УДК 004.05 DOI: 10.18698/2541-8009-2023-9-931

АТРИБУТЫ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.С. Власенко vlasenkoas@student.bmstu.ru
Д.В. Серёгина seryoginadv@student.bmstu.ru
А.А. Козаченко kozachenkoAA@student.bmstu.ru
А.П. Звездин zap17u813@student.bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Наряду с требованиями к функционалу разрабатываемого программного обеспечения важную роль играет качество конечного продукта, которое позволяет судить о преимуществах данного продукта перед аналогами. В статье сформулированы ключевые понятия в области оценки качества программного обеспечения, рассмотрены методы работы с нефункциональными требованиями на разных этапах разработки. Подробно представлены основные модели качества и существующие подходы к классификации и формализации атрибутов качества программного обеспечения. Рассмотрены роли и обязанности членов рабочей группы при работе с качественными требованиями. Для демонстрации практического применения проанализированного материала разработана обобщенная модель качества клиентской части веб-приложения.

Ключевые слова

Атрибуты качества, оценка качества, модель качества, нефункциональные требования, разработка программного обеспечения, проектирование программного обеспечения, формализация требований, архитектура программного обеспечения

Поступила в редакцию 18.07.2023

Введение. Одним из ключевых разделов проектирования программного обеспечения (ПО) является анализ и оценка качества проектирования ПО (*Software Design Quality Analysis and Evaluation*) [1]. Архитектурно-экономический цикл разработки ПО показывает, что качественные требования к ПО определяются коммерческими целями и мотивами. Исходя из подобных требований можно выделить ряд критериев, которые будут сопровождать функциональные требования к системе. Такие критерии называют атрибутами качества ПО.

Цель данной работы — подробное рассмотрение атрибутов качества с точки зрения их разработки, поддержки и технических характеристик, а также их влияния на процесс проектирования ПО.

В рамках данной работы использованы следующие термины:

– *программное обеспечение (ПО)* — программа или множество программ, используемых для управления компьютером;

– *проектирование ПО* — процесс определения архитектуры, компонентов, интерфейсов, других характеристик системы и конечного состава программного продукта;

– *архитектура ПО* (*software architecture*) — совокупность важнейших решений об организации программной системы.

Нефункциональные требования. Нефункциональные требования служат неотъемлемой частью повседневной жизни людей. Например, покупатель может отказаться от покупки кресла из-за того, что оно неудобное или сделано из некачественного материала. Несмотря на то что кресло обеспечивает необходимый функционал (на нем можно сидеть), оно может не понравиться пользователям. Аналогичным образом, успех программы определяется не только предоставляемыми ею функциональными возможностями, но также и некоторыми абстрактными ожиданиями (в том числе ожиданиями пользователей), насколько хорошо должен работать продукт. В целом эти нефункциональные требования называют *факторами качества* [2], а атрибуты качества наряду с ограничениями и требованиями к интерфейсу составляют основной класс нефункциональных требований. Существует множество проблем с формализацией как нефункциональных требований, так и атрибутов качества в частности. Например, не всегда может быть понятно, является ли та или иная потребность функциональной или нефункциональной. В связи с этим не существует строгого общепринятого определения атрибутов качества, а разные источники выдвигают разные их наборы и классификации. На практике обычно подразумевают, что атрибуты качества представляют собой набор мер, формализующих отличия между продуктами с одинаковым функционалом.

Взаимодействие атрибутов качества. Как уже было сказано, атрибут качества представляет собой некоторое нефункциональное требование к системе, которое является источником некоторого множества функциональных требований. А это значит, что реализация атрибута затрагивает все этапы разработки ПО (проектирование, реализацию, развертывание и тестирование). При этом для сложных систем реализация набора атрибутов может стать невозможной задачей, поскольку каждый атрибут влияет на всю систему целиком и может оказаться несочетаемым с другим атрибутом качества. Предположим, одним из требований к программному продукту является возможность его использования в разных средах — данный атрибут называется *переносимостью*. Существует множество стратегий реализации переносимости, но необходимое условие — создание абстрактного интерфейса между логикой приложения и системными функциями, т. е. изолирование системных зависимостей. Тогда в ходе исполнения системы будут возникать непроизводительные издержки, а значит, снизится производительность, которая также служит одним из атрибутов качества. Поэтому хорошо реализованные программные продукты являются результатом оптимального баланса противоречивых характеристик качества, что подробно описано в [3].

