УДК 004.423

О.В. Бойченко,
доктор технічних наук, доцент,
О.С. Лєнков,
Л.В. Охрамович

МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ АРХІТЕКТУРИ СПЕЦИФІКАЦІЇ

У статті проаналізовано можливість застосування методології проектування архітектури специфікації в вирішенні завдань розроблення нової технології пошарового проектування стійкого програмного забезпечення інформаційної системи спеціального призначення (ІССП).

Використанням причинно-наслідкових зв'язків між завданнями системи, заснованої на їхній взаємній залежності за даними, створюються умови для одержання схеми виконання задач, вільної від конфліктів, породжених відсутністю синхронізації даних.

Застосування методологій, орієнтованих на оброблення за рахунок методу функціональної декомпозиції, методу проектування структур даних і т. ін., дозволяє створити умови визначення детального формату кожного елемента вхідних даних ІССП.

Ключові слова: *методологія, архітектура специфікації, метод перетворення графів, технологія пошарового проектування, стійке програмне забезпечення.*

В статье проанализирована возможность применения методологии проектирования архитектуры спецификации в решении задач разработки новой технологии послойного проектирования устойчивого программного обеспечения информационной системы специального назначения (ИССН).

Использованием причинно-следственных связей между задачами системы, основанной на их взаимной зависимости по данным, создаются условия для получения схемы выполнения задач, свободной от конфликтов, порожденных отсутствием синхронизации данных.

Применение методологий, ориентированных на обработку за счет метода функциональной декомпозиции, метода проектирования структур данных и т.д., позволяет создать условия определения детального формата каждого элемента входных данных ИССН.

Ключевые слова: *методология, архитектура спецификации, метод преобразования графов, технология послойного проектирования, устойчивое программное обеспечение.*

Paper explores the possibility of applying the methodology of designing architecture specification in solving the problems of the development of new technologies for the sustainable design of layered software information system for special purposes (ISSP).

By using cause-effect relationships between the tasks of the system based on their mutual dependence on the data, the conditions for obtaining the scheme of tasks free from conflicts generated by the lack of synchronization.

Applying of the methodologies, oriented at the processing at the expense of the method of functional decomposition, as well as of the design method of data structures etc. allows the determination of the conditions of a detailed format of each input element ISSP.

Keywords: *methodology, architecture specification, the method of converting graphs layered design technology, stable software.*

Вирішення проблеми підвищення відмовостійкості**,** надійності та живучості інформаційних систем спеціального призначення (ІССП), що часто функціонують в умовах значних обсягів даних, невизначеності та обмежених термінах обробки та надання інформації для прийняття управлінського рішення вимагають розробки стійкого програмного забезпечення та систем контролю несанкціонованого впливу на інформаційні ресурси системи.

Розробленням методологічних основ оброблення, заснованих на модульному принципі, методі функціональної декомпозиції, методів проектування з використанням потоку даних, методів проектування, заснованих на використанні структур даних для проектування стійкого програмного забезпечення ІССП, свого часу займалися такі фахівці, як: Йодан Е., Холстед М.Х., Майерс Г., Джексон М., Сбітнев А.І. та інші [1–5].

Необхідність проведення наукових досліджень щодо вирішення завдання підвищення стійкості програмного забезпечення ІССП обумовлена широкими можливостями методології архітектури специфікації, яка є базисом створення нової технології пошарового проектування стійкого програмного забезпечення автоматизованих комп'ютерних систем.

Методології, орієнтовані на оброблення, надають особливого значення процесу і структурі в створенні архітектури програми.

Зокрема, найбільш відомі методології, орієнтовані на оброблення, засновані на використанні 5 основних методів (рис. 1).

Основні концепції модульного програмування [1]:

кожний модуль реалізує єдину незалежну функцію;

кожний модуль має єдину точку входу/виходу;

розмір модуля по можливості треба мінімізувати;

кожний модуль може бути спроектований і закодований різноманітними членами бригади програмістів і може бути окремо протестований;

уся система побудована з модулів.

