УДК 004.054
DOI: 10.15827/0236-235X.138.208-214
Дата подачи статьи: 11.11.21, после доработки: 03.03.22
2022. Т. 35. № 2. С. 208–214

Об уточнении принципа организации контроля качества программных продуктов

О.В. Тиханычев [1], *к.т.н., зам. начальника отдела управления перспективных разработок, tow65@yandex.ru*

[1] *«Т1 Интеграция», г. Москва, 111395, Россия*

Предметом данного исследования является процесс разработки ПО АСУ. Объект исследования – система контроля качества этого процесса.

В настоящее время нормативные документы и модели оценки качества ПО построены на основе принципа, определяющего, что качество программ проверятся на соответствие исключительно требованиям технического задания на их разработку. Но, как показала практика, такой подход не отвечает в полной мере современным условиям, обеспечивая не контроль качества, а проверку соответствия программ ожиданиям заказчика, сформулированным еще на начальном этапе разработки. С учетом того, что требования заказчика могут быть сформулированы недостаточно полно и уточняться в ходе работы, сформированные показатели и критерии, определяющие оценку качества, в подобной ситуации не гарантируют обеспечения качества итоговых оценок. Этот тезис является актуальным при использовании как гибких, так и каскадных методов разработки.

Для решения проблемы в статье использованы общенаучные методы анализа и синтеза. На основе анализа существующих подходов к оценке качества разработки ПО синтезированы предложения по уточнению базовых принципов его оценки.

Сформулирована постановка научно-практической задачи и предложен один из подходов к ее решению, основанный на уточнении используемого в настоящее время подхода к оценке качества, перехода от заранее задаваемой жесткой модели к расширенной, оценивающей не только требования технического задания, но и условия их выполнения.

Практическая значимость предлагаемого подхода в том, что его реализация обеспечит общее повышение эффективности автоматизированного управления за счет повышения эффективности и безопасности применения прикладных программ на основе уточнения базового принципа оценки качества, перехода к применению динамической модели оценки качества разрабатываемого ПО.

Ключевые слова: *система управления, поддержка принятия решений, ПО, качество ПО, оценка качества программ, принципы оценки качества*.

Важнейшей составной частью практически любой сложной технической системы, во многом обеспечивающей выполнение функционала этих систем, является ПО.

Одна из главных задач при разработке программных продуктов – контроль их качества, обеспечивающего эффективность и безопасность функционирования систем, которые управляются с применением ПО. Для обеспечения контроля качества создаваемых программ в настоящее время разработано и используется достаточно большое количество методик и средств их реализации, в том числе автоматизированные системы тестирования.

К сожалению, как показывает практика, эффективность данных методик и средств тестирования не всегда обеспечивает безопасность эксплуатации сложных систем [1–3]. Примерами могут служить достаточно большое количество технических происшествий, связанных с некорректным функционированием ПО, например, авария российского разгонного блока «Фрегат» в декабре 2017 года. Как показало расследование, причина случившегося в ошибке ПО, не выявленной на этапе тестирования и проявившейся через два десятилетия эксплуатации [4]. Аналогичным примером является и сбой ПО автоматизированной системы противовоздушной обороны «Си Вулф» английского фрегата «Бродсворд» во время боя с аргентинскими самолетами в мае 1982 года, когда пара самолетов была воспринята как одна цель, а потом зафиксирована еще раз как две раздельные цели [5]. Подобных примеров аварий, обусловленных ошибками ПО, можно привести множество: непоражение обнаруженной иракской ракеты «Скад» 25 февраля 1991 года комплексом Patriot из-за не выявленного ранее накопления ошибки округления внутреннего таймера, сбой электроники истребителей F-22 Raptor в момент пересечения линии перемены дат, отказ компьютерной системы амери-

канского ракетного крейсера «Йорктаун» в 1997 году из-за ошибки деления на ноль, приведшая к катастрофе некорректная работа автоматики системы MCAS самолетов «Боинг» в 2018 году в Индонезии, в 2019 году в Эфиопии и другие. У этих ситуаций различные природа и последствия, но их объединяет причина возникновения – нештатная работа ПО.

