повідомлення. Для формування списків факультетів, спеціальностей та груп здійснюється SQL-запит до БД шляхом 2 (рис. 2).

При здійснення вибору певного запису із переліку на екрані та натисненні кнопки «Далее» програма запам'ятовує вибір користувача, який є значенням відповідного параметру, а на останньому кроці 4 із значень параметрів формується URL-адреса для запиту до веб-сервера. При наявності доступу до веб-сервера йому надсилається запит. Веб-сторінка, що повертається від сервера, відображається компонентою TWebBrowser. Після цього програма очікує певний час та автоматично повертається у початковий стан (головне вікно терміналу).

Висновки. Особливістю запропонованої архітектури є те, що на веб-сервер покладена задача формування веб-сторінок за запитом, який містить певні параметри, що впливають на вміст сторінки. Запропонований підхід характеризується рядом переваг:

- Передбачається жорстка "траєкторія руху" користувача під час вибору значень параметрів, що мінімізує кількість неоднозначних та конфліктних ситуацій;
- На екрані відображається необхідна і достатня інформація для здійснення однозначного вибору на кожному кроці;
- Постійний моніторинг щодо виникнення критичних для роботи програми ситуацій (відсутність інтернет-звязку, доступу до БД) і у випадку необхідності – інформування користувачів;
- Можливість додавання додаткових можливостей терміналу за рахунок використання драйверів периферійного обладнання, API Windows;
- Можливість реалізації довільних часових затримок, повернення програми до первісного стану у випадку простою або відсутності дій користувача під час незавершеного вибору, обмеження часу перегляду розкладу для зменшення черги тощо.
- В наявності є можливості реалізації додаткового функціоналу, що розширюють задачі, які може вирішувати інформаційний термінал: відображення повідомлень від деканатів, об'яв, трансляція відео та аудіо в реальному часі тощо.

Література

1. Таненбаум Э. Распределенные системы. Принципы и парадигмы / Э.Таненбаум, М. Ван Стен. – СПб.: Питер, 2003г. – 877 с.

2. Архангельський А.Я. Программирование в Delphi 7 / А.Я. Архангельський. – М.: ООО "Бином-Пресс", 2003г. – 1152 с.

УДК 681.3

Жураковський Б.Ю., к.т.н.; **Варфоломеєва О.Г.,** к.т.н.; **Гладких О.В.,** асп.
(Державний університет інформаційно-комунікаційних технологій)
Хахлюк О.А. *(Алкатель-Луцент)*

ОБ'ЄКТНО-ОРІЄНТОВАНА ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ СИСТЕМ УПРАВЛІННЯ

Жураковський Б.Ю., Варфоломеєва О.Г., Гладких О.В., Хахлюк О.А. Об'єктно-орієнтована технологія проектування систем управління. Розглядається об'єктно-орієнтована технологія проектування систем управління мережею. Пропонується специфікація системи управління при об'єктному підході.

Ключові слова: ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ПРОГРАМУВАННЯ, ОБ'ЄКТНО-ОРІЄНТОВАНА ТЕХНОЛОГІЯ, СИСТЕМИ УПРАВЛІННЯ, МОДУЛЬНІСТЬ, ІНКАПСУЛЯЦІЯ

Жураковский Б.Ю., **Варфоломеєва О.Г., Гладких О.В., Хахлюк А.А. Объектно-ориентированная технология пректирования систем управления**. Рассматривается объектно-ориентированная технология

проектирования систем управления сетью. Предлагается спецификация проектируемой системы управления при объектном подходе.

Ключевые слова: ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА, ПРОГРАММИРОВАНИЕ, ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ ТЕХНОЛОГИЯ, СИСТЕМА УПРАВЛЕНИЯ, МОДУЛЬНОСТЬ, ИНКАПСУЛЯЦИЯ

Zhurakovskyi B.Iu., **Varfolomeieva O.H., Hladkykh O.V., Khakhliuk O.A**. **Object-oriented technology in a complex control systems.** The object-oriented technology of planning of network control system is examined. The specification of designed control system at objective approach is offered .

