УДК 681.3;658.56
А.Мушак; О.Провотар, докт.фіз.-мат.наук
Інститут кібернетики ім. В.М. Глушкова

ДИСТАНЦІЙНЕ НАВЧАННЯ: ВІД ПОБУДОВИ МОДЕЛЕЙ ДО ГЕНЕРАЦІЇ ПРОГРАМНОГО КОДУ

За допомогою мови моделювання UML, послуговуючись CASE-засобом Rational Rose, побудовано модель Системи Гнучкого Дистанційного Навчання. Систему розглянуто з двох точок зору: проаналізовано предметну область та досліджено систему з функціональної точки зору. Продемонстрована можливість генерації програмного коду на основі діаграми класів для окремо взятого прецедента. Описано переваги технології створення програмного забезпечення на основі UML-моделі.

1. Передмова

Одним із напрямків розвитку технологій навчання є дистанційне навчання на основі глобальних комп'ютерних комунікацій. Сьогоднішній день вимагає якнайповнішого використання засобів інтерактивної мультимедіа, за допомогою яких реалізується інтерфейс програмної реалізації Системи Гнучкого Дистанційного Навчання (СГДН), в курси дистанційного навчання. Відтак, постає питання про місце інтерактивної мультимедіа в екземплярі програмної реалізації СГДН відповідно до тієї чи іншої навчальної дисципліни. Через це очевидною є необхідність побудови моделі СГДН, яка б сприяла подальшій програмній реалізації фрагментів дистанційних курсів, а отже, й спрощувала б їх побудову. При цьому, модель СГДН будується за допомогою сучасної технології створення інформаційних систем, яка передбачає послуговування об'єктно-орієнтованим підходом задля проектування чи аналізу, мовою моделювання Unified Modelling Language (UML) і деяким CASE-засобом, наприклад, Rational Rose (RR), для автоматизації побудови діаграм моделі та генерації на їх основі програмного коду.

Програмна реалізація деяких фрагментів дистанційних курсів здійснюється із використанням елементів методології програмування, що ґрунтується на основі Композиційно-Структурного Моделювання (КСМ-технології) [1,2], яка дозволяє конструювати програмні системи із готових компонент — модулів, що реалізують ті чи інші (залежно від потреб) алгоритми з визначеним семантичним змістом.

Відтак, суть проблеми, яка вирішується у статті полягає у гармонійному поєднанні вже існуючих інформаційних технологій з метою автоматизації найтрудомісткіших етапів побудови дистанційних курсів. Використання елементів КСМ-технології програмування для побудови дистанційних курсів пропонується вперше.

2. Модель Системи Гнучкого Дистанційного Навчання

Дану систему розглядатимемо з двох точок зору. Спочатку проаналізуємо предметну область, що еквівалентно дослідженню Процесу Дистанційного Навчання (ПДН). При цьому акцент робитимемо на інтерактивних засобах, за допомогою яких відбувається взаємодія користувачів з екземпляром програмної реалізації СГДН.

Наступним етапом буде функціональний аналіз системи з метою опису інтерфейса користувача програмної реалізації СГДН.

2.1. Порядок моделювання

Процес побудови моделі СГДН мовою моделювання UML зводиться до створення набору діаграм. Хід моделювання різних предметних областей і створення на основі даних моделей програмних систем не є строго детермінованим, але існує деякий загальний порядок роботи [3,4]. Враховуючи це, а також специфіку системи, що розглядається, моделювання в нашому випадку передбачає такий порядок роботи: будуємо діаграму варіантів використання системи, далі розглядаємо кожен прецедент і стосовно нього будуємо діаграму діяльності, після цього означуємо об'єкти системи і, маючи їх набір, будуємо діаграми станів, діаграми взаємодій і діаграми класів. Дана послідовність створення діаграм моделі не пов'язана із тим чи іншим процесом виготовлення програмного продукту. Проте відомо, що один спосіб організації — Раціональний Уніфікований Процес (РУП) найкраще пристосований до UML.

Скористаємося методикою РУП з метою побудови моделі СГДН, провівши її об'єктно-орієнтований аналіз. В об'єктно-орієнтованому аналізі ми намагаємося змоделювати світ, означуючи класи та об'єкти, які формують лексику проблемної області [5].

