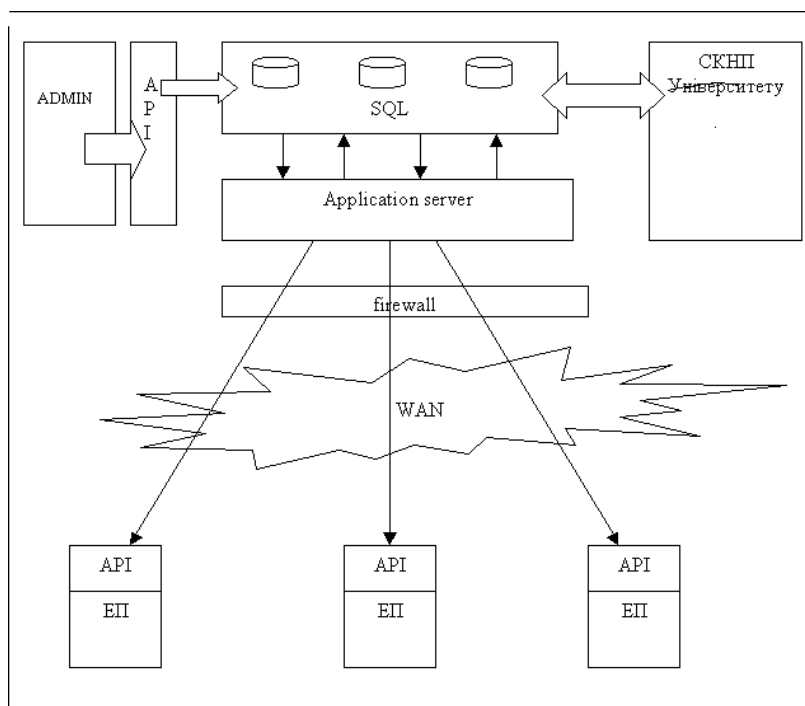


Рисунок 2.3. Структура ПСПЕО

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 спілкування можливе або через прямий інтерфейс до бази даних (поганий варіант – відкидаємо зразу),

або за допомогою спеціалізованого API. Розробці та специфікації такого API присвячений даний розділ.

ЕК зв'язуються не напряму з сервером баз даних, а зі спеціалізованим сервером який і 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 уже будуть частиною цієї системи. Проте з такою ж вірогідністю можна сподіватися що такої системи і не буде, або ПСПЕО буде запроваджена скоріше чим СКНПР. Тому розробляти такі бази даних потрібно, цьому і присвячений наступний розділ.

2.4.3 Опис баз даних СКНП

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

Головною таблицею тут будуть користувачі системи (Таблиця 2.1).

Таблиця 2.1. 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

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 буде звичайно залежати від конкретних задач, але в загальному може виглядати приблизно так, як зображено на таблиці 2.2.

Таблиця 2.2 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

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 здав/нездав) і як завгодно складними, залежно від складності поточних задач.

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

2.4.4 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(Таблиця 2.3).

Таблиця 2.3. Формат 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 ціллю віддати для обробки на сервер статистику 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(arcenoid11)
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 та зовнішніх програм для зв'язку розрізнених типів ЕК з платформою – сервером ПСПЕО. Клієнтський API буде складатися з набору таких програм та протоколу взаємодії, описаного у попередньому розділі. В загальному випадку головним завданням такого API буде аутентифікація користувачів та курсів на сервері, їх 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 даний час це не є складною проблемою. Весь ір-трафік пропонується пропускати лише через ssl тунель, використовуючи для цього засоби stunnel. Така організація є досить звичною для сучасного системного адміністратора та не потребує ніяких додаткових зусиль. Проте, якщо така схема буде не можливою потрібно використовувати допоміжну зовнішню утиліту шифрування. Можна використовувати готові безкоштовні продукти (Mypgp).

2.5 Організація взаємодії між автоматизованою системою управління навчальним закладом та системою керування навчанням

Автоматизована система управління навчальним закладом (далі АСУНЗ) що пов'язана з системою керування навчанням (далі СКН) через координаційну систему EPCS, що з метою узгодження дій користувачів двох систем для уникнення конфліктних ситуацій. Координаційна система являє собою зовнішнє незалежне веб-застосування, яке використовувало інформацію з баз даних систем MAMS і Moodle. В даному розділі описується процес додавання функціональності системи EPCS до функцій системи MAMS, тобто власне інтеграція автоматизованої системи управління навчальним закладом із системою управління навчальним контентом.

2.5.1 Огляд АСУНЗ MAMS

АСУНЗ MAMS — багатомодульна система, побудована з використанням, зокрема, таких технологій як Spring, Tapestry та Hibernate. На рівні програмного проекту вона

складається з чотирьох частин: AAS (Agent Automation System — Автоматизована агентна система), NI (NewtonIdeas HR Automation System — Автоматизована система керування людськими ресурсами NewtonIdeas), EIAS (Educational Institution Automation System — Автоматизована система освітнього закладу) та UKMA (University of Kiev-Mohila Academy Automation System — Автоматизована система університету «Києво-Могилянська академія»).

Проект «УКМА» цікавить нас в першу чергу, оскільки містить реалізацію функцій, що відповідають робочим обов'язкам передбачених системою адміністративних ролей, таких як «завідувач відділу кадрів», «секретар приймальної комісії», «методист факультету», «методист кафедри» тощо. Завданням системи EPCS було координувати роботу між викладачем та методистом факультету, тому нас цікавить саме ця роль. Наша задача: додати до переліку інструментів методиста зчитування оцінок за предмети, викладання й оцінювання яких здійснюються в СКН Moodle. Для цього потрібно зробити дві речі: зв'язати MAMS із базою даних СКН Moodle та написати клас, що керує заповненням відповідної веб-сторінки.

2.5.2 Зв'язок із базою даних СКН Moodle

Щоб навчити систему працювати ще з однією базою даних, треба зробити таке:

1. Дописати в конфігураційні файли усю необхідну інформацію про неї, а саме: адресу jar-файлу драйвера взаємодії з тією СКБД, під якою розгорнута БД СКН Moodle; тип БД (MySQL, MS SQL, Oracle тощо); назву БД; адресу сервера баз даних; ім'я та пароль користувача, від чийого імені буде здійснюватися доступ до бази даних; діалект Hibernate, який вказує цій бібліотеці, з яким саме типом БД ми працюємо; клас, що містить JDBC-драйвер для роботи з потрібною СКБД MySQL; повну адресу бази даних для доступу за протоколом JDBC.

2. Створити клас *MoodleAwareAasDataSource*, який в залежності від контексту, встановленого методом *setCurrentmoodlecontextholder* класу *MoodlePerspectiveContextHolder*, рівним одному зі значень перелічуваного типу *MoodlePerspectiveMode*, спрямовує запити щодо обробки даних до того чи іншого джерела даних. Клас *MoodleAwareAasDataSource* є нащадком базового класу *org.springframework.jdbc.datasource.AbstractDataSource*, який з'явився в каркасі Spring лише починаючи з версії 2.0.1, тому MAMS було переведено на використання Spring 2.8.0. На разі перелічуваний тип *MoodlePerspectiveMode* може приймати лише два значення: DEFAULT (база даних АСУНЗ MAMS) та MOODLE (база даних СКН Moodle), але за потреби перелік можна розширювати.

3. На базі класу *MoodleAwareAasDataSource* створити джерело даних, з яким буде працювати система. При цьому передбачити два режими його роботи: з оригінальною базою даних системи MAMS та з БД СКН Moodle.

4. Створити для кожної з таблиць бази даних СКН Moodle, яким адресуватимуться запити, відповідні Java-класи, екземпляри яких відображатимуть окремі записи в цих таблицях, та hbm.xml-файли відображення (mapping files), в яких задаються відповідності між атрибутами цих класів і полями таблиць.

5. Змінити конфігурацію Java-бобу «*aasSessionFactory*», який базується на класі *org.springframework.orm.hibernate3.LocalSessionFactoryBean* і являє собою локальний екземпляр інтерфейсу *SessionFactory* з бібліотеки Hibernate, відповідального за створення сеансів зв'язку з джерелами даних. Його атрибут «*mappingResources*» треба доповнити указниками на створені на попередньому кроці файли відображення.

2.5.3 Функція внесення результатів сесії з Moodle

Доповнимо головне меню методиста факультету (рисунок. 2.4) пунктом «Внести результати сесії з Moodle».

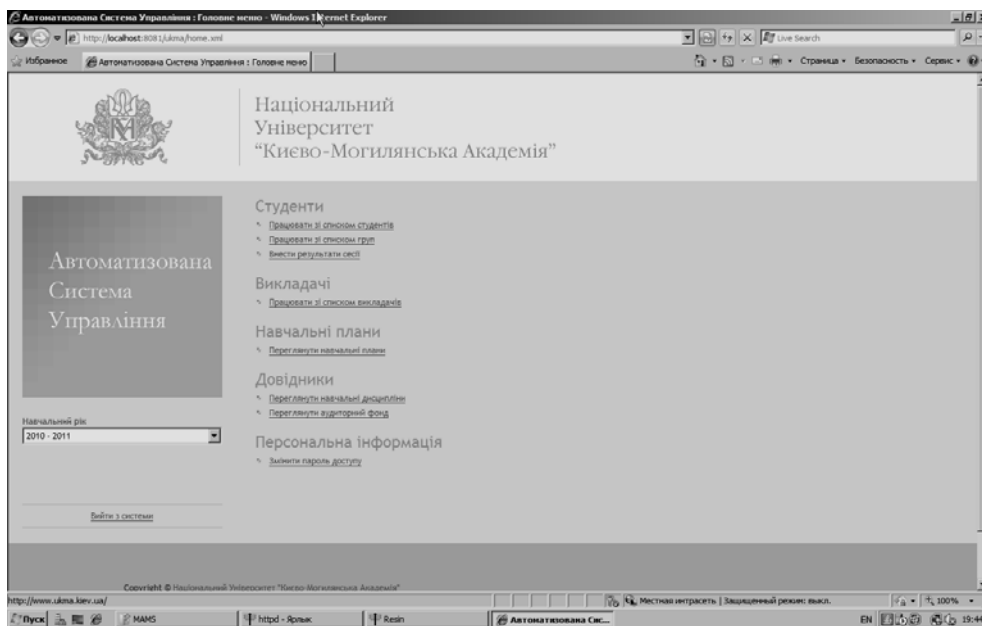


Рисунок 2.4 Головне меню методиста факультету

За обробку цього пункту відповідають одразу кілька файлів:

- *MoodleCourseResults.html* — HTML-код сторінки, яка завантажується при натисканні на пункт меню;
- *MoodleCourseResults.java* — java-код класу і його методів, що відповідають за роботу елементів веб-сторінки;

- *MoodleCourseResults.page* — xml-документ, що містить специфікацію веб-сторінки із зазначенням того, які її елементи якими саме атрибутами і методами вищевказаного Java-класу обробляються; потрібний для обробки сторінки каркасом Tapestry;
- *MoodleCourseResults.properties* — xml-документ, в якому зберігаються властивості елементів сторінки, зокрема пов'язані з ними рядки тексту в форматі Юнікод;
- *MoodleCourseResults.script* — містить тексти використовуваних на сторінці скриптів.

Інтерфейс сторінки «Внести результати сесії з Moodle» аналогічний інтерфейсу сторінки «Внести результати сесії» і відрізняється лише відсутністю випадаючих списків для вибору року навчання і триместру викладання предмета. При початковому виклику сторінки випадаючий список «Предмет» заповнюється назвами всіх курсів, зареєстрованих у СКН Moodle. Коли користувач обирає один з них, сторінка перезавантажується, і цього разу заповнюється список «Група». Після обрання групи з'являється таблиця оцінок студентів, що до неї входять. Відповідальність за зчитування даних та їх виведення в потрібних місцях бере на себе каркас Tapestry. Від програміста вимагається лише суворо дотримуватись правил іменування атрибутів та компонентів сторінки, інакше вона не зможе завантажитись (компілятор такі проблеми не виявляє).

Підсумковий вигляд сторінки наведено на рисунку 2.5

Студент	Проміжний тест - 1	Проміжний тест - 2	Проміжний тест - 3	Підсумковий тест	Всього:
One Student	25.0	25.0	25.0	5.0	80.0
Two Student	0.0	0.0	0.0	0.0	0.0
Zero Student	25.0	0.0	25.0	20.0	70.0

Рисунок 2.5 Таблиця оцінок для обраної групи

Структура таблиці спеціально підібрана так, щоб повністю відповідати формату звітів про оцінки, які генерує СКН Moodle (рисунки 2.6).

Reading of Programs : Перегляд: Grader report

Moodle ► RP ► Оцінки ► Перегляд ► Grader report

Choose an action ...

Grader report







Reading of Programs						
Ім'я / Прізвище	1st Test	2nd Test	3rd Test	Final Test	Course total	
 Student Five	0,00	25,00	25,00	-	50,00	
 Student Four	25,00	0,00	25,00	-	50,00	
 Student One	25,00	25,00	25,00	5,00	80,00	
 Student Three	0,00	25,00	0,00	-	25,00	
 Student Two	0,00	0,00	0,00	-	0,00	
 Student Zero	25,00	0,00	25,00	20,00	70,00	
Overall average	12,50	12,50	16,67	12,50	45,83	

Рисунок 2.6 Звіт про оцінки в СКН Moodle

Як і в системі EPCS, у разі, якщо обраний курс якраз оцінюється викладачем, групу обрати не вдасться, а на сторінці з'явиться попередження (рисунок 2.7).

Головне меню : Студенти : Внести результати сесії з Moodle

Пов'язані задачі

Звіти

Заліково-іспитова відомість

Предмет

Reading of Programs

Обраний курс зараз оцінюється викладачем Glibovets Mykola і тимчасово недоступний. Будь-ласка, спробуйте пізніше.

Вибрати

Рисунок 2.7. Попередження про те, що курс зараз зайнятий

Інформація про те, зайнятий курс чи ні, береться з доданої до бази даних СКН Moodle таблиці «epcs_course_results», структура якої аналогічна структурі таблиці «active_tasks» БД «epcs» [25, С. 94]. Вона зв'язана з таблицею предметів, тому перевірка на зайнятість не вимагає виконання додаткового запиту. Достатньо в файлі відображення, що відповідає таблиці предметів (в БД СКН Moodle це mdl_course), вказати таке:

```
<set name="activeTasks" lazy="false">
  <key column="resource"/>
  <one-to-many class="MOODLE_epcs_active_tasks"/>
</set>
```

Атрибут *lazy* = “*false*” означає відмову від так званого «лінивого» виклику, тобто такого, що вимагає окремого запиту, тому при першому ж зверненні до таблиці предметів для кожного з них буде побудована множина відповідних йому активних завдань. Якщо вона виявиться порожньою, значить, курс вільний.

2.6 Програмна система підтримки колаборативного навчального середовища

2.6.1 Огляд інструментальних засобів підтримки дистанційного навчання

Помітний прогрес в останні роки спостерігається в такій сфері дистанційної взаємодії в мережі Інтернет як дистанційна освіта. Починаючи з вісімдесятих років, вона пройшла шлях від перших підходів до навчання за допомогою комп'ютера до інтерактивних веб-сайтів, мультимедійних систем та розподілених універсальних і легко доступних баз даних. Дистанційна освіта відійшла від традиційної системи передачі знань, побудованої навколо вчителя, перетворившись на віртуальне навчальне середовище, орієнтоване на студента. Створення таких складних систем потребує спеціалізованих інструментальних засобів, тому не дивно, що досить швидко були розроблені універсальні програмні продукти для автоматизації їх побудови та, водночас, забезпечення сумісності між ними. До таких продуктів належать Moodle та ILIAS.

Moodle (Modular Object-Oriented Dynamic Learning Environment) — програмний продукт, що дозволяє створювати навчальні курси та веб-сайти, що базуються в мережі Internet. Він розповсюджується безкоштовно в якості програмного забезпечення з відкритим кодом (Open Source) за ліцензією GNU Public License. Це означає, що користувачі мають право копіювати, використовувати та, що найважливіше, змінювати програмний код, завдяки чому цей проект постійно розвивається. Він може бути встановлений на будь-якому комп'ютері, що підтримує SQL та бази даних типу SQL, та запущений на операційних системах Windows, Mac OS та багатьох різновидах Linux.

Основною рисою Moodle є його орієнтація на парадигму соціального конструкціонізму, тобто засвоєння знань в першу чергу за рахунок їх відтворення для інших. Підходить як для повністю дистанційного навчання, так і в якості допоміжного засобу для традиційного навчання. Вбудовані механізми каталогізації та пошуку дозволяють швидко знаходити потрібний курс та матеріали до нього, а інтегрована поштова служба дозволяє швидко розсилати копії повідомлень на форумі, відгуків вчителя на роботу студента або групи тощо.

До складу Moodle входять одинадцять модулів: „Урок” (Lesson), „Завдання” (Assignment), „Чат” (Chat), „Опитування” (Choice), „Форум” (Forum), „Глосарій” (Glossary), „Тест” (Quiz), „Ресурс” (Resource), „Огляд” (Survey), „Семінар” (Workshop) та „Вікі” (Wiki).

1. „Урок”. Урок — це послідовність сторінок, яку можна представляти у лінійному, як слайд-шоу, чи деревоподібному вигляді. Вони можуть містити не лише навчальні матеріали, але й поля для відповідей на запитання вчителя. Для збільшення інтерактивності модуль безпосередньо зв’язаний з глосарієм та системою оцінювання.
2. „Завдання”. Студенти завантажують свої роботи на сервер в довільному форматі. Оцінювання робіт може здійснюватись вчителем на одній сторінці в одній загальній формі, а система потім додає його коментар на сторінку персонального завдання кожного студента та надсилає електронною поштою повідомлення про надходження оцінки.
3. „Чат”. Засіб зручної синхронної взаємодії в текстовому режимі з підтримкою гіперпосилань, вбудованого HTML, зображень тощо.
4. „Опитування”. Використовується для голосування за щось чи в якості засобу зворотнього зв’язку зі студентами.
5. „Форум”. Можна створювати форуми різних типів, таких як „лише для вчителів”, „новини курсу”, „відкритий для всіх” та „один користувач — одна тема”. Передбачена можливість індивідуальної та централізованої підписки на сповіщення про появу нових повідомлень на форумі в цілому чи в окремих темах.
6. „Глосарій”. Один із модулів, що найяскравіше ілюструють перевагу Moodle над традиційним аудиторним навчанням. Кожний користувач може вести свій перелік термінів та їх визначень та додавати власні, які в майбутньому можуть стати основою завдання для їх наступників. Посилання на внесені в глосарій термін автоматично додаються до зв’язаних з даним предметом документів. Терміни можна групувати за категоріями для полегшення їх знаходження та кращої структурованості і наочності курсу в цілому.
7. „Тест”. Тести генеруються автоматично на основі складеної вчителем бази даних питань. Допускаються різні типи питань. Тести можуть бути тренувальними, з можливістю миттєвої перездачі, та такими, що складаються лише один раз.

8. „Ресурс”. Відображення будь якого електронного контенту, в тому числі Flash- та відеороликів, що зберігається локально чи віддалено. Зовнішні матеріали можна легко підключати до матеріалів за допомогою посилань чи безпосередньої вбудови в структуру сторінки.
9. „Огляд”. Використовуються вбудовані засоби аналізу онлайнових класів (COLLES, ATTLS), завдяки яким кожен студент може отримати звіт про свою успішність порівняно з середнім рейтингом класу.
10. „Семінар”. Надає вчителю широкі можливості для запровадження гнучкої системи оцінювання знань і роботи студентів. Реалізована функція надання учням зразків відповідей для тренувальних опитувань.
11. „Вікі”. Ще один елемент дійсно колаборативного навчання. Надає засоби колективної роботи над документами з веденням архіву всіх попередніх версій, які можна в будь-який момент часу відновити.

Як вже зазначалося, система швидко розвивається. Ведеться динамічна база даних пропозицій щодо вдосконалень системи, а детальна документація розробника дозволяє кожному налагодити систему під власні потреби.

Іншим потужним проектом з відкритим кодом є розроблена в Кельнському університеті система керування навчанням (СКН) ILIAS. Її версія 3.9.0 стала першою СКН з відкритим кодом, що отримала сертифікат відповідності 3-й редакції стандарту SCORM 2004. ILIAS дозволяє ефективно керувати курсами та навчальними матеріалами за допомогою стандартизованих інструментів та схем-шаблонів навчального та робочого процесів, в тому числі інтегрованих засобів навігації та адміністрування. Важливим елементом системи є особистий робочий стіл кожного користувача, на якому збираються всі потрібні для навчання та виконання поточних завдань ресурси. Як і в Moodle, є можливість створювати глосарії часто вживаних термінів для подальшого звертання до них при вивченні інших предметів.

Кооперативне навчання реалізується за допомогою вбудованого механізму групової взаємодії і не вимагає використання зовнішніх засобів. Всі необхідні інструменти керування користувачами та файлами, а також поштовий клієнт, чати та форуми вже є частиною системи. Реалізовано механізм керування подкастами для швидкого та зручного поширення відеозаписів лекцій чи навчально-тренувальних матеріалів, в яких використовується анімація.

Багато характерних для асинхронного типу навчання рис робить привабливим використання мультиагентних систем (МАС). МАС — це система, в якій декілька агентів можуть спілкуватися один з іншими, здійснювати обмін поточною інформацією та

взаємодіяти між собою. Цей напрямок розподіленого штучного інтелекту розглядає рішення однієї задачі декількома інтелектуальними системами, при чому задача розбивається на декілька підзадач, які розподіляються між агентами. Іншою сферою застосування МАС є забезпечення взаємодії між агентами, коли один агент може здійснити запит до іншого агента на передачу певних даних або виконання визначених дій, та передавання знань від агента до агента.

Кожний з учасників процесу представляється в системі у вигляді агента. Задачі виконуються паралельно, тобто, якщо предметна область представляється у вигляді спільноти агентів, то незалежні задачі можуть виконуватись різними агентами. При цьому контроль та відповідальність за дії розподілені між декількома агентами. Якщо один агент вийде з ладу, то система не припиняє функціонувати. Завдяки модульності МАС можна її легко нарощувати та видозмінювати, оскільки легше додати одного агента з потрібними властивостями, ніж перепрограмувати всю програму. Системи, які змінюють свої параметри з часом, можуть бути представлені сукупністю агентів.

При цьому враховуються особливості кожного студента, такі як попередній багаж знань, навички в роботі, зацікавленість в певних курсах, здатність до сприйняття матеріалу, кінцева мета навчання й т. д. Справа в тому, що для мобільного агента існує ще одна важлива мета — прогнозування потреб свого „хазяїна” на основі аналізу його інтересів.

Саме цього і потребує дистанційне освіта. Завдяки створенню моделі навчання, в якій присутній професор, одночасно забезпечується індивідуальний підхід до кожного студента й створюється віртуальне середовище для спільної групової роботи. При цьому для побудови ефективного процесу дистанційного навчання необхідно зважати на педагогічний досвід й знання про учнів взагалі, знання про предметну галузь та мати уяву про конкретного студента.

Саме тому МАС зручні для побудови комп'ютерних систем підтримки дистанційної освіти. По-перше, студенти віртуального класу в Інтернет знаходяться на великих відстанях один від одного, а кількість потенційних учасників необмежена. Через це статична і централізована система не відповідає нагальним вимогам, а розподілена мультиагентна система з персональними агентами для кожного студента навпаки, вельми приваблива. По-друге, класи за своєю природою динамічні, тобто навчальні матеріали і методології вивчення курсів, знання та майстерність активних студентів будуть з часом змінюватись. По-третє, студенти мають різну підготовку, і кожен студент — особистість, тому задля підвищення ефективності навчання його методологія повинна бути пристосована до особистості кожного студента і його знань. До того ж, студенти часто

реєструються в декількох курсах одночасно, тож необхідна координація навчання з різних тем. Нарешті, студенти мають тенденцію збиратися для обговорення як тем, що вивчаються, так і спільних інтересів. Тому мультиагентні системи стали багатообіцяючим зразком в освіті, і одним з найкращих програмних комплексів такого типу є система IDEAL.

IDEAL — це Web-базована розподілена мультиагентна системою навчання з триланковою архітектурою. Система пов'язує Web-клієнтів (для студентів) і основні інформаційні сервери (для навчальної системи і параметрів студента) разом з мультиагентним механізмом керування ресурсами. Інформація і агенти підтримуються розподіленою системою, яка складається з робочих станцій і накопичувальних пристроїв, які з'єднуються через високо-пропускні мережі. IDEAL розроблена з використанням поширених технологій, таких як Інтернет, WWW, програмних агентів і цифрових бібліотек.

IDEAL складається з ряду спеціалізованих агентів з різною кваліфікацією. Кожному студенту приписується унікальний персональний агент, який керує особистим профілем студента, включаючи його підготовку, стиль навчання, інтереси, обрані ним курси і т.д. Персональний агент спілкується з іншими агентами в системі через різні канали зв'язку. Онлайн-курс підтримується спільними зусиллями навчальних та курсових агентів. Курсові агенти керують матеріалами курсу і специфічними методами його опанування. Різні курсові агенти знаходяться на окремих сайтах, що забезпечує кращу ефективність, гнучкість і корисність. Навчальні агенти можуть спілкуватися з будь-яким курсовим агентом і часто вибирають найближчого (найдоступнішого) для покращення продуктивності. Крім того, курсові агенти виступають посередниками для комунікації між студентами.

Успіх системи IDEAL, що стала базовим стандартом розробки комп'ютерних систем підтримки дистанційної освіти, доводить, що мультиагентні системи є потужним та ефективним засобом побудови систем підтримки ДВІ. Саме тому в пошуках платформи для програмної реалізації нашої системи ми звернулися до них.

2.6.2 Агентні технології та середовище JADE

Сучасні мультиагентні системи відійшли від локалізованого до розподіленого типу взаємодії. Кожний агент — це процес, який володіє достатнім обсягом знань про об'єкт своєї діяльності та має можливість обмінюватися цими знаннями з іншими агентами. З

точки зору парадигми об'єктно-орієнтованого програмування агент можна розглядати як набір функцій у поєднанні з інтерфейсом, за допомогою якого він надсилає запити та отримує відповіді. Але від простої програми, запущеної кимось на виконання на віддаленому вузлі чи завантаженої в мережу, агент відрізняється в першу чергу механізмом самостійного формування власної мети, який виводить його на принципово новий рівень автономії [26 X.8]. Саме автономність та націленість на досягнення мети, яку агент здатен самостійно для себе згенерувати та в подальшому раціонально діяти, виходячи з неї, є визначальними рисами інтелектуальних агентів [27 X.3, 28 X.12].

Найчастіше в агентних технологіях використовуються такі мови:

- 1) універсальні мови програмування (Java);
- 2) мови, орієнтовані на знання
 - a. мови представлення знань (KIF);
 - b. мови переговорів та обміну знаннями (KQML, AgentSpeak, April);
 - c. мови специфікацій агентів;
- 3) спеціалізовані мови програмування агентів (TeleScript);
- 4) мови сценаріїв та скриптів (Tcl/Tk);
- 5) символічні мови та мови логічного програмування (Oz).

JADE [29 X.10] — це технологія проміжного рівня (middle-ware) для розробки та виконання однорангових децентралізованих (peer-to-peer) застосувань, в основі якої лежить агентна парадигма. Це проект з відкритим початковим кодом, повністю написаний мовою програмування Java з використанням таких розвинутих можливостей як Java RMI, Java CORBA IDL, Java Serialization та Java Reflection API. JADE Це програмна розробка, метою якої є створення та розповсюдження мультиагентних систем та застосувань, що відповідають стандартам FIPA для інтелектуальних агентів.

JADE спрощує розробку мультиагентних систем завдяки використанню специфікацій FIPA та інструментів, які підтримують фази виправлення помилок та розгортання системи. Ця агентна платформа може розповсюджуватись серед комп'ютерів з різними операційними системами, а її конфігурування може здійснюватись віддалено за допомогою графічного інтерфейсу. Процес конфігурування цієї платформи досить гнучкий, його можна проводити навіть під час виконання програм, для цього лише потрібно перемістити агентів з однієї машини на іншу. Єдиною вимогою цієї системи є встановлення на машині Java Run Time 1.4. Комунікаційна архітектура пропонує гнучкий та ефективний процес обміну повідомленнями, де JADE створює чергу та керує потоком ACL-повідомлень, приватними для кожного агента. Агенти мають доступ до черги

шляхом поєднання кількох режимів своєї роботи: блокування, голосування, перерви в роботі та співставлення з еталоном.

JADE пропонує графічний інтерфейс платформи адміністрування з використанням RMA агента. Цей агент показує стан агентної платформи (Agent Platform) та пропонує різноманітні інструменти адміністрування, виправлення помилок та тестування застосувань, що базуються на JADE.

RMA — це об'єкт Java, екземпляр класу *jade.tools.rma.rma*, що можна запустити: з командного рядка, як звичайного агента в цій системі (наприклад, командою *java jade.Boot myConsole:jade.tools.rma.rma*), або запустивши gui-інтерфейс (*java jade.Boot -gui*). На одній платформі можна запустити багато RMA-агентів. Кожний екземпляр такого агента буде мати своє локальне ім'я на відміну від агента-контейнера.

Dummy Agent — це простий і дуже корисний інструмент для відслідковування обміну повідомленнями між JADE-агентами. Цей агент дозволяє посилати ACL-повідомлення агентам, отримувати та керувати повідомленнями, що надходять від агентів, читати та зберігати отриману інформацію в файл.

Sniffer Agent — це Java-застосування, призначене для слідкування за повідомленнями, обмін якими відбувається в середовищі JADE. Sniffer повністю інтегрований в JADE та є достатньо корисним для виправлення помилок в поведінці агентів. Коли користувач вирішує відслідкувати агента або групу агентів за допомогою Sniffer-агента, то всі повідомлення, що надходять агенту/групі агентів, та всі, що надсилаються цим агентом/групою, перехоплюються та відображаються в GUI-інтерфейсі агента Sniffer. Користувач може переглядати кожне повідомлення та за потреби зберігати їх на диск як текстовий файл.

SocketProxyAgent — це JADE-агент, призначений для встановлення зв'язку з віддаленими клієнтами. Агент дозволяє підтримувати 50 паралельних підключень. Кожне повідомлення, що надходить, аналізується, перевіряється його адресат по лістингу агентних імен та передається агенту в пункт призначення.

DF (Directory Facilitator) agent. Дозволяє співпрацювати з іншими DF-агентами та контролювати (реєструвати, модифікувати та шукати агентів за їх характеристиками) всю мережу DF-агентів. Цей тип агентів пропонує іншим агентам сервіс т. зв. „жовтих сторінок” (“yellow pages”), на яких вони можуть зареєструвати свої послуги за допомогою DF-агента або „попросити” його знайти підтримку в інших DF-агентів. Останні можуть об'єднуватись та створювати федерації агентів-фасилітаторів.

2.6.3 Опис програмної системи підтримки взаємодії LMS і CMS

Оскільки КМ в нашій моделі включає в себе механізми створення сеансів та контролери рівнів, що регламентують доступ до ресурсів, природно уявити його як агента, до якого надходять запити від користувачів (як людей, так і прикладних програм чи інших агентів) на використання потрібного ресурсу. Іншим варіантом реалізації може бути сукупність агентів, кожен з яких відповідає за свій сеанс або рівень. В деяких випадках доцільно використовувати проміжний варіант, коли одні агенти відповідають кожен за свій сеанс або ресурс, інші — за два, за три тощо. Наприклад, у тому випадку, коли ми хочемо унеможливити керівництво одночасно кількома сеансами чи одночасне використання якихось ресурсів одним і тим самим користувачем.

Таким чином, в загальному вигляді багатоагентна система, що моделює роботу КМ колаборативної системи, складатиметься з таких агентів:

1) User_Agent — агент-користувач, що використовує ресурси системи для вирішення своєї чи спільної з іншими агентами задачі;

2) Session_agent — агент, що відповідає за створення сеансу, доступ до нього та його закриття. Повинен слідкувати за тим, щоб в кожний момент часу у сеансу був один і тільки один керівник, тобто користувач, який має право його закрити. Зазвичай це той самий користувач, що створив сеанс, і його примусовий вихід із системи супроводжується закриттям сеансу. Але у випадку дуже важливих сеансів доцільно зробити так, щоб цього не траплялося, і тоді контролер сеансу має взяти на себе керівництво сеансу доти, доки всі інші користувачі не від'єднаються від нього, або доки попередній користувач не ввійшов у систему знову. Інший можливий варіант — обирання з-поміж користувачів сеансу нового керівника за одним з алгоритмів обрання лідера.

3) Floor_agent — агент, що відповідає за надання взаємовиключного доступу до ресурсу. Також відповідає за коректне звільнення ресурсу в разі, якщо користувач, що зайняв його, аварійно від'єднується від системи. Може надавати можливість призупинення використання ресурсу без його звільнення, в цьому разі повинен слідкувати за тим, щоб пауза не затягувалася надовго, інакше примусово звільняє ресурс. Може підтримувати черги користувачів, тоді після звільнення ресурсу надає його тому, хто чекає найдовше. Реалізація рівневого контролю також може бути різною. Структура взаємодії між об'єктами системи показана на рисунку 2.8.

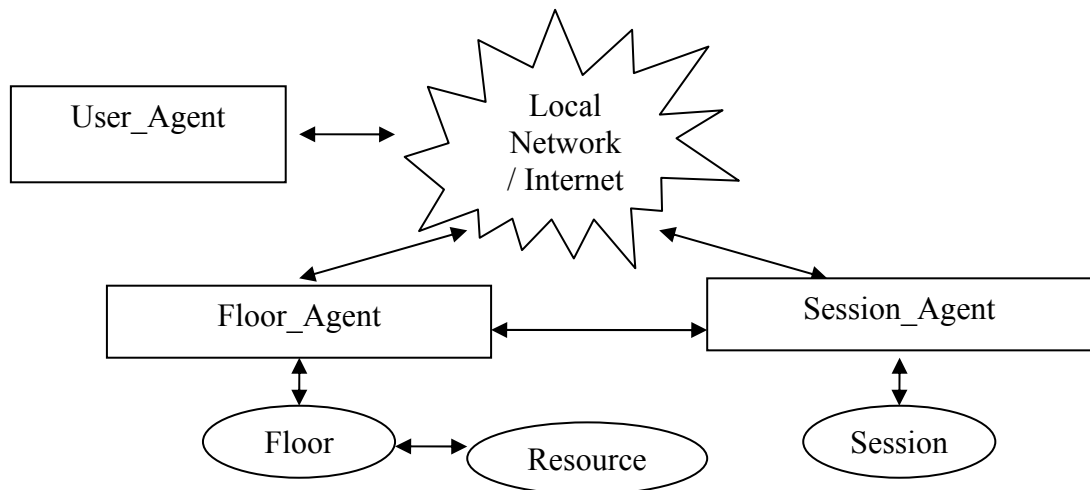


Рисунок 2.8 Структура взаємодії об'єктів в системі

Для побудови системи було використано агентну архітектуру JADE [30 X.1; 31 X.10]. Структура системи виглядає так:

а) пакет Session — складається з агента Session_agent та поведінки MaintainSessionBehaviour. Ця поведінка включає в себе всі операції агента по забезпеченню коректного функціонування сеансу, зокрема коректне від'єднання користувачів та стеження за тим, щоб в кожний момент часу сеанс мав одного і тільки одного чітко визначеного керівника. Поведінка MaintainSessionBehaviour реалізована на базі класу CyclicBehaviour, тобто виконується нескінченно довго і зупиняється лише одночасно з агентом, що її запустив.

