КОНЦЕПЦИЯ АНАЛИТИЧЕСКОЙ ОЦЕНКИ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНЫХ КОМПОНЕНТОВ

Е.М. Лаврищева, А.М. Рожнов

Институт Программных Систем НАН Украины, просп. Глушкова, 40, Киев, 03187, Тел.: (044) 266 34–70, E-mail: lem@isofts.kiev.ua

Рассматривается проблема обеспечения качества программных компонентных систем. Дано определение модели качества таких систем. Предложена концепция и подход к аналитической оценке атрибутов характеристик качества компонентов. Получено аналитическое выражение для комплексной оценки качества компонента и системы.

Розглядається проблема забезпечення якості програмних компонентних систем. Визначено модель якості таких систем. Запропоновані концепція та підхід щодо аналітичної оцінки атрибутів характеристик якості компонентів. Отримано аналітичний вираз комплексної оцінки якості компонента та системи з комонентів.

Problem of quality for components program systems are discussed. Model of quality such systems are considered. Conception and approach of analytic estimation for quality attributers about components are proposed. Analytical formula for complex estimation of quality about component and systems are received .

Вступление

Разработка программных систем достигла такого уровня развития, что требуются инженерные методы программирования для обеспечения качества компонентов этих систем и является главной задачей разработчиков и заказчиков. Для ее решения сформировалась инженерия требований как дисциплина анализа, документирования и преобразования предложенных заказчиком требований к системе в требования к программному обеспечению (ПО). Инженерия требований и современные методы и средства инженерия качества – основные направления инженерии программирования компонентов и системы в целом [1-3].

Качество ПО, согласно ДСТУ [4–6], это совокупность свойств ПО, которые обеспечивают его способность удовлетворять потребности заказчика, в соответствии с его назначением. Свойства ПО соответствуют реализованным требованиям к характеристикам качества (функциональность, надежность и др.), анализ которых проводится путем их верификации и аттестации (валидации), а также проверкой соответствующих проектных решений на всех этапах жизненного цикла .

Анализ публикаций по проблеме качества показывает, что они относятся к ПО и системам, а работы по обеспечению качества программных компонентов ПО и их оценки практически отсутствуют. В данной работе ставится задача заполнить этот пробел и рассмотреть подходы к гарантии качества в компонентном программировании и предложить концепцию оценки качества совокупности компонентов системы, обладающей определенными свойствами и характеристиками, разработанными в соответствии с заданными заказчиком требованиями к качеству. При этом под *программным компонентом* (далее компонентом**)** понимается независимо реализованный программный объект, который обеспечивает выполнение заданных функций, доступ к которым осуществляется с помощью интерфейсов, определяющих порядок обращения к операциям и функциональным возможностям этого компонента. Вопросы обеспечения качества интерфейса находятся в стадии исследования и будут рассмотрены в следующей публикации.

Требования, предъявляемые к конкретному компоненту, могут относится к классу компонентов, имеющих эквивалентные свойства и условия выполнения операций. Компонентная программа создается путем композиции компонентов или представителей эквивалентных классов компонентов, удовлетворяющих функциональным требованиям, требованиям к взаимодействию компонентов и их качеству. Концепция аналитической оценки характеристик качества компонентов является обобщением подходов к измерению качества ПО.

Определение модели качества компонентов

Для всех видов и типов ПО, компонентов и программных систем модель качества имеет такой общий вид:

$$\mathbf{M}_{кач} = \{\mathbf{Q, A, M, W}\} \qquad (1),$$

где $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_i\}_{i = 1,\ldots, 6}$ – множество характеристик качества (Quality – Q);

$\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_j\}_{j = 1,\ldots, J}$ – множество атрибутов (Attributes –A), каждый из которых фиксирует свойство $\mathbf{q}_i$ –характеристики качества;

$\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}_{k =1,\ldots,K}$ – множество метрик (Metrics –M) для каждого элемента a_j атрибута;

$\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}_{n = 1,\ldots, N}$ – множество весовых коэффицентов (Weights – W) для метрик M.