Для того, щоб означити класи та об'єкти створюваної моделі, необхідно проаналізувати вимоги, які ставлять до системи.

2.2. Дійові особи ПДН та їх функції

СГДН не функціонує ізольовано. Вона взаємодіє із користувачами, які послуговуються нею задля досягнення своїх цілей. Кожен із користувачів очікує, що система буде поводити себе визначеним, цілком передбаченим чином. Іншими словами, аналіз системи передбачає розгляд деякого набору варіантів її використання.

Беручи до уваги концептуальну модель мови UML, акторами СГДН є: абітурієнт (кандидат у студенти), студент, викладач, адміністратор. Варто розглядати й узагальненого актора „Користувач", для якого вищезазначені актори є спеціалізованими.

Актори взаємодіють із системою задля використання її у своїх цілях. Відтак, розглянемо варіанти використання системи всіма означеними нами акторами. Варіанти використання системи, згідно термінології мови UML, мають назву „прецеденти".

У СГДН виділимо такі прецеденти:

- запрошувати на навчання;
- зареєструвати;
- визначити рівень знань щодо послуговування Комунікаційними та Інформаційними Технологіями (КІТ);
- визначити рівень знань з навчальної дисципліни;
- зарахувати студентом;
- познайомити з іншими студентами, викладачем, адміністратором;
- дати настанови щодо проходження навчання;
- сприяти самостійному конструюванню знань, формуванню навичок та умінь;
- презентувати готові знання, формувати навички та уміння під керівництвом викладача;
- реагувати на студентську ініціативу;

- проконтролювати рівень оволодіння навчальним матеріалом;
- тотально проекзаменувати;
- підвести підсумок навчання;
- сприяти налагодженню неформального спілкування;
- розширювати кругозір;
- спонукати до оволодіння новими КІТ;
- підтримувати професійний рівень;
- забезпечити заробітною платою.

Встановимо прецеденти-нащадки для окремих із варіантів використання і пов'яжемо їх відношеннями узагальнення. Варіації поведінки окремих прецедентів виділимо у нові прецеденти, пов'язуючи їх відношенням розширення. Подаємо прецеденти, акторів і відношення між ними на діаграмі прецедентів (рис.1).

На цій діаграмі інтерес для подальшого вивчення становлять лише ті прецеденти, які стосуються інтерактивної взаємодії між студентом, викладачем та адміністратором. Ці варіанти використання системи зафарбовані чорним кольором. Серед них виберемо лише один прецедент, наприклад, „Консультувати", і для нього опишемо весь хід побудови діаграм, набір яких є частиною моделі СГДН.

Встановлення прецедентів є специфікуванням поведінки СГДН. Оскільки задля побудови моделі СГДН застосовуватимемо об'єктно-орієнтований аналіз системи, то „розкриття" будь-якого прецеденту передбачає оперування поняттям об'єкта, що є ключовою одиницею такого аналізу.

