ЗАСТОСУВАННЯ UML ПРИ ПРОЕКТУВАННІ ЗАСОБІВ ДИСТАНЦІЙНОГО НАВЧАННЯ

О.І., Гороховський Т.ІТрояновська

Вступ

Загальновідомо, що проектування засобів дистанційного навчання є складним та ітеративним процесом. Найскладнішим завданням є проектування самого курсу. Часто стається так, що помилки (часто глобальні) і прорахунки системи виявляють себе на етапах тестування або використання програмного продукту. Вирішенням таких задач є спочатку процес моделювання структури курсу дистанційного навчання, а потім – наповнення курсу змістовною інформацією та навчальним матеріалом.

Процес створення курсу починається з визначення предметної галузі. Якщо справа йде про мультимедійний продукт, то зразу визначається для якої дисципліни він створюється. Будь-яка дисципліна складається з розділів, які між собою пов'язані посиланнями. Гіпертекст, гіпермедіа дозволяють створити такі зв'язки, і відповідно, в межах курсу виникають так звані "шляхи" (trails), які залежать від того, якими посиланнями користуватиметься студент при вивченні дисципліни за допомогою даного курсу. Шляхи вивчення можуть бути різними. Це дозволяє індивідуалізувати навчання, що вигідно відрізняє дистанційне навчання від звичайного стаціонарного чи заочного, де програма курсу вибудовується під певний "середній рівень". Однак грамотно спроектувати систему реального часу надто важко, тому існуючі системи дистанційного навчання поки що не здатні конкурувати з академічними.

Проблема постає при створенні загального формату, який би враховував усі особливості дистанційного навчання, включаючи загальний формат курсу та індивідуальний підхід до кожного студента. Для того, щоб отримати таку модель, можна створити сценарії (стилі), які б перетворювали документи до певного загального вигляду.

Ця архітектура може мати вигляд:

1. Схема типу (форми) документа;
2. Набір документів даного типу;
3. Листи DSSSL-стилів, які задавали перетворення.

Аналогічну архітектуру пропонує сучасна, більш досконала щодо можливостей та гнучкості, мова XML. Саме вона і використовується за основу конструювання курсів.

Однак при наскрізній розробці курсу за допомогою даної мови виникає інша проблема – це все ж таки мова, з своєю граматикою, синтаксисом та правилами. Таким чином, її використання в нашій методиці підвищує вимоги до автора курсу – він принаймні має бути програмістом. Це дуже звужує коло аудиторії для нашої методики.

Вихід можна знайти в іншій мові – Universal Modeling Language (UML). Це перша в світі мова діаграм та зв'язків, які дозволяють моделювати будь-які комплекси (чи то програмні, чи то технологічні, або навіть апаратні) за допомогою порівняно простих дій над візуальними символами. За допомогою UML можна створити найбільш важку частину курсу – тип документу та стилі перетворення. Нижче ми розглянемо технологію такого підходу детально.

При створенні дистанційних курсів найбільш частою проблемою є подібна до висвітленої вище проблеми різноманітності програм для їх обробки. Кожен фрейм (фреймом будемо вважати найменшу частину дистанційного курсу, не маючи на увазі аналогічне поняття в мові розмітки документів HTML, і яка містить логічно завершений та легко зрозумілий інформаційний пакет), по суті, є документом, який потрібно відображати відповідно вимогам користувача (або апаратним вимогам його комп'ютера), перетворювати відповідно режиму перегляду (як довідник, або ж лекцію, або ж контрольну роботу), виокремлювати з фреймів контрольні питання та формувати підсумкові контрольні роботи.

Для цього можна застосовувати аналогічний підхід, створивши загальну схему документа-"фрейма", а потім за нею створити набір сценаріїв-стилів перетворення, за якими формуватиметься представлення документу тощо.

В даній статті викладена спроба показати, як можна за допомогою мови універсального моделювання UML можна значно полегшити процес проектування курсів дистанційного навчання, і досягти якісно кращого рівня.