Выбор набора атрибутов качества. Нефункциональные требования определяют в процессе проектирования ПО. Для этого выделяют рабочие группы,

задачи членов которых состоят в определении, проверке и утверждении таких требований. Для постановки нефункциональных требований необходимо сотрудничество аналитиков, архитекторов, ключевых разработчиков продукта, а также тестировщиков и пользователей. Состав такой команды может показаться избыточным, но на практике каждое ее звено необходимо. Задачи членов команды указаны в табл. 1.

Таблица 1. **Задачи членов рабочей группы при определении нефункциональных требований**

Роль	Задача
Тестировщик	Оценивает параметры, которые используются для постановки качественных требований. Параметры, как правило, оказываются привязанными к пользовательским сценариям, в которых должны выполняться определенные действия с определенными ограничениями
Системный аналитик	Осуществляет сбор, анализ, документирует и систематизирует качественные требования
Системный архитектор	Участвует в определении и анализе нефункциональных требований и проверяет их на реализуемость
QA-инженер	Разрабатывает сценарии тестирования для проверки нефункциональных требований
Технический руководитель	Участвует в определении и формализации нефункциональных требований и их реализации как на уровне глобальных решений, так и на уровне отдельных проектов

Формализация атрибутов качества. Несмотря на абстрактность определения нефункциональных требований, существуют подходы, позволяющие структурировать атрибуты качества, что необходимо для корректного процесса проектирования программного обеспечения. Можно выделить ряд критериев качественных требований, при соответствии которым работа с требованиями становится проще. Такие требования рассматриваются в [2]. Ниже приведены некоторые критерии качественных требований:

– *полнота* (отдельного требования и системы требований) — требование должно содержать всю необходимую информацию для его реализации;

– *однозначность* — требование должно быть внутренне непротиворечиво и все сотрудники, работающие с ним, должны понимать его одинаково;

– *корректность* — требование не должно содержать в себе неверной, неточной информации, а отдельные требования в системе требований не должны противоречить друг другу;

– *необходимость* — требование должно отражать возможность или характеристику ПО, действительно необходимую пользователям или вытекающую из других требований;

– *осуществимость* — включаемое в спецификацию требование должно быть выполнимым при заданных ограничениях среды;

– *проверяемость* — существует конечный и разумный по стоимости процесс ручной или машинной проверки того, что ПО удовлетворяет этому требованию.

Качество нефункциональных требований непосредственно определяет качество разрабатываемого продукта и достигается с помощью итеративного процесса определения и анализа нефункциональных требований. Определение необходимых атрибутов качества базируется на выбранной для конкретного программного продукта модели качества. В индустрии ПО есть несколько общепринятых моделей качества. В основном эти модели были разработаны в период с 1975 по 1995 г. и с тех пор глобально не изменились, несмотря на их постоянные усовершенствования. Далее рассмотрим некоторые из стандартов, устанавливающих модели качества ПО.

Стандарт ISO 9126 (ГОСТ ИСО/МЭК 9126–93). Данный стандарт был выпущен 19 декабря 1991 г., а в 2001 г. ISO/IEC 9126:1991 был преобразован и расширен в систему из четырех взаимосвязанных стандартов [3]. Модель качества, установленная в первой части стандарта ISO 9126–2001, классифицирует качество программного обеспечения согласно шести наборам характеристик, которые, в свою очередь, детализированы подкатегориями.

Функциональность — «набор атрибутов, которые влияют на существование набора функций и их заданных свойств. Функции — это характеристики ПО, которые удовлетворяют заявленные или подразумеваемые потребности». Каждую подкатегорию качества далее подразделяют на атрибуты качества. Атрибуты не определены в стандарте, поскольку они могут различаться для разных ПО. Таким образом, в процессе проектирования остается возможность точно определить свою собственную модель. Схематично структура стандарта показана на рис. 1. Подробно данная модель качества рассмотрена в [3].