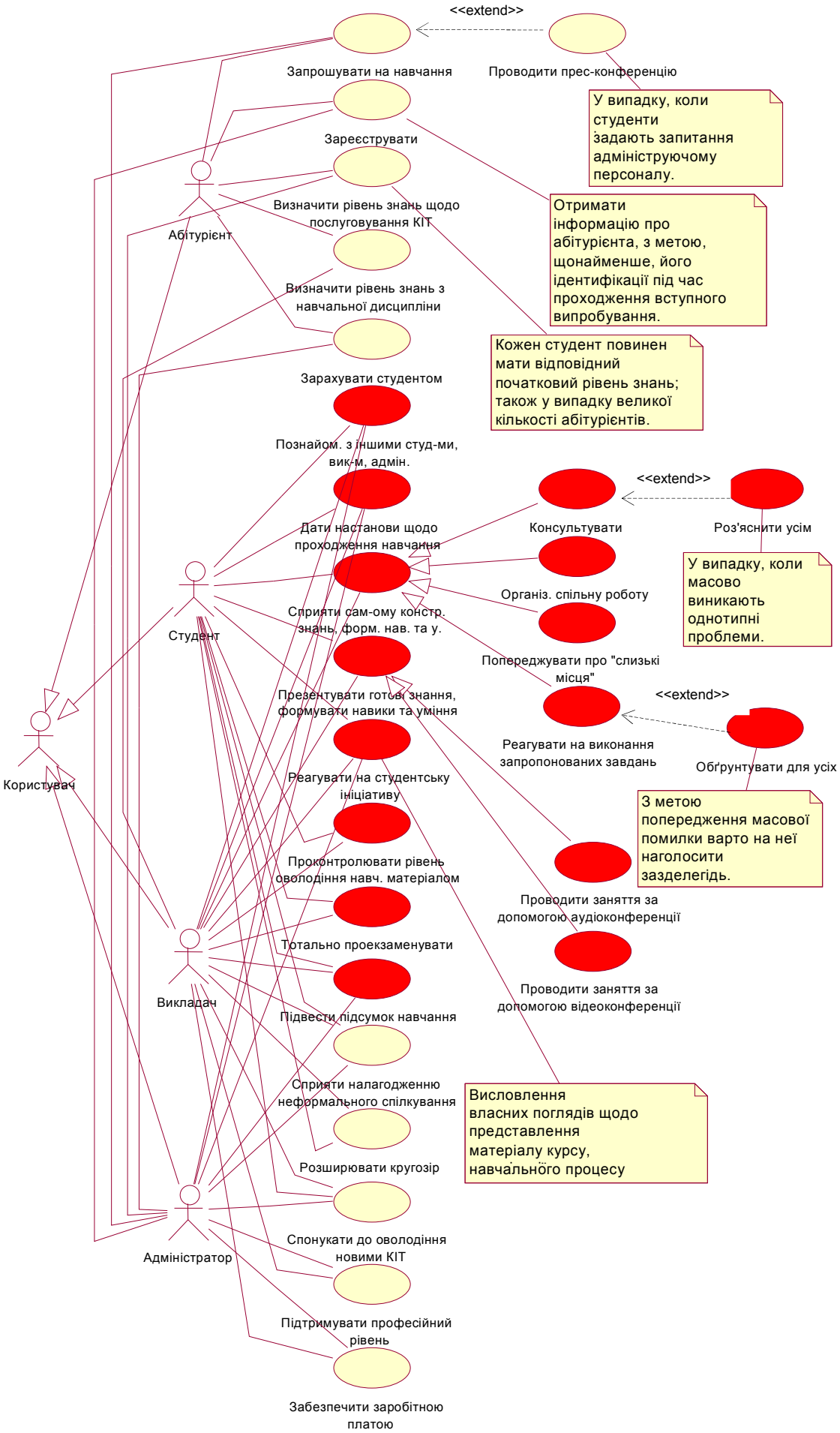

Рис. 1. Діаграма прецедентів „СГДН”. Модель предметної області

2.3. Опис поведінки об'єктів ПДН в контексті прецедента "Консультувати"

Динаміку поведінки сукупності об'єктів, що відповідає вибраному прецедентові, змоделюємо за допомогою діаграми діяльності, де увага сконцентрована, насамперед, на змісті діяльності, в якій беруть участь об'єкти (рис. 2).

Для моделювання поведінки окремого об'єкта використовують автомат. Змоделюємо життєвий цикл об'єкта „Студент" за допомогою діаграми станів (рис. 3), що показує автомат.

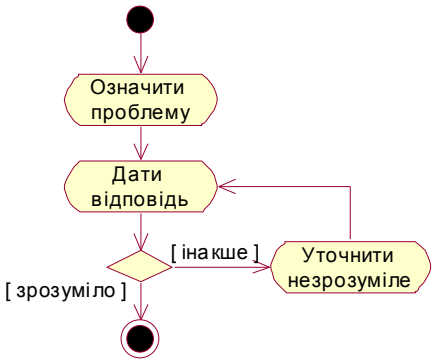

Рис. 2. Діаграма діяльності „Консультувати"

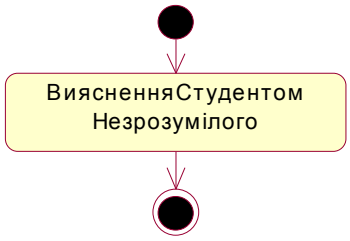

Рис. 3. Діаграма станів „Консультувати"

У ході життєдіяльності системи об'єкти не залишаються статичними: вони взаємодіють один з одним, постійно обмінюючись повідомленнями. Для моделювання динамічних аспектів кооперацій використовують, зокрема, взаємодії.

