Информационное общество и право

АНАЛИЗ СОСТОЯНИЯ ТЕРМИНОЛОГИИ ПО ПРИЧИНАМ И ВИДАМ ОШИБОК В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ В РАБОТАХ РОССИЙСКИХ И ЗАРУБЕЖНЫХ АВТОРОВ

Статья рекомендована к публикации председателем редакционного совета Ю. Е. Хохловым 29.05.2023.

Ерахтина Ольга Сергеевна

Кандидат юридических наук, доцент
Национальный исследовательский университет «Высшая школа экономики», Пермский филиал, кафедра гражданского и предпринимательского права, доцент
Пермь, Российская Федерация
oerahtina@hse.ru

Попова Дарья Владиславовна

Национальный исследовательский университет «Высшая школа экономики», Пермский филиал, магистр права
Пермь, Российская Федерация
dvpopova2000@gmail.com

Аннотация

В статье рассматриваются классификации причин и видов ошибок программного обеспечения, содержащиеся в международных и национальных стандартах, научных работах российских и зарубежных специалистов. Авторами определены пределы ответственности разработчика за качество программного обеспечения в зависимости от причины и вида допущенной ошибки.

Ключевые слова

причины ошибок программного обеспечения; виды ошибок программного обеспечения; ответственность разработчика

Введение

Критерии качества программного обеспечения (далее - ПО) определены стандартами (ГОСТ Р ИСО/МЭК 9126-93, ISO/IEC 9126-1, ISO/IEC 25030, ISO/IEC 25010 и др.) и описывают идеальное состояние программного продукта. Но в реальных компьютерных системах, как правило, встречаются недостатки, потому что ПО создается людьми, а люди склонны делать ошибки.

Интересным представляется исследование компании Stack Overflow, представители которой ежегодно проводят опросы среди программистов, затрагивая разные аспекты их профессиональной жизни. По результатам опроса в 2018 году, можно сделать вывод о том, что сами разработчики не равнодушны к качеству своего продукта: из ста тысяч опрошенных 80 % согласились, что чувствуют ответственность за этические последствия своего кода. Другой значимый факт - более половины программистов считают, что реальную ответственность за продукт несет руководство, 23% указали, что ответственность на том, кем была предложена идея, и только 20 % посчитали, что ответственность остается за самим разработчиком [17].

Как отмечено в исследовании О.С. Ерахтиной и К.М. Лабутиной, разработчики ссылаются на сложность программных продуктов, непредсказуемость результатов, аргументируя исключение ответственности за качество программного обеспечения. [8]. По этой причине необходимо

https://doi.org/10.52605/16059921_2024_01_77

установить вид ошибки и причину ее появления, прежде чем привлекать разработчика к ответственности.

1 Определение понятия «ошибка ПО»

Для начала следует понять, что подразумевается под «ошибкой программного обеспечения».

Е.Б. Дроботун определяет ошибку ПО как «погрешность или искажение кода программы, неумышленно внесенные в нее в процессе разработки, которые в ходе функционирования этой программы могут вызвать отказ или снижение эффективности функционирования» [7]. Таким образом, теоретик утверждает, что единственной причиной некорректной работы ПО являются действия разработчика. Е.Б. Дроботун также классифицирует ошибки в зависимости от тяжести на:

1. Несущественные (ПО функционирует менее эффективно, а вероятность отказа (прекращение функционирования) крайне низкая);
2. Существенные (снижается эффективность работы ПО, что может вызвать отказ ПО);
3. Критические (высокая вероятность отказа ПО).

