Корнілов І.С.
студент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МОДУЛЬНА АРХІТЕКТУРА: ОСНОВА ЕФЕКТИВНОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На сьогоднішній день існує чимало відомих програмних продуктів з відкритим вихідним кодом. Проаналізувавши їх деяку частину, можна сказати, що вони зроблені з використанням модульної архітектури. Модульність не є новою концепцією, але за останні роки вона швидко набирає оберти. Це можна пояснити тим, що сучасні технології розробки програмного забезпечення дозволяють створювати дійсно складні системи. При цьому складні системи повинні бути гнучкими та відкритими для розширення функціональності. Для досягнення цієї мети від проектувальників та розробників вимагається високий рівень дисципліни в проектуванні архітектури програмного забезпечення. В іншому випадку їх буде очікувати хаос в архітектурі та вихідному коді програмного продукту і неможливість швидкого та легкого розширення функціональності системи.

Метою даної роботи є розгляд модульної архітектури як основи ефективної розробки програмного забезпечення, а також принципів створення модульної архітектури.

Суть модульної архітектури полягає в тому, що система повинна бути не монолітом, а повинна складатися з деякої кількості незалежних підсистем. Тобто система декомпозується на менш складні незалежні функціональні модулі, що дає можливість розробляти кожен модуль окремо [1].

Модульна архітектура має наступні переваги:

1. Масштабованість. Це можливість легкого розширення функціональності системи за рахунок додавання нових модулів.

2. Можливість відносно легкого тестування. Надається можливість тестувати кожен модуль окремо, а також можливість заміни потрібних модулів «заглушками», тобто імітаціями модулів.

3. Ремонтопридатність. Виправлення помилок в одних модулях не впливає на роботу інших модулів. Це надає системі стійкості та стабільності.

4. Взаємозамінність модулів. Модулі в будь-який момент часу можна забрати з системи або замінити його іншим модулем. Такі маніпуляції з модулями ніяк не впливають на стабільність роботи системи (система працює без збоїв, але можливо без тієї функціональності, яку містив від'єднаний чи підмінений модуль).

5. Повторне використання. Існуючі модулі можна використовувати в інших системах із сумісним інтерфейсом. Тобто маючи деякий набір модулів можна створювати різноманітні системи з невеликими затратами часу, який піде лише на створення інфраструктури системи, бо вся функціональність є в існуючих модулях.

Концепція модульної архітектури програмного забезпечення не існує окремо, а включає в себе такі принципи як: використання принципів SOLID, широке використання декомпозиції, використання шаблонів розробки ПЗ.

Декомпозиція може бути ієрархічною та функціональною. У випадку ієрархічної декомпозиції система розбивається на великі функціональні блоки, які абстрактно описують роботу системи. Далі ці блоки деталізуються і виділяються менші функціональні блоки. У випадку функціональної декомпозиції модулі виокремлюються виходячи із задач, які вирішує система. Тобто на кожну таку задачу виділяється модуль, адже модуль – це не просто набір деяких класів та об'єктів, а підпрограма, яка вирішує конкретну задачу, може працювати окремо і незалежно.

Головною задачею декомпозиції є отримання модулів, які всередині максимально зосереджені на вирішення однієї певної задачі (принцип єдиної відповідальності), та мінімально пов'язані з іншими модулями.

Для отримання модулів, які відповідають головній задачі декомпозиції, вони повинні мати ряд властивостей:

1. Модуль повинен цілком і повністю реалізовувати певний функціонал і містити в собі все для того, щоб виконувати єдину задачу.

2. Результат роботи модулю не повинен залежати від роботи інших модулів, він повинен залежати лише від вхідних даних.

3. Модуль повинен мати слабкі інформаційні зв'язки з іншими модулями. Тобто потрібно уникати прямих викликів між модулями.

Головною проблемою в проектуванні модульної архітектури є послаблення зв'язків між модулями. Для того, щоб максимально зменшити кількість зв'язків між модулями потрібно слідувати наступним порадам:

1. Використовувати принцип спеціалізованих інтерфейсів [2, с. 200]. Тобто потрібно абстрагувати частини модулю в спеціалізовані інтерфейси і винести їх у фасад модулю.

2. Використовувати принцип інверсії залежностей [2, с. 191]. З даним принципом використовуються такі шаблони проектування як: фабричний метод, абстрактна фабрика, впровадження залежностей, інверсія контролю.

3. Надавати перевагу обміну повідомленнями між модулями замість прямих залежностей. З даним принципом використовуються шаблони проектування як: спостерігач, посередник, команда. Суть в тому, що модуль або оповіщає інші модулі, які підписалися на розсилку повідомлень, або використовує посередника для зв'язку з іншими модулями.

Отже, в даній статті розглянуто модульну архітектуру як основу для створення програмного забезпечення. Також було розглянуто переваги, які дає модульність системи, принципи створення модульності системи та поради до створення модулів.

Список використаних джерел:

1. Knoernschild K. Patterns of Modular Architecture [Electronic resource] – 2012 – Refcard #166 – Access mode: https://dzone.com/refcardz/patterns-modular-architecture

2. Мартин Р. С. Принципы, паттерны и методики гибкой разработки на языке C#. / Мартин Р. С., Мартин М. – СПб.: Символ-Плюс, 2011. – 768 с.