2.4. Опис сценаріїв функції "Консультувати" ПДН

Взаємодія — це поведінка, що виражається в обміні повідомленнями між множиною об'єктів у деякому контексті, в результаті чого є досяжною певна мета.

Згідно з UML, будь-яку взаємодію можна представити, акцентуючи увагу на часовій впорядкованості подій або ж на їх послідовності в контексті структурної організації об'єктів, що розглядається. Відтак, для прецедента „Консультувати" матимемо відповідно діаграму послідовностей (рис. 4) та діаграму кооперації (рис. 5), що є ізоморфними.

Для специфікації поведінки системи (діаграми послідовностей, кооперації, діяльності та станів) оперують поняттям об'єкта, тобто екземпляра класу. Кожен клас є складовою опису статичної структури предметної області, що розглядається, а їх набір становить словник модельованої системи.

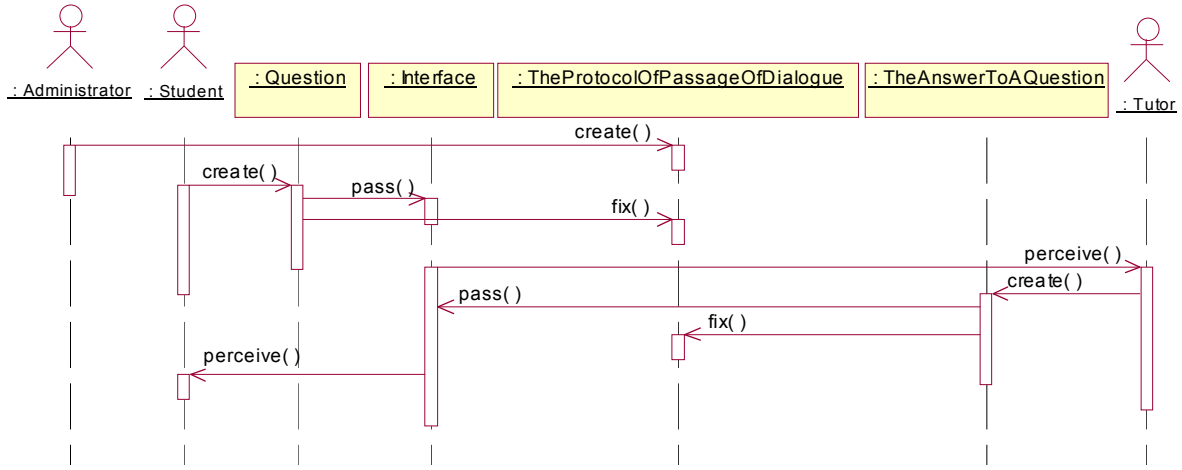

Рис. 4. Діаграма послідовностей „Консультувати"

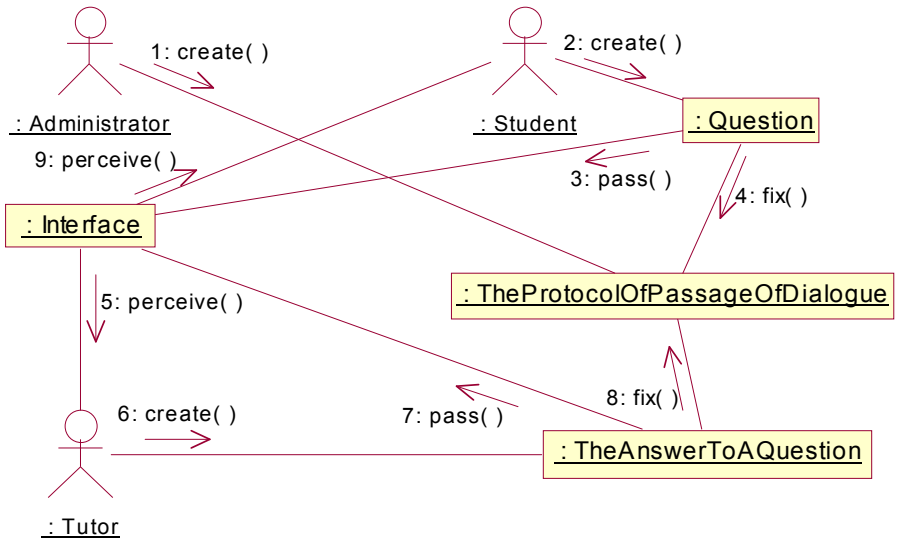

Рис. 5. Діаграма кооперації „Консультувати”