В своем исследовании Е.Б. Дроботун ссылается на положения ГОСТа 51901.12–2007 «Менеджмент риска. Метод анализа видов и последствий отказов». Надо отметить, что в 2022 году стандарт прекратил действие, и был заменен на ГОСТ Р 27.303–2021. Новый стандарт определяет ошибку в контексте неверных действий человека и содержит соответствующий термин «ошибка человека» [2]. Кроме того, стандарт вводит еще одно понятие «дефект ПО» и отождествляет его с понятием «ошибка ПО» (приложение Е к ГОСТ Р 27.303-2021). Вопреки описанной позиции необходимо отметить, что к ошибке может привести множество факторов, не зависящих от разработчика, например, неисправность оборудования, некорректное использование программы пользователем и другое. Можно предположить, что определения ошибки, указанные в ГОСТ Р 27.303–2021 и работе Е.Б. Дроботуна, являются узкими, или же существуют иные термины, описывающие проблемы некорректного поведения ПО.

В диссертации А.А. Блеванцева приведены следующие термины:

- ошибка (некорректный участок исходного кода программы, который может привести к аварийному завершению программы или выводу неверных данных);
- дефект (некорректный участок исходного кода программы, который может ухудшить эксплуатационные характеристики);
- уязвимость (ошибка в программе, которая может быть использована третьим лицом для намеренного нарушения безопасности системы) [4].

Таким образом, теоретик разделяет понятия «ошибка» и «дефект» и употребляет еще один термин - «уязвимость». Однако, как сам отмечает диссертант, на практике возникает сложность с использованием данной классификации. Во-первых, даже сами разработчики не всегда могут найти выходные данные, в которых может появиться ошибка. Во-вторых, не всякий дефект может привести к ошибке. В-третьих, почти нереально установить, является ли найденная ошибка уязвимостью [4].

Еще один термин содержится в ГОСТе 56939-2016. В национальном стандарте упоминается, что ошибка может быть допущена не только в процессе проектирования, но и в рамках реализации программы. Такая ошибка именуется как «недостаток программы» [1].

Зарубежные специалисты также считают ошибку ПО лишь одним из возможных недостатков. В некоторых стандартах и исследованиях такие недостатки названы «аномалиями» (anomalies), под которым следует понимать любое состояние программного обеспечения, отличающееся от ожидаемого [3; 14; 15 и др.].

В стандарте IEEE Standard Classification for Software Anomalies приводится следующая классификация аномалий:

1. Дефект (defect) – недостаток в работе программного обеспечения, при котором ПО не удовлетворяет требованиям к качеству, установленным в документации;
2. Ошибка (error) – недостаток в работе программного обеспечения, возникший в результате действий человека;
3. Сбой (failure) – недостаток, при котором программное обеспечение не способно функционировать так, как это было ранее;

4.Отказ (fault) – некорректная работа программного обеспечения. [3].

Отметим, что появляются ранее не рассмотренные в статье термины «сбой» и «отказ».

Итак, как следует из указанных выше национальных и международных стандартов, научных работ, до сих пор отсутствует единый подход к толкованию терминов «ошибка ПО», «дефект ПО», «уязвимость», «недостаток ПО», «сбой ПО», «аномалия ПО».

В соответствии с положениями стандартов, «аномалия» - синоним слова «недостаток программы». Поскольку последнее является выбором отечественных специалистов, в работе будем использовать его. Кроме того, анализ научной литературы показывает, что сами теоретики не могут определить разницу понятий «недостаток», «ошибка», «дефект». По этой причине в данном исследовании слова будут использоваться как синонимы. Полагаем, что термин «сбой программы» описывает поведение программного обеспечения вследствие допущенной ошибки. В определении «уязвимости программы» следуем подходу А.Ю. Герасимова, который использует данный термин для описания процесса «получения несанкционированного доступа к данным, обрабатываемым программой, или в значении несанкционированного исполнения программного кода» [5].

2 Причины и виды ошибок ПО

Можно заметить, что многие научные труды посвящены проблеме классификации ошибок программного обеспечения. Вопрос представляет интерес, поскольку в зависимости от вида ошибки меняются пределы ответственности разработчика.

