І. ПРОБЛЕМИ МЕТОДИКИ НАВЧАННЯ МАТЕМАТИЧНИХ ДИСЦИПЛІН

УДК 371.315

О.Ф. Баранюк

Кіровоградський державний педагогічний університет імені Володимира Винниченка

СИСТЕМАТИКА ШАБЛОНІВ У КОНТЕКСТІ ОБ'ЄКТНИХ ТЕХНОЛОГІЙ

Сучасні програмні системи відносяться до складних систем, одним із способів подолання складності є декомпозиція, причому переваги має об'єктна декомпозиція. Об'єктні технології залишаються складними не тільки при розробці реальних програмних систем, але й при вивченні їх студентами. На допомогу новачкам у програмуванні приходять шаблони, які акумулюють кращий досвід об'єктно-орієнтованої розробки програм.

Оскільки кількість шаблонів обчислюється сотнями, а єдиної і повної класифікації шаблонів не існує, в роботі здійснена спроба систематизації шаблонів за фазами розробки програмних систем та рівнями абстракції. При вивченні курсу «Проектування програмних систем» пропонується націлювати студентів на використання представленої систематизації шаблонів з метою полегшення пошуку шаблонів, потрібних для реалізації навчальних завдань і проектів.

Ключові слова: *об'єктна технологія, проектування програмних систем, об'єктно-орієнтоване програмування, шаблон, шаблон проектування, навчання на основі шаблонів.*

Постановка проблеми. Для розробки програм у промислових масштабах потрібні високоякісні фахівці, озброєні знаннями мов програмування, методологіями проектування і технологіями розробки програмного забезпечення. Випускники вищих навчальних закладів не завжди відповідають вимогам реального виробництва програм, найчастіше їм бракує практичних навичок розробки програмного забезпечення.

Вищі навчальні заклади змушені реагувати на виклики сьогодення і вдосконалювати навчальні плани підготовки фахівців. Керівництво з розробки навчальних планів в галузі комп'ютерних наук (Computer Science Curricula 2013) наголошує на необхідності зміщення акцентів від основ програмування (Programming Fundamentals) до основ розробки програм (Software Development Fundamentals. Потрібно зосередитися на цілісному процесі створення програм, включаючи аналіз і розробку алгоритмів, базові концепції програмування і структури даних, основні методи та засоби розробки і тестування програм.

Аби студенти були готові до розробки реальних програмних проектів, під час навчання вони мають ознайомитися з кращими практиками розробки програм – модульним тестуванням, системами контролю версій, промисловими інтегрованими середовищами розробки (IDE), шаблонами проектування. Це дасть можливість студентам своєчасно усвідомити виклики, пов'язані з розробкою реальних програмних проектів.

Керівництво з розробки навчальних планів в галузі програмної інженерії (Software Engineering 2014) серед основних принципів підготовки фахівців відзначає *технічну обізнаність* – здатність виявляти розуміння і застосовувати відповідні теорії, моделі, методики, які становлять основу для аналізу проблем, проектування, розробки, впровадження, верифікації та документування програм, та *професійну обізнаність* – володіння знаннями і навичками програмної інженерії та професійних стандартів,

необхідних для початку роботи як інженера-програміста.

Сучасні програмні системи належать до складних систем, їх складність перевищує інтелектуальні здібності однієї людини, тому розробка програмного забезпечення здійснюється колективно на основі добре продуманих методологій і промислових технологій. Принцип боротьби зі складністю давно відомий – декомпозиція, причому для складних програмних систем найкраще себе зарекомендувала об'єктно-орієнтована методологія побудови систем.

Один із способів озброїти студентів кращими зразками проектування та програмування систем – ознайомити їх із шаблонами, які можна використовувати на різних етапах життєвого циклу розробки систем.

