УДК 004.412

УПРАВЛЕНИЕ КАЧЕСТВОМ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ ПРОГРАММНЫХ СИСТЕМ В ПРОЦЕССЕ ИХ РАЗРАБОТКИ

Рябченко В.В., Федяев О.И.
Донецкий национальный технический университет,
кафедра прикладной математики и информатики
E-mail: VladRia@mail.ru

Аннотация

Рябченко В.В., Федяев О.И. Управление качеством объектно-ориентированных программных систем в процессе их разработки. *В статье проведен анализ существующих методов измерения характеристик программного обеспечения и процесса его разработки. Предложен и обоснован метрический аппарат для оценки качества объектно-ориентированных программных проектов. Разработан инструментарий (программное средство «ОО- Measurement») для оценки характеристик качества программных систем на этапах проектирования, конструирования и сопровождения.*

Постановка и описание проблемы

За последние два десятилетия программная инженерия (ПИ) превратилась в одну из самых дорогостоящих индустрий. Жесткая конкуренция, сжатые сроки, высокая сложность и объёмность программного обеспечения (ПО), ограниченные ресурсы, обеспечение стандартов качества, необходимость долговременного сопровождения – текущие реалии и актуальные проблемы этой индустрии [1]. Объектно-ориентированная (ОО) парадигма программирования явилась важной вехой в развитии ПИ и позволила решить многие проблемы, повысив скорость разработки, управляемость и модифицируемость проектов. Качество проектов с точки зрения ПИ характеризуется лёгкостью разработки, тестирования, отладки, сопровождения и портируемости ПО.

Проблема заключается в создании средств измерения объективных характеристик качества продуктов ПИ, выявления недостатков в анализируемых ПС и выработки рекомендаций по улучшению ОО ПС на основе полученных оценок. В данной научной работе описывается метод (метрический аппарат) для оценки ОО ПС и управления их созданием на основе метрик. Доказывается, что этот метод анализирует все ключевые характеристики, влияющие на качество ПО, формирует объективные показатели в числовом виде, на основании которых предлагаются обоснованные рекомендации по корректировке процесса и продуктов проектирования. Для проведения автоматизированного анализа ОО ПС разработан программный инструментарий «OO-Measurement».

Состояние работ по анализу и управлению проектами на основе метрик

Для решения поставленной задачи необходимо выделить характеристики, оценивая которые, можно говорить о качестве ПО. Определенные результаты в решении этих проблем были получены американскими учеными в области теории и практики программной инженерии. Абреу Ф. в своих работах утверждает, что основополагающие концепции ООП и являются теми основными свойствами ОО ПС, прямо влияющими на их качество: инкапсуляция, наследование, полиморфизм, посылка сообщений [2]. Оцениваются эти свойства с помощью метрик – функций, которые дают числовые значения, зависящие от опорных, непосредственно измеряемых, характеристик проекта.

Набор из шести метрик Чидамбера С. и Кемирера К. [ссылка, см. Орлова, стр.295] - это первая попытка развития формальных показателей для ООП. Коллекция метрик Лоренца М. и Кидда Д. [ссылка, см. Орлова, стр.302] – это результат практического, промышленного

подхода к оценке ОО-проектов, который даёт неплохие результаты, объективно оценивая все основные характеристики объектно-ориентированной парадигмы программирования [2]. За основу метода анализа ОО ПС в данной научной работе взят набор метрик Ф. Абреу. Ниже описывается используемый метрический аппарат, как совокупность функций числовых показателей качества ОО-проектов.

Описание метода оценки программного обеспечения

Программное обеспечение, как объект разработки программной инженерии, обладает характеристиками, которые важно контролировать в процессе его создания. Для этого рекомендуется проводить периодические «измерения» проекта, которые направлены на получение числовых показателей, отражающих качество ПО. Эти показатели, как в итоге, так и в срезе времени дают на уровне управления представление о качественном развитии проекта и позволяют управлять им. Для каждой основополагающей концепции ООП приведены метрики для получения числовых показателей, с помощью которых в данной работе предлагается комплексная оценка качества ОО ПО. Объясняются значения этих показателей и даны рекомендации к улучшению качества на их основе, т.к. использование метрик предполагает чёткое понимание результатов измерений и дальнейших действий.