Элементы этой модели образуют четырехуровневую иерархическую модель. На первом уровне находятся характеристики, каждая из которых определяется набором атрибутов – свойств второго уровня,

детализирует разные аспекты этой характеристики в готовом продукте (например, для надежности – свойство безотказной работы, восстанавливаемости и др.).

На третьем уровне находятся метрики атрибутов каждой характеристики. Метрика - это комбинация метода измерения атрибута и шкалы измерения значений атрибутов [8, 9]. При оценке конкретного атрибута заданной характеристики на этапах жизненного цикла (при экспертизе документации, программ и результатов тестирования программ – число ошибок и отказов и др.) метрика уточняется с помощью весовой меры (весовых коэффициентов) для получения конечного количественного значения этого атрибута.

В зависимости от назначения, особенностей и условий разработки и сопровождения компонентов выбираются наиболее важные и приоритетные характеристики качества, которые отражаются в требованиях на разработку. Для программных систем, при разработки которых в требованиях не указан приоритет характеристик качества, используется приоритет эталона, т.е. класс компонентов или ПО, в котором требование имеет место [1– 8, 13–15].

По формальной модели (1) формируется конкретная модель для создаваемой компонентной системы, которая может включать необходимые характеристики (не менее одной) из множества Q, заданных заказчиком в требованиях к системе. Эти характеристики определяют соответствующие подмножества из представителей элементов множеств A, M, W. По конкретной модели проводится оценка конкретных атрибутов компонентов в соответствии со значениями метрик и их весовыми коэффициентами, отражающими данную систему. При измерения значений атрибутов, входящих в эту модель, используются экспертные и аналитические методы их оценки.

Метрики качества программных компонентов

Каждый отдельный компонент обладает свойствами, которые соответствуют заданным в требованиях к атрибутам и характеристик качества, для измерения которых используются метрики и методы их измерения. Метрика качества ПО [3, 8, 9] включает метод и шкалу измерения атрибутов для определения меры атрибутов, которые получают значения в процессе обработки компонента (например, список дефектов, число отказов и др.). Метод – это действие для получения текущего значения метрики (например, действия по подсчету ошибок, отказов и др.). Шкала измерения задается натуральными числами >0 и имеет согласно [3] следующие виды шкал:

- абсолютная, которая задает конкретное (абсолютное) значение некоторой составляющей метрики (например, 20 ошибок, 10 отказов и др.);
- относительная, которая задает конкретное значение при измерении некоторым инструментальным средством (например, время между отказами, размер компонента в символах и др.);
- интервальная (дата, время прогонки компонента и др.);
- порядковая задает статические свойства компонента (количество параметров, функций, название языка программирования компонента и др.).

Метрику целесообразно измерять на всех этапах эизненного цикла (т.е. получать промежуточные и внутренние значения) и на этапе тестирования – внешней для готовой системы. Существует три типа метрик: готового продукта, процесса его создания и сопровождения. Будем рассматривать метрики готового программного компонента, включающего внешние и внутренние метрики и отражающие соответственно свойства, ориентированные на пользователя и разработчика программной системы.

При определении требований к компонентам/системе задаются внешние характеристики качества Q_i, определяющие разные стороны функционирования и управления продуктом в заданной среде. Им соответствуют метрики и диапазон значений мер для измерения атрибутов этих характеристик.

Известными внешними метриками программных компонентов являются метрики Холстеда, которые выявляются экспертным путем по статической структуре программного компонента на языке программирования и включают: число вхождений часто встречающихся операндов и операторов; длину как сумму числа вхождений всех операндов и операторов и др. На их основе можно вычислить время программирования, уровень компонента (например, структурность, объектность и др.) и языка программирования.

К внутренним метрикам относятся:

- размер памяти компонентов;
- сложность (структурная, цикломатическая и др.);
- стиль и технологии создания отдельных компонентов и документов.

Эти метрики позволяют определить производительность, трудоемкость и размер системы, включая компоненты и данные. Конкретные метрики задаются на этапе формирования требований и анализируются при проверки планов достижения качества реализованных компонентов системы.