В литературе по разработке программного обеспечения наиболее часто встречается классификация американского специалиста по экономике ПО Барри Боэма. Теоретик выделяет следующие виды недостатков в зависимости от стадии жизненного цикла ПО.

Во-первых, ошибки кодирования. К ним теоретик относит ошибки, которые были допущены разработчиком в исходном коде. Следовательно, ответственность за такие ошибки несет разработчик.

Во-вторых, ошибки документации. Это ошибки, которые были допущены в документации (по эксплуатации и сопровождению ПО), что привело к неверному толкованию и, как следствие, некорректному использованию ПО. Поскольку такая документация готовится разработчиком, ответственность за качество также лежит на нем.

В-третьих, Барри Боэм пишет об ошибках, которые возникли на этапе проектирования программного обеспечения [12].

М.А. Нестеренко, М.Ю. Умницын также предлагают классифицировать ошибки ПО в зависимости от этапа, на котором они были совершены. По мнению теоретиков, для проектирования свойственны такие ошибки как логическая несогласованность требований, неточности алгоритмов и модульного интерфейса и др. На этапе кодирования разработчик может допустить логические и синтаксические ошибки, организовать вычислительный процесс неоптимально. Во время тестирования может быть неверно подобран набор тестовых данных, что не позволяет обнаружить ошибку или недокументированные возможности программы, или само тестирование может быть проведено некорректно [10].

Полагаем, что обе классификации имеют значение лишь при определении ответственности внутри команды, работающей над программным продуктом, поскольку на каждом этапе ответственными за реализацию могут быть разные лица (разработчики, тестировщики, инженеры, обеспечивающие техническую поддержку, архитекторы и т.д.).

В исследовании группы ученых под руководством Р. Чилларедж, опубликованном еще в 1992 году, была предложена концепция ортогональной классификации дефектов (ОКД) программного обеспечения. Теоретики предлагают описывать и анализировать ошибки ПО, ориентируясь на категории «тип дефекта», «фаза происхождения», «серьезность». По мнению исследователей, определение типа дефекта позволяет установить природу ошибки: синтаксическая, логическая, ошибка интерфейса, ошибка данных и д.р. Теоретики не дают подробного описания типов дефектов, но указывают, что они напрямую зависят от фазы реализации проекта по разработке ПО. Поскольку для каждого проекта характерен свой набор стадий, исследователями перечислены основные возможные этапы: проектирование, кодирование, тестирование. Под серьезностью ошибки ученые понимают значение допущенной ошибки на функционирование программного обеспечения: критичная, значительная, незначительная [13]. Подход Р. Чилларедж, И.С. Бхандари, Дж.К. Чаарома, М.Дж. Холлидея, Д.С. Моебусома, Б.К. Рэема и М.Й. Вонгома позволяет

провести всесторонний и структурированный анализ недостатков ПО. Однако такая классификация применима преимущественно для контроля и управления процессом разработки ПО, то есть для работы внутри команды.

Анастасия Хамидулина выделяет следующие виды ошибок.

1. Логические ошибки. Следствием их совершения является некорректное исполнение функций, либо снижение работоспособности программы. Как отмечает теоретик, не все разработчики способны обнаружить ошибки такого вида.
2. Синтаксические ошибки (опечатки) никак не влияют на работоспособность программы. Компилятор обнаруживает их в коде, поэтому разработчику не составляет труда исправить их.
3. Ошибки взаимодействия с аппаратным или программным обеспечением. Устранить ее может только разработчик, переписав участок кода.
4. Ошибки компилирования. Следствие - невозможность создания функционирующей программы по исходному коду.
5. Ошибки среды исполнения возникают когда вычислительная среда (виртуализированная или реальная) не в состоянии продолжить выполнение программы. В данном случае ответственным является разработчик, который не учел особенности среды или поведения программы.
6. Арифметические - ошибки в операциях или константах, использованных разработчиком в коде. [11].

