Анализ моделей оценки качества вычислительной системы

К. В. Нарышкин

***Аннотация*. В статье рассматриваются вопросы оценки качества закрытой вычислительной системы в процессах приобретения и поставки. Рассматриваются описательные модели качества, предложенные в научной литературе и нормативных документах. Проведено совершенствование аналитической модели качества программной системы. Разработана теоретическая основа для расчета показателей качества «пропускная способность» и «ресурсоемкость». Введено понятие влияния ошибок в бинарном коде на оценки показателей качества. На основе методики расчета оценки влияния уязвимости в системе на безопасность информации предложена методика расчета оценки влияния ошибок на остальные характеристики качества вычислительной системы. Представлены достоинства и недостатки методов поиска ошибок в вычислительной системе в условиях отсутствия доступа к процессу разработки. В заключении обозначены проблемы методов оценки показателей качества и поиска ошибок в закрытых вычислительных системах.**

***Ключевые слова:* оценка качества, поиск ошибок ПО, вычислительные системы, информационные технологии, CVSS, модель качества ПО, анализ бинарного кода.**

I. ВВЕДЕНИЕ

В процессах приобретения и поставки иностранных информационных технологий необходимо оценивать качество получаемой продукции. Понятие «качество информационной системы» определяется как степень удовлетворения системой заявленных и подразумеваемых потребностей различных заинтересованных сторон, которая позволяет таким образом оценить достоинства [1]. Под качеством понимается весь объем признаков и характеристик продукции или услуги, который относится к их способности удовлетворять установленным или предполагаемым потребностям [2].

Полнота, достоверность и точность оценки качества задаются требованиями заинтересованных сторон. Качество создаваемой или существующей информационной инфраструктуры зависит от оценки и прогнозирования качества поставляемых технологий, в том числе иностранных.

Модели качества программных средств и систем учитывают множество состояний оцениваемого продукта, а также ограничения, с которыми могут столкнуться все заинтересованные стороны при оценке качества. В научной литературе и в руководящих документах предлагаются различные методы оценки показателей качества (ПК) [1, 2]. Существующие методы оценки основаны на метрическом контроле системы и развиваются в контексте поддержки качества в процессах разработки и сопровождения продукта производителем. В таком случае, следует учитывать тот факт, что имеющиеся методы контроля качества не позволяют рассчитывать метрики готовой продукции.

Существующие модели качества не позволяют обоснованно предъявлять многокритериальные требования к переносимым информационным технологиям (ИТ). Модели и методы способны оценить качество ИТ только в том случае, когда все базовые показатели качества рассчитаны, или рассчитаны риски, которые компенсируют неопределенность в показателях. В ином случае точность оценки не будет удовлетворять текущим требованиям. Оценка качества иностранных информационных технологий (ИИТ) сталкивается с проблемами применения существующих моделей и методов. К таким проблемам относятся невозможность рассчитать ПК, невозможность измерить элементы ПК (ЭПК), отсутствие источников данных, отсутствие экспертного сообщества, и как следствие, полнота, достоверность и точность оценки качества не удовлетворяют первоначальным требованиям. В сложившейся ситуации приходится интегрировать технологии, точность оценки которых занижена.

Представленная модель качества информационной системы отражает особенности оценки качества.

II. АНАЛИЗ МОДЕЛЕЙ КАЧЕСТВА

Большое количество общих признаков и характеристик в разнородных программно-аппаратных продуктах способствует созданию структурированной системы. Такие системы разнородных характеристик называют моделями качества информационных систем. В научной литературе разработано и описано множество моделей качества, как программного обеспечения, так и программно-аппаратных систем. Нормативные документы [1, 2] по оценке качества

информационных систем обобщают и расширяют модели Боэма, МакКола, FURPS, Дроми, Гилба, GQM.

ГОСТ Р ИСО/МЭК 25010-2015 разделяет качество на две категории: модель качества при использовании и модель качества системы.

Модель качества при использовании применяется в случаях оценки человеко-машинных систем в определенных условиях. Такая модель состоит из пяти характеристик качества:

- эффективность;
- производительность;
- удовлетворенность;
- свобода от риска;
- покрытие контекста.

Модель качества системы характеризует информационную систему или программное обеспечение по статическим и динамическим свойствам. К характеристикам модели относятся:

- функциональная пригодность;
- уровень производительности;
- совместимость;
- удобство использования;
- надежность;
- защищенность;
- сопровождаемость;
- переносимость.

Характеристики качества разделяют на подхарактеристики, которые, в свою очередь, состоят из ПК. ПК определяются функциями, а входными параметрами являются ЭПК. ЭПК получают с помощью расчетных, измерительных, экспертных методов.

В работах [3, 4] определена аналитическая модель качества, которая несколько отличается содержанием, но структурно идентична тем моделям, что представлены в ГОСТ [1] и [2].

Уточним эталонную модель качества информационных систем, которая описана в работе [3], за счет детализации показателей качества и применения систем массового обслуживания для расчета производительности.