На современном этапе программирования, когда при разработке требований и проектировании структуры системы, используется UML–моделирование. Результат моделирования задается набором визуальных моделей, определяющих диаграммы требований, варианты использования и действующих лиц, а также диаграммы взаимодействия объектов, представлений компонентов и др.

Примерами внутренних метрик в классе объектно-ориентированного проектирования являются:

- общее число объектов и число повторно используемых;
- общее число операций объекта, повторно используемых и новых операций;
- число классов, наследующих специфические операции;
- число классов, от которых зависит данный класс;
- число пользователей класса или операций и др.

При оценки общего количества некоторых величин часто используются средне статистические метрики (например, среднее число операций в классе, среднее число наследников класса или операций класса и др.).

С помощью метрик готовых компонентов можно измерить степень удовлетворения потребностей пользователя для решения его задач в процессе эксплуатации системы. Примером может служить точность и полнота реализации задач пользователя, а также ресурсы, потраченные на эффективное решение этих задач. В [3, 8–10] определены следующие типы мер:

- размер компонентов и систем (числом операторов, числом компонентов и др.);
- время выполнения системы (по каждому компоненту отдельно, протоколам передачи данных и др.);
- усилия, направленные на создание системы (трудоемкость, производительность разработчиков системы и др.);
- учет ошибок, отказов в компоненте, попыток исправления обнаруженных ошибок и др.

В основном, меры являются субъективными и зависят от знаний экспертов, производящих оценку атрибутов компонентов ПО.

Специальной мерой может выступать уровень использования повторных компонентов и измеряемый отношением размера продукта из готовых компонентов, к размеру системы в целом. Эта мера может использоваться при определении стоимости продукта и его качества.

Оценка качества программной системы, состоящей из компонентов, согласно четырех уровневой модели качества (1) начинается с нижнего уровня, т.е. с атрибутов множества A для каждой характеристики качества в соответствии с установленными для них мерами. На этапе проектирования компонентов устанавливают значения весовых мер для каждого атрибута, включенного в требования к ним. Сначала измеряется значение атрибута для отдельного компонента, а затем для их совокупности.

Метрики качества используются при оценки степени тестируемости после проведения испытаний ПО на множестве тестов (безотказная работа, выполнимость функций, удобство применения интерфейсов пользователей, БД и т.п.) [13–15].

Очень часто оценка компонента проводится по числу строк. При сопоставлении двух компонентов, реализующих одну прикладную задачу предпочтение отдается более короткому, так как его создает более квалифицированный программист, в нем меньше скрытых ошибок и легче модифицировать. По стоимости он дороже, хотя времени на отладку и модификацию уходит больше. Т.е. длину программы можно использовать в качестве вспомогательного свойства при сравнении программных компонентов с учетом одинаковой квалификации разработчиков, единого стиля разработки и общей среды.

Подход к аналитической оценке показателей качества

В стандарте [3] и в ядре знаний SWEBOK [16], которое представляет сумму знаний компьютерных специалистов в области обеспечения качества, определено шесть базовых характеристик качества ПО:

q_1: функциональность (functionality),
q_2: надежность (realibility),
q_3: удобство применения (usability),
q_4: эффективность (efficiency),
q_5: сопровождаемость (maitainnability),
q_6: переносимость (portability).

Далее дадим толкование приведенным характеристикам качества применительно к компонентам, входящих в модель $M_{кач}$, и соответствующим атрибутам A этой модели, а также концепции аналитической их оценки .