Аналіз останніх досліджень і публікацій. Питанням підвищення якості підготовки майбутніх фахівців у галузі програмування присвячено роботи багатьох вітчизняних і зарубіжних вчених, серед них заслуговують на увагу дослідження з теми Гришко Л.В., Жалдака М.И., Ільясової Ф.С., Морзе Н.В. Сейдаметової З.С., Семерікова С.О., Caspersen M.E., Haberman B., Muller O., Wallingford E., Winslow L. та багатьох інших.

Зокрема, дослідження Л.В. Гришко присвячені методичним основам підготовки фахівців з програмування, проблемам формування професіоналізму інженерів-програмістів, навчанню основ програмування у вищій школі, використанню засобів навчання програмуванню та іншим питанням.

Питанням викладання основ програмування у вищій школі, використанню хмарних сервісів у навчанні програмуванню, формуванню професійних навичок у програмуванні, методичним основам рівневої підготовки інженерів-програмістів та багатьом іншим питанням підготовки програмістів у вищій школі присвячено дослідження З.С. Сейдаметової.

Вивчення об'єктних технологій вимагає розвиненого абстрактного мислення студентів, особливо, коли об'єктно-орієнтоване програмування запроваджується як вступний курс до програмування (CS1). Виникає конфлікт між високорівневими абстракціями, необхідними для проектування об'єктно-орієнтованого рішення, і низькорівневими деталями програмної реалізації обраного рішення. Є. Волінгфорд пропонує організувати вивчення основних понять об'єктно-орієнтованого програмування на основі шаблонів [12]. Шаблони дають можливість прокласти місток від заданого типу проблеми до ефективного набору класів і його ефективної реалізації у програмному коді.

В роботі [7] наголошується, що підручники з програмування зазвичай організовані навколо лексичних конструкцій певної мови програмування, в результаті студенти зосереджуються на вивченні мови програмування, але по закінченню курсу виявляються не здатними написати більш-менш пристойну реальну програму. Тому дослідники пропонують організувати навчання з використанням шаблонів, заснованих на проблемних ситуаціях, а не на програмних конструкціях. Наголошується, що студенти повинні виробляти навички розпізнавання потрібних шаблонів у проблемних ситуаціях та навички модифікації шаблонів під потреби конкретної задачі.

Розвитку навичок об'єктного мислення студентів-початківців присвячені дослідження [4], які пропонують під час навчання зосередити увагу студентів на процесі розробки програмного забезпечення від аналізу предметної області і виявлення концептуальних класів до програмної реалізації конкретних класів і надають для цього шаблон і чіткі покрокові інструкції з його застосування.

Група ізраїльських вчених [10], яка досліджує проблеми навчання програмуванню на основі освітніх шаблонів, пропонує будувати процес навчання та розвивати здібності студентів до розв'язування задач на базі шаблонів, дає вказівки щодо складання набору необхідних шаблонів та приклади задач, розв'язування яких може бути полегшене на основі шаблонів.

Класичною роботою з використання шаблонів для побудови об'єктно-орієнтованих систем залишається фундаментальна книга «банди чотирьох» (Еріх Гамма, Річард Хелм, Ральф Джонсон та Джон Вліссідес) «Шаблони проектування: Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення» (Design Patterns: Elements of Reusable Object-Oriented Software) [1]. Ця книга дає поняття шаблону проектування, наводить структуру опису шаблонів, містить каталог із 23 шаблонів проектування у трьох категоріях (твірні, структурні, поведінкові), детально описує ці шаблони, наводить приклади реалізації і поради з використання шаблонів. Фактично з цієї книги почалося масове використання шаблонів у програмуванні.

Отже, програмістами-професіоналами, науковцями, і викладачами проведена значна робота з виявлення, опису і каталогізації шаблонів, пов'язаних з використанням об'єктних технологій, хоча більшість наукових досліджень висвітлюють лише окремі аспекти їх застосування. Поки що відсутня цілісна картина застосування шаблонів при підготовці майбутніх фахівців з програмування.