Keywords: COMPUTING ENGINEERING, PROGRAMMING, OBJECT-ORIENTED TECHNOLOGY, SYSTEM CONTROL, MODULARITY, ENCAPSULATION

Об'єктно-орієнтована технологія розвивається в різних областях обчислювальної техніки як засіб вирішення проблем, пов'язаних зі складністю створюваних систем. Об'єктний підхід застосовується не тільки в програмуванні, але також у проектуванні інтерфейсу користувача, баз даних, баз знань, комп'ютерної архітектури і навіть систем управління.

Сенс такого широкого підходу полягає в тому, що він дозволяє застосувати об'єктну орієнтацію для вирішення всього кола проблем, пов'язаних зі складними системами. Зважаючи, що система управління є складною системою, можемо застосувати об'єктно-орієнтовану технологію для дослідження і проектування таких систем. В основі об'єктно-орієнтованого проектування лежить уявлення про те, що систему необхідно проектувати як сукупність взаємодіючих один з одним об'єктів, розглядаючи кожен об'єкт як примірник певного класу, причому класи утворюють ієрархію [1].

Підвищення інтересу розробників до цієї методології обумовлено тим, що методи структурного аналізу і проектування не забезпечують подальшого зниження трудомісткості розробки. Об'єктно-орієнтований підхід найбільш природно відповідає реальному процесу розробки систем і не тільки програмних, який є ітеративним і може зажадати внести зміни до вже розроблені і налагоджені компоненти системи. Складовими частинами об'єктно-орієнтованої методології (ООМ) є: об'єктно-орієнтований *аналіз*; об'єктно-орієнтоване *проектування*; об'єктно-орієнтоване *програмування*.

Об'єктно-орієнтований аналіз. На об'єктний підхід вплинули попередні етапи розвитку програмних засобів. Традиційні прийоми структурного аналізу основані на потоках даних в системі. Об'єктно-орієнтований аналіз (ООА) спрямований на створення моделей, більш близьких до реальності, з використанням об'єктно-орієнтованого підходу. Це методологія, при якій вимоги формуються на основі понять класів та об'єктів, що становлять словник предметної області [2]. На результатах ООА формуються моделі, на яких ґрунтується об'єктно-орієнтоване проектування. Об'єктно-орієнтоване проектування у свою чергу створює основу для остаточної реалізації системи з використанням методології об'єктно-орієнтованого програмування.

Головними перевагами ООМ порівняно зі структурними методами є: *можливість* подолати обмеження, пов'язані зі складністю розроблювальних систем; *використання* на стадії аналізу моделей близьких до реальності; *застосування* при аналізі і проектуванні як інформаційних систем, так і систем реального часу і апаратно-програмних комплексів; *забезпечення* можливості повторного використання розробленого програмного забезпечення, що дозволяє істотно скоротити терміни і знизити витрати на розробку кожної наступної системи; *підтримка* ітеративного, а не лавиноподібного, як у структурному підході, процесу проектування; *природна* робота з різнорідною інформацією, що використовується в мультимедійних системах; *створення* більш відкритих систем; *повне* використання описових можливостей об'єктно-орієнтованих мов програмування.

Об'єктно-орієнтоване проектування. Об'єктно-орієнтоване проектування – це методологія проектування, що сполучає в собі процес об'єктної декомпозиції і прийоми подання як логічної і фізичної, так і статичної та динамічної моделей системи, що проектується [2]. У цьому визначенні містяться дві важливі частини:

1) об'єктно-орієнтоване проектування веде до об'єктно-орієнтованої декомпозиції;

2) використовується різноманіття прийомів подання моделей, що відображають логічну (структури класів і об'єктів) та фізичну (архітектура модулів і процесів) структуру системи.

Саме підтримка об'єктно-орієнтованої декомпозиції відрізняє об'єктно-орієнтоване проектування від структурного проектування.

Об'єктно-орієнтоване програмування. Об'єктно-орієнтоване програмування – це методологія програмування, яка основана на представленні програми у вигляді сукупності об'єктів, кожен з яких є реалізацією певного класу, а класи утворюють ієрархію на принципах спадкування [3].

