ТЕОРІЯ І МЕТОДИ ПРОЕКТУВАННЯ СКЛАДНИХ СИСТЕМ

УДК 681.3.049

Д. Корпильов
Національний університет “Львівська політехніка”, кафедра САПР

МЕТОДОЛОГІЯ СТВОРЕННЯ ОБ’ЄКТНО-ОРІЄНТОВАНИХ ПРОГРАМНИХ СИСТЕМ ЗА ДОПОМОГОЮ МОВИ UML

Unified Modeling Language (UML) – це уніфікована мова моделювання для опису, візуалізації та документування об`єктно-орієнтованих систем в процесі їх розробки. UML призначена для підтримки процесу моделювання складних систем, що дозволяє на основі об`єктно-орієнтованої концепції організовувати взаємозв`язок концептуальних і програмних понять, що відображають проблеми масштабування складних систем. Мова UML визначає набір діаграм, що описують моделі та базову нотацію. В документації семантика подається як звичайною мовою, так і в нотації UML у вигляді метамоделі. Опис графічної нотації супроводжується прикладами діаграм. Моделювання складної системи засобами UML зводиться до її опису у різних проекціях. Кожна проекція описує визначений аспект системи, яка розробляється, а всі разом вони характеризують систему з відповідною повнотою, правильністю й адекватністю.

Unified Modeling Language (UML) is unified language for description, visualization and documentation of the object-oriented systems in the process of their development. UML assigned to support the process of the complex systems modeling. UML also assigned to organize interconnection of conceptual and program notions based on object-oriented paradigm which reflects problems of complex systems scaling. UML language determines a set of diagrams that describe. Semantics in documentation is showed in general language and in UML notation in metamodel view. Description of graphical notation is accompanied by diagram examples. Modeling of complex system using UML consists of its description in different projections. Every projection describes identified aspect of system being developed, and all together they determine system with corresponded measure of fullness, rightfulness and adequacy.

1. Вступ

Моделювання предметної області є одним із найважливіших етапів роботи при проектуванні програмних систем.

На даний час для цілей моделювання предметної області на ринку програмних продуктів використовують великий спектр Aided Software Engineering (CASE) – засобів. Найбільш популярними CASE – засобами є Rational Rose, BPwin, Silverrun, Process Analyst.

У моделюванні предметної області в цих засобах є багато спільного. Важливим також є комплексний підхід і використання єдиної уніфікованої нотації не тільки на етапі моделювання предметної області, але і на наступних етапах розробки програмної системи.

Моделювання предметної області з використанням уніфікованої нотації, що базується на викристанні Уніфікованої Мови Моделювання (УММ), гармонійно об'єднує переваги структурних та об'єктних методів проектування.

UML – це уніфікована мова моделювання для опису, візуалізації та документування об`єктно-орієнтованих систем в процесі їх розробки. UML призначена підтримувати процес моделювання складних систем на основі об`єктно-орієнтованої концепції, організовувати взаємозв`язок концептуальних і програмних понять, що відображають проблеми масштабування складних систем. Мова UML визначає набір діаграм, що описують моделі та базову нотацію. В документації семантика наведена як на звичайній мові, так і в нотації UML у вигляді метамоделі. Опис графічної нотації супроводжується прикладами діаграм. Моделювання складної системи засобами UML зводиться до її опису у різних проекціях. Кожна проекція описує визначений аспект системи, яка розробляється, а всі разом вони визначають систему з відповідним ступенем повноти, правильності й адекватності.

2. Моделювання програмних систем

Для успішної реалізації проекту об'єкт проектування (у нашому випадку ПЗ) повинен бути насамперед адекватно описаний, тобто має бути побудована повна і несуперечлива **архітектура** ПЗ. Архітектура ПЗ наводиться у вигляді сукупності моделей, що будуються для того, щоби зрозуміти й осмислити структуру й поведінку майбутньої системи, полегшити керування процесом її створення і зменшити можливий ризик, а також документувати прийняті проектні рішення. Розробка архітектури системи ПЗ промислового характеру на стадії, що передує її реалізації або відновленню, такою ж мірою необхідна, як і наявність проекту для будівництва великого будинку. Це твердження справедливе як у випадку розробки нової системи, так і при адаптації типових продуктів класу R/3 або BAAN, у складі яких також є власні засоби моделювання. Добрі моделі є основою взаємодії учасників проекту і гарантують коректність архітектури.