Для унификации заменим понятие «атрибут качества» на «ПК» и добавим понятие «ЭПК». Модель качества информационной системы записывается в виде

$$M_Q = \{Q, A, M, E, W\},$$

где $Q = \{q_1, q_2, \dots, q_i\}\ i = 1, \dots, I$, - множество характеристик качества;

$A = \{a_1, a_2, \dots, a_j\}\ j = 1, \dots, J$, - множество ПК, каждый a_j отражает свойство характеристики качества q_i;

$M = \{m_1, m_2, \dots, m_k\}\ k = 1, \dots, K$, - множество метрик для нормализации каждого ПК a_j;

$E = \{e_1, e_2, \dots, e_n\}\ n = 1, \dots, N$, - множество ЭПК ПК a_j;

$W = \{w_1, w_2, \dots, w_l\}\ l = 1, \dots, L$, - множество весовых коэффициентов для множества M.

В работе Лаврищевой Е. М. [3] выбирается шесть характеристик качества:

- функциональность;
- надежность;
- удобство применения;
- эффективность;
- сопровождаемость;
- переносимость.

Липаев В. В. [4] также выделяет шесть характеристик с единственным отличием в том, что он использует понятие «мобильность» вместо «переносимость».

Понятие «характеристика качества» объединяет множество ПК, что требует детализации и формализации. Оценка характеристик «функциональная пригодность», «надежность», «производительность» и «защищенность» обладает полнотой при оценке качества вычислительной системы. Оценка таких характеристик, как «удобство применения», «сопровождаемость», «переносимость», вторична в случаях, когда система не удовлетворяет требованиям к базовым характеристикам. Выделим те характеристики, требования к которым должны быть строгими для всех программно-аппаратных систем. К ним относятся:

q_1 - функциональная пригодность;
q_2 - надежность;
q_3 - производительность;
q_4 - защищенность.

Существующие методики оценки информационных систем разделяют понятия качества и безопасности. Поэтому характеристика «защищенность» относится к общему понятию безопасности информации, и таким образом её выделяют не как характеристику качества, а как характеристику безопасности.

III. АНАЛИТИЧЕСКАЯ МОДЕЛЬ КАЧЕСТВА ИНФОРМАЦИОННОЙ СИСТЕМЫ

На основе эталонной модели качества информационной системы составляется модель качества, по которой оцениваются базовые характеристики.

Функциональная пригодность определяется *функциональной полнотой* (a_{11}), *корректностью* (a_{12}), *точностью* (a_{13}), *целесообразностью* (a_{14}), *согласованностью* (a_{15}):

$q_1 = \{a_{11}, a_{12}, a_{13}, a_{14}\}$.

При расчетах функциональной пригодности важно выделить функциональные компоненты системы. К таким компонентам относятся физические компоненты (процессоры, цифровые преобразователи сигналов, чипы памяти, устройства ввода/вывода) и программные компоненты (операционные системы, программные функции, микропрограммы контроллеров, базы данных). Рассмотрим

характеристику «функциональная пригодность» на примере, где в качестве компонента системы выступает программная функция (ПФ).

Параметры ПФ делятся на два класса: общие и заданные в спецификации. К общим параметрам ПФ могут быть отнесены следующие элементы e_i^f множества E_f:

e_1^f - количество входных переменных;
e_2^f - количество выходных переменных;
e_3^f - типы входных данных;
e_4^f - время выполнения;
e_5^f - количество циклов;
e_6^f - количество условий;
e_7^f - количество обработок ошибок;
e_8^f - количество вызываемых функций;
e_9^f - количество точек вызова функции в системе;
e_{10}^f - уникальность кода функции;
e_{11}^f - уникальность функции для системы;
e_{12}^f - функция реализована в соответствии со стандартами языка программирования;
e_{13}^f - функция реализована в соответствии с принятыми отраслевыми стандартами;
e_{14}^f - функция реализована в соответствии со стандартами предприятия.

Параметры ПФ, которые задаются в требованиях, определяют либо точное значение общих параметров, либо их диапазон; а также требования задают применение конкретных алгоритмов, стандартов, протоколов взаимодействия, форматов данных. Если в требованиях нет явного указания на общие параметры реализации функций, то необходимо учитывать допустимость значений.

Функциональная полнота – это отношение всех реализованных компонент (F^c) в вычислительной системе к компонентам, заданным в требованиях (F^m):

$$a_{11} = \sum_{i=1}^{N} F^c / \sum_{j=1}^{K} F^m$$

ПФ считается реализованной, если она способна выполнять свое предназначение. При расчете ПК не учитываются общие и специальные характеристики функций. Проводится подсчет ПФ, которые содержатся в программном коде, и функций, заданных в требованиях.

Корректность – это соответствие реализованных компонент (F^c) заданным компонентам в требованиях (F^m):

$$a_{12} = 1 - (|F^m \backslash F^c|)/|F^m|$$

Функция F^c соответствует функции F^m, если множество её параметров $\{e_1^c, e_2^c, \dots, e_n^c\}$ соответствует множеству параметров $\{e_1^m, e_2^m, \dots, e_n^m\}$, заданных в требованиях для функции F^m, и они находятся в границах допустимых значений.