Функциональность – совокупность свойств, определяющих способность компонентной системы предоставлять требуемое множество функций для решения задач в соответствии с требованиями. Она задается множеством атрибутов $q_1 = \{a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}\}$, семантика которых рассматривается ниже.

a_{11}: функциональная полнота – свойство компонента, которое показывает степень достаточности реализованных в нем функций для решения задач в соответствии с его назначением. Данный атрибут можно представить в виде отношения всех функций F^c, реализованных в компонетной системе (с), к функциям $F^т$ (т –требование), заданных в требованиях

$$a_{11} = \sum_{i=1}^{N} F^c \Big/ \sum_{j=1}^{K} F^т \qquad (2)$$

a_{12}: корректность – атрибут, который показывает на степень соответствия каждой функции F^T , заданной в требовании, и каждой функции F^c, реализованной в компонентной системе. При этом система обладает свойством полной корректности, если $F^T = F^c$ и частично корректной, если $F^T \subset F^c$. Для большинства систем достаточно частичной корректности. Степень корректности компонента можно определить как степень функциональной корректности по формуле:

$$\sqrt{} = 1 - (\text{card}\ (F^T / F^c)) / \text{card}\ F^T \qquad (3);$$

a_{13} : точность – свойство, определяющее получение системой согласованных результатов с необходимой степенью точности. Данный атрибут может оценивается отношением ∇ разности значения функции $F^c_i(D_i)$ компонента и значения функции $F^T_i(D_i)$, заданной требованиями на D_i входном наборе к значению функции в соответствии с выражением:

$$\nabla = \sum_{i=1}^{\text{card}\ F^T} \left(\left(F^c_i(D_i) - F^T_i(D_i) \right) / F^T_i(D_i) \right) / \text{card}\ F^T \qquad (4);$$

a_{14}: интероперабельность – свойство компонента системы взаимодействовать с другими компонентами и операционной средой;

a_{15} : защищенность – атрибут, который показывает возможность компонента (системы) фиксировать дефекты, которые являются следствием субъективных ошибок в его реализации или выхываемх программными или аппаратными средствами при выполнении, а также ошибок, связанных с данными. Оценку степени защищенности можно провести с помощью выражения: $a_{15} = fal^z / fal$, где

fal^z – количество дефектов, от которых компонент защищен,

fal – общее количество дефектов в компоненте или системе;

a_{16}: согласованность – атрибут, который показывает степень соблюдения стандартов, правил и др. соглашений процесса разработки и оценивается экспертно, а затем полученные качественные оценки нивелируется с помощью соответствующих весовых коэффициентов.

Таким образом, характеристика качества q_1 (функциональность) может быть вычислена путем суммирования атрибутов с учетом метрик и их весовых коэффициентов:

$$\mathbf{q_1} = \sum_{j=1}^{6} \mathbf{a_{1j}\, m_{1j}\, w_{1j}} \qquad (5).$$

Надежность ПО будем определять как вероятность того, что компоненты системы или сама система функционирует безотказно в течение фиксированного периода времени в заданных условиях операционного окружения/среды. Надежность определяется на множестве атрибутов $\mathbf{q_2 = \{a_{21}, a_{22}, a_{23}, a_{24}\}}$,, которые определяют способность системы преобразовывать исходные данные в результаты при условиях, зависящих от периода времени жизни системы (износ и его старение не учитывается). Снижение надежности компонентов происходит из-за ошибок в требованиях, проектировании и выполнении. Отказы и ошибки в программных компонентах могут появляться на заданном промежутке времени функционирования компонента/системы [11–20]. Рассмотрим каждый атрибут надежности.

a_{21}: безотказность – свойство системы, которое определяет функционирование системы без отказов (программных компонентов или оборудования). Если компонент содержит дефект, вызванный субъективными ошибками при разработке, то во множестве $D=\{De|e\in L\}$ всех дефектов, можно выделить подмножество $E \subseteq D$, для которых результаты не соответствуют функции F^T, заданной в требованиях на разработку. Вероятность p беотказного выполнения компонента на De, случайно выбранном из D среди равновероятных равно:

$$p = 1 - \text{card}\ \{E\} / \text{card}\ \{D\} \qquad (6).$$

Под отказом (failure) понимается отклонение поведения системы от предписанного, которое заключается в том, что система перестаёт выполнять предписанные ей функции. Другое определение отказа (fault), как причины ошибки, которая ее вызывает. И наконец, ошибка (error) соответствует состоянию системы, которое вызывает отказ. Если ошибка сделана человеком, то используется термин mistake. Когда различие между fault и failure не является критическим, используется термин – defect, который означает либо fault (причина), либо failure (действие). Связь между этими понятиями можно представить так: fault → error → failure.

