УДК 004.942

Р.І. Дзінько, А.М. Гордійчук, О.І. Лісовиченко

ВИКОРИСТАННЯ ШАБЛОНІВ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У МОДЕЛЮВАННІ РТК

Аннотация: В роботі розглядається застосування шаблонів проектування програмного забезпечення при моделюванні структури та поведінки робототехнічних комплексів. Виконується аналіз недоліків та переваг підходів до розробки програмного забезпечення: процедурного, об'єктно-орієнтованого та компонентного.

Ключові слова: Процедурний підхід, об'єктно-орієнтований підхід, компонентний підхід, робототехнічний комплекс, клас, об'єкт, функція, інтерфейс, адаптер, архітектура компонентного проектування.

Вступ

Правильний вибір архітектури програмного забезпечення (ПЗ) для моделювання робототехнічного комплексу (РТК) є однією з найважливіших задач при створенні моделей РТК. Шаблон проектування програмного забезпечення (далі "шаблон проектування") – це архітектурна конструкція багаторазового застосування, що надає вирішення загальної проблеми проектування в рамках конкретного контексту, та описує значимість цього вирішення.

Архітектура РТК є по своїй природі компонентною, тому використання компонентної архітектури програмного забезпечення для моделювання структури і поведінки РТК та його складових на базі шаблонів проектування є виправданою.

Шаблони проектування показують відношення та взаємодію між класами (**Клас** – деяка сутність, що визначає абстрактні характеристики групи об'єктів, включаючи характеристики самих об'єктів (їх атрибути або властивості) та дії, які вони здатні виконувати (їх поведінку, методи або можливості) і об'єктами (**Об'єкт** – деяка сутність, що володіє станом та поведінкою) без визначення того, які кінцеві класи чи об'єкти програмного додатку будуть застосовуватись [1].

Постановка задачі

Необхідно розробити програмне забезпечення, що вирішує наступні задачі:

- структурно-параметричне моделювання РТК;
- функціональне навантаження модулів та РТК в цілому;
- моделювання транспортно-складської підсистеми [3].

З точки зору програмування модель РТК є дуже складною системою (особливо, якщо відбувається деталізація на всіх рівнях), тому спочатку необхідно обрати архітектуру програмного забезпечення (***Архітектура***

програмного забезпечення **–** це представлення, яке дає інформацію про складові ПЗ, про взаємозв’язки між цими складовими, та про правила, що регламентують ці взаємозв’язки [2]), яка б найкраще підходила для досягнення таких його властивостей як:

- структурованість програмного коду;
- масштабованість;
- повторне використання коду;
- наближення загальної схеми програмного коду моделі до структури реального РТК.

Найбільш доцільним з цієї точки зору є використання компонентної архітектури побудованої на базі шаблонів проектування. Для доведення цієї думки необхідно розглянути всі використовувані підходи до побудови архітектури програмного забезпечення.

Принципи компонентної взаємодії та компонентна архітектура

Компонентна архітектура являє собою спосіб організації частин програмного забезпечення в незалежні один від одного компоненти багаторазового використання (рис.1, а), які взаємодіють між собою згідно заздалегідь визначеного інтерфейсу (***Компонент***–деяка сутність, що надає визначений інтерфейс (***Інтерфейс***– декларація способу використання об’єкта) і за допомогою цього інтерфейсу може взаємодіяти з іншими компонентами).

На рис. 1 показано порівняння модульного принципу компоновки робота та архітектури компонентної моделі програмного забезпечення для моделювання робота [3].

Як показано на рис. 1, кожна окрема частина робота взаємодіє з іншою за певними встановленими правилами, що визначаються інтерфейсами. Адаптер (***Адаптер*** **–** програмна реалізація надання одного інтерфейсу через звернення до іншого інтерфейсу, що надається об’єктом [1]) виконує роль елемента, який з’єднує два модулі робота. Таким чином, розробнику ПЗ, як і технологу виробництва, не важливо, як саме сконструйований адаптер. Головне в даному випадку те, щоб він взаємодіяв з іншими компонентами згідно визначеного інтерфейсу.

Наведений приклад є показовим на рівні компонування робота, проте ці принципи можуть використовуватись на всіх рівнях деталізації при проектуванні РТК.

Отже, даний підхід дає можливість концентрувати увагу на проектуванні схеми взаємодії між елементами РТК та налаштуванні технологічного циклу створення продукції, не вдаючись до проробки окремих конкретних деталей.