К описанному набору ошибок можно добавить:

1. Граничные ошибки, которые возникают в процессе обработки экстремальных (максимальных или минимальных) значений. Следствием может стать непредвиденное поведение программы или ее аварийное завершение.
2. Ошибки безопасности. Они связаны с наличием уязвимостей и возможным использованием программы третьими лица с целью нанесения ущерба.

Все перечисленные виды ошибок программного обеспечения связаны с действиями лиц, занимающихся созданием программного обеспечения. Вместе с тем существуют и иные причины недостатков ПО.

В частности, Д.В. Грузенкин, А.О. Бондаренко, А.А. Черненко выделяют четыре основные причины дефектов программного обеспечения: ошибки в коде, недопустимые действия пользователя, некорректные входные данные и неисправность техники [6]. Аналогичный подход представлен в книге Лаура Л. Пуллум «Техники и реализация программного контроля наличия ошибок». В дополнение автором определены такие причины как ошибка требований, ошибка проектирования [16]. Думаем, что дефект по причине некорректных входных данных относится к недопустимым действиям пользователя, поскольку ошибка этого вида свойственна только для этапа эксплуатации ПО. В связи с изложенным считаем, что можно выделить три основные причины недостатков программного обеспечения. А именно:

1. Ошибки, совершенные до момента непосредственной разработки продукта - ошибки требований, проектирования. Эти этапы согласовываются с заказчиком, поэтому разработчик не должен нести ответственность.
2. Ошибки, допущенные разработчиком в процессе создания продукта (и документации к нему) - ошибки кодирования и документации. Очевидно, что разработчик несет ответственность за недостатки такого вида.
3. Ошибки, возникшие из-за недопустимых действий пользователя, действий, противоречащих положениям руководства пользования - эксплуатационные ошибки, ответственность за которые несет заказчик.

Причина ошибки может быть определена самим разработчиком, либо независимым аудитором, как это описано в идее С.С. Коротких и К.С. Синюшина о проверке качества государственных цифровых сервисов [9].

Полагаем, как только будет установлено, что дефект образовался по вине разработчика, ему следует присвоить категорию «серьезности»:

1. Блокирующая ошибка (blocking). Такая ошибка позволяет запустить программное обеспечение, однако его использование невозможно до момента ее устранения.
2. Критическая (critical) ошибка означает, что части программы не выполняют заявленные функции.

3. Существенная ошибка (major). Ее наличие сильно затрудняет выполнение программой основных функций.
4. Незначительная ошибка (minor) означает, что некоторые дополнительные функции исполняются некорректно.
5. Тривиальная ошибка (trivial) никак не нарушает функционирование программы, а несет лишь эстетические недостатки [11].

В ситуациях, когда правоотношения сторон продолжаются, сторонам рекомендуется определить приоритет ошибок, если таких несколько.

1. Наивысший приоритет (top). Наличие ошибки, имеющей данный приоритет, может привести к отказу ПО, поэтому ее нужно устранить в первую очередь.
2. Высокий приоритет (high). Дефект такого вида может затруднить работу ПО.
3. Обычный приоритет (normal). Ошибка такого вида не приводит к отказу системы, однако некоторые функции могут выполняться некорректно.
4. Низкий (low). Наличие дефекта такого вида никак не влияет на работу программы.

Заключение

Резюмируя изложенное выше, можно отметить, что недостатки в программном обеспечении могут возникнуть по разным обстоятельствам. В связи с этим решая вопрос об ответственности разработчика, важно установить причину ошибки. В исследовании предложено классифицировать причины ошибок следующим образом: ошибки требований, проектирования; ошибки кодирования и документации; эксплуатационные ошибки. Установлено, что разработчик несет ответственность только за дефекты, возникшие в процессе кодирования и подготовки документации. Поэтому определяя причину ошибки, в первую очередь необходимо установить, на каком этапе проекта она была допущена. В статье сделан вывод о том, что причина может быть определена как самим разработчиком, так и независимым экспертом. Поскольку законом не установлены пределы ответственности разработчика за качество программного обеспечения, на этапе составления договора стороны могут согласовать границы ответственности в зависимости от вида ошибки: блокирующая, критическая, существенная, незначительная, тривиальная ошибка. Помимо этого стороны могут определить сроки для устранения ошибок в зависимости от приоритета (наивысший, высокий, обычный, низкий).

