13. Об утверждении государственных требований к профессиональной переподготовке и повышению квалификации государственных гражданских служащих Российской Федерации: Постановление Правительства РФ от 06.05.2008 № 362 (ред. от 02.06.2016) // Собрание законодательства РФ. 12.05.2008. № 19. Ст. 2194. П. 2, ч. 1.

14. *Латышев В. Л.* Интеллектуальные обучающие системы: теория и технология создания и применения. – М.: Образование и информатика, 2003.

15. Бюрократия и власть в новой России: позиции населения и оценки экспертов: аналитический доклад. – М.: Центр комплексных социальных исследований Института социологии РАН, 2005. 98 с.

16. *Роберт И. В.* Информационно-коммуникационная предметная среда // Ученые записки ИУО РАО. 2001. № 5. С. 3–30.

IT usage for professional education of civil servants

***Latyshev Vladimir Leonidovich**, doctor of Pedagogic Sciences, professor of Moscow Aviation Institute (State University of Aerospace Technologies)*

***Illarionova Nataliya Gennadyevna**, post-graduate student of sociological faculty of Moscow State University*

The article describes IT usage for professional education of civil servants. Results of an opinion poll evaluating professional competence of civil servants are also shown.

Keywords: information and communication technology (ICT), distance learning programs, professional training, professional retraining, refresher training, civil servants, professional education.

УДК 004.053, 004.054

РАЗРАБОТКА МЕТРИК СЛОЖНОСТИ КОДА МОДУЛЯ ТЕСТОВ

***Галина Михайловна Рудакова**, канд. физ.-мат. наук,*
профессор кафедры информационно-управляющих систем,
email: gmrfait@gmail.com,
***Дмитрий Олегович Кожевников**, аспирант,*
кафедра информационно-управляющих систем,
email: d.o.kozhevnikov@gmail.com,
Сибирский государственный аэрокосмический университет
им. академика М. Ф. Решетнёва,
http://www.sibsau.ru

В статье проводится анализ проблем модульного тестирования и разработка требований к построению метрики, применимой для автоматизированного измерения сложности кода модульных тестов.

Ключевые слова: модульное тестирование; слабая связность; паттерны проектирования; внедрение зависимостей; IoC-контейнер; тестовый двойник; конструктор; рефакторинг.
DOI: 10.21777/2500-2112-2017-2-33-36


Г.М. Рудакова


Д.О. Кожевников

В разработке программного обеспечения в настоящее время широко применяются модульные тесты. Такие тесты должны соответствовать определенным требованиям. Так как от модульных тестов требуется высокая скорость работы, хорошим тестом обычно считается простой тест. Однако если для малого числа тестов их сложность можно оценить вручную, то при большом количестве тестов эта задача усложняется.

Тем не менее наблюдаемый в последние годы рост популярности модульных тестов связан почти исключительно с распространением так называемых «легких» методологий, и особенно с экстремальным программированием. Начало этой тенденции положил Кент Бек своей книгой «Extreme Programming Explained», увидевшей свет в 1999 году, где, помимо прочего, были сформулированы основные идеи Test-Driven Development (TDD).

Структура типичного модульного теста включает три секции, описываемые паттерном AAA (Arrange-Act-Assert). В секции Arrange находится подготовка тестовых данных и конфигурация объекта тестирования, формируется тестовый набор, в секции Act выполняется тестируемое действие, в секции Assert находятся проверки ожидаемых результатов действия.

Проверка последних блоков Act и Assert на сложность весьма проста. Если в модульном тесте более одного тестируемого метода, значит, его следует упростить. Проблема заключается в том, что для блока Arrange однозначных критериев не существует. Неизвестно, сколько объектов можно добавить в модульный тест без снижения скорости его работы. Для решения этой проблемы необходимо разработать метрику, которая даст численное выражение сложности теста в зависимости от содержащихся в нем объектов.

Основным методом измерения сложности кода является использование разнообразных метрик. Общая классификация метрик, которая наиболее часто встречается в научной литературе, представлена на рис. 1. Из рисунка видно, что классификация метрик подразумевает шесть основных классов. На рис. 2 представлена классификация метрик С. Чидамбера и К. Кемерера.