2.5. Опис документообігу ПДН в контексті прецедента "Консультувати"

Означимо класи, породжувані об’єкти яких взаємодіють в межах прецедента “Консультувати”, за допомогою діаграми класів (рис. 6).

Процес побудови моделі предметної області СГДН є свого роду її дослідженням з технологічної точки зору. Результатом цього наукового розгляду є встановлення вимог до програмної реалізації системи. Оскільки послуговування засобами інтерактивної мультимедіа зводиться до взаємодії користувача з інтерфейсом програмної реалізації СГДН, то природним є дослідження СГДН в контексті реалізації інтерфейсу, загалом дослідження системи з функціонального погляду.

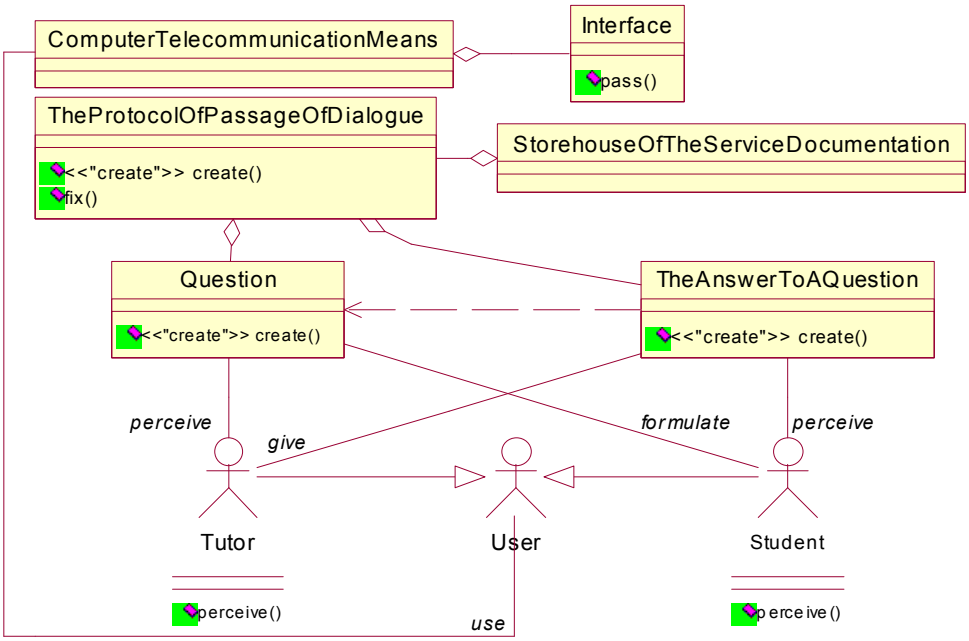

Рис. 6. Діаграма класів „Консультувати”

2.6. Функціональна модель СГДН

Для побудови моделі СГДН у функціональному аспекті використана та сама методика, що й при побудові моделі предметної області. Діаграма прецедентів має такий вигляд (рис. 7).