б) пакет Floor — складається з агента Floor_agent та поведінки MaintainFloorBehaviour. Поведінка охоплює всі функції забезпечення взаємовиключного доступу до ресурсу та слідкує за тим, щоб жоден користувач не міг використовувати ресурс впродовж невизначено довгого часу. Якщо користувач призупиняє використання ресурсу, не звільнюючи рівень, і не поновлює роботу з ним впродовж певного наперед визначеного проміжку часу, то у випадку, якщо на доступ до цього ресурсу очікує інший користувач, контролер може примусово звільнити рівень для нього, а попереднього утримувача занести в чергу з пріоритетним правом на одержання ресурсу, коли від нього надійде команда „поновити роботу”. Поведінка MaintainSessionBehaviour також реалізована на базі класу CyclicBehaviour.

в) пакет User — складається з агента User_Agent та його поведінок CreateJoinSessionBehaviour і UseFloorBehaviour. Перша поведінка відповідає за створення сеансу або приєднання до вже існуючого та, відповідно, від'єднання від нього та завершення, якщо більше ніхто цим сеансом не користується. Друга поведінка відповідає за доступ до ресурсу та його використання з можливістю тимчасового призупинення без звільнення рівня (щоб потім не довелося повторно на нього чекати). Обидві поведінки

реалізовано на базі класу `TickerBehaviour`, тобто їх методи виконуються через рівні, наперед задані проміжки часу [32 X.11, 33 С. 30]. Таким чином моделюється певна послідовність дій користувачів, при чому запуск кількох агентів з різними періодами очікування дає змогу відстежувати взаємодію кількох користувачів та роботу координаційного механізму.

Робота системи відбувається наступним чином:

1) Стартують агенти-контролери сеансів та рівнів, реєструються в директорії DF (`Directory Facilitator` — агент, що надає іншим учасникам спільної роботи сервіс «жовтих сторінок», завдяки якому вони можуть знаходити та користуватися службами один одного, після чого додають до себе поведінки `MaintainSessionBehaviour` та `MaintainFloorBehaviour` відповідно. При цьому сеанси вважаються нествореними, а рівні — вільними.

2) Стартують агенти-користувачі, реєструються в директорії DF, після чого додають до себе поведінку `CreateJoinSession`. Отримавши від DF-агента список наявних сеансів, вони під'єднуються до того, який їм потрібен. Якщо сеанс ще не створений, користувач його створює і стає його керівником. Якщо створений – приєднується до нього як простий учасник.

3) Приєднавшись до сеансу, користувач додає поведінку `UseFloorBehaviour`. Якщо потрібний йому рівень із наданого DF-агентом переліку зайнятий, він заноситься в чергу з пріоритетним правом на доступ до ресурсу після його звільнення. Якщо рівень вільний, він його займає і працює доти, доки не звільнить сам або не буде від'єднаний системою.

4) Якщо користувач не є керівником сеансу та не займає рівень, він може без проблем вийти із системи. Якщо він є керівником сеансу, то може покинути систему лише тоді, коли ніхто інший цим сеансом не користується. В іншому випадку користувачу доведеться зачекати, доки всі інші учасники покинуть сеанс.

5) Після того, як користувач використав та звільнив ресурс та від'єднався від сеансу, його поведінки `UseFloorBehaviour` та `CreateJoinSessionBehaviour` знищуються і агент припиняє роботу.

Запускаючи довільну кількість агентів-контролерів сеансів та рівнів та користувачів, можна моделювати поведінку як завгодно великої системи.

3 Системи управління навчальним процесом та контентом: СУНП (MAMS) та СУНК (Moodle)

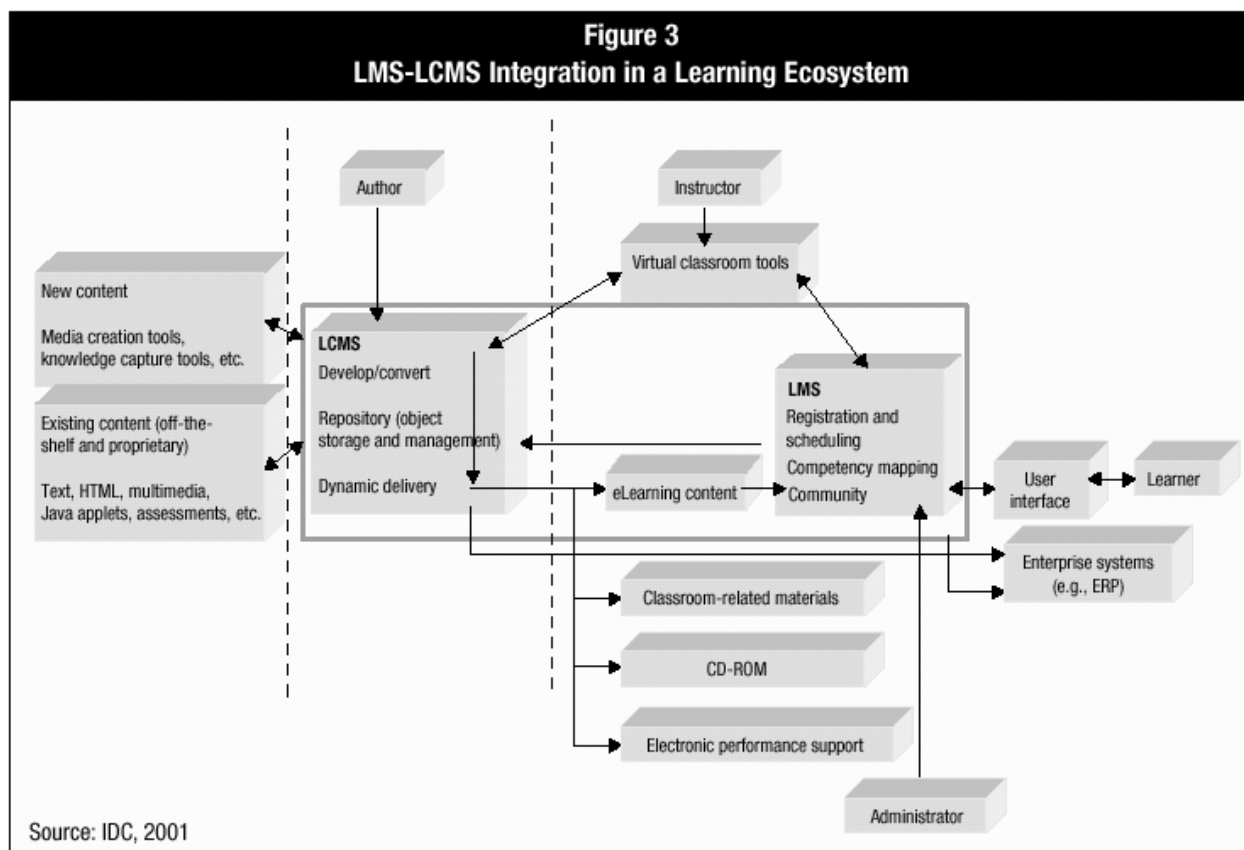
Часто на практиці плутаються між системою управління навчальним процесом (learning management system — LMS) та системою управління навчальним контентом (learning content management system — LCMS). Вони не тільки мають подібні назви — деякі виробники позиціюють СУНК як «нову хвилю» СУНП. По суті ж, СУНП й СУНК — це взаємодоповнюючі, але дуже різні системи, які обслуговуються різними фахівцями й призначені для рішення зовсім різних бізнес-завдань. Будучи інтегровані, ці дві системи й складають **колаборативне електронне навчальне середовище**.

СУНП — це високорівневе, стратегічне рішення для планування, проведення й керування всіма навчальними заходами в організації, включаючи онлайн-навчання, віртуальні заняття й курси, які веде викладач. Основне завдання — заміна ізольованих і розрізнених навчальних програм систематизованими методиками з оцінки й поліпшення компетентності й рівня продуктивності в масштабах організації. Наприклад, СУНП сильно спрощує міжнародну сертифікацію, дозволяє компаніям співвідносити навчальні ініціативи зі стратегічними завданнями, розробляти ефективні методики масштабу підприємства по керуванню знаннями. Спрямованість СУНП — це керування студентами, відстеження їхнього прогресу і росту в усіх типах навчальних заходів. Вона здійснює такі витратні адміністративні завдання, як наприклад генерація звітів і даних для HR й інших ERP систем, але в більшості випадків не використовується для створення навчальних курсів.

Всупереч цьому, основна направленість СУНК — це навчальний контент. Вона надає авторам, дизайнерам й експертам засоби для більш ефективного створення навчальних матеріалів. Головне бізнес-завдання, яке розв'язує СУНК — це створення необхідного контенту за необхідний час для задоволення потреб окремих студентів або груп. Перш ніж розробляти безпосередньо курс й адаптувати його для численної аудиторії, дизайнери створюють об'єкти, які багаторазово використовуються, й надають їх всім розробникам курсів в організації. Це виключає дублювання зусиль розроблювачів і дозволяє швидко «збирати» кастомізований контент.

3.1 СУНК в інфраструктурі СУНП

Оскільки СУНП прямо впливає на роботу тисяч студентів й управляє всіма аспектами навчання в організації, експерти рекомендують починати саме з СУНП, що потім може бути швидко інтегрована з СУНК(Рисунок 3.1).



Рисисунок 3.1 Діаграма інтеграції СУНП-СУНК

Дана діаграма опублікована у звіті IDC «Системи керування навчальним контентом: Виникнення нового сегмента ринку E-Learning» й ілюструє, як СУНП може запускати курси, розроблені в СУНК і вставляти СУНК-дані у звіти.

Звіт IDC «*Системи керування навчальним контентом: Виникнення нового сегмента ринку E-Learning*» дає пояснення: “СУНК і СУНП не тільки не протиставляються одна іншій, навпаки, вони дуже добре доповнюють одна одну. При тісній інтеграції, дві системи можуть обмінюватися інформацією, що дає чудові результати для поліпшення навчання для користувачів і надає ще більше відмінних інструментів для адміністраторів навчання. СУНП дозволяє управляти спільнотами користувачів, надаючи кожному з них для запуску необхідні об'єкти, що зберігаються й

підконтрольні СУНК. При виконанні контенту СУНК також запам'ятовує індивідуальний прогрес студентів, зберігає результати тестування й відправляє їх назад в СУНП для формування звітів.”

3.2. Порівняльна характеристика СУНП та СУНК

Обидві системи, СУНП й СУНК управляють змістом курсів і відслідковують результати навчання. Обидва інструменти можуть управляти й відслідковувати контент, аж до рівня навчальних об'єктів. Але СУНП, у той же час, може управляти й відслідковувати змішане навчання, яке включає онлайн контент, заходи у навчальних аудиторіях, зустрічі у віртуальних аудиторіях та інше. На противагу цьому, СУНК не може управляти змішаним навчанням, зате може управляти контентом на рівні грануляції нижче навчального об'єкта, що дозволяє організації більш просто здійснювати реструктуризацію й перенацілювання онлайн-контенту. Додатково, просунуті СУНК уміють динамічно будувати навчальні об'єкти відповідно до профілів користувачів або стилів навчання. Якщо обидві системи дотримуються стандартів XML, інформація може бути просто перенесена в СУНП на рівні навчальних об'єктів [34].

Нижче наведена таблиця, що базується в основному на даних дослідження Брендона Хала (Brandon Hall), підсумовує можливості й відмінності між двома системами.

Таблиця 3.1 Відмінності між двома системами

	СУНП	СУНК
Для кого призначена?	Всі студенти; організація	Розробники контенту; Студенти, яким потрібен персоналізований контент
В основному забезпечує керування:	Навчальний процес; вимоги до навчання; навчальні програми й планування	Навчальний контент
Управляє e-learning-ом	Так	Так
Управляє традиційними формами навчання	Так	Ні
Відстежує результати	Так	Так
Підтримує спільну роботу студентів	Так	Так

Включає керування профілями навчання	Так	Ні
Надає можливість HR й ERP системам використовувати дані навчання	Так	Ні
Розклад заходів	Так	Ні
Аналіз профілів компетенцій/карти знань	Так	Ні
Повідомлення про реєстрацію на курс, вимоги для перегляду й повідомлення про анулювання курсу	Так	Ні
Створення питань і управління тестами	Так	Так
Підтримка динамічного попереднього тестування й адаптивного навчання	Ні	Так
Підтримка створення контенту	Ні	Так
Організація контенту, що використовується багаторазово	Так	Так
Засоби документообігу для керування процесом створення контенту	Ні	Так
Розробка засобів навігації по контенту та інтерфейсу	Ні	Так

3.3 Системи управління навчальним контентом

Системи Управління Навчальним Контентом (СУНК) є веб-застосування, що означає, що вони виконуються на сервері та доступ до них здійснюється через веб-оглядач. Сервер, як правило, розташований в університеті чи у відділі, але фактично може бути будь-де в світі. Викладачі і студенти можуть отримати доступ до системи з будь-якої точки, де є мережа Інтернет.

Основною метою СУНК є надання викладачам інструментів для створення сторінки курсу та контролю доступу лише зареєстрованих студентів. З точки зору контролю доступу СУНК пропонують широкий спектр інструментів, які можуть зробити вивчення предмету більш ефективним. Вони забезпечують легкий спосіб завантаження та обміну матеріалами, ведення онлайн-дискусій та чатів, створення вікторин та опитувань, збору та перегляду завдань та виставляння оцінок.

- **Завантаження та обмін матеріалами**

Більшість СУНК пропонують інструменти для того, щоб можна було легко опублікувати контент. Замість використання редактору HTML та відправки документів на сервер через FTP, використовується веб форма для збереження

програми курсу на сервері. Багато викладачів завантажують на сервер програму курсу, конспект лекцій, завдання для читання, а також статті, для того щоб студенти могли мати доступ до цих матеріалів в будь-який момент.

- **Форуми і чати**

Онлайн форуми і чати забезпечують зв'язок і спілкування поза межами аудиторних занять. Спілкуючись на форумі, студенти мають більше часу для генерування відповідей, тому це дає можливість вести більш глибокодумні дискусії. Чати, з іншого боку, дають можливість швидко і просто спілкуватися з віддаленими студентами. Вони можуть бути використані для будь-яких потреб – починаючи від оголошень і закінчуючи цілими лекціями. Робочі групи студентів можуть використовувати онлайн дискусії для своїх проєктів.

- **Вікторини і опитування**

Онлайн вікторини та опитування можна моментально оцінити. Вони прекрасно справляються з задачею, коли потрібно сказати свою думку студентам про їх успішність, а також про їх розуміння навчального матеріалу. Багато видавців мають блок тестових запитань до розділу книжки, тому викладач курсу маркетингу, наприклад, може проводити щотижневі тести для контролю студентів і їх засвоювання матеріалу лекцій, а потім провести загальний тест по матеріалу всього курсу.

- **Збір та перегляд завдань**

Перевірка студентських домашніх робіт — важка та громіздка робота. Здача робіт онлайн — простий спосіб перевірки домашніх завдань. Також дослідження показали, що коли студенти можуть бачити, як прогресують їх одногрупники, це мотивує їх власну роботу та підвищує ефективність роботи.

- **Виставляння оцінок**

Онлайн журнал дозволяє студентам бачити оновлену інформацію про їх успішність. Він також є вирішенням проблеми заборони публікації особистих даних в публічних місцях. Отже, журнал дозволяє студенту бачити лише його власні оцінки, і ні за яких умов оцінки його колег. Також є можливість отримати оцінки в форматі Excel для детальних обчислень.

За останні п'ять років СУНК стрімко розвиваються. Зараз їх відсутність є критичною для багатьох вищих навчальних закладів. Один з найбільших вендорів, Blackboard, недавно стала відкритою.

3.4 Елементи систем управління навчальним контентом

IDC визначає СУНК як систему, що створює, зберігає, збирає й програв персоналізований навчальний контент у формі навчальних об'єктів. У той час як СУНП управляє всіма формами навчання в організації, СУНК концентрується на онлайнівому навчальному контенті, звичайно у формі навчальних об'єктів [35].

Навчальний об'єкт — це ізольована (замкнена) частина навчального матеріалу. Звичайно він включає три компоненти: досягнення мети, (що студент повинен зрозуміти або чого він досягне, коли завершить навчання), навчальний контент, що вимагається для досягнення мети (текст, відео, ілюстрації, структуровані слайди, демонстрації, симулятори) і різні форми оцінювання, що дозволяють визначити, досягнуто мету чи ні.

Навчальний об'єкт також включає метадані, або теги, що описують його зміст і використання в СУНК. Метадані можуть включати таку інформацію, як автор, мова, рівень версії й іншу. У документі «Методики, метадані й E-Learning» автор, Дейв Феєзі (Dave Feasy) з Eueropping Design, робить прогноз: «Зі збільшенням їхньої важливості, за нашим досвідом можна чекати вибуху метаданих, подібно до інформаційному вибуху, який ми зараз спостерігаємо, і до якого треба бути готовим. Керування величезними наборами метаданих незабаром може стати областю, замкненою на себе саму. Будемо сподіватися, що ця нова система не буде називатися LMMS (система керування навчальними метаданими).»

Так як же навчальні об'єкти використовуються для створення контенту? В СУНК навчальні об'єкти зберігаються в центральному репозиторії, і дизайнери можуть брати їх і вставляти у свої персоналізовані курси. Це дає перевагу розробникам і студентам, оскільки традиційні курси змушені містити більше контенту, ніж може засвоїти окремий студент, щоб зрозуміти суть теми. Дроблення контенту курсу на навчальні об'єкти й використання їх надалі як основу, дозволяє розробникам створювати потрібні курси в потрібний час. У кінцевому результаті це збільшує продуктивність, оскільки користувачам не потрібно витрачати зайвий час на те, щоб переривати купу матеріалу, що втратив актуальність.

Незважаючи на численні варіації можливостей СУНК, вона повинна включати наступні ключові компоненти:

- **Репозиторій навчальних об'єктів.** Репозиторій навчальних об'єктів — це центральна база даних, що зберігає й управляє навчальним контентом. Із цієї точки окремі навчальні об'єкти доступні користувачам або як окремі елементи або як частина в складі більшого навчального модуля, що у свою чергу може бути частиною повного курсу, цей процес визначається залежно від індивідуальних вимог до навчання. Кінцевий продукт може бути доступний через Web, CD-ROM, або в паперовому виді. Кожен об'єкт, залежно від вимог, може бути використаний кілька разів і з різними цілями. Інтегрованість контенту забезпечується незалежно від методу доставки. Для окремих елементів це забезпечується логікою програмного коду з використанням XML.
- **Програмне забезпечення автоматизованого ауторинга.** Це ПЗ використовується для створення навчальних об'єктів, які використовуються багаторазово і потім будуть доступні в репозиторії. Застосування автоматизує розробку, надаючи авторам шаблони й архівні зразки, що містять основні принципи дизайну навчального контенту. Використовуючи ці шаблони, автори можуть розробляти курси, застосовуючи наявні об'єкти з репозиторію, створюючи нові об'єкти, або використовуючи комбінацію з нових і старих об'єктів. Авторами можуть бути експерти по тематиці, дизайнери навчальних курсів, творці медіа-продукції і так далі. Цей інструмент також може бути використаний для швидкої конвертації існуючих в організації бібліотек навчального контенту, таких як додаткові аудіовізуальні матеріали, спеціальні інтерфейси й методики навчання. Автор може працювати в організації або здійснювати аутсорсингову розробку.
- **Інтерфейс відображення (програвання контенту).** Для представлення навчальних об'єктів відповідно до профілю навчання, для попереднього тестування й/або відповідно до запитів користувачів, потрібен інтерфейс відображення матеріалів. Цей компонент також забезпечує трекінг результатів, посилання на відповідні джерела інформації й різні варіанти оцінки й зворотного зв'язку від користувачів. Цей інтерфейс може бути налаштований для конкретної організації, що використовується СУНК. Для прикладу, контент може бути поданий на веб-сторінках, що містять емблему організації й елементи оформлення, прийняті в потоковому корпоративному стилі. Крім цього, елементи керування й оформлення можуть бути локалізовані під необхідний регіон.
- **Засоби адміністрування.** Цей додаток використовується для керування обліковими записами студентів, запуском курсів з каталогу, відстеження результатів, складання звітів про процес навчання й інших простих адміністративних функцій. Ця інформація

може бути передана в СУНП, яка призначена для здійснення більше просунутої адміністративної функціональності.

Тіньова сторона застосування СУНК полягає у тім, що вона дозволяє дати великий поштовх плануванню й одержанню навичок дизайну ефективних навчальних об'єктів, оскільки надає для використання шаблони й приклади. Дизайнери повинні мислити нелінійно й добрі розуміти всі різні варіанти контексту, до яких об'єкт буде необхідний або може бути використаний. Наприклад, якщо навчальний об'єкт виходить за рамки контексту або надає недостатньо засобів допомоги, те він принесе скоріше шкоду, ніж користь. Деякі курси, такі як вимоги з техніки безпеки або сертифікатні програми, повинні містити певний набір розділів у певному порядку й не можуть бути розділені на шматки.

3.5 Структура СУНП

СУНП забезпечує єдину точку доступу до різних навчальних ресурсів. Це автоматизує адміністрування навчальних програм і відкриває безпрецедентні можливості для розвитку студентів. Можна ідентифікувати слухачів, яким потрібні специфічні курси й розповісти їм, як вони зможуть це застосувати на практиці, коли це можливо, як це можливо (навчальний клас, он-лайн або CD-ROM), чи є які-небудь попередні вимоги і як вони можуть виконати ці вимоги. Після того, як студент завершує курси, СУНП може призначити тести, скласти звіт за результатами тестів і рекомендувати подальші кроки. Завдяки цим можливостям, СУНП — це інструмент для підтримки організацій, що пред'являють тверді вимоги по сертифікації, у таких вертикальних ринках, як охорона здоров'я, фінанси або державне керування [36].

Уважніше розглянемо ці можливості СУНП:

- **Підтримка змішаного навчання.** Люди вчаться різними способами. СУНП повинна надавати можливості простим чином змішувати навчання в навчальних класах і віртуальні навчальні курси. У комбінації ці можливості активізують як звичайне, так і персоналізоване навчання.
- **Інтеграція з HR.** Коли системи інтегровані, кадровики вводять інформацію в HR систему, і співробітник автоматично підписується на тренінги, спеціально призначені для його ролі в компанії.
- **Інструменти адміністрування.** СУНП повинна давати можливість адміністраторам управляти реєстрацією користувачів і профілями, визначати ролі, визначати сертифікаційні діаграми, призначати викладачів, авторів курсів, управляти контентом й адмініструвати внутрішні бюджети, платежі користувачів і збитки. Адміністраторам необхідний повний доступ до бази дані навчання,

можливість створювати стандартні й кастомізовані звіти по індивідуальних і групових показниках. Звіти повинні масштабуватися аж до можливості включення всього навчального закладу чи компанії. Система повинна давати можливість встановлювати розклад для учнів, інструкторів і навчальних класів. По можливості, всі функції повинні мати здатність до керування через автоматизований дружелюбний інтерфейс.

- **Інтеграція контенту.** Дуже важливо для СУНП забезпечувати підтримку широкого кола курсів від сторонніх виробників. Коли ви купуєте СУНП, майте на увазі, що деякі СУНП сумісні з інструментом розробки тільки власного виробництва, а інші дуже обмежено сумісні зі стандартами навчального контенту. Постачальник СУНП повинен сертифікувати контент виробництва третіх фірм, і доступ до курсів повинен бути так само простий, як використання випадаючого меню.
- **Дотримання стандартів.** СУНП повинна підтримувати стандарти, такі як SCORM й AICC. Підтримка стандартів означає, що СУНП може імпортувати й управляти контентом і курсами, які скомпільовані у відповідності зі стандартами, не залежно від засобів розробки.
- **Можливості тестування.** Модулі оцінки й тестування дозволяють розробникам створювати програми тривкі у часі. Дуже гарний варіант, використати модуль тестування й включати тест як частину кожного курсу.
- **Керування знаннями.** Модуль керування знаннями дозволяє організації визначити необхідність у навчанні й ідентифікувати область додаткових зусиль, базуючись на компетенції робочого колективу в конкретній області. Оцінка знань може бути отримана з різних джерел.

3.6 Інтеграція СУНП з СУНК

Гарна СУНП забезпечує інфраструктуру, що дозволяє компанії планувати, проводити й управляти навчальними програмами будь-яких форматів на вибір. Вона також підтримує численні засоби розробки курсів і легко інтегрується із провідними СУНК системами. У цій ролі, як каталізатор загального навчального середовища, СУНП може інтегрувати СУНК навчальні об'єкти через технічні специфікації й стандарти й відповідати за керування всього навчального контенту, включаючи програвання й трекинг, зберігання контент-репозиторію, об'єднання й роз'єднання об'єктів контенту, інкорпорація об'єктів контенту в змішані процеси, трекинг результатів навчання по окремим курсам.

Ключ до успіху інтеграції — це відкрита, інтероперабельна архітектура. У цей час провідні постачальники СУНП й СУНК здійснюють сертифікаційні програми по визначенню сумісності й інтероперабельності між своїми продуктами. Поряд з тимчасовими й грошовими витратами для постачальників, сертифікаційні програми жадають від замовників, які бажають здійснити інтеграцію, узгодження з постачальниками списку виправлень і змін, які вони повинні впровадити.

Керування на рівні об'єктів вирішує безліч ІТ-проблем, але не є панацеєю. Один аналітик приводить нижченаведену метафору для ілюстрування функцій СУНК: традиційні курси — це пакети із сумішшю бобів, навчальні об'єкти — це боби без пакетиків, а СУНК — це система, що відкриває всі пакетики, звалює всі боби в одна велику посудину, приклеює етикетку з описом на кожен біб, так що вони можуть бути перепаковані в нові пакети при необхідності.

4 Принципи реалізації

4.1 Структура середовища

Дослідження автоматизованої системи управління навчальним закладом (АСУНЗ), що забезпечує виконання стандартних функцій керування навчальним процесом та допоміжними структурними підрозділами: створення внутрішньої інформаційної мережі університету (Intranet), інформаційних порталів факультетів та основних структурних підрозділів університету з забезпеченням прозорого видаленого доступу до інформаційно-довідкових та навчальних матеріалів засобами Інтернету, порталів навчально-методичного відділу та відділу кадрів з забезпеченням "дружнього" інтерфейсу, що має можливість видаленого доступу, який забезпечить зручне ведення облікової інформації, складання розкладів занять, розрахунки навантаження викладачів, запис студентів на вибіркові курси, ведення та облік оцінок студентів за всі види навчальних робіт, документообіг, пошукові послуги тощо.

Проблема є актуальною, оскільки її розв'язання забезпечить: значну економію часу на управлінські рішення, чіткість керування, інформованість керівництва, реальне підґрунтя для прийняття рішень (з можливістю використання автоматизованої експертної системи по виробленню таких рішень).

Однією з задач досліджень була розробка архітектурного рішення і реалізація платформи, які б забезпечили можливість ефективного виконання довільної функціональності згідно з вимогами, що будуть розроблятися під час поетапної автоматизації навчального закладу.

4.1.1 Вибір технології

Вибір архітектури та технології розробки системи базувався на їх можливості забезпечити гнучку розробку системи протягом великого періоду часу багатьма незалежними розробниками. Обрана компонентно-орієнтована технологія забезпечує розвиток системи на незалежні модулі та гнучке зв'язування модулів в одну систему. Більш того, при необхідності внесення змін до будь-якого модуля у майбутньому, кожен компонент може бути замінений на модифіковану версію без перекомпіляції всього коду. Такий підхід не тільки покращує параметри підтримки коду, пришвидшує розробку системи, організує вихідний код, а й надає можливість дуже чітко протестувати системні модулі, що підвищує стабільність всієї системи.

Основним підходом при побудові компонентної структури є так звана концепція Inversion of Control (IoC). Основна ідея IoC характеризується висловом: «Не клич мене, я покличу тебе». Тобто відповідальність за виклик будь-якого компонента в системі лягає не на самі компоненти, а на каркас, в якому вони поєднані. Таким чином, компоненти не повинні «знати» про інші компоненти для своєї роботи, а також, як їх знайти та викликати. Каркас, що реалізує IoC контейнер, сам визначає, який компонент потребує якого і надає його під час виконання.

В [36] надано більш точну класифікацію IoC та описано патерн Dependency Injection (DI), що більш точно характеризує принцип, що пов'язує компоненти в каркасі на основі інтерфейсів, відділяючи реалізацію компонентів від їх конфігурації, таким чином реалізуючи незалежність компонентів і можливість їх легкої заміни всередині системи.

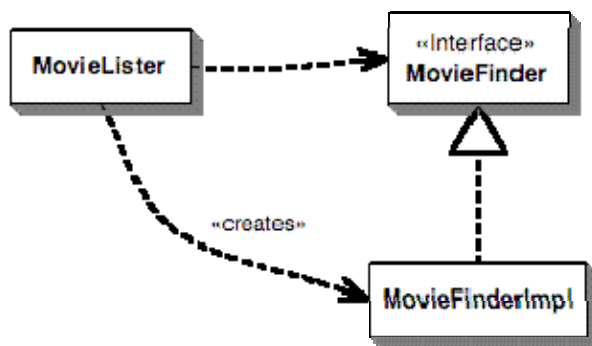


Рисунок 4.1. Залежності за звичайної побудови класів.

Прикладом задачі [1], що вирішує DI, може бути ситуація, показана на Рисунку 4.1. Існує клас MovieLister, який для своєї роботи потребує компонент, що реалізує інтерфейс MovieFinder, тобто він залежить від конкретної реалізації MovieFinderImpl. Задача патерну DI полягає в тому, щоб прибрати цю залежність, тобто надати можливість вибирати ту реалізацію інтерфейсу MovieFinder, яка необхідна різним компонентам.

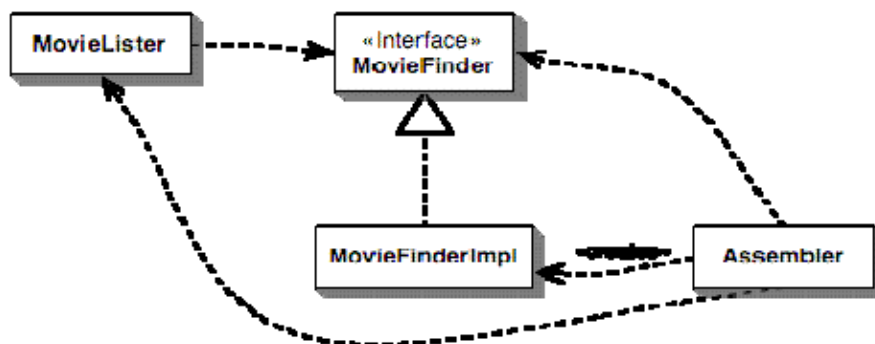


Рисунок. 4.2 Залежності за використання DI.

Основна ідея DI полягає у використанні окремого об'єкта Assembler, який надасть класу MovieLister значення конкретної реалізації MovieFinder інтерфейсу, прибираючи

залежність MovieLister від конкретної реалізації та залишивши прив'язку до інтерфейсу MovieFinder (що важливо для збереження типів).

Spring Framework

В залежності від типу надання об'єктом Assembler інформації про конкретну реалізацію, розрізняють три форми DI: Constructor Injection (вставка через конструктор), Setter Injection (вставка через поля класу) та Interface Injection (вставка через інтерфейс). Всі три форми описані в [36]. При розробці АСУНЗ був використаний каркас Spring Framework [37], який оснований на підході IoC і використовує вставку через спеціальні методи класу – сеттери, через які клас отримує конкретні дані ззовні. Spring реалізує IoC контейнер, який використовує XML файли для зберігання конфігурації кожного компонента (або інші методи зберігання) і за допомогою якої виставляє (injects) необхідні параметри компонентам.

Використаний підхід дозволяє будувати компоненти з чітко визначеним інтерфейсом, які не залежать один від одного, а їх поєднання в одне ціле відбувається за допомогою окремої конфігурації. Слід зазначити, що компоненти не залежать від Spring каркасу, а сам каркас виконує службові функції, наприклад читає конфігураційні файли та виставляє необхідні значення.

Таким чином, використання Spring каркасу має багато переваг: надає «легкий» контейнер J2EE застосуванням, використовує зв'язування компонентів за інтерфейсами, не зобов'язує застосування використовувати Spring і залежати від нього, підвищує можливість якісного тестування кінцевого продукту за допомогою unit тестування. Слід зазначити, що Spring не реалізує і не намагається реалізувати конкретні підкаркаси, наприклад роботи з базами даних, забезпечення транзакцій, ведення журналу – він просто надає методи для їх зв'язування, отже не лімітує розробників у використанні найбільш зручних і корисних для проекту технологій.

4.1.2 Архітектура

Цикли документообігу

Основною метафорою при розробці архітектури було розбиття роботи навчального закладу на цикли документообігу. Розроблена архітектура відображає реальні цикли документообігу у навчальному закладі і продовжує їх на проміжках, які можна автоматизувати. Основою для обробки є віртуальні об'єкти моделі даних, які наближено відображають реальні об'єкти. Наближення вибрано таким чином, щоб об'єкти були якомога простішими для системи, забезпечивши збереження їх інформаційних характеристики, які мають бути враховані при автоматизації.

Для можливості роботи з базовими об'єктами системи узгоджено і цілісно, в архітектурі використовується метафора документу. Документ є підсистемою реально існуючої структури даних, в межах якої можуть проводитися зміни. Кожен документ виділяється таким чином, щоб зміни в структурі об'єктів, які входять в його склад, були атомарними, тобто не впливали на всю структуру в цілому. Документ також є об'єктом.

Кожен документ передається між прошарками системи і для кожного прошарку достатньо працювати з документом, щоб забезпечити конкретні кроки будь-якого автоматизованого процесу. Кожному прошарку відома структура даних системи і кожен прошарок може модифікувати документ в межах цієї структури.

Будь-який запит користувача виконується як проведення певної операції над документом. Можливі дії над документом визначаються циклами і підциклами документообігу (workflow cycles). Кожен цикл документообігу виконується в транзакції і трасується, поки не завершиться. Кожен серверний виклик передбачає те, що виклик проходить в якомусь конкретному циклі, який знаходиться в конкретному стані.