У цьому визначенні можна виділити три частини: *1) об'єктно-орієнтоване* програмування використовує в якості елементів конструкції об'єкти, а не алгоритми; *2) кожен* об'єкт є реалізацією певного класу; *3) класи* організовані ієрархічно.

Принципи об'єктного підходу. Об'єктна модель, яка є концептуальною базою об'єктно-орієнтованої методології, має чотири головні елементи: *абстрагування*; *обмеження* доступу або інкапсуляція; *модульність*; *ієрархія*.

Без будь-якого з цих елементів модель не буде об'єктно-орієнтованою. Крім головних є три додаткові елементи: *типізація*; *паралелізм*; *зберігання* або стійкість (збереження).

Називаючи їх додатковими ми маємо на увазі, що вони корисні в об'єктній моделі, але не обов'язкові.

Апарат *абстракції* – зручний інструмент для боротьби зі складністю реальних систем. Створюючи поняття в інтересах будь-якого завдання, ми відволікаємося (абстрагуємося від несуттєвих характеристик конкретних об'єктів, визначаючи тільки істотні характеристики. Абстракція виділяє істотні характеристики деякого об'єкта, що відрізняють його від усіх інших видів об'єктів і, таким чином, чітко визначає його концептуальні кордони.

Інкапсуляція – це процес відділення один від одного елементів об'єкта, що визначають його будову і поведінку. Інкапсуляція служить для того, щоб ізолювати контрактні зобов'язання абстракції від їх реалізації.

Абстракція і інкапсуляція доповнюють один одного: абстрагування направлено на спостережуване поведінку об'єкта, а інкапсуляція займається внутрішньою будовою. Найчастіше інкапсуляція виконується за допомогою приховування інформації, тобто маскуванням всіх внутрішніх деталей, що не впливають на зовнішню поведінку. Зазвичай приховуються і внутрішня структура об'єкта, і реалізація його методів. *Інкапсуляція приховує деталі реалізації об'єкта.*

Модульність – це властивість системи, пов'язана з можливістю декомпозиції на ряд внутрішньо, але слабко пов'язаних між собою модулів.

У традиційному структурному проектуванні модульність – це мистецтво розкладати підпрограми по групах так, щоб в одну групу потрапляли підпрограми, які використовують одна одну або змінні разом. В об'єктно-орієнтованому програмуванні ситуація дещо інша: необхідно фізично розділити класи та об'єкти, які складають логічну структуру проекту. *Модульність дозволяє зберігати окремо абстракції.*

На основі наявного досвіду можна перерахувати прийоми і правила, які дозволяють складати модулі з класів і об'єктів найбільш ефективним чином. Кінцевою метою декомпозиції програми на модулі є зниження витрат на програмування за рахунок незалежної розробки і тестування. Структура модуля повинна бути досить *простою* для сприйняття; *реалізація* кожного модуля не повинна залежати від реалізації інших модулів; повинні бути вжиті *заходи* для полегшення процесу внесення змін там, де вони найбільш вірогідні. Прагматичні міркування ставлять межу цим керівним принципам.

На практиці перекомпіляція тіла модуля не є трудомісткою операцією: заново компілюється тільки даний модуль, і програма перекомпоновується. Перекомпіляція інтерфейсної частини модуля, навпаки, більш трудомістка. В строго типізованих мовах доводиться перекомпілювати інтерфейс і тіло самого зміненого модуля, потім всі модулі, пов'язані з даними, модулі, пов'язані з ними, і так далі по ланцюжку. У результаті для дуже

великих програм можуть знадобитися багато часу на перекомпіляцію (якщо тільки середовище розробки не підтримує фрагментарну компіляцію), що явно небажано. Тому слід прагнути побудувати модулі так, щоб об'єднати логічно пов'язані абстракції і мінімізувати взаємні зв'язки між модулями.

В процесі розділення системи на модулі можуть бути корисними два правила. *По-перше,* оскільки модулі служать в якості елементарних і неподільних блоків програми, які можуть використовуватися в системі повторно, розподіл класів і об'єктів по модулям має враховувати це. *По-друге,* багато компіляторів створюють окремий сегмент коду для кожного модуля. Тому можуть з'явитися обмеження на розмір модуля.