Метою даної статі є спроба систематизації та визначення сфери застосування шаблонів при викладанні дисциплін, пов'язаних з вивченням об'єктних технологій.

Виклад основного матеріалу дослідження. Вивчення предметів, пов'язаних з використанням об'єктних технологій, викликає значні труднощі у студентів. Об'єктно-орієнтоване проектування та програмування вимагає неабияких здібностей різнорівневого абстрактного мислення. Об'єктні технології важко даються студентам, у першу чергу через те, що їм бракує практичного досвіду.

Допомогти студентам засвоїти об'єктно-орієнтовану методологію розробки програм можуть шаблони. Останнім часом навчання на основі шаблонів (Pattern-Based Instruction, Pattern-Oriented Instruction) набуло значного поширення [7, 8, 10, 12]. Шаблони дають можливість економити час і зусилля під час розробки програмного забезпечення, більше уваги приділяти високорівневому вирішенню проблеми, використовувати кращі зразки проектних рішень і програмного коду від професіоналів у галузі розробки програмного забезпечення.

Шаблон – це зразок, придатний для наслідування. Батьком шаблонів вважають Крістофера Александера, який розробив і описав значну кількість архітектурних шаблонів для будівництва та став одним із засновників «мови шаблонів» [6]. Шаблон у програмуванні це спроба описати успішне рішення повторюваної проблеми. Шаблони дають можливість використовувати колективний досвід професійних програмістів. Програмні шаблони спеціально не придумують, їх помічають у попередніх реалізованих програмах, описують, поміщають в каталоги і використовують в наступних проектах. «Шаблони документують існуючий, добре випробуваний досвід проектування. Вони не винаходяться, не створюються штучно» [3]. «Шаблони виводяться з практичного досвіду реальних проектів» [9].

Шаблони покликані поширювати фахові знання, завдяки чому вони є цінними для студентів та новачків у програмуванні. Шаблони можна використовувати на різних етапах життєвого циклу розробки програмних систем. Проблема лише в тому, що чим більше

шаблонів, тим важче їх знаходити і підбирати. Якщо пошук потрібного шаблону вимагає перегляду сотень описів в каталогах, то система шаблонів є малоефективною. Постає питання розумної кількості шаблонів, їх класифікації та систематизації.

Єдиної і усталеної системи класифікації шаблонів не існує. Різні автори у своїх дослідженнях, присвячених певній групі шаблонів, так чи інакше торкалися питання їх класифікації. Так, автор книги [3], присвяченої архітектурі програмних систем, говорить, що схема класифікації шаблонів, пов'язаних з розробкою програмних систем, повинна мати такі властивості: бути простою і легкою для вивчення і використання; складатися лише з кількох ознак класифікації; кожна ознака класифікації повинна відображати природні властивості шаблону (наприклад, тип проблеми), а не штучні критерії; вести користувача до потенціальної підмножини потрібних шаблонів; бути відкритою до інтеграції нових шаблонів. Отже, вимоги до класифікації шаблонів не можуть бути простими і несуперечливими. Проте, Ф. Бушманн [3] пропонує здійснювати класифікацію шаблонів лише за двома ознаками: категоріями шаблонів та категоріями проблем.

Категорії шаблонів. За категоріями шаблонів пропонується виділяти архітектурні шаблони, шаблони проектування та ідіоми [3].

Архітектурні шаблони формують базову структурну організаційну схему для програмних систем на початку проектування. Вони надають набір наперед визначених підсистем, визначають їх обов'язки, включають правила організації зв'язків між ними.

Шаблони проектування надають схеми для уточнення компонентів програмної системи та зв'язків між ними. Вони описують часто повторювані структури взаємодіючих компонентів, що вирішують загальні проблеми проектування у заданому контексті.

Ідіоми це низькорівневі зразки, специфічні для мови програмування. Ідіоми описують як реалізувати певні аспекти компонентів або відношення між ними, використовуючи особливості заданої мови.