Инкапсуляция – свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации. Для расчёта степени инкапсуляции предлагается применять следующие две метрики:

Метрика 1. Фактор закрытости метода MHF (Method Hiding Factor)

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)} \cdot 100\ \%\ , \qquad (1)$$

где, $M_v(C_i)$ – количество видимых методов в классе C_i (интерфейс класса);

$M_h(C_i)$ – количество скрытых методов в классе C_i (реализация класса);

$M_d(C_i) = M_v(C_i) + M_h(C_i)$ – общее количество методов, определенных в классе C_i;

TC – количество классов в системе.

В числителе формулы MHF (1) – сумма закрытости всех методов во всех классах – процентное количество классов, из которых данный метод невидим. Знаменатель MHF – общее количество методов, определенных в рассматриваемой системе. С увеличением MHF уменьшаются плотность дефектов в системе и затраты на их устранение. Рекомендованное значение метрики – от 60 до 100% закрытых методов [3].

Метрика 2. Фактор закрытости свойства AHF (Attribute Hiding Factor) определяется по аналогии с формулой (1) как отношение сумма закрытости всех свойств во всех классах к общему количеству свойств, определённых в рассматриваемой системе. В идеальном случае все свойства должны быть скрыты и доступны только для методов соответствующего класса (AHF = 100%). Рекомендуется не меньше 95% закрытых свойств [3].

Наследование – свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Наследование исследуемой ПС характеризуют следующие две метрики:

Метрика 3. Фактор наследования метода MIF (Method Inheritance Factor)

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)} \cdot 100\%\ , \qquad (2)$$

где, $M_i(C_i)$ – количество унаследованных и не переопределенных методов C_i;

$M_a(C_i) = M_d(C_i) + M_i(C_i)$ – общее количество методов, доступных в классе C_i.

Числителем MIF является сумма унаследованных (и не переопределённых) методов во всех классах рассматриваемой системы. Знаменатель MIF – это общее количество доступных методов для всех классов. Умеренное использование наследования позволяет снизить плотность дефектов и затрат на доработку. Значение MIF = 0 указывает, что в системе отсутствует эффективное наследование. С увеличением MIF уменьшаются плотность дефектов и затраты на исправление ошибок. Однако очень большие значения MIF (75-85%) приводят к обратному эффекту. Рекомендовано иметь от 30 до 70% переопределенных методов.

Метрика 4. Фактор наследования свойства AIF (Attribute Inheritance Factor) определяется по аналогии с (2) как отношение суммы унаследованных (и не переопределенных) свойств во всех классах рассматриваемой системы к общему количеству доступных свойств (локально определенных и унаследованных) для всех классов. Рекомендовано иметь от 15 до 85% переопределенных свойств [3].

Полиморфизм – свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. Оценивается метрикой POF.

Метрика 5. Фактор полиморфизма POF (Polymorphism Factor)

$$POF = \frac{\sum_{i=1}^{TC} M_0(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]} \cdot 100\% \ , \qquad (3)$$

где, $M_0(C_i)$ – количество унаследованных и переопределённых методов в классе C_i;

$M_n(C_i)$ – количество новых (не унаследованных и переопределённых) методов в C_i;

$DC(C_i)$ – количество потомков класса C_i .

Формула (3) является отношением реального количества возможных полиморфных ситуаций для класса C_i к максимальному. Умеренное использование полиморфизма уменьшает как плотность дефектов, так и затраты на доработку [3]. Однако при POF > 15% возможен обратный эффект.

Посылка сообщений – свойство, характеризующее взаимодействие между классами в системе: связность, сцепленность и т.д. Это взаимодействие фиксируется функцией $is\_client(C_i, C_j)$ и говорит о наличие между классами отношения «клиент-поставщик».

$$COF = \frac{\sum_{i=1}^{TC} [\sum_{j=1}^{TC} is\_client(C_i, C_j)]}{TC^2 - TC} \cdot 100\% \ , \qquad (4)$$

Числитель COF фиксирует реальное количество сцеплений, не относящихся к наследованию. Знаменатель COF соответствует максимально возможному количеству сцеплений в системе между классами, но без рефлексии. С увеличением сцепления классов плотности дефектов и затрат на доработку также возрастают. Сцепления отрицательно влияют на качество ПО, их нужно сводить к минимуму [3]. Рекомендовано иметь не более 15% сцепленных классов.