При этом все перечисленные системы проходили полный цикл приемо-сдаточных испытаний с использованием разнообразных методологий и средств тестирования, в том числе автоматизированных. Результаты тестирования во всех случаях были положительными, но сбой в процессе функционирования все равно произошел. Значит, причина не в методах и средствах проверки, а в чем-то другом. Анализируя процесс разработки ПО, можно предположить, что проблема кроется в базовом принципе организации тестирования – признании абсолютной достоверности постановки задачи на разработку программных средств управления и использовании ее как единственной модели для тестирования.

Таким образом, анализ практического опыта разработки ПО для сложных технических систем, в том числе личного опыта автора по разработке прикладных программ управления ресурсным обеспечением, показывает, что одной из значимых причин сложившихся ситуаций может быть использование аксиомы, полагающей, что все документы, задающие разработку ПО, являются достаточно полными и абсолютно корректными. Таким образом, проверка ПО на соответствие данным документам является основной целью контроля качества разрабатываемых программ. Но, как показывает практика, эта аксиома не всегда является истинной. Поэтому актуально требование по уточнению подходов к проверке качества ПО, в первую очередь – относительно назначения, цели и выбора задач контроля.

Обзор средств и методологий контроля качества

В современных условиях при организации управления качеством разработки и процессов программной инженерии принято ориентироваться на две основные методологии оценки качества ПО.

Первая из них, TickIT, функционирует в рамках общей системы менеджмента качества программных проектов ISO 9001-00. Вторая методология, CMMI, предоставляет рекомендации по совершенствованию процесса разработки. Непосредственно с управлением качеством связаны предметные области (области компетенции) CMMI:

– обеспечение качества процесса и продукта (категория процессов CMMI Support);

– проверка (категория Engineering);

– аттестация (категория Engineering).

Обе эти методологии не противоречат друг другу, реализующие их стандарты рассматриваются специалистами как взаимодополняющие. Методологии детализируются в нормативных документах (ГОСТ Р ИСО/МЭК 9126-93, ISO 8402:94, IEEE Std 610.12-1990 и других).

В рамках реализации указанных методологий в соответствии с нормативно-технической документацией [6, 7] ПО до внедрения в систему проходит целый ряд приемо-сдаточных испытаний, оценивающих качество разрабатываемых программ. Объем и содержание проверок определяются нормативно-технической документацией, например, ГОСТами 19.301-79, 34.603-92 или Р 15.301-2016.

Для повышения эффективности проводимых проверок используются различные средства и методы тестирования. Но, как показывает анализ, все они рассчитаны на эвристические методы поиска ошибок, а повышение их эффективности предлагается обеспечивать за счет максимального увеличения количества испытаний в разных режимах, в том числе с использованием автоматизированных систем тестирования.

В настоящее время разработано и применяется достаточно много программных средств статического и динамического тестирования, разделяемых по следующим параметрам:

– объект тестирования (функциональное и конфигурационное тестирование, тестирование производительности, интерфейса пользователя, безопасности, локализации, совместимости);

– необходимый уровень знания внутреннего строения системы (тестирование по стратегии черного, белого и серого ящиков);

– степень автоматизации процесса тестирования (ручное, автоматизированное и полуавтоматизированное);

– степень изолированности (тестирование компонентов, интеграционное и системное тестирование);

– этап проведения тестирования (альфа- и бета-тестирование, приемо-сдаточные испытания);

– признак позитивности сценариев (позитивное и негативное тестирование);

– уровень тестирования (тестирование по документации, иногда называемое формальным, и интуитивное тестирование).