Категорії проблем. Кожний шаблон пов'язаний із певною проблемою, яка повстає під час розробки програмних систем. Категорії проблем стають корисними для підбору шаблонів, призначених для вирішення конкретних проблем. Книга [3] пропонує такі категорії проблем і відповідні їм групи архітектурних шаблонів: *від безладу до структури* (шаблони декомпозиції систем); *розподілені системи*; *інтерактивні системи*; *адаптивні системи*; *структурна декомпозиція*; *організація роботи*; *контроль доступу*; *менеджмент* (керування компонентами); *комунікація*; *керування ресурсами*.

Автори роботи [11] пропонують виділяти шаблони аналізу, проектування і реалізації, вводячи при цьому поняття *концептуальний шаблон* (виражається в термінах предметної області), *шаблон проектування* (виражається категоріями проектування, такими як об'єкти, класи, наслідування, агрегація тощо) та *програмний шаблон* (виражається конструкціями мови програмування).

Крім численних шаблонів, що мають безпосереднє відношення до фаз проектування та програмування, є також цілий ряд інших категорій шаблонів. Відомий фахівець у галузі розробки об'єктно-орієнтованих систем Мартін Фаулер описує *шаблони аналізу* [9], які охоплюють концептуальні моделі предметної області і можуть бути повторно використані при аналізі інших систем. Шаблони аналізу це група концепцій, які представляють загальні конструкції бізнес-моделювання. Вони можуть бути специфічні для певної предметної області або охоплювати кілька областей.

Відомий дослідник і автор книг Джеймс Коплієн описує чотири групи *організаційних*

шаблонів [5], які документують загальні методики керування або структури організацій: 1) Керування проектами (Project Management) – організаційні аспекти керування проектами; 2) Поступове зростання організації (Piecemeal Growth of the Organization) – ріст і розвиток організації в часі; 3) Стиль організації (Organizational Style) – способи роботи організації; 4) Люди і код (People and Code) – вплив людей на код і навпаки. Організаційні шаблони безпосередньо не стосуються розробки програм, їх можна визначити як шаблони підтримки.

Скотт Амблер [2] визначає *шаблони процесів* як колекцію загальних методик, дій, задач для розробки об'єктно-орієнтованого програмного забезпечення. Вони описують, що слід робити, але не дають подробиць того, як це слід виконати. Амблер виділяє три типи шаблонів процесів: 1) *Шаблони задач* (Task process patterns) – описують детальні кроки виконання певних задач; 2) *Шаблони етапів* (Stage process patterns) – описують кроки одного етапу, які зазвичай виконуються ітеративно, можуть включати шаблони задач; 3) *Шаблони фаз* (Phase process patterns) – описують взаємодію етапів у межах фази життєвого циклу розробки систем. Шаблони фаз застосовуються послідовно і складаються з шаблонів етапів, які виконуються ітеративно.

Найбільш повну класифікацію шаблонів, які мають відношення до розробки програмних систем, можна знайти в роботі Владана Деведжіча [6], де виділяються і коротко характеризуються шаблони проектування (твірні, структурні, поведінкові), ідіоми, архітектурні шаблони, шаблони аналізу, організаційні шаблони та шаблони процесів, наводиться структура опису шаблонів різних груп. Згадуються також менш популярні шаблони: шаблони керування ризиками, компонентно-орієнтовані шаблони, шаблони соціальних сил, шаблони предметної області, антишаблони та деякі інші.

Пропонується систематизувати існуючі категорії шаблонів навколо фаз життєвого циклу розробки програмних систем та рівнів абстракції програмної архітектури. Систематизація переконує, що існують категорії шаблонів, тісно пов'язані з етапами життєвого циклу розробки програмних систем (шаблони аналізу, проектування, програмування), які утворюють умовний горизонтальний (послідовний) зріз схеми. Вертикальний зріз представляє категорії, пов'язані з рівнями архітектури програмної системи (архітектурні шаблони, шаблони проектування, програмні шаблони).