Фаза проектування

Розглянемо проектування курсу з позицій абстрагування даних. Представимо галузі знань у вигляді окремих просторів імен (namespace), які пов'язані між собою зв'язком типу "посилання". Таким чином ми

закладаємо в проект глобальні зв'язки між галузями знань. Це буде найвищий рівень абстракції. Наступним рівнем абстракції, очевидно, є внутрішньогалузеві зв'язки. Однак якщо на попередньому рівні зв'язки були одновимірні (між об'єктами одного рівня), то тепер вони перетворяться на двовимірні (у випадку, якщо якийсь розділ в галузі пов'язаний з іншою галуззю знань).

Якщо продумати структуру курсу і записати її у вигляді діаграм UML, то за допомогою спеціальних програмних засобів (наприклад, Rational Rose) автоматично створюється структура сайту дистанційного навчання, а при використанні C++ чи Java – готову модель для програмного продукту.

Порівняти курс дистанційного навчання, спроектований вищенаведеним способом з традиційним проектування м найкраще по структурі. Розбиття навчального матеріалу на фрейми – невеличкі порції знань і встановлення між ними гіпертекстових посиланьстворює одновимірні зв'язки, і їх кількість зростає за експоненційним законом із зростанням кількості фреймів. Тому, при проектуванні складного та комплексного курсу ми можемо стикнутись з непід'ємною складністю зв'язків, яка призведе до спрощень, обмеження гіперзв'язків, а відтак втрачається гнучкість.

Більш практичним аспектом використання UML є не стільки використання його при проектуванні моделі курсу, а швидше проектування вже означених вище "шляхів" та тестів як діаграми процесів. Загальновідомо, що в даний час вони моделюються за допомогою блок-схем алгоритмів. В принципі, це є правильним, однак можливості блок-схеми дуже обмежена.

Специфікація

Для того, щоб описати доцільність запропонованого методу проектування, варто почати зі специфікації.

Розглянемо пакет "Дії" (Actions) зі специфікації UML [1,2]. Тут цілісна система опису будь-якої послідовності дій, поділена на дві частини:

1. Специфікація дії (описана як взаємодія різних компонентів, що беруть в ній участь);
2. Модель виконання дії (у вигляді простої діаграми процесу, що складається з станів "очікування", "готовність", "виконання" та "завершення").

За допомогою специфікації дії можна створити, наприклад, модель тесту, яка включатиме не лише елементи керування (поля введення, радіокнопки, прапорці тощо), а й допоміжні матеріали, а також можна описати систему оцінювання знань (закласти в специфікацію). Відповідно, за допомогою програмного засобу Rational Rose чи Rational Process Workbench, ми зможемо після планування моделі отримати вже готовий кістяк програми.

Але це не єдине можливе застосування. При розгляді пакету "Поведінкові моделі", варто зазначити, що з ним пов'язаний і вищезгаданий пакет "Дії" – він має приблизно аналогічний синтаксис. Тут з'являються кілька конструктивних механізмів.

По-перше, це "взаємодії". За допомогою цього набору синтаксичних елементів можна спроектувати групову роботу студентів (участь в форумі), чи зробити значне вдосконалення – створити модель лабораторної роботи, яку можуть виконувати кілька студентів не просто одночасно, а розподіляючи певні задачі, і взаємодіючи між собою.

По-друге, там є так звані "діаграми активності". У поєднанні з вищезгаданою методикою поділу рівнів абстракції це дає змогу промоделювати всі можливі шляхи вивчення студентом конкретного курсу, а відтак прослідкувати всі можливі переходи, що дозволить створити гнучку систему оцінки знань.

Дійсно, якщо студент поверхнево вивчив якийсь необов'язковий розділ знань, то в тести, відповідно, не потрібно включати питання з нього, і навпаки, якщо він його уважно продивився, то було б корисно включити відповідні контрольні запитання. Передбачити таку можливість нам дає поєднання діаграм активності та механізм опису автоматів зі станами (Active State Machines), де можна продивитись, як проходить шлях опанування студентом матеріалу курсу на рівні моделі, а не на етапі тестування результуючого програмного продукту.