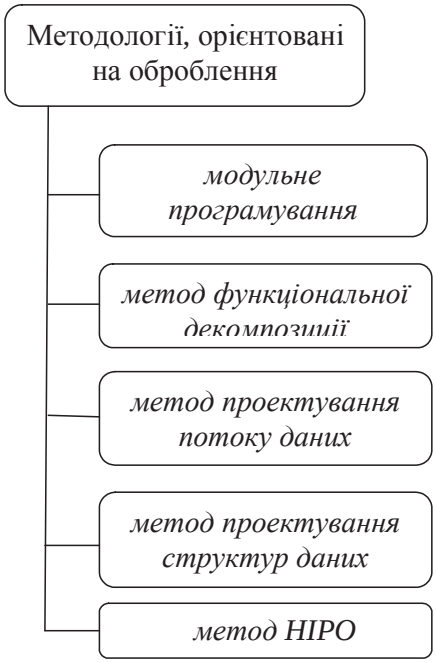

Рис. 1. Найбільш відомі методології, орієнтовані на оброблення

При такому підході проектована система розділяється на декілька частин, одночасно утворюваних різноманітними програмістами.

Кожний модуль реалізує єдину функцію. Розмір модуля невеликий, тому тестування кероване і може бути проведено ретельно. Після кодування і тестування всіх модулів відбувається їхня інтеграція і тестується вся система. При супроводі тестується й налагоджується тільки той модуль, що погано працює. Очевидні переваги в полегшенні написання і тестування програм, зменшується вартість їх супроводу.

Модуль не повинний змінювати команди іншого модуля і, як правило, не повинний зберігати історію своїх викликів (хоча остання властивість істотно спрощує трасування при збої).

Відомі позитивні якості модульних програм:

легкість упорядкування і налагодження; функціональні компоненти такої програми пишуться й налагоджуються порізно; можливе добре структуроване налагодження як “нагору”, так і “униз”;

легкість супроводу і модифікації;

можливість розподілу модулів між програмістами різноманітної кваліфікації відповідно до рівня складності;

можливість створювати бібліотеки найбільше вживаних програм;

спрощення процедури завантаження в оперативну пам’ять великої задачі, що потребує сегментації;

виникнення численних природних контрольних точок для спостереження за просуванням програми, що використовуються для підвищення усталеності ПЗ.

Передача керування в модульній структурі відбуваються лише по вертикальних лініях, що з’єднує модулі в схемі ієрархії.

Будь-який модуль може активізувати підпорядкований модуль і одержати керування після завершення його роботи. Модуль більш низького прошарку не може викликати модуль більш високого прошарку.

Модуль зобов'язаний повернути керування тому модулю, що його викликав. Прийняття рішень модулями нижчого рівня за модулі вищого рівня не припускається. Надмірна кількість аргументів, що подаються модулю, указує на необхідність поділу функції (основна мета при цьому − скорочення числа аргументів). У роботі [2] наведена формула для числа модулів у програмі:

$$M = П/6,$$

де М − число модулів; П − число єдиних за змістом вхідних і вихідних параметрів, включених в алгоритм.

Формула заснована на припущенні, що для ідеального модуля число параметрів дорівнює 6.

Функціональна декомпозиція істотно базується на стратегії "розділяй і керуй".

Відомий фахівець в галузі методології програмування Парнас, що спробував формалізувати процедуру функціональної декомпозиції у формі покрокової деталізації, як критерій декомпозиції системи запропонував концепцію приховування інформації. При використанні цього критерію кожний модуль характеризується суб'єктивним рішенням проектувальника.

Тільки деяка інформація про цей модуль потрібна іншим модулям, зв'язки між модулями організуються за допомогою добре визначених інтерфейсів. Іншою важливою ідеєю є проектування програмної системи у вигляді набору віртуальних машин замість традиційного підходу, що використовує блок-схеми. Перевага функціональної декомпозиції в її придатності. Хиби − непередбачуваність і мінливість.