Существует большое разнообразие видов отказов ПО, типичные из них: внезапные, постепенные, перемещающиеся (сбои). Причины отказов могут быть физические, структурные, отказы взаимодействия и др. Они могут возникать естественным путём, вносится человеком или внешней операционной средой в период создания или эксплуатации системы , а также быть постоянными или носить временный характер.

Наработка на отказ, как атрибут надежности определяет среднее время между появлением угроз, нарушающих безопасность, и обеспечивает трудно измеримую оценку ущерба, которая наносится соответствующими угрозами.

Для вычисления среднего времени T наработки на отказ применяется формула:

$$T = \sum_{I=1}^{De} \nabla t_i^E \ / N \qquad (7),$$

где ∇t_i^E – интервал времени безотказной работы компонента i – отказа;
N – количество отказов в системе.

a_{22}: устойчивость к ошибкам – свойство компонентов системы, которое показывают на способность программной системы выполнять функции при аномальных условиях (сбоях аппаратуры, ошибках в данных и интерфейсах, нарушениях в действиях оператора и др.). Оценку устойчивости можно получить по формуле:

$$\Upsilon = N^v/N, \qquad (8),$$

где N^v – количество разных типов отказов, для которых предусмотрены средства восстановления;
N – общее количество всех отказов в системе.

a_{23}: восстанавливаемость – свойство системы, которое показывают на способность возобновлять функционирование системы после отказов и восстанавливать в ней поврежденные компоненты и/или данные для повторного исполнения. Среднее время восстановления компонента можно определить по формуле:

$$T = \sum_{I=1}^{De} \nabla t_i^b \ / De \qquad (9),$$

где ∇t_i^b – время восстановления работоспособности компонента после i – отказа;
De – количество дефектов и отказов в системе.

a_{24}: согласованность – атрибут, который показывает степень соблюдения стандартов, технологии, правил и др. соглашений на стадиях разработки и тестирования системы для поиска разного рода ошибок разработки. Этот атрибут оценивается экспертами, которые проверяют соблюдение технологии и стандартов, фиксируют ошибки в специальных картах и дают экспертную оценку надежности системы.

Некоторые типы систем реального времени, безопасности и др. требуют высокой надежности (недопустимость ошибок, точность, достоверность и др.), которая в значительной степени зависит от числа оставшихся и не устраненных ошибок в процессе ее разработки на этапах жизненного цикла. В ходе эксплуатации ошибки могут также обнаруживаться и устраняться. Если при исправлении ошибок не вносятся новые, вносится их меньше, чем устраняется, то в ходе эксплуатации надежность системы непрерывно растает. Чем интенсивнее проводится эксплуатация, тем интенсивнее выявляются ошибки и быстрее растет надежность. К факторам, влияющим на надежность ПО, относятся также:

- риск – совокупность угроз, приводящих к неблагоприятным последствиям и ущербу системы или среды;
- угроза – явление, которое приводит к нарушению безопасности системы;
- целостность – способность совокупности компонентов системы сохранять устойчивость работы и приводить к рискам.

Риск уменьшает свойства надежности, особенно если обнаруженные ошибки могут быть результатом проявления угрозы извне.

Так как надежность является одной из ключевых проблем современных программных систем и ее роль будет постоянно возрастать, так как постоянно повышаются требования к качеству компьютерных систем. Новое направление – инженерия программной надёжности (Software reliability engineering –SRE) ориентировано на количественное изучение операционного поведения компонентов системы, по отношению к пользователю, ожидающему надежную работу системы [13], включает:

1. Измерение надёжности, т.е. проведение ее количественной оценки с помощью предсказаний, сбора данных о поведении системы в процессе эксплуатации и современных моделей надёжности;

2. Стратегии и метрики конструирования и выбора готовых компонентов, процесс разработки компонентной системы, а также среда функционирования, влияющая на надёжность работы системы.

