4. *Мартин Р.С.* Принципы, паттерны и методики гибкой разработки на языке C#. / Р.С. Мартин. – СПб.: Символ, 2012. – 768 с.

5. *Эванс Э.* Предметно-ориентированное проектирование. Структуризация сложных программных систем/ Э. Эванс. – М.: ООО «И.Д. Вильямс», 2011.– 448 с.

6. *Kozhevnikov D.O., Rudakova G.M.* History of structured programming and origin of objective-oriented paradigm. //2nd International Academic Conference on Applied and Fundamental Studies, March 8-10, 2013., St. Louis, USA. Publishing House «Science & Innovation Center», 2013. P. 174-180.

7. *Kozhevnikov D.O., Rudakova G.M*. Evolution of the object-oriented paradigm in software development. // 2nd International Academic Conference on Applied and Fundamental Studies, March 8-10, 2013. – St. Louis, USA. Publishing House «Science & Innovation Center», 2013. P. 180-186.

The reasons and driving forces of stage-by-stage change of approaches in an object-oriented paradigm of programming

Galina Mikhailovna Rudakova, PhD, professor
Siberian State Technological University, Chair of Information Technologies.

Dmitry Olegivich Kozhevnikov, postgraduate
Siberian State Technological University, Chair of Information Technologies.
http://www.kit-sibstu.ru/

In work the characteristic of the major cognitive events of history of a paradigm, the prerequisite, authorship and importance of its main scientific concepts are presented. The following periodization of evolution of the object-oriented paradigm (OOP) in the theory of software development is offered: academic OOP; early OOP; mature OOP; modern OOP. The progress is caused by aspiration to engage human potential in development of evolving program complexes.

Key words: object-oriented paradigm, software, concepts of design patterns, refactoring.

УДК 519.6

ИСПОЛЬЗОВАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ МЕТРИК ДЛЯ АНАЛИЗА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

***Ольга Алексеевна Самойликова**, магистр*
Тел.: 8 (902) 992-54-57, e-mail: olga.samoylikova@yandex.ru
***Галина Михайловна Рудакова**, к.ф.-м.н., профессор,*
зав .кафедрой информационных технологий
Тел.: +7 (983) 150 7529, e-mail: gmrfait@gmail.com
Сибирский государственный технологический университет

В работе рассматривается использование объектно-ориентированных метрик для анализа и количественной оценки программного обеспечения. Количественная оценка сложности информационной системы производится при помощи метрик, которые принято подразделять на семь классов. Наиболее подробно описывается класс объектно-ориентированных метрик. Особое внимание уделяется набору метрик Чидамбера и Кемерера.

Ключевые слова: метрики, информационная система, проект, класс, метод, инкапсуляция, наследование, экземплярная переменная.набор метрик Чидамбера и Кемерера.

О.А. Самойликова

На сегодняшний день ситуация в мире информационных технологий складывается таким образом, что большинство крупных проектов разрабатываются на объектно-ориентированных языках программирования. Обеспечение качества проекта является главной задачей, как руководителей проекта, так и самих разработчиков. Для эффективной оценки программного обеспечения используются метрики.

Определение метрики в контексте информационных технологий, согласно стандарту ISO 14598 звучит как количественный масштаб и метод, который может использоваться для измерения. Другое определение метрики – мера, позволяющая получить численное значение некоторого свойства программного обеспечения [1]. Объектно-ориентированные метрики вводятся с целью: улучшить понимание качества продукта, оценить эффективность процесса конструирования, улучшить качество работы на этапе проектирования.

Г.М. Рудакова

Для любой информационной системы определяемые наборы метрик должны ориентироваться на особенности и уникальные характеристики. Большинство метрик, были получены опытным пут`м в ходе работы над проектами информационных систем.

Важно учитывать, не только особенности проекта при выборе метрик, но и его специфику.