Литература

1. ГОСТ Р 56939-2016. Национальный стандарт Российской Федерации. Защита информации. Разработка безопасного программного обеспечения. Общие требования.
2. ГОСТ Р 27.303-2021. Национальный стандарт Российской Федерации. Надежность в технике. Анализ видов и последствий отказов.
3. IEEE Standard Classification for Software Anomalies. IEEE Computer Society, IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, 2010.
4. Белеванцев А.А. Многоуровневый статический анализ исходного кода для обеспечения качества программ: дис… д-ра. физ-мат. наук. Москва, 2017. 229 с.
5. Герасимов А.Ю. Классификация предупреждений о программных ошибках методом динамического символьного исполнения программ: дис… кан-та. физ-мат. наук. Москва, 2019. 129 с.
6. Грузенкин Д.В., Бондаренко А.О., Черненко А.А. Причины возникновения ошибок в программном обеспечении и восстановление после них // Новая наука: от идеи к результату. 2016. № 12-3. С. 56-59.
7. Дроботун Е.Б. Критичность ошибок в программном обеспечении и анализ их последствий // Фундаментальные исследования. 2009. № 4-S. С. 73-74.
8. Ерахтина О.С., Лабутина К.М. Гражданско-правовая ответственность разработчика за качество программного обеспечения: научные подходы и правоприменительная практика // Информационное общество. 2022. № 5. С. 71-79.
9. Коротких С.С., Синюшин К. С. Грани ответственности: заказчик, разработчик, пользователь // Этика и «Цифра»: от проблем к решениям. 2021. С. 150-156.

10. Нестеренко М.А., Умницын М.Ю. Анализ угроз, уязвимостей, ошибок кода программного обеспечения, 2016. [Электронный ресурс] URL: https://sci-article.ru/stat.php?i=1457559822 (дата обращения 22.04.2023).
11. Хамидулина А. Что такое программные ошибки и как их избежать [Электронный ресурс] URL: https://sky.pro/media/chto-takoe-programmnye-oshibki/ (дата обращения 21.04.2023).
12. Boehm B.W. Software Engineering. IEEE Transactions on Computers. 1976 C-25(12). P.1226-1241.
13. Chillarege R., Bhandari I.S. ect. Orthogonal defect classification-a concept for in-process measurements // IEEE Transactions on Software Engineering. 1992. v. 18. p. 943-956.
14. Peng W.W., Walles D.R. Software Error Analysis. Gaithersburg, 1993. 120 p.
15. Pezze M., Young M. Software Testing and Analysis: Process, Principles and Techniques. John Wiley & Sons. Hoboken, 2008. 510 p.
16. Pullum L.L. Software Fault Tolerance Techniques and Implementation. Artech House Publishers; Illustrated edition. 2001. 360 p.
17. Riggins J. Are Programmers Ethically (and Legally) Responsible for Their Code? 16.08.2018 / URL: https://thenewstack.io/are-programmers-ethically-and-legally-responsible-for-their-code/ (дата обращения 18.12.2022).

ANALYSIS OF THE STATE OF TERMINOLOGY FOR CAUSES AND KINDS OF SOFTWARE ERRORS IN WORKS OF RUSSIAN AND FOREIGN AUTHORS

Erakhtina, Olga Sergeevna

Candidate of legal sciences, associate professor
National Research University Higher School of Economics – Perm, Department of civil and business law, associate professor
Perm, Russian Federation
oerahtina@hse.ru

Popova, Daria Vladislavovna

National Research University Higher School of Economics – Perm, master of legal sciences
Perm, Russian Federation
dvpopova2000@gmail.com

Abstract

The article presents research results concerning the classifications of causes and types of errors in software. These classifications are described in various standards and are reviewed by Russian and foreign specialists. The authors define the limits of the developer's responsibility for the quality of software depending on the cause and type of an error.