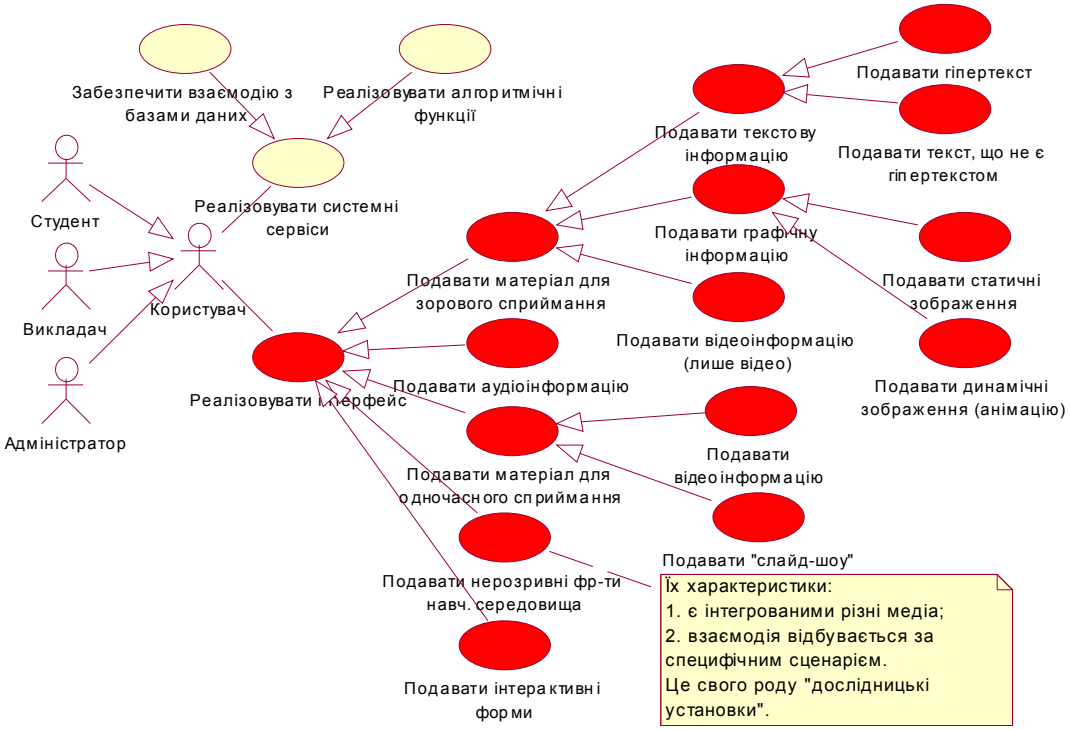

Рис. 7. Діаграма прецедентів „СГДН”. Функціональна модель системи

В кінцевому випадку інтерес для подальшого розгляду становлять діаграми класів, через те, що за допомогою них не тільки якнайповніше вдається представити статичні аспекти даної системи, але й через те, що саме на основі діаграм класів, використовуючи той чи інший CASE-засіб, генерують програмні коди, які можуть виступати будівельними блоками окремої прикладної програми.

3. Від діаграм до коду

За допомогою програмного пакету RR, що в ньому побудовані вищеописані діаграми, утворимо мовою Java програмний код-каркас для класу “TheProtocolOfPassageOfDialodue” діаграми класів „Консультувати”. Отримуємо наступний лістинг (послуговуємося інструментальною системою JBuilder 5 Personal) (рис. 8).

Із лістинга видно, що в класі, що розглядається, означуються дані — об’єктні змінні theStorehouseOfTheServiceDocumentation, theTheAnswerToAQuestion, theQuestion та означується програмний код (складовий методів TheProtocolOfPassageOfDialogue(), create(), fix()), який керує цими даними. Щоправда, враховуючи демонстраційний характер даного класу (немає прив’язки до конкретної прикладної задачі), тіло кожного з методів є порожнім. Окрім означення методів і змінних, лістинг містить коментарі, в тому числі й коментарі документації [6].

Маючи набір ось таких класів-каркасів (.java-файли), розробники конструюють програмні системи відповідно до поставлених задач.

Висновки

Розглядаючи еволюцію розробки програмного забезпечення, можна простежити перехід від розробки програм „зверху вниз” до об’єктно-орієнтованої технології створення програмних комплексів. Перехід на якісно новий рівень розробки назрів

через постійне ускладнення програм, їх важкодоступність для огляду. Ця об'єктно-орієнтована технологія передбачає обов'язкове використання деякого CASE-засобу з метою підтримки технологічного процесу розробки програмних систем від задуму до завершення життєвого циклу програми. Використаний для побудови моделі СГДН CASE-інструмент RR дозволив зекономити час та якісно виконати роботу. Спроектований, насамперед, для спрощення побудови моделей, цей засіб дозволяє створювати виразні й зрозумілі діаграми для всіх розробників, залучених у проект, до того ж, що важливо, не тільки в момент розробки, але й через декілька місяців по тому. Якщо в проект залучають нову людину, вона, проаналізувавши діаграми RR, оперативно ввійде в курс справи.