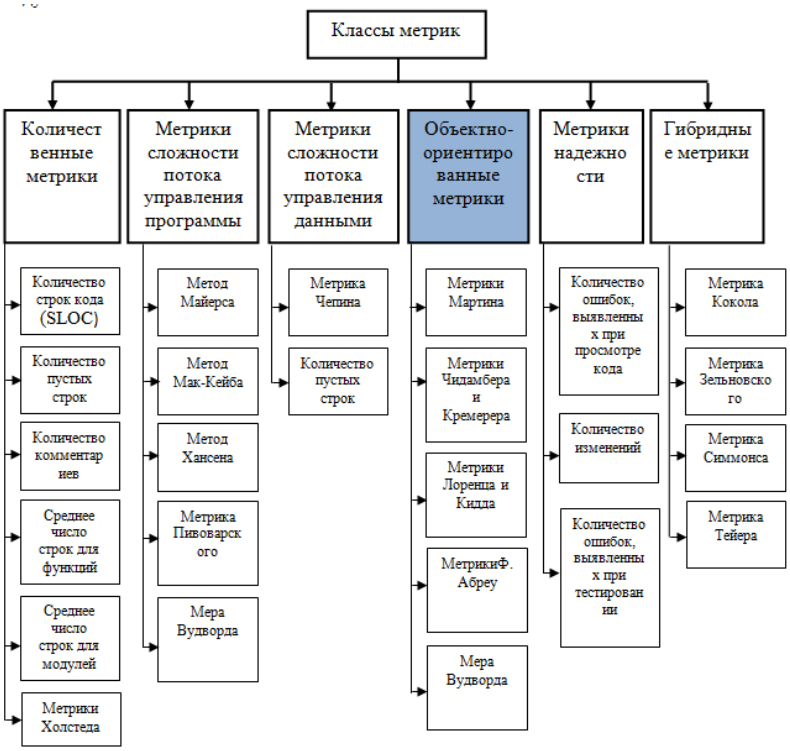

Рис. 1. Классификация метрик

С точки зрения метрик выделяют пять характеристик объектно-ориентированных систем: локализацию, инкапсуляцию, информационную закрытость, наследование и способы абстрагирования объектов. Каждая из этих характеристик оказывает непосредственное влияние на метрики. Общая классификация метрик, которая наиболее часто встречается в научной литературе, представлена на рис. 1. Из рисунка видно, что классификация метрик состоит из семи основных классов. В каждом классе указаны наиболее часто используемые виды метрик. Самыми простыми в использовании считаются количественные метрики. В данный класс входят такие метрики как: количество пустых строк; количество комментариев; процент комментариев (отношение числа строк, содержащих комментарии к общему количеству строк, выраженное в процентах); среднее число строк для функций (классов, файлов); среднее число строк, содержащих исходный код для функций (классов, файлов); среднее число строк для модулей.

На схеме, представленной на рис.1 видно, что каждый класс обладает набором метрик. Метрик, входящих в определ`нный класс, существует большое количество, поэтому в схеме отображены наиболее часто встречающиеся наборы метрик, однако далеко не все.

Для анализа и качественной оценки проектов, созданных при помощи языков объектно-ориентированного программирования, используется класс объектно-ориентированных метрик. В данном классе наиболее часто используемыми являются набор метрик Мартина, набор метрик Чидамбера и Кемерера.В статье более подробно будет рассмотрен набор мтерик Чидамбера и Кемерера [2].

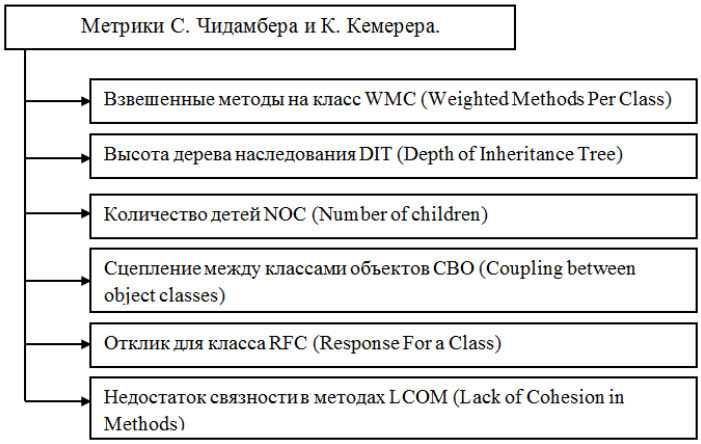

Рис.2. Шесть метрик С. Чидамбера и К. Кемерера

В 1994 году С. Чидамбер и К. Кемерер предложили шесть проектных метрик, ориентированных на классы [3]. Класс – представляет собой основной, фундоментальный элемент объектно-ориентированной информационной системы. Метрики для отдельного класса, иерархии классов и взаимодействия классов являются наиболее ценными для руководителя проекта, который оценивает качество работы. На рис. 2 представлена классификация метрик С. Чидамбера и К. Кемерера.