Проектування частини дистанційного курсу навчання

Враховуючи обсяг даної статті, неможливо привести повний цикл моделювання курсу, доцільно зосередитись на детальному описі моделювання мовою UML лише однієї її частини – входу або реєстрації в системі користувача, структурно представлену на рис. 1 [4].

Зайшовши на веб-сайт центру системи ДО абітурієнт проходить *Анкетування* і отримує доступ до *Демо-версії курсу*, яка включає *Демо-лекцію*, *Демо-тест*, йому пропонується пройти лекцію і тест і написати *Відгук*. *Правила прийому* і *Додаткову інформацію* абітурієнт може відправити собі додому на *Друк*.

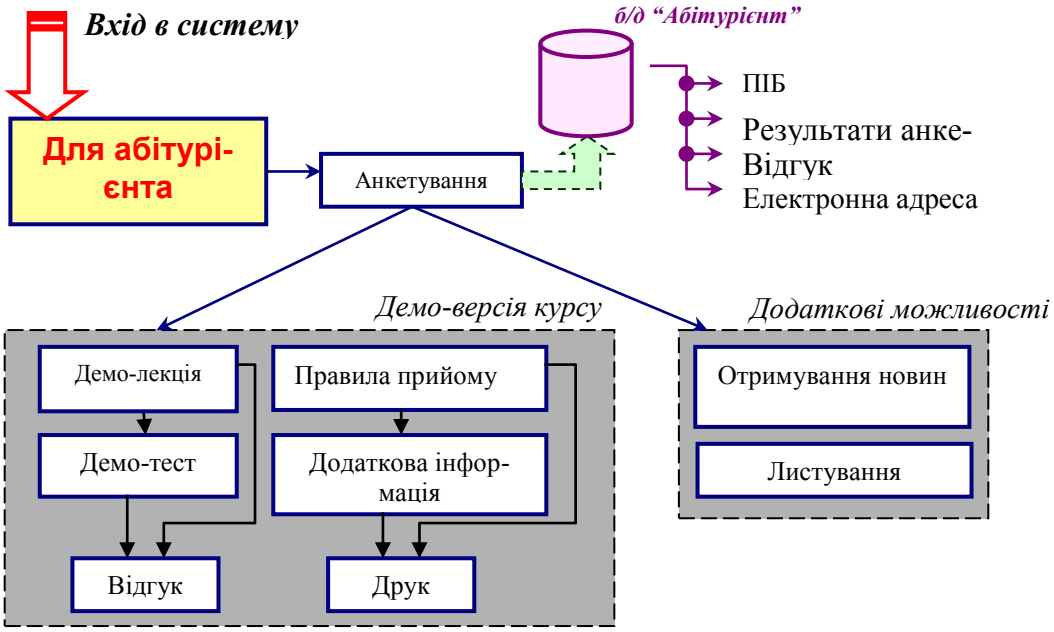


Рис 1. Вхід у систему дистанційної освіти для абітурієнта

Абітурієнтові також доступна служба *Додоткових можливостей,* що включає *Отримування новин* (якщо на сайті з'являються нові спеціальності чи нові умови і можливостідля навчання, він знатиме про них з електронної пошти) і *Листування* з центром системи ДО [4].

При моделюванні якоїсь частини системи за допомогою UML процес проектування завжди починається з діаграми акторів (або діаграми взаємодії), яка дозволяє проілюструвати компоненти, задіяні в даній частині системи, а також визначити основні елементи протоколу (правил взаємодії) між ними.

Визначимо основні типи акторів, які будемо використовувати в діаграмі. Першим, очевидно, повинен стати "пристрій введення". Цей тип актора, загалом, може бути легко замінений актором "користувач", оскільки пристроєм введення в даному випадку керувати може тільки реальний суб'єкт, а не автоматизована система чи підсистема передачі даних, як вказано в [1]. Однак з міркувань універсальності така заміна понять не рівноцінна. Наступним типом актора є "інтерфейс", що в даному випадку репрезентуватиме Web-форму, яку заповнює користувач при вході чи реєстрації в системі. Далі нам буде потрібний актор "керуючий об'єкт", що виконуватиме функції посередника між актором "сховище даних" та інтерфейсом користувача.