3. Применение современных методов инспектирования, верификации , валидации и тестировании при разработке систем., а также при эксплуатации.

Верификация – это процесс определения соответствия готового ПО установленным спецификациям, а валидация – процесс установления соответствия системы требованиям пользователя, которые были предъявлены заказчиком.

С теоретической точки зрения в инженерии надежности появился новый термин dependability [10] сильнее термина reliability [3]), который обозначает способность системы обладать свойствами, желательными для пользователя этой системы и который должен иметь уверенность в качественном выполнении функций, заданных в требованиях к системе. Dependability (пригодноспособность, т.е. надежность в широком смысле) характеризуется дополнительным числом атрибутов, которыми должна обладать система, а именно :

- готовность к использованию (availability),
- готовностью к непрерывному функционированию (reliability),

- безопасность для окружающей среды, т.е. способность системы не вызывать катастрофических последствий в случае отказа (safety);
- секретность и сохранность информации (confidential),
- способность к сохранению системы и устойчивости к самопроизвольному ее изменению (integrity),
- способность к эксплуатации ПО, простоту выполнения операций обслуживания, а также устранения ошибок, восстановление системы после их устранения и т.п.(maintainability),
- готовность и сохранность информации (security) и др.

Достижение требуемой надежности системы обеспечивается путем предотвращения отказа (fault prevention), устранения отказа (removal fault), возможного выполнения системы при наличии отказа и оценки возможности появления новых отказов и мер борьбы с ним. Это объясняется случайным характером таких явлений, анализ, которые основываются на методах теории вероятностей и случайных процессов.

Каждый программный компонент, его команды и данные обрабатываются в дискретные моменты времени б, 2б,…nб и это может быть произведено «удачно» или нет. Пусть за время Т после первого неудачно обработанного компонента системы появился отказ и пусть q_b – вероятность этой неудачи, тогда $P\{ T > n\delta \} = (1 - q_\delta)^n$, а среднее время ожидания $T = \delta / q_\delta$.

Положим, что δ убывает так, что время Т остаётся фиксированным, тогда имеем

$$P\{ T > t \} = (1 - \delta / T)^{t/\delta} \approx e^{-t/T} \qquad (10)$$

Т.е. время до отказа программной системы данном случае является непрерывной величиной с…l распределённой экспоненциально с параметром Т…1/Т.

Практически оценка надёжности ПО является трудоемким процессом, в нем важное место занимает метод создания устойчивости системы к отказам ПО, т.е. вероятности того, что система восстановится самопроизвольно в некоторой точке после возникновения в ней отказа (fault).

Количественная оценка характеристики надежности системы по всем ее рассмотренным количественным атрибутам и соответствующим метрикам имеет вид:

$$q_2 = \sum_{j=1}^{4} a_{2j}\, m_{2j\ w\,2j} \qquad (11).$$

Удобство применения - это множество свойств программной системы, которые показывают на необходимые и пригодные условия использования (диалоговое или не диалоговое) определенным кругом пользователей для получение результатов выполнения. Эта характеристика определяется на множестве атрибутов, которые удовлетворяют эргономичности, и включают:

a_{31}: понимаемость – атрибут, который определяет усилия, затрачиваемые на распознавание логических концепций и условий применения программной системы;

a_{32}: изучаемость – атрибут, который определяет усилия пользователей при определении применимости ПО, посредством операционного контроля, ввода, вывода, диагностики, а также процедур +анализа документации;

a_{33}: оперативность – атрибут, который показывает на реакцию системы при выполнении операций и операционного контроля;

a_{34}: согласованность – атрибут, который показывает соответствие программной системы требованиям стандартов, соглашений, правил, законов и предписаний.