Выделяют полную корректность, когда все реализованные функции соответствуют заданным в требованиях ($F^c = F^m$), и частичную корректность, когда $F^c \cap F^m \neq \emptyset$.

Точность – это свойство системы, которое отражает правильность полученных значений функциональными компонентами системы на входном наборе данных. Точность оценивается как сумма разницы значений индикаторных функций, которые зависят от входных данных

$$a_{13} = \sum_{i=1}^{|F_m|}((F_i^c(D_i) - F_i^m(D_i))/F_i^m(D_i))/|F^m|,$$

где D – это множество всех входных значений;

Dr – это множество, на котором функция $F_i^m(D_i) = 1$, отсюда следует, что функция $F_i^m(D_i) = \begin{cases} 1, \text{если } D_i \in Dr \\ 0, \text{если } D_i \notin Dr \end{cases}$ (1)

Индикаторная функция $F_i^c(D_i)$ в области формальной верификации программ может быть определена как *корректность функции* через логику Хоара.

Пусть $F_i^c(D_x, D_z)$ – функция F с входными параметрами D_x и выходными D_z. Предусловие функции обозначим $F_{pre}(D_x)$, а постусловие – $F_{post}(D_x, D_z)$.

Определение 1. Функция $F_i^c(D_x, D_z)$ корректна частично или полностью по отношению к предусловию $F_{pre}(D_x)$ и постусловию $F_{post}(D_x, D_z)$, если из истинности $F_{pre}(D_x)$ следует, что для функции $F_i^c(D_x, D_z)$ гарантируется выполнение $F_{post}(D_x, D_z)$ при условии завершения функции на этом входе.

Из формулы (1) следует, что если функция $F_i^c(D_i)$ всегда гарантирует выполнение постусловия, то $D = Dr$, показатель $a_{13} = 0$, и можно сделать вывод о том, что разницы нет между требуемыми значениями и получаемыми от системы. В таких случаях говорят, что система работает точно в соответствии с требованиями.

Целесообразность – это отношение компонент системы, обеспечивающих её функционирование (F^a), к общему количеству реализованных компонент в системе (F^c). Избыточность в системе является дополнительным источником возникновения отказа или сбоя в системе. Для программных функций целесообразность рассчитывается следующим образом:

$$a_{14} = \sum_{i=1}^{N} F^a / \sum_{j=1}^{K} F^c,$$

где F^a – это индикаторная функция, которая принимает значение 1, когда параметры e_i^c функции F_i^c имеют такие значения, которые доказывают необходимость использования ПФ в системе.

Согласованность – это отношение компонент системы, реализованных в соответствии со

стандартами (F^s), ко всем реализованным компонентам (F^c):

$$a_{15} = \sum_{i=1}^{N} F^s / \sum_{j=1}^{K} F^c,$$

где F^s – это индикаторная функция, которая принимает значение 1, когда параметры соответствия e_i^c (например, e_{12}^f, e_{13}^f, e_{14}^f) функции F_i^c принимают ненулевые значения.

Надежность вычислительных систем определяется *безотказностью* (a_{21}), *устойчивостью к ошибкам* (a_{22}), *восстанавливаемостью* (a_{23}).

Безотказность – это способность системы функционировать без отказов программных и аппаратных компонент. Безотказность оценивается вероятностью p безотказного выполнения компонента на случайно выбранном наборе входных данных. Для вычислительных систем применяется такой показатель, как средняя наработка на отказ:

$$a_{21} = \sum_{i=1}^{m} t_i / m,$$

где t_i - интервал времени безотказной работы при наступлении i-го отказа;

m – количество отказов.

Устойчивость к ошибкам – это способность отдельных компонент и системы в целом функционировать в нехарактерных условиях. Степень устойчивости системы оценивается как отношение различных типов отказов (N^*) к общему количеству отказов, возникающих в системе (N):

$$a_{22} = \frac{N^*}{N}$$

Восстанавливаемость – это способность системы восстанавливать функционирование после возникновения отказа. Восстанавливаемость оценивается с помощью измерения среднего времени, затраченного на восстановление:

$$a_{23} = \sum_{i=1}^{m} t_i^r / m$$

где t_i^r – интервал времени, который необходим системе для восстановления после i-го отказа;

m – количество отказов.

Производительность вычислительных систем определяется *пропускной способностью* (a_{31}), *временем отклика* (a_{32}) и *ресурсоемкостью* (a_{32}).

Модели систем массового обслуживания смешанного типа [5] позволяют описать вычислительную систему с целью расчета показателей производительности.

Пропускная способность – это отношение среднего числа обработанных заявок за единицу времени к среднему числу поданных заявок. Относительная пропускная способность системы – это вероятность того, что заявка в момент времени t не получит отказ.

$$a_{31}(t) = 1 - p_n(t),$$

где $p_n(t)$ – вероятность того, что заявка в момент времени t получит отказ.