На вибір розбиття на модулі можуть впливати і деякі зовнішні обставини. При колективній розробці програм розподіл роботи здійснюється, як правило, за модульним принципом і правильний розподіл проекту мінімізує зв'язки між учасниками. При цьому більш досвідчені програмісти зазвичай відповідають за інтерфейс модулів, а менш досвідчені – за реалізацію. На більш високому рівні такі ж співвідношення справедливі для відносин між субпідрядниками. Абстракції можна розподілити так, щоб швидко встановити інтерфейси модулів за угодою між компаніями, які беруть участь в роботі.

Що стосується проекту документування, то він будується, як правило, також за модульним принципом – модуль є одиницею опису та адміністрування. Десять модулів замість одного зажадають в десять разів більше описів, і тому, на жаль, іноді вимоги щодо документування впливають на декомпозицію проекту (в більшості випадків негативно). Можуть позначатися і вимоги секретності: частина коду може бути несекретною, а інша – секретною; остання тоді виконується у вигляді окремого модуля (модулів).

Звести воєдино настільки суперечливі вимоги досить важко, але головне з'ясувати: виокремлення класів і об'єктів у проекті та організація модульної структури – незалежні дії. Процес виділення класів і об'єктів становить частину процесу логічного проектування системи, а поділ на модулі – етап фізичного проектування. Зрозуміло, іноді неможливо завершити логічне проектування системи, не завершивши фізичне проектування, і навпаки. Ці два процеси виконуються паралельно.

Абстракція – річ корисна, але завжди, крім простих ситуацій, число абстракцій в системі набагато перевищує наші розумові можливості. ***Інкапсуляція*** дозволяє певною мірою усунути цю перешкоду, прибравши з поля зору внутрішній зміст абстракцій. ***Модульність*** також спрощує завдання, поєднуючи логічно пов'язані абстракції в групи. Але цього виявляється недостатньо.

Значне спрощення розуміння складних завдань досягається за рахунок утворення з абстракцій ієрархічної структури. Визначимо ієрархію наступним чином:

Ієрархія – це упорядкування абстракцій, розташування їх по рівнях.

Моделі системи управління (СУ), що розробляються при об'єктному підході, основані на предметах і явищах реального світу. В основі цих моделей також лежить опис необхідної поведінки СУ, що розроблюється, тобто її функціональності, але ця поведінка пов'язана з станами елементів (об'єктів) конкретної предметної області. Таким чином, на етапі аналізу ставляться дві задачі: *уточнити* необхідну поведінку СУ, що розробляється; *розробити* концептуальну модель її предметної області з точки зору поставлених завдань.

Модель використання являє собою опис функціональності СУ з точки зору користувача (рис.1). Логічна модель описує ключові абстракції СУ (класи, інтерфейси і т.п.), тобто кошти, що забезпечують необхідну функціональність.

Модель реалізації визначає організацію програмних модулів в середовищі розробки.

Модель процесів відображає організацію обчислень і оперує поняттями «*процеси*» і «*з'єднання*». Вона дозволяє оцінити продуктивність, масштабованість і надійність.

Модель розгортання показує особливості розміщення програмних компонентів на конкретному обладнанні.

Таким чином, кожна з вказаних моделей характеризує певний аспект проектованої системи, а всі разом вони складають відносно повну модель СУ, що розробляється.