Виявлено також категорії шаблонів, які стоять осібно від загальної схеми. Це організаційні шаблони та шаблони процесів, які використовуються на різних етапах життєвого циклу розробки. Це так звані фазонезалежні та мультифазові шаблони, що потребують додаткової класифікації з метою приведення до запропонованої схеми.

Таблиця 1

Категорії шаблонів	Аналіз вимог	Проектування	Реалізація
Шаблони аналізу	+		
Архітектурні шаблони		+	
Шаблони проектування		+	
Програмні шаблони			+
Організаційні шаблони	+	+	+
Шаблони процесів	+	+	+

Примітка. Знаком «+» в таблиці позначено фази переважного використання шаблонів.

Як бачимо, за час існування об'єктних технологій напрацьована велика кількість шаблонів у різних категоріях. Кількість шаблонів сьогодні вимірюється сотнями. Шаблони виявлялися, описувалися і каталогізувалися професіоналами з розробки програмних систем у першу чергу для того, аби вони могли бути використані іншими професіоналами. З цієї причини шаблони не так легко використовувати новачкам, зокрема студентам. Існують навіть шаблони з використання шаблонів. Це ускладнює навчальний процес.

З метою полегшення вивчення студентами основ об'єктних технологій групою вчених з різних університетів світу був започаткований проект «Педагогічні шаблони» (The Pedagogical Patterns Project) [8]. Педагогічні шаблони акумулюють успішні практики викладання об'єктних технологій, описують їх в компактній формі з метою поширення серед спільноти викладачів інших навчальних закладів. На сьогодні проект зібрав і опублікував понад сімдесят педагогічних шаблонів, спрямованих на вирішення таких проблем як вибір тем і визначення їх порядку при складанні навчальної програми дисципліни, мотивація студентів, прийоми навчання студентів, методика проведення занять, оцінка діяльності студентів та багатьох інших. Кожний педагогічний шаблон описується за схемою: проблема, контекст, рушійні сили, рішення, обговорення і наслідки, спеціальні ресурси, пов'язані шаблони, приклади, протипоказання.

Пропонується при вивченні курсу «Проектування програмних систем» націлювати студентів на використання представленої систематизації шаблонів з метою полегшення пошуку та підбору шаблонів, потрібних для реалізації навчальних завдань і проектів.

Висновки. Об'єктно-орієнтоване проектування з'явилося у відповідь на виклики, пов'язані зі все зростаючою складністю програмних систем. При розробці складних систем об'єктна декомпозиція значно виграє у порівнянні з алгоритмічною. Сучасні програмні системи складні не тільки для розробки, а й для вивчення, тому студенти відчувають значні труднощі при вивченні об'єктних технологій. Один із можливих виходів із ситуації – використання шаблонів, в яких накопичено досвід професійних програмістів та розробників програмних систем. Набір шаблонів для різних аспектів розробки програмного забезпечення досить об'ємний, тому виникає проблема ідентифікації потрібних шаблонів і адекватного підбору шаблонів для вирішення конкретних питань проектування чи програмування. Допомогти у вирішенні цієї проблеми може проста і зрозуміла класифікація, яка дасть можливість студентам у відповідності з етапом розробки, рівнем абстракції і типом проблеми підібрати необхідний шаблон.

У подальшому потрібно зосередитися на формуванні навчального набору шаблонів різних категорій для різних фаз розробки та видів навчальної діяльності, які передбачається використовувати у навчальних завданнях і студентських проектах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб. : Питер, 2001. – 366 с.

2. Ambler S. W. An Introduction to Process Patterns / Scott W. Ambler // Process Patterns: Building Large-Scale Systems Using Object Technology. – Cambridge : SIGS Books/Cambridge University Press, 1998. – 582 p.