Время отклика (a_{32}) – это сумма среднего время ожидания заявки ($T_{\text{ож}}$) и среднего времени обслуживания ($T_{\text{об}}$):

$$a_{32} = T_{\text{ож}} + T_{\text{об}}$$

Ресурсоемкость – это количество используемых ресурсов при выполнении функции системы. Ресурсы компьютерной системы – это совокупность вычислительной мощности процессора(-ов) и используемой памяти. Ресурсоемкость за единицу времени выражается как совокупность загруженности процессора и памяти

$$a_{33} = \{Cl(t), Vm(t)\},$$

где $\mathrm{C}l(t)$ – это загруженность процессора за время выполнения функции; $Vm(t)$ – степень используемой памяти от общего количества за время выполнения функции.

Количественная оценка каждой характеристики качества может быть рассчитана как сумма всех нормализованных ПК, умноженных на весовой коэффициент

$$q_i = \sum_{j=1}^{N} a_{ij} m_{ij} w_{ij},$$

где N – количество ПК в q_i-ой характеристике.

В свою очередь интегральная оценка качества продукта есть сумма всех характеристик качества.

$$Q = \sum_{i=1}^{3} q_i$$

Представленная выше модель качества вычислительной системы в значительной степени зависит от ошибок в бинарном коде. Рассмотрим подробнее влияние ошибок на характеристики качества системы.

IV. ВЛИЯНИЕ ОШИБОК НА ХАРАКТЕРИСТИКИ КАЧЕСТВА

Ошибки в бинарном коде системы способны влиять на все характеристики качества. Особенно важно оценивать количество и качество ошибок, которые затрагивают базовые характеристики (функциональность, надежность, производительность и безопасность).

Во время эксплуатации системы отказы или нарушения безопасности возникают вследствие преднамеренных и случайных ошибок, оставленных в

бинарном коде системы. Причины и следствия преднамеренных ошибок характеризуются компромиссом между значениями характеристик качества в условиях ограниченных возможностей при разработке системы. Например, межсетевые экраны сталкиваются с проблемой снижения пропускной способности сетевого канала, когда в проходящем трафике присутствует сверхбольшое количество детектируемых опасных данных для пользователей сети. В такой ситуации разработчики программно снижают уровень безопасности в пользу производительности.

Причины и следствия непреднамеренных ошибок имеют случайный характер. Значительная неопределенность влияния таких ошибок на базовые характеристики, даже после выявления и изменения ошибочного кода, оставляет высокую вероятность ошибки, если не учитывалось влияние на все характеристики качества. Например, отказ в обслуживании, вызванный закончившимся местом в оперативной памяти, не может быть устранен увеличением объема памяти, если источником отказа стала ошибка утечки памяти в программном коде.

Далее определим, каким образом ошибки в бинарном коде влияют на характеристики качества вычислительной системы.

В теории надежности систем важным критерием является количество отказов в системе. Отказы обусловлены внешними и внутренними факторами. К внешним факторам относятся действия пользователей. Внутренними факторами являются дефекты в системе. Дефекты в вычислительной системе прежде всего связаны со случайными или преднамеренными ошибками в программном коде и аппаратных компонентах. Надежность системы характеризуется такой цепочкой понятий, как «ошибка – дефект – сбой – системный сбой – локальный отказ – системный отказ». Отсюда следует, что ошибки в бинарном коде системы уменьшают время безотказной работы t_i и степень устойчивости системы к ошибкам t_i^r.

При расчете функциональной пригодности оценка ошибок в бинарном коде системы обуславливает такие ПК, как функциональная полнота, корректность и точность системы. Ошибки в системе сокращают количество реализованных и корректных компонент, так как согласно определению 1 невозможно гарантировать выполнение постусловия программной функции.

Ошибки в вычислительной системе также влияют на показатели производительности. При сокращении каналов обработки заявок из-за ошибок сокращается пропускная способность системы и увеличивается время отклика, причем сама система может функционировать без сбоев. А в системах, где присутствуют ошибки работы с памятью («утечка памяти»), зачастую увеличивают её расход в сравнении с теми же функциями без ошибок, таким образом сокращая доступный ресурс системы.

Случайность возникновения ошибок обуславливает неопределенность в целевой функции, а большой объем бинарного кода повышает сложность задачи выявления ошибок и определения степени их влияния на характеристики качества. Поэтому для решения такой задачи необходимо проведение исследований и создание методик.

V. ПРОБЛЕМА НЕОДНОРОДНОСТИ ОШИБОК В БИНАРНОМ КОДЕ

Неоднородность ошибок в программно-аппаратных системах обусловлена контекстом и природой их проявления. Будет неправильно говорить только о количестве обнаруженных ошибок при расчетах ПК системы. Важной составляющей каждой ошибки являются условия её проявления, влияние на другие компоненты системы и ряд других метрик. Отечественные [6, 7, 8, 9] и международные [10] научные организации проводят исследования по разработке методов оценки влияния ошибок на информационную безопасность системы. Наиболее известна международная методика Common Vulnerability Scoring System Version 3.0 (CVSS v3.0). Методика состоит из 8 базовых (таблица 1), 3 временных и 11 контекстных метрик [6, 7].