Большинство этих методов успешно реализованы в программных системах, эффективность их использования подтверждена практикой [8, 9]. То есть теоретически существующий спектр средств и методов тестирования должен полностью обеспечить проверку качества функционирования программ в любых условиях [10, 11]. Но, как показывают приведенные ранее примеры, этого не происходит.

Обзор содержания применяемых методов, средств и технологий подтверждает тезис о том, что целью всех видов контроля является формальная проверка соответствия разрабатываемого ПО требованиям задающей документации, разрабатываемой еще на этапе формирования заданий на разработку, в первую очередь – *технического задания* (ТЗ) и постановок на разработку задач. При этом не подвергаются сомнению и анализу качество и полнота указанных документов, не ставится задача управления показателями и критериями оценки качества при изменении требований и ожиданий заказчика. Как показала практика, этот подход не всегда корректен, использование его может приводить к ошибкам, в ряде случаев – критичным.

Анализ возможных источников возникновения проблем

В данной статье уже сделано предположение о том, что одним из источников наличия неустраненных ошибок, не позволяющих решить проблему обеспечения качества программ, является существующий базовый принцип его оценки – принятие документов, задающих разработку ПО как аксиому, и проверка исключительно на соответствие требованиям этих документов.

Анализ показывает, что существующие принципы оценки качества основаны на следующих допущениях:

– заказчик уже на начальном этапе разработки ПО точно знает, чего он ожидает от конечного продукта;

– заказчик заранее и точно определил границы применимости системы, допущения и ограничения;

– алгоритмисты и программисты организации-разработчика абсолютно верно и без искажений трактуют требования заказчика.

При принятии этих допущений качество разрабатываемой программы считается однозначно удовлетворительным, если она отвечает требованиям ТЗ. То есть при использовании существующих принципов оценки соответствие требованиям ТЗ считается необходимым и достаточным условием подтверждения качества разработанной программной продукции.

Однако практика показывает, что существует не менее трех потенциальных источников ошибок, подвергающих сомнению эффективность применяемых в настоящее время базовых принципов оценки качества ПО [12].

Первое – это недостаточно полное описание содержания автоматизируемых процессов разработчиками ТЗ и постановок на разработку программ группами аналитиков и экспертов предметной области. Это вполне объяснимо: объем разрабатываемых документов ограничен, поэтому разработчики могут опускать описание процессов и функций, которые они считают типовыми или очевидными. В результате такие функции выпадают из процесса контроля программ, реализующих требования ТЗ. Автор на практике в ходе разработки программно-технических систем управления ресурсным обеспечением неоднократно сталкивался с ситуациями, когда заказчик опускал контроль некоторых параметров управляемых объектов, считая ситуацию очевидной. А программист и тестировщик подходили к процессу серьезно, но формально и не включали в программу и ее проверку контроль таких показателей, хотя система собирала и обрабатывала их в автоматизированном режиме. Проблемы обнаруживались только на этапе апробации программы, когда с ней начинали работать пользователи, в результате требовались доработка и проведение повторных испытаний. Можно, конечно, обвинить разработчиков ТЗ в недостаточно подробном описании исходных данных, но, как уже указывалось, объем ТЗ, хотя и не ограничивается нормативными документами, не бесконечен, его увеличение способствует повышению качества разработки только до определенного предела, далее наступает перенасыщение информацией. Подобная ситуация может возникнуть не только из-за формального отношения к описанию задач экспертом или аналитиком: порядок разработки ТЗ, определяемый существующими руководящими документами, практически не подразумевает управление требованиями заказчика в процессе разработки программ, поэтому нужная информация может быть упущена случайно, а ее уточнение в ходе

разработки ПО возможно, но, как правило, требует повторного согласования задающей документации, что затратно по времени.

Вторая причина аналогична первой, но в данном случае уже тестировщик, составляя методики проверки, может пропустить ряд функций, которые кажутся ему несущественными. Кроме того, возможны варианты неверной интерпретации каких-либо требований при формировании программ и разработке методик проверок. Последствия такой ошибки тоже проявляются не сразу. Результат, как правило, получается такой же, что и в первом случае.