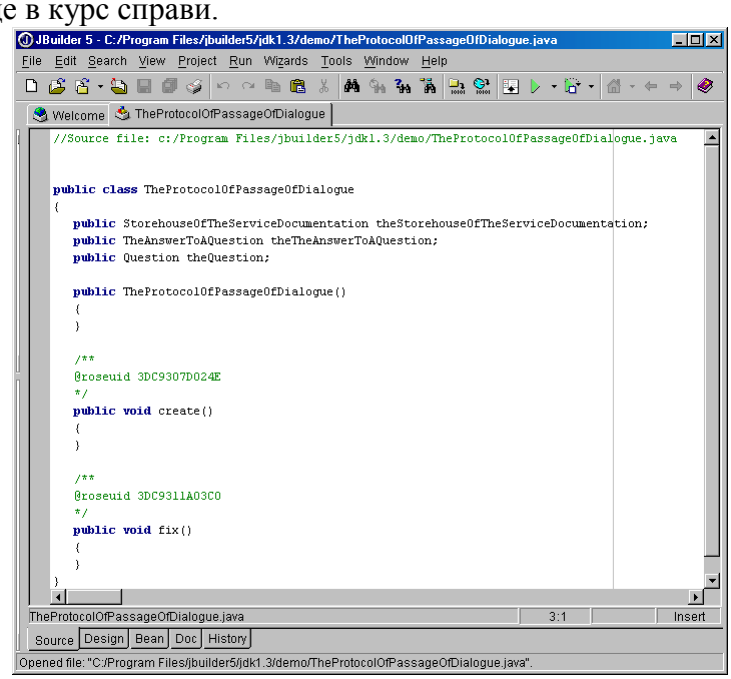

Рис. 8. Вміст файла TheProtocolOfPassageOfDialodue.java

Для проектувальника переваги використання RR не викликають сумнівів, а для розробника коду послуговування цим CASE-засобом дає серйозні плюси в роботі. Адже із проекту RR на основі набору діаграм створюють первинний код деякою мовою програмування. Відтак, зменшується доля „ручного" програмування, що здатне ховати у собі помилки і таке, що характеризується внесеними кожним розробником особистими перевагами в програму. Варто зазначити, що такий пакет зможе створити основу для майбутньої програмної системи, заготовки класів разом із їх взаємодією, а наповнювати методи змістом повинен все-таки програміст. Але поправивши що-небудь навіть у структурі класів, він завжди зможе отримати візуальне відображення цих змін у діаграмах.

Елементи КСМ-технології програмування можуть бути використані для розв'язання таких задач, які розбиваються на підзадачі, кожна з яких є окремим модулем деякої програмної системи. Прикладом такої задачі може бути задача про побудову графіків довільних функцій із елементарних функцій.

На сучасному етапі досліджуються питання створення програмного середовища для побудови довільних схем програм із аплетів мовою Java, що в результаті дозволить автоматизувати процес розробки дистанційних курсів, які включають розв'язання задач, що допускають розбиття на підзадачі.

The model of Flexible Distance Learning System is presented in this paper. This model is developed with the help of UML with the usage of CASE-tool named "Rational Rose". The opportunity to generate the program code on the basis the class diagram "Advise" is demonstrated. The advantages of software creation on the basis UML-model are described.

Література

1. Парасюк И.Н., Провотар А.И., Заложенкова И.А. О методологии композиционного структурно-модульного программирования//Кибернетика и системный анализ, 1995. — №1. — С.146-154.
2. Парасюк И.Н., Калита А.В., Провотар А.И. Case-система структурно-модульного композиционного программирования: концептуальные основы//Кибернетика и системный анализ, 1993. — №2. — С.140-144.
3. Трофимов С.А. CASE-технологии: практическая работа в Rational Rose. — М.: Бином-Пресс, 2002. — 288 с.
4. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. — М.: «Издательство Бином», СПб.: «Невский диалект», 1999. —560 с.
5. Волш А.И. Основы программирования на Java для World Wide Web. — К.: Диалектика, 1996. — 512 с.
6. Нотон П., Шилдт Г. Полный справочник по Java. — К.: Диалектика, 1997. — 592 с.

Одержано 25.11.2002 р.