Р. Я. Слука

МЕТОДЫ ОЦЕНКИ ГИБКОСТИ АРХИТЕКТУРЫ ПРИЛОЖЕНИЙ

Рассматриваются некоторые принципы построения архитектуры приложения и исследуется их влияние на программный код, а также методы, которыми можно измерить полезность этого влияния.

Введение

Существуют различные принципы для построения архитектуры приложения. В основе них лежит опыт ведущих специалистов в области разработки программного обеспечения. Исследованы примеры, с целью найти методы оценки этих принципов.

I. Принцип подстановки Барбары Лисков

Пусть $q(x)$ является свойством, верным относительно объектов x некоторого типа T. Тогда $q(y)$ также должно быть верным для объектов y типа S, где S является подтипом типа T.

В приложении есть обработка запросов на доставку еды из магазина. Строка запроса: "store=(smth)/menu=(smth)/address=(smth)". Есть обработчик, настроенный на приём запроса такого формата. Назовём формат базовым типом. Запрос для любого магазина должен наследовать тип базового. Однако, один из разработчиков при добавлении нового магазина изменил в строке запроса *menu* на $food$. и намерен использовать тот же обработчик. Однако теперь этот запрос не может быть подтипом базового запроса, т.к. он нарушает принцип подстановки (см.рис.1.). Теперь для корректной работы программы нам необходимо добавить условие $if(store =' other')$. Вывод: при невыполнении данного принципа нам необходимо добавить по одному лишнему условию для каждого неправильного подтипа. Также есть нарушение другого принципа - открытости/закрытости. Обработчик, использующий иерархию классов с нарушениями принципа Лисков, помимо оперирования ссылкой на базовый класс, также вынужден знать и о подклассе.

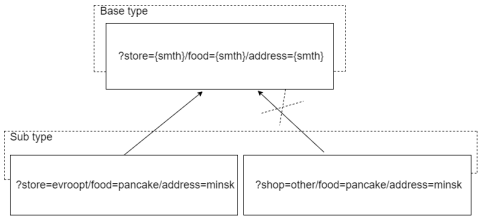

Рис. 1 – Иерархия наследования

II. Принцип устойчивых зависимостей

Зависимости должны быть направлены в сторону устойчивости [1].

Комоненты в системе должны быть разной метрики устойчивости. Иначе, если все компоненты будут устойчивы, такую систему тяжело будет изменить. Устойчивость можно измерить по количеству входящих и выходящих из компонента зависимостей по формуле $I = \frac{out}{in+out}$. Где *out* - число выходящих зависимостей, *in* - входящих (см.рис.2.).

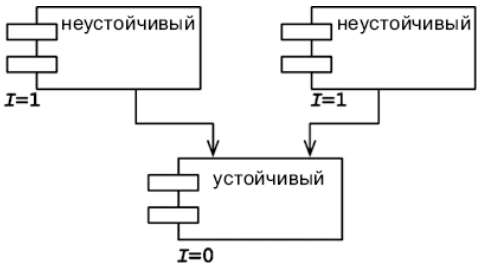

Рис. 2 – Система с тремя компонентами

При нарушении принципа компоненты ниже, будут иметь метрику устойчивости больше чем у верхних. Тогда изменение нижних будет усложнено и должно будет согласовываться со всеми верхними компонентами.

III. Выводы

Были найдены методы оценки принципов, как они влияют на изменение программного кода. На основе исследований можно создать статический анализатор кода, контролирующий соблюдение принципов.

1. Чистая архитектура. Искусство разработки программного обеспечения./ Р. Мартин. – СПб.: Питер, 2019. – 352 с.

Слука Роман Янушевич, магистрант кафедры информационных технологий автоматизированных систем БГУИР, roma.sluka.97@mail.ru.

Научный руководитель: Таранчук Валерий Борисович, профессор кафедры компьютерных технологий и систем ФПМИ БГУ, доктор физико-математических наук, taranchuk@bsu.by.