АНАЛИЗ ПРИМЕНЕНИЯ КОНЦЕПЦИИ SOLID

Примеров М. В., Солодухина Е. Е.
Научный руководитель – к.т.н, доц. Филиппенко И.В.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, пр. Науки, 14, каф. АПОТ, тел. (057) 702-13-26)
primerovmax@gmail.com, catherinesolodoukhina@gmail.com

The given work describes the conception for effective code writing, managing and refactoring. The conception contains five object-oriented principles chosen by Robert C. Martin.

SOLID – мнемотический акроним, сочетающий в себе пять принципов объектно-ориентированного программирования и проектирования, разработанных Робертом Мартином в 2009 году. Использование данной концепции подразумевает повышение вероятности написания программистом легко поддерживаемой, расширяемой и отлаживаемой системы. Аббревиатуру SOLID составляют:

The Single Responsibility Principle (SRP) – принцип единственной ответственности. Данный принцип обозначает, что каждый класс должен иметь только одну ответственность, и данная ответственность должна быть полностью инкапсулирована в класс. А так же, все методы класса связаны с этой ответственностью.

The Open Closed Principle (OCP) – принцип открытости/закрытости. Использование этого принципа подразумевает, что классы открыты для расширения функциональности, и закрыты – для её изменения. Само расширение функциональности осуществляется с помощью наследования расширяемого класса. Для обеспечения данного принципа проектирование классов должно обеспечивать то, что для изменения их поведения не придется изменять их код. Во всей системе классов должна обеспечиваться гибкость.

The Liskov Substitution Principle (LSP) – принцип подстановки Барбары Лисков. Принцип был предложен Барбарой Лисков в 1987 году.

Формулировка Лисков была следующей: «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».

Существует также альтернативное определение принципа, описанное Робертом С. Мартином: «Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом».

Таким образом, данный принцип подразумевает замену базовых классов их наследниками без возникновения ошибок с дополнением, но не изменением базового. Кроме того, данный принцип запрещает

использование оператора newметодов класса-наследника и обеспечивает формирование простой иерархии наследования.

The Interface Segregation Principle (ISP) – принцип разделения интерфейса. Принцип, следование которому позволяет системе оставаться гибкой, расширяемой и пригодной для рефакторинга.

Определение Роберта С. Мартина было следующим: «Программные сущности не должны зависеть от методов, которые они не используют».

Следование этому принципу подразумевает отказ от многофункциональных классов и использование большого количества интерфейсов, содержащих специфический функционал, который должен быть минимальным.

The Dependency Inversion Principle (DIP) – принцип инверсии зависимостей. Принцип подразумевает преобладание зависимостей классов от абстракций. Все модули должны зависеть от абстракций. Абстракции же не должны зависеть от деталей, но детали должны зависеть от абстракций. Модули верхних уровней не должны зависеть от модулей нижних уровней.

Выводы. В данном докладе были рассмотрены пять основных принципов концепции SOLID. Использование данной концепции:

1) существенно повышает простоту и читаемость кода, что обеспечивается принципом SRP, 2) обеспечивает совместимость будущих версий с предыдущими и отсутствие ошибок во взаимодействии классов, разрабатываемых различными командами программистов за счет использования OCP, 3) упрощает систему наследования при использовании LSP, позволяет системе оставаться гибкой и, в то же время, 4) легко расширяемой и затрачивать минимальное количество памяти за счёт минимизации интерфейсов, 5) упрощает проектирование системы классов и абстракций, которые они реализуют.

Список источников:

1. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс Быстрая разработка программ. Принципы, примеры, практика — Вильямс, 2004, ISBN 5-8459-0558-3, ISBN 0-13-597444-5.

2. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# — Символ-Плюс, 2011, ISBN 5-93286-197-5, ISBN 978-5-93286-197-4, ISBN 0-13-185725-8, ISBN 978-0-13-185725-4.

3. Fenton, Steve (2017). Pro TypeScript: Application-Scale JavaScript Development. p. 108. ISBN 9781484232491.