На рис. 2 наведено діаграму взаємодії, яка ілюструє процес входу та реєстрації користувача. Оскільки при взаємодії інтерфейсу та керуючого об'єкту використовується комплексний об'єкт "Персоналія", на рис.3 зображено UML-діаграму агрегації, яка показує структуру цього типу даних.

На діаграмі взаємодії (див. рис.2) показано, як об'єкти динамічно спілкуються між собою, отримуючи та відсилаючи повідомлення. Ця діаграма представляє структурну організацію взаємодіючих об'єктів, зображених у вигляді прямокутників і з'єнувальних дуг (прямих). Позначені стрілки поряд з дугами визначають ім'я повідомлення і напрямок його передачі між об'єктами.

На діаграмі агрегації (див. рис.3) представлено відношення ієрархії агрегування і композиції. Ці відношення типу ціле/частина. Відношення позначається ромбом. Ромб однією вершиною прилягає до прямокутника класу, який є частиною у відношенні виду частина/ціле.

Діаграма взаємодії (див. рис. 2) дозволяє визначити на перших етапах проектування, які компоненти будуть задіяні в процесі роботи даної частини системи. Стрілки на діаграмі показують напрямок руху даних, при цьому спочатку вказується тип взаємодії, а через двокрапку необхідні для неї вхідні дані. Таким чином зразу визначається інтерфейс взаємодії, за яким будуть працювати компоненти. Використовуючи Rational Rose, за кожним зв'язком генерується інтерфейс (мається на увазі абстрактний тип даних C++, або ж interface мови Java), який потім реалізовуватиме або наслідувати клас, екземпляром якого буде даний об'єкт, або спеціалізований об'єкт обміну даних.

