Организация автоматизированного контроля качества в жизненном цикле программных средств критически важных систем

Марковский А. С., Самонов А. В., Бурова И. О.
Военно-космическая академия им. А. Ф. Можайского
Санкт-Петербург, Россия
lexx26@list.ru, a.samonov@mail.ru, burova@mamaev.spb.ru

Аннотация. **В статье представлены результаты анализа состояния и перспектив в области разработки программных средств (ПС) критически важных систем. Установлено, что основными направлениями решения накопившихся в данной области проблем являются совершенствование нормативно-методической базы, внедрение современных технологий разработки и реализация сквозного контроля качества ПС на всех этапах их жизненного цикла (ЖЦ). Предложена соответствующая современным технологиям разработки программного обеспечения усовершенствованная модель ЖЦ ПС, в которой точно определены место и роль процессов контроля и обеспечения качества. Описана классификация показателей качества ПС. Представлен метод и технологии реализации автоматизированного сквозного контроля качества на всех этапах жизненного цикла ПС.**

Ключевые слова: **верификация, жизненный цикл, контроль качества, критически важные системы, модель жизненного цикла, показатели качества программных средств, программные средства.**

Введение

На сегодня активное и повсеместное внедрение автоматизированных систем, важнейшим элементом которых являются программные средства (ПС), в том числе в критически важных системах (КВС), обусловило высокий уровень требований к надежности, производительности, защищенности и другим характеристикам ПС. В то же время, как показывает практика в нашей стране и за рубежом, чем сложнее и объемнее программное обеспечение (ПО), тем больше в нем дефектов.

В качестве подтверждения данного тезиса можно привести результаты исследований исходных кодов открытого и проприетарного ПО, выполненных компанией Coverity [1]. В ходе исследований была проведена проверка на наличие ошибок и уязвимостей более 300 миллионов строк кода 41 проприетарного программного продукта и 37 миллионов строк кода 45 программных продуктов с открытым исходным кодом. Установлено, что в среднем на 1000 строк кода приходится 1 ошибка. С учетом того, что средние размеры проектов с открытым исходным кодом 800 тысяч, а проприетарного ПС – 7,5 миллионов строк кода, получается, что в реальных проектах содержатся сотни ошибок, на выявление и устранение которых расходуются значительные интеллектуальные, временны́е и финансовые ресурсы.

Известные объективные и субъективные трудности создания высококачественных ПС в нашей стране усугублены следующими факторами (обстоятельствами):

• современные ПС разрабатываются на основе устаревших неэффективных технологий и инструментальных средств;

• отечественная нормативно-методическая база в области разработки ПС не соответствует современному уровню информационно-коммуникационных технологий и средств разработки;

• на предприятиях, разрабатывающих ПО для КВС, не обеспечивается требуемый уровень контроля качества ПС.

В связи с изложенным становится очевидной необходимость совершенствовать методологическое, нормативно-организационное и технологическое обеспечение процессов разработки, эксплуатации и сопровождения ПС, используемых в КВС.

Базовые модели ЖЦ ПС

Вопросы развития и совершенствования технологий и средств проектирования, разработки, эксплуатации и сопровождения ПО исследуются в рамках системной программной инженерии (СиПИ). Одним из основных понятий в СиПИ является понятие жизненного цикла (ЖЦ) ПС. В общем случае ЖЦ определяется моделью и описывается в форме методологии или метода. Модель ЖЦ определяет концептуальный взгляд на организацию ЖЦ, его основные фазы, условия и порядок перехода между ними. Методология ЖЦ задает комплекс работ, их детальное содержание и ролевую ответственность специалистов на всех этапах выбранной модели ЖЦ, обычно определяет и саму модель, а также предлагает практические рекомендации, позволяющие максимально эффективно воспользоваться соответствующей методологией.

Таким образом, ЖЦ ПС – это непрерывный процесс, описывающий все, что происходит с ПС с момента принятия решения о необходимости создания ПС до его изъятия из эксплуатации. Модель ЖЦ ПС – это определенным образом упорядоченная совокупность этапов ЖЦ (видов деятельности и событий), условия и порядок переходов между ними, обеспечивающих достижение цели проекта в установленные сроки в рамках доступного бюджета времени, людских и финансовых средств [2, 3].

Основными понятиями, в терминах которых описывается ЖЦ ПС, являются: виды деятельности, роли (исполнители) и артефакты (результаты).