Очевидно, що основна мета розробки ПЗ – це не моделювання, а одержання працюючих додатків (коду). Діаграми у кінцевому рахунку – це усього лише наочні зображення. Тому при використанні графічних мов моделювання дуже важливо розуміти, чим це допоможе, коли справа дійде до написання коду. Можна навести такі причини, що спонукають їх використовувати:

- *вивчення методів проектування*. Безліч людей відзначає наявність серйозних труднощів, зв'язаних, наприклад, з освоєнням об'єктно-орієнтованих методів і, у першу чергу, зміну парадигми. Графічні засоби дозволяють полегшити вирішення цієї проблеми;
- *спілкування з експертами організації.* Графічні моделі дозволяють дати зовнішнє уявлення про систему і пояснюють, що ця система буде робити (рис. 2, 3);
- *одержання загального уявлення про систему.* Графічні моделі допомагають швидко одержати загальне уявлення про систему (рис. 4.), сказати про те, якого роду абстракції існують усередині системи і які її частини мають потребу в подальшому уточненні.

З усього сказаного вище можна зробити висновок, що моделювання складних програмних систем за допомогою CASE-засобів є самостійним і самодостатнім видом діяльності в процесі створення ПЗ.

Водночас практичне впровадження CASE-технології в організаціях, які розробляють ПЗ, зв'язано з деякими проблемами. Вони досить повно викладені в американському стандарті IEEE Std. 1348-1995. IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools. Ціль наведених у стандарті рекомендацій –підвищити імовірність успішного впровадження CASE-технології. Ці рекомендації досить актуальні і цінні, оскільки відбивають досвід, накопичений багатьма закордонними користувачами і розроблювачами CASE-засобів.

3. Застосування в життєвому циклі розробки програмних систем UML

Реально процес розробки складається з декількох стадій. Причому ці стадії існують у будь-яких моделях життєвого циклу проектів, вони лише відрізняються назвами й угрупованням дій (рис.1).

Мета стадії **аналізу вимог** полягає у тому, щоб зрозуміти процеси, які керують підприємством або системою, визначити сферу діяльності системи і вимоги користувача. Система розглядається з погляду кінцевого користувача як «чорна скриня», складається уявлення, що система буде робити, не розглядаючи, як вона це буде робити.

На даній стадії за допомогою UML створюється **модель прецедентів** системи. Вона дозволяє виділити зовнішні системи, що контактують із системою, основні процеси і їхній взаємозв'язок. Діаграми прецедентів дають змогу виділити функціональну структуру системи, не вдаючись у деталі її реалізації. Крім того, попередньо з'ясовуються та класифікуються об'єкти системи. На підставі побудованої моделі складається план розробки системи.

