УДК 004.4'22

МЕТОДИ РЕІНЖИНІРИНГУ ПРОГРАМНИХ СИСТЕМ

К. т. н. С.С. Великодний, Одеська національна морська академія

У тезах доповіді узагальнено питання застосування різноманітних методів програмування при еволюційному оновленні програмних систем. Особливу увагу приділено організації складального програмування на основі компонентів повторного використання.

В тезисах доклада обобщенно вопрос применения разнообразных методов программирования при эволюционном обновлении программных систем. Особое внимание уделено организации сборочного программирования на основе компонентов повторного использования.

The theses of the report summarizes the issues the use of different programming techniques in evolutionary upgrade of software systems. Particular attention is given to the organization of assembly programming based on components reuse.

***Ключові слова**: метод програмування, складальне програмування, реінжиніринг, компонент повторного використання, програмна система, життєвий цикл, програмний продукт*

Одне з головних завдань сучасного програмування – створення теоретичних і прикладних основ побудови складних програм з більш простих програмних елементів, які написані на сучасних мовах програмування (МП). Фактично рішення цієї задачі здійснюється шляхом збирання, об'єднання або інтеграції різнорідних програмних ресурсів та компонентів повторного використання (КПВ), включаючи модулі та програми реалізації деякої предметної області.

Таким чином, мета поданої наукової праці – систематизація методів інтеграції у нові програмні структури КПВ, програм, та готових ресурсів, що накопичено людством за визначений час.

Комплектуючі елементи складального програмування

Процес складання будь-яких виробів характеризується: комплектуючими деталями та вузлами, схемою складання (взаємозв'язками окремих компонентів й правилами взаємодії), складальним конвеєром (технологією складання). Конкретизуємо ці поняття з точки зору методу складального програмування. При цьому будемо припускати, що використовуються тільки готові програмні продукти.

Комплектуючими в методі складального програмування – є прості програмні елементи (модулі, об'єкти, компоненти, сервіси тощо).

З них збираються програмні об'єкти різної складності та ступеня готовності: програми, комплекси, пакети прикладних програм, програмні системи (ПС) тощо.

Модуль – незалежна функціональна частина програми, до якої можна звертатися як до самостійної одиниці через зовнішній інтерфейс.

Об'єкт – базове поняття у об'єктно-орієнтованому програмуванні, яке має властивості спадкування, інкапсуляції та поліморфізму. Об'єднує дані й операції (методи). Об'єкти взаємодіють між собою через повідомлення. Об'єкти із загальними властивостями і методами утворюють клас, будучи у ньому екземплярами класу.

Компонент – програмний об'єкт, який реалізує деяку функціональність та є базовим поняттям компонентного програмування й компонентно-орієнтованої розробки (component-based development – CBD). Основна форма подання компонента – каркас та контейнер.

Каркас – високорівнева абстракція, у якій функції відокремлені від завдань управління.

Контейнер – оболонка, всередині якої реалізовані функції у вигляді екземплярів компонентів, забезпечує взаємодію із сервером через стандартні інтерфейси (function, home interface). Примірники звертаються один до одного через системні сервіси даного контейнера або іншого.

Сервіс – це програмний ресурс, який реалізує деяку функцію, у тому числі й бізнес функцію. Містить незалежний інтерфейс з іншими сервісами та ресурсами. Веб-сервіс забезпечує реалізацію завдань інтеграції додатків різної природи та використовується як провайдер. Сукупність взаємодіючих сервісів, веб-сервісу та їх інтерфейсів утворює сервісно-орієнтовану архітектуру (service oriented architecture), доступ до яких відбувається через веб-мови та протоколи.

Результатом складання може бути будь-який програмний об'єкт, за винятком елементарного елемента – модуля, компонента, сервісу тощо. Однак для практичного застосування вибирається декілька базових типів програмних об'єктів, які розглядаються як готові компоненти та результат використання методу складального програмування.