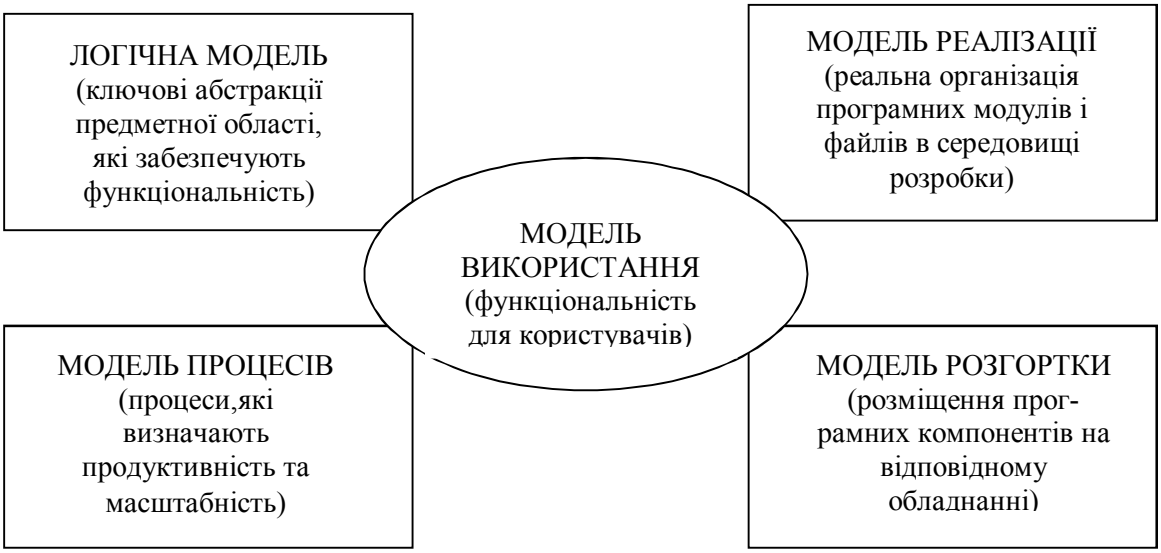

Рис.1. Повна специфікація СУ, що розробляється, при об’єктному підході

Висновки. Об'єктно-орієнтована технологія дозволяє застосувати об'єктну орієнтацію для вирішення всього кола проблем, пов'язаних зі системами управління. В основі об'єктно-орієнтованого проектування лежить уявлення про те, що систему управління необхідно проектувати як сукупність взаємодіючих один з одним об'єктів. Складовою частиною об'єктно-орієнтованої методології є об'єктно-орієнтований аналіз, який спрямований на створення моделей, більш близьких до реальності, з використанням об'єктно-орієнтованого підходу. В статті запропонована повна специфікація систем управління, яка складається з декількох моделей, що характеризують всі аспекти системи при проектуванні.

Література

1. Буч Г. Объектно-ориентированное проектирование с примерами применения : пер. с англ. / Г. Буч. – М.: Конкорд, 1992. – 519 с.

2. Молчанов А.А. Моделирование и проектирование сложных систем / А.А. Молчанов. – К.: Вища школа, 1988. – 359 с.

3. Шлеер С. Объектно-ориентированный анализ: моделирование мира в состояниях : пер. с англ. / С. Шлеер, С. Меллор. – К.: Діалектика, 1993. – 240 с.

УДК 004.713

Рябцов А.В., к.т.н. *(Одесская государственная академия холода)*

СПЕЦИАЛЬНЫЕ ТИПЫ ПЬЕЗОЭЛЕКТРИЧЕСКИХ АКТУАТОРОВ ДЛЯ ОПТИЧЕСКИХ КОММУТАЦИОННЫХ УСТРОЙСТВ

Рябцов О.В. Спеціальні типи п'єзоелектричних актуаторів для оптичних комутаційних пристроїв. Стаття присвячена застосуванню спіральних п'єзоелектричних елементів в якості актуаторів кутового переміщення дзеркал для комутації сигналів в сучасних повністю оптичних мережах. Показано, що застосування спіральних актуаторів дозволяє підвищити ефективність мікродзеркальних відхилюючих систем для оптичних комутаційних пристроїв.

Ключові слова: ОПТИЧНА МЕРЕЖА, ОПТИЧНА КОМУТАЦІЯ, СПІРАЛЬНИЙ П'ЄЗОЕЛЕКТРИЧНИЙ АКТУАТОР, МІКРОДЗЕРКАЛЬНА ВІДХИЛЯЮЧА СИСТЕМА

Рябцов А.В. Специальные типы пьезоэлектрических актуаторов для оптических коммутационных устройств. Статья посвящена применению спиральных пьезоэлектрических элементов в качестве актуаторов углового перемещения зеркал для коммутации сигналов в современных полностью оптических сетях. Показано, что применение спиральных актуаторов позволяет увеличить эффективность микрозеркальных отклоняющих систем для оптических коммутационных устройств.