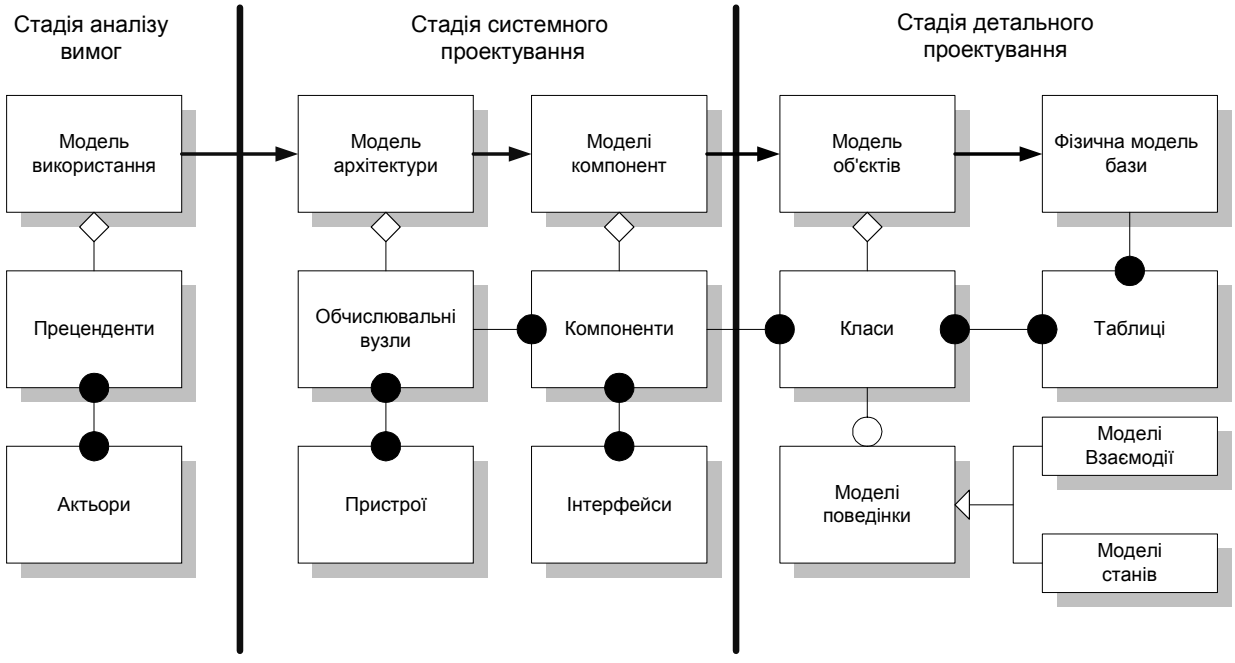

Рис. 1. Стадії розробки об'єктно-орієнтованих програмних систем

Стадія системного проектування містить рішення верхнього рівня щодо розробки системи загалом. Тут розробляється архітектура системи за допомогою **діаграми розгортання**. Вона дозволяє виділити обчислювальні ресурси, пристрої, які використовуються

ними, і з'єднання між ними, спроектувати розміщення окремих процесів і компонентів, що виконуються, на визначених пристроях, що особливо важливо при проектуванні складних систем і інтернет-додатків.

За допомогою **діаграми компонентів** робиться поділ програмної системи на компоненти, що виконуються. На підставі побудованих діаграм вибираються технологія і засоби розробки.

Протягом **стадії детального проектування** повинні бути описані способи розв'язання задач, виконуваних системою. Ця стадія цілком описує функції, класи системи і графічний інтерфейс із користувачем.

На даній стадії розробляються **діаграми класів**, включаючи відносини між класами і їхніми атрибутами, що дозволяє зробити класифікацію об'єктів, що функціонують у системі. Застосовуючи **діаграми поведінки (діаграми послідовності рис. 3, діаграми взаємодії, діаграми станів рис. 2. і діаграми активності)**, розробляється модель поведінки об'єктів у системі. Широкий набір засобів і методів дозволяють виділити ті аспекти поведінки об'єктів, що якнайкраще відтворюють їхні властивості.

За допомогою розробленої моделі поведінки встановлюються залежності між класами, виробляється поділ системи на модулі і виділення класів, реалізованих у даних модулях для того, щоб можна було ефективно організувати роботу команди розробників.

При розробці систем, що використовують реляційні бази даних, на підставі діаграми класів створюється фізична модель бази даних для збереження даних об'єктів постійних класів. Усі рішення, зв'язані з побудовою об'єктно-орієнтованої моделі програмної системи, тут повинні бути довершені.

Протягом **стадії реалізації** моделі, створені на стадіях проектування системи, переводяться у вихідний код 3GL або 4GL мов програмування і розробляється база даних системи.