ГОСТы серии 34. В ГОСТах серии 34 «Информационные технологии. Комплекс стандартов на автоматизированные системы» установлены термины и определения основных понятий в области автоматизированных систем в различных областях (включая проектирование), а также состав, содержание и правила оформления технического задания на создание системы. Документация данного стандарта содержится в [3]. В ГОСТах серии 34 используется собственная терминология, однако в целом выделяются следующие категории атрибутов качества:

1) степень приспособляемости системы к изменению процессов и методов управления, к отклонениям параметров объекта управления;

2) допустимые пределы модернизации и развития системы;

3) вероятностно-временные характеристики, при которых сохраняется целевое назначение системы;

4) состав и количественные значения показателей надежности для системы в целом или ее подсистем;

5) требования к надежности технических средств и программного обеспечения;

6) требования к информационной совместимости со смежными системами;

7) требования к защите данных от разрушений при авариях и сбоях в электропитании системы.

Рис. 1. Модель качества ПО согласно ISO 9126

Модель качества по МакКоллу. Первой широко известной и востребованной моделью качества ПО стала модель, которую предложил МакКол в 1977 г. Подробное описание модели можно найти в [1]. В ней характеристики качества были разбиты на три группы. Факторы (factors) описывают ПО с позиций пользователя и задаются как требования (внешние качества). Критерии (criteria) описывают ПО с позиций разработчика и задаются как цели (внутренние качества). Метрики (metrics) используются для количественного описания и измерения качества.

Факторы качества, а их было выделено 11, подразделены на три группы по различным способам взаимодействия разработчиков и пользователей с ПО. Полученная структура изображается в виде треугольника МакКола (рис. 2).

Рис. 2. Треугольник МакКола

Критерии качества — это числовые значения факторов, которые были поставлены в качестве целей при разработке. МакКол ввел метрики качества, которые с его точки зрения оптимально измерять и оценивать. Оценки в его шкале принимают значения от 0 до 10. Каждая метрика влияет на оценку нескольких факторов качества. Числовое выражение фактора представляет собой линейную комбинацию значений влияющих на него метрик. Коэффициенты этого выражения определяются по-разному для разных организаций, команд разработки, видов ПО, используемых процессов и т. п.

Классификация атрибутов качества. Для классификации атрибутов качества применяют множество различных схем, которые чаще всего описаны одним из стандартов моделей качества ПО или удобны в сочетании с ним. Среди классификаций чаще всего выделяют: DeGrace и Stahl (1993) [4], IEEE (1998) [5], ISO/IEC (2007), Miller (2009) [1], ISO/IEC (2011).

Некоторые авторы разработали обширные иерархии, которые группируют связанные атрибуты на несколько основных категорий и вводят дополнительные аспекты и инструменты для формализации оценок. Так, в [6] рассмотрен вопрос использования различных типов шкал измерений для оценки значений метрики, а в [7] предложено использовать формулу Байеса для формирования апостериорного распределения вероятностей на множестве гипотез о том, что качество ПО соответствует одному из уровней ранжирования. Многие из классификаций атрибутов качества ПО рассмотрены в [1]. Однако все подходы во многом схожи друг с другом. Например, в сочетании с моделью качества по МакКолу используют классификацию, также основанную на разделении характеристик, которые проявляются в период выполнения (*внешнее качество*), и тех, что не проявляются в период выполнения (*внутреннее качество*). В соответствии с данной классификацией внешние характеристики главным образом важны для пользователей, а внутренние имеют значение для разработчиков и службы технической поддержки. Внутреннее качество косвенно влияет

на мнение клиента благодаря упрощению возможных улучшений продукта, его корректировки, тестирования и миграции на другие платформы. Примечательно, что оценка внутренних качеств часто хорошо поддается автоматизации. Например, в работе [8] предложен подход статистического анализа внутренних качеств с помощью кластеризации и дерева решений. Внешние и внутренние качества ПО показаны в табл. 2 и 3 соответственно.