И третье – в соответствии с принципом эмерджентности автоматизация системы управления как любое системное действие может придавать системе автоматизированного управления новые свойства, которых не было ранее и которые, соответственно, не описаны в ТЗ. При существующем подходе к организации проверки качества эти свойства будут просто проигнорированы.

Данные проблемы приводят к снижению качества тестирования, увеличению объема необходимых доработок в процессе апробации программ и опытной эксплуатации, затягиванию процесса разработки.

При использовании гибкой методологии разработки влияние указанных факторов несколько снижается за счет поэтапного уточнения требований заказчика, но, как показывает практика, полностью не компенсируется. При использовании методологии каскадной разработки ситуация становится буквально критической.

Как подтверждает проведенный анализ, указанные проблемы порождают принятые принципы оценки качества, по которым формируется модель его оценки и уточняются ее показатели. Именно инертность и жесткость применяемых принципов снижают эффективность гибких методов разработки ПО и порождают критичные проблемы в каскадных.

Предложения по решению проблемы

Для решения сформулированной проблемы требуется выполнить ряд организационных мер, направленных на уточнение существующих подходов к оценке качества программных продуктов. Учет анализа подходов к оценке их качества, положительных сторон и недостатков позволяет сделать вывод, что данные меры должны заключаться в следующем.

Во-первых, необходимо формально уточнить сам базовый принцип оценки, заменив в нормативно-технической документации заявленный подход к организации контроля качества с существующего жесткого на гибкий, позволяющий уточнять систему показателей и критериев в ходе разработки программ. Выполнение этого положения обеспечит динамическую настройку модели качества ПО в соответствии с гибко уточняемыми требованиями к разрабатываемому продукту, а в итоге – адекватность разрабатываемых программ и методик испытаний актуальным требованиям по состоянию ПО на момент выхода на приемо-сдаточные испытания.

Во-вторых, в ходе уточнения базовых принципов оценки предлагается изменить область проверки качества, расширив ее от формальной проверки требований ТЗ к дополнительному анализу возможности их реализации, то есть проверять как сами требования, так и условия, необходимые для их выполнения, но не описанные в ТЗ в явном виде. Реализация данного положения обеспечит проверку всего диапазона свойств программы, а не только напрямую указанных в ТЗ, как это делается в настоящее время. Такой подход позволит обнаружить ошибки разработки, которые при существующих методах проверки пропускаются из-за отсутствия учета системных связей разрабатываемого программного продукта.

В-третьих, для реализации гибких принципов оценки качества потребуется расширение спектра и соотношения используемых методов тестирования: от проверки формальных признаков отдельных программ, описанных в ТЗ (UNIT-тестирование), к преимущественному тестированию системы по существу (функциональное тестирование), в том числе в процессе закрытия внутренних этапов разработки в ходе реализации гибких методологий. При таком подходе потребуется активное привлечение к проведению проверок на всех этапах разработки аналитиков, а в ряде случаев и экспертов предметной области (конечных пользователей). Его применение обеспечит не только расширение перечня проверяемых как количественных, так и качественных параметров, но и вынесение части объема проверок на этап разработки. Последнее позволит увеличить сам объем проверок без увеличения срока приемосдаточных испытаний, а также обеспечить динамичное совершенствование компонентов программы в ходе ее разработки. Это отно-

сится и к таким важным, но ограниченно тестируемым в настоящее время компонентам, как пользовательский интерфейс программы [13].

Следует отметить, что возможное увеличение объема работ на этапе проверки дает возможность сэкономить время на ввод системы в строй и уменьшить объем последующих доработок. Учитывая многовариантность процесса управления требованиями и накопление ошибок по мере их выполнения, можно говорить о том, что чем раньше выявляются проблемы, тем проще их устранить. То есть увеличение объема работ на более ранних этапах потенциально сократит его на последующих.