Виды деятельности – это набор действий, направленных на решение одной задачи или группы тесно связанных задач в рамках разработки и сопровождения ПС. Например, анализ предметной области, описание требований, проектирование, разработка кода, тестирование, управление конфигурациями, развертывание и т. д.

Исполнители – это профессиональная специализация людей, участвующих в создании или сопровождении ПС и имеющих одинаковые интересы или решающих одни и те же задачи по отношению к этому ПС.

Артефакты – различные информационные сущности, документы, модели, программы, создаваемые или используемые в ходе разработки и сопровождения ПС. Например, техническое задание, описание архитектуры, модель предметной области, исходный код, результаты испытаний и т. д.

В настоящее время создание ПС реализуется в рамках одной из трех базовых моделей ЖЦ: каскадной, инкрементной или спиральной, результаты анализа которых приведены в табл. 1. Выбор той или иной стратегии определяется характеристиками проекта, требованиями к конечному продукту, командой разработчиков и конечными пользователями.

Таким образом, использование каскадной стратегии наиболее эффективно в следующих случаях:

1) при разработке проектов с четкими, неизменяемыми в течение ЖЦ требованиями и с понятной реализацией;

2) при разработке проектов невысокой сложности;

3) при выполнении больших проектов в качестве составной части моделей ЖЦ, реализующих другие стратегии разработки.

Использование инкрементной модели наиболее эффективно в следующих случаях:

1) при разработке проектов, в которых большинство требований можно сформулировать заранее, но часть из них можно уточнить через определенный период времени;

2) при разработке сложных проектов с заранее сформулированными требованиями;

3) при необходимости максимально быстро поставить потребителю продукт, имеющий определенные базовые функциональные свойства;

4) при разработке проектов с низкой или средней степенью рисков;

5) при выполнении проекта с применением новых технологий.

Использование эволюционной стратегии наиболее эффективно в следующих случаях:

1) при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или требуют уточнения;

2) при разработке сложных проектов, в том числе тех, которые используют новые технологии.

Анализ нормативно-методических документов, регламентирующих разработку ПС КВС в части состава, структуры и содержания этапов создания и эксплуатации ПС [4–9], позволил определить следующие недостатки и проблемные вопросы:

1) состав, структура и содержание этапов создания ПС ориентированы на реализацию каскадной модели ЖЦ ПС,

Таблица 1

Достоинства и недостатки моделей ЖЦ ПС

Достоинства	Недостатки
Каскадная	
1) Стабильность требований в процессе разработки; 2) необходимость пройти только один этап разработки, что обеспечивает простоту применения стратегии; 3) простота планирования, контроля и управления проектом; 4) доступность для понимания заказчиками	1) Сложность полного формулирования требований в начале процесса разработки и невозможность их динамического изменения на протяжении ЖЦ; 2) линейность процесса разработки, разрабатываемые ПС или система обычно слишком велики и сложны; из-за возврата к предыдущим шагам для решения возникающих проблем увеличиваются финансовые затраты и нарушается график работ; 3) непригодность промежуточных продуктов для использования; 4) недостаточное участие пользователя в процессе разработки ПС – только при разработке требований и во время приемочных испытаний – не позволяет пользователю предварительно оценить качество ПС или системы
Инкрементная	
1) Возможность получить функциональный продукт после реализации каждого инкремента; 2) быстрое создание инкремента; это сокращает сроки начальной поставки, позволяет снизить затраты на первоначальную и последующие поставки программного продукта; 3) предотвращение реализации громоздких спецификаций требований; стабильность требований во время создания определенного инкремента; возможность учитывать изменившиеся требования; 4) снижение рисков по сравнению с каскадной стратегией; 5) включение в процесс пользователей, что позволяет оценить функциональные возможности продукта на более ранних этапах разработки и в итоге приводит к повышению качества программного продукта, снижению затрат и времени на его разработку	1) Необходимость полного функционального определения ПС или системы в начале жизненного цикла для обеспечения планирования инкрементов и управления проектом; 2) возможность текущего изменения требований к ПС или к системе, которые уже реализованы в предыдущих инкрементах; 3) сложность планирования и распределения работ; 4) проявление человеческого фактора, связанного с тенденцией к оттягиванию решения трудных проблем на поздние инкременты, может нарушить график работ или снизить качество программного продукта
Спиральная	
1) Возможность уточнения и внесения новых требований в процессе разработки; 2) пригодность промежуточного продукта для использования; 3) возможность управления рисками; 4) обеспечение широкого участия пользователя в проекте начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества; 5) реализация преимуществ каскадной и инкрементной стратегий	1) Неизвестность точного количества необходимых итераций и сложность определения критериев для продолжения разработки на следующей итерации; это может вызвать задержку реализации конечной версии системы или ПС; 2) сложность планирования и управления проектом; 3) необходимость активного участия пользователей в проекте, что не всегда осуществимо; 4) необходимость в мощных инструментальных средствах и методах прототипирования; 5) возможность отодвинуть решение трудных задач на последующие циклы может привести к несоответствию полученных продуктов требованиям заказчиков