Такий підхід до розробки необхідної моделі має наступні переваги:

1) прозора властивість структурованості та можливість створення “каркасу” програмного коду;

 ISSN 1560-8956

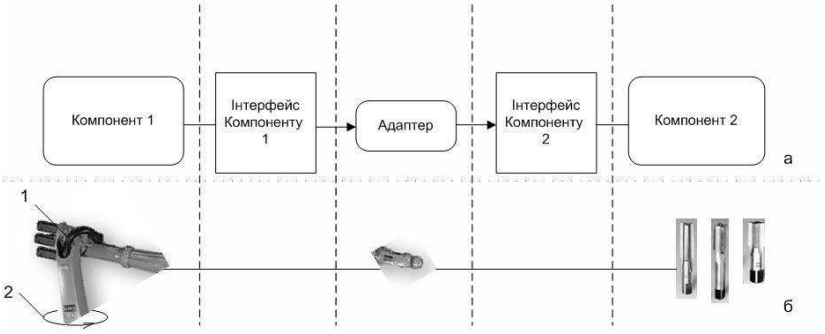

Рис. 1 – Архітектура моделі робота (а), компоновка робота (маніпулятор – перехідник – знімний інструмент) (б)

2) наявність інтерфейсів дозволяє програмістам, які супроводжують даний програмний код та розробляють доповнення, не заглиблюватись в вивчення методів та класів, а одразу отримувати поняття про те, що виконує кожен з об’єктів;

3) масштабованість через розширення функцій інтерфейсів, використоуючи властивості об‘єктно-орієнтованого програмування (шляхом наслідування чи доповнення);

4) можливість повторного використання вже написаного коду, а також зменшення кількості повторювань програмного коду шляхом введення адаптерів, що дозволяють лише модифікувати вже готовий програмний код, не переписуючи великі його частини з нуля.

Порівняння підходів проектування програмного забезпечення

1. Процедурний підхід

Процедурний підхід передбачає створення програмного коду шляхом написання певної кількості необхідних процедур та функцій. Наприклад, розглянемо робот-маніпулятор типу “Пума” з трьома знімними оброблюючими інструментами. Параметри робота та інструментів зберігаються у вигляді глобальних змінних, а функції, що виконує робот, за допомогою інструментів, у вигляді набору процедур. Схема процедурного підходу представлена на рис 2. Зазвичай такий підхід доцільно використовувати при низькій деталізації моделі РТК, коли моделюються всього декілька параметрів, а також, коли створювана модель не потребує подальшого розширення, наприклад, при проектуванні лінії для масового виробництва.

Такий підхід має наступні недоліки:

1) надлишкова структурованість коду (при великій кількості модельованих параметрів кількість глобальних змінних виростає настільки, що розвивати модель стає дуже важко, чи практично неможливо);

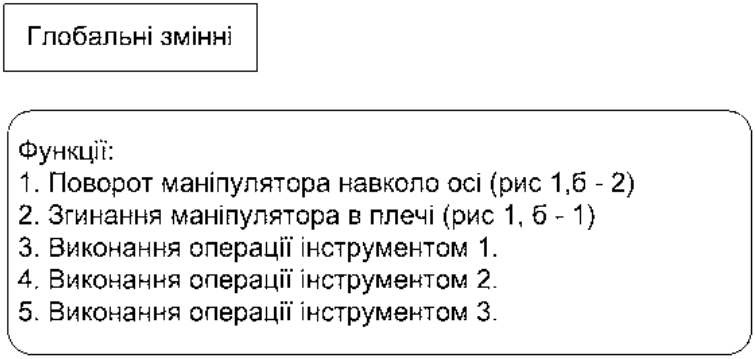

Рис. 2 – Схема програмного коду при процедурному підході

2) відсутність можливості повторного використання коду для різних типів роботів-маніпуляторів, які відрізняються між собою лише декількома параметрами, через потребу переписування функцій;

3) властивість масштабування майже не використовується;

4) ускладнений алгоритм управління моделлю шляхом використання великої кількості розгалужень (рис. 3).

Рис. 3 – Уривок алгоритму вибору функції моделювання обробки деталі

2. Об’єктно-орієнтований підхід

Об’єктно-орієнтований підхід передбачає створення класів, та, відповідно, організації властивостей маніпулятора та оброблюючих елементів у вигляді полів класів, а їх функцій - у вигляді методів (рис. 4).

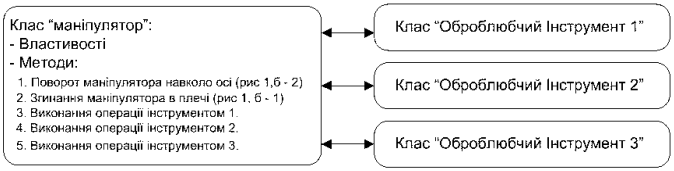

Рис. 4 – Схема програмного коду при об’єктно-орієнтованому підході

Такий підхід є набагато кращим, оскільки, у порівнянні з процедурним, має наступні переваги:

 ISSN 1560-8956

- кращу структурованість програмного коду;
- надає можливість повторного використання фрагментів програмного коду, а також масштабування моделі шляхом наслідування класів;
- алгоритм управління моделлю позбавлений складності завдяки можливості використання властивостей класу, наприклад, “оброблюючий інструмент” описується як поле класу “маніпулятор”.