Зміни до інформації в системі здійснюються в основному через відповідні документи. Якщо необхідно зробити зміни до системи, наприклад змінити прізвище студента, потрібно відкрити документ, який відповідає реєстраційній картці студента і внести необхідні зміни. Модель захисту чітко відслідковує, які операції користувач може здійснювати над документом і що потрібно для того, щоб документ міг внести зміни в діючу систему. Наприклад, якщо для зміни стану будь-якого бізнес-об'єкту потрібні додаткові санкції, то документ відповідним чином обробляється перед тим, як він змінить стан системи. Система записує зміни, які потім зберігаються у сховищі даних.

Система складається з таких основних прошарків:

1. Інтерфейс користувача
 - a. Показує певну частину стану системи користувачу
 - b. Обробляє дії користувача і фіксує їх в змінених документах
 - c. Отримує документи від моделі і передає змінені документи в модель для обробки
2. Модель
 - a. Формує базову структуру даних
 - b. Обробляє модифіковані документи згідно з бізнес-правилами
 - c. Отримує дані від прошарку даних і зберігає дані через нього
3. Прошарок обробки даних
 - a. Забезпечує правильне і узгоджене зберігання даних та повернення даних на запити

Інтерфейс користувача

На рівні інтерфейсу користувача можна змінювати документ (але не можна створювати новий документ), проглядати документ, додавати нові об'єкти в документ в межах структури даних. Інтерфейс може як завгодно довго утримувати документ в сесії і крок за кроком обробляти його, проте такі зміни розцінюються як заявка на модифікацію і не є реальною зміною для системи. Система не може реагувати на ці модифікації як на щось важливе (хоч для користувача можуть видаватися реальні помилки і попередження про неправильність вводу і т.ін.).

Щоб не обробляти щоразу дані за схожою схемою в конкретних класах на рівні інтерфейсу, однакові принципи обробки виносяться в методи роботи з контентом документа. Якщо якась комплексна зміна чи відтворення стосується скоріше вмісту аніж презентації даних, вона обов'язково обробляється методами контенту документа.

Якщо на певному етапі потрібно активізувати зміни, документ передається на рівень моделі і бізнес-логіки.

Модель (сервіс користувача)

На цьому рівні документ обробляється як вже повністю підготовлений. Модель обробляє документ цілком. Тут може перевірятися узгодженість документа з іншими даними, дозвіл на модифікацію та вірність внесених або модифікованих даних. На цьому рівні може бути видана помилка і обробка документу може бути перервана. Якщо з певними документами пов'язані певні правила їх обробки, вони застосовуються на цьому рівні.

Прошарок обробки даних

Документ обробляється одним з трьох можливих способів: створюється, зберігається, відтворюється. Кожна операція реалізується окремо для кожного типу контенту документа. Для обробки кожного документу часто потрібно знати не лише те, що знаходиться в документі (часто документи можуть охоплювати досить великі структури даних, навіть рекурсивно пов'язані в потенційно нескінченні ланцюги структур), а те, що саме з усього контенту документу змінено (або потрібно створити). Для цього існують фільтри, які передаються операціям поруч з документами.

Фільтр є структурою функціональних об'єктів, що накладається на контент (можливий контент) документу для формування обмеженого контенту. При застосуванні в операціях відтворення документа фільтр вказує яку частину контенту з усього доступного для документа контенту очікують інші прошарки для обробки. Проте фільтр не треба плутати з природнім обмеженням контенту документа – воно враховується незалежно від фільтрів. При застосуванні в операціях збереження, фільтр вказує в якій частині контенту

документа відбулися зміни, щоб не потрібно було обробляти весь контент задля збереження операційних ресурсів.

Фільтри

Фільтр утворюється простими операціями з атомарних фільтрів. Кожен атомарний фільтр формується на рівні прошарку обробки даних і передається на запит через рівень моделі на рівень інтерфейсу користувача. Об'єкти інтерфейсу можуть дістати будь-яку кількість атомарних фільтрів в будь-який момент і сформувати з них будь-яку структуру, використовуючи доступні базові операції. Цю структуру вони передають на обробку як додаток до документу.

Далі фільтр передається на рівень роботи з даними, якому відомо як його застосувати для формування або збереження об'єкту.

4.1.3 Нестандартні архітектурні рішення

Неконтрольовані модифікації

В системі можна часто спостерігати таке явище, коли кілька ролей можуть змінювати деякі дані, відповідно до доступної на даний момент інформації про дані, але загальний контроль за даними насправді має обмежена кількість ролей або одна роль. З одного боку не можна заборонити іншим ролям змінювати критичні дані в системі (бо вони можуть застаріти), а з іншого - змінені критичні дані можуть стати помилковими, оскільки роль, яка має над ними контроль не помічає зміни.

В такій ситуації використовується механізм фіксування даних в сталих документах. Окрім динамічної структури даних в базі, контролююча роль може зберегти певну частину даних в окремому документі (зафіксувати), також поряд зі стандартним збереженням.

Пізніше якщо хтось вносить зміни до даних, контролююча роль може звірити зафіксовані дані з реальними і виправити або реальні дані (скоректувати помилки) або зафіксовані дані (затвердити зміни).

Отже, будь-які модифікації можуть бути перекриті послідовним контролем над ними.

Передача параметрів через фільтр

Часто виникає необхідність на рівні обробки даних знайти деякі значення чітко визначених властивостей об'єктів, що використовуються для фільтрації. Наприклад, вибираються всі викладачі з одного факультету, а значення цього факультету треба використати ще в інших операціях. Шукати ці значення в графі фільтру не зовсім зручно і не зовсім правильно. А передавати такі параметри додатково як особливу FilterFeature також не так стабільно, бо додає зайву функціональність у фільтр і ускладнює як обробку так і реалізацію контенту документа. Тому для цього використовується фільтрування за прикладом (example). Сам фільтр розуміє, що треба відібрати елементи за рівністю

встановлених непустих значень, а інші структури рівня даних можуть знаходити сталі значення властивостей із прикладу і використовувати їх. Такий підхід вважатиметься найбільш вдалим.

Ієрархічні форми та заповнення даних

Для заповнення даних в складні ієрархічні структури використовуються ієрархічні форми з аналогічною структурою компонентів (приблизно кожен компонент відповідає певному типу даних в структурі). Основною була вибрана стратегія централізованого доступу/модифікації даних в структурі. Всі компоненти інтерфейсу забезпечують лише інформацію про шлях до певного параметру в структурі і дозволяють діставати або змінювати параметр за цим шляхом. Базовий компонент повинен обробляти задані шляхи і виконувати операції зі структурою об'єктів. Це дозволяє гнучко працювати зі структурою даних в будь-який спосіб.

Видалення і зміни сутностей

Видаленню підлягають тільки сутності, не пов'язані з іншими сутностями. Якщо зв'язок певної сутності з іншими сутностями має місце, тоді користувач не повинен мати можливості видалити сутність, поки не будуть видалені пов'язані сутності або отримує повідомлення про помилку при спробі видалення.

Змінювати дозволяється все, що не заборонено правилами документу, який забезпечує зміни.

Протоколювання (ведення журналу змін)

В системі ведеться запис виконаних операцій і запис змін даних. Всі зміни зведені до конкретного значення поля таблиці даних і поділені на три категорії: додавання поля, модифікація поля і видалення поля. Будь-яка зміна стану бази даних може розцінюватися як очікувана, неочікувана, неконтрольована. Якщо запис для зміни протоколюється, зміна вважається очікуваною і стан бази даних після зміни і до зміни може бути пізніше відтворений за журналом. Якщо запис певного роду змін принципово не ведеться, такі зміни вважаються неконтрольованими. Всі інші зміни є неочікуваними змінами і можуть бути наслідком збоїв у системі, злому системи або помилкою імплементації системи.

Протоколювання змін ведеться для всіх простих типів полів. Для бінарних даних протоколювання ведеться тільки якщо їх розмір не перевищує певного визначеного розміру. Для текстових полів ведеться запис лише перших 1000 символів, вважається що вони є найінформативнішими із всього контенту поля.

4.1.4 Реалізація

Основним каркасом в реалізації описаної архітектури є Spring Framework. Для реалізації конкретних прошарків або функціональних частин були застосовані додаткові каркаси. Загальна модель приводиться на рисунку 4.3.

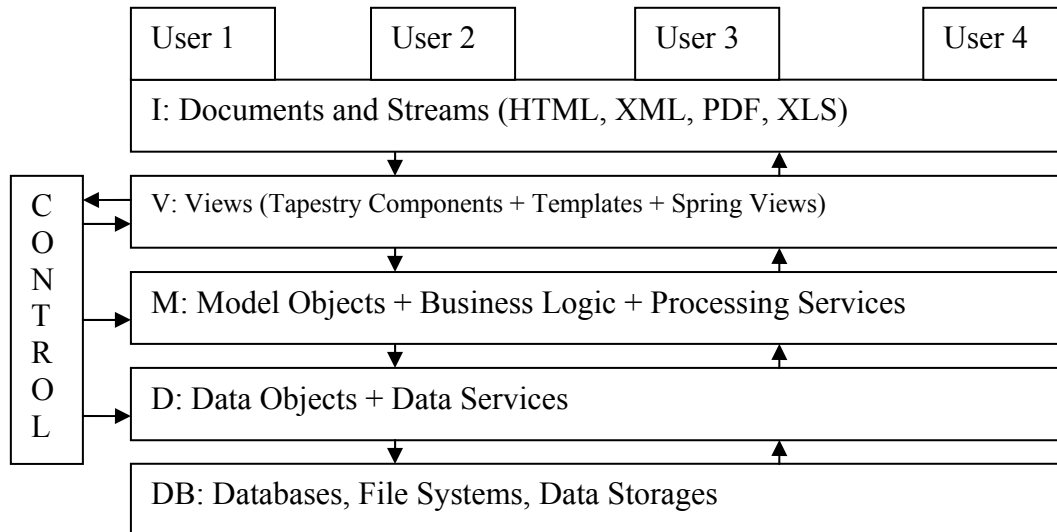


Рисунок. 4.3. Загальна модель системи.

Прошарок роботи з даними (Hibernate)

Одним з основних прошарків є прошарок роботи з даними. При виборі технології реалізації прошарку, велику увагу було приділено незалежності від системи, де дані зберігаються, можливості чітко співставляти бізнес-об'єкти системи з даними і швидкість роботи. На момент вибору технології, найкращою системою підтримки зберігання об'єктів і їх зв'язків, а також пошуку таких об'єктів став Hibernate [38]. Hibernate дозволяє моделювати наслідування, асоціацію і композицію об'єктів на реляційних базах даних. Також надає можливості ефективного пошуку об'єктів за допомогою, як стандартного SQL, так і специфічного HQL (Hibernate Query Language). Важливою перевагою останнього є те, що запити на пошук об'єктів задаються в SQL-подібному вигляді і виконуються досить швидко.

Можливості Hibernate надають величезних переваг для загальної системи, особливо при зберіганні таких об'єктів як конкретні види документів, що можуть бути дуже великими і відображатися на величезну кількість таблиць реляційної бази даних. В рамках Hibernate можна просто викликати об'єкт, змінити необхідні параметри, як будь-якого класу Java, та записати модифіковані дані. Hibernate сам вирішує, які реляції треба змінити, причому сама зміна робиться в контексті транзакції.

Як вже було відзначено, Spring Framework не реалізує конкретні засоби роботи з даними, але дозволяє використовувати будь-яку реалізацію, якою в нашому конкретному випадку є Hibernate. Таким чином, об'єкти, що зберігаються (persistent), поєднуються за

допомогою патерну DI, а це дає змогу наприклад конфігурувати такі об'єкти через зовнішні конфігураційні файли, зберігати SQL/HQL запити в зовнішніх конфігурація і т.п. Таким чином об'єкти прошарку даних є повністю незалежними від інших об'єктів, проте легко з ними пов'язуються.

Інтерфейс користувача

При розробці архітектури, використовувався патерн MVC (Model-View-Control), що дозволило досягнути незалежності від представлення будь-яких даних. Так, одні й ті самі дані можуть бути відображені HTML сторінкою, PDF документом, або SVG діаграмою.

Основною концепцією при побудові інтерфейсу користувача було створення тонкого клієнту — веб застосування, що надає такі переваги до системи АСУНЗ: доступ з будь-якого місця Інтернет, відсутність необхідності інсталиувати систему на робочих станціях, одночасна доступність змін одразу всім користувачам, невибагливість до програмного забезпечення клієнтів та потужності клієнтських комп'ютерів, безкоштовність платформи для клієнтів, прості у користуванні інтерфейси. Незважаючи на це, архітектура підтримує і інші види інтерфейсів (наприклад повноцінний графічний інтерфейс або інтерфейс для мобільних пристроїв) і можуть бути використані поруч зі стандартним веб інтерфейсом у разі необхідності.

При розробці тонкого клієнту був вибраний каркас Tapestry [39], основною відмінністю якого у порівнянні зі схожими розробками (JSP, Velocity) є чітка компонентна структура і використання параметрів JavaBeans для задання конфігурації веб-компонентів. Це дає низку переваг: чітке розділення представлення від коду, всі компоненти в Tapestry можуть повторно використовуватися в будь-якому місці веб застосування; підвищення можливості якісного тестування компонент представлення, що в умовах інших каркасів майже недосяжно; додаткові переваги у захисті, а також можливість роздільної роботи команди програмістів та технічних дизайнерів.

Використання JavaBeans і компонентного підходу, як основного в Tapestry, дуже добре інтегрується в каркас IoC Spring Framework, а отже ми отримуємо додаткові переваги від їх поєднання. Як і з іншими компонентами, веб-компоненти Tapestry є незалежними і конфігуруються окремими конфігураціями, які потім виставляються контейнером Spring.

Каркас зв'язування

Використавши Spring в якості основного каркасу системи, що реалізує принцип IoC, ми змогли природно зв'язати додаткові каркаси роботи з даними, а також каркаси представлення. Як видно з Рисунку 4.3. різні прошарки системи передають інформацію в обох напрямках через відкриті інтерфейси, виділяється чіткий прошарок бізнес-логіки і презентаційний шар ніколи напряму не працює з прошарком даних. Слід зазначити, що всі

прошарки даних працюють уніфіковано, так як взаємодія проходить через визначений інтерфейс документу (Document), проте кожен прошарок обробляє тип контенту документа згідно власних зобов'язань та прав.

Така уніфікована робота практично не застосовується в інших подібних системах АСУНЗ. Зазвичай шари систем працюють неуніфіковано, шар інтерфейсу викликає різні методи сервісів, а сервіси викликають потрібні їм методи шару роботи з даними. А такий архітектурний підхід знижує контроль над виконанням операцій, призводить до розсіювання подібних функцій по багатьох класах і збільшує можливість помилок при обробці даних в різних контекстах.

Новизна розробленої архітектури полягає у тому, що поточний контекст задається типом контенту документа. Такий підхід забезпечує виконання важливої вимоги — робота з даними завжди відбувається у відповідності до контексту, в якому вони збираються, обробляються і зберігаються.

4.1.5 Нефункціональні та інші вимоги

При побудові архітектури, а також реалізації системи, велику увагу було приділено нефункціональним вимогам, архітектурні рішення для підтримки яких є ще однією перевагою АСУНЗ від схожих розробок.

Захист

В системі реалізована модель захисту, що базується на ролі користувача. Кожна роль має набір службових обов'язків (duties), які вона може виконувати. Кожен фізичний користувач в системі може мати грати різні ролі у системі, тобто може мати різні рівні доступу до різних частин системи. Наприклад, якщо користувач є і викладачем і керівником магістерської програми одночасно, він зможе використовувати функціональність системи, що призначена для обох ролей.

Модель захисту функціонально розширює модель захисту платформи Java. Метод аутентифікації, що використовується по замовчанню – логін користувача і пароль, система легко підтримує і інші методи, наприклад використання сертифікатів (Kerberos, X.509). Сесія підтримується за рахунок захищених cookie.

На рівні доступу до бази даних, система захищена від прямого доступу до даних (доступ здійснюється тільки через проміжний шар). Кожному *об'єкту* дії користувача (вищезгадане duty) встановлюється необхідний доступ до бази даних на час виконання дії. Це означає, що відповідний користувач встановлюється не тільки для певної категорії дій бази даних з окремими правами, а навіть для заданої ситуації (яка дія, який користувач, в яких умовах він виконує дію, що він виконував до того чи який стан системи на момент виконання дії). Використовується деперсоналізація користувача при доступі до даних.

Доступ відбувається використовуючи попередньо визначений набір користувачів в самій базі даних.

Застосовано і не прямий захист. Використання журналу внесення змін у систему і протоколювання всіх без винятку операцій в системі, дозволяють чітко відслідковувати намагання втрутитись у систему, повертати видалені, або спотворені дані, лімітує навмисну зміну даних та несанкціонований доступ.

Розширюваність

Побудована на компонентному підході із застосуванням принципу IoC система без принципових обмежень може бути розширена до рівня автоматизації будь-якого навчального закладу, або будь-якої установи, де документообіг відіграє важливу роль, без обмеження функціональності поточних доробок. Найбільш вразливими для подібних змін є тільки інтерфейси користувача (вони можуть суттєво відрізнятися для кожного нового застосування системи) та специфічні для окремого випадку використання бізнес-об'єкти.

Структура бази даних та структура класів повністю відповідає реальним об'єктам світу і не вносить суттєвих обмежень на майбутню розширюваність системи. Основні ймовірні ускладнення системи можуть скоріше привести до додавання нових структур і зміни ролі існуючих структур, аніж до їх повної заміни і повторної розробки.

Підхід до обробки інформації в системі (через цикли документообігу) є стандартним для будь-якої системи документообігу, а тому система пристосована до розширення для підтримки більшості адміністративних функцій.

Масштабованість

Застосування гнучкої архітектури дозволяє розподіляти частини системи, будуючи розподілене застосування, що особливо важливо при великій кількості користувачів і інформації, що зберігається в системі. В такому випадку вдається вирішити проблеми піків навантаження, ввести пріоритети в обробці інформації та забезпечити безперервне функціонування системи. Опис вже існуючих інтерфейсів в небінарному форматі, може відкрити доступ до їх використання будь-якій іншій системі, використовуючи наприклад технологію веб сервісів.

Поточну реалізацію системи було протестовано в поточному варіанті на роботи з кількістю інформації, що в 10 раз перевищує заплановану завантаженість. При поточному режимі користування системою, вона може працювати на одному сервері з задовільним часом відгуку.

Можливість тестування системи

Використання Spring Framework, чітке виділення інтерфейсів та підхід до уніфікованого інтерфейсу Document дозволяє покрити практично всю критичну

функціональність unit-тестами, за допомогою JUnit, або подібних систем. Крім того, використовуючи Tapestry каркас, ми досягли високого рівня можливості тестувати компоненти презентаційного рівня, що в комплексі дає можливість повністю проводити підхід паралельного тестування (test-driven approach) [40] при розробці системи.

Такий підхід не тільки забезпечує стабільність системи в цілому, а й полегшує її розвиток і підтримку у майбутньому.

4.1.6 Функціональність системи

Система складається з ряду автоматизованих робочих місць: секретаря приймальної комісії, методиста з навчально-методичної роботи, методиста управління навчальним процесом, декана, керівника магістерських програм, студента, викладача. Останні два АРМи надають тільки лімітовані можливості, проте плануються до розширення у майбутньому.

Система повністю автоматизує процес роботи приймальної комісії, включаючи реєстрацію, спрощену реєстрацію випускників НаУКМА (вступ до магістерських програм), внесення результатів вступних випробувань, зарахування у склад студентів, генерація звітів, а також всіх документів, що використовуються в процесі, включаючи залікові відомості, накази про зарахування, і інші.

Використовуючи АРМ, методисти мають можливість працювати зі студентами та викладачами (включаючи розширені види пошуку), створювати групи, працювати з навчальними планами, вносити оцінки протягом сесій, генерувати всі необхідні звіти та документи, що використовуються у навчальному процесі.

Дуже потужною функцією є автоматизація процесу складання розкладу. Як відомо, задача складання розкладу є NP-повною великої розмірності. Втім, застосування генетичних алгоритмів надає можливість в більшості випадків розв'язати цю задачу за прийнятний час. Підхід, застосований при розробці, описаний у [41].

4.1.7 Нефункціональні характеристики

Окрім запрограмованих функцій АСУ в системі також підтримуються декілька цілісних нефункціональних вимог, серед яких найбільш очевидні наступні:

- 1) Особлива структура захисту доступу до користування системою, її даних та операцій
- 2) Розширюваність та гнучка конфігурація
- 3) Надійність та стійкість при неправильному використанні функцій
- 4) Використання найновішої платформи і сучасних стандартів
- 5) Продуктивність обробки даних

6) Дружній інтерфейс

КОНЦЕПЦІЯ БЕЗПЕКИ

Кожен користувач має свій окремий доступ до системи з використанням паролю, про який не можуть знати інші користувачі. Будь-який процес взаємодії з системою починається з авторизації користувача і називається *сесією* обміну даними. Сесія забезпечується шифрованим обміном інформацією між браузером і сервером, тому ніхто зовні не може перехопити доступ до відкритої сесії. Таким чином гарантується захист від несанкціонованого доступу та неавторизованого виконання функцій системи, при умові що користувачі не повідомляють реквізити доступу до системи іншим особам.

Кожен користувач виконує *тільки визначені* операції з системою в межах своєї ролі. Кожній операції відповідає певна сукупність даних, які в даній ролі можна переглядати, додавати, видаляти чи змінювати. Також за архітектурою безпеки один користувач не може виконувати операції іншого користувача (навіть якщо вони не відрізняються за суттю).

Всі операції з даними реєструються. Для них вказується користувач, який виконав операцію. Всі зміни в даних, що відбуваються на кожній операції також окремо записуються, щоб в будь-який момент можна було з'ясувати що саме було змінено певним користувачем і повернутися при крайній необхідності до попереднього стану даних.

ГНУЧКІСТЬ ТА МОЖЛИВІСТЬ РОЗШИРЕННЯ

Система розроблена з врахуванням того, щоб додаткова функціональність можна було легко додати до вже існуючої функціональності. Всі елементи системи розроблялися з дотриманням можливостей для змін і розширень. Автоматичні тести функціональності дозволяють слідкувати за станом та якістю вже впроваджених можливостей при введенні нових.

При необхідності в систему можна додати будь-які інші аспекти робочих процесів університету зі збереженням існуючої функціональності.

Також у випадку зміни автоматизованих процесів чи їх характеристик в університеті, відносно нескладно адаптувати систему та результати її роботи до цих змін.

НАДІЙНІСТЬ

Система працює в режимі підвищеної надійності роботи з даними і зовнішніми системами. Помилки, які можуть виникати через неправильні дії користувача, випадкові впливи на процес роботи з системою чи конфлікти взаємних змін не порушують цілісність даних, що накопичуються в системі. При виникненні будь-яких непорозумінь використовується техніка «швидкого переляку» — якомога раніше показати помилку без

реальних змін в даних, аніж дійти до незворотніх порушень накопиченої інформації або процесів.

В системі виключена будь-яка можливість самовільного або незрозумілого модифікування чи знищення даних. Кожна роль користувача системи має доступ до читання чи модифікації лише тих даних, за які вона відповідальна і всі зміни можливі лише шляхом усвідомлених операцій через інтерфейс системи. При цьому ведеться постійний запис всіх змін внесених всіма користувачами, так що у випадку змін невизначеного походження можна знайти користувача і час, коли він змінив ту чи іншу інформацію в системі.

СУЧАСНІ СТАНДАРТИ

В системі використовуються найновіші і найефективніші досягнення і рішення в області проектування та розробки систем автоматизації університетів. Робота з системою відбувається за моделлю клієнт-сервер, причому клієнтом виступає стандартний веб-броузер. Функціональні (серверні) блоки системи побудовані на платформі Java.

Зовнішні системи, які використовують XML-комунікації (наприклад інші B2B системи) можуть користуватися функціями системи через прямий доступ без використання пристосованих для людини-оператора інтерфейсів.

ПРОДУКТИВНІСТЬ

Незважаючи на велику кількість масивів даних, які обробляються системою, система максимально оптимізована для нормативної продуктивності. Деякі функціональні елементи, що часто використовуються, підлягають постійному моніторингу і постійній оптимізації продуктивності.

Загалом, в основу архітектури системи закладені резерви для можливої оптимізації часу роботи в майбутньому на всіх основних критичних шляхах виконання функцій.

ДРУЖНІЙ ІНТЕРФЕЙС КОРИСТУВАЧА

Взаємодія користувачів з системою змодельована таким чином, щоб користувачі могли якомога швидше звикнути до виконання різноманітних операцій через систему, щоб максимально унеможливити ймовірність помилкових дій через непорозуміння в роботі з системою, і щоб якомога більше операцій були простими та інтуїтивно зрозумілими для користувачів.

Також ті операції, які використовуються частіше і для яких важлива швидкість їх проведення, спроектовані таким чином, щоб ними можна було легко і швидко скористатися. Але разом з тим при цьому збережена відносна простота інтерфейсів.

ОБРОБКА ПОМИЛОК

Всі осмислені помилки при внесенні даних чи виконанні операцій виводяться користувачам в відповідному контексті роботи якомога раніше і максимально

обгрунтовано. Проте в роботі системи (особливо на стадії її опробування) можуть траплятися неспецифічні помилки, які є або результатом недостатньо правильного використання можливостей системи або просто програмними помилками в реалізації системи – такі помилки користувачам пояснюються в обмеженому обсязі, тому що користувачі не в змозі їх виправити і їх повинні опрацьовувати оператори підтримки системи.

4.2 Технічний опис впровадження

Після дослідження, а потім і опису концептуальних основ Автоматизованої Системи Управління нашого університету, що заклали фундамент MAMS, далі описано технічні процедури, що були здійснено в рамках роботи.

Дослідження цих основ було здійснено на базі вже існуючої АСУ як прототипу СУНП, що була запущена та працювала в НаУКМА кілька років раніше. Проект розроблено за допомогою студії Eclipse, кросплатформенному інтегрованому середовищі обробки на *Java*. Проект скомпільовано за допомогою *apache-ant-1.6.5*.

З метою використання потенціалу MAMS для потреб навчального процесу знову, в даній роботі розгорнуто АСУ на серверній платформі.

Для цього здійснено було такі підготовчі заходи:

- Виділено спеціальний сервер
- Встановлено серверну операційну систему *Windows 2003 Server*

Після того проінстальовано такі продукти для платформи MAMS:

- *MS SQL 2000 SP3*
- *Java Standard Development Kit j2sdk-1_4*
- *Apache Tomcat Server jakarta-tomcat ver. 5.0.25*

Наступний етапом була власне компіляція проекту.

На рисунку 4.4 зображено скріншот баз даних MAMS, з якими працює застосування. (До речі, і БД, і застосування фізично розміщені на одному сервері.)

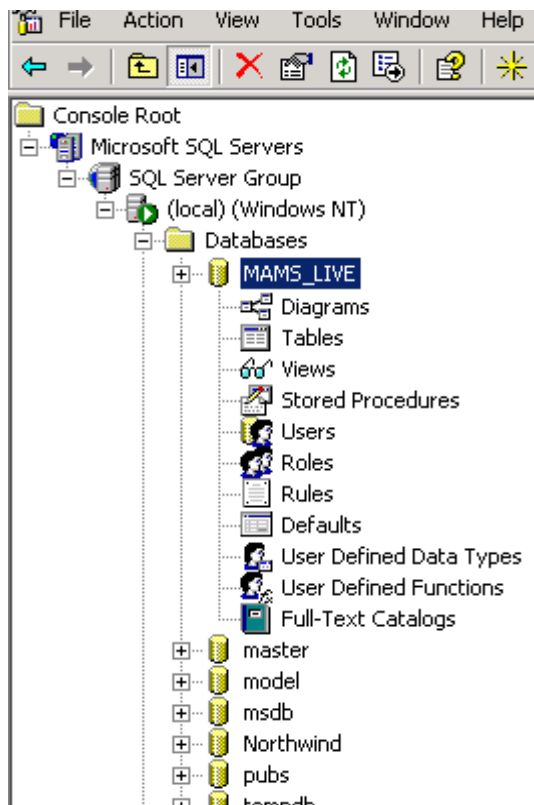


Рисунок 4.4. Жива БД MAMS

З'єднання з MAMS відбувається по web протоколу http. Він доступний за адресою:
<http://enigma.ukma.kiev.ua:8080/eias/home.xml>.

Функціональність системи надається правами доступу, які визначаються логіном і паролем певного користувача:

Входи (на даний момент):

- секретар приймальної комісії
- методист кафедри
- методист факультету
- методист НМВ
- методист студентського відділу кадрів

Список користувачів можна гнучко розширити для майбутніх практичних потреб використання.

ІНТЕГРАЦІЯ. СОЦІАЛЬНА МЕРЕЖА ПАТОС

Як і планувалось, здійснено перші кроки інтеграції. Реалізована спільна аутентифікація користувачів з платформою соціальної моголянської мережі Патос, яка написана новітньою мовою програмування *Ruby*. Це динамічна, повнофункціональна об'єктно-орієнтована мова програмування, синтаксис якої – синтез *Perl* разом із

властивостями характерними для *Smalltalk*. *Ruby* виникла в Японії в середині 90-х років: була розроблена і спроектована Юкіхіро Матсумото. Вона підтримує мультипарадигмальність, має багато спільного з мовами *Lisp*, *Perl*, *Python* [14].

Для інтеграції необхідно було внести зміни у вихідний код і дописати деякі модулі.

Ось так виглядає раніше описана АСУ, наприклад, у режимі доступу методиста факультету (можна працювати зі списком викладачів, зі списком груп студентів, переглядати навчальні плани, навчальні дисципліни, змінювати пароль доступу) у нижньому фреймі. Верхній фрейм належить власне MAMS. Верхній – це Патос, з яким MAMS інтегровано.

Для цього тут задаємо 2 фрейми:

```
<html>
<frameset rows="30,75">

  <frame name="pathos_src" src="/integration/ukma_src"/>
  <frame id="mams_iframe" name="mams"
src="http://enigma.ukma.kiev.ua:8080/eias/" width="900" height="900"
style="<%= !current_user.ukma_login.nil? ? "" : "display: none;" %>">
</frame>
</frameset>
</html>
```

Потім дописуємо відповідний клас *AddUkmaIntegration*:

```
class AddUkmaIntegration < ActiveRecord::Migration
  def self.up
    add_column :users, :ukma_login, :string, :default => nil
    add_column :users, :ukma_password, :string, :default => nil
  end

  def self.down
    remove_column :users, :ukma_login
    remove_column :users, :ukma_password
  end
end
```

Ось так виглядає нова сторінка з двома фреймами різних платформ(рисунки 4.5, 4.6)

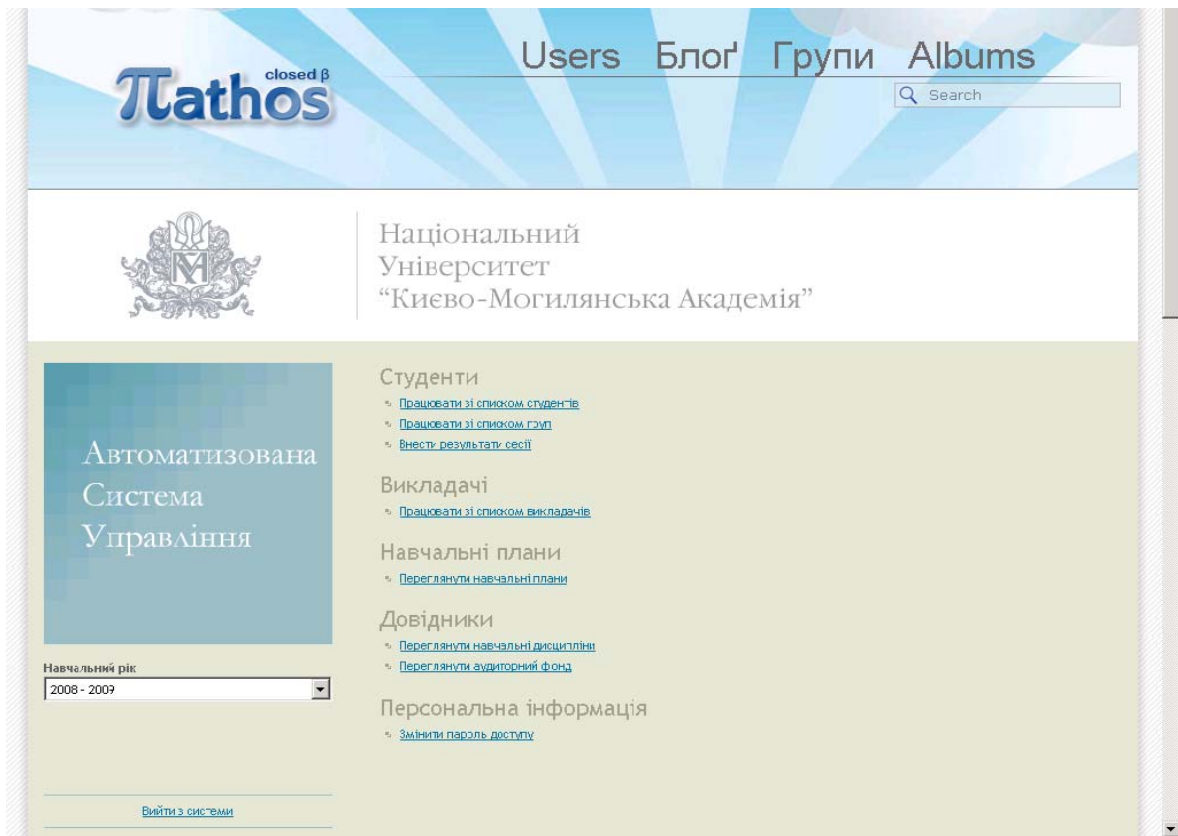


Рисунок 4.5. MAMS інтегровано в Патос

З Патоса користувач потрапляє у відповідний його аккаунт на MAMS. Інтерфейс реалізовано у вигляді двох фреймів, де у верхній частині розміщено функціонал Патос, а у нижньому – сам MAMS.



Рисунок 4.6. Функціонал Патос

Контроллер інтеграції:

```
class IntegrationController < ApplicationController
```

```

def ukma
  @user = User.find(current_user.id)
end

def save_ukma_credentials
  current_user.ukma_login = params[:user][:ukma_login]
  current_user.ukma_password = params[:user][:ukma_password]
  #user = User.find(current_user.id)
  #user = current_user
  current_user.save
  redirect_to :controller => 'integration', :action => 'ukma'
end

def ukma_src
  @user = User.find(current_user.id)
end
end

```

Тут відбувається автоматичний перехід на MAMS з thePathos. Логін з паролем передаються на АСУ, і у разі відмови видається повідомлення про помилку.