Keywords

causes of errors in software; types of errors in software; responsibility of software developer

References

1. GOST R 56939-2016. Naczional`ny`j standart Rossijskoj Federaczii. Zashhita informaczii. Razrabotka bezopasnogo programmnogo obespecheniya. Obshhie trebovaniya.
2. GOST R 27.303-2021. Naczional`ny`j standart Rossijskoj Federaczii. Nadezhnost` v tekhnike. Analiz vidov i posledstvij otkazov.
3. IEEE Standard Classification for Software Anomalies. IEEE Computer Society, IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, 2010.
4. Belevanczev A.A. Mnogourovnevy`j staticheskij analiz iskhodnogo koda dlya obespecheniya kachestva programm: dis… d-ra. fiz-mat. nauk. Moskva, 2017. 229 s.
5. Gerasimov A.Yu. Klassifikacziya preduprezhdenij o programmny`kh oshibkakh metodom dinamicheskogo simvol`nogo ispolneniya programm: dis… kan-ta. fiz-mat. nauk. Moskva, 2019. 129 s.
6. Gruzenkin D.V., Bondarenko A.O., Chernenko A.A. Prichiny` vozniknoveniya oshibok v programmnom obespechenii i vosstanovlenie posle nikh // Novaya nauka: ot idei k rezul`tatu. 2016. № 12-3. S. 56-59.
7. Drobotun E.B. Kritichnost` oshibok v programmnom obespechenii i analiz ikh posledstvij // Fundamental`ny`e issledovaniya. 2009. № 4-S. S. 73-74.
8. Erakhtina O.S., Labutina K.M. Grazhdansko-pravovaya otvetstvennost` razrabotchika za kachestvo programmnogo obespecheniya: nauchny`e podkhody` i pravoprimenitel`naya praktika // Informaczionnoe obshhestvo. 2022. № 5. S. 71-79.
9. Korotkikh S.S., Sinyushin K. S. Grani otvetstvennosti: zakazchik, razrabotchik, pol`zovatel` // E`tika i «Czifra»: ot problem k resheniyam. 2021. S. 150-156.
10. Nesterenko M.A., Umniczy`n M.Yu. Analiz ugroz, uyazvimostej, oshibok koda programmnogo obespecheniya, 2016. [E`lektronny`j resurs] URL: https://sci-article.ru/stat.php?i=1457559822 (data obrashheniya 22.04.2023).
11. Khamidulina A. Chto takoe programmny`e oshibki i kak ikh izbezhat` [E`lektronny`j resurs] URL: https://sky.pro/media/chto-takoe-programmnye-oshibki/ (data obrashheniya 21.04.2023).
12. Boehm B.W. Software Engineering. IEEE Transactions on Computers. 1976 C-25(12). P.1226-1241.

13. Chillarege R., Bhandari I.S. ect. Orthogonal defect classification-a concept for in-process measurements // IEEE Transactions on Software Engineering. 1992. v. 18. p. 943-956.
14. Peng W.W., Walles D.R. Software Error Analysis. Gaithersburg, 1993. 120 p.
15. Pezze M., Young M. Software Testing and Analysis: Process, Principles and Techniques. John Wiley & Sons. Hoboken, 2008. 510 p.
16. Pullum L.L. Software Fault Tolerance Techniques and Implementation. Artech House Publishers; Illustrated edition. 2001. 360 p.
17. Riggins J. Are Programmers Ethically (and Legally) Responsible for Their Code? 16.08.2018 / URL: https://thenewstack.io/are-programmers-ethically-and-legally-responsible-for-their-code/ (дата обращения 18.12.2022).