Таблица 2. **Внешние качества**

Внешнее качество	Описание
Доступность	При каких условиях сервисы системы могут быть применены
Удобство установки	Насколько просто установить, удалить или обновить приложение
Целостность	Насколько хорошо система защищена от потери данных
Совместимость	Насколько просто система может взаимодействовать с другими системами и компонентами
Производительность	Как быстро система реагирует на события
Надежность	Как долго система работает без сбоев
Устойчивость	Насколько качественно система реагирует на нестандартные условия работы
Безопасность	Насколько качественно система защищена от неправомерного доступа к данным

Таблица 3. **Внутренние качества**

Внутреннее качество	Описание
Эффективность	Насколько эффективно используются ресурсы системы
Возможность модификации	Насколько удобно обслуживать и модифицировать систему
Переносимость	Насколько легко запустить систему в другой среде
Возможность повторного использования	Насколько компоненты системы могут использоваться в других системах
Масштабируемость	Как хорошо система справляется с увеличением числа пользователей (данных, серверов и т. д.)
Тестируемость	Как быстро тестировщики могут подтвердить корректность работы системы

Разработка модели качества. Как уже было сказано, атрибуты качества служат неотъемлемой частью любого программного обеспечения на всех этапах его разработки. В качестве практического примера использования изложенного материала в данном разделе разработана возможная обобщенная модель качества клиентской части некоторого веб-приложения.

В качестве стандарта модели качества был выбран ISO/IEC 9126–2011, поскольку данный стандарт является наиболее гибким при выделении специфических для данного продукта атрибутов качества. Для построения модели прежде всего необходимо определить необходимый и достаточный набор характеристик. К основным характеристикам качества клиентской части веб-приложений могут быть отнесены функциональность, переносимость, удобство использования, надежность и ремонтопригодность. Из данного набора исключена характеристика эффективности, так как ее анализ скорее относится к серверной части приложения. Функциональность служит одной из основных характеристик, поскольку на основе тестирования функциональной пригодности определяется готовность продукта к релизу. Совместимость также является важной для клиентской части и заключается в кроссбраузерности и кроссплатформенности разработанного продукта. Удобство использования — одна из основных характеристик качества продукта с точки зрения пользователей, эта характеристика не может быть исключена. Надежность клиентской части важна в контексте ее подхарактеристики «завершенность», оценка которой основана на подсчете количества ошибок в коде клиентской части веб-приложения. Ремонтопригодность (сопровождаемость) также является важным свойством клиента, однако атрибуты качества этой характеристики зависят от архитектурных и технологических решений и потому не могут быть полноценно рассмотрены в обобщенном примере.

Для построения интегральной оценки на основе отдельных метрик можно использовать метод, приведенный в ГОСТ 28195–99. Для каждой *j*-й метрики качества m_j определяют весовой коэффициент $V_j^{\text{М}}$, при котором сумма весовых коэффициентов всех метрик, относящихся к одной и той же подхарактеристике качества, постоянна и равна единице: $\sum_{1}^{J} V_j^{\text{М}} = 1$, где J — количество метрик подхарактеристики, М — признак метрики. Далее оценивают каждую *i*-й подхарактеристику качества $S_i^{\text{П}} = \dfrac{\sum_{j=1}^{J_i} m_j V_j^{\text{М}}}{\sum_{j=1}^{J_i} V_j^{\text{М}}}$, где J_i — количество метрик, используемых при оценке *i*-й подхарактеристики, а П — признак подхарактеристики. Для каждой *i*-й подхарактеристики качества $S_i^{\text{П}}$ определяется весовой коэффициент $V_i^{\text{П}}$, при этом $\sum_{i=1}^{I} V_i^{\text{П}} = 1$, где I — количество атрибутов качества некоторой характеристики. Далее рассчитывают оценку каждой *k*-й характеристики качества $C_k^X = \dfrac{\sum_{i=1}^{I_k} S_i^{\text{П}} V_i^{\text{П}}}{\sum_{i=1}^{I_k} V_i^{\text{П}}}$, где I_k — количество атрибутов качества, исполь-

зуемых при оценке *k*-й характеристики, X — признак характеристики. Для каждой *k*-й характеристики качества C_k^X также определяют весовой коэффициент V_k^X, при этом $\sum_{k=1}^{K} V_k^X = 1$, где K — количество характеристик в разработанной модели качества. Интегральную оценку качества Q рассчитывают по формуле $Q = \sum_{k=1}^{K} (C_k^X V_k^X)$. таком случае Q будет принимать значения на интервале $\left[0; 1\right]$, причем чем больше значение Q, тем выше качество программного продукта.