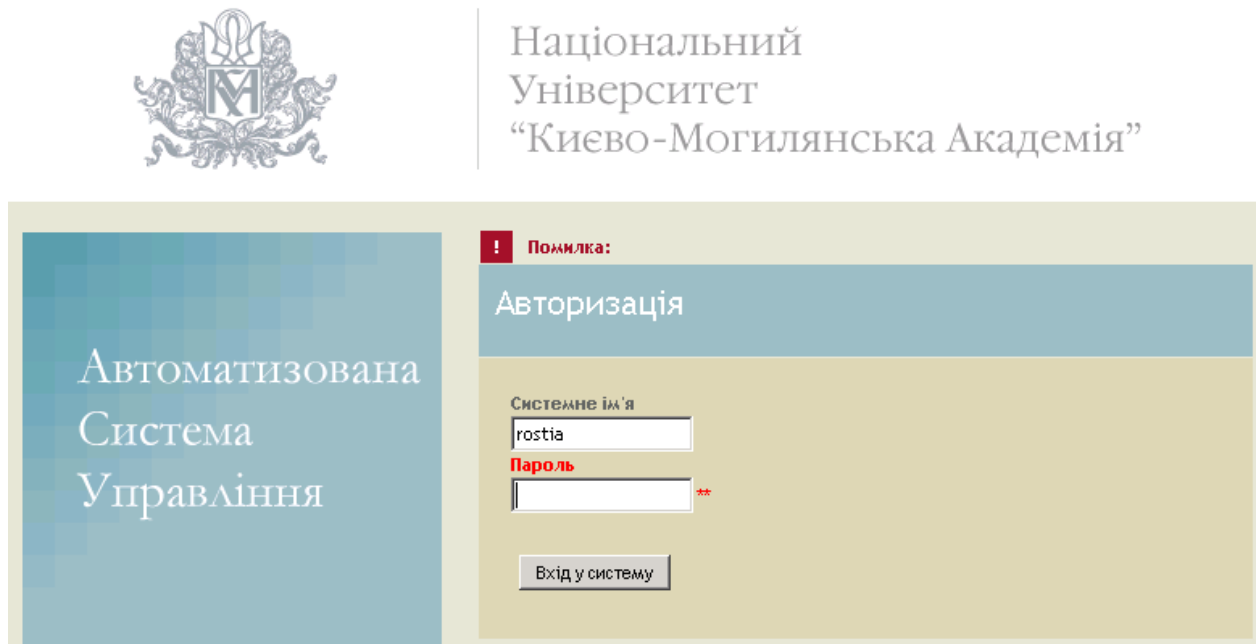


Рис. 10. Помилка аутентифікації

Фрагмент коду для технічної ілюстрації вищенаведеного малюнка:

```

<%= render :partial => "/layouts/doctype" %>
<title>Pathos: Integration</title>
<%= stylesheet_link_tag "inputs" %>
<%= render :partial => "/layouts/header" %>
<div class="notice"><span><%= flash[:notice] %></span></div>

<div style="<%= !current_user.ukma_login.nil? ? "" : "" %>"
class="wideColumn">

<% form_tag({:action => 'save_ukma_credentials'}, {:target => 'mams'}) do %>
<div class="paperStyleInputHolder">
<div class="paperStyleInputItem">
<%= text_field "user", "ukma_login" %><label for="login">Ukma login</label>

```

```

</div></div>
<div class="paperStyleInputInfoHolder">
Your UKMA system login</div>
<div class="clearLeft"></div>

<div class="paperStyleInputHolder">
<div class="paperStyleInputItem">
<%= password_field "user", "ukma_password" %><label for="password">Ukma
password</label>
</div></div>
<div class="paperStyleInputInfoHolder">
Your UKMA system password</div>
<div class="clearLeft"></div>

<div class="paperStyleInputHolder"><div class="paperStyleInputItem"><%=
submit_tag 'Submit data' %></div>
<div class="clearLeft"></div>

<% end -%>

</div>

</div>
</div>
<br/>
<br/>

<form name="submitter" action="http://enigma.ukma.kiev.ua:8080/eias/home.xml"
target="mams">
  <input type="hidden" name="service" value="direct/0/Home/$Login.form"/>
  <input type="hidden" name="sp" value="S0"/>
  <input type="hidden" name="Form0"
value="usernameField,passwordField,submitButton"/>
  <input type="hidden" name="usernameField" value="<%= @user.ukma_login %>"/>
  <input type="hidden" name="passwordField" value="<%= @user.ukma_password
%>"/>
</form>

<script type="text/javascript">
  document.forms["submitter"].submit();
</script>

<%= render :partial => "/layouts/footer" %>

```

Висновки до розділів 3 і 4

В розділі описано принципи роботи СУНП та СУНК як обов'язкових складових навчального сердовища сучасних прогресивних вузів. Обґрунтовано відмінність та взаємодоповнюваність обох систем на прикладі MAMS і Moodle та їх необхідність в наш час стрімкого розвитку інформаційних технологій.

В ході даної роботи було розвинуто принципи роботи Системи Управління Навчальним Процесом (СУНП) шляхом розгортання MAMS на новому сервері. Здійснені перші кроки на шляху її інтеграції з рештою діючих електронних платформ НаУКМА. Перший крок являє собою спільну аутентифікацію з Pathos (хоч на даний момент це поки

швидше стратегічний хід, ніж реальний через поки що мінімальний перетин в аудиторії користувачів).

Сервер, на якому зараз працює система, введено в домен. Для експлуатації MAMS реалізовано доступ по веб (протокол *http*) та по *rdp* для віддаленого адміністрування чи підтримки. Для останнього обом серверам для АСУ призначені реальні IP адреси:

- 194.44.142.24 – «загадковий» сервер *enigma*;
- 192.44.142.197 – сервер *ukma-mams*

Відтак, можемо ствердно заявити, що система готова для подальшої експлуатації в реальних умовах.

5 Методи і засоби підтримки автоматизованих практикумів

Різні джерела пропонують відмінні одне від одного означення поняття «навчального практикуму», серед них можна виділити наступні:

Практикум (розгорнуте визначення) - форма організації навчального процесу, самостійне виконання учнями практичних і лабораторних робіт. Проводять, зазвичай, по завершенню великих розділів навчальних курсів або в кінці періоду навчання. Перелік робіт, які входять до практикуму, визначений в навчальній програмі. Перед практикумом проводиться інструктаж. Багато практичних робіт, які включено до практикуму, є дослідженнями і спрямовані на перевірку достовірності певних наукових закономірностей, припущень, гіпотез, та ін. Під час практикуму учні часто вирішують завдання творчого характеру (наприклад, конструкторські задачі). Вирішення таких задач передбачає не тільки розробку ідеї, але й демонстрацію її практичного втілення на прикладі, або моделі. [42]

Практикум - в спеціальних навчальних закладах: практичні заняття з деякої навчальної дисципліни. [43]

Практикум - учбово-практичне видання, що призначене для закріплення пройденого матеріалу і перевірки знань студентів різноманітними методами. Містить практичні завдання і приклади, що сприяють закріпленню пройденого матеріалу, основними різновидами практикумів є збірники прикладів та задач, збірники іноземних тестів, збірники описів лабораторних робіт (збірники описів практичних робіт, лабораторні практикуми), збірники планів семінарських занять, збірники контрольних робіт. [44]. Незважаючи на розмаїття визначень поняття «навчального практикуму», неважко виділити спільну основу, що присутня у кожному з них і дозволяє розуміти навчальний практикум як практичне заняття з окремої навчальної дисципліни.

Виходячи із сформульованого визначення, під час навчального практикуму студенти/учні мають змогу опрацьовувати та демонструвати набуті навички і закріплювати теоретичні знання, отримані під час лекцій та самостійної роботи, а також отримувати нові знання шляхом виконання навчальних вправ.

Навчальний практикум характеризується наступними властивостями:

1. можливість самостійного виконання практичного завдання, розв'язування задачі, тощо;
2. можливість перевірки або тестування одержаного результату самим виконавцем;
3. організація взаємодії студентів при виконанні завдання у складі групи виконавців;
4. консультації студентів один з одним або з викладачем;
5. взаємна перевірка (рецензування) робіт студентами;
6. перевірка робіт викладачем;
7. аналіз і виправлення допущених помилок.

Застосування інформаційно-комунікаційних технологій дозволяє значно підвищити ефективність практикумів за рахунок організації віртуального спілкування учасників навчального процесу та оперативного доступу до навчальних матеріалів та студентських робіт.

Процес навчання в середовищі електронного навчання здатний замінити частину персональної взаємодії віртуальною, яка може відбуватися як в он-лайн так і офф-лайн режимі. Засобами цифрової асинхронної комунікації є електронна пошта, форуми, дошки оголошень, тести і т.п.. Засоби цифрової синхронної комунікації є телеконференції, чати, і подібні їм засоби, що передбачають одночасну взаємодію учасників учбового процесу.

Як один із багатьох перспективних напрямів систем електронної освіти e-learning, планування та реалізація систем підтримки автоматизованих практикумів, основною метою створення яких є:

1. організація електронної взаємодії між викладачем та студентом в межах освітнього процесу;
2. структуризація множини виконаних студентами практичних завдань за номером, датою, роком навчання, тощо;
3. забезпечення можливості компіляції та тестування навчальних проєктів на веб-сервері у межах процесу перевірки та оцінювання набутих знань та навичок слухачів курсу

даний напрямок проектування ПО вимагає чіткого формулювання концепції робочого зошиту програміста у глобальній мережі, що тісно пов'язана із схемою роботи модуля

системи управління навчальним контентом Moodle і організацією електронних портфоліо, *вибірково акумулює і інтегрує риси обох підходів в одне ціле з метою отримання повнофункціональної системи*(даному випадку модуля) *хронологічної систематизації та тестування практичних робіт з програмування в межах e-learning.*

У першому підрозділі коротко описані основні сучасні напрями розвитку *e-learning* та визначене місце серед них концепції особистого робочого простору студента у мережі загалом, та *систем автоматизації навчальних практикумів зокрема*, розставлені базові акценти, яке функціональне навантаження несуть подібні системи на практиці і в якому вигляді вони мають бути представлені користувачеві(розглянуто на прикладі синтезу вже існуючих систем , та визначений вплив переходу до Web 2.0 на системи електронної освіти, до яких у повною мірою належить проєктований у рамках теми освітній модуль, інтегрований до міжнародного порталу студентського співтовариства JEP (Joint European Projects).

Реалізація системи підтримки електронного практикуму з програмування певним чином відрізняється від решти аналогів, оскільки, згідно до шостої характеристики навчального практикуму, вимагає специфічної форми перевірки виконаних завдань з програмування на віддаленому сервері, що є складною з точки зору технічної реалізації задачею і вимагає комплексного підходу до свого вирішення. У другому підрозділі будуть розглянуті основні передумови, що сприяли розвиткові концепцій віддаленої компіляції програмного коду(без необхідності встановлення середовищ програмування на локальній клієнтській машині). Визначається також, що передумови переходу від локального, до глобального ПЗ (у тому числі і стосовно систем e-learning) тісно пов'язані із численними перевагами так званої архітектури тонкого клієнту.

Також другий підрозділ коротко описує деякі із основних підходів, що стосуються віддаленої компіляції програмних кодів використовуючи стандартизований набір компіляторів для Unix-подібних систем - GCC(GNU Compiler Collection), на прикладі веб-інструменту Online-GCC будуть проілюстровані основні можливості та напрями розвитку новітніх Веб-орієнтованих середовищ програмування.

5.1 Формалізація поняття індивідуального робочого простору користувача в інтернет у межах електронної освіти

Електронне навчання(e-Learning або eLearning) - це загальний термін, застосовний до форми навчання, яка характеризується тим, що викладач та студент є розділеними у

просторі та часі і не взаємодіють безпосередньо, натомість взаємодія відбувається завдяки мережевим технологіям.

Окремі фргменти e-learning використовується у різних сферах освіти та бізнесу. Нерідко це стосується стратегій, що використовуються у локальних мережах фінансових компаній з метою поширення навчальних курсів серед співробітників. Університети, що практикують дистанційну освіту, широко використовують e-Learning, який визначається як *детально спланована взаємодія студента та викладача, що задіє цілий ряд переважно Інтернет-технологій з метою встановлення зв'язку із учнями на чималій території*. Значна кількість технологій може бути використана і активно використовується у Інтернет-навчанні, від Інтернет-блогів до інтегрованих робочих середовищ, електронні портфоліо та віртуальні лабораторії. Більшість навчальних ситуацій вимагають використання даних технологій у сукупності.

У порівнянні із термінами навчальні технології та інструкційні технології, термін Інтернет *Osvima(e-Learning)* позначає набагато більш загальні поняття технологій у сфері освіти, аніж комп'ютерні тренінги або інструкції. Це також більш широке поняття , ніж Інтернет-, або Дистанційне навчання, які насправді означають освіту, що базується суто на web-технологіях. У випадках використання мобільних технологій з метою навчання частіше використовують термін **M-learning**. Термін E-learning може бути застосовним не лише до технологій, що реалізують електронне навчання, але й описує суцільну картину процесу навчання із застосуванням комп'ютерних систем.

Системи E-learning зазвичай пристосовані до концепцій електронної освіти, проте також можуть бути використані, як об'єднуючий елемент із традиційним навчанням. У даному контексті найчастіше використовується термін Змішане навчання (**Blended learning**). У системі вищої освіти набувають популярності так звані Віртуальні Освітні Середовища, які сполучаються із *системами управління інформацією* з метою створення інтегрованих середовищ управління навчальним процесом, в яких усі аспекти курсів відображаються відповідно затвердженому стандарту інтерфейсів системи. Кількість традиційних та он-лайн університетів, що постійно збільшується почали пропонувати своїм учням сертифікатні програми через Інтернет із широким спектром рівнів акредитації та великим вибором навчальних дисциплін. У той час, коли деякі з означених програм вимагають особистої присутності студента на лекційних та практичних заняттях, чимало з них повністю дистанційні. На додаток, чимало університетів пропонують своїм учням електронні служби підтримки , на зразок он-лайн

порадника та запису на курси, можливості придбати підручники через Інтернет, студентські культурні заходи та інформаційні он-лайн газети.

e-Learning може також стосуватися освітніх web-сайтів на зразок:

- інформаційних освітніх порталів
- робочі зошити он-лайн
- інтерактивні списки завдань для учнів

Цей термін також активно вживаний у колах бізнесу і визначає професійні он-лайн тренінги для співробітників.

Оцінюючи рівень і стан розвитку інформаційних технологій в освіті, електронну освіту прийнято сприймати, як підхід до покращення і вдосконалення освіти шляхом використання персональних комп'ютерів, засобів мультимедіа та комп'ютерних мереж. Головне завдання електронної освіти полягає у покращенні доступу до навчальних ресурсів і служб та підтримці віддаленого обміну і співробітництва. Тому бачення електронної освіти виходить розробки і застосування окремої платформи електронного навчання. Воно обов'язково включає цілий комплекс заходів і засобів, до яких належать техніко-технологічні передумови, система управління навчальним процесом, засоби інформування і документування, засоби оцінювання знань і засоби спілкування. Якщо говорити про техніко-технологічні передумови, то необхідний мінімум складають мережі пропускної спроможності, достатньої для передачі мультимедійних матеріалів і підтримки особистих навчальних середовищ кожного учасника навчального процесу.

Залежно від потреб курсу кваліфікації і можливостей учасників технічних умов використання навчального центру допускає різні сценарії:

Сценарій першого рівня передбачає просте розповсюдження інформації, практично система зводиться тоді до рівня електронної бібліотеки та системи лінків та дозволяє використання навчальних матеріалів навіть без персоніфікованого доступу.

Сценарій другого рівня, назвемо його інтерактивним, передбачає взаємодію системи із користувачем. Кожен учасник навчального процесу одержує особисте навчальне середовище, система відстежує і веде прогрес навчального процесу. Сюди відносяться

віртуальні наочні мультимедійні і анімаційні засоби, засоби опитування, тестування та оцінювання знань, відповіді на типові питання, тощо.

Сценарій третього рівня, назвемо його колаборативним, передбачає співпрацю учасників навчального процесу як в автономному, так і в оперативному режимах. Платформа електронної освіти поряд із звичайними зсобами підтримки електронної пошти, групових розсилок, форумів, чатів, календарів, повинна виходити на рівень віртуальних навчальних заходів: лекцій, семінарів і практикумів, забезпечувати можливість групової розробки навчальних проектів у віртуальному середовищі.

До функцій центру електронної освіти, крім, власне, супроводу навчального порталу та хостінгу, належать також надання консультативних послуг, адміністративно-організаційні заходи та методична підтримка процесу електронного навчання. У сукупності комплекс заходів, здійснюваних центром електронної освіти, створює додану вартість для осіб, що навчаються, яку можна визначити як ефективне навчання у співпраці за рахунок:

- Доступності матеріалів для всіх студентів;
- Сприяння фаховій співпраці студентів з викладачем;
- Незалежності від місця і часу;
- Сприяння підвищенню рівня самоосвіти;
- Розширення власної цифрової компетенції;

Додана вартість для викладача перш за все визначається підвищенням репутації навчального закладу за рахунок:

- Розширення власного портфоліо;
- Утворення науково-навчальної мережі;
- Підвищення шансів на залучення сторонніх коштів;
- Появи технічної інфраструктури для розбудови мережі підвищення кваліфікації;
- Незалежність від місця і часу(наприклад для сумісників);
- Підвищення ефективності викладання, спілкування і керівництва студентами.

Соціальні дискусійні групи - ця педагогічна схема взаємодії студентів особливо гарно зарекомендувала себе в якості дискусійних форумів, блогів та інтегрованих он-лайн робочих середовищ. Така форма мережевої співпраці в межах e-Learning

відкриває нові можливості для створення освітнього контенту для ширшої аудиторії, включаючи самих студентів.

Модель он-лайн взаємодії Лауріалда - ідентична до базової моделі побудови контенту у eLearning і надає можливість використовувати так звані дискусійні дошки.

Модель когнітивної перспективи - сфокусована на процесі пізнання та засвоєння знань.

Емоційна перспектива - фокусується на емоційній складовій знань, на зразок мотивацій, належності до цілого, зацікавленості.

Поведінкова перспектива - фокусується на поведінкових схемах учасників навчального процесу(емуляція соціальних та професійних ролей)

Контекстуальна перспектива - фокусується на соціальних аспектах, що можуть стимулювати освітній процес. Взаємодія з іншими людьми, *колаборативні напрямки досліджень та важливість впливу оточуючих* – це ті напрямки, яких торкається контекстуальна перспектива розвитку e-learning.

5.2 Можливість повторного використання матеріалів, стандарти та освітні об'єкти

Можливість повторного використання матеріалів в системах електронної освіти потребує значних зусиль, а особливо ті, що стосуються створення повторно - використовуваних *Освітніх Об'єктів*. Це самодостатні інформаційні одиниці, що містять теги у якості внутрішніх зв'язків, або так звані *метадані* та часто зберігаються у форматі файлів [XML](#). Створення навчального курсу вимагає зіставлення в одне ціле обумовленої послідовності навчальних об'єктів . Для зберігання навчальних курсів використовують різноманітні навчальні репозитарії, що можуть бути комерційними та некомерційними.

Стандарт, розроблений спеціально для e-learning, SCORM(Sharable Content Object Reference Model) дозволяє транспортувати «навчальні об'єкти» або категоризовані метадані(LOM).

Більшість стандартів e-learning знаходяться на ранній стадії розвитку - найдавніші з них нараховують не більше 8 років, тому мають чимало вад і не завжди відображають реальні потреби сучасної електронної освіти.

5.3 Формулювання основних принципів концепції індивідуального електронного «робочого зошита» з програмування на віддаленому сервері)

Згідно до описаних вище аспектів переходу до стандартів Web 2.0, індивідуальний робочий простір слухача довільного навчального курсу може бути охарактеризований як mash-up наступних веб-сервісів:

- Електронних портфоліо або Webfolio;
- Окремих модулів електронної системи управління навчанням Moodle, а саме: модуль завдань, модуль користувачів, модуль підтримки практикумів.

Враховуючи особливості вимог до проектування он-лайн робочих зон в рамках системи електронної освіти, варто зазначити, що властивості обох згаданих веб-сервісів стосуються означуваної системи лише частково. Так, *односторонність* та *відсутність взаємодії* «викладач-студент», якими характеризується поняття електронного портфоліо, вступає у протиріччя з основним принципом побудови колаборативних середовищ, частиною якого індивідуальний робочий зошит з програмування:

Зазначені у списку модулі Moodle, навпаки, мають розвинену схему взаємодії «студент-викладач», але їм бракує хронологічності, довготривалості освітніх процесів та сторонньої оцінки виконаних робіт, яка повною мірою реалізована у системах Webfolio.

Наступні пункти визначають поняття електронного портфоліо і коротко описують основні властивості модулів Moodle, які допомагають наблизитися до розуміння базових характеристик індивідуального он-лайн робочого зошита з програмування.

5.4 Властивості електронних Портфоліо

За визначенням *портфоліо* - це набір наукових або навчальних робіт, які зібрані у колекції з метою демонстрації навчального прогресу особи у часі для демонстрації її вмінь та навичок. Портфоліо можуть належати до певної дисципліни, або дуже широко описувати історію набуття нових знань окремою особистістю протягом життя. У контексті теми даної кваліфікаційної роботи в якості прикладу буде розглядатися

перший тип портфоліо. У якості одиниць вмісту окремого портфоліо може використовуватися велика кількість форматів файлів:

- Відео
- Дослідницькі проекти
- Фотографії
- Рукописи
- Відгуки викладачів та співучнів

Основною метою створення електронного, як і будь-якого іншого, портфоліо *є одержання авторитетного відгуку на вкладені матеріали*[45].

Двома останніми ключовими пунктами є те, що електронні портфоліо візуально *демонструють навчальний та дослідницький прогрес особи у часі* [46], та те, що конструювання власного портфоліо *є невід'ємною частиною навчального процесу* [47].

Усі три ключові поняття, що стосуються електронного портфоліо, у деякій мірі можуть бути застосовані до формулювання ключових концепцій та ролей, які відіграє у процесі навчання «електронний робочий зошит» студента з програмування.

5.5 Властивості функціональних модулів MOODLE

Модуль завдань

- Завдання можуть бути охарактеризовані датою здачі і максимальною оцінкою;
- Студенти можуть завантажувати свої завдання на сервер(у будь-якому файловому форматі);
- Пізно виконані завдання дозволяються, але час запізнення показується викладачу;
- Для кожного окремого завдання може бути оцінений цілий клас(оцінка і відгук) на одній сторінці в одній формі;
- Відповідь викладача приєднується до сторінки із завданням кожного студента і надсилається попередження;
- Викладач може дозволити перездачу завдань після оцінювання.

Підтримка користувачів

- Метою є скорочення впливу адміністратора до мінімуму, проте забезпечення

високого рівня безпеки;

- Підтримка ряду механізмів аутентифікації впроваджує легку інтеграцію з системами, що існують;
- Стандартні методи email: студенти можуть створювати власні логін-профілі. Поштові адреси перевіряються підтвердженням;
- LDAP-метод: логін-профілі можуть бути перевірені на LDAP-сервері. Адміністратор може вказати, які поля використовувати. IMAP, POP3, NNTP логін-профілі перевіряються на пошті чи сервері новин. SSL, сертифікати і Tь8 підтримуються;
- Зовнішня база даних: будь-яка база даних, що містить принаймні два поля може використовуватися як зовнішнє джерело аутентифікації;
- Кожній людині необхідний лише один профіль на цілий сервер - кожен профіль може мати різний доступ;
- Адміністратор контролює створення курсів і створює профілі викладачів, записуючи користувачів на курси;
- Автор курсів лише створює курси та публікує їх;
- Викладачі можуть не мати привілеїв, які б дозволяли їм модифікувати курси (наприклад, позаштатні викладачі);
- Безпека - викладачі можуть додавати «реєстраційний ключ» до власних курсів для неможливості прослуховування їх не студентами. Ключ може передаватися особисто, чи електронної поштою;
- За необхідності викладачі можуть вручну записувати студентів;
- Викладачі також можуть вручну виписувати студентів, у іншому випадку студенти автоматично виписуються після деякого періоду часу(визначеного адміністратором);
- Заохочується створення студентами он-лайнних профайлів, включаючи фотографії, описи. При бажанні поштові адреси можуть не висвітлюватися;
- Кожен користувач може обрати мову для інтерфейсу Moodle (англійську, французьку, німецьку та ін.) записуючи користувачів на курси.

Підтримка практикумів

- Лектор має повний контроль над усіма налаштуваннями, включаючи обмеження щодо інших викладачів;
- Курси можуть обиратися в залежності від тижня, теми, дискусії;
- Останні зміни до курсів від часу минулого входження в систему можуть

висвітлюватися на домашній сторінці курсів;

- Усі оцінки за форуми, журнали, квізи завдання можуть бути переглянуті на спеціалізованій сторінці;
- Практикуми можуть бути запаковані у стандартний zip-архів.

Наступний підрозділ містить базову описову характеристику тонкого клієнта та методи організації он-лайн компіляції навчальних проектів з програмування, що складають основну вимог до функціональності проекту електронного робочого зошиту з програмування.

Згідно з методом структурної декомпозиції, такий список властивостей цільового модуля під назвою "Store Space":

1. Аутентифікація та супровід користувачів: реалізація базових ієрархій та ролей;
 - a. Планування та реалізація функцій профіля «Лектор»
 - b. Планування та реалізація функцій профіля «Викладач»
 - c. Планування та реалізація функцій профіля «Студент»
2. Файловий репозитарій студентських програмних практикумів, підтримка деревовидної структури каталогів;
 - a. Структуризація проектів
 - b. Програмна реалізація деревовидної структури каталогів
3. Віддалена компіляція C++ проектів: Застосування GCC;
4. Оцінювання та робота з часом.
 - a. Оцінювання студентських практикумів
 - b. Хронологічний підхід до організації практикуму
 - i. Організація контролю над строками здачі проектів
 - ii. Автоматизація сортування профілів студентів за часом реєстрації.

5. Задача інтеграції систем "Store Space", "E-folio" та системи підтримки навчальних курсів у мережі.

Наступні пункти буде присвячено висвітленню кожного елемента списку характеристик цільової системи.

Аутентифікація та супровід користувачів: реалізація базових ієрархій та ролей

Планування та реалізація функцій профіля «лектор»

Згідно до вимог до цільової системи, користувач із повноваженнями «Лектор» повинен володіти всіма доступними можливостями модуля управління автоматизованими практикумами «Store Space», серед них:

1. Додавання та видалення нових та вже зареєстрованих користувачів системи;
2. Перегляд комплексного дерева каталогів студентських робіт за всі роки навчання;
3. Управління всіма студентськими програмними проектами, що були поміщені у базу даних ресурсу:
 - Додавання
 - Видалення
 - Перегляд
 - Компіляція
4. Надання зареєстрованих на ресурсі користувачам повноважень «Викладач»
5. Призначення перескладання проєктів
6. Оцінювання знань всіх підписаних на курс студентів.

Система реалізована так, що підтримує лише одного користувача із повноваженнями «Лектор», він визначається суперадміністратором порталу JEP.

Планування та реалізація функцій профіля «Викладач»

Користувачі із повноваженнями «Викладач» мають подібні права у системі до «Лекторів», за відмінністю, що кількість викладачів у системі необмежена. «Лектор» призначає «Викладачів» із окремого списку зареєстрованих на порталі користувачів.

«Викладачам» надаються такі повноваження в системі підтримки автоматизованих практикумів з програмування «Store Space»(Рисунок 5.1)

1. Додавання та видалення нових та вже зареєстрованих користувачів системи, за умови, що користувач вказав, що належить до групи даного «Викладача»;
 2. Перегляд комплексного дерева каталогів студентських робіт за всі роки навчання власної групи;
 3. Управління всіма студентськими програмними проектами, що були поміщені у базу даних ресурсу студентами окремого «Викладача»:
- Додавання
 - Видалення
 - Перегляд
 - Компіляція
4. Призначення перекладання проєктів
 5. Оцінювання знань студентів, що належать до групи даного викладача.

store space

» Список студентів

» Управління лабораторними роботами

» Оцінки

» Вихід

Welcome vlad

» Admin Area

» Settings

» Profile

» Logout

» LAN_EFOLIO

» GForge

Since your last visit there have been no news items, no forum posts

Select Language

English

Set Language

Global Projects

ITSoftTeam

EduVisIm

Нові бажанчі записатися на курс

ID	USER NAME	LOGINNAME	SS ADMIN			
1	DRUSHA	DRUSHA	1	DRUSHECKA@ukr.net	19/05/2008 02:33:32	<div></div> <div></div>
2	Argus	Argus	1	sanchiku@mail.ru	19/05/2008 02:34:07	<div></div> <div></div>
3	ingo-eric	ingo-eric	1	iem.schmidt-braul@iba-berlin.de	19/05/2008 02:34:27	<div></div> <div></div>
4	Lyova	Lyova	1	olex_levytsky@ukr.net	19/05/2008 02:34:33	<div></div> <div></div>

Зареєстровані слухачі курсу

ID	USER NAME	LOGINNAME	SS ADMIN			
1	rage	rage	0	dkrasikov@gmail.com	13/05/2008 03:16:53	<div></div>
2	hamster	hamster	1	hamsterr1988@gmail.com	19/05/2008 02:29:25	<div></div>
3	Andrew Afonin	Andrew Afonin	0	andrew.afonin@gmail.com	13/05/2008 02:37:52	<div></div>
4	phantom	phantom	1	anton.shabinskiy@gmail.com	19/05/2008 02:20:22	<div></div>
5	marined	Uruk	3	asdass	16/05/2008 20:56:48	<div></div>

Рисунок 5.1 STORE SPACE

5.6 Планування та реалізація функцій профілю «Студент»

«Студентам» у системі підтримки автоматизованих практикумів надається найменша кількість повноважень. Серед них:

1. Підпис на курс та доступ до матеріалів через суміжний модуль управління курсами;

2. Перегляд та компіляція лише власних програмних проєктів;
3. Додавання та видалення вмісту виділених даному «Студентові» робочих каталогів;
4. Можливість перегляду балів, отриманих за проєкти.

Підписатися на курс може кожен зареєстрований в системі порталу ШП користувач, далі його запит на реєстрацію переглядається користувачем із повноваженням «Лектор», або «Викладач» і у разі його схвальної відповіді, додається до списку записаних на курс слухачів(Рисунок 5.2).

Рисунок 5.2. Запис на курси.

Файловий репозитарій студентських програмних практикумів, підтримка деревовидної структури каталогів

Структуризація проєктів

Згрупованість та підтримка визначеної структури відіграє вирішальну роль у коректній роботі модуля управління навчальними практикумами та ефективній контролюючій місії викладача. Каталоги із студентськими проєктами структуровані наступним чином:

Рік навчання -> викладач -> номер групи -> ім'я студента(латинськими літерами) -> номер лабораторної роботи -> файли проєкту (доступні формати: .h, .cpp)

Кожен студент матиме ту кількість каталогів, яка визначена викладачем у полі «Лабораторні роботи».

Програмна реалізація деревовидної структури каталогів

Деревовидна структура каталогів найкраще відображає структуру навчальних проектів і попереджує утворення безладу у файловій системі. Реалізована за допомогою мови Веб-застосувань JavaScript, деревовидна структура каталогів дозволяє легко маніпулювати теками, приховуючи та демонструючи їх в залежності від повноважень поточного користувача системою.

В залежності від повноважень користувача, кожен файл може бути переглянутий у тестовому редакторі, відкомпільований та видалений.

5.7 Віддалена компіляція c++ проектів: застосування gcc

ОСС часто вибирається для розробки програмного забезпечення, що повинно працювати на різних апаратних платформах. Розходження між «рідними» для кожної з апаратних платформ компіляторами приводять до труднощів при розробці коду, який би коректно компілювався різними компіляторами. При використанні GCC для компіляції коду під різні платформи використовується один і той же синтаксичний аналізатор. Тому якщо вдалося зібрати програму для однієї із цільових платформ, те велика ймовірність, що програма нормально збереться й для інших платформ.

Зовнішній інтерфейс GCC є стандартом для компіляторів на платформі Unix. Користувач викликає керуючу програму, що називається *gcc*. Вона інтерпретує аргументи командного рядка, визначає й запускає для кожного вхідного файлу свої компілятори потрібної мови, запускає, якщо необхідно, асемблер і компоувальник.

Команда *G++* викликає компілятор для програм написаних на C++. При подальшій розробці проекту є можливість розробити універсальне середовище, що буде компілювати, злінковувати та генерувати виконавчі файли для таких мов як C, OBJECTIVE-C, Java, Fortran і Ada, і тому можна буде в повній мірі реалізувати роботу з набагато більшою кількістю курсів з об'єктно-орієнтованого програмування.

5.8 Оцінювання та робота з часом

Оцінювання студентських практикумів

Студентські практичні роботи з програмування, подібно до робіт із інших дисциплін, можуть бути переглянуті та оцінені користувачем із повноваженнями «Лектор», або «Викладач». За основу взята найпростіша система оцінювання якості виконання

практичної роботи з програмування -компіляція проекту із подальшим його переглядом та виведенням оцінки, виходячи із якості виконаної роботи. Система оцінювання представлена у вигляді таблиці із стовпцями, що відповідають за номер лабораторної роботи і рядками, в які внесені прізвища слухачів курсу. Передбачена можливість викладачеві власноруч заповнювати відповідні позиції таблиці балами.

Хронологічний підхід до організації практикуму

Окрім якості написання коду та відповідності результатів компіляції програмного проекту визначеному зразку, система управління навчальними практикумами надає можливість узгоджувати час здачі окремих проектів із передбаченими для виконання кожного з них строками. Проекти, що були здані із запізненням маркуються відповідно у списку зданих лабораторних робіт.

Організація контролю над строками здачі проектів

Невчасно здані проекти можуть бути відхилені викладачем. Також існує можливість реабілітації проектів минулих років на вимогу студента-автора з метою демонстрації, або аналізу результатів роботи програм.

Автоматизація сортування профілів студентів за часом реєстрації

Застарілі та неактуальні списки студентів - проблема, яку доводиться вирішувати більшості сучасних LMS в межах e-Learning. Неможливість структурування та відсіювання неактуальних даних стосовно складу студентських груп згідно до періода навчання нерідко призводить до захаращування та втрати чіткої структури каталогів. З метою запобігти цьому було розроблено програмний фільтр, що сортує студентів за вказаним ними роком навчання і виводить лише ті каталоги, які належать студентам, що прослуховують курс C++ на поточний момент часу, решта каталогів переводиться системою у стан тимчасової неактивності.