Первым видом метрик является метрика – взвешенные меры на класс (WMC). Использование данной меры осуществляется при помощи формулы:

$$WMC = \sum_{i=1}^{n} C_i,$$ где С – класс; n – методы со со сложностью $c_1, c_2, \ldots c_n$

Сложность методов может оцениваться любой метрикой сложности. Основным условием является то, что данная метрика должна быть нормализована так, чтобы номинальная сложность для метода принимала значение 1. Количество методов и их сложность являются индикатором затрат на реализацию и тестирование классов.

С ростом количества методов в классе его применение становится все более специфическим, темсамым ограничивается возможность многократного использования. По этим причинам метрика WMC должна иметь разумно низкое значение.

Также применяют упрощённую версию метрики. При этом полагают *Ci*= 1, и тогда WMC приравнивается к количеству методов в классе.

Для подсчёта количества методов в классе используют два противоположных варианта учёта.

1. Подсчитываются только методы текущего класса. Унаследованные методы игнорируются. Обоснование – унаследованные методы уже подсчитаны в тех классах, где они определялись.
2. Подсчитываются методы, определённые в текущем классе, и все унаследованные методы.

На практике применяются оба варианта. Основным условием применения является постоянное использование одного варранта учёта методов. Только в этом случае обеспечивается корректный сбор метрических данных [3].

Метрика WMC даёт относительную меру сложности класса. Можно предположить, что класс который имеет максимальное количество методов среди классов одного с ним уровня, является наиболее сложным.

Второй вид метрик – высота дерева наследования (DIT). Она определяется как максимальная длина пути от листа до корня дерева наследования классов. Пример дерева наследования классов схематически представлен на рис. 3.

Для примера дерева классов, представленного на рисунке 3, метрика DIT равна 4. Для отдельного класса DIT, это длина максимального пути от данного класса до корневого класса в иерархии классов. G

По мере роста DIT вероятно, что классы нижнего уровня будут наследовать много методов. Это приводит к трудностям в предсказании поведения класса. Высокая иерархия классов (большое значение DIT) приводит к большей сложности проекта, так как означает привлечение большего количества методов и классов.

Вместе с тем, большое значение DIT подразумевает, что многие методы могут использоваться многократно.

Третий вид метрик – количество детей NOC. Подклассы, которые непосредственно подчинены классу, называются его детьми. Значение NOC равно количеству детей, то есть количеству непосредственных наследников класса в иерархии классов. На рисунке 3 класс C_{11} имеет двух детей – подклассы C_{111} и C_{112}. В свою очередь подкласс C_{112} имеет одного наследника – C_{1121}. Количество детей характеризует потенциальное влияние класса на проект.

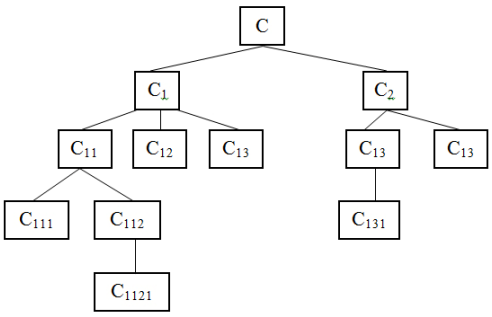

Рис. 3. Дерево наследования классов

По мере роста NOC возрастает количество тестов, необходимых для проверки каждого ребёнка.

Четвертый вид метрик – сцепление между классами объектов (CBO). Данная мера определяет количество классов, с которыми взаимодействует рассматриваемый класс. Данная метрика характеризует статическую составляющую внешних связей классов. С увеличением значения CBO многократность использования класса, уменьшается. Чем больше независимость класса, тем легче его повторно использовать в другом приложении. Высокое значение CBO усложняет модификацию и тестирование, которое следует за выполнением модификации.

Пятый вид метрик – отклик для класса (RFC). RFC — это количество методов класса плюс количество методов других классов, вызываемых из данного класса.

Метрика RFC является мерой потенциального взаимодействия данного класса с другими классами, позволяет судить о динамике поведения соответствующего объекта в системе. Данная метрика характеризует динамическую составляющую внешних связей классов. Если в С ростом RFC увеличивается сложность класса. Наихудшая величина отклика может использоваться при определении времени тестирования [3].

Шестой вид метрик – недостаток связности в методах (LCOM).Каждый метод внутри класса обращается к одному или нескольким свойствам. Метрика *LCOM* показывает, насколько методы не связаны друг с другом через свойства. Если все методы обращаются к одинаковым свойствам, то LCOM = 0.