Таблица 1 - Базовые метрики CVSS

Вектор атаки (AV)			
Сетевой (N) 0.85	Смежная сеть (A) 0.62	Локальный (L) 0.55	Физический (P) 0.2
Сложность атаки (AC)			
Высокая (H) 0.44		Низкая (L) 0.77	
Уровень привилегий (PL)			
Высокий (H) 0.27	Низкий (L) 0.62		Не требуется (N) 0.85
Взаимодействие с пользователем (UI)			
Требуется (R) 0.62		Не требуется (N) 0.85	
Влияние на другие компоненты системы (S)			
Не оказывает (U)		Оказывает (C)	
Влияние на конфиденциальность (C)			
Не оказывает (N) 0	Низкое (L) 0.22		Высокое (H) 0.56
Влияние на целостность (I)			
Не оказывает (N) 0	Низкое (L) 0.22		Высокое (H) 0.56

Влияние на доступность		
Не оказывает (N) 0	Низкое (L) 0.22	Высокое (H) 0.56

Методика оценки ошибок CVSS v3.0 систематизирует ошибки в системе. Шкала от 0 до 10 делит ошибки по уровню влияния: низкий (0.1−3.9), средний (4.0−6.9), высокий (7.0−8.9) и критический (9.0−10.0).

Оценка влияния ошибки на информационную безопасность по базовым метрикам может быть использована как математическая модель для создания подобных формализаций в других характеристиках качества. При создании CVSS [8] коэффициенты для формул подбирались на основе статистического анализа и уточнялись от версии 1.0 до 3.1. Весовые коэффициенты из таблицы 1 подбирались в рамках расчета простого среднего взвешенного значения оценки.

Оценка состоит из трёх следующих критериев: оценка влияния по показателям безопасности информации (E_{iss}) (2), оценка сложности применения ошибки с целью нарушения безопасности информации (E_e) (3) и влияние на другие компоненты (S).

$$E_{iss} = 1 - [(1 - C) * (1 - I) * (1 - A)], \quad (2)$$

где C – конфиденциальность;
I – целостность;
A – доступность.

$$E_e = 8.22 * AV * AC * PR * UI \ , \quad (3)$$

где AV – вектор атаки;
AC – сложность атаки;
PR – уровень привилегий;
UI – взаимодействие с пользователем.

Влияние ошибки на другие компоненты системы учитывается в (4)

$$E_i = \begin{cases} 6.42 * E_{iss}, \text{если } S = 0 \\ 7.52 * (E_{iss} - 0.029) - 3.25 * (E_{iss} - 0.02)^{15}, \end{cases} \quad (4)$$

где S – оценка влияния на другие компоненты системы.

Оценка влияния ошибки на информационную безопасность (5) имеет три состояния:

$$E_{bs} = \begin{cases} 0, \text{если } E_i \le 0 \\ \min\big((E_i + E_e)\ ,10\big), \text{если } S = 0 \\ \min(1.08 * (E_i + E_e)\ ,10), \text{если } S = 1 \end{cases} \quad (5)$$

Представленная выше методика будет иметь другие метрики и коэффициенты для функциональной полноты, корректности выполнения функций, точности выполнения программно реализованных алгоритмов, пропускной способности канала обработки данных, времени отклика на обработку заявки, объема вычислительных ресурсов; вероятности безотказной работы, времени на восстановление после отказа, вероятности отказа в обслуживании, что требует исследования в этой области. В рамках исследования требуется провести адаптацию метрик CVSS, а те из них, которые относятся непосредственно к безопасности, заменить на более подходящие к конкретной характеристике.

Как и для безопасности, метрики оценки влияния ошибки на характеристики качества делятся на 3 группы:

1. Возможность выполнения ошибочного кода в системе;
2. Влияние ошибки на компоненты системы;
3. Влияние на ПК.

В таблице 2 представлена декомпозиция CVSS на метрики в базисе функциональности, производительности и надежности.

Таблица 2 – Метрики оценки влияния ошибки на систему

Характеристика	Метрика	Значения	Описание
1 группа метрик			
Безопасность	Вектор атаки	Сетевой Смежная сеть Локальный Физический	Указывает на расположение источника возможной угрозы безопасности.
	Сложность атаки	Высокая Низкая	Комплексная оценка сложности эксплуатации ошибки злоумышленником.
	Уровень привилегий	Высокий Низкий Не требуется	Выбирается максимально низкий уровень привилегий, которыми должен обладать злоумышленник, чтобы проэксплуатировать ошибку.
	Взаимодействие с пользователем	Требуется Не требуется	Значение зависит от участия пользователя системы при эксплуатации ошибки.
Функциональность	Компонент системы	Сетевой Программный	Указывает на подсистему, где обнаружена ошибка.