3. Buschmann F. Pattern-Oriented Software Architecture - Volume 1: A System of Patterns // Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. – Chichester : John Wiley & Sons, 1996. – 476 p.

4. Caspersen, M. E. Novice's Process of Object-Oriented Programming / M. E. Caspersen, M.A. Kölling // Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming

Systems, Languages, and Applications (OOPSLA). – Portland : ACM, 22–26 October 2006. – pp. 892-900.

5. Coplien J. Organizational Patterns of Agile Software Development / James O. Coplien, Neil Harrison. – Pearson Prentice Hall, 2005. – 401 pp.

6. Devedzic V. Software Patterns / V. Devedzic // Chang, S.K. Handbook of Software Engineering and Knowledge Engineering. – Singapore : World Scientific Publishing Co., 2002. – pp. 645–671.

7. East, J.P. Pattern-based programming instruction / J.P. East, S. R. Thomas, E. Wallingford, W. Beck, J. Drake // Proceedings of the ASEE Annual Conference and Exposition. – Washington, 1996. – Режим доступу: http://www.cs.uni.edu/~wallingf/patterns/pa-pers/asee96.pdf

8. Eckstein J. Pedagogical patterns: capturing best practice in teaching object technology / J. Eckstein, M. L. Manns, M. Voelter // Software Focus. – John Wiley & Sons, Ltd., 2001. – pp. 9-12.

9. Fowler M. Analysis Patterns: Reusable Object Models / Martin Fowler. – Boston : Addison-Wesley Professional, 1997. – 357 pp.

10. Muller O. (An almost) pedagogical pattern for pattern-based problem-solving instruction / O. Muller, B. Haberman, H. Averbuch // ACM SIGCSE Bulletin. – 2004. – Vol. 36. – No. 3. – pp. 102–106.

11. Riehle D. Understanding and Using Patterns in Software Development / D. Riehle, H. Züllighoven // Theory and Practice of Object Systems. – 1996. – Vol. 2. – pp.3–13.

12. Wallingford E. Toward a first course based on object-oriented patterns / E. Wallingford // Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education. – Philadelphia, 1996. – P. 27–31. – Режим доступу: http://jon.crazybaglok.com/strat/intro_with_ patterns.pdf

Baranyuk O.F.

Kirovohrad Volodymyr Vynnychenko State Pedagogical University

SYSTEMATICS OF PATTERNS IN THE CONTEXT OF OBJECT TECHNOLOGIES

Contemporary industrial software systems belong to complex systems. The complexity of such systems exceeds the human intellectual capacity. Essential property of all large systems is that we can only master their complexity, but never eliminate it. Using patterns is an effective way to master complexity and transfer knowledge of professionals to novices, including students, because patterns capture chunks of professional knowledge. It's important that software developers do not invent patterns; they discover and describe patterns from experience in developing real systems.

There is large number of patterns in different categories but unique whole and simple classification of them was not yet proposed. Analysis patterns, architectural patterns, design patterns, programming patterns, organizational patterns, and process patterns are the most popular categories of patterns among developers. It is not so simple to find and select appropriate pattern for novices.

One of potential attempts for patterns systematization was made in this paper. Our systematization is based on the key principle: patterns should be organized around phases of software development and levels of abstraction. Our research shows that some pattern categories are related to software development phases and another are related to different levels of system architecture. Some standalone pattern categories are phase-independent or multi-phased. We hope that proposed categories systematization will help students to find appropriate pattern quick and easy.

Key words: *object technology, software systems design, object-oriented programming, pattern, design pattern, pattern-based instruction*.

Баранюк А.Ф.