Для наглядного представления использования метрик введём обозначения:

η – количество пар методов без общих экземплярных переменных;

θ – количество пар методов с общими экземплярными переменными;

I_j – набор экземплярных переменных, используемых методом *Mj*.

$$\eta = card\{I_{ij} | I_i \cap I_j = 0\}$$
$$\theta = card\{I_{ij} | I_i \cap I_j \neq 0\}$$

Тогда формула для вычисления недостатка связности в методах примет вид

$$LCOM = \begin{cases} \eta - \theta, \text{если } \eta > \theta \\ 0 \text{ в противном случае} \end{cases}$$

Связность методов внутри класса должна быть высокой. Если LCOM имеет высокое значение, то методы слабо связаны друг с другом через свойства. Это увеличивает сложность, что увеличивает вероятность ошибок при проектировании [3].

Высокие значения LCOM означают, что класс, вероятно, надо спроектировать лучше (разбиением на два или более отдельных класса). Любое вычисление LCOM помогает определить недостатки в проектировании классов, так как эта метрика характеризует качество упаковки данных и методов в оболочку класса. Связность в классе желательно сохранять высокой, то есть следует добиваться низкого значения LCOM.

Набор метрик Чидамбера-Кемерера является основой оценки проекта реализованного на языке объектно-ориентированного программирования. На данный момент времени существуют большое количество модификаций, которые предлагают оптимальные решения оценки проекта. В частности дополнением к набору метрик Чидамбера-Кемерера считается метрика – *поведенческая закрытость информации BIH* (Behaviourial Information Hiding)

Формула применения метрики:

BIH=WEOC/WIEOC,

где WEOC – взвешенные внешние операции на класс (фактически это WMC);
WIEOC – взвешенные внутренние и внешние операции на класс.

Если BIH = 1, класс показывает другим классам все свои возможности. Если BIH $\approx 0{,}2$ поведение класса скрывается от других классов. BIH может рассматриваться и как мера сложности. Сложные классы, имеют малые значения BIH. Простые классы имеют значения, близкие к 1.

Использование наборов метрик Чидамбера-Кемерера позволяет не только определять сложность проекта, но модифицировав определенные характеристики, учесть особенности разрабатываемого проекта, провести эффективный анализ информационной системы, выявить уязвимые места. В настоящее время при оценке проектов, реализованных на языках объектно-ориентированного программирования не достаточно использование одного набора метрик Чидамбера-Кемерера.

Наряду с рассмотренными метриками используются метрики Лоренца и Кидда (размер класса CS (Class Size), количество операций, добавленных подклассом NOA (Number of Operations Added by a Subclass), индекс специализации SI (Specialization Index), средний размер операции OSAVG (Average Operation Size), Сложность операции OC (Operation Complexity), среднее количество параметров на операцию NPAVG (Average Number of Parameters per operation), количествоописанийсценариев NSS (Number of Scenario Scripts), количествоподсистем NSUB (Number of SUB system)) [4].

Набор метрик *MOOD*(MetricsforObjectOrientedDesign), предложенный Ф. Абреу (фактор закрытости метода (MHF), фактор закрытости свойства (AHF), фактор наследования метода (MIF),фактор наследования свойства (AIF), фактор полиморфизма (POF),фактор сцепления (COF)) и многие другие. Большое количество метрик, разработанных для оценки информационных систем на языках объектно-ориентированного программирования, позволяет сформировать необходимый набор мер для оценки проекта.

Литература

1. Международныйстандарт ISO/IEC 25040:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process. [Электронный ресурс]. URL: http://webstore.iec.ch/preview/info_isoiec25040%7Bed1.0%7Den.pdf

2. *Орлов С.А.* Технологии разработки программного обеспечения. Разработка сложных программных систем: учебник. – СПб.: Питер, 2002. – 464 с.

3. *Черноножкин С.К.* Методы и инструменты метрической поддержки разработки качественных программ: автореферат, – Новосибирск, 1998.

4. *Богданов Д.В.* Стандартизация жизненного цикла программных средств. – СПб., 2000. – 210 с.

Using object-oriented metrics for software analysis

OlgaAlexeevnaSamoylikova, masterdegree
SiberianStateTechnologicalUniversity

Galina Mihaylovna Rudakova, PhD, professor
Siberian State Technological University,
Chair of Information Technologies.

In this paper discusses the use of object-oriented metricsfor analysis and quantification software.Quantitative evaluation complexity of the information system is made using metrics. Metrics are usually subdivided into seven classes. The most detailed description of the class of object-oriented metrics. Particular attention is paid to the set of metrics Chidamber and Kemerer.

Keywords: metrics, information system, project, class, method, encapsulation, inheritance, instance variables. set of metrics Chidamber and Kemerer.