Характеристика	Метрика	Значения	Описание
		Аппаратный	
	Сложность алгоритма нарушения функциональности	Высокая Низкая	Оценка сложности пути до кода, содержащего ошибку. Оценка состоит из вероятности формирования данных, обработка которых приводит к ошибке, и вероятности попадания потока управления на код, содержащий ошибку.
	Уровень привилегий	Высокий Низкий Не требуется	Выбирается тот уровень привилегий, которым должен обладать пользователь, чтобы проявилась ошибка.
	Взаимодействие с пользователем	Требуется Не требуется	Значение зависит от участия пользователя системы для проявления ошибки.
Производительность	Источник нагрузки	Сетевой Смежная сеть Локальный Физический	Указывает на расположение источника данных, при обработке которых возникает ошибка, влияющая на производительность системы.
	Сложность алгоритма возникновения ошибки	Высокая Низкая	Оценка сложности алгоритма, который приводит к выполнению ошибочного кода.
	Уровень привилегий	Высокий Низкий Не требуется	Выбирается тот уровень привилегий, которым должен обладать пользователь, чтобы проявилась ошибка.
	Взаимодействие с пользователем	Требуется Не требуется	Значение зависит от участия пользователя системы для проявления ошибки

Таблица 2 - Продолжение

Характеристика	Метрика	Значения	Описание
Надежность	Источник данных отказа	Сетевой Смежная сеть Локальный Физический	Указывает на расположение источника данных, при обработке которых возникает ошибка, вызывающая сбой или отказ в системе.
	Сложность алгоритма возникновения отказа	Высокая Низкая	Оценка сложности возникновения отказа из-за ошибки.
	Уровень привилегий	Высокий Низкий Не требуется	Выбирается тот уровень привилегий, которым должен обладать пользователь, чтобы сформировать данные, которые вызовут отказ или сбой в системе из-за ошибки
	Взаимодействие с пользователем	Требуется Не требуется	Значение зависит от участия пользователя системы для вызова отказа или сбоя в системе из-за ошибки.
2 группа метрик			
Безопасность	Влияние на другие компоненты системы	Не оказывает Оказывает	Принимается значение «1» или «0» соответственно в зависимости от наличия или отсутствия влияния ошибки на смежные компоненты системы.
Функциональность	Влияние на другие компоненты системы	Не оказывает Оказывает	
Производительность	Влияние на другие компоненты системы	Не оказывает Оказывает	

Надежность	Влияние на другие компоненты системы	Не оказывает Оказывает	
3 группа метрик			
Безопасность	Влияние на конфиденциальность Влияние на целостность Влияние на доступность	Не оказывает Низкое Высокое	Определяет средневзвешенное значение вероятности влияния ошибки на такие ПК безопасности, как конфиденциальность, целостность и доступность информации.
Функциональность	Влияние на полноту Влияние на корректность Влияние на точность	Не оказывает Низкое Высокое	Определяет средневзвешенное значение вероятности влияния ошибки на такие ПК функциональности, как полнота реализации, корректность и точность выполнения программно реализованных алгоритмов.
Производительность	Влияние на пропускную способность Влияние на время отклика Влияние на объем ресурсов	Не оказывает Низкое Высокое	Определяет средневзвешенное значение вероятности влияния ошибки на такие ПК производительности, как пропускная способность канала обработки данных, время отклика на обработку заявки и объем вычислительных ресурсов.
Надежность	Влияние на безотказность Влияние на восстанавливаемость Влияние на доступность	Не оказывает Низкое Высокое	Определяет средневзвешенное значение вероятности влияния ошибки на такие ПК надежности, как вероятность безотказной работы, среднее время на восстановление после отказа, вероятность отказа в обслуживании.

Для определения коэффициентов из формул (2–5) требуются статистические данные. Источником статистических данных могут служить отчеты об исправленных ошибках в системах с открытым исходным кодом, но такое решение будет неполным, так как промышленные программно-аппаратные системы не предоставляют доступ к исходной проектной документации и исходным кодам ПО. В таком случае целесообразно использовать методы восстановления проектной документации и исходных кодов с помощью обратного проектирования.

Также адаптировать методику CVSS возможно с использованием аппарата нечеткой логики. Существующая методика позволяет проверить полученную модель на точность и достоверность. Таким образом, используя таблицу 2, возможно создать базу нечетких правил, по которым, выбрав один из методов дефазификации, рассчитать базовую оценку влияния.

Расчет оценки влияния ошибок на систему по разным характеристикам качества зависит от того, насколько точно будут определены метрики. Определить метрики ошибок, возникающих в «черном ящике», – трудоемкий процесс, который использует методы косвенного определения. На этапах проектирования и разработки оценщик имеет доступ к структуре системы, технической спецификации и имеет неограниченные возможности по доступу ко всем компонентам системы. Восстановление технической документации, структуры системы, исходного кода необходимо для определения значений в метриках влияния ошибок.

Из вышесказанного следует, что выявление ошибок в компонентах системы занимает ключевую роль в оценке качества информационных систем. Выявление ошибок включает две задачи:

1. Поиск ошибок в реализации компонент системы.
2. Оценка влияния ошибок на характеристики качества системы.

VI. МЕТОДЫ ПОИСКА ОШИБОК В БИНАРНОМ КОДЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

В работах Лаврищевой Е. М. [3], Липаева В. В. [4], Белеванцева А. А. [11], Новикова В. А. [12], Буйневича М. В. [13] рассматриваются методы

обнаружения ошибок на этапах проектирования, разработки и ввода в эксплуатацию.