Для технологічності зборки всі об'єкти повинні мати паспорти, що містять дані, необхідні для інформаційного сполучення та організації спільного функціонування в рамках програмного об'єкта більш складної структури. Важлива умова складання полягає у наявності великої кількості різноманітних комплектуючих, тобто ПС, які забезпечують вирішення широкого спектру завдань з різних предметних областей.

Схема складання програмних об'єктів

Під схемою складання програмних об'єктів розуміється схема їх взаємодії, що визначається безпосередніми зверненнями до компонентів (типу оператора CALL) або послідовністю їх виконання. При цьому взаємодія кожної пари об'єктів залежить від спільного використання даних. У загальному випадку, схема зборки складається із сукупності моделей, що відображають різні типи зв'язків між компонентами:

а) передача управління;

б) обмін інформацією;

в) умови спільного функціонування тощо.

Операції складання виконуються згідно із паспортами об'єктів та правилами сполучення. Інформація у паспортах повинна бути систематизована та виділена у такі окремі групи:

а) дані, що передаються;

б) типи цих даних;

в) об'єкти, що викликаються;

г) файли, що спільно використовуються тощо.

Правила сполучення визначають сумісність поєднуваних об'єктів та містять опис функцій, необхідних для узгодження різних характеристик, наведених у паспортах.

Організація складального конвеєру

Процес складання може проводитися ручним, автоматизованим та автоматичним способами. Як правило, останній спосіб практично нездійсненний, що пов'язано із недостатньо формальним визначенням програмних об'єктів та їх інтерфейсів. Ручний спосіб недоцільний тому, що складання готових компонентів являє собою великий обсяг дій, які носять скоріше рутинний, ніж творчий характер. Найбільш прийнятний – автоматизований спосіб збірки у середовищі системи, яка за заданими специфікаціями (моделями) програм здійснює складання за допомогою стандартних правил сполучення під управлінням людини.

Засоби, що підтримують даний спосіб збірки, називаються інструментальними засобами складального програмування.

До них відносяться засоби комплексування (об'єднання компонентів у більш складний об'єкт); засоби інтерфейсів (реалізація сполучення об'єктів згідно їх паспортів та стандартними правилами зв'язку); засоби опису та використання моделей складального програмування (сукупність моделей складання різних програмних об'єктів).

Із розглянутої схеми складання виділимо задачі складального програмування та умови його застосування:

а) вибір компонентів із числа готових програмних об'єктів для забезпечення процесу вирішення класу задач з певної предметної області;

б) проектування структури (моделей) створюваного об'єкта, елементами якого є готові програмні елементи, визначені на множині даних обраної предметної області;

в) складання (згідно моделям) програмного продукту, що реалізує функції, які відповідають цілям та задачам автоматизації предметної області.

Необхідні умови застосування даного методу програмування включають у себе:

а) наявність великої кількості різноманітних програмних продуктів, як об'єктів складання;

б) паспортизація програмних об'єктів складання;

в) наявність достатньо повного набору стандартних правил сполучення й алгоритмів їх реалізації, засобів автоматизації процесу складання;

г) створення технологій застосування розроблених об'єктів для використання у більш складних програмних продуктах.

Остання умова означає, що повинні існувати певні форми уявлення ПС як знань про предметні області, універсальні з точки зору проектування та розробки ПС.

Основне питання складання – це реалізація інтерфейсів між окремими модулями та / або компонентами, що забезпечують їх «стикування» або зв'язок.

Базові методи реінжинірингу програмних об'єктів

Інтеграція програмних структур спочатку виконувалася за допомогою готових підпрограм бібліотек різного призначення шляхом їх вставки до ПС, що інтегрується. Потім з'являлися різні методи реінжинірингу (конкретизуюче, синтезуюче, композиційне тощо), які вирішували проблему комплексування програмних об'єктів методами, близькими до складання різнорідних об'єктів.