Все шесть метрик, посчитанные на определенном этапе разработки, позволяют оценить в целом качество ПС и выявить определенные проблемы проектирования и реализации, которые в дальнейшем желательно устранить. Если для какой-либо характеристики метрика показывает выход за пределы рекомендуемых значений, то это означает отход разработчиков от основных концепций ООП. Для повышения качества разрабатываемого ПО следует обратить на это внимание, иначе в будущем проект столкнётся с такими проблемами как повышение вероятности появления ошибок в ПС, трудностями тестирования и сопровождения, увеличение затрат [4].

В стандартный метрический аппарат авторы вводят дополнительные метрики для анализа иерархической структуры ОО ПС, а именно: метрику 7 - Высота дерева наследования (ДН); метрику 8 - Ширина ДН; метрику 9 - Количество отношений наследования; метрику 10 - Количество классов [5].

Естественно, что для лучшего контроля необходимо периодически производить оценку разрабатываемого ПО. Формально процесс управления качеством выглядит следующим образом: P – программу, как объект разработки, развивают во времени по какой-нибудь модели проектирования (инкрементной, спиральной или др.) и получают последовательность прототипов $P_1, P_2, P_3, \ldots, P_n$. Вычисляя метрики для каждой версии, можно получить набор числовых характеристик её качества, который определяет табличную функцию $k = k(P_n)$. Эта функция показывает динамику изменения качества разрабатываемой ПС, благодаря которой разработчики имеют возможность контролировать тенденцию изменения качества ПС в ходе её разработки и вносить своевременно коррективы для устранения выявленных отклонений в спектре оцениваемых параметров качества k.

Программа «OO-Measurement» для автоматической оценки качества ПС

Для автоматического проведения оценки качества ПС был разработан программный инструментарий “OO-Measurement”, который получает значения рассмотренных метрик на основе анализе исходного кода программного продукта. Главное достоинство этого инструментария – это быстрое получение значений всех метрик с последующей всесторонней оценкой качества ОО ПС. Функциональные возможности инструмента “OO-Measurement” соответствуют описанному ранее метрическому аппарату: шесть метрик для оценки воплощения концепции ООП и четыре метрики, оценивающие иерархическую структуру диаграммы классов. Создана база данных, в которой возможно хранение результатов расчётов множества проектов для обнаружения тенденций развивающегося проекта. Реализованы средства визуализации графиков изменения качества во времени (рис.1) и показателей метрик (рис.2). Инструмент «OO-Measurement» демонстрирует эффективность предложенной технологии оценки качества для любых проектов, написанных в соответствии с объектно-ориентированной парадигмой программирования.