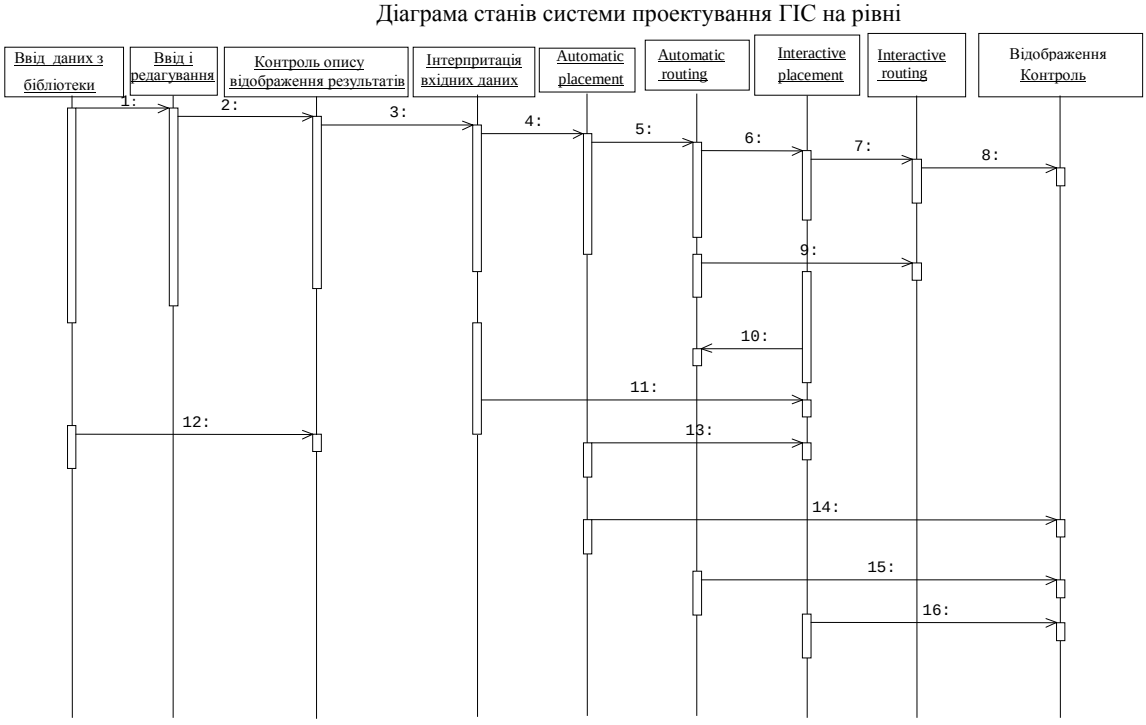

Рис. 2. Діаграма станів ситеми проектування ГІС

Рис. 3. Діаграма створення топології ГІС

Засоби автоматичної кодогенерації дозволяють перевести моделі мовою UML у вихідний код обраної мови програмування. CASE-засоби, що підтримують UML (Rational Rose, Paradigm Plus, Select Enterprise, Microsoft Visual Modeler for Visual Basic і ін.), дають можливість працювати з багатьма мовами програмування, такими як C++, Java, Delphi, Power Builder, Visual Basic, Centura, Forte, Ada, Smalltalk та ін., а також генерувати бази даних на більшості з існуючих SQL-серверів. Моделі, розроблені в UML, дозволяють значно спростити процес кодування і скерувати зусилля програмістів безпосередньо на реалізацію системи.

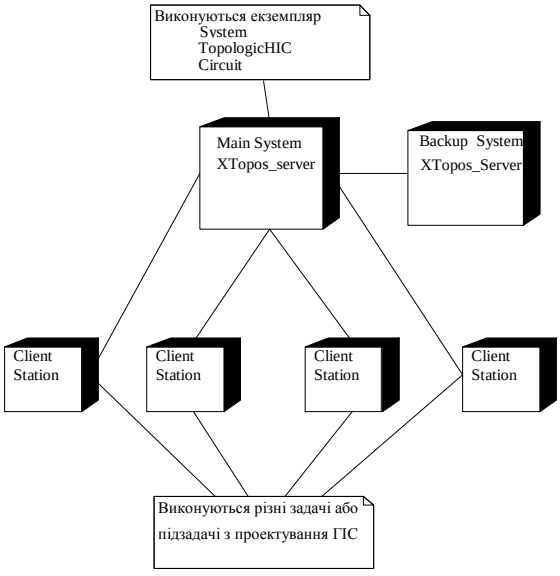

Рис. 4. Архітектура розподіленої системи проектування топології ГІС

На **стадії тестування** перевіряється, чи задовольняє система усі вимоги і робить очікувані дії. За допомогою моделі легко розробляються сценарії тестування модулів і програми загалом. Крім того, об'єктна модель системи дозволяє уникнути багатьох проблем несумісності окремих модулів додатка, що значно зменшить кількість помилок у програмі і полегшить роботу на даній стадії.

На **стадії впровадження** завершується розробка поточної версії системи, її установка, здійснюється технічна підтримка і навчання персоналу. Діаграми підвищують супровідність проекту і полегшують розробку документації.

Оскільки реальні системи часто змінюються з часом, то і до програмної системи висуваються нові вимоги. У результаті система **вимагає модифікації**. Об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни життєздатності системи. Використання побудованої моделі при модифікації системи дає змогу усунути небажані наслідки змін, оскільки вони не ламають стабільної структури системи, а тільки змінюють поведінку об'єктів.

При описі процесів повинні бути виявлені зв’язки між різними підсистемами системи при вирішенні конкретних задач конструювання.

Висновок

UML підтримує всі стадії життєвого циклу проекту, його застосування досить для повної підтримки розробки системи. Особливість UML у тім, що вона оптимізована для застосування при розробці програмних систем, що змогу максимально прискорити розробку програмних продуктів і помітно поліпшити якість системи, що розробляється.

Успіх розробки середовища САПР ГІС “TOPOS” із використанням мови UML залежить від методологічного і технологічного базисів, які застосовуються розробниками на етапах проектування й реалізації компонентів інформаційної системи. Важливим позитивним чинником стає наявність програмної інфраструктури як підмурівку, що істотно полегшує процес побудови гетерогенних розподілених інформаційних систем, а також побудова системи відповідно до міжнародних стандартів САПР.

1. Крег Л. Применение UML и шаблонов проектирования. – М., 2001. – 496 с. 2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированого проектирования. – СПб. 2001. – 368 с. 3. Object Management Group Web site: http://www.omg.org 4. Буч Г., Рамбо Д., Джекобсон А., Язык UML. Руководство пользователя. – М., 2000. – 432 с. 5. Г. Буч, Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд. – М., СПб, 1998.