Важной проблемой является достаточность исходных данных для задач выявления ошибок. На разных этапах ЖЦ системы достаточность исходных данных отличается, но чем ближе система к готовому продукту, тем больше источников данных требуется для обнаружения ошибок в системе, так как состав методов зависит от доступности исходных данных. Методы поиска ошибок в информационной системе представлены в таблице 3, где красным и розовым цветом обозначена повышенная сложность, с которой сталкиваются специалисты при применении методов поиска ошибок при приобретении и поставке программно-аппаратных систем.

Таблица 3 – Методы поиска ошибок в ПАС

Метод поиска ошибок	Готовый образец системы	Исходные коды	Информативность	Скорость	Эффективность
Анализ открытой технической документации	Не требуется	Не требуется	Низкая	Высокая	Низкая
Статический и динамический анализ исходного кода	Требуются	Требуются	Высокая	Средняя	Высокая
Статический и динамический анализ бинарного кода	Требуются	Не требуются	Высокая	Низкая	Средняя

Таблица 3 - Продолжение

Метод поиска ошибок	Готовый образец системы	Исходные коды	Информативность	Скорость	Эффективность
Измерение параметров исходного кода	Требуется	Требуется	Средняя	Высокая	Средняя
Измерение параметров функционирования системы	Требуется	Не требуется	Высокая	Средняя	Средняя
Аппаратная отладка системы	Требуется	Не требуется	Высокая	Низкая	Средняя
Модульное, интеграционное и системное тестирование	Требуются	Требуются	Высокая	Средняя	Высокая
Нагрузочное тестирование	Требуется	Не требуется	Средняя	Низкая	Средняя
Поблочное тестирование	Требуется	Требуется	Высокая	Средняя	Высокая

В общем случае точность оценки качества готовой информационной системы зависит от наличия следующих исходных данных:

1. Готовый образец системы.
2. Техническая документация на систему.
3. Исходные коды ПО.
4. Технические требования к образцу системы.

В случае приобретения и поставки иностранных информационных систем зачастую имеются только готовый образец продукта и его техническое описание, что снижает точность и скорость метода оценки качества (таблица 3). В таких случаях метод оценки сводится к тестированию системы и анализу открытых источников данных, например, анализ дерева отказов в аналогичных системах.

Для того чтобы пользоваться всем набором известных методов, необходимо разработать метод восстановления информации и на её основе составить параметрические модели системы.

К восстанавливаемой информации вычислительной системы относятся:

1. Исходные коды системы.
2. Алгоритмы вычислений.
3. Логические алгоритмы.
4. Общие алгоритмы функционирования.
5. Внутрисистемные протоколы обмена данными.
6. Сетевые протоколы обмена данными.
7.

Таким образом, при наличии готового образца системы, восстановленных исходных кодов, восстановленных алгоритмов функционирования системы и технических требований к образцу системы задача по оценке оставленных ошибок в информационной системе сводится к той, что решается разработчиками на этапах проектирования, разработки и внедрения. Отсюда следует, что, используя методы восстановления информации, возможно рассчитывать ПК и дать интегральную оценку качества информационной системе в условиях ограниченного набора исходных данных.

VII. ЗАКЛЮЧЕНИЕ

Определены базовые характеристики качества информационной системы и составлена математическая модель. Показана связь между основными ПК и ошибками, которые присутствуют в системе, а также оценка их влияния на систему. Неоднородность ошибок не позволяет использовать только количественные метрики, необходимо оценить влияние на функциональность, производительность и надежность. Часто используемый ЭПК «количество ошибок» не коррелирует с ПК, так как не учитывает качественные метрики самих ошибок. Основная проблема при оценке качества приобретаемой информационной системы связана с отсутствием исходных данных, что препятствует обнаружению ошибок и расчету ключевых ПК. Решением этой проблемы может стать метод восстановления исходных кодов и алгоритмов функционирования. Разработка метода поиска ошибок в готовом образце системы без исходных текстов на основе восстановленной информации позволит повысить скорость оценки качества систем, а методика оценки влияния каждой найденной ошибки на систему способна повысить точность ПК.

БИБЛИОГРАФИЯ

[1] ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. – М.: Стандартинформ, 2015 – 36 с.

[2] ГОСТ 28195-89 Оценка качества программных средств. Общие положения. – М.: ИПК издательство стандартов, 1989 – 31 с.

[3] Лаврищева Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства: Учеб. для вузов — 2-е изд., испр. — М.: Издательство Юрайт, 2016. — 280 с.

[4] Липаев В. В. Программная инженерия: методологические основы: Учеб. для вузов – М.-Берлин: Директ-Медиа, 2015. — 608 с.

[5] Вентцель Е. С. Теория вероятностей: Учеб. для вузов. — 6-е изд. стер. — М.: Высш. шк., 1999.— 576 с.

[6] Банк данных угроз безопасности: сайт ФСТЭК России. [Электронный ресурс]. 2022. Дата обновления: 12.10.2022. URL: https://bdu.fstec.ru/calc31 (дата обращения: 14.07.2023).