Проте при такому підході поняття управління маніпулятора, управління маніпулятором та перехідника інкапсульовані в одному класі “маніпулятор”, що не є логічним з точки зору модульного принципу компонування промислових роботів-маніпуляторів.

3. Інтерфейсно-орієнтований компонентний підхід

Компонентний підхід передбачає використання таких шаблонів проектування, як:

- інтерфейс (фундаментальний шаблон);
- адаптер (структурний шаблон);
- фабрика (твірний шаблон).

При цьому схема програмного коду може бути представлена наступним чином, рис 5.

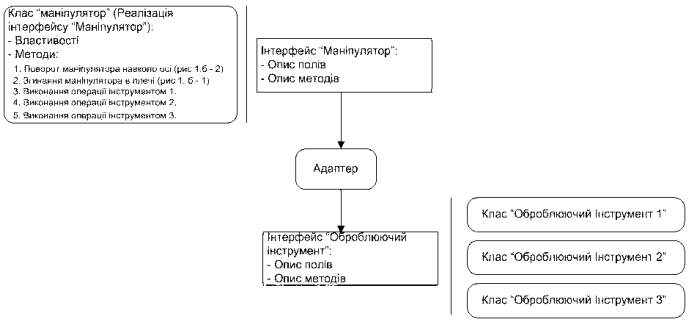

Рис. 5 – Схема програмного коду при інтерфейсно-орієнтованому компонентному підході

Як бачимо з рис. 5, такий підхід виокремлює поняття перехідника від класу “маніпулятор”, а тому надає ряд переваг порівняно з об’єктно-орієнтованим підходом:

1) при використанні перехідників з різними властивостями, не потрібно клонувати класи чи виконувати наслідування, достатньо створити новий адаптер. Адаптер може являти собою клас чи функцію, що реалізує механізм взаємодії маніпулятора та оброблюючого інструмента. Завдяки цьому нам не потрібно створювати програмний код взаємодії з маніпулятором для кожного оброблюючого інструмента, як у випадку з об’єктно-орієнтованим підходом (як показано на рис. 4), адже весь

механізм взаємодії винесений в адаптер. Таким чином підвищується коефіцієнт повторного використання готового програмного коду;

2) введення інтерфейсів дає змогу задати формальний опис правил взаємодії між об’єктами та адаптерами, наближає модель програмного коду до реального об’єкта в більшій мірі, ніж об’єктно-орієнтована модель. Це відбувається завдяки тому, що описуються не тільки сутності, але й їх взаємодія, при цьому ми бачимо взаємодію між різними класами явно, тоді як в ООП взаємодія виражається неявно (один об’єкт викликає методи іншого об’єкта).

Висновки

В межах даної роботи були розглянуті шаблони проектування програмного забезпечення в контексті моделювання структури та поведінки робототехнічного комплексу.

Під час дослідження був зроблений аналіз переваг та недоліків застосування кожного шаблону проектування для розробки моделі структури та поведінки частини РТК, а саме маніпулятора зі знімними оброблюючими елементами. При цьому дослідження показало, що кожна модель має свої недоліки, проте в найбільш незначній мірі вони проявляються при застосуванні інтерфейсно-орієнтованого компонентного підходу, а саме ускладнюється структура програмного коду.

Саме тому при проектуванні програмного забезпечення для моделювання РТК пропонується використання інтерфейсно-орієнтованого компонентного підходу, що демонструє наступні переваги:

1) компонентний підхід найбільш логічно і точно відображає модульний принцип компонування роботів на різних рівнях деталізації, причому програмний компонент можна утотожнити компонувальному модулеві;

2) компонентна архітектура має найкращу структурованість програмного коду;

3) компонента архітектура дозволяє повторне використання програмного коду в більших масштабах, ніж інші.

Література

1. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, USA, 1994.
2. Steve McConnell. “Code Complete: A Practical Handbook of Software Construction, 2nd Edition”, Addison-Wesley, USA, 2004.
3. Ямпольський Л.С., Мельничук П.П., Самотокін Б.Б., Поліщук М.М., Ткач М.М., Остапченко К.Б., Лісовиченко О.І. “Гнучкі комп’ютеризовані системи: проектування, моделювання і управління”, Житомир, ЖДТУ, 2005.

Отримано 04.11.2011 р.

 ISSN 1560-8956