Задача інтеграції функціональних модулів "store space", "efolio" та системи підтримки навчальних курсів у мережі

Окрім модуля підтримки електронних практикумів з програмування, який було розроблено як результат теоретичних досліджень в межах теми даної кваліфікаційної роботи, у напрямку розширення функціональності міжнародного освітнього порталу JEP також розроблялися суміжні модулі підтримки електронних курсів та студентські

електронні портфоліо. Дані модулі контекстуально пов'язані і доповнюють функціональність одне одного. Так, модулі підтримки електронних практикумів з програмування та електронних курсів поділяють доступ до спільної таблиці `lab_tabl`, що містить список лабораторних робіт, доступних для виконання у межах даного курсу. Такий вид зчеплення модулів називають **зчепленням за спільною областю пам'яті**. Модуль, що реалізує студентське електронне портфоліо, пов'язаний із «Store Space» найбільш вдалим з точки зору компонентно-орієнтованого програмування видом зчеплення - **параметричним зчепленням**, оскільки модулі викликають методи одне одного і активно обмінюються параметрами.

Не зважаючи на те, що вихідна система має окремі вади та неточності, у тому числі неможливість генерації тестових випадків для програмних проєктів, та відсутності профіля «Зовнішнього рецензента» програмних робіт, вихідна система являє собою комплексний функціональний продукт, який, як і багато інших, покликаний оптимізувати систему електронної освіти та суттєво зменшити незручності, що виникають в результаті несумісності компіляції одних і тих самих мовних структур програмного коду на різних платформах.

Висновки до розділу 5

Електронна освіта (e-Learning) виходить за межі дослідницьких лабораторій, охоплюючи не лише гуманітарні та природничі дисципліни, але й технічні, зокрема основи програмування. Поступово виникають новітні технічні засоби та програмні рішення, які дозволяють не лише ефективно проводити та оцінювати студентські практикуми з програмування без особистісної взаємодії «викладач-студент», але й, базуючись на архітектурі тонкого клієнта, не вимагають для успішної компіляції навчальних проєктів кропіткої установки коштовного та ресурсомого програмного забезпечення на локальному комп'ютері користувача на зразок середовищ програмування VisualC++, C++ Builder. Системи на зразок онлайн ССС вже сьогодні дозволяють створювати власні учбові та науково-дослідницькі проєкти у мережі Інтернет та успішно компілювати їх на віддаленому сервері з будь-якої точки світу, маючи лише пристрій доступу до всесвітньої мережі (ПК/мобільний пристрій) та веб-браузер. Незважаючи на технологічну та функціональну недосконалість подібних систем на сьогодні, варто відмітити, що перший крок у напрямку перенесення локального робочого простору програміста у глобальну мережу було зроблено, і враховуючи бурхливий розвиток Веб-технологій та поступове поширення ПЗ на базі архітектури

тонкого клієнта, серед розробників програмного забезпечення інтерес до он-лайн середовищ з часом лише зростає.

Результатом досліджень деяких із технологічних особливостей набору компіляторів GCC та методів електронної освіти, як webfolio та окремих функціональних компонентів платформи електронної освіти Moodle, стала система підтримки автоматизованих навчальних практикумів з ООП. Процес проектування та розробка цільової системи входив у рамки компонентно-орієнтованого програмування, оскільки об'єкт розробки являв собою спеціалізований функціональний модуль, приєднуваний до порталу міжнародного освітнього проекту JEP(Joint European Projects) і певним чином інтегрований із суміжними модулями efolio та системою підтримки навчальних курсів. До списку властивостей спроектованого модуля підтримки автоматизованих практикумів входять:

- Збереження та компіляція програмних проєктів на сервері;
- Хронологічна систематизація практикумів та учасників навчального процесу.

Однією із найважливіших цілей створення даної системи можна назвати задачу інтеграції систем "Store Space", "Efolio" та системи підтримки навчальних курсів у мережі.

Разом три зазначені модулі утворюють комплексну повнофункціональну систему підтримки електронного навчання, яку можна охарактеризувати наступним чином:

Контент (модуль підтримки навчальних курсів та лекційних матеріалів) + зворотній зв'язок(модуль підтримки автоматизованих практикумів) + історія навчання(модуль Efolio)

Незважаючи на те, що описана система має окремі недоліки, пов'язані із недосконалістю систем електронного захисту від несанкціонованого доступу та не найкращими методами зчеплення модулів, проектування подібних програмних проєктів є важливим кроком на шляху розвитку електронної освіти в цілому.

Існує велика ймовірність, що з часом проектування он-лайн робочих середовищ вийде на рівень з соціальними мережами і стане одним із найбільш перспективних напрямів розвитку Інтернет-технологій.

6 Розробка інтерактивних навчальних матеріалів

6.1 Вивчення курсів з програмування в системах електронної освіти

Для управління навчальним процесом зазвичай використовуються LMS системи.

Варто згадати про такі існуючі LMS платформи:

- ATutor - Open Source LCMS
- Claroline – безкоштовна LMS
- Dokeos – веб-застосування для електронного навчання та управління курсами
- eFront - Open Source LMS
- Fle3 – віртуальне навчальне середовище
- ILIAS - Open Source LMS
- KEWL.nextgen – безкоштовна платформа для електронного навчання
- LON-CAPA – відкрита LMS з пов'язаними доменами
- Moodle – відкрита CMS (Course Management System)
- OLAT - Open Source LMS
- Sakai Project – колаборативне та навчальне середовище

Системи управління навчанням поступово розвиваються в системи

управління навчальним контентом, що являє собою мультикористувацьке середовище, де студенти можуть створювати, зберігати, застосовувати нове використання (reuse), управляти матеріалами електронного навчання, що зосереджені в певному центральному репозиторії [48]. Різниця очевидна: наприклад, LMS, не може створювати і маніпулювати навчальними курсами, не може повторно використовувати матеріали одного курсу для розбудови іншого. ЇСМ8 дозволяє створювати та контролювати не тільки модулі навчання, але й взагалі весь каталог курсу, керувати зв'язками тощо. Зараз часто використовують узагальнене поняття CLCIMS (Computer Learning Content Information Management System) для визначання навчальних середовищ.

Все ж в своїй більшості всі згадані системи в основному спрямовані на організацію взаємодії між викладачем та студентом (завдяки тематичним форумам, формам контакту, організації здачі домашніх завдань тощо). Також іноді використовуються портфоліо, основна задача яких - представлення/збереження матеріалів певного суб'єкту процесу навчання. Всі ці рішення мають певні недоліки з огляду на специфіку технічних наук:

а) відсутність орієнтації на специфічні формати матеріалів з конкретної галузі (в даному випадку програмування);

- b) відсутність інтерактивності при роботі з матеріалами з різних мов програмування;
- c) часові затрати на демонстрацію/виконання/від лагодження практичних робіт

Також важливим інструментом є *віртуальний клас*. Він є користувацьким ядром освітнього ІТ-середовища і являє собою комплексну розподілену систему. В неї зазвичай входять інфраструктурні програмні й технічні компоненти, що віртуально поєднують робочі місця викладача та студентів в навчальну групу, яка працює в мережі (локальній або глобальній). Тим не менше віртуальний клас також не вирішує проблему представлення навчальних матеріалів з ООП, він орієнтований на інтерактивну взаємодію учасників навчального процесу.

6.2 Аналіз засобів створення та супроводу робочого зошита викладача

Робочий зошит викладача - це те середовище, де буде зосереджено всі навчальні матеріали, які викладач забажає розмістити, причому це не просто репозитарій, а певна динамічна структура, яка повинна мати належну функціональність, адже цей зошит використовуватимуть і викладачі, і студенти як на лекціях та практичних, так і при «віддаленому навчанні». Отже, вимоги повинні бути такі:

- має бути суцільним середовищем
- можливість швидкої демонстрації виконання програм, без потреби «збирати» проект на різних локальних машинах
- можливість працювати як з демонстративними матеріалами (демонстраційні програми), так і матеріалами - завданнями для студентів
- керування своїм власним робочим простором
- підтримка віртуальних класів (більше стосується інтеграції з LMS)

Серед готових рішень, на які можна орієнтуватись, можна виділити такі:

- стандартні засоби відомих LMS (що поступово переходять в LCMS) для наочного представлення матеріалів
- системо eFolio (часто входять до складу LMS).

Також є певні функціональні рішення, які певною мірою вирішують проблему інтерактивності, але все ж потребують централізованості:

- флеш-уроки (наприклад курси CISCO з мережних технологій)
- он-лайн компілятори

Отже, очевидно, що постає потреба в продукті, що об'єднував би в собі всі кращі сторони всіх вищезгаданих програмних продуктів, дозволяючи таким чином централізувати початковий процес, але з іншої сторони не був би занадто ускладненим, даючи змогу ефективно використовувати його ресурси.

6.3 Поняття веб-порталу та платформи в e-learning

Веб-портал з електронної освіти надає студентам доступ до інформації, в яку входять всі необхідні ресурси навчання й послуги. Короткий список типових компонентів

порталу:

Організація

Навчальні плани

Розклади

Оголошення

Програми курсів

Список реєстрацій

Інформація й документація

a. Бібліотеки

b. Оригінали лекцій та примітки

c. Супровідні лекційні матеріали: тексти, графіка, відео й звукові матеріали

d. Презентації

e. Підручники

f. Практичні завдання

g. Домашні завдання

h. Webfolio

Оцінювання

a. Записи оцінювань

b. Тести

c. Само оцінювання

d. Зразки іспитів

Комунікація

a. Віртуальна лабораторія

b. Навчальні проекти, (командна) розробка

c. Форуми: Діалог з викладачем, студентські дискусії

d. Он-лайн комунікації: (веб-трансляції) лекції, потокове відео

e. Семінари он-лайн (webinars), веб-конференції

Корисно буде коротко описати поняття платформи електронного навчання, досить близьке до поняття порталу. *Платформа Е-навчання* - спеціалізоване програмне забезпечення, *віртуальне середовище навчання* або *система керування навчанням (LMS)*.

Це програмне забезпечення що здатне забезпечувати послуги для розвитку курсів е-навчання. LMS – це зазвичай розподілена он-лайн система, що має вихід до Інтернету через інтерфейс *веб-порталу*.

Е-навчання приносить новий вимір у процес дослідження. Відповідно до древнього китайського прислів'я “я чую, і я забуваю, я бачу, і я пам'ятаю, я роблю і я розумію” в такий спосіб навчання через виконання роботи стає популярнішим. Із введенням е-навчання процес дослідження став практичнішим й активним, і менш академічним. В е-навчанні студент вже не тільки одержувач інструкцій; в е-навчанні студент стає членом багатьох освітніх програм розвитку або дослідження. Всі дії відкриті, щоб бути перевіреним й оціненими широким академічним співтовариством, в яке можуть входити однокурсники, викладачі, майбутні роботодавці тощо. Контроль сприяє покращенню всього процесу.

Одна з розповсюджених помилок в контексті е-навчання, полягає в тому, що вважають, ніби е-навчання необхідно використовувати при віддаленому процесі освіти. Насправді оптимальний варіант - це *змішане навчання*, що комбінує різні методи навчання, наприклад контактний і віртуальний.

Дослідження стверджують, що студенти надають перевагу курсам змішаного типу (гібридні). Гібридні курси використовуються в 40 % американських університетів число пропозицій росте більше ніж на 10 % щорічно.

6.4 Розробка системи управління навчальними матеріалами в робочому зошиті викладача

При розробці структури проекту основна увага була зосереджена на тому щоб зробити максимально простий для користувача, і водночас здатний задовольнити вимоги викладачів інструмент. Така концепція і була запропонована. На рисунку 6.1 зображений вигляд сторінки при перегляді матеріалів курсу:

Навчальні курси

Навчальний курс:

Object Oriented Programming with C++

Автор (викладач):

DmytroIvchenko (Переглянути всі курси автора)

Опис курсу:

Концепції сучасного програмування в рамках парадигми процедурно-орієнтованого програмування, абстрактних типів даних (об'єктне програмування) і об'єктно-орієнтованого (ієрархічного) у їх розвитку і взаємозв'язку. Основу вивчення складають поняття типу, функції, в тому числі узагальненої, об'єкта і класу, в тому числі параметризованого. Проблеми розширення областей визначення функцій, специфікації класів, побудови ієрархії об'єктів і класів, зокрема поліморфізму, успадкування інтерфейсу і реалізації.

1. МООП II

Проекти C++:

- Тестовий проект в МООП
- тестовий приклад №5

2. Вступ до C++!!!

Проекти C++:

- Лабораторна №2
- тестовий приклад 4
- Лабораторна 23

Вкладення:

- завантажити лекцію
- Вкладення Joomla!

Рисунок 6.1. Сторінка перегляду матеріалів курсу

Як видно з зображення, при перегляді курсу можна продивлятися самі матеріали, також проглядати код лабораторних робіт, запускати ці лабораторні роботи на виконання і переглядати результат цього виконання, читати коментарі викладача. Також можна працювати зі звичайними вкладеннями. Викладач, в свою чергу, може змінювати код програм, і також одразу запускати їх на виконання, таким чином маючи змогу ефективно демонструвати поставлені задачі на лекції/практичному занятті. Також викладач може контролювати хід навчального процесу, активуючи або дезактивуючи відповідні матеріали.

Ця ж сторінка при перегляді викладачем (автором курсу) виглядає вже так:

Рисунок 6.2. Вигляд сторінки з погляду викладача

Створення інтерактивних навчальних матеріалів з об'єктно-орієнтованого програмування

























































































Редагування контенту

Навчальний курс:

Object Oriented Programming with C++

Концепції сучасного програмування в рамках парадигм процедурно-орієнтованого програмування, абстрактних типів даних (об'єктне програмування) і об'єктно-орієнтованого (ієрархічного) у їх розвитку і взаємозв'язку. Основу вивчення складають поняття типу, функції, в тому числі узагальненої, об'єкта і класу, в тому числі параметризованого. Проблеми розширення областей визначення функцій, специфікації класів, побудови ієрархії об'єктів і класів, зокрема поліморфізму, укладування інтерфейсу і реалізації.

Опис курсу:

1. МООН II								
Проекти C++:								
> Тестовий проект в МООН	   							
> тестовий приклад №5	   							
2. Вступ до C++!!!								
Проекти C++:								
> Лабораторна №2	   							
> тестовий приклад 4	   							
> Лабораторна 23	   							
Вкладення:								
+ завантажити лекцію	 							
+ Вкладення Joomla	 							
+ тимчасово неактивно	 							

Графічно зв'язки можна зобразити таким чином Рисунок 6.3:

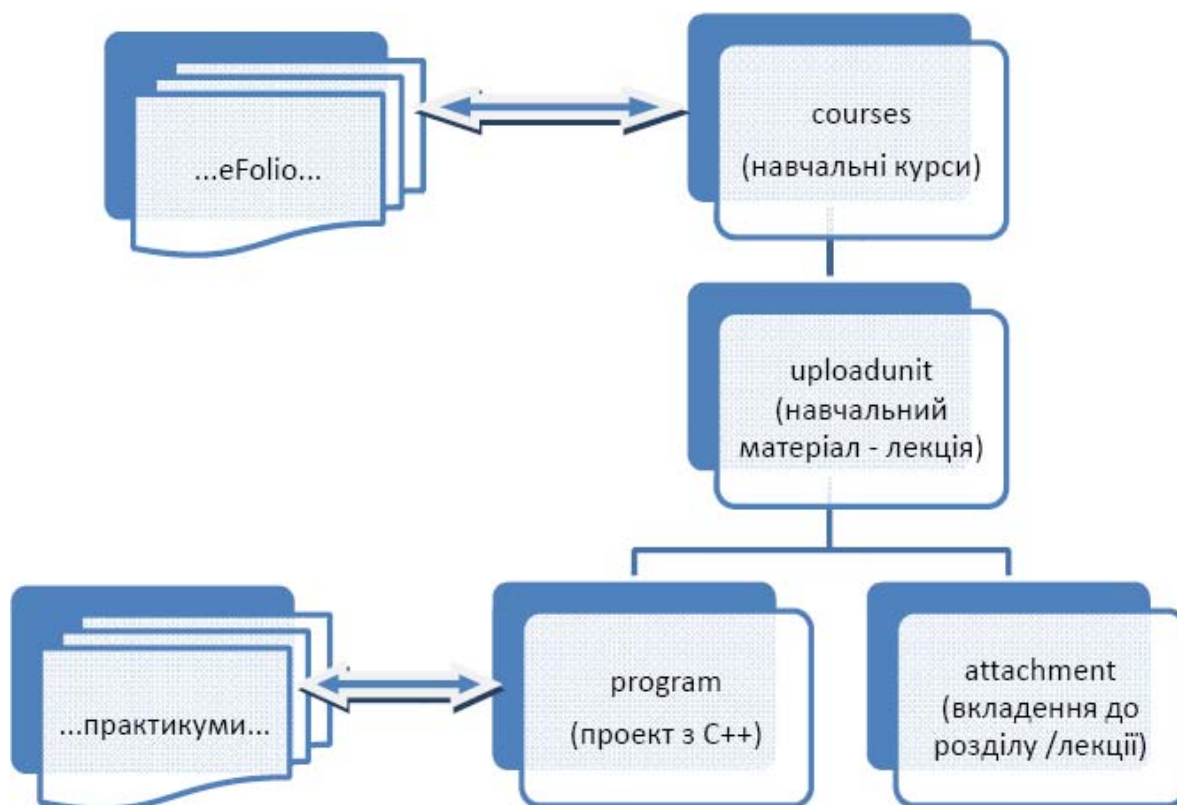


Рисунок 6.3. Зв'язки таблиць баз даних

В правій частині зображена схема представлення даних в LCMS, стрілками показані динамічні зв'язки з системою електронного портфоліо та студентських автоматизованих практикумів. Така схема водночас робить процес ознайомлення з новим веб-інструментом, максимально інтуїтивним, забезпечує відповідну функціональність та дає змогу розвивати цю систему в майбутньому без перешкод. Це також стосується он-лайн компіляції не тільки проектів з C++, а будь-яких інших практикумів, написаних на мові, що підтримується бібліотекою GCC, але про це буде сказано в наступних розділах.

Впровадження основних концепцій:

Створення/перегляд курсів та їх розділів

Процес додавання контенту реалізовано таким чином: викладач створює курс, до створеного курсу додає матеріал у вигляді веб-сторінки або флеш-презентації, до цього матеріалу може додавати проекти з C++ (можливо також з інших мов програмування) або будь-які вкладення. Всі результати цих дій відображаються як на головній сторінці плагіну, так і на сторінці редагування у автора курсу.

Сумісність форматів, конвертація в HTML, FLASH

Веб-сторінки, що додає користувач (автор курсу) - це файли, отримані шляхом конвертації документів (зазвичай Місгозой ^огд) в формат HTML. В даній роботі навмисне цей процес покладено на плечі авторів курсу, адже це запобігає проблем сумісності форматів при автоматичній конвертації, а також робить систему створення інтерактивних навчальних матеріалів універсальною, тобто відкритою до створення навчальних матеріалів з будь-яких наук, дозволяє додавання будь-яких документів, що конвертуються в HTML або Flash. Це дозволяє розширити рамки застосування даного проекту за межі об'єктно-орієнтованого програмування.

Налаштування php-архіваторів

Навчальні матеріали завантажуються на сервер в основному в вигляді архівів. Тому в даному проекті було використано бібліотеку *pclzip*. [49] З її допомогою можна проводити багато операцій над архівами формату .zip: створювати, розархівовувати, вилучати, зливати в один, дублювати тощо

Реалізація інтерактивності навчання

В даній роботі застосоване динамічне оновлення та компілювання програмного коду в матеріалах з об'єктно-орієнтованого програмування. Динамічність полягає в змозі переглядати виконання проекту одразу після внесення змін в програмний код. Всі дії відбуваються на сервері, через веб-інтерфейс браузера. Також для проектів, що потребують введення даних в програму вже після процесу компіляції і лінування, передбачена можливість завантажити виконавчий файл на комп'ютер і вже на ньому виконувати програму. Також при помилках в програмному коді автор може бачити які повідомлення про помилки та попередження генерує компілятор GCC на сервері.

Мульти-користувацький інтерфейс

Кожен викладач має повний контроль над своїми матеріалами, і ніяк не може доступитись до матеріалів інших авторів. Ці налаштування пов'язані з інтегруванням в систему авторизації(та аутентифікації) користувачів в системі управління контентом E107. На головній сторінці відображаються матеріали всіх авторів, однак можна вибрати конкретного автора і продивлятись тільки його матеріали. Адміністратор в свою чергу має доступ до матеріалів всіх викладачів і може контролювати наповнення. Але він не може редагувати матеріали, може тільки активувати/деактивувати або видаляти.

Налаштування роботи GCC (GNU Compilers Collection) під ОС FreeBSD

GNU Compiler Collection (скорочено GCC) — набір компіляторів для різних мов програмування, розроблений в рамках проекту GNU. [50] GCC є відкритим програмним забезпеченням, поширюється на умовах ліцензій GNU GPL і GNU LGPL. Він використовується як стандартний компілятор для відкритих Unix-подібних операційних систем, а також для Apple Mac OS X.

GCC був започаткований Річардом Столлменом [51], який реалізував перший варіант GCC в 1985 на нестандартному діалекті мови Паскаль; пізніше компілятор було переписано мовою Сі Леонардом Тауером і Р. Столлменом, і випущено в 1987 як компілятор для проекту GNU.

Сьогодні GCC підтримується групою програмістів із усього світу. GCC є лідером по кількості процесорів й операційних систем, які він підтримує.

Окрім того, що GCC є офіційним компілятором системи GNU, включаючи варіації системи, побудовані на ядрі Linux (GNU/Linux), GCC також є головним компілятором для ряду інших операційних систем.

ОСС часто вибирається для розробки програмного забезпечення, що повинно працювати на різних апаратних платформах. Розходження між «рідними» для кожної з апаратних платформ компіляторами приводять до труднощів при розробці коду, який би коректно компілювався різними компіляторами. При використанні GCC для компіляції коду під різні платформи використовується один і той же синтаксичний аналізатор. Тому якщо вдалося зібрати програму для однієї із цільових платформ, те велика ймовірність, що програма нормально збереться й для інших платформ.

Зовнішній інтерфейс GCC є стандартом для компіляторів на платформі Unix. Користувач викликає керуючу програму, що називається *gcc*. Вона інтерпретує аргументи командного рядка, визначає й запускає для кожного вхідного файлу свої компілятори потрібної мови, запускає, якщо необхідно, асемблер і компоувальник. [52]. Команда *G++* викликає компілятор для програм написаних на C++. При подальшій розробці проекту є можливість розробити універсальне середовище, що буде компілювати, злінковувати та генерувати виконавчі файли для таких мов як C, OBJECTIVE-C, Java, Fortran і Ada, і тому можна буде в повній мірі реалізувати роботу з набагато більшою кількістю курсів з об'єктно-орієнтованого програмування.

6.5 Особливості реалізації. Інтеграція з порталом Joint European Projects на основі стандарту CMS E107

e107 [20] - це набір скриптів, написаних на PHP, які взаємодіють з MySQL, що в цілому забезпечує відмінну готову портальну систему. З її допомогою можна створювати як величезні портальні системи, так і просто невеликі сайти. В нашому випадку най цій системі базується навчальний портал Joint European Projects (<http://jep.ukma.kiev.ua>), і оскільки E107 є модульною системою (орієнтована на роботу з функціональними розширеннями, модулями), розроблений інструмент по оперуванню навчальними матеріалами був реалізований в виді плагіну. Основні особливості CMS E107:

- високоефективне файлово-орієнтоване кешування
- інтегрована системи новин та RSS
- спрощений процес створення та використання шаблонів
- валідний XHTML код
- потужна система підтримки форумів
- зручний підхід до створення нових модулів ^ зручна система адміністрування

Спільна інтеграція з проектом підтримки автоматизованих практикумів та системою електронних портфоліо

В напрямі інтеграції зі студентськими практикумами був вкладений такий концепт, щоб викладач міг швидко переходити від перегляду свого робочого зошиту (система створення курсів і наповнення їх контентом) до перегляду робіт студентів, що наприклад виконують певну вибрану лабораторну роботу, і навпаки. Для того, щоб забезпечити таку функціональність, для зберігання даних в LCMS та

практикумах про лабораторні роботи, задані викладачем, використовується спільна таблиця *e107_coursecreator_program*. Практичні роботи, збережені в цій таблиці, відображаються як навчальні матеріали певного розділу певного навчального курсу. Студентські практикуми прив'язані до певних робіт, створених викладачем. Таким чином викладач при перегляді курсу, може переходити по посиланням і мати змогу бачити скільки студентів що зробили по конкретній лабораторній роботі. Аналогічно він може потім повернутись на свій робочий зошит, адже перебуваючи в робочому зошиті студента, ми знаємо до якої лабораторної роботи і якого курсу «прив'язаний» його практикум.

Також існує можливість подібної навігації між системою створення навчальних курсів та між системою електронних портфоліо. При перегляді курсів можна переходити по посиланню на особисте електронне портфоліо автора курсу, або переглянути всі курси даного автора. В свою чергу через інтерфейс *eFolio* можна бачити інформацію про матеріали конкретного викладача (автора курсу).

Висновки до розділу 6

В даному розділі детально розглянуто процес електронного навчання, визначені основні принципи, що поступово перетворюється e-learning в e-learning 2.0. Сьогодні ми так чи інакше стикаємося з електронною освітою, навіть не маючи географічних чи будь-яких інших обмежень.

В межах виконання роботи був розглянутий процес вивчення курсів з програмування в системі електронної освіти.

Результатом досліджень та пошуку рішень стала реалізація веб-інструменту, що дозволяє певною мірою розв'язати проблеми, що існують. Створена LCMS дозволяє:

- прискорити і якісно поліпшити процес навчання;
- зробити його максимально наочним;
- внести елемент інтерактивності при роботі з матеріалами з мов програмування;
- уникнути проблеми несумісності платформ;
- централізувати розрізнений навчальний матеріал;

Отже в контексті стрімкого розвитку веб-технологій це дозволяє наблизитись до якісно нового рівня представлення процесу навчання, причому як електронного, так і змішаного типу.

7 Архітектура та інформаційна модель програмної системи підтримки електронних портфоліо

Історія розробки електронного портфоліо (ЕП) та їх використання в освіті розпочинається на початку 90-тих. Найбільш важливий та цінний внесок був зроблений у 2002 році Глобальним консорціумом із навчання IMS (IMS Global Learning Consortium) розробкою специфікації портфоліо. Для підтримки імплементації ЕП Європейський інститут дистанційної освіти у 2005 році створив Консорціум Європортфоліо, метою якого є трансформація сучасних поглядів на освітні технології. Він об'єднує багато європейських університетів та організацій, а також проводить міжнародні конференції стосовно ЕП та європейські конференції із дистанційної освіти.

Наразі розроблено чимало систем та інструментів для створення та підтримки ЕП [53,54]. Вони успішно випробовуються і наразі використовуються у багатьох навчальних організаціях. Однак, ця царина дослідження все ще досить нова, а більшість таких систем розроблялася без урахування міжнародних стандартів та специфікацій.

При побудові програмної системи підтримки електронного портфоліо (ПСПЕП), яка забезпечувала б можливість співпраці з іншими програмними системами, що використовуються світовою спільнотою, доцільно скористатися виробленими стандартами (специфікаціями) для розумного зв'язування даних, задоволення вимог інтероперабельності та можливості ефективного застосування технологій та інструментаря розробки.

У цій праці ми проаналізуємо основні з поширених нині стандартів (специфікацій), запропонуємо архітектуру та інформаційну модель ПСПЕП.

7.1 Базові специфікації

На сьогодні у розробці електронного портфоліо (ЕП) використовуються і рекомендуються до використання такі стандарти [53]: IMS Learner Information Packaging, IMS e-Portfolio, HR-XML Résumé, X/HTML, RDF та інші.

Почнемо розгляд із Portfolio Package. Основні структурні частини портфоліо, відношення, презентації та представлення (PortfolioParts, Relationships, Presentations, Views), а також інші допоміжні ресурси із специфікації інформаційної моделі електронного портфоліо, наприклад, посилання *продуктів на медіафайли*, в портфоліо або набору портфоліо, становлять разом пакет портфоліо (Portfolio Package).

Формат пакету портфоліо дещо складний і вимагає досить глибокого знання кількох інших специфікацій, тому у майбутніх версіях його очікується спрощення [54]. Формат пакету у його поточній формі базується на специфікації IMS Content Package. Кожна

частина портфоліо, відношення, презентація чи представлення включена окремим файлом XML. Маніфест стилів контент-пакету описує всі включені компоненти. У маніфесті елементи організації та заголовку використовуються для позначення мети створення та типу кожного ресурсу через перерахування набору правил. Для кожного з компонентів можуть долучатися записи метаданих навчальних об'єктів (Learning Object Metadata, LOM). Ці записи метаданих можуть використовуватися для позначення важливості компонента для портфоліо. Наприклад, метадані можуть додаватися для позначення, що деяка презентація необхідна для правильного розуміння портфоліо, створеного спеціально для переконання аудиторії у тому, що автор прекрасний графічний дизайнер.

Ось приклад портфоліо мінімальної складності для експорту лише одного продукту “письмового завдання (есе)” запропонованого IMS ePortfolio Best Practice and Implementation Guide [55]. Директорія для цього пакету портфоліо (minimalPortfolio) містить такі файли: маніфест ('imsmanifest.xml'), файл сутності ідентифікації ('minimalIdentification.xml'), сутності продукту ('minimalProduct.xml'), робочий продукт ('Midterm_research_Project_hollandp_Arthur.doc').

Схематичне зображення отриманого таким чином пакету портфоліо [55] показано на рисунку 7.1.

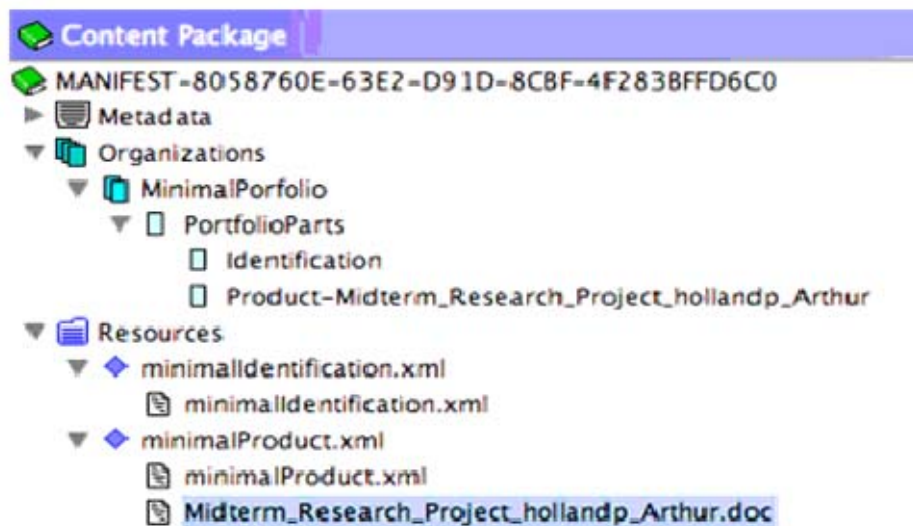


Рис.1. Графічне зображення прикладу портфоліо мінімальної складності

У 2002 році консорціум HR-XML (консорціум по створенню стандартів із обміну даними ресурсів) опублікував нову специфікацію резюме XML v. 2.0. Це резюме є XML-схемою і містить всі елементи притаманні резюме. Воно не підтримує висловлювання і планування безпосередньо, але більш корисне своєю експортною

функцією і рекомендується для забезпечення інтероперабельності [56].

X/HTML є найпростішою технологією і першочерговим стандартом для розробки ЕП. HTML легко сприймається людиною, але, на жаль, не EOM і використовується для створення веб-блогів та простих сторінок.

Формат опису ресурсів (Resource Description Format) часто розглядають як можливий спосіб представлення ЕП. RDF та споріднені технології *семантичної мережі* надають досить конкурентноспроможні засоби розміщення різноманітних робіт і опису складних відношень [57]. Існують кілька корисних словників RDF для електронних портфоліо (Friend of a Friend, FOAF, RSS 1.0). Частина експертів висловлюють думку, що інформаційна модель електронного портфоліо IMS може прив'язуватись до RDF, замість LIP.

Скотт Віллсон пропонує підхід до розроблення портфоліо [5], що складається з чотирьох етапів: визначення X/HTML, атомарний вивід, FOAF/vCard для ідентифікації; модуляризація стилів Atom/RSS та зв'язування частин для забезпечення інтероперабельності; введення елементів із IMS ePortfolio і додавання метаданих; побудова онтології для поглибленого зв'язування.

Даррен Кембрідж висуває думку, що із появою онтологій та семантичної мережі надається кращий фреймворк для обміну і повторного використання даних у мережі [58].

7.2 АРХІТЕКТУРА ЕЛЕКТРОННИХ ПОРТФОЛІО

Перед тим, як описати пропоновану нами архітектуру, наведемо два проекти для висвітлення двох базових суттєво різних підходів до побудови архітектури ЕП.

‘Generic’ ePortfolio, досвід Університету міста Ньюкасл.

“Родове” (‘generic’) електронне портфоліо розроблялося у Ньюкаслському університеті як частина спільного проекту FDTL4 [59]. Воно створене із використанням таких платформи-незалежних інструментів із відкритим кодом як Веб-сервер Apache, середовище публікації ZOPE та бази даних MySQL. Як стверджують автори цієї системи [60], на етапі дизайну розглядалися деякі з потенційних додаткових позитивних функцій, характерних для ІТ-підходу (на відміну від традиційних, паперових портфоліо): багато вибіркових опцій, підтримка великої кількості структур/представлень, полегшення взаємодії із керівниками та іншими користувачами (Sharable), полегшення створення взаємних посилань, можливість пошуку, можливість трансферу даних для підтримки навчання протягом всього життя, часткове/поглиблене адміністрування, підтримка

завантаження записів у великій кількості форматів, відновлювання та зменшення вимог до фізичного розташування.

Наскільки це можливо, ці функції були “вбудовані” в ePortfolio (рисунк 7.2).



Досвід проекту EIfEI **Рисунок 7.2 ‘Generic’ ePortfolio [60]**

Проект EIfEI (Європейський інститут дистанційного навчання) розпочався в 2001 році. Отже, можна сказати, що учасники проекту мають гарний досвід, який може бути цінним при виборі архітектури майбутньої системи електронних портфоліо.

Автори проекту розрізняють електронне портфоліо (eP) та систему управління електронними портфоліо (ePMS). Вони зазначають [61] - проблема створення сучасних систем портфоліо полягає у нерозумінні багатьма розробниками ЕП того, що функціональність ePMS виходить за межі *хостингу* електронних портфоліо. Насправді, має відбуватися цілком протилежне: основною функцією ePMS є не розміщення e-folio, а керування процесом, у якому використовується або створюється електронне портфоліо.

ePMS, структура якого зображена на рисунку 3 (взятого з праці [61]).