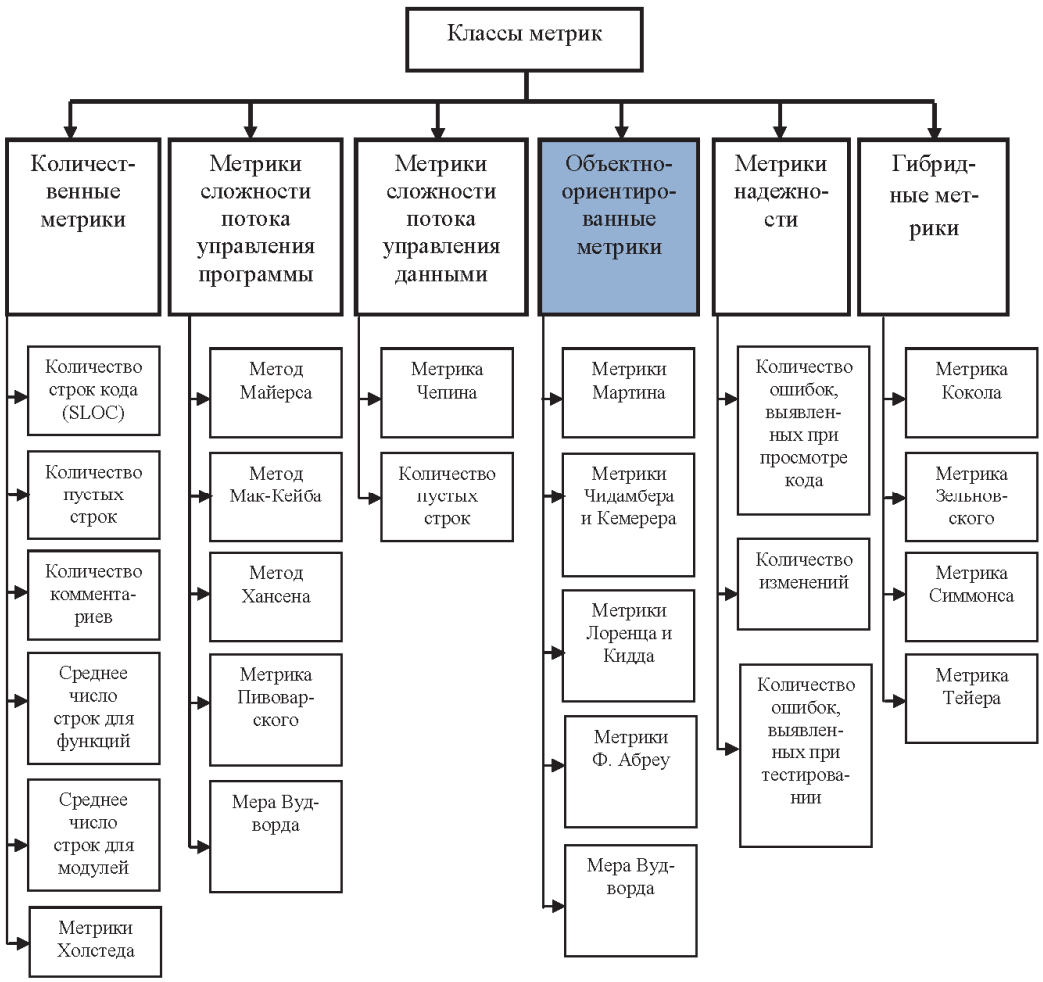

Рис. 1. Классификация метрик

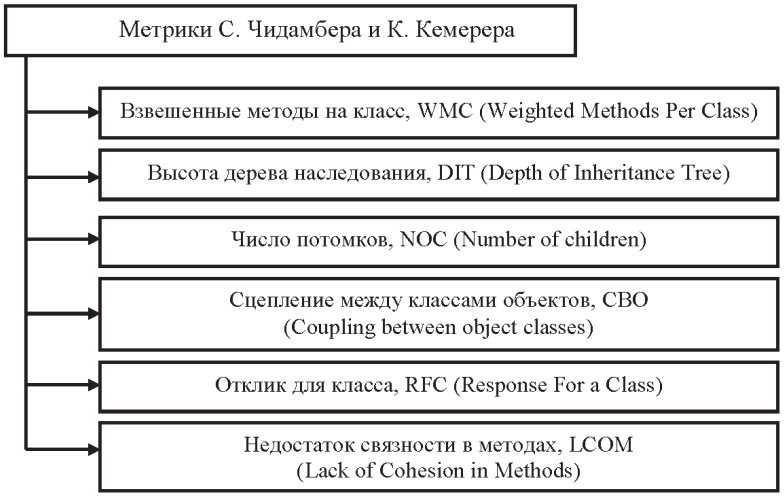

Рис. 2. Классификация метрик С. Чидамбера и К. Кемерера

В 1994 году С. Чидамбер и К. Кемерер предложили шесть проектных метрик, ориентированных на классы. Класс представляет собой основной, фундаментальный элемент объектно-ориентированной информационной системы.