Методи проектування з використанням потоку даних використовують потік даних як рушійну силу процесу проектування програми. При цьому використовуються різноманітні функції відображення, що перетворюють потік інформації в структуру програми.

Структурне проектування засноване на концепції, яка висунута в роботах Йодана (1979 р.) і Майерса (1978 р.). Його іноді називають композиційним проектуванням або трансформаційним проектуванням. Метод намагається боротися з хибою, властивою методу функціональної декомпозиції, при використанні якого не можна управляти якістю декомпозиції функції. Цей підхід складається з концепції структурного проектування, генеральної лінії композиційного проектування і деталізації проекту, критерію міри, способів аналізу проекту.

Підхід полягає у відображенні потоку даних проблеми в структуру програми з використанням деяких способів аналізу проекту. Прийнята така процедура:

ідентифікується потік даних і відтворюється граф потоку даних;

ідентифікуються вхідні, центральні і вихідні перетворюючі елементи;

формується ієрархічна структура програми, що використовує ці елементи;

деталізується й оптимізується структура програми, сформульована на 3-му кроці.

Такий підхід звичайно застосовується при відсутності яскраво виражених структур даних.

Технологія структурного аналізу проекту SADT заснована на структурному аналізі, запропонованому Россом. SA – графічна мова, яка використовується для зрозумілого вираження ієрархічних і функціональних зв'язків між будь-якими об'єктами і діями. Структура системи, подана графічно, вип'ячує інтерфейси між компонентами структурно, модульно й ієрархічно. SADT включає процедури планування керуванням, розробленням і керування конфігурацією, засоби організації працюючих спеціалістів у бригади і зв'язки між ними.

SADT успішно застосовується в різноманітних галузях. Метод особливо ефективний на ранніх і пізніх стадіях розвитку системи і менше ефективний при деталізації. Водночас, дозволяючи кожному проектувальнику створювати незалежні діаграми, можна одержати додаткові складнощі в процесі їх перегляду.

Метод перетворення графів заснований на одержанні системної специфікації у вигляді графа керування задачами з графа інформаційного зв'язку між ними. В основі методу – використання причинно-наслідкових зв'язків між задачами системи, заснованої на їхній взаємній залежності за даними. Основна мета – одержати схему виконання задач, вільну від конфліктів, породжених відсутністю синхронізації даних [3].

Щодо проектування, заснованого на використанні структур даних, існують два підходи, розвинені незалежно Джексоном і Уорнером. Обидва використовуються як для конструювання архітектури, так і для здійснення деталізації проекту.

У методології Джексона структура даних використовується як ключовий елемент у побудові гарного програмного проекту. Основна структура системи програми визначається структурою даних, яку вона опрацьовує. Програма розглядається як механізм, за допомогою якого вхідні дані перетворюються у вихідні. Використовуючи вхідні і вихідні структури як основу, намагаються одержати добре структуровані програми.

Основна перевага методології Джексона в тому, що якість результуючого проекту не залежить від досвіду проектувальника, кожний крок проектування може бути верифікований, різноманітні проектанти, працюючи незалежно над однією і тією самою проблемою, одержують такий самий результат.

Проте у методології немає рекомендацій, як структурувати дані. Основні дії зводяться до такого [4]:

ідентифікувати і показати структуру вхідних даних і структуру вихідних даних;

показати структуру програми, поєднуючи зображення цих структурних елементів;

визначити дискретні операції, які складають програму;

перетворити операції в текст програми.

Методологія Уорнера подібна методології Джексона в тому, що ключем до проекту програми є структури даних. Проте процедура проектування більш деталізована. Використовуються чотири види подання проекту: діаграми організації даних, діаграми логічного проходження, список інструкцій, псевдокод. Діаграма організації даних описує вхідні і вихідні дані. Діаграми логічного проходження подають логічний потік цих даних. Список інструкцій містить команди, що

використовуються в проекті. Псевдокод потрібний при описі кінцевих результатів проектування.