в то время как сейчас стандартом де-факто при разработке сложных ПС является спиральная модель ЖЦ;

2) противоречия в используемой терминологии и в видах деятельности, выполняемых на одних и тех же этапах ЖЦ ПС;

3) избыточность и излишняя детализация отдельных видов деятельности и артефактов;

4) не используются преимущества применения концепции процессного подхода к разработке и применению ПС;

5) недостаточно полно и конкретно определены место и роль процессов управления качеством ПС.

В табл. 2 представлена модель ЖЦ ПС КВС, в которой устранены названные недостатки. Структура данной модели описывается с помощью трех основных понятий: вид деятельности, исполнители и артефакты.

Таблица 2

Усовершенствованная модель ЖЦ ПС КВС

№	Этапы, виды деятельности (процессы)	Исполнители	Артефакты (содержание и форма)
1	Анализ потребности, разработка концепции и обоснование требований		
1.1	Анализ потребности и обоснование необходимости ПС	Аналитики Заказчики Потребители Организации научного и технического сопровождения (НС, ТС)	Обоснование места и роли ПС в составе КВС. Аналитический отчет
1.2	Разработка концепции построения, функционирования и применения ПС	Аналитики Заказчики Потребители Организации НС, ТС	Концепция применения ПС в составе КВС
1.3	Разработка технического задания (ТЗ) на ПС		
1.3.1	Разработка проекта ТЗ на ПС	Аналитики Заказчики Потребители	Проект ТЗ на ПС (функциональные и эксплуатационные требования)
1.3.2	*Верификация требований к ПС*	*Заказчик Организации НС, ТС*	*Заключение о полноте, непротиворечивости и корректности требований к ПС*
1.3.3	Согласование и утверждение ТЗ	Заказчик Потребители Организации НС, ТС	Утвержденное ТЗ
2	Разработка ПС		
2.1	Планирование разработки ПС	Руководитель проекта Проектировщик	Сквозной график разработки ПС
2.2	Проектирование ПС		
2.2.1	Разработка проекта (архитектуры) ПС	Проектировщик (конструктор)	Проект архитектуры и компонентов ПС
2.2.2	*Верификация проекта (архитектуры) ПС*	*Организации НС, ТС*	*Заключение о соответствии проекта заданным в ТЗ условиям*
2.3	Разработка программ и программной документации		

Продолжение табл. 2

№	Этапы, виды деятельности (процессы)	Исполнители	Артефакты (содержание и форма)
2.3.1	Разработка программ в соответствии с проектом ПС	Руководитель проекта Разработчики Технические писатели	Программный код и программная документация
2.3.2	*Тестирование на контрольных примерах (модульное, функциональное комплексное, нагрузочное)*	*Тестировщики*	*Протоколы и акты испытаний. Предложения по доработке*
2.3.3	Возврат к этапам 2.2.1 или 2.3.1 на доработку или завершение разработки, включая подготовку рабочей конструкторской, программной и эксплуатационной документации (РКД, ПД и ЭД)	Руководитель проекта Проектировщик Разработчики Технические писатели	ПС, РКД, ПД, ЭД, задание на доработку ПС
2.3.4	*Верификация РКД, ПД и ЭД*	*Отдел Технического Контроля (ОТК) предприятия Организации НС, ТС*	*Акты соответствия РКД, ПД и ЭД заданным в ТЗ условиям*
2.4	Проведение испытаний ПС		
2.4.1	*Предварительные, приемочные, государственные испытания ПС*	*Разработчики Организации НС, ТС Потребители Заказчики*	*Протоколы и акты