1. PDP - parallel distributed processing
2. SWOT – чотири складові котрі необхідно проаналізувати. Strengths — сильні сторони, Weakness — слабкі сторони, Opportunities — сприятливі можливості, Threats — потенційні джерела неприємностей.

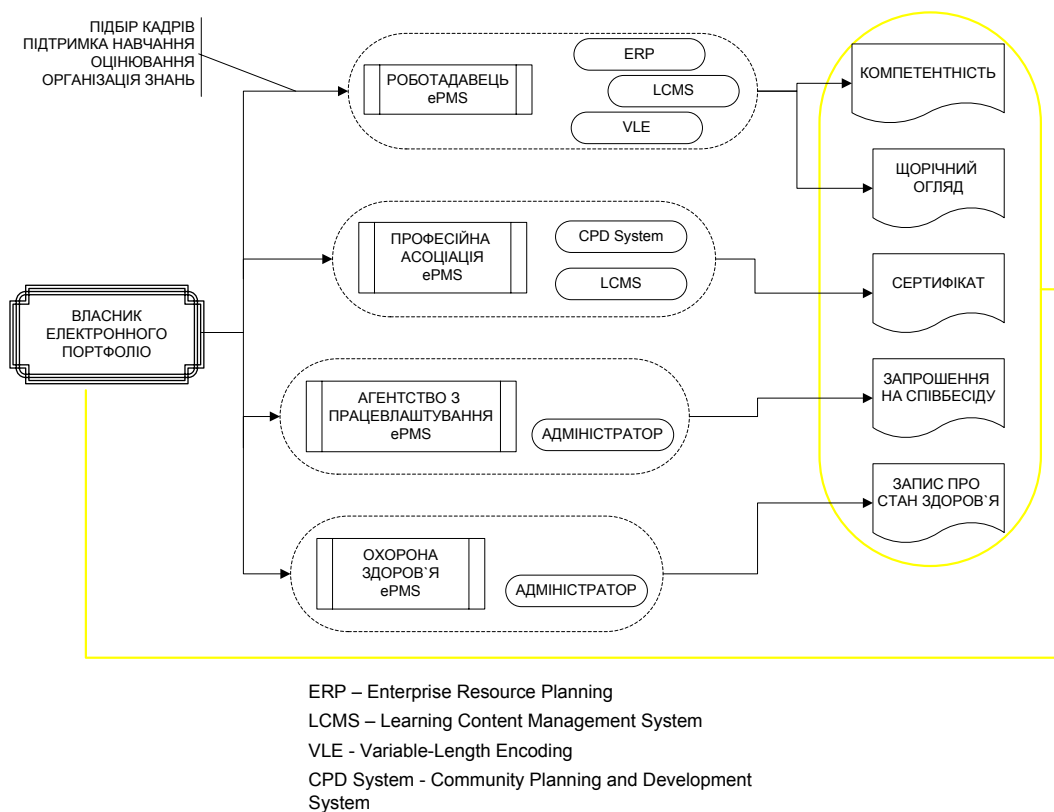


Рис.3. Система управління електронними портфоліо

Розглядається як інструмент, основною функцією якого є створення ЕП для конкретної цілі і в межах конкретного контексту. Ось визначення, надане Сержем Раве, одним із учасників проекту EIfEl [9]: “Система управління електронними портфоліо це система управління (створення, користування і експлуатації) елементами окремих портфоліо для конкретної мети - супроводження навчання, самостійних завдань, найму на роботу, керування кваліфікацією, організаційного навчання, керування знаннями тощо.”

Ось список видів ePMS, запропонований авторами системи: для найму на роботу, для планування особистого розвитку, для постійного професійного розвитку, для оцінювання попередньо вивченого чи засвоєного, для ознайомчої освіти, для керування кваліфікаціями та кар'єрою.

Очевидно, що ці ePMS сильно відрізняються за функціональністю і типами стандартів, необхідних для забезпечення інтероперабельності із середовищем застосування. Як правило, такі системи розробляються для задоволення потреб деякої організації. ePMS належить цій організації та відображає її інтереси, цінності та філософію.

Наступним елементом в архітектурі системи за EIfEl є організатор електронного портфоліо (ePO). На відміну від ePMS, ePO належать окремим особам. Вони надають

можливість створювати і управляти своїм цифровим профілем (digital identity). Органайзер ЕП можна розглядати як деяке “дзеркало”, яке забезпечує зворотній зв'язок із своїм власником або дає вихід на зовнішній світ у формі електронного портфоліо.

Як кажуть, одним із найбільших недоліків у сучасних технологіях органайзерів електронних портфоліо є те, що, незважаючи на можливість збирання та виставлення тегів на даних, немає інструменту надання зворотного зв'язку подібно “дзеркалу”. Тому необхідно розробити технології динамічного аналізу електронних портфоліо за допомогою добування даних (data mining) і просторового подання (spatial representation).

Очевидно, що запропонована у проекті EIFEl архітектура дуже складна і досить нелегка для імплементації у малих закладах, оскільки вона має за мету поєднання різних рівнів управління. Не зважаючи на те, що така архітектура існує тільки в цьому проекті, результати її реалізації можуть мати надзвичайне значення для дистанційної освіти.

Архітектури ПСПЕП

Наведемо власне бачення архітектури ПСПЕП.

Перш за все, наголосимо на значенні ЕП у електронному навчанні. ЕП зв'язує дві основні підсистеми програмної системи підтримки електронного навчання (рисунок 7.4) але не належить повністю жодній з них. Цими підсистемами є підсистема управління навчальним процесом LMS (Learning Management System,) і підсистема управління контентом CMS (Content Management System). Електронне портфоліо має елементи цих систем і може агрегувати деякі сервіси обох систем, але існує окремо від них.

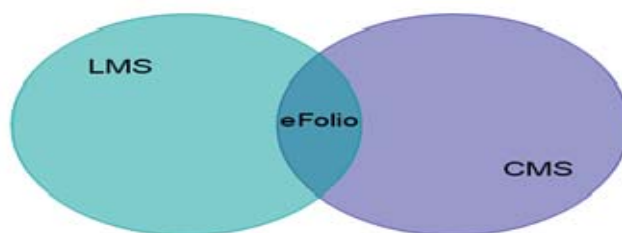


Рисунок 7.4. Місце ЕП в системі електронного навчання

Рисунок 7.5 відображає найпростішу архітектуру системи управління електронними портфоліо, яка буде ефективною для розробки. Ми прагнули долучити до цієї архітектури всі елементи, необхідні користувачеві для створення, підтримки та розробки власного портфоліо.

Усі елементи електронного портфоліо включені до однієї системи керування, імплементація якої в систему електронного навчання полегшить користувачам роботу зі своїми портфоліо. Наша система, перш за все, безпосередньо надає доступ до ЕП і забезпечує комунікацію між користувачем (власником) і всією системою. За допомогою інтерфейсу ЕП користувач отримує можливість роботи із портфоліо, створюючи нові елементи, пишучи посилання тощо.

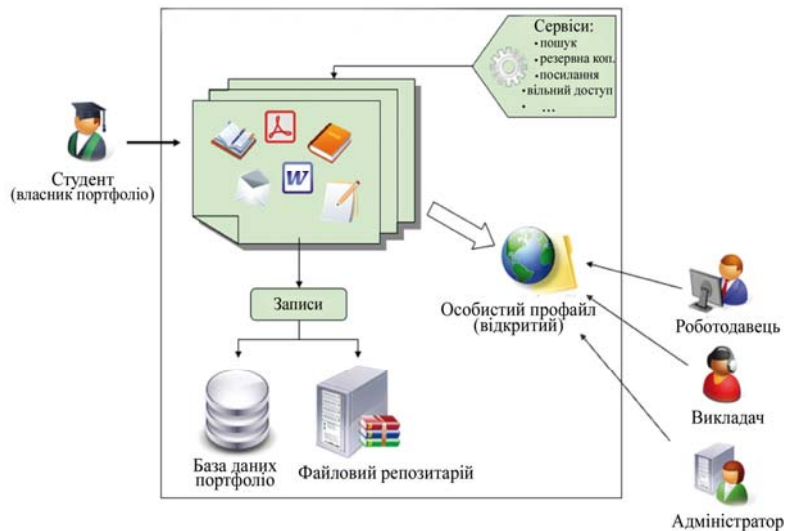


Рисунок 7.5. Архітектура системи управління електронними портфоліо

Всі записи, створені у системі, зберігаються або у базі даних портфоліо (існує для електронних повідомлень, конфігурації, зберігання посилань) або у файловому репозитарії (існує для різних долучень до елементів портфоліо, шматків робіт користувача, і, як зазначено IMS, продуктами, які становлять все завантажене коли-небудь до портфоліо).

Головними рисами електронного портфоліо є сервіси. Як зазначено вище, у сполученні вони можуть формувати або систему LMS, або CMS, або розроблятися незалежно. Сервіси полегшують керування електронними портфоліо, здійснюючи такі речі, як пошук у портфоліо або у всій системі; збереження портфоліо для подальшого його відновлення у іншій системі або простого збереження безпечної копії портфоліо; необхідний для співпраці та комунікації обмін інформацією, а також багато інших сервісів.

Оскільки кожен користувач може ділитися інформацією з особистого профіля, створюється так звана публічна частина електронного портфоліо. Цю частину видно усім зареєстрованим користувачам системи, або, якщо дозволяє портал, усім, хто має право переглядати опубліковану частину ЕП. Користувачу надзвичайно важливо можливість демонстрації портфоліо перед потенційними роботодавцями, лекторами для рекомендацій, спільнотою системи тощо.

Описана система досить проста і містить всі необхідні елементи для ефективного навчання, роботи або особистого розвитку. Гадаємо, що така архітектура може бути легко поєднана із іншою ПСПЕН і у деяких аспектах може використовуватися як система

управління контентом.

7.3 ІНФОРМАЦІЙНА МОДЕЛЬ ОРГАНІЗАЦІЇ ДАНИХ

Вибір архітектури інформаційної системи вже зроблено. Залишилося описати організацію даних у системі. Для визначення інформаційної моделі електронного портфоліо, використаємо специфікацію інформаційної моделі (IMS) Глобального консорціуму із навчання (IMS Global Learning Consortium). Ця специфікація ще не повністю протестована та імплементована на практиці, але такий вибір базується на повній відсутності стандартизації у галузі e-folio. Специфікація інформаційної моделі електронного портфоліо IMS саме й спрямована на створення таких стандартів. Певні частини цієї специфікації, а також опис найкращих випадків використання та вказівок щодо імплементації електронних портфоліо в IMS вже ввійшли до складу освітнього стандарту SCORM. Ще одним приводом застосування специфікації у нашій розробці була потреба у забезпеченні інтероперабельності ПСПЕП з іншими системами, що підтримують електронні портфоліо.

Розробники специфікації інформаційної моделі (IMS) пропонують наступну організацію даних у пакеті e-folio. Насамперед, він має містити файл маніфестації 'manifest.xml'. Файл маніфестації для пакету електронного портфоліо – це XML-репрезентація всього портфоліо, де кожен елемент портфоліо постачається як ресурс із пакету контенту.

Відповідно до специфікації IMS, архів електронних портфоліо або загальне портфоліо може містити на додачу таку інформацію [62]: суб'єкт ePortfolio; діяльність, у якій брав, бере чи планує участь суб'єкт; компетенція (навички тощо) суб'єкта; досягнення і успіхи суб'єкта, незалежно від факту їх сертифікації; переваги суб'єкта; цілі та плани суб'єкта; інтереси та цінності суб'єкта; нотатки, висловлювання або самостійні завдання стосовно будь-якої іншої частини системи; результати всіх тестів чи екзаменувань суб'єкта;

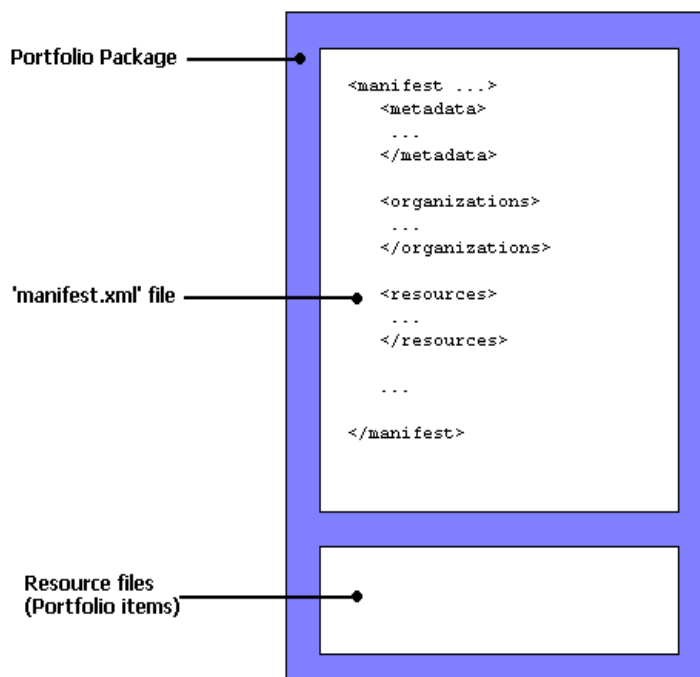


Рисунок 7.6. Пакет портфоліо

контекстна інформація для коректної інтерпретації будь-яких результатів; дані про створення і авторів частин електронного портфоліо.

Рисунок 7.6 (взятий з роботи [62]) відображає структуру пакету ЕП.

Інформаційна модель визначає необхідні компоненти портфоліо. Портфоліо містить п'ять таких компонентів. По-перше, воно містить зібрання гетерогенних *Частин* (*Parts* або як їх ще називають, *Елементів – Items* [63]), які використовуються для створення цінних якостей портфоліо. Другий компонент асоціюється із *Власником* (*Owner*), автором портфоліо. Далі, портфоліо містить набір *Відношень* (*Relationships*) між *частинами*. Ці відношення визначають, яким чином елементи портфоліо синтезуються в одне ціле і як вони можуть інтегруватися системами, що підтримують використання та розробку електронних портфоліо. Четвертою компонентою є *Презентації* (*Presentations*), інструкції щодо того, яким чином і в якій послідовності цільова аудиторія має отримувати враження від портфоліо, уточнення щодо гіпертекстової структури і візуального дизайну. Накінець, для полегшення перенесення даних, портфоліо також може містити *Представлення* (*Views*), які є по суті вибірками елементів, відношень та презентацій для конкретної цільової аудиторії або досягнення певної мети.

Всі ці компоненти поєднуються у пакет. Традиційно [64] ЕП має структуру представлену на рисунку 7.7

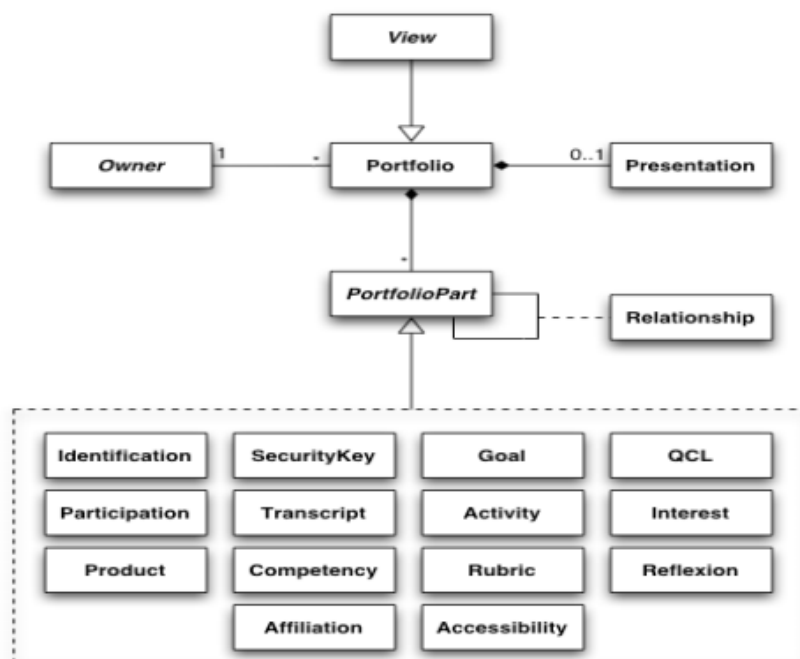


Рисунок 7.7. Опис структури портфоліо.

7.4 рекомендації щодо розробки та імплементації

Рисунок 7.8, взятий з роботи [65] змальовує організаційні етапи розробки електронного портфоліо у деталях. Із просуванням по етапах, фокус уваги зсувається від потреб слухача до проблем імплементації для закладу чи галузі.

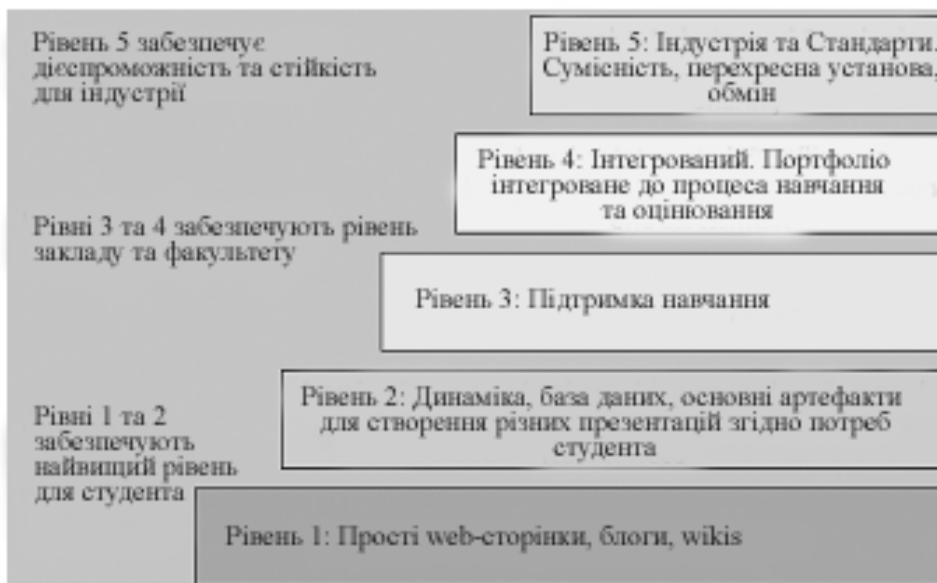


Рисунок 7.8. Етапи розвитку електронного портфоліо

Рівень 1 може містити відносно прості вебсайти та вбудовані блоги або вікіпедії, включно із обмеженою навігацією по контенту. Рівень 2 складається із динамічних веб-сторінок. Доступні й функції навігації та пошуку. Рівень 3 вимагає підтримки електронних портфоліо закладом, включно із рекомендаціями використання. Заклад може надавати й хостинг програмній розробці. Рівень 4 потребує інтеграції розробки і використання портфоліо у навчальному процесі закладу. Рівень 5 визначає вимоги до підтримання стандартів які забезпечать інтероперабельність портфоліо.

Висновки до розділу 7

Викладення матеріалу велось таким чином, щоб на конкретних прикладах описати весь процес розробки ПСПЕП. Особлива увага приділена побудові архітектури ПСПЕП та інформаційної моделі.

За належної розробки, імплементації та використання ЕП можуть стати суттєвим інструментом у навчанні. Вони уможливлюють креативне і рефлексивне мислення, посилюють мотивацію у процесі навчання. Все це створює нові виклики для розробників і дизайнерів. Ось чому впродовж кількох останніх років, розглянуто і прийнято значну кількість специфікацій та рекомендацій стосовно імплементації ЕП, які проаналізовано в цій праці.

8 Математична модель взаємодій в програмній системі підтримки електронного навчання

8.1 Основні підходи до побудови моделей взаємодій

Розвиток мережі Інтернет практично розмив кордони і зробив можливим безпосереднє спілкування і співпрацю людей з різних частин світу. Електронна пошта набагато пришвидшила обмін діловою та особистою кореспонденцією, а автоматичні листи розсилки, групи новин та форуми ще більш полегшили спілкування та зробили його інтерактивним. Але для вирішення практичних задач таких засобів часто виявляється недостатньо, через що розробляються як комерційні засоби забезпечення спільної роботи над документами та керівництва проектами (Lotus Notes, Microsoft Project, BSCW, платна версія Open-Xchange Server), так і відкриті протоколи на зразок WebDAV та його розширення GroupDAV для роботи із серверами групових проектів із відкритим кодом. Втім, всі вони мають свої недоліки, як то ціна у випадку комерційних платформ чи недостатня підтримка незамінних при спільній роботі над груповими проектами функцій, таких як кросплатформенність та система керування версіями.

З огляду на це зрозуміло, чому відносно новий метод співробітництва в мережі, реалізований в системі Wiki, швидко набув популярності. За визначенням його автора, програміста Уорда Канінгема (Ward Cunningham), “Wiki Wiki” (в перекладі з гавайської — „швидко”), це „неформальний набір веб-сторінок, які будь хто може віддалено редагувати за допомогою будь-якого веб-броузера”. Ці сторінки зберігаються в базі даних, і керування ними здійснюється за допомогою CGI (common gateway interface) скриптів мови Perl. Система підтримує просту і оригінальну систему гіперзв'язків між сторінками, які, до того ж, проіндексовані, що значно пришвидшує і полегшує пошук.

На разі існують багато її реалізацій (найвідомішою з яких є Wikipedia) та клонів-розширень, що містять спеціалізовані функції для полегшення завдань, специфічних для окремих галузей. Один з таких клонів, TWiki, розроблено спеціально для використання у корпоративних середовищах.

Важливим напрямком застосування колаборативних систем, який активно розвивається, є системи дистанційної освіти. У праці [66] Кіншук та інші розглядають проблему концептуального моделювання системи дистанційної освіти та пропонують власний підхід до її розв'язання, що полягає у використанні діаграм станів. При цьому моделювання відбувається в два етапи. Спочатку моделюється стратегія навчання (teaching strategy) як послідовність станів студентів (beginner, intermediate, advanced) та сукупність правил, що регламентують перехід студентів з одного стану в інший

(складання відповідних іспитів, досягнення певного рейтингу, тощо). Після цього отримана модель перетворюється на модель підтримки навчального веб-сайту. Кінцевим результатом роботи є побудова на базі моделі стратегії навчання концептуальної моделі системи дистанційної освіти, яка використовує відповідну стратегію навчання. Також розписано механізми інтеграції в систему навчальних курсів. Втім, в роботі не розглядаються власне механізми співробітництва між групою студентів та викладачем, моделювання відбувається на рівні зв'язків в предметній області, а не взаємодії учасників процесу чи складових частин програмного комплексу.

Також заслуговують на увагу роботи Кіншука та Патела [67] та Кіншука та інших [68]. В першій пропонується концептуальна модель інтелектуальних навчальних систем (intelligent tutoring systems), що функціонують в мережі Інтернет. Така система складається з множини інтелектуальних навчальних аплетів для окремих предметів, створюваних незалежно один від одного різними вчителями. Ці аплети можна під'єднувати та від'єднувати від системи за бажанням, формуючи таким чином конкретну навчальну програму з тієї чи іншої спеціальності. В роботі [69] розглядається модель синхронного навчання через мережу Інтернет за допомогою прямого (відеоконференція) чи опосередкованого (Video-On-Demand) спілкування. Проте увага приділяється радше педагогічному аспекту такої співпраці, аніж технічному.

Майєрс та інші [70] та Доммель та Гарсія-Луна-Ачевес [71] розглядають фундаментальний протокол взаємодії учасників колаборативного середовища — рівневий контроль (floor control). Питання організації рівневого контролю автоматично виникають тоді, коли потрібно організувати доступ кількох користувачів до спільного ресурсу. Перша стаття цікава насамперед запропонованою авторами класифікацією можливих політик рівневого контролю та спробою експериментальним шляхом визначити найефективнішу з них в рамках проекту PEBBLES (вивчення можливостей одночасної взаємодії кількох КПК з одним ПК). В свою чергу, Доммель та Гарсія-Луна-Ачевес детально розглядають питання мультимедійної взаємодії учасників групової взаємодії, роблять всесторонній огляд рівневого контролю з точки зору як системи, так і користувача. Крім цього, автори співставляють рівневий контроль з іншими парадигмами керування, насамперед сеансовим (session), паралельним (concurrency) та контролем доступу (access control).

Пападопулос та Арбаб [72] розглядають два різновиди координаційних моделей та мов: базовані на даних (data-driven) та на керуванні (control-driven, also task- or process-oriented). В моделях та мовах, базованих на даних, стан обчислення в будь-який момент часу визначається в термінах як значень даних, що пересилаються або одержуються, так і

конфігурації складових, дії яких координуються. Тобто, координатор відповідає не лише за координацію, але й за роботу з даними. Тому на синтаксичному рівні процеси в системі не можна чітко розділити на обчислювачі та координатори, і цю задачу має вирішувати програміст. Натомість у моделях та мовах, базованих на керуванні, координація та обчислення практично повністю відокремлені одне від одного, і конкретні значення оброблюваних даних майже ніколи не грають ролі. Мови та моделі першого типу розвинулися навколо концепції спільного простору даних (Shared Dataspace), хоча в деяких пізніших варіантах з метою посилення безпеки застосовано механізми обміну повідомленнями чи або спільний простір даних, обмежений одним буфером або глобальною синхронізаційною змінною. В основі другого підходу лежить принцип Оккама про набір окремих сутностей, що взаємодіють з навколишнім світом за допомогою чітко окреслених інтерфейсів. До першого типу автори відносять такі координаційні мови та моделі як Linda та похідні від неї (Bonita, LAURA, Ariadne/HOPLa, Sonia), а також GAMMA, LO, COOLL та інші. Другий тип представлено такими мовами як PCL (Proteus Configuration Language), Conic, Darwin, Durra, CSDL, POLYLITH та інші.

Цікавий підхід до моделювання взаємодії користувачів в мережі Інтернет пропонують Цю та інші в роботі [73]. Вони розглядають Інтернет-середовища співпраці та пропонують досить потужний загальний підхід до їх моделювання з використанням парадигми MVC (Model View Controller paradigm). DOM-застосування (DOM — Document Object Model), що функціонують в мережі Інтернет, за допомогою MVC представляються у вигляді веб-сервісів, що співпрацюють один з одним через систему колаборативних подій [74].

В роботі [71] Доммель та Гарсія-Луна-Ачевес представляють свій погляд на структуру колаборативного середовища в комп'ютерній мережі. Колаборативне середовище вони подають як набір $\Gamma = \langle S, U, R, F \rangle$, де S — множина сеансів, U — множина користувачів, R — набір спільних ресурсів, F — множина рівнів, що керують ресурсами. Кожна з цих сутностей має свій набір атрибутів. Розглянемо кожен з них детально.

Сеанс (session) — це набір $\Sigma = \langle \text{Sid}, T_i, T_e, A_s, L \rangle$ де Sid — унікальний ідентифікатор сеансу в межах середовища Γ , T_i — час ініціалізації сеансу, T_e — час завершення, A_s — список атрибутів, що характеризують даний сеанс на поверсі (level) L . Сеанс надає інфраструктуру для взаємодії та співпраці користувачів. Набір атрибутів A_s описує призначення та перебіг роботи сеансу в термінах **членства** (membership), **організації** (organization) та **контролю** (control). Членство визначає принципи формування групи користувачів сеансу (розмір групи, ступінь інтерактивності, чи потрібно

користувачам вказувати своє ім'я, чи відкритий до неї доступ тощо). Організація — як саме працює сеанс (час життя сеансу, чи однорідні дані, чи є групи користувачів ієрархічними тощо). Контроль відповідає за стан сеансу, принцип керування сеансом (централізований/розподілений), модерування сеансу та рівень безпеки сеансу (чи шифруються повідомлення) .

Користувач (user) — набір $U = \langle \text{Uid}, \text{Sid}, \text{Loc}, T_j, T_l, A_U \rangle$, де Uid — унікальний ідентифікатор в межах сеансу Sid, Loc — локальне чи віддалене місцезнаходження (IP-адреса чи номер хоста), T_j — час приєднання, T_l — час від'єднання, A_U — список атрибутів. Атрибути визначають роль користувача (простий користувач, адміністратор; слухач, спікер). особу користувача, чи може він змінювати дані тощо [10]. Користувачами, в залежності від контексту, можуть бути не лише люди, що працюють в системі, а й прикладні програми чи засоби самої системи.

Ресурс (resource) — набір $R = \langle \text{Rid}, \text{Sid}, \text{Pid}, \text{Uid}, T_c, T_d, A_R \rangle$, де Rid — унікальний ідентифікатор ресурсу, яким розпоряджається користувач Uid в межах сеансу Sid. Pid — ідентифікатор батьківського для Rid ресурсу, T_c — час створення, T_d — час видалення, A_R — список атрибутів. Ресурс може бути як дискретний, так і неперервний, тип визначається відповідним атрибутом. Також за атрибутами визначається доступність ресурсу широкому загалу, пріоритетність передачі відповідної інформації каналами зв'язку, QoS, ступінь захищеності ресурсу [V.3, С. 4]. Ресурсом в залежності від контексту може бути що завгодно, від on-line конференції чи прикладної програми до окремого вікна чи спільної змінної.

Рівень (floor) — набір $F = \langle \text{Fid}, \text{Rid}, \text{Uid}, T_i, T_d, A_F \rangle$, де Fid — унікальний ідентифікатор в межах спільного простору роботи з ресурсом Rid рівня, наданого користувачеві Uid в час T_i і деактивованого в час T_d , з набором атрибутів A_F . При цьому одному ресурсу Rid можуть відповідати кілька Fid для керування різними його фрагментами, але кожен рівень керує виключно одним ресурсом. Рівні є складовими фундаментального протоколу взаємодії учасників колаборативного середовища — рівневого контролю (floor control), детально описаного в працях [V.1; V.4; V.8]. Володіти рівнем — означає мати привілейований доступ до відповідного ресурсу. Рівні розрізняються за **напрямлєністю** (відправник/одержувач), **станом** (вільний, зайнятий, простоє, заморожений, пошкоджений, обробляє запит, відновлений), **дубльованістю** (скільки екземплярів рівня може одночасно існувати в системі), **прозорістю** (чи потрібно кінцевим користувачам вручну звільняти/займати рівень), **політикою** (стосовно запитів та часу перебування користувачів в системі), **модальністю** (головний/запасний канал),

стратегією (оптимістичний/песимістичний підходи до організації роботи рівня) [V.3, С. 5—610].

Контролери кожного з цих елементів сукупно утворюють координаційний механізм, що забезпечує комфортну роботу користувачам колаборативної системи.

8.2 Модель взаємодій на основі мереж Петрі

Формальна модель колаборативного середовища повинна ґрунтуватися на детальній структурі формальної специфікації та перевірки властивостей колаборативних систем. При побудові нашої моделі було взято за основу структуру, запропоновану Доммелем та Гарсія-Луна-Ачевесом. Колаборативне середовище вони подають як набір $\Gamma = \langle S, U, R, F \rangle$, де S — множина сеансів, U — множина користувачів, R — набір спільних ресурсів, F — множина рівнів (floor), що керують ресурсами.

До складу колаборативної системи входять N користувачів, M сеансів, L ресурсів та координаційний механізм — сукупність позицій та переходів мережі Петрі, що зв'язує користувачів, сеанси та ресурси [75].

Користувачі в системі репрезентуються своїми **профілями**. Під профілем користувача розуміється його „особова справа”, що зберігається на головному сервері. Профіль визначає права користувача в межах даної мережі та слугує джерелом інформації про поточні дії користувача.

Поки користувач не увійшов у систему, профіль перебуває у стані „поза системою”. Увійшовши в систему, користувач приєднується до вже існуючої сесії або, якщо має для цього повноваження, створює свою (якщо брати систему дистанційної освіти, то розпочати семінар, наприклад, може лише викладач). Потім користувач працює в мережі (цю „роботу” слід розуміти в найширшому сенсі, як довільну послідовність передбачених дій). Якщо користувач робить щось недозволене, система може примусово від'єднати його.

Контролер сеансу регламентує створення сеансу та користування ним, забезпечує стабільність роботи, уможливорює доступ користувачів до потрібних їм ресурсів. Він також несе відповідальність за коректне завершення сеансу та звільнення зайнятих ресурсів.

Початковим станом контролера є стан бездіяльності (Idle). Коли надходить запит на встановлення сеансу зв'язку, контролер опрацьовує його і ухвалює рішення. В разі, якщо у користувача немає права на таке з'єднання, або задані ним параметри не

відповідають можливостям системи, контролер сеансу відмовляє йому. В іншому випадку сеанс створюється.

У разі виникнення помилки часу виконання контролер сеансу робить все можливе для того, щоб мінімізувати її вплив на роботу користувачів. В ідеалі користувачі взагалі не мають знати, що протягом певного часу щось було не так. Це стосується таких випадків як відмова запиту про виділення певного ресурсу чи аварійне завершення роботи одного з користувачів.

Якщо ж помилка занадто серйозна, контролер розпочинає процедуру коректного завершення сеансу, в рамках якої звільняються всі виділені ресурси, коректно від'єднуються всі користувачі, прибирається „сміття”. Якщо і на цьому етапі виникає невіpravна позаштатна ситуація, контролер вимикає сеанс та повідомляє про помилку всім задіяним сторонам (профілям користувачів, контролерам ресурсів), щоб вони зробили „прибирання” в себе. Після цього він повертається до початкового стану.

Контролер ресурсу відповідає за надання ресурсу за запитом та за коректне його вилучення (в разі аварійного завершення роботи системи).

Життєвий цикл контролера ресурсу починається зі створенням ресурсу. Після цього контролер переходить в пасивний стан очікування на запити на використання ресурсу. Коли ресурс виділено, контролер слідкує за його використанням, і в разі надходження команди про знищення ресурсу (наприклад, закриття доступу до спільно використовуваної папки) або якоїсь помилки (сервер, на якому ця папка знаходилася, впав) повертається у пасивний стан. Якщо ресурс передбачає можливість паралельного використання кількома користувачами, контролер бере на себе функції забезпечення взаємовиключення та синхронізації використання ресурсу з метою уникнення взаємного блокування [75]. Ще одним завданням контролера є прибирання „сміття”: коректне звільнення пам'яті, каналів зв'язку тощо.

Протокол рівневого контролю домагається доступу до спільно використовуваних об'єктів, надаючи доступ до рівнів відповідно до політики, визначеної для групи обслуговування. Фактично зайняття рівня рівносильне одержанню права говорити [76]. Саме через **контролер рівня** відбуваються практично всі дії з ресурсом (окрім створення, яке робиться безпосередньо).