Вони сприяли поступовому розвитку індустріальних основ виготовлення ПС: КПВ, життєвий цикл (ЖЦ), вимір та оцінювання робіт на процесах ЖЦ й самого продукту на окремі показники якості. Пізніше сформувалися й інші стилі програмування, які використовують готові компоненти в процесі створення ПС методами складального типу – комплексування, інтеграція, композиція. До них можна віднести: об'єктно-орієнтоване, компонентне, генеруюче, аспектне, агентне тощо. Розглянемо базові методи реінжинірингу з елементами інтеграції, комплексування та синтезу.

Конкретизуюче програмування базується на виділенні з деякої універсальної програми окремої її частини, налаштованої на особливі, певні умови виконання. Можна відзначити два типи такого виділення. Перший характеризується формуванням конкретної програми та аналогічний процесу макрогенерації. Другий тип пов'язаний із конкретизацією інформаційних структур у програмному засобі, що використовується.

При *синтезуючому* програмуванні будується модель програми за специфікацією завдання, за якою буде синтезована програма її вирішення. Специфікація задається у термінах деякої формальної мови. На її основі та правил побудови алгоритмів опису конкретної предметної області відбувається формування необхідної програми.

Композиційне програмування базується на принципах функціональності та композиційності, які розглядають програми як набір функцій, що будуються з інших функцій за допомогою спеціальних операцій, названих композиціями. На основі композиційного уточнення (експлікації – explication) створюється логіко-математична система композиційної побудови програм, яка об'єднує сучасні парадигми програмування (структурне, функціональне, об'єктно-орієнтоване тощо) у рамках єдиної концептуальної, експлікативної платформи. Її основу складають три типи об'єктів: самі об'єкти, методи композиції та засоби побудови з одних об'єктів інших об'єктів або функцій.

Складальне програмування характеризується складальною побудовою програм із готових «деталей», якими є програмні об'єкти різного ступеня складності. Елементи процесу складання присутні у багатьох методах програмування: згори-униз, знизу-догори тощо.

Програмісти, розробляючи програми без застосування будь-яких методів програмування, виділяють повторно використовувані оператори та

оформляють їх у вигляді окремих, самостійних фрагментів або підпрограм для подальшого використання.

Виникають питання: у чому суть складального програмування та що дозволяє виділити його у вигляді окремого методу реінжинірингу. Для відповіді на поставлене питання перш за все відзначимо, що таке складальне програмування.

Складальне програмування:

а) є одним з методів програмування та підкоряється загальним закономірностям;

б) представляє одну із форм повторного використання програмних засобів;

в) якісно відрізнятися від процесів складання у інших методах.

Під методом складання розуміється спосіб сполучення різномовних програмних об'єктів у МП, заснований на теорії специфікації й відображення (Mapping) типів та структур даних МП, представлених алгебраїчною системою. Основу алгебраїчного формалізму складають типи даних, операції над ними та функції релевантного, еквівалентного перетворення одних типів у інші.

Сполучення пар об'єктів у МП здійснюється через оператор виклику, у списку параметрів якого задаються значення формальних параметрів, які перевіряються на відповідність типів даних твердженнями алгебраїчної системи, що доводять необхідні й достатні умови відображення даних у класі МП.

Результат відображення – оператори релевантного перетворення типів даних у спеціальному модулі-посереднику для кожної пари об'єктів, що сполучаються.

Методом близьким до збірки – є генерація різних об'єктів до одного загального вихідного коду й середовища функціонування. Поняття генерації програм виникло майже одночасно із поняттям збірки та, на сьогоднішній день, воно отримало новий розвиток у зв'язку із орієнтацією на опис моделі предметної області (домену) засобами мови DSL (Domain Specific Language), що відображає специфіку цієї галузі.

Такий новий напрям ще не має стандартних рішень щодо самої проблеми поступової трансформації опису у цій мові та виконання інструментів генерації, налагодження та інтеграції для отримання кінцевої ПС.

ПС як форма представлення знань

Програмні засоби – це фактично один із способів представлення знань про вирішувані завдання у конкретних предметних областях. З даної позиції розглянемо більш докладно процес розробки та використання ПС.