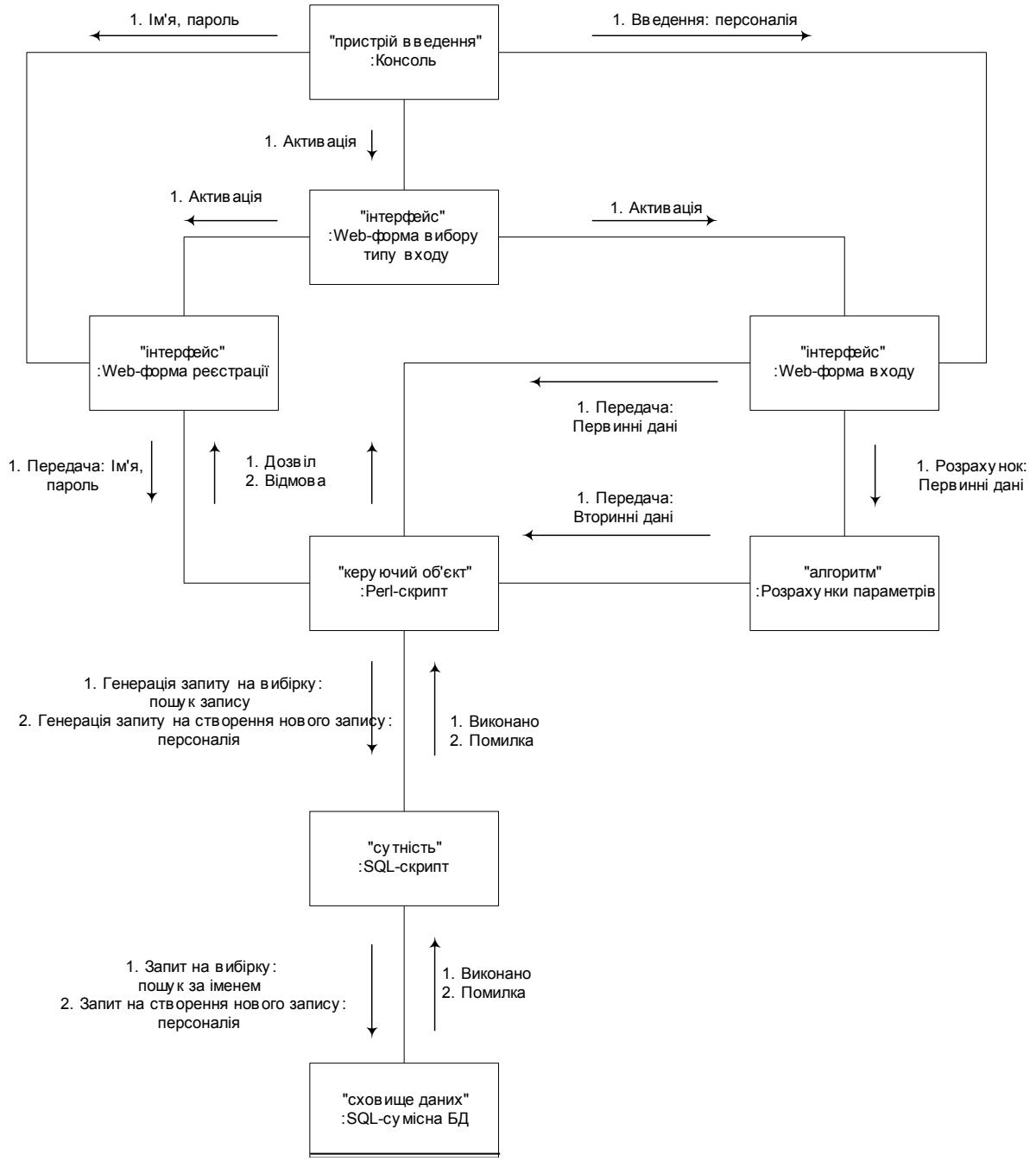


Рисунок 2 - Діаграма взаємодії

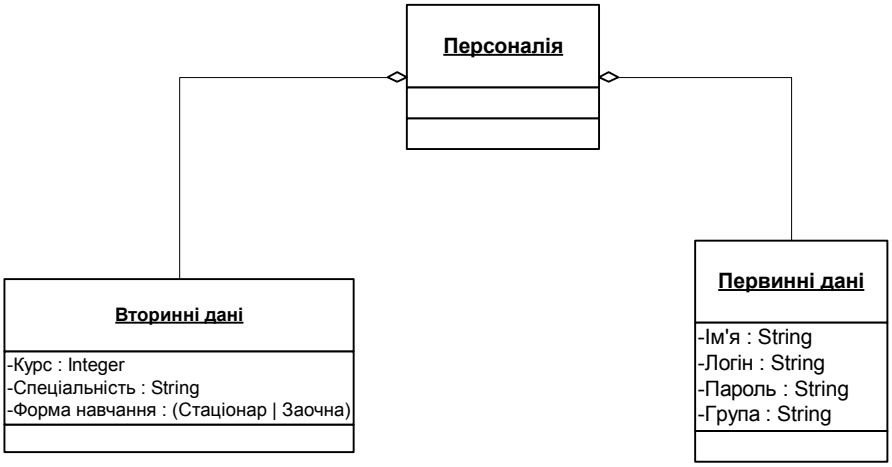
Припустимо, що такий спеціалізований об’єкт використовуватиметься при передачі імені та пароля до керуючого об’єкту. Такий об’єкт варто створити з міркування захисту даних, адже при прямому обміні об’єктами у Java, наприклад, автоматично підключається захищене з’єднання SSH. Отже, застосуємо Rational Rose, і отримаємо код:

```
public interface AuthenticInterface
{
  String getName();
  String getPasswd();
}
public class AuthenticInfo implements AuthenticInterface
{
  private transient String name;
  private transient String passwd;
  synchronized void setPasswd(String passwd)
```

```
 {
   this.passwd = passwd;
 }
  synchronized void setName(String name)
  {
    this.name = name;
  }
  synchronized String getName()
 {
   return name;
 }
 synchronized String getPasswd()
 {
   return passwd;
 }
}
```

Розглянемо діаграму 3, оскільки вона є основою для структурного типу даних, який буде використовуватись при створенні акторів “інтерфейс”, а також використовуватиметься як тип даних у інших акторах. Очевидно, що такими є “керуючий об’єкт” та “інтерфейс”.