Можливі стани рівня такі. *Вільний (Free)* — доступний, невикористовуваний рівень. *Обробка (Proc)* — обробка запиту, що надійшов (виділення ресурсу, заволодіння рівнем тощо). *Очікування (Waiting)* — очікування на повідомлення про виділення ресурсу. *Зайнятий (In use)* — вже наданий і зайнятий рівень. *Пауза (Paused)* — тимчасове припинення роботи з ресурсом без звільнення рівня (тривалість такої перерви

визначається правами користувача). В разі помилки в роботі системи контролер рівня відповідає за коректне закриття підпорядкованого йому ресурсу. Однією з найважливіших проблем при побудові моделі контролера рівня є забезпечення його коректної та оперативної реакції на падіння системи, яке може статися в будь-який момент, і відповідно контролер має реагувати на це незалежно від стану, в якому перебуває в даний момент.

В загальному випадку мережа Петрі, яка моделює роботу координаційного механізму колаборативної системи, до складу якої входять N користувачів (позначимо їх U), M сеансів (S) та L рівнів (F), складатиметься з:

- $3 * N$ позицій та $4 * N$ переходів, що описують поведінку користувачів;
- $N * L$ переходів типу „ U_i хоче створити ресурс R_k ” ($i \in 1 \dots N$; $k \in 1 \dots L$);
- $4 * M$ позицій та M переходів, що описують роботу контролера сеансу;
- $9 * L$ позицій, що відповідають контролерам рівнів;
- $3 * L$ позицій та $3 * L$ переходів, що описують роботу контролера рівня та ресурсу, що йому відповідає;
- $3 * N * M$ позицій та $4 * N * M$ переходів, що регламентують взаємовиключне створення сеансів;
- $5 * N * L$ позицій та $10 * N * L$ переходів, що відповідають зв'язкам між контролерами рівнів та користувачами.

У випадку із 2 користувачами, 1 рівнем і 1 ресурсом матимемо мережу, що складатиметься з 38 позицій та 42 переходів. Було доведено, що ця мережа обмежена та активна.

8.3 Універсальність моделі

В розділі продовжується розгляд задачі побудови високопродуктивної колаборативної системи. В попередній роботі [75] було проаналізовано моделі часткового випадку такої системи, побудовані мовою мереж Петрі. Доведемо, що ця модель адекватно відображає роботу колаборативної системи довільної конфігурації.

В праці [77] розглядалися та аналізувалися дві моделі колаборативної системи. Модель $M1$, що відповідала колаборативній системі з одним користувачем, одним сеансом, одним рівнем та одним ресурсом, та модель $M2$, в якій діяли два користувачі. Результати проведених досліджень показали, що обидві мережні моделі обмежені, тобто

кількість фішок мережі Петрі не перевищує певного максимального значення, та активні, бо немає тупикових ситуацій. Постає питання, чи зможе модель, побудована за цими принципами, зберегти ці властивості, якщо в системі буде як завгодно багато користувачів, сеансів та/або ресурсів.

Доведемо, що це так. Спершу покажемо, що побудований згідно з нашою моделлю КМ здатен забезпечити взаємовиключний доступ до єдиного спільного ресурсу та взаємовиключний контроль єдиного спільного сеансу для довільної кількості користувачів.

Теорема 1.3.1. Координаційний механізм (далі КМ) вирішує задачу координації для колаборативної системи з довільною кількістю користувачів, одним сеансом та одним ресурсом, доступ до якого регламентується одним рівнем (далі ЗК1).

Доведення. Для доведення використаємо метод математичної індукції.

База індукції: КМ вирішує ЗК1 для моделі М1. Це справді так, бо відповідна мережа Петрі обмежена та активна, що було доведено в праці [77].

Крок індукції: Нехай M_n — модель, що містить n користувачів, один сеанс та один ресурс, доступ до якого регламентується одним рівнем. Нехай КМ вирішує ЗК1 для M_n . Тоді, за твердженням індукції, КМ має вирішувати ЗК1 для M_{n+1} , іншими словами, додавання одного користувача не повинно призводити до недієздатності КМ. Так воно і є, як свідчить приклад моделі М2, яка є розширенням моделі М1 з додаванням ще одного користувача, і яка зберігає всі властивості моделі М1. Доведено.

Тепер розповсюдимо отриманий результат на колаборативну систему з довільною кількістю користувачів, сеансів, рівнів та ресурсів.

Теорема 1.3.2. КМ вирішує задачу координації для колаборативної системи з довільною кількістю користувачів, сеансів та ресурсів, доступ до яких регламентується своїм рівнем для кожного з них (далі ЗК $_n$).

Доведення. Як зазначалося в праці [12], всі варіанти, коли кількість користувачів, сеансів та рівнів різна ($N = n, M = m, L = l; (n \neq m) \vee (l \neq n) \vee (m \neq l)$), можна окремо не розглядати, оскільки вони зводяться до випадку, коли $N = M = L = \min(n, m, l) = x$ (при $x \geq 2$).

Додавання додаткових сеансів не призведе до недієздатності КМ, оскільки, в найгіршому випадку, коли всі користувачі змагатимуться за один сеанс, КМ працює згідно з теоремою 1.3.1, а змагання окремих груп користувачів за різні сеанси зводиться до

змагання меншої кількості користувачів за один сеанс, і в цьому разі КМ також працездатний за індукцією. Аналогічно виглядають справи зі змаганням за рівні. Доведено.

Як бачимо, наша модель здатна коректно відобразити процес роботи як завгодно складної та багатoelementної колаборативної системи. Керуючись цією інформацією, розглянемо процес побудови програмного агента, здатного моделювати роботу КМ нашої системи в як завгодно складному середовищі.

8.4 Складність задачі верифікації координаційного механізму

Метою даного розділу буде вирішення питань, пов'язаних із розв'язанням задачі верифікації координаційного механізму (далі КМ) для колаборативної системи, побудованої в рамках проекту зі створення моделі системи програмної підтримки колаборативного середовища мовою мереж Петрі, розпочату в праці [77]. Запропонована в попередніх роботах [75] мережна модель колаборативної системи базується на структурі колаборативного середовища, складовими елементами якої є сеанси, користувачі, спільні ресурси та рівні, за допомогою яких реалізується рівневий протокол доступу до ресурсів. Схематично структуру мережної моделі системи можна зобразити таким чином (рисунок 8.1):

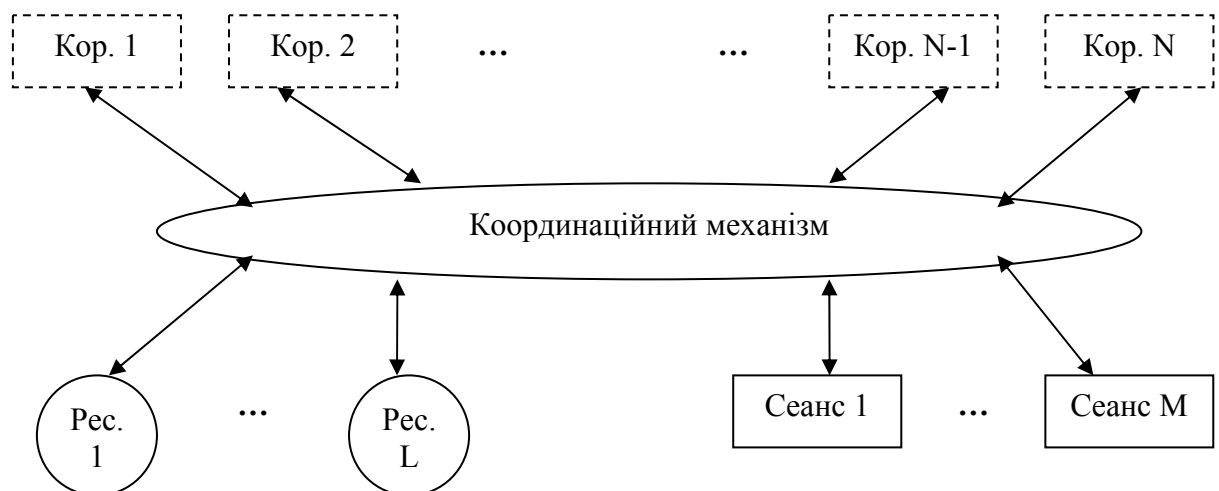


Рисунок 8.1. Структура мережної моделі колаборативного середовища

Як бачимо, до складу колаборативної системи входять N користувачів, M сеансів, L ресурсів та координаційний механізм — сукупність позицій та переходів мережі Петрі, що зв'язує користувачів, сеанси та ресурси. В нашій моделі це блок, що включає в себе контролери рівнів (кожному рівню відповідає один ресурс) та механізм забезпечення взаємовиключення при створенні сеансу. Складається він з таких позицій мережної моделі:

- а) Механізм забезпечення взаємовиключного створення сеансу для кожного з користувачів та для кожного сеансу. Це позиції «Користувач U_i не є керівником сеансу S_j », «Користувач U_i є керівником сеансу S_j », «Користувач U_i запитує дозвіл на створення сеансу S_j » та переходи, «користувач U_i хоче розпочати сеанс S_j », «користувач U_i хоче завершити сеанс S_j », «користувачу U_i дозволено створити сеанс S_j », «користувачу U_i не дозволено створити сеанс S_j ».
- б) Контролер рівня. Позиції: «Рівень F_k вільний», «Рівень F_k обробляє запит», «Рівень F_k в стані очікування», «Надати ресурс R_k », «Ресурс R_k надано», «Звільнити ресурс R_k », «Ресурс R_k звільнено», «Видалити ресурс R_k », «Ресурс R_k видалено».
- в) Механізм забезпечення взаємовиключного доступу кожного з користувачів до кожного з рівнів. Позиції: «Користувач U_i не є утримувачем рівня F_k », «Користувач U_i запитує доступ до рівня F_k », «Користувач U_i є утримувачем рівня F_k », «Рівень F_k активно використовується користувачем X », «Рівень F_k пасивно використовується користувачем U_i ». Переходи: «користувач U_i хоче отримати доступ до рівня F_k », «користувачеві U_i відмовлено у доступі до рівня F_k », «користувачеві U_i дозволено доступ до рівня F_k », «рівень F_k зайнято користувачем U_i », «користувач U_i призупиняє використання рівня F_k », «користувач U_i поновлює використання рівня F_k », «користувач U_i хоче видалити ресурс R_k », «ресурс R_k видалено користувачем U_i », «користувач U_i хоче звільнити рівень F_k », «користувач U_i звільнив рівень F_k ».

При цьому $i \in 1 \dots N$; $j \in 1 \dots M$; $k \in 1 \dots L$.

Отже, в загальному випадку мережа Петрі, що моделює роботу координаційного механізму нашої колаборативної системи, складатиметься з:

- $3 * N * M$ позицій та $4 * N * M$ переходів, що регламентують взаємовиключне створення сеансів;

- $9 * L$ позицій, що відповідають контролерам рівнів;
- $5 * N * L$ позицій та $10 * N * L$ переходів, що відповідають зв'язкам між контролерами рівнів та користувачами.

На неформальному рівні задача верифікації КМ полягає у визначенні придатності даного КМ до використання в колаборативній системі, тобто, іншими словами, перевірці відповідності КМ певним специфікаціям, що регламентують принципи та результати роботи системи. Перед тим, як виписати формальне визначення цієї задачі, дамо наступні означення.

Недопустимими називатимемо стани, в яких два або більше користувачів одночасно є керівниками одного й того самого сеансу та/або утримувачами одного й того самого рівня. Всі інші стани називатимемо *допустимими*.

Тоді на формальному рівні задачу верифікації КМ можна визначити таким чином:

Дано: N користувачів, L рівнів (кожному рівню відповідає один окремий ресурс), M сеансів, координаційний механізм.

Відповісти: «Так», якщо за будь-якого досяжного варіанту маркування мережі Петрі, що відповідає КМ, виконується умова допустимості маркування (тобто маркування відповідає допустимому стану). Інакше відповісти «Ні».

В такому вигляді задача верифікації КМ подібна до задачі верифікації агентів, розглянутої Вулдріджем та Данном [78 VIII.1]. Вони описують залежність складності задачі від характеристик середовища, в якому працює агент, та від складності специфікації задачі ψ , яка визначається як предикат над множиною пробігів агента в системі (позначається R):

$$\psi : R \rightarrow \{\text{true}, \text{false}\}$$

Пробіг агента в системі являє собою послідовність станів середовища та дій агента, внаслідок яких і відбувається зміна станів середовища.

КМ можна співставити з агентом, що виконує задачу підтримки — утримує середовище в одному з допустимих станів.

У випадку колаборативного середовища можливих пробігів нескінченно багато. Але для перевірки надійності нашого координаційного механізму нам достатньо перевірити, що кожний користувач може почергово в якості керівника кожного з можливих сеансів доступитися до кожного з можливих ресурсів. При цьому зручно накласти додаткову умову, що лише керівник сеансу може одержати доступ до ресурсу, оскільки просте приєднання до вже активованого сеансу не є взаємовиключним. Таким

чином, всю множину можливих пробігів можна звести до одного скінченного пробігу (умова зупинки — всі користувачі в якості керівників всіх можливих сеансів скористалися всіма наявними ресурсами). Такий пробіг (позначимо його R') зручно представити як послідовність певних етапів. *Етапом пробігу* називатимемо такий стан системи, в якому всі складові нашого середовища використовуються максимально повно. Це означає, що якщо, наприклад, $N = M = L$, то кожен користувач керує рівно одним своїм сеансом та утримує рівно один рівень. Насправді всі інші варіанти, коли кількість користувачів, сеансів та рівнів різна ($N = n, M = m, L = l; (n \neq m) \vee (l \neq n) \vee (m \neq l)$), можна окремо не розглядати, оскільки вони зводяться до випадку, коли $N = M = L = \min(n, m, l) = n$ (при $n \geq 2$). Пояснюється це таким чином:

а) за надлишкові сеанси та/або рівні користувачі не змагаються, тож окремо координувати їх використання немає потреби;

б) якщо користувачів забагато, то на кожному етапі робота КМ зводиться до узгодження роботи m користувачів (по одному на сеанс). Ті ж, кому не вистачило сеансу, з його точки зору перебувають поза системою, оскільки, згідно з додатковою умовою, доступ до ресурсу можливий лише для керівників сеансу, і дії цих користувачів, знов-таки, координувати не потрібно.

в) умова $n \geq 2$ потрібна для того, щоб ми не одержали вироджений випадок з одним користувачем, одним сеансом та одним рівнем, оскільки в цьому разі ні про яку конкуренцію та координацію вже не йдеться. Якщо ж в нас дійсно лише один сеанс та/або рівень, кількість користувачів прирівнюється до 2 (найменша кількість, за якої має місце конкуренція за ресурс).

Для побудови предикату ψ введемо такі допоміжні предикати:

- $KC(i, j)$ — значення «істина» \leftrightarrow в мережі Петрі в позиції «Користувач U_i є керівником сеансу S_j » стоїть фішка, інакше «хиба»;
- $UP(i, k)$ — значення «істина» \leftrightarrow в мережі Петрі в позиції «Користувач U_i є утримувачем рівня F_k » стоїть фішка, інакше «хиба»;
- $KCBV(j) \equiv \forall i_1 \forall i_2 (\neg(KC(i_1, j) \wedge KC(i_2, j) \wedge (i_1 \neq i_2)))$, де $i_1, i_2, j \in 1 \dots n$; аббревіатура означає «Контроль сеансу взаємовиключний». Вираз без кванторів позначимо $KCBV'(j)$;
- $UPBV(k) \equiv \forall i_1 \forall i_2 (\neg(UP(i_1, k) \wedge UP(i_2, k) \wedge (i_1 \neq i_2)))$, де $i_1, i_2, k \in 1 \dots n$; аббревіатура означає «Утримування рівня взаємовиключне». Вираз без кванторів позначимо $UPBV'(k)$;

- $Доп_стан(КМ) \equiv \forall j \forall k (КСВВ(j) \wedge УРВВ(k)) \equiv \forall i_1 \forall i_2 \forall j \forall k (КСВВ'(j) \wedge УРВВ'(k))$.

Тоді предикат $\psi(R')$ матиме такий вигляд:

$$\psi(R') = \begin{cases} \text{істина – якщо предикат } Доп_стан(КМ) \\ \text{виконується для всіх маркувань пробігу;} \\ \\ \text{хиба – в іншому випадку} \end{cases} \quad (1.4.1)$$

Для подальшої роботи зручно перейти від предикату до квантифікованої булевої формули (КБФ). Робиться це таким чином:

- $КС(i, j)$ замінюється булевою змінною $x_{i,j}$;
- $УР(i, k)$ замінюється булевою змінною $y_{i,k}$;
- $КСВВ(j)$ замінюється формулою:

$$\forall x_{1,1} \forall x_{1,2} \dots \forall x_{1,n} \forall x_{2,1} \dots \forall x_{n,n} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,1} \wedge x_{3,1}) \wedge \dots]; \quad (1.4.2)$$

- $УРВВ(k)$ замінюється формулою:

$$\forall y_{1,1} \forall y_{1,2} \dots \forall y_{1,n} \forall y_{2,1} \dots \forall y_{n,n} [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,1} \wedge y_{3,1}) \wedge \dots]; \quad (1.4.3)$$

- $Доп_стан(КМ)$ замінюється формулою:

$$\forall x_{1,1} \forall x_{1,2} \dots \forall x_{1,n} \forall x_{2,1} \dots \forall x_{n,n} \forall y_{1,1} \forall y_{1,2} \dots \forall y_{1,n} \forall y_{2,1} \dots \forall y_{n,n} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,1} \wedge x_{3,1}) \wedge \dots \wedge \neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,1} \wedge y_{3,1}) \wedge \dots]. \quad (1.4.4)$$

Наприклад, для $n=2$ ці формули виглядатимуть так:

$$(1.4.2): \forall x_{1,1} \forall x_{1,2} \forall x_{2,1} \forall x_{2,2} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,2} \wedge x_{2,2})];$$

$$(1.4.3): \forall y_{1,1} \forall y_{1,2} \forall y_{2,1} \forall y_{2,2} [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,2} \wedge y_{2,2})];$$

$$(1.4.4): \forall x_{1,1} \forall x_{1,2} \forall x_{2,1} \forall x_{2,2} \forall y_{1,1} \forall y_{1,2} \forall y_{2,1} \forall y_{2,2} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,2} \wedge x_{2,2}) \wedge \neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,2} \wedge y_{2,2})].$$

Згідно з [12], задача верифікації агентів для \sum_u^p -складних специфікацій задачі за обчислювальною складністю є \prod_{u+1}^p -повною. Звернімо увагу на поліноміальні ієрархії класів \sum_u^p та \prod_u^p . Вони охоплюють безліч класів, кожному з яких відповідає певне

значення $u \in \mathbb{N}$ (для $u < 0$ $\sum_u^p = \emptyset$). Припускається, що специфікацію Ψ можна представити такою машиною Тьюринга T_Ψ , яка приймає лише ті пробіги, які

задовольняють цій специфікації. Будь-яку \sum_u^p специфікацію можна задати альтернуючою машиною Тьюринга (alternating Turing machine), яка вирішує задачу за

поліноміальний час з використанням не більше u чергувань між станами існування та станами узагальнення (що відповідають кванторам існування та кванторам узагальнення).

Таким чином, $\sum_0^P \equiv P$, $\sum_1^P \equiv NP$.

Кожен клас складності $O(t(n))$ є множиною мов, які можна вирішити за $O(t(n))$ на відповідній машині Тьюринга. Зокрема, клас NP складається з мов, що вирішуються за поліноміальний час на недетермінованих машинах Тьюринга. В загальному випадку

$$\prod_u^P \equiv \text{co-} \sum_u^P.$$

Проте безпосередньо результат з праці [10] до нашої задачі не можна, оскільки наша специфікація задачі є co-NP -повною (або \prod_1^P), бо містить лише квантори узагальнення. Згідно з Пападімітріу [III.1, Розділ 10], co-NP — клас складності, до якого належать задачі із стислим відкиданням (succinct disqualification), на відміну від класу NP , який об'єднує задачі із стислим підтвердженням (succinct certificate). Оскільки задача тавтологічності булевого виразу (boolean validity problem) є co-NP -повною, наша специфікація задачі (формула 1.4.4) також є co-NP -повною. Тому для визначення складності задачі верифікації КМ потрібно довести лему:

Лема 1.4.1: Задача верифікації агентів для co-NP -повних специфікацій задачі буде за обчислювальною складністю co-NP -повною.

Доведення. co-NP -складність впливає безпосередньо зі складності специфікації задачі. Щоб довести co-NP -повноту, зведемо задачу визначення істинності КБФ, яка містить лише квантори узагальнення, до задачі верифікації агентів. Матимемо таку структуру:

$$\forall \bar{x}_1 \forall \bar{x}_2 K \forall \bar{x}_n \chi(\bar{x}_1, K, \bar{x}_n) \quad (1.4.5)$$

де:

- кожне \bar{x}_i — скінчений набір булевих змінних;
- $\chi(\bar{x}_1, K, \bar{x}_n)$ — формула логіки висловлювань на множині булевих змінних \bar{x}_1, K, \bar{x}_n .

Задача верифікації агентів на основі формули (1.4.5) будується таким чином. Нехай $\bar{x}_1 = x_1^1, x_1^2, \dots, x_1^m$ — найбільш зовнішній набір квантифікованих змінних. Кожній з цих змінних x_1^i поставимо у відповідність два можливих стани середовища: $e_{x_1^i}$ та $e_{-x_1^i}$, що відповідають значенням «істина» та «хиба» змінної x_1^i . Початковий стан середовища

позначимо e_0 . Середовище дозволяє агенту виконувати лише дію a_0 , і на i -те виконання цієї дії середовище відповідає переходом у стан $e_{x_1^i}$ або $e_{\neg x_1^i}$. Після m -го виконання дії a_0 пробіг завершується. Таким чином, кожний пробіг приписує кожній узагальнено квантифікованій змінній з множини $\bar{x}_1 = x_1^1, x_1^2, \dots, x_1^m$ своє значення, при цьому множина всіх можливих пробігів відповідає множині всіх можливих комбінацій значень цих змінних. Для даного пробігу r $\chi(\bar{x}_1, K, \bar{x}_n)[r/\bar{x}_1]$ — булева формула, отримана з формули $\chi(\bar{x}_1, K, \bar{x}_n)$ шляхом заміни кожної змінної x_1^i її значенням («істина» чи «хиба»), яке ця змінна одержала в рамках пробігу r .

Тоді специфікація задачі ψ матиме вигляд:

$$\psi(r) = \begin{cases} \text{істина} - \text{якщо со} - NP - \text{складна формула} \\ \quad \forall \bar{x}_2 K \forall \bar{x}_n \chi(\bar{x}_1, K, \bar{x}_n)[r/\bar{x}_1] \\ \text{набуває значення "істина"}; \\ \text{хиба} - \text{в іншому випадку} \end{cases} \quad (1.4.6)$$

Вхідна формула (1.4.5) набуватиме значення «істина» тоді, коли всі пробіги агента задовольняють специфікацію (1.4.6). Оскільки зведення поліноміальне, задача верифікації є за обчислювальною складністю со-NP-повною, що і треба було довести. ■

Тепер визначимо складність задачі верифікації КМ:

Теорема 1.4.1: Задача верифікації КМ є за обчислювальною складністю со-NP-повною.

Доведення. Специфікація задачі верифікації КМ (1.4.1) за обчислювальною складністю со-NP-повна. Згідно з лемою 1.4.1, задача верифікації агентів, яка є подібною до задачі верифікації КМ, у випадку со-NP-повної специфікації задачі є за обчислювальною складністю со-NP-повною. Отже, задача верифікації КМ є за обчислювальною складністю со-NP-повною, що і треба було довести. ■

8.5 Доведення тотальності алгоритму координації дій при виникненні помилок

В попередніх розділах було описано загальну структуру моделі колаборативного середовища (далі КС). Але досі ми розглядали лише безвідмовне середовище, в якому не виникає жодних позаштатних ситуацій. Реальне середовище такої властивості немає, тому наступним етапом буде моделювання поведінки системи в разі виникнення помилок,

викликаних як діями окремих користувачів, так і збоями обладнання чи програмного забезпечення системи.

Розглянемо, як зміниться мережа, описана в розділі 8.2, після додавання до неї позицій та переходів, що моделюватимуть реакцію системи на позаштатні ситуації. Спершу розглянемо її реакцію на помилки, зумовлені діями користувачів (наприклад, спробу порушити роботу системи чи доступитися до закритої інформації). Вважатимемо, що користувач скоює недозволені дії, маючи контроль над ресурсом. Маємо два можливих варіанти: а) він одночасно є керівником сеансу, б) не є керівником сеансу.

Почнемо з найпростішого випадку, коли порушник — простий учасник сеансу. В цьому випадку реакція системи обмежиться примусовим від'єднанням порушника від системи та звільнення зайнятого ним ресурсу. Змоделюємо цю реакцію за допомогою переходів „ U_i здійснив недозволену дію, не будучи керівником жодного з сеансів та займаючи при цьому F_k ”, ($i \in 1 \dots N$; $k \in 1 \dots L$). Таким чином, всього нам знадобиться $N * L$ таких переходів. Вхідними позиціями цих переходів для фіксованих користувача U_x і рівня F_y будуть:

- всі позиції виду „ U_x не є керівником S_j ” ($j \in 1 \dots M$);
- позиція „ U_x є утримувачем F_y ”;
- позиція „ F_y активно використовується U_x ”;
- позиція „Ресурс R_y використовується”;
- позиція „ U_x перебуває в системі”.

Вихідними позиціями будуть:

- „ U_x знаходиться поза системою”;
- „ F_y вільний”;
- „Ресурс R_y вільний”.

В результаті спрацювання цих переходів користувач опиниться поза системою, а зайняті ним рівень та ресурс будуть вільними та готовими до використання іншими користувачами.

Тепер розглянемо випадок, коли користувач, який здійснив нелегальну дію, був не лише утримувачем рівня, а й керівником сеансу. В цьому випадку при його аварійному відключенні треба прослідкувати, щоб воно якомога менше відбилося на роботі інших учасників цього сеансу. В залежності від реалізації системи та особливості ситуації, що склалася, можна або призначити нового керівника сеансу з-поміж його учасників, або коректно від'єднати їх. Змоделюємо другий сценарій. Для цього нам знадобляться $N * M * L$ переходів, позначених „ U_i здійснив недозволену дію, будучи керівником S_j та займаючи

при цьому F_k ”, ($i \in 1 \dots N$; $j \in 1 \dots M$; $k \in 1 \dots L$). Вхідними позиціями для фіксованих користувача U_x , рівня F_y та сеансу S_z будуть всі ті самі позиції, що й в попередньому випадку, крім однієї, „ U_x не є керівником S_z ”, місце якої займе „ U_x є керівником S_z ”, плюс позиція „ S_z активний”. До переліку вихідних позицій додадуться „ S_z вимикається” та нова спеціальна позиція „ U_x примусово від’єднується від системи” і перехід від неї до позиції „ U_x знаходиться поза системою”. Вона потрібна для більш чіткої синхронізації переходу користувача у стан „поза системою” з переходом сеансу у стан „вимкнений”. Отже, в підсумку маємо $N * M * L + N$ додаткових переходів та N додаткових позицій.

Тепер змодельємо поведінку системи в разі виникнення критичної помилки, яка унеможлиблює її подальшу роботу. Система повинна подолати таку ситуацію з мінімальними втратами і можливістю відновити роботу в повному обсязі. В термінах мережної моделі це означатиме повернення до початкового стану, коли всі користувачі знаходяться поза системою, сеанси неактивовані, рівні вільні. Розглянемо помилку, що відбувається у найнесприятливіший момент, коли всі користувачі активно працюють у системі чи то в якості утримувачів рівнів, чи в якості керівників сеансів. Щоб охопити якнайбільше користувачів, припустимо, що $N = M + L$, і що одні M користувачів є контролерами сеансів, а інші L — займають рівні. Щоб змодельювати поведінку такої системи мережею Петрі, нам знадобиться множина переходів виду „ U_1 є керівником S_1 , U_2 є керівником S_2 , ..., U_M є керівником S_M , U_{M+1} є утримувачем F_1 , U_{M+2} є утримувачем F_2 , ..., U_N є утримувачем F_L ” до „ U_N є керівником S_1 , U_{N-1} є керівником S_2 , ..., U_{N-M} є керівником S_M , U_{N-M-1} є утримувачем F_1 , U_{N-M-2} є утримувачем F_2 , ..., U_1 є утримувачем F_L ”. Всього таких переходів буде $N!$.

У випадку двох користувачів, одного сеансу та одного рівня мережа (позначимо її M) складатиметься з 40 позицій та 52 переходів. Використовуючи графічний редактор РМ Editeur та Petri Nets Toolbox, переведемо мережну модель в матричний вигляд, що оброблятиметься програмою MATLAB. В результаті обробки MATLAB’ом отримуємо дерево досяжності для нашої мережі.

Теорема 1.5.1. Мережа моделі колаборативної системи M обмежена і активна.

Доведення. Проаналізуємо дерево досяжності, побудоване за допомогою програми MATLAB (Додаток). Воно містить 186 можливих варіантів маркування, тому M обмежена (в іншому випадку маркувань було б нескінченно багато). Крім того, вона активна, оскільки дерево досяжності не містить термінальних вершин. Доведено.

Розглянемо алгоритм реагування КС на помилку з точки зору тотальності. Тотальним називається такий алгоритм мережної взаємодії, згідно з яким ухвалення рішення потребує участі всіх процесів в мережі [77]. Поняття тотальності алгоритмів є

важливим для розробки алгоритмів керування мережею, оскільки, по-перше, алгоритми розв'язання багатьох важливих задач мережної взаємодії є необхідно тотальними, а по-друге, для розв'язання цих задач можна застосувати будь-який тотальний алгоритм.

Покажемо, що алгоритм відкату системи після помилки в початковий стан тотальний. Ми можемо це припустити, якщо порівняємо його з розглянутим в праці [77] алгоритмом ресинхронізації, який полягає в тому, що спершу всі процеси в мережі переводяться до стану *synch*, а потім — до стану *normal*, при чому процес може перейти до стану *normal* лише після того, як в певний момент часу всі процеси одночасно перебувають в стані *synch*. В нашій моделі станом *synch* для процесів-користувачів буде стан „ U_i перебуває поза системою” ($i \in 1 \dots N$), для процесів-контролерів сеансу — „ S_j вимкнений” ($j \in 1 \dots M$), для процесів-контролерів рівнів — „ F_k вимкнений” ($k \in 1 \dots L$), для ресурсів — „ R_k відсутній” ($k \in 1 \dots L$). Стану *normal* відповідатиме стан користувача „ U_i намагається увійти в систему” ($i \in 1 \dots N$). Вводити стани-аналоги *normal* для інших компонентів системи немає потреби, оскільки, згідно з нашою моделлю, вони не зможуть перейти в жоден інший стан без команд користувачів. Тому в нашому випадку точкою ухвалення системою рішення вважатимемо перехід всіх користувачів в стан *normal*.

Доведемо це. Спершу наведемо формальне визначення тотальності.

Означення 1.5.1. Подія a передуює події b , якщо:

- 1) $a = b$, або a та b відбуваються в одному й тому самому процесі, і a відбувається раніше за b ;
- 2) a — подія надсилання повідомлення, а b — відповідна подія одержання повідомлення;
- 3) існує така подія c , що $a \rightarrow c$ та $c \rightarrow b$.

Означення 1.5.2. Виконання алгоритму *тотальне*, якщо принаймні один процес p ухвалює рішення, і для кожного $q \in P$ і кожного p , що ухвалює рішення, $e_q \rightarrow d_p$ (де P — множина всіх процесів, e_q — перша подія процесу e , d_p — подія ухвалення рішення процесом p). Алгоритм *тотальний* тоді, коли всі можливі його виконання тотальні.

Для доведення тотальності алгоритму роботи КС в разі виникнення помилки використаємо теорему:

Теорема 1.5.2. Нехай $a_1 \dots a_k$ — виконання деякого алгоритму A , та нехай $a_{\sigma(1)} \dots a_{\sigma(k)}$ — така перестановка подій, що з $a_{\sigma(i)} \rightarrow a_{\sigma(j)}$ випливає, що $i \leq j$. Тоді $a_{\sigma(1)} \dots a_{\sigma(k)}$ — також можливе виконання алгоритму A .

Таким чином, маємо:

Теорема 1.5.3. Будь-який алгоритм роботи КС в разі виникнення помилки тотальний.

Доведення. Нехай A — нетотальний алгоритм роботи КС в разі виникнення помилки, тобто існує таке виконання алгоритму A , в якому для деякого процесу q e_q не передуює ухваленню рішення. З теореми 2 випливає, що можна побудувати таке виконання, в якому ухвалення рішення передуює переходу q в стан *synch*, тому A — некоректний. Доведено. ■

Теорема 1.5.4. Для реалізації поведінки КС в разі виникнення помилки можна застосувати будь-який тотальний алгоритм.

Доведення. Нехай A — тотальний алгоритм. Змінимо його таким чином: кожний процес q змінює свій стан одразу після залучення в A , тобто в момент e_q , а кожний процес p (в нашому випадку, користувач) змінює стан на *normal* тоді, коли ухвалює рішення, тобто в момент d_p . Оскільки A тотальний, отриманий в результаті алгоритм роботи КС в разі виникнення помилки буде коректним. Доведено. ■

Результатом роботи є подальший розвиток та уточнення загальної моделі системи програмної підтримки мережної співпраці. Доведення теорем 1.5.3 і 1.5.4 дозволяє пов'язати нашу модель з одержаними раніше важливими теоретичними результатами, що полегшить подальшу роботу в цьому напрямку.

Література

1. Глибовець М. М. Інтелектуалізація навчання в телекомунікаційних системах дистанційної освіти // Вісник КНУ ім. Т.Г. Шевченка. Серія „Фізико-математичні науки”. — 2002. — №3. — С. 203—211.
2. Shang Yi, Shi Hongchi, Chen Su-Shing An Intelligent Distributed Environment for Active Learning. - 2001. - Available from <<http://www10.org/cdrom/papers/207/>>.
3. Чаплига В.М., Абашина Н.М., Вільдштейн Д.В. Система дистанційного навчання (СДН) “Академік”, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.174-176.
4. Ю.Рашкевич, Д.Пелешко, М.Пасєка, А.Стецюк Проектування Web-орієнтованих розподілених навчальних систем, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.143-152.
5. IEEE LTSC. <http://ltsc.ieee.org>
6. IMS. <http://www.imsproject.org>
7. Енциклопедія електронної освіти (Глибовець М.М....)
8. Networking Foundation for Collaborative Computing at Internet Scope --
9. Применение UML и шаблонов проектирования Введение в объектно-ориентированный анализ и проектирование. Крэг Ларман. Издательский дом «Вильямс» Москва, Санкт-Петербург, Киев, 2001.
10. XML. Натанья Питс-Моултис, Черил Кирк. BHV Дюссельдорф, Киев, Москва, Санкт-Петербург
11. Булах І. Поняття і основні принципи дистанційного навчання. Матеріали семінару „Методи та принципи дистанційного навчання” для випускників академічних програм обміну Інформаційного Агентства США (USIA). Рада Міжнародних Наукових Досліджень та Обмінів (IREX), 27 лютого 1999р.
<http://www.irex.kiev.ua/dl.html>
12. Подоляка О.И., Богдановский В.Г. Практические аспекты построения информационной системы дистанционного обучения. Тези доповідей 1-ої міжнародної науково-практичної конференції „Проблеми впровадження інформаційних технологій в економіці та бізнесі”, секція „Інформаційні технології в освіті”. <http://www.distance-learning.com.ua/lib/conf/4.html>
13. Андреев А.А. Введение в дистанционное обучение. – М.: - 1997 г.