При розробці баз знань необхідно розглянути два ключових питання: уявлення знань та їх використання. Методами представлення знань у розглянутій області по суті – є методи та засоби розробки ПС як знання фахівців-розробників. При цьому мовами подання знань служать мови:

а) специфікацій;

б) програмування;

в) опису даних;

г) управління завданнями тощо.

Самі розроблені ПС представляються як частина знань про предметну область, до якої належить розв'язувана задача.

Використання знань пов'язано із процесами впровадження (частково) та супроводу ЖЦ ПС, тобто її застосування. Метод використання ПС – технологія її застосування. Результати розглянутого підходу до уявлення знань про програмні продукти, що повторно використовуються, відображені на рис. 1. Зокрема, цією схемою задовольняють всі три зазначених вище види програмування систем на основі готових компонентів.

Відмінність складального програмування від інших методів

Головна відмінність складального програмування від інших методів повторного використання ПС полягає у наявності «стійкого зворотного зв'язку» (перевизначення (redefines) – на рис. 1).

«Зворотній зв'язок» означає, що до технології використання розробленого програмного продукту входять методи його застосування при проектуванні інших ПС, де програмний продукт входить як складовий елемент. Термін «стійкий» підкреслює, що методи застосування розроблених програмних продуктів у подальшій розробці ПС є необхідною умовою використання методу складального програмування.

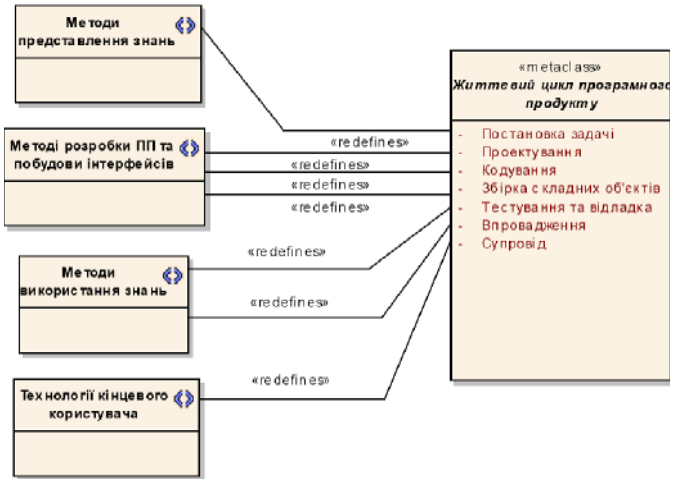

Рис. 1. Діаграма профілів із представленням перевизначення методів розробки та використання програмного продукту у рамках його життєвого циклу

Таким чином, метод складального програмування є узагальненням традиційних методів розробки, оскільки до нього додатково входять й методи побудови інтерфейсів між окремо розробленими програмними продуктами.

Отже, змінюється й ЖЦ. Процес проектування включає вибір готових КПВ, які повністю або частково вирішують поставлену задачу та вирішують проблеми інтерфейсів між компонентами. Якщо потрібна додаткова розробка нових та складних ПС, то формується часткове завдання, що розв'язується із застосуванням традиційних методів програмування, а сам процес кодування спрощується за наявності готових декількох або усіх необхідних компонентів.

Основну увагу при цьому приділено комплексному налагодженню програмних об'єктів, що створено із готових компонентів. Головний напрямок процесу об'єднання накопичених у цих базах знань різнопланових програмних ресурсів – це розвиток сучасних методів програмування взагалі та виділення з них – методів реінжинірингу зокрема.

СПИСОК ЛІТЕРАТУРИ:

1. Тимченко А.А. Основи системного проектування та системного аналізу складних об'єктів. Кн. 1. Основи САПР та системного проектування. – К.: Либідь, 2000. – 272 с.

2. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. доп. и перераб. – К.: Наук. думка, 2009. – 372с.