Набор атрибутов качества и соответствующих метрик, который может быть выделен в рамках разработки клиентской части веб-приложения, показан в табл. 4.

Таблица 4. **Выделенные атрибуты качества**

Атрибут качества	**Метрика**	**Вычисление метрики**
Функциональность		
Функциональное соответствие	Соответствие целям пользователя	$f = \frac{n}{N}$, где n — количество достигнутых целей; N — общее количество целей пользователей
	Соответствие задачам пользователя	$f = \frac{n}{N}$, где n — количество пользователей приложения, успешно выполнивших задачи; N — общее количество пользователей
Совместимость		
Поддержка	Взаимодействие с браузерами	$f = \frac{n}{N}$, где n — количество поддерживаемых браузеров; N — общее количество различных браузеров, заданное в спецификации
	Взаимодействие с операционными системами	$f = \frac{n}{N}$, где n — количество поддерживаемых операционных систем; N — общее количество операционных систем, заданное в спецификации
Адаптивность	Поддержка разрешений экрана	$f = \frac{n}{N}$, где n — количество поддерживаемых разрешений, а N — общее количество разрешений экрана, заданное в спецификации

Окончание табл. 4

Атрибут качества	Метрика	Вычисление метрики
Удобство использования		
Сходство с аналогами	Время выполнения задачи при первом знакомстве с веб-приложением	$f = \frac{1}{n}\sum_{i=1}^{n} H_i$, где $H_i = 1 - \frac{T_i}{T_i^{\max}}$, если $T_i < T_i^{\max}$, 0 — в противном случае; T_i — среднее время выполнения *i*-й задачи при первом знакомстве с приложением пользователями; $T_i^{\max}$ — максимально допустимое среднее время выполнения *i*-й задачи при первом знакомстве с приложением, заданное в спецификациии
Реактивность	Необходимость действий для обновления какой-либо информации	$f = \frac{n}{N}$, где n — количество сценариев, требующих обновления; N — общее количество сценариев, заданное в спецификации
Простота использования	Потребность в документации	$f = \frac{n}{N}$, где n — количество задач (функций), которые могут быть выполнены без обращения к справке; N — общее количество задач (функций), заданное в спецификации
	Понятность назначения экранов	$f = \frac{n}{N}$, где n — количество экранов, назначение которых понятно пользователю при первом знакомстве с приложением; N — общее количество экранов приложения

Разработанную модель качества можно использовать для построения совокупной оценки, отражающей соответствие клиентской части веб-приложения базовым нефункциональным требованиям. Данную модель можно модифицировать для более детального анализа с учетом особенностей предметной области, архитектурных и технологических решений, а также назначения разрабатываемого приложения.

Заключение. Рассмотрены основные подходы к формализации, постановке и классификации атрибутов качества программного обеспечения, а также разработана обобщенная модель качества клиентской части веб-приложения. Понимание рассмотренного материала необходимо для эффективной реализации программного обеспечения независимо от используемых подходов на всех этапах разработки программного обеспечения.

Литература

[1] Galin D. *Software quality assurance from theory to implementation*. Harlow, Pearson Education Limited, 2004, 617 p.

[2] Белик А.Г., Цыганенко В.Н. *Качество и надежность программных систем*. Омск, ОмГТУ, 2018, 80 с.

[3] *ИСО ISO/IEC*. URL: https://www.iso.org/ru/standard/35746.html (дата обращения 08.09.2021).

[4] DeGrace P., Hulet Stahl L. *Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms*. N.J., Yourdon Press, 1990, 244 p.

[5] Виггерс К., Бити Д. *Разработка требований к программному обеспечению*. Санкт-Петербург, Русская редакция, 2014, 737 с.

[6] Колина А.М. Метрики и атрибуты оценки качества программного обеспечения. *Инновации в науке, образовании и бизнесе*, 2018, № 1, с. 92–99.