Все атрибуты оцениваются экспертами, которые в зависимости от их уровня знаний дают соответствующие качественные значения. Количественная оценка данной характеристики – удобство применения системы зависит от оценок экспертов и количественного атрибута a_{33} и имеет вид:

$$q_3 = \sum_{j=1}^{4} a_{3j}\, m_{3j\ w\,3j} \qquad (12).$$

Эффективность - множество атрибутов – свойств системы, которые показывают взаимосвязь между уровнем ее выполнения, количеством используемых ресурсов (аппаратуры, материалов - бумага для печатающего устройства и др.) и услуг штатного обслуживающего персонала и др.

К атрибутам эффективности относятся:

a_{41}: реактивность – атрибут, который показывает время отклика, обработки и выполнения функций компонента/системы;

a_{42}: эффективность – атрибут, показывающий количество используемых ресурсов при выполнении функций ПО и их и продолжительность вычислений;

a_{43}: согласованность – атрибут, который показывает соответствие данного атрибута с заданными стандартами, правилами и предписаниями.

Количественная оценка характеристики – эффективность системы по всем рассмотренным качественным и количественно измеряемым атрибутам, экспертно и аналитически с учетом соответствующих метрик имеет вид:

$$q_4 = \sum_{j=1}^{3} a_{4j}\, m_{4j\, W\, 4j} \qquad (13).$$

Сопровождаемость – множество свойств, которые показывают на усилия, которые надо затратить на проведение модификаций, включающих корректировку, усовершенствование и адаптацию системы при изменении среды, требований или функциональных спецификаций. Она включает атрибуты:

a_{52}: анализируемость – атрибут, определяющий необходимые усилия для диагностики отказов в ПО или идентификации частей, которые будут модифицироваться;

a_{53}: изменяемость – атрибут, определяющий усилия, которые затрачиваются на модификацию компонента, удаление ошибок или внесение изменений, дополнение новых возможностей в систему или в среду функционирования;

a_{54}: стабильность – атрибут, указывающий на риск проведения модификации компонента /системы;

a_{53}: тестируемость – атрибут, показывающий на усилия при проведении верификации с целью обнаружения ошибок и несоответствий требованиям (валидация), а также на необходимость исправления обнаруженных ошибок и проведения сертификации системы;

a_{54}: согласованность – атрибут, который показывает соответствие данного атрибута с определенными в стандартах, соглашениях, правилами и предписаниях.

Количественная оценка характеристики – сопровождаемость готовой системы по всем ее рассмотренным качественным и количественно измеряемым атрибутам, экспертно и аналитически с учетом соответствующих метрик имеет вид:

$$q_5 = \sum_{j=1}^{3} a_{5j}\, m_{5j\, W\, 5j} \qquad (14).$$

Переносимость – множество атрибутов, указывающих на способность компонентов системы приспосабливаться к работе в новых условиях среды выполнения: организационной, аппаратной и программной. Перенос компонентов или всей системы связан с совокупностью действий, направленных на обеспечение возможности его функционирования в новой среде, отличной от той среды, в которой эта система создавалось. К атрибутам относятся:

a_{61}: адаптивность – атрибут, определяющий усилия, затрачиваемые на адаптацию системы к различным операционным средам. Этот атрибут можно представить в виде

$a_{61} = Za / Zd$, где

Za – затраты на адаптацию в новой операционной среде;

Zd – затраты на разработкоу новой системы для новой операционной среды.

a_{62}: настраиваемость – атрибут, который определяет необходимые усилия для запуска или инсталяции данного программного продукта в другой среде;

a_{63}: сосуществование – атрибут, который определяет возможность использования специального ПО в среде действующей системы;

a_{64}: заменяемость – атрибут, который обозначает возможность взаимодействия (интероперабельности) с другими программами при совместной их работе и инсталляции или адаптации системы;

a_{65}: согласованность – атрибут, который показывают на соответствие стандартам или соглашениям и правилам переноса программной системы в другую среду.