Реализация предложенных изменений обеспечит контроль качества разрабатываемых программ на всем диапазоне реализуемого функционала, а не только на определяемом ТЗ, что позволит решить важную научно-практическую задачу повышения безопасности разрабатываемого ПО за счет раннего выявления ошибок разработки. Решение данной задачи особенно актуально в условиях все более активного использования ПО во всех сферах деятельности – в настоящее время оно является составной частью практически любых сложных систем.

На практике выполнение предлагаемых мер позволит сформировать расширенные принципы контроля качества, реализующие процесс тестирования (см. таблицу).

Предлагаемые изменения необходимо зафиксировать в нормативно-технической документации, задающей объем и порядок работ при разработке ПО, доработать описанный в ней методический аппарат испытаний.

Изменение базовых принципов организации контроля качества программ может принципиально изменить подход к его оценке и в определенных условиях обеспечит существенный прирост полноты и достоверности результатов проверок. Это происходит в том числе и потому, что в рамках нового подхода предлагается не просто корректировать цель тестирования по мере уточнения требований к программе: новый подход подразумевает расширение спектра тестирования, не ограничиваясь проверкой требований ТЗ. В результате формируется своеобразная версия гибких методологий разработки, но гибкими являются не требования заказчика, а модель контроля качества, формируемая в более широком диапазоне. В подобной постановке уточнение модели системы контроля качества прикладных программ в настоящее время нигде не выполняется.

Доработки по уточнению существующих принципов оценки качества программ

Improvements to clarify the existing principles for assessing program quality

Процесс тестирования	**Описание в существующей нормативно-технической документации**	**Возможные доработки**
Проверка выполнения требований ТЗ	Описано в полном объеме	Не требуются
Выявление и контроль выполнимости дополнительных условий, необходимых для выполнения требований ТЗ	Не описаны	Структурный анализ, расширение на его основе перечня проверяемых параметров
Выявление синергетических эффектов и проверка их безопасности	Не описаны	Функциональный анализ, уточнение на его основе перечня проверяемых параметров и критериев проверки

Заключение

Анализ возникновения критичных ошибок функционирования ПО показал, что, хотя сами ошибки порождаются некачественной постановкой задачи и низкой квалификацией разработчиков, существенной причиной возникновения проблемы является качество тестирования, не позволяющее обнаружить и устранить ошибки разработчиков. В результате появляются ошибки эксплуатации ПО, порождаемые в том числе использованием существующих принципов оценки качества программной продукции, выдвигающих как необходимое и достаточное условие выполнение требований ТЗ.

Предложенное в статье изменение базовых принципов оценки качества, при котором выполнение требований ТЗ является только необходимым условием, а достаточные условия дополняются потребностями в реализации этих требований и безопасности системных проявлений, может обеспечить существенное повы-

шение вероятности выявления ошибок на всех этапах эксплуатации программных и технических систем, управляемых с использованием ПО. Отказ от использования существующих принципов оценки позволит выстроить динамичный процесс контроля качества, подобный реализуемому в настоящее время для управления требованиями заказчика в гибкой методологии, и обеспечить управление качеством разрабатываемой программной продукции по тем же алгоритмам.

Следует отметить, что предложенное изменение базового принципа контроля качества – это не прямое внедрение гибких подходов к управлению требованиями заказчика в буквальном понимании, когда под меняющиеся требования с некоторой периодичностью подстраивается модель качества, а более широкий подход к оценке, когда в ходе тестирования проверяются не только выраженные в явном виде требования, но и условия, необходимые для их выполнения.

Литература

1. Тиханычев О.В. О показателях качества программного обеспечения автоматизированных систем управления // Программные системы и вычислительные методы. 2020. № 2. С. 22–36. DOI: 10.7256/2454-0714.2020.2.28814.