[7] Бураков Д.П., Кожомбердиева Г.И. Использование формулы Байеса при оценивании качества программного обеспечения согласно стандарту ISO/IEC 9126 ЕС 9126. *Программные продукты и системы*, 2019, № 1, с. 34–41. http://doi.org/10.15827/0236-235X.125.034-041

[8] Шабанов Р.М., Левченков А.Н. Объединение кластеризации и классификации для оценки качества программного обеспечения. *Молодой исследователь Дона*, 2019, т. 3, № 18, с. 104–105.

Власенко Артем Сергеевич — студент кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Серёгина Дарья Владимировна — студентка кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Козаченко Александр Алексеевич — студент кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Звездин Александр Петрович — студент кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Бекасов Денис Евгеньевич, старший преподаватель кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Ссылку на эту статью просим оформлять следующим образом:

Власенко А.С., Серёгина Д.В., Козаченко А.А., Звездин А.П. Атрибуты качества программного обеспечения. *Политехнический молодежный журнал*, 2023, № 09 (86). http://dx.doi.org/10.18698/2541-8009-2023-9-931

SOFTWARE QUALITY ATTRIBUTES

A.S. Vlasenko vlasenkoas@student.bmstu.ru
D.V. Seryogina seryoginadv@student.bmstu.ru
A.A. Kozachenko kozachenkoAA@student.bmstu.ru
A.P. Zvezdin zap17u813@student.bmstu.ru

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

As well as the requirements for the functionality of the software, the quality of the final product plays an important role, which allows us to judge about advantages of product over analogues. In this paper considered the key concepts in the field of software quality assessment and methods of working with non-functional requirements at different stages of development. The main quality models and existing approaches to the classification and formalization of software quality attributes are analyzed. Also the roles and responsibilities of the members of the group working with quality requirements are formulated. To demonstrate the practical application of the analyzed material, a generalized quality model of the client part of the web application has been developed.

Keywords

Quality attributes, quality assessment, quality model, non-functional requirements, software development, software design, requirements formalization, software architecture

Received 18.07.2023

References

[1] Galin D. *Software quality assurance from theory to implementation*. Harlow, Pearson Education Limited, 2004, 617 p.
[2] Belik A.G., Tsyganenko V.N. *Kachestvo i nadezhnost' programmnykh sistem* [Quality and reliability of software systems]. Omsk, OmSTU Publ., 2018, 80 p. (In Russ.).
[3] *ISO ISO/IEC*. URL: https://www.iso.org/ru/standard/35746.html (accessed September 08, 2021).
[4] DeGrace P., Hulet Stahl L. *Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms*. N.J., Yourdon Press, 1990, 244 p.
[5] Viggers K., Biti D. *Razrabotka trebovaniy k programmnomu obespecheniyu* [Software requirements]. Sankt-Petersburg, Russkaya redaktsiya Publ., 2014, 737 p. (In Russ.).
[6] Kolina A.M. Metrics and attributes of quality assessment of software. *Innovation in science, education and business*, 2018, no. 1, pp. 92–99. (In Russ.).
[7] Burakov D.P., Kozhomberdieva G.I. Using the Bayes' theorem within software quality evaluation according to ISO/IEC 9126 standard. *Programmnye produkty i sistemy*, 2019, no. 1, pp. 34–41. (In Russ.). http://doi.org/10.15827/0236-235X.125.034-041
[8] Shabanov R.M., Levchenkov A.N. Combining clustering and classification for software quality evaluation. *Young Don researcher*, 2019, vol. 3, no. 18, pp. 104–105. (In Russ.).

Vlasenko Artem Sergeevich — Student, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Seryogina Darya Vladimirovna — Student, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Kozachenko Aleksandr Alekseevich — Student, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Z**vezdin Aleksandr Petrovich** — Student, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Bekasov D.E., Senior Lecturer, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Please cite this article in English as:
Vlasenko A.S., Seryogina D.V., Kozachenko A.A., Zvezdin A.P. Software quality attributes. *Politekhnicheskiy molodezhnyy zhurnal*, 2023, no. 09 (86). (In Russ.).
http://dx.doi.org/10.18698/2541-8009-2023-9-931