14. Атанов Г.А. Пятикомпонентная предметная модель обучаемого — Telematics and Life-Long Learning, International Workshop TLLL – 2001
15. Norman D. A., Defending Human Attributes in the Age of the Mashine, First Person CD-ROM, New York, Voyager, 1995.
16. Смирнов А.В., Шереметов Л.Б. Многоагентная технология проектирования сложных систем. // Автоматизация проектирования №3-1998, №1-1999
- 17.
18. Proceedings of the 20th Word Conference on Open and Distance Learning “The Future of Learning – Learning for the Future: Shaping the Transition”, Dusseldorf, Germany, 1-5 April 2001, <http://www.icde.org>
19. Scriven M. The methology of evaluation/ Perspectives of curriculum evaluation (AERA monograph series on curriculum evaluation, Eds. R.W.Tyler, R.M Gagne, M.Scriven), Chicago: Rand McNally, 1967, p.39-83.
20. D.Newman, V.Scheirer, W.Shadish, C Wye Guiding Principles for Evaluators. A Report from the AEA Task Force on Guiding Principles for Evaluators, 1994
21. Чаплига В.М., Абашина Н.М., Вільдштейн Д.В. Система дистанційного навчання (СДН) “Академік”, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.174-176.
22. Ю.Рашкевич, Д.Пелешко, М.Пасека, А.Стецюк Проектування Web-орієнтованих розподілених навчальних систем, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.143-152.
23. Тихомиров.В.П., Титарев Л.Г., Шевченко К.К. Технологические системы в открытом образовании, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.168-171.
24. Грицай В.П. Информационная технология “Динамический гипертекст ТЕТ-А-ТЕТ”, Вестник Международного Соломонова университета, 2000, №2
25. Глибовець М.М., Олецкий О.В. Штучний інтелект, ВД “КМ Академія”, Київ, 2002, 365с.
26. Nunan T. Flexible delivery – a discussion of issues. University of South Australia: Distance Education Centre, 1995
27. Эпштейн В.Л. Введение в гипертекст и гипертекстовые системы. [URL:http://www.ipu.rssi.ru/publ/epstu.htm](http://www.ipu.rssi.ru/publ/epstu.htm)
28. Landa L., Algorithmization in Learning and Instruction, Englewood Cliffs, NJ: Educational Technology Publications, 1974
29. Атанов Г.А. Деятельностный подход в обучении, Донецк: ЕАИ-пресс, 2001.

30. Martin Fowler, Inversion of Control and the Dependency Injection pattern
(<http://www.martinfowler.com/articles/injection.html>)
31. Spring Framework, (<http://www.springframework.org/>)
32. Hibernate - Relational Persistence For Idiomatic Java (<http://www.hibernate.org/>)
33. Tapestry (<http://jakarta.apache.org/tapestry/>)
34. Scott W. Ambler, Introduction to Test Driven Development
(<http://www.agiledata.org/essays/tdd.html>)
35. Медвідь С.О., Комбінований компетентний паралельний генетичний алгоритм та його застосування для задачі побудови розкладів, «Проблеми програмування», спеціальний випуск, №2-3, 2004 – 261
36. Leonard Greenberg, LMS and LCMS: What's the Difference?
(<http://www.learningcircuits.org/NR/exeres/72E3F68C-4047-4379-8454-2B88C9D38FC5.htm>)
37. Shelley R. Robbins, The Evolution of the Learning Content Management System
(<http://www.learningcircuits.org/2002/apr2002/robbins.html>)
38. Dave Evangelisti, The Must-Have Features of an LMS
(<http://www.learningcircuits.org/2002/mar2002/evangelisti.html>)
39. Dave Egan, Top 10 LMS Purchasing Mistakes (and How to Avoid Them)
(<http://www.learningcircuits.org/2002/mar2002/Egan.htm>)
40. Гагарін О.О., Титенко С.В. Дослідження і аналіз методів та моделей інтелектуальних систем безперервного навчання // Наукові вісті НТУУ "КПІ". – 2007. – № 6(56). – С. 37-48.
41. <http://thepathos.org/> - home page
42. ред. Б. М. Бим-Бада, «Педагогический энциклопедический словарь»]
43. www.vedu.ru/ExpDic/
44. Гречихін, А. А., *Вузівська учбова книга: типологія, стандартизація, комп'ютеризація.*
45. Abrami & Barret, 2005
46. Baret, 2000; Challis, 2005
47. 8тШі & Tillema, 2003
48. Глибовець М.М., Іващенко С.А., Крусь О.О. Розробка системи управління вищого навчального закладу УДК 681.3.01
49. Вікіпедія - http://en.wikipedia.org/wiki/Ruby_%28programming_language%29
50. Бублик В.В. (2005). Розвиток інформаційних освітніх технологій. Сучасні методи і засоби комп'ютерної освіти 2005 (2) [Україна]

51. . V Boublik, W. Hesser, (2005). E-learning: Challenges and perspectives. (Періодичне видання) Наукові НаУКМА, том 36, 2005 (1).[Україна].
52. isburgh Mitchaell,(2004). Tips f Educational Technology & Society,
[www.pilotonlinenlernen.com]
53. Chapple Jeremy, (2004) Challenge of virtual education. For the Europe and the Ukraine in particular. International Journal of Artificial Intelligence in Education 15–18. Kiev, 2004. [emerecu.ukma.kiev.ua/docs]
54. Johnstone Sally, M., (2002). Sign of times: Changes is coming for E-learning
November/December, 2002.
55. Valiathan Purnima, (2005). Blended Learning Models -
[www.learningcircuits.org/2002/aug2002/vilathan.html]
56. Karrer, T., (2006) What is eLearning 2.0?
[www.elearningtech.blogspot.com/2006/02/what-is-elearning-20.html]
57. Welling L., Thomson L., раз риложений с помощью PHP и уSQL, 2-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2004.
58. Kosc Peter. Some Issues on E-learning implementation in EHEA with respect to virtual and physical mobility// Bologna and the challenges of e-Learning and distance education, Ghent, Belgium, 4-5, June 2004.
59. Dalziel James. Open Standards Versus Open Source in E-Learning –
[www.educause.edu/ir/library/pdf/EQM0340.pdf]
60. Hogue Dawn. Interworking: professional development through connection//English Journal – V.93 - #2 – November 2003.
61. Гречихин, А. А. Вузівська учбова книга: Типологія, стандартизація, комп'ютеризація /А. А. Гречихин, Ю. Г. Древс. – М.: Логос; МГУП, 2000. 13.
«Педагогический энциклопедический словарь» під ред. Б. М. Бим-Бада
62. Наукове видання :«Большая Российская энциклопедия» Москва 2002; 63
програмування, Київ, 3-4, 2003.
63. Монографія Горбунова-Посадова М.М. "Расширяемые программы", Москва, 1998.
64. <http://jep.ukma.kiev.ua/> - офіційний сайт міжнародного освітнього порталу JEP(Joint European Projects) сайт інтернет-проекту Online-GCC
65. <http://jep.ukma.kiev.ua> - офіційний сайт міжнародного освітнього порталу JEP(Joint European Projects)
66. http://www.online-gcc.com/sections/description_sections.php - офіційний сайт інтернет-проекту Online-GCC
67. <http://moodle.org> - офіційний сайт системи управління навчанням Moodle

68. <http://www.w3schools.com> - найбільший сайт з підтримки web-технологій та он-лайн посібників
69. <http://gcc.gnu.org/> - офіційний сайт набору компіляторів GNU
70. Boublik V., Hänßgen K., Towards development of Information Technologies in Education, Наукові записки НаУКМА - комп'ютерні науки, Т-19-20, 2002р., с. 28-34
71. Boublik V., Hesser W., E-Learning: Challenges and Perspectives, Наукові записки НаУКМА - комп'ютерні науки, Т-36, 2005р., с. 58-65
72. Downes, S (2005) E-Learning 2.0. <http://www.downes.ca/post/31741>
73. Ershov Andrey. Programming, the second literacy, 3rd IFIP Vvbrld Conference on Computers in Education (WCCE81).— Lausanne, 1981
74. Greenberg L., LMS and LCMS: What's the Difference?, December 9, 2002, <http://www.learningcircuits.org/NR/exeres/72E3F68C-4047-4379-8454-2B88C9D38FC5.htm>
75. Karrer, T., (2006) What is eLearning 2.0? <http://elearningtech.blogspot.com/2006/02/what-is-elearning-20.html>
76. Karrer, T., (2007) Understanding eLearning 2.0 <http://www.learningcircuits.org/2007/0707karrer.html>
77. Katz Richard N., Signifying a lot: what is really happening with IT in Higher education? CAUBO Annual Conference, Saskatoon, Saskatchewan, June 14, 2004
78. O'Reilly T., What Is Web 2.0, - 2005, <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
79. Plattner H., Trends and Concepts 2007 in the Software Industry, Lecture 2007.
80. Stallman, Richard M. (2001) "Contributors to GCC," in Using and Porting the GNU Compiler Collection (GCC) for gcc version 2.95 (Cambridge, Mass.: Free Software Foundation)
81. Welling L., Thomson L., разработка web-приложений с помощью PHP и MySQL, 2-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2004. 32
82. Електронне дистанційне навчання в Україні: Вісник UDL System, випуски 2000-2001, <http://www.udl.org.ua>
83. Интернет-порталы: содержание и технологии. Сборник научных статей. Выпуск 3. / Редкол.: А.Н. Тихонов (пред.) и др.; ФГУ ГНИИ ИТТ "Информика". - М.: Просвещение, 2005. - 590 с.: ил. – [<http://www.ict.edu.ru/ft/005511//index.html>]
84. http://en.wikipedia.org/wiki/GNU_Compiler_Collection - стаття про GCC в он-лайн енциклопедії Wikipedia

85. http://en.wikipedia.org/wiki/Virtual_learning_environment - стаття в Wikipedia про віртуальні навчальні середовища
86. <http://gcc.gnu.org/> - офіційний сайт збірки компіляторів GNU
87. <http://moodle.org/> - офіційний сайт системи управління навчанням Moodle
88. <http://php.net/> - сайт підтримки розвитку PHP та он-лайн документації
89. <http://wiki.e107.org/> - wiki-сторінки популярної CMS E107
90. <http://wikipedia.org/> - відкрита багатомовна wiki-енциклопедія.
- <http://www.ilias.de/> - офіційний сайт системи управління навчанням ILIAS
91. <http://www.mysql.com/> - офіційний сайт MySQL
92. <http://www.phpconcept.net/pclzip/> - PHP бібліотека PCLZip
93. <http://www.w3.org/> - світовий веб-консорціум
94. <http://www.w3schools.com/> - найбільший сайт з підтримки web-технологій та он-лайн посібників
95. Smith K., Tillema H. The Challenge of Assessing Portfolios- in Search for Criteria in A. Havnes & L. McDowell (Eds.) Balancing Dilemmas in Assessment and Learning in Contemporary Education / K. Smith, H. Tillema // New York: Routledge, Taylor and Francis Group, pp. 183-195. 2008.
96. Cambridge D. 2005. Integral ePortfolio Interoperability with the IMS ePortfolio Specifications [Electronic resource]. – Available at: <http://www.imsglobal.org/ep/>
97. IMS ePortfolio Best Practice and Implementation Guide [Electronic resource]. – Available at: http://www.imsglobal.org/ep/epv1p0/imsep_bestv1p0.html
98. XML Résumé specification v. 2.0. HR-XML Consortium. [Electronic resource]. – Available at: <http://xml.coverpages.org/HR-XML-ResumeSpecification200205.html>
99. Willson S. 2006. I Before E: Identity & e-Portfolios [Electronic resource]. – Available at: <http://zope.cetis.ac.uk/members/scott/blogview?entry=20060521192937>
100. Cambridge B., Cambridge D. 2003. The Future of Electronic Portfolio Technology: Supporting What We Know about Learning [Electronic resource]. – Available at: <http://ncepr.org/darren/publications.html>
101. “Generic” ePortfolio [Electronic Resource]. – Available at: <http://www.eportfolios.ac.uk>
102. Cotterill S., McDonald T., Drummond P., Hammond G. Design, implementation and evaluation of a ‘generic’ ePortfolio: the Newcastle experience / S. Cotterill, T. McDonald, P. Drummond, G. Hammond // University of Newcastle. - 2004. – pp.255-261.

103. S. Ravet. For an ePortfolio Enabled Architecture: ePortfolios, ePortfolio Management Systems and Organisers. EIFEL [Electronic Resource]. – Available at: <http://www.eife-l.org/publications/eportfolio/proceedings2/ep2007/proceedings-pdf-doc/eportfolio-2007.pdf>
104. IMS ePortfolio Information Model [Electronic Resource]. – Available at: http://www.imsglobal.org/ep/epv1p0/imsep_infov1p0.html
105. Grant S. 2005. Clear e-portfolio definitions: a prerequisite for effective interoperability [Electronic resource]. – Available at: <http://www.simongrant.org/pubs/ep2005/>
106. Willson S. Introducing IMS ePortfolio [Electronic Resource]. – Available at: <http://zope.cetis.ac.uk/members/scott/blogview?entry=20050718083203>
107. An Overview of E-Portfolios [electronic resource]: (The EDUCAUSE Learning Initiative) / G. Lorenzo, J. Ittelson // The EDUCAUSE Learning Initiative - July 2005. – Available at <http://www.educause.edu/ELI/AnOverviewofEPortfolios/156761>
108. Kinshuk A Conceptual Framework for Internet-based Intelligent Tutoring Systems / Kinshuk, A. Patel // Knowledge Transfer. — Vol. 2. — London, UK: pAce, 1997. — P. 117—124.
109. Chen N. S. Synchronous Learning Model over the Internet / N. S. Chen, H.-C. Ko, Kinshuk, T. Lin // Innovations in Education and Teaching International. — 2005. — Vol. 42 (2). — P. 181—194.
110. Shen H. H. Access Control for Collaborative Environments / H. H. Shen, P. Dewan // 1992 ACM conference on Computer-supported cooperative work, October 31 — November 4, 1992.: proc. — Toronto (Canada), 1992. — P. 51—58.
111. Floor Control in a Highly Collaborative Co-Located Task / Myers B. A., Chuang Yu Shan A., Tjandra M. et al. — [Submitted for publication]. — Режим доступа: www.cs.cmu.edu/~pebbles/papers/pebblesfloorcontrol.pdf.
112. Dommel H.-P. The Challenges of Ambient Collaboration / H.-P. Dommel // Richard Tapia Celebration of Diversity in Computing Conference, October 19—22, 2005.: proc. — Albuquerque (USA), 2005. — P. 10—13.
113. Candan K. S. Towards a Theory of Collaborative Multimedia / K. S. Candan, V. S. Subrahmanian, P. V. Rangan // IEEE International Conference on Multimedia Computing and Systems, June 17—23, 1996.: proc. — Hiroshima (Japan), 1996. — P. 279—282.
114. Papadopoulos G. A. Coordination Models and Languages / G. A. Papadopoulos, F. Arbab // Advances in Computers. — 1998. — No. 46. — P. 330—401.

115. Qiu X. Internet Collaboration using the W3C Document Object Model / X. Qiu, B. Carpenter, G. C. Fox // 2003 International Conference on Internet Computing, June 23-26, 2003.: proc. — Las Vegas (USA), 2003. — P. 643—647.
116. Elaine Allen I. Making the Grade - Online Education in the United States / I. Elaine Allen, J. Seaman. — Sloan-C, 2006. — 21 p. — Режим доступу: http://www.sloan-c.org/publications/survey/pdf/making_the_grade.pdf.
117. Глибовець М. М. Формальна модель координаційно-орієнтованої мережі для колаборативної системи навчання / М. М. Глибовець, Д. К. Гломозда // Проблеми програмування. — 2006. — № 2—3. Спец. вип. — С. 402—412
118. Гломозда Д. К. Формальна модель функціонування колаборативного середовища / Д. К. Гломозда, М. М. Глибовець // Третя Міжнародна конференція «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD'2006), 5-8 груд. 2006 р.: тези доп. — Київ (Україна), 2006. — С. 225—230
119. Глибовець Н. Н. Сложность задачи верификации координационного механизма системы программной поддержки совместной сетевой работы / Н. Н. Глибовець, Д. К. Гломозда // Кибернетика и системный анализ. — 2008. — № 4. — С. 15—19.

Перелік посилань

1. Глибовець М. М. Інтелектуалізація навчання в телекомунікаційних системах дистанційної освіти // Вісник КНУ ім. Т.Г. Шевченка. Серія „Фізико-математичні науки”. — 2002. — №3. — С. 203—211.
2. Shang Yi, Shi Hongchi, Chen Su-Shing An Intelligent Distributed Environment for Active Learning. - 2001. - Available from <<http://www10.org/cdrom/papers/207/>>.
3. Енциклопедія електронної освіти (Глибовець М.М....)
4. Networking Foundation for Collaborative Computing at Internet Scope --
5. Применение UML и шаблонов проектирования Введение в объектно-ориентированный анализ и проектирование. Крэг Ларман. Издательский дом «Вильямс» Москва, Санкт-Петербург, Киев, 2001.
6. XML. Натанья Питс-Моултис, Черил Кирк. BHV Дюссельдорф, Киев, Москва, Санкт-Петербург
7. Булах І. Поняття і основні принципи дистанційного навчання. Матеріали семінару „Методи та принципи дистанційного навчання” для випускників академічних програм обміну Інформаційного Агентства США (USIA). Рада Міжнародних Наукових Досліджень та Обмінів (IREX), 27 лютого 1999р. <http://www.irex.kiev.ua/dl.html>
8. Подоляка О.И., Богдановский В.Г. Практические аспекты построения информационной системы дистанционного обучения. Тези доповідей 1-ої міжнародної науково-практичної конференції „Проблеми впровадження інформаційних технологій в економіці та бізнесі”, секція „Інформаційні технології в освіті”. <http://www.distance-learning.com.ua/lib/conf/4.html>
9. Андреев А.А. Введение в дистанционное обучение. – М.: - 1997 г.
10. Атанов Г.А. Пятикомпонентная предметная модель обучаемого — Telematics and Life-Long Learning, International Workshop TLLL – 2001
11. Norman D. A., Defending Human Attributes in the Age of the Mashine, First Person CD-ROM, New York, Voyager, 1995.
12. Proceedings of the 20th Word Conference on Open and Distance Learning “The Future of Learning – Learning for the Future: Shaping the Transition”, Dusseldorf, Germany, 1-5 April 2001, <http://www.icde.org>
13. Scriven M. The methology of evaluetion/ Perspectives of curriculum evaluation (AERA monograph series on curriculum evaluation, Eds. R.W.Tyler, R.M Gagne, M.Scriven), Chicago: Rand McNally, 1967, p.39-83.

14. D.Newman, V.Scheirer, W.Shadish, C Wye Guiding Principles for Evaluators. A Report from the AEA Task Force on Guiding Principles for Evaluators, 1994
15. Чаплига В.М., Абашина Н.М., Вільдштейн Д.В. Система дистанційного навчання (СДН) “Академік”, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.174-176.
16. Ю.Рашкевич, Д.Пелешко, М.Пасєка, А.Стецюк Проектування Web-орієнтованих розподілених навчальних систем, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.143-152.
17. Тихомиров.В.П., Титарев Л.Г., Шевченко К.К. Технологические системы в открытом образовании, Збірник матеріалів міжнародної наради “Телематика і безперервна освіта”, Київ, 2001., ст.168-171.
18. Грицай В.П. Информационная технология “Динамический гипертекст ТЕТ-А-ТЕТ”, Вестник Международного Соломонова университета, 2000, №2
19. Глибовець М.М., Олецкий О.В. Штучний інтелект, ВД “КМ Академія”, Київ, 2002, 365с.
20. Nunan T. Flexible delivery – a discussion of issues. University of South Australia: Distance Education Centre, 1995
21. Эпштейн В.Л. Введение в гипертекст и гипертекстовые системы.
[URL:http://www.ipu.rssi.ru/publ/epstu.htm](http://www.ipu.rssi.ru/publ/epstu.htm)
22. Landa L., Algorithmization in Learning and Instruction, Englewood Cliffs, NJ: Educational Technology Publications, 1974
23. Атанов Г.А. Деятельностный подход в обучении, Донецк: ЕАИ-пресс, 2001.
24. Martin Fowler, Inversion of Control and the Dependency Injection pattern
(<http://www.martinfowler.com/articles/injection.html>)
25. Spring Framework, (<http://www.springframework.org/>)
26. Hibernate - Relational Persistence For Idiomatic Java (<http://www.hibernate.org/>)
27. Tapestry (<http://jakarta.apache.org/tapestry/>)
28. Scott W. Ambler, Introduction to Test Driven Development
(<http://www.agiledata.org/essays/tdd.html>)
29. Медвідь С.О., Комбінований компетентний паралельний генетичний алгоритм та його застосування для задачі побудови розкладів, «Проблеми програмування», спеціальний випуск, №2-3, 2004 – 261
30. Leonard Greenberg, LMS and LCMS: What's the Difference?
(<http://www.learningcircuits.org/NR/exeres/72E3F68C-4047-4379-8454-2B88C9D38FC5.htm>)

31. Shelley R. Robbins, The Evolution of the Learning Content Management System (<http://www.learningcircuits.org/2002/apr2002/robbins.html>)
32. Dave Evangelisti, The Must-Have Features of an LMS (<http://www.learningcircuits.org/2002/mar2002/evangelisti.html>)
33. Dave Egan, Top 10 LMS Purchasing Mistakes (and How to Avoid Them) (<http://www.learningcircuits.org/2002/mar2002/Egan.htm>)
34. Гагарін О.О., Титенко С.В. Дослідження і аналіз методів та моделей інтелектуальних систем безперервного навчання // Наукові вісті НТУУ "КПІ". – 2007. – № 6(56). – С. 37-48.
35. <http://thepathos.org/> - home page
36. Глибовець М.М., Іващенко С.А., Крусь О.О. Розробка системи управління вищого навчального закладу УДК 681.3.01
37. Вікіпедія - http://en.wikipedia.org/wiki/Ruby_%28programming_language%29
38. Бублик В.В. (2005). *Розвиток інформаційних освітніх технологій*. Сучасні методи і засоби комп'ютерної освіти 2005 (2) [Україна]
39. V Boublik, W. Hesser, (2005). *E-learning: Challenges and perspectives*. (Періодичне видання) Наукові НаУКМА, том 36, 2005 (1).[Україна].
40. Weisburgh Mitchaell,(2004). *Tips f* Educational Technology & Society, [www.pilotonlinenlernen.com]
41. Chapple Jeremy, (2004) *Challenge of virtual education. For the Europe and the Ukraine in particular*. International Journal of Artificial Intelligence in Education 15–18. Kiev, 2004. [emerecu.ukma.kiev.ua/docs]
42. Johnstone Sally, M., (2002). *Sign of times: Changes is coming for E-learning* November/December, 2002.
43. Valiathan Purnima, (2005). *Blended Learning Models* - [www.learningcircuits.org/2002/aug2002/vilathan.html]
44. Karrer, T., (2006) *What is eLearning 2.0?* [www.elearningtech.blogspot.com/2006/02/what-is-elearning-20.html]
45. Welling L., Thomson L., *раз риложений с помощью PHP и ySQL, 2-е издание*. : Пер. с англ. – М. : Издательский дом «Вильямс», 2004.
46. Kosc Peter. *Some Issues on E-learning implementation in EHEA with respect to virtual and physical mobility// Bologna and the challenges of e-Learning and distance education*, Ghent, Belgium, 4-5, June 2004.
47. Dalziel James. *Open Standards Versus Open Source in E-Learning* – [www.educause.edu/ir/library/pdf/EQM0340.pdf]

48. Hogue Dawn. *Interworking: professional development through connection*//*English Journal* – V.93 - #2 – November 2003.
49. Гречихин, А. А. *Вузівська учбова книга: Типологія, стандартизація, комп'ютеризація* /А. А. Гречихин, Ю. Г. Дреус. – М.: Логос; МГУП, 2000. 13. «Педагогический энциклопедический словарь» під ред. Б. М. Бим-Бада
50. Наукове видання : «*Большая Российская энциклопедия*» Москва 2002; 63 програмування, Київ, 3-4, 2003.
51. Монографія Горбунова-Посадова М.М. *"Расширяемые программы"*, Москва, 1998.
52. <http://jep.ukma.kiev.ua/> - офіційний сайт міжнародного освітнього порталу JEP(Joint European Projects) сайт інтернет-проекту Online-GCC
53. <http://jep.ukma.kiev.ua> - офіційний сайт міжнародного освітнього порталу JEP(Joint European Projects)
54. http://www.online-gcc.com/sections/description_sections.php - офіційний сайт інтернет-проекту Online-GCC
55. <http://moodle.org> - офіційний сайт системи управління навчанням Moodle
56. <http://www.w3schools.com> - найбільший сайт з підтримки web-технологій та он-лайн посібників
57. <http://gcc.gnu.org/> - офіційний сайт набору компіляторів GNU
58. 1. Boublik V., Hänßgen K., *Towards development of Information Technologies in Education*, Наукові записки НаУКМА - комп'ютерні науки, Т-19-20, 2002р., с. 28-34
59. 2. Boublik V., Hesser W., *E-Learning: Challenges and Perspectives*, Наукові записки НаУКМА - комп'ютерні науки, Т-36, 2005р., с. 58-65
60. 3. Downes, S (2005) *E-Learning 2.0*. <http://www.downes.ca/post/31741>
61. 4. Ershov Andrey. *Programming, the second literacy*, 3rd IFIP Wvbrld Conference on Computers in Education (WCCE81).— Lausanne, 1981
62. 5. Greenberg L., *LMS and LCMS: What's the Difference?*, December 9, 2002, <http://www.learningcircuits.org/NR/exeres/72E3F68C-4047-4379-8454-2B88C9D38FC5.htm>
63. 6. Karrer, T., (2006) *What is eLearning 2.0?* <http://elearningtech.blogspot.com/2006/02/what-is-elearning-20.html>
64. 7. Karrer, T., (2007) *Understanding eLearning 2.0* <http://www.learningcircuits.org/2007/0707karrer.html>
65. 8. Katz Richard N., *Signifying a lot: what is really happening with IT in Higher education?* CAUBO Annual Conference, Saskatoon, Saskatchewan, June 14, 2004

66. 9. O'Reilly T., *What Is Web 2.0*, - 2005,
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
67. 10. Plattner H., *Trends and Concepts 2007 in the Software Industry*, Lecture 2007.
68. 11. Stallman, Richard M. (2001) "*Contributors to GCC*," in *Using and Porting the GNU Compiler Collection (GCC) for gcc version 2.95* (Cambridge, Mass.: Free Software Foundation)
69. 12. Welling L., Thomson L., *розробка web-приложень з допомогою PHP и MySQL, 2-е издание*. : Пер. с англ. – М. : Издательский дом «Вильямс», 2004. 32
70. 13. *Електронне дистанційне навчання в Україні: Вісник UDL System*, випуски 2000-2001, <http://www.udl.org.ua>
71. 14. *Интернет-порталы: содержание и технологии. Сборник научных статей*. Выпуск 3. / Редкол.: А.Н. Тихонов (пред.) и др.; ФГУ ГНИИ ИТТ "Информика". - М.: Просвещение, 2005. - 590 с.: ил. – [<http://www.ict.edu.ru/ft/005511/index.html>]
72. 15. http://en.wikipedia.org/wiki/GNU_Compiler_Collection - стаття про GCC в он-лайн енциклопедії Wikipedia
73. 16. http://en.wikipedia.org/wiki/Virtual_learning_environment - стаття в Wikipedia про віртуальні навчальні середовища
74. 17. <http://gcc.gnu.org/> - офіційний сайт збірки компіляторів GNU
75. 18. <http://moodle.org/> - офіційний сайт системи управління навчанням Moodle
76. 19. <http://php.net/> - сайт підтримки розвитку PHP та он-лайн документації
77. 20. <http://wiki.e107.org/> - wiki-сторінки популярної CMS E107
78. 21. <http://wikipedia.org/> - відкрита багатомовна wiki-енциклопедія.
79. 22. <http://www.ilias.de/> - офіційний сайт системи управління навчанням ILIAS
80. 23. <http://www.mysql.com/> - офіційний сайт MySQL
81. 24. <http://www.phpconcept.net/pclzip/> - PHP бібліотека PCLZip
82. 25. <http://www.w3.org/> - світовий веб-консорціум
83. 26. <http://www.w3schools.com/> - найбільший сайт з підтримки web-технологій та он-лайн посібників
84. Smith K., Tillema H. The Challenge of Assessing Portfolios- in Search for Criteria in A. Havnes & L. McDowell (Eds.) *Balancing Dilemmas in Assessment and Learning in Contemporary Education* / K. Smith, H. Tillema // New York: Routledge, Taylor and Francis Group, pp. 183-195. 2008.
85. Cambridge D. 2005. Integral ePortfolio Interoperability with the IMS ePortfolio Specifications [Electronic resource]. – Available at: <http://www.imsglobal.org/ep/>

86. IMS ePortfolio Best Practice and Implementation Guide [Electronic resource]. – Available at: http://www.imsglobal.org/ep/epv1p0/imsep_bestv1p0.html
87. XML Résumé specification v. 2.0. HR-XML Consortium. [Electronic resource]. – Available at: <http://xml.coverpages.org/HR-XML-ResumeSpecification200205.html>
88. Willson S. 2006. I Before E: Identity & e-Portfolios [Electronic resource]. – Available at: <http://zope.cetis.ac.uk/members/scott/blogview?entry=20060521192937>
89. Cambridge B., Cambridge D. 2003. The Future of Electronic Portfolio Technology: Supporting What We Know about Learning [Electronic resource]. – Available at: <http://ncepr.org/darren/publications.html>
90. “Generic” ePortfolio [Electronic Resource]. – Available at: <http://www.eportfolios.ac.uk>
91. Cotterill S., McDonald T., Drummond P., Hammond G. Design, implementation and evaluation of a ‘generic’ ePortfolio: the Newcastle experience / S. Cotterill, T. McDonald, P. Drummond, G. Hammond // University of Newcastle. - 2004. – pp.255-261.
92. S. Ravet. For an ePortfolio Enabled Architecture: ePortfolios, ePortfolio Management Systems and Organisers. EIFEL [Electronic Resource]. – Available at: <http://www.eifel.org/publications/eportfolio/proceedings2/ep2007/proceedings-pdf-doc/eportfolio-2007.pdf>
93. IMS ePortfolio Information Model [Electronic Resource]. – Available at: http://www.imsglobal.org/ep/epv1p0/imsep_infov1p0.html
94. Grant S. 2005. Clear e-portfolio definitions: a prerequisite for effective interoperability [Electronic resource]. – Available at: <http://www.simongrant.org/pubs/ep2005/>
95. Willson S. Introducing IMS ePortfolio [Electronic Resource]. – Available at: <http://zope.cetis.ac.uk/members/scott/blogview?entry=20050718083203>
96. An Overview of E-Portfolios [electronic resource]: (The EDUCAUSE Learning Initiative) / G. Lorenzo, J. Ittelson // The EDUCAUSE Learning Initiative - July 2005. – Available at <http://www.educause.edu/ELI/AnOverviewofEPortfolios/156761>
97. Kinshuk A Conceptual Framework for Internet-based Intelligent Tutoring Systems / Kinshuk, A. Patel // Knowledge Transfer. — Vol. 2. — London, UK: pAce, 1997. — P. 117—124.
98. Chen N. S. Synchronous Learning Model over the Internet / N. S. Chen, H.-C. Ko, Kinshuk, T. Lin // Innovations in Education and Teaching International. — 2005. — Vol. 42 (2). — P. 181—194.
99. Shen H. H. Access Control for Collaborative Environments / H. H. Shen, P. Dewan // 1992 ACM conference on Computer-supported cooperative work, October 31 — November 4, 1992.: proc. — Toronto (Canada), 1992. — P. 51—58.

100. Floor Control in a Highly Collaborative Co-Located Task / Myers B. A., Chuang Yu Shan A., Tjandra M. et al. — [Submitted for publication]. — Режим доступу: www.cs.cmu.edu/~pebbles/papers/pebblesfloorcontrol.pdf.
101. Dommel H.-P. The Challenges of Ambient Collaboration / H.-P. Dommel // Richard Tapia Celebration of Diversity in Computing Conference, October 19—22, 2005.: proc. — Albuquerque (USA), 2005. — P. 10—13.
102. Candan K. S. Towards a Theory of Collaborative Multimedia / K. S. Candan, V. S. Subrahmanian, P. V. Rangan // IEEE International Conference on Multimedia Computing and Systems, June 17—23, 1996.: proc. — Hiroshima (Japan), 1996. — P. 279—282.
103. Papadopoulos G. A. Coordination Models and Languages / G. A. Papadopoulos, F. Arbab // Advances in Computers. — 1998. — No. 46. — P. 330—401.
104. Qiu X. Internet Collaboration using the W3C Document Object Model / X. Qiu, B. Carpenter, G. C. Fox // 2003 International Conference on Internet Computing, June 23-26, 2003.: proc. — Las Vegas (USA), 2003. — P. 643—647.
105. Elaine Allen I. Making the Grade - Online Education in the United States / I. Elaine Allen, J. Seaman. — Sloan-C, 2006. — 21 p. — Режим доступу: http://www.sloan-c.org/publications/survey/pdf/making_the_grade.pdf.
106. Глибовець М. М. Формальна модель координаційно-орієнтованої мережі для колаборативної системи навчання / М. М. Глибовець, Д. К. Гломозда // Проблеми програмування. — 2006. — № 2—3. Спец. вип. — С. 402—412
107. Гломозда Д. К. Формальна модель функціонування колаборативного середовища / Д. К. Гломозда, М. М. Глибовець // Третя Міжнародна конференція «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD'2006), 5-8 груд. 2006 р.: тези доп. — Київ (Україна), 2006. — С. 225—230
108. Глибовець Н. Н. Сложность задачи верификации координационного механизма системы программной поддержки совместной сетевой работы / Н. Н. Глибовець, Д. К. Гломозда // Кибернетика и системный анализ. — 2008. — № 4. — С. 15—19.