2. Wagner S., Goeb A., Heinemann L. et al. Operationalised product quality models and assessment: The Quamoco approach. Information and Software Technology, 2015, no. 62, pp. 101–123. DOI: 10.1016/j.infsof.2015.02.009.

3. Krill P. Software testers balk at ISO 29119 standards proposal. InfoWorld Tech Watch, 2014. URL: https://www.infoworld.com/article/2608932/software-testers-balk-at-iso-29119-standards-proposal.html (дата обращения: 27.10.2021).

4. Tikhanychev O.V. On improving indicators for assessing the decision support systems' software quality. IOP Conf. Ser.: Mater. Sci. Eng., 2020, vol. 919, art. 052009. DOI: 10.1088/1757-899x/919/5/052009.

5. Danglot B., Vera-Perez O., Yu Z., Zaidman A., Monperrus M., Baudry B. A snowballing literature study on test amplification. J. of Systems and Software, 2019, vol. 157, art. 110398. DOI: 10.1016/j.jss.2019.110398.

6. Lewis W.E., Dobbs D., Veerapillai G. Software Testing and Continuous Quality Improvement. Boca Raton, CRC Press Publ., 2017, 688 p. DOI: 10.1201/9781439834367.

7. Тиханычев О.В., Макарцев Л.В., Гахов В.Р. Рациональная организация процесса разработки прикладного программного обеспечения как предпосылка успешной автоматизации поддержки принятия решений // Программные продукты и системы. 2017. Т. 36. № 4. С. 706–710. DOI: 10.15827/0236-235X.120.706-710.

8. Струбалин П.В., Фатьянова А.А. Управление качеством программного обеспечения // Вестн. СГСЭУ. 2019. № 2. С. 108–111.

9. Ransome J., Misra A. Core Software Security: Security at the Source. Boca Raton, CRC Press Publ., 2014, 351 p.

10. Никулина И.Е., Николаенко В.С. Становление и развитие концепций управления проектами и риск-менеджмента // Государственное управление. 2018. № 68. URL: https://portal.tpu.ru/SHARED/n/NIKOLAENKOVS/Scientific_work/Tab/nikulina_nikolaenko%202.pdf (дата обращения: 27.10.2021).

11. Бугорский В.Н., Голоскоков К.П. Управление качеством в процессе испытаний средств электронной техники // Прикладная информатика. 2011. № 1. С. 50–60.

12. Майерс Г., Баджетт Т., Сандлер К. Искусство тестирования программ. М.: Диалектика, 2019. 272 с.

13. Wiegers K. Creating a Software Engineering Culture. NY, Dorset House Publ., 2013, 359 p.

Software & Systems
DOI: 10.15827/0236-235X.138.208-214

Received 11.11.21, Revised 03.03.22
2022, vol. 35, no. 2, pp. 208–214

On the clarification of the principle of organizing software products quality control

O.V. Tikhanychev [1], *Ph.D. (Engineering), Deputy Head of the Department of Advanced Development, tow65@yandex.ru*

[1] "*T1 Integration Group*", *Moscow, 111395, Russian Federation*

Abstract. The subject of the research is the process of developing software for automated control systems. The object of research is the quality control system of this process. Currently, regulatory documents and models

for assessing the quality of software are built on the basis of a paradigm that determines that the quality of programs is checked for compliance with the terms of reference for development. But, as practice has shown, such a paradigm does not fully correspond to modern development conditions, providing not quality control, but verification of the compliance of programs with the customer's expectations formulated at the initial stage of development. Taking into account the fact that the customer's requirements may not be fully formulated, and may also be refined in the course of work, the list of indicators and criteria that determine the quality assessment model formed at the beginning of the work may not ensure the quality of control. This thesis is relevant both when using "agile" and "waterfall" development methods.