Существует огромное количество разнообразных метрик программного обеспечения. Но большинство из них не дают никакой полезной информации, если их применить к модульным тестам. Так, например, цикломатическая сложность, которую измеряет метрика Мак-Кейба, для любого корректно написанного модульного теста будет равна единице. Далее, есть много метрик, измеряющих поведение тестов. Здесь статистически измеряется число отказов, количество найденных ошибок, время выполнения тестов etc. Но и эти показатели не могут служить признаком сложности теста.

Сформулируем основные требования к метрике измерения сложности модульных тестов.

Прежде всего, нужно выделить измеряемые величины, с которыми будет работать метрика. В модульном тесте можно отметить следующие интересующие нас элементы:

- Stub – объект-заглушка, не обладающий никакой функциональностью;
- Mock – также объект-заглушка, но, в отличие от Stub'а, Mock позволяет настраивать свое поведение;
- Target – таким образом отмечается метод, функциональность которого нам нужно протестировать;
- Target Call – вызов тестируемого метода. Как правило, происходит в блоке Act;
- Assert – проверка работы модульного теста. Assert сравнивает ожидаемый результат работы тестового метода с полученным в процессе выполнения. Именно этот элемент определяет успешно или неуспешно завершилось выполнение тест.

Помимо того, что модульные тесты должны отражать функциональность тестируемого программного продукта, основное требование – скорость работы. Разработчик должен иметь возможность запустить набор тестов в любой момент времени и выполниться этот набор должен настолько быстро, насколько это возможно. Отсюда первое требование к метрике сложности: метрика должна измерять количество Target, Target Call и Assert в тесте. В идеальном случае эти значения должны быть единичными. Это связано с тем, что отдельный тест должен покрывать один и только один тестируемый метод. Соответственно, есть какая-то инициализация, есть один вызов одного Target'а и есть один Assert.

Инициализация проходит с использованием таких элементов, как Stub и Mock. И для них уже четкого критерия нет. Известно, что чем меньше тест, тем лучше. Но предельное число Stub'ов и Mock'ов, при достижении которого с уверенностью можно

сказать, что данный тест из категории «хороших» перешел в категорию «плохих», к сожалению, неизвестно.

Есть возможный путь решения этой проблемы, но он слишком обширен для этой темы. Необходимо собрать достаточно большую базу из модульных тестов, которые были использованы в известных проектах с открытым исходным кодом. Далее эксперты должны оценить качество кода, а затем можно посчитать значение метрики для тех тестов, которые эксперты признали «хорошими». Таким образом будет известно среднее значение для условно хороших тестов, и его можно будет применять в качестве своеобразного эталона.

Литература

1. *Кожевников Д. О.* Актуальные проблемы организации модульного тестирования классов программного кода / Образовательные ресурсы и технологии. 2014. № 1 (4). С. 134–142.

2. *Рудакова Г. М., Кожевников Д. О.* Причины и движущие силы поэтапной смены подходов внутри объектно-ориентированной парадигмы программирования / Образовательные ресурсы и технологии. 2014. № 1 (4). С. 68–78.

Development of the test module code complexity metrics

***Galina Mihaylovna Rudakova**, PhD, professor, Siberian State Aerospace University*

***Dmitry Olegovich Kozhevnikov**, postgraduate, Siberian State Aerospace University*

The article analyses problems unit testing and development requirements for building metrics applicable to automated unit test code complexity measurement.

Keywords: unit testing, low coupling, design patterns, dependency injection, IoC-container, test double, constructor, refactoring.

УДК 334.024, 316.472.43

БЮРОКРАТИЯ С ТОЧКИ ЗРЕНИЯ ТЕОРИИ САМООРГАНИЗАЦИИ

***Ибрагим Эсенович Сулейменов**, д-р хим. наук, канд. физ.-мат. наук,*
профессор кафедры ИКТ, зав. лабораторией наноэлектроники,
e-mail: esenych@yandex.ru,
Алматинский университет энергетики и связи,
http://www.aipet.kz
***Ануар Абаевич Нуртазин**, преподаватель,*
e-mail: anuar_nurtazin@mail.ru,
Крымский федеральный университет,
http://www.cfuv.ru,
Западно-Казахстанский государственный медицинский университет,
http://www.zkgmu.kz,
***Олег Аршавирович Габриелян**, д-р филос. наук, профессор,*
e-mail: gabroleg@mail.ru,
Таврическая академия Крымского федерального университета,
http://www.ta.cfuv.ru,
***Дина Бернаровна Шалтыкова**, канд. хим. наук, ст. преп.,*
e-mail: dina_65@mail.ru,
Алматинский университет энергетики и связи,
http://www.aipet.kz,