Рисунок 1 – График изменения качества инкапсулированности ПО во времени

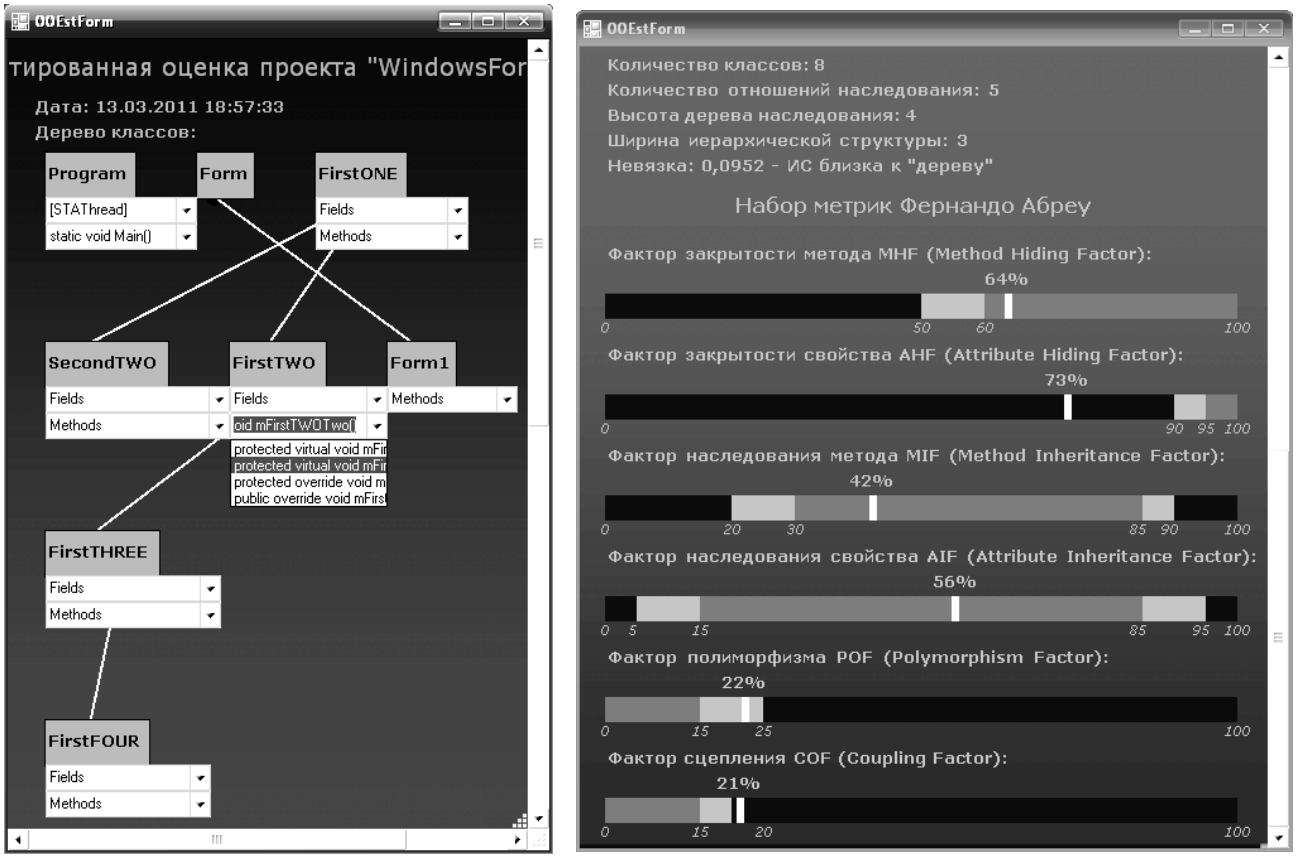

Рисунок 2 – Работа ПС «OO Measurement»

Выводы

В сфере информационных технологий самой актуальной проблемой является обеспечение качества программного обеспечения, а также качества процесса его разработки, тестирования и сопровождения. В данной статье рассмотрены существующие методы оценки качества и сложности объектно-ориентированных ПС, а также их недостатки. Предложен метрический аппарат, основанный на совокупности метрик, с помощью которого можно получать объективные показатели качества ПО. В качестве инструментария была разработана система “OO-Measurement” для всесторонней оценки качества объектно-ориентированного ПО, разрабатываемого как в профессиональной деятельности, так и в обучающем процессе.

Литература

1. Рябченко В.В., Федяев О.И. Методы оценки качества программных систем и процесса их разработки / ИКТ-2010 – Донецк: ДонНТУ, 2010. – 448 с.

2. Орлов, С.А. Технологии разработки программного обеспечения [Текст] / С. А. Орлов. – СПб.: Питер, 2002. – 464 с.

3. Jones, Capers. Applied Software Measurement; McGraw Hill, 3rd edition 2008; ISBN 978-0-07-150244-3. – 575 pages.

4. Kan, Stephen H. Metrics and Models in Software Quality Engineering, 2nd edition; Addison Wesley Longman, Boston, MA; ISBN 0-201-72915-6; 2003. – 528 pages.

5. Shari Lawrence Pfleeger. Software Metrics: A Rigorous and Practical Approach, Revised; Prentice Hall, ISBN-10: 9780137203840, 2008. – 282 pages.