05.11.2022 г.

# Модульный принцип разработки ПО

## Основные критерии оптимизации модулей

Оптимизация программного кода - это модификация программ, выполняемая оптимизирующим компилятором или интерпретатором с целью улучшения их характеристик, таких как производительности или компактности без изменения функциональности.

Задачи оптимизации:

1. Подготовить такой набор тестов и применить к ним ПС (программное средство), чтобы обнаружить в нем большее число ошибок;

2. Определить момент окончания отладки ПС или отдельные его компоненты.

Основная цель модульного тестирования: удостовериться в соответствии требованиям каждого отдельного модуля системы перед тем, как будет произведена его интеграция в состав системы.

Основные задачи модульного тестирования:

1. Поиск и документирование несоответствий требованиям;
2. Поддержка разработки и рефакторинга низкоуровневой архитектуры системы и межмодульного взаимодействия;
3. Поддержка рефакторинга модулей;
4. Поддержка устранения дефектов и отладки.

## Информационная закрытость. Связность. Виды связности

Принцип информационной закрытости - содержание модулей должно быть скрыто друг от друга. Модуль должен определяться и проектироваться так, чтобы его содержимое (процедуры и данные) было недоступно тем модулям, которые не нуждаются в такой информации (клиентам).

Информационная закрытость означает следующее:

1. Все модули независимы, обмениваются только информацией, необходимой для работы;
2. Доступ к операциям и структурам данных модуля ограничен.

Достоинства информационной закрытости:

1. Обеспечивается возможность разработки модулей различными, независимыми коллективами;
2. Обеспечивается легкая модификация системы (вероятность распространения ошибок очень мала, так как большинство данных и процедур скрыто от других частей системы).

Идеальный модуль играет роль «черного ящика», содержимое которого невидимо клиентам. Он прост в использовании — количество «ручек и органов управления» им невелико (аналогия с эксплуатацией телевизора). Его легко развивать и корректировать в процессе сопровождения программной системы. Для обеспечения таких возможностей система внутренних и внешних связей модуля должна отвечать особым требованиям.

Связность модуля (Cohesion) - это мера зависимости его частей. Связность - внутренняя характеристика модуля. Чем выше связность модуля, тем лучше результат проектирования, то есть тем «черней» его ящик (капсула, защитная оболочка модуля), тем меньше «ручек управления» на нем находится и тем проще эти «ручки».

Для измерения связности используют понятие силы связности (СС). Существует 7 типов связности:

1. Связность по совпадению (СС=0). В модуле отсутствуют явно выраженные внутренние связи.
2. Логическая связность (СС=1). Части модуля объединены по принципу функционального подобия. Например, модуль состоит из разных подпрограмм обработки ошибок. При использовании такого модуля клиент выбирает только одну из подпрограмм. Недостатки:
   1. Сложное сопряжение;
   2. Большая вероятность внесения ошибок при изменении сопряжения ради одной из функций.
3. Временная связность (СС=3). Части модуля не связаны, но необходимы в один и тот же период работы системы. Недостаток: сильная взаимная связь с другими модулями, отсюда – сильная чувствительность внесению изменений.
4. Процедурная связность (СС=5). Части модуля связаны порядком выполняемых ими действий, реализующих некоторый сценарий поведения.
5. Коммуникативная связность (СС=7). Части модуля связаны по данным (работают с одной и той же структурой данных).
6. Информационная (последовательная) связность (СС=9). Выходные данные одной части используются как входные данные в другой части модуля.
7. Функциональная связность (СС=10). Части модуля вместе реализуют одну функцию.

## Сцепление модулей. Типы сцепления модулей

Сцепление модулей - это мера относительной независимости модулей.

Слабое сцепление определяет высокий уровень независимости модулей.

Модули являются полностью независимыми, если каждый из них не содержит о другом никакой информации. Чем больше информации о другом модуле в них

используется, тем менее они независимы и тем более сцеплены.

Независимое сцепление возможно, если модули не вызывают друг друга и не обрабатывают одну и ту же информацию.

Модули сцеплены по данным, если они имеют общие простые элементы данных, которые передаются от одного модуля к другому как параметры. (пример, структура – массив, а передается в качестве параметра элемент массива; при этом изменения в структуре данных не повлияют на другой модуль).

Модули со сцеплением по данным не имеют общей области данных (глобальных переменных).

Модули сцеплены по образцу, если в качестве параметров используются структуры данных (например, в качестве параметра передается массив). Недостаток: оба модуля должны содержать информацию о другом модуле (внутренней структуре данных), т.е. изменения должны отображаться и в др. модуле.

Модули сцеплены по общей области, если они разделяют одну и ту же глобальную структуру данных.

Модули сцеплены по управлению, если какой-либо из них управляет решениями внутри другого с помощью передачи флагов, переключателей и т.д.

Модуль сцеплен по внешним ссылкам, если у него есть доступ к данным другого модуля через внешнюю точку входа.

Модули сцеплены по кодам, если коды их команд объединены друг с другом (используют общий участок памяти).

Если модули косвенно обращаются друг к другу (через другие модули), то между ними также существует сцепление.

Вывод: сцепление модулей зависит от спроектированной структуры данных и способов взаимодействия между модулями. Необходимо использовать простые параметры и не применять глобальных данных.

# Конструирование ПО

## Рефакторинг. Определение причины и цели. Приёмы рефакторинга.

Рефакторинг – это контролируемый процесс улучшения кода без написания новой функциональности.

Результат рефакторинга – это чистый код и простой дизайн.

Чистый код – это код, который просто читать, понимать и поддерживать. Чистый код улучшает предсказуемость разработки и повышает качество продукта.

Приёмы рефакторинга:

1. Составление методов;
2. Перемещение функции между объектами;
3. Организация данных;
4. Упрощение условных выражений;
5. Упрощение вызовов методов;
6. Решение задач обобщения.

1) Состоит из извлечения метода, встраивания метода, извлечение переменной, встраивание переменной, замена переменной, метод возвращения переменной, расщепление переменной, удаление присваиваний параметром, замена метода объектом метода, замена алгоритма.

2) перемещение метода, перемещение поля, извлечение класса, встраивание класса, скрытие делегирования, введение внешнего метода, введение локального расширения.

3) самоинкапсуляция поля, замена простого поля объектом проблема: в классе или группе классов есть свое... , замена значения ссылкой проблема: есть много одинаковых экземпляров одного класса, которого можно заменить одним объектом, решение: превратить ссылку, замена ссылки значением, замена поля массива объектом, дублирование видимых данных, замена однонаправленный связи в двунаправленный, замена двунаправленной связи однонаправленной.

4) инкапсуляция поля, инкапсуляция коллекции, замена кодирования типа классом, замена кодирования типа подклассами, замена кодирования типа состоянием ... , замена подкласса полями, упрощение условных выражений.

5) разбиение условного оператора, объединение условных операторов, объединение дублирующихся операторов, удаление управляющего флага, замена вложенных условных операторов граничным оператором, замена условного оператора полиморфизмом, введение null объекта, введение проверки утверждения.

<https://oop.afti.ru/materials/212>  
<https://speakerdeck.com/baramiyadenis/liektsiia-13-riefaktoringh-ch-dot-2?slide=17>

23.03.2023

<https://infourok.ru/lekcii-po-discipline-razrabotka-programmnyh-modulej-5219726.html>

06.04.2023

<https://intuit.ru/studies/courses/691/547/lecture/12371>