To solve the problem, the article uses general scientific methods of analysis and synthesis. Based on the analysis of existing approaches to assessing the quality of software development, proposals have been synthesized to refine the paradigm of its assessment. The article formulates the formulation of a scientific and practical problem and proposes one of the approaches to its solution, based on the refinement of the currently used quality assessment paradigm, the transition from a "rigid", predetermined model, to a refined one in the course of work. The solution of the formulated problem will provide a general increase in the efficiency of automated control by clarifying the paradigm for quality assessment, transition to the use of a dynamic model for assessing the software being developed.

Keywords: automated control system, decision support, software, software quality, program quality assessment, quality assessment paradigm.

References

1. Tikhanychev O.V. On the quality indicators of automated control systems software. *Software Systems and Computational Methods*, 2020, no. 2, pp. 22–36. DOI: 10.7256/2454-0714.2020.2.28814 (in Russ.).

2. Wagner S., Goeb A., Heinemann L. et al. Operationalised product quality models and assessment: The Quamoco approach. *Information and Software Technology*, 2015, no. 62, pp. 101–123. DOI: 10.1016/j.infsof.2015.02.009.

3. Krill P. Software testers balk at ISO 29119 standards proposal. *InfoWorld Tech Watch*, 2014. Available at: https://www.infoworld.com/article/2608932/software-testers-balk-at-iso-29119-standards-proposal.html (accessed October 27, 2021).

4. Tikhanychev O.V. On improving indicators for assessing the decision support systems' software quality. *IOP Conf. Ser.: Mater. Sci. Eng*., 2020, vol. 919, art. 052009. DOI: 10.1088/1757-899x/919/5/052009.

5. Danglot B., Vera-Perez O., Yu Z., Zaidman A., Monperrus M., Baudry B. A snowballing literature study on test amplification. *J. of Systems and Software*, 2019, vol. 157, art. 110398. DOI: 10.1016/j.jss.2019.110398.

6. Lewis W.E., Dobbs D., Veerapillai G. *Software Testing and Continuous Quality Improvement*. Boca Raton, CRC Press Publ., 2017, 688 p. DOI: 10.1201/9781439834367.

7. Tikhanychev O.V., Makartsev L.V., Gakhov V.R. Rational organization of an application software development process for decision support successful automation. *Software and Systems*, 2017, vol. 36, no. 4, pp. 706–710. DOI: 10.15827/0236-235X.120.706-710 (in Russ.).

8. Strubalin P.V., Fatyanova A.A. Quality management for software. *Bull. of SGSEU*, 2019, no. 2, pp. 108–111 (in Russ.).

9. Ransome J., Misra A. *Core Software Security: Security at the Source*. Boca Raton, CRC Press Publ., 2014, 351 p.

10. Nikulina I.E., Nikolaenko V.S. Formation and development of project management and risk management concepts. *Public Administration*, 2018, no. 68. Available at: https://portal.tpu.ru/SHARED/n/NIKOLAENKOVS/Scientific_work/Tab/nikulina_nikolaenko%202.pdf (accessed October 27, 2021) (in Russ.).

11. Bugorsky V.N., Goloskokov K.P. Quality management when testing electronic equipment. *J. of Applied Informatics*, 2011, no. 1, pp. 50–60 (in Russ.).

12. Myers G., Badgett T., Sandler C. *The Art of Software Testing*. Word Association Publ., 2012, 240 p. DOI: 10.1002/9781119202486. (Russ. ed.: Moscow, 2019, 272 p.).

13. Wiegers K. *Creating a Software Engineering Culture*. NY, Dorset House Publ., 2013, 359 p.

Для цитирования

Тиханычев О.В. Об уточнении принципа организации контроля качества программных продуктов // Программные продукты и системы. 2022. Т. 35. № 2. С. 208–214. DOI: 10.15827/0236-235X.138.208-214.

For citation

Tikhanychev O.V. On the clarification of the principle of organizing software products quality control. *Software & Systems*, 2022, vol. 35, no. 2, pp. 208–214 (in Russ.). DOI: 10.15827/0236-235X.138.208-214.