Методологія Уорнера може бути узагальнена в такий спосіб [5]:

ідентифікувати усі вхідні дані системи;

організувати вхідні дані в ієрархічну форму;

визначити детальний формат кожного елемента вхідного файлу і зафіксувати число їх появ;

повторити кроки 1–3 для вихідних даних;

специфікувати деталі програми, ідентифікуючи типи команд, що містяться в проекті, в такому порядку: читання, розгалуження, обчислення, входи, виходи, виклик підпрограм;

використовувати діаграми типу блок-схем для показу логічної послідовності інструкцій, використовуючи спеціальні символи для подання початку процесу, кінця процесу, розгалуження і вкладення;

пронумерувати елементи логічної послідовності і розкрити їх за допомогою інструкцій, записаних у кроці 5.

HIPO (Ієрархія плюс Вхід, Оброблення, Вихід) – це метод ієрархічних діаграм, розвинений фірмою IBM. Основні характеристики:

спроможність подавати зв'язок між вхідними/вихідними даними і процесом опрацювання;

можливість декомпозувати систему ієрархічно, не залучаючи зайві дрібні деталі;

використання трьох елементів: вхід, оброблення, вихід.

Оброблення (процес) специфікується як центральний блок діаграми, вона сполучена з елементами, що відображують вхід і вихід.

Основна процедура проектування з використанням HIPO:

почати з найвищого рівня абстракції;

ідентифікувати вхід, вихід і опрацювання;

з'єднати кожний елемент входу і виходу з відповідним обробленням;

документувати кожний елемент системи, використовуючи HIPO діаграми;

деталізувати діаграму, використовуючи кроки 1–4.

Програмування великих систем ПЗ істотно відрізняється від програмування малих програмних систем. Більшість програмних мов не дає засобів специфікації взаємодії між модулями. Першою мовою, що вирішує цю проблему, очевидно, варто вважати мову MIL (мова взаємозв'язку між модулями), що формально специфікує модульну структуру і зв'язки в ній. Крім описів внутрішньо модульних зв'язків і атрибутів модулів, MIL є засобом керування проектом і засобом підтримки процесу проектування.

Іншим варіантом подібної мови є SDL. У ньому описуються керування інтерфейсами модуля і керування системою.

Визначене місце в цьому займають мови подання графів. Запропоноване в методі перетворення графів використання причинно-наслідкових зв'язків між задачами системи, заснованої на їхній взаємній залежності за даними подання, підтримано мовою кортежів.

Функціонально-подійно-режимна декомпозиція (ФПРД) покладена в основу технології проектування ПЗ систем реального часу (зокрема АСУ). Вона є узагальненням методу функціональної декомпозиції у бік використання елементів

реального часу та підвищення усталеності і живучості програмних систем. У цілому технологія, заснована на ФПР-декомпозиції, інтегрує в собі багато елементів перерахованих вище підходів, розвиваючи частину з них самостійно і незалежно.

Застосування методологій, орієнтованих на оброблення за рахунок методу функціональної декомпозиції, методу проектування структур даних і т. ін., дозволяє створити умови визначення детального формату кожного елементу вхідних даних ІСПР у розробленні нової технології пошарового проектування стійкого програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Йордан Э.* Структурное проектирование и конструирование программ / Э. Йордан. – М. : Мир, 1979. – 415 с.

2. *Холстед М.Х.* Начало науки о программах / *М.Х. Холстед*. – М. : Финансы и статистика, 1981. – 128 с.

3. *Сбитнев А.И.* Структурная организация и проектирование ПОАСУТП / А.И. Сбитнев. – К. : УСМ, 1982. – № 5. – С. 38–42.

4. *Jakson M. A.* Principles of Program Design / *M. A. Jakson*. – N.Y. : Academic Press. – 1975. – P. 288.

5. A software design system based on a unified design metodology // J. of Information Proc. / O. Shigo et al. – 1980. – Vol 3. – № 3. – PP. 186–196.

Отримано 20.01.2014