3. Фаулер М. Рефакторинг: улучшение соответствующего кода. – СПб.: Символ-Плюс, 2003. – 432 с.

4. Пантелеймонов А.А. Аспекты реинженерии приложений с графическим интерфейсом пользователя // Проблемы программирования. – 2001. – № 1–2. – С. 53–62.

5. Реінженерія програмних систем [Электронный ресурс]. – Режим доступа: http://www.programsfactory.univ.kiev.ua/content/books/2/108

УДК 681.326

СИНТЕЗ МАТРИЧНЫХ УМНОЖИТЕЛЕЙ С ВСТРОЕННОЙ СХЕМОЙ САМОТЕСТИРОВАНИЯ

К.т.н. Я.Ю.Королева, Национальный технический университет «Харьковский политехнический институт»

Рассматривается моделирование схемы ячейки умножителя и выбора наборов, обеспечивающих выполнение условий С-тестируемости схемы.

Розглядається моделювання схеми комірки помножувача і вибору наборів, які забезпечують виконання умов С-тестованості схеми.

Deals with the modeling circuit cell multiplier and select sets, providing that the conditions C-testability of the circuit.

Ключевые слова: *конвейерный умножитель, тестирование схемы.*

Введение

Матричные умножители относятся к классу двумерных сетей, предназначенных для арифметического умножения двоичных чисел, которые широко применяются в качестве расширителей арифметически-логических устройств, устройств кодирования и декодирования, в составе функциональных преобразователей и устройств цифровой обработки информации.

В [1] предлагается использовать регулярность структуры умножителя и модификацию ячейки сети с целью обеспечения свойства С-тестируемости. При этом предполагается, что обнаруживаемые неисправности ограничены классом F1. Предложенная модификация ячеек сети и функциональный подход к синтезу проверяющих тестов обеспечивают проверку исправности умножителя 16-ю тестовыми наборами при увеличении аппаратурных затрат на модификацию структуры сети ~ на 12-15 %.

С другой стороны, с ростом размерности операндов число ячеек сети возрастает в квадратичной зависимости от числа разрядов умножаемых чисел, что создает значительные трудности как при реализации самого умножителя на отдельном кристалле, так и при выполнении процедуры диагностирования умножителя.

Актуальность исследований

Традиционные подходы к построению проверяющих тестов для двумерных сетей дают положительные результаты лишь при ограничении класса обнаруживаемых неисправностей неисправностями константного типа и невысокой размерности сети.

Следовательно, актуальной является проблема обнаружения неисправностей константного типа на входах-выходах умножителя и внутренних узлах каждой ячейки на вентильном уровне ее описания, а так же исключения необходимости модификации ячейки умножителя, приводящей к увеличению аппаратурных затрат при реализации схемы.

Формулирование цели и постановка задачи

Цель данной работы - найти множество тестов, удовлетворяющих условиям С-тестируемости схемы умножителя и обнаруживающих все неисправности константного типа на входах-выходах умножителя и внутренних узлах каждой ячейки на вентильном уровне ее описания.

Для достижения поставленной цели необходимо решить задачи путем моделирования схемы ячейки умножителя и выбора наборов, обеспечивающих выполнение заданных условий.

Конвейерно-арифметические матрицы

Одним из подходов, позволяющим решить эти проблемы, является тестопригодное проектирование конвейерных арифметических матриц[2].

Схема конвейерного матричного умножителя состоит из двух типов ячеек: MS -ячейка мультиплексора «2 на 1» и ячейка полного одноразрядного комбинационного сумматора, расширенного схемой И, как показано на рисунке 1.

Конвейерный умножитель имеет четыре ступени. В первой и третьей ступенях используются обычные схемы полных одноразрядных сумматоров, во второй и четвертой ступенях функции полных одноразрядных сумматоров совмещены с функцией запоминания информации. В [3] предложены и подробно описаны различные варианты схем элементов памяти, обеспечивающие высокую производительность функционирования конвейерных арифметических матриц.

.