Количественная оценка характеристики – переносимость компонента/системы по рассмотренным качественным и количественно атрибутам, измеряемым экспертным и аналитическим путем с учетом соответствующих метрик имеет вид:

$$q_6 = \sum_{j=1}^{5} a_{6j}\, m_{6j\, W\, 6j} \qquad (15).$$

Комплексная оценка. В формулах (6, 11-15) приведены оценки отдельных показателей качества отдельного компонента с использованием метрик $a_i \subset A$ и весомых коэффициентов $wi \subset W$ для каждого атрибута. Полученные значения q_j с помощью весовых коэффициентов $wi \subset W$ приведены к единой системе измерения.

Используя полученные оценки q_j характеристик качества, представленные формулами (6, 11-15), применительно к отдельному компоненту (com), получаем интегральную оценку качества одного компонента в виде:

$$Q_{com} = \sum_{j=1}^{6} q_j \qquad (16).$$

Если программная система содержит N–компонентов и для них проведена количественная оценка аналогично (16), то получаем окончательную комплексную оценку качества системы (sys):

$$Q_{sys} = \sum_{l=1}^{N} Q^{l}_{com} \qquad (17).$$

Таким образом, данный подход к аналитической оценке атрибутов показателей качества программных компонентных систем показывает степень качества готовой программной системы

Комплексное значение качества компонентной системы является критерием эффективности и целесообразности применения метода компонентного проектирования систем и методик постепенного накопления значений оцениваемых результатов на стадиях жизненного цикла.

Заключение

В работе представлены результаты исследований по проблематике качества компонентных систем. Предложена подход к аналитической оценке отдельных атрибутов показателей качества, классификация которых представлена в стандартах качества на ПО и которыми должны обладать современные программные системы. Отмечается, что на оценку качеств программных систем влияют методы инженерии требований к ПО и методы, гарантирующие достижение заданных характеристик на этапах жизненного цикла. Дана комплексная оценка качества компонента и системы.

Литература

1. NASA–STD–2201/ Software Assurance Standart, 1993.
2. ANCI/IEEE 730–1. IEEE Standard for Software Guality Analisys Plans. –1989.
3. ISO/IEC 9126–2. Infofmation Technology. - Software Quality Characteristics and metrics.– 1997.
4. ДСТУ 2844–1994. Программные средства ЭВМ. Обеспечение качества. Термины и определения..
5. ДСТУ 2850–1994. Программные средства ЭВМ. Обеспечение качества. Показатели и методы оценки качества программного обеспечения.
6. ДСТУ 3230–1995. Управление качества и обеспечение качества. Термины и определения.
7. ISO 14598. Information Technology – Software product evolution- Part1: General overview.–1996.
8. *Meyer B.* The role of Object–Oriented Metrics.– Computer, 1998. –№11.–p.23–125.
9. IEEE Software. Measurement.– March/April, 1997.
10. *Haag S., Raja H.K., Sekade L.L.* Quality Function Deployment. Usage in Software Development// Comm. of ACM.–1998. –9. –N1.
11. *Кулаков А.Ю.* Оценка качества программ ЭВМ.–Киев: Техніка.–1984.–167с.
12. *Мороз Г.Б., Лаврищева Е.М.* Модели роста надежности программного обеспечения.- Киев.-Препринт 92–38, 1992.– 23с.
13. *Андон Ф.И., Коваль Г.И., Коротун Т.М., Суслов В.Ю.* Основы инженерии качества программных систем.–К: Академпериодика, 2002.–502с.
14. *Липаев В.В.* Выбор и оценивание характеристик качества программных систем. Методы и стандарты.– М.: СИНТЕГ, 2001.–224с.
15. *Липаев В.В.* Методы обеспечения качества крупномасштабных программных систем.–М.: СИНТЕГ, 2003.–510с.
16. http:/www.swebok.org.html
17. *Липаев В.В.* Надежность программных систем .–.М.: СИНТЕГ, 1998.–321с.
18. *Майерс Г.* Надежность программного обеспечения .– М.: Мир, 1980.–360с.
19. *Гласс Г.* Руководство по надежному программированию.–М.: Финансы и Статистика, 1982.–256с.
20. *Тейер Т., Липов Р.* Нельсон Э. Надежность программного обеспечения.–М.: Мир, 1981.– 325с.