Рисунок 3 - Діаграма агрегації



Отже, тип даних “Персоналія” є агрегованим типом, який містить у собі екземпляри типів “Первинні дані” та “Вторинні дані”, кожен з яких містить в собі декілька інформаційних полів. Зазначимо також, що другий має тип “transient”, тобто ці дані не зберігатимуться. Застосуємо XML Spy для генерації еталонного XML-файлу для даної схеми (передати її можна з Rational Rose за допомогою того ж таки XML-формату), і отримуємо такий варіант:

```
<xml version=”1.0”>
<personalia>
  <name value=”” />
  <login value=”” />
  <passwd value=”” />
  <group value=”” />
</personalia>
```

Вторинні дані не входять до згенерованого XML-документа, оскільки в UML-діаграмі мають тип “transient”. Це обумовлено, зокрема, тим, що вони розраховуються в процесі роботи системи за допомогою актора «алгоритм». Цей документ можна використати для передачі частини SOAP-повідомлення (що дає автоматичну інтеграцію в мережу глобальних Web-сервісів на основі цього протоколу), знімаючи необхідність у розробці додаткового протоколу для обміну даними між компонентами.

В згенерованому тексті зразу вказано для даних тип transient, що означає, що дані, які містяться в цих змінних не будуть зберігатись після знищення об’єкту в локальній пам’яті, а будуть стерті “збірником сміття” (що автоматично дає певний рівень захисту даних), а для методів доступу до даних автоматично вказаний модифікатор доступу synchronized, що вказує на можливе використання роботи через мережевий протокол. Цей модифікатор також вказує, що при передачі повинен використовуватись контроль з’єднання. Це унеможливить перехоплення даних зловмисником шляхом штучного розірвання з’єднання, і підміни па-

ролю. Звісно, цих заходів недосить для повноцінного захисту даних, однак воно одержані на етапі проектування системи, що значно скорочує час на розробку, адже у випадку роботи “з нуля” це все потрібно було б виконувати програмісту. Додання таких простих речей знімає з нього необхідність займатись елементарним і дозволяє зосередитись на розробці більш комплексних рішень.

Висновок

Використання UML на етапі проектування засобів дистанційного навчання дозволить визначити та розв’язати задачі і проблеми, які зазвичай виникають вже після реалізації готового продукту. Причому, етапі проектування, що значно – майже на 30% [3] скоротить час на розробку курсу і підвищить його якісні показники, а також дозволить реалізувати ті функції, які донедавна були привілеєм складних замовних систем.

Таким чином, всі сценарії генерації коду є зовнішніми і написані на простій макромові, яка дозволяє навіть посередньому спеціалісту визначити необхідну граматику, а по-друге, стало можливим на основі схеми документа формувати сценарії для перетворення документів і також автоматично формувати інтерактивні форми для редагування документів, що робить його майже ідеальним інструментом для запропонованого методу проектування дистанційних курсів.

Література

1. UML Specification – Object Management Group, March 2003
2. UML – Хассан Гома, Москва 2002
3. G. Booch, J. Rumbaugh, I. Jacobson – UML User’s Guide – Addison-Wesley, 2003
4. Гороховский О.И., Трояновская Т. И. Исследование и разработка создания дистанционных курсов. Азербайджан – Украина – Болгария: ИОН.–с.190-193.

Гороховський Олександр Іванович, к.т.н., доцент, декан ФПК СКТ, кафедра ОТ, Вінницький національний технічний університет, Хмельницьке шоссее, 95, Вінниця, 21021, Україна, тел.: (0432) 44-04-05, e-mail: goroh@lili.vstu.vinnica.ua

Трояновська Тетяна Іванівна, магістрант, Вінницький національний технічний університет, Хмельницьке шоссее, 95, Вінниця, 21021, Україна, e-mail: trtet@mail.ru