Кировоградский государственный педагогический университет имени Владимира Винниченко

СИСТЕМАТИКА ШАБЛОНОВ В КОНТЕКСТЕ ОБЪЕКТНЫХ ТЕХНОЛОГИЙ

Современные программные системы принадлежат к сложным системам, одним из способов преодоления сложности выступает декомпозиция, причем преимущества имеет объектная декомпозиция. Объектные технологии остаются сложными не только при разработке реальных программных систем, но и при изучении их студентами. На помощь новичкам в программировании приходят шаблоны, которые аккумулируют лучший опыт объектно-ориентированной разработки программ.

Поскольку количество шаблонов исчисляется сотнями, а единой и полной классификации шаблонов не существует, в работе предпринята попытка систематизации шаблонов по фазам разработки программных систем и уровням абстракции. При изучении курса «Проектирование программных систем» предлагается нацеливать студентов на использование представленной

систематизации шаблонов с целью облегчения поиска шаблонов, необходимых для реализации учебных заданий и проектов.

Ключевые слова: *объектная технология, проектирование программных систем, объектно-ориентированное программирование, шаблон, шаблон проектирования, обучение на основе шаблонов.*

ВІДОМОСТІ ПРО АВТОРА

Баранюк Олександр Филимонович – доцент кафедри інформатики Кіровоградського державного педагогічного університету імені Володимира Винниченка, кандидат технічних наук.

Коло наукових інтересів: моделювання інформаційних систем, проблеми викладання комп'ютерних наук у вищій школі.

УДК 378.147.88

Д.Є. Бобилєв

Криворізький державний педагогічний університет

МЕТОД ПРОЕКТІВ ПРИ НАВЧАННІ ФУНКЦІОНАЛЬНОМУ АНАЛІЗУ МАЙБУТНІХ УЧИТЕЛІВ МАТЕМАТИКИ

Розглянуто застосування деяких методів, форм і засобів навчання функціонального аналізу при використанні проектної методичної системи. Запропоновано методичні рекомендації по організації етапів роботи над проектом, з'ясовано вплив методичної системи на формування професійно важливих якостей майбутніх учителів математики. Показано, що методика навчання функціонального аналізу, яка базується на застосуванні методу проектів дозволяє, розробити педагогічну технологію, що забезпечує зростання пізнавальних потреб студентів і підвищення ефективності процесу навчання. Розглянуто семестровий навчальний проект, що складається з міні-, локальних проектів, спрямований на виявлення властивостей оберненого функціоналу з питань, не відображених у літературі, на визначення зв'язків між вихідним і оберненим функціоналом. При виконанню проекту студентам необхідно сформулювати отримані результати у вигляді теорем, тверджень, а також визначити, чи будуть інваріантні всі властивості для вихідного і оберненого функціоналу.

Ключові слова: *функціональний аналіз, метод проектів, майбутні учителі математики, функціонали.*

Постановка проблеми. Соціальні та економічні зміни в Україні, швидкий технічний прогрес, інформатизація суспільства ставлять нові цілі перед освітою, однією з яких є формування творчого мислення і продуктивної творчої діяльності студента – майбутнього вчителя математики, як умова його самореалізації в житті.

Доступність і обсяг спеціалізованої фахової інформації виводить на перший план здатність легко орієнтуватися в сучасних розділах математики, самостійно аналізувати проблеми, виявляти перспективні цілі і планувати оптимальні шляхи їх досягнення, втілювати незалежно прийняті рішення на практиці і оцінювати їх наслідки та результати.

Розробка методики навчання функціонального аналізу, спрямованої на розвиток самостійності, критичного мислення, творчої активності – це складний процес. В даний час для актуалізації та закріплення знань, умінь і навичок з функціонального аналізу використовуються невеликі задачі (розв'язання кожної з них не перевищує одного заняття). Однак у формуванні мотиваційної сфери студентів при цьому виникають суттєві труднощі, не створюються умови для самостійного придбання знань студентами, обсяг отриманих знань не знаходить свого застосування в конкретних ситуаціях.

Дана стаття присвячена розробці методики, що дозволяє оптимально поєднувати