[7] Липаев В. В. Функциональная безопасность программных средств // М.: СИНТЕГ, 2004. – Т. 348. – С. 348.

[8] Василенко М. Н. и др. Ошибки в технической документации железнодорожной автоматики и телемеханики и их влияние на безопасность движения поездов // Автоматика на транспорте. – 2019. – Т. 5. – №. 1. – С. 94-112.

[9] Н. В. Пакулин, Е. М. Лаврищева, А. Г. Рыжов, С. В. Зеленов. Анализ методов оценки надежности оборудования и систем. Практика применения методов // Труды ИСП РАН. – 2018. – Т. 30. - № 3. – С. 99–12.

[10] Общая система оценки уязвимостей версии 3.1: Спецификация стандарта: сайт FIRST. [Электронный ресурс]. 2022. Дата обновления: 19.01.2021. URL: https://www.first.org/cvss/v3.1/specification-document (дата обращения: 14.09.2022).

[11] Белеванцев А. А., Велесевич Е. А. Анализ сущностей программ на языках Си/Си++ и связей между ними для понимания программ // Труды ИСП РАН. – 2015. – Т. 27. – №. 2.

[12] Новиков В. А., Фонарёв М. О. Рекомпиляция дизассемблированных текстов программ // Вопросы защиты информации. – 2007. – №. 2. – С. 51-54.

[13] Буйневич М. В., Израилов К. Е. Исследование возможности применения машинного обучения для поиска уязвимостей в программном коде в процессе его статического анализа // Интеграция науки, общества, производства и промышленности: проблемы и перспективы. – 2020. – С. 17-22.

Статья получена 24.07.2023.

Константин Викторович Нарышкин, ООО «Статус Комплайнс», (e-mail: konstantin.n.v@outlook.com)

Analysis of models for assessing the quality of a computer system

K.V. Naryshkin

Abstract **- The article discusses the issues of assessing the quality of a proprietary computing system in the acquisition and delivery processes. Descriptive quality models proposed in scientific literature and regulatory documents are considered. The analytical model of the quality of the software system has been improved. A theoretical basis has been developed for calculating the quality indicators "throughput" and "resource intensity". The concept of the influence of errors in binary code on the evaluation of quality indicators is introduced. Based on the methodology for calculating the impact of vulnerability in the system on information security, a method for calculating the impact of errors on other characteristics of the quality of the computing system is proposed. The advantages and disadvantages of error detection methods in a computer system in the absence of access to the development process are presented. In conclusion, the problems of methods for assessing quality indicators and finding errors in proprietary computing systems are outlined.**

Keywords **- quality assessment, software error detection, computing systems, information technology, CSS, software quality model, software reverse engineering.**

REFERENCES

[1] GOST R ISO/IEC 25010-2015 Information technology. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. – M.: Standartinform, 2015 – 36 p.

[2] GOST 28195-89 Quality control of software systems. General principles. – M.: IPK Publishing House of Standards, 1989 – 31 p.

[3] Lavrishcheva E. M. Software engineering. Paradigms, technologies and CASE tools: Studies for universities — 2nd ed., ispr. — M.: Yurayt Publishing House, 2016. — 280 p.

[4] Lipaev V. V. Software engineering: methodological foundations: Studies for universities – M.-Berlin: Direct-Media, 2015. — 608 p.

[5] Wentzel E. S. Probability theory: Studies for universities. — 6th ed. ster. — M.: Higher School, 1999.— 576 p.

[6] The data bank of security threats: the website of the FSTEC of Russia. URL: https://bdu.fstec.ru/calc31 (visited: 07/14/2023).

[7] Lipaev V. V. Functional security of software tools // Moscow: SINTEG, 2004. – Vol. 348. – p. 348.

[8] Vasilenko M. N. et al. Errors in the technical documentation of railway automation and telemechanics and their impact on the safety of train traffic // Automation in transport. – 2019. – Vol. 5. – No. 1. – pp. 94-112.

[9] N. V. Pakulin, E. M. Lavrishcheva, A. G. Ryzhov, S. V. Zelenov. Analysis of methods for assessing the reliability of equipment and systems. The practice of applying methods // Proceedings of the ISP RAS. – 2018. – Vol. 30. - No. 3. – pp. 99-12.

[10] General Vulnerability Assessment System version 3.1: Specification of the standard: site FIRST. URL: https://www.first.org/cvss/v3.1/specification-document (visited: 14.09.2022).

[11] Belevantsev A. A., Velesevich E. A. Analysis of the entities of programs in C/C++ languages and the connections between them for understanding programs // Proceedings of the ISP RAS. – 2015. – Vol. 27. – No. 2.

[12] Novikov V. A., Fonarev M. O. Recompilation of disassembled program texts // Information security issues. - 2007. – No. 2. – pp. 51-54.

[13] Buinevich M. V., Izrailov K. E. Investigation of the possibility of using machine learning to search for vulnerabilities in software code in the process of its static analysis // Integration of science, society, production and industry: problems and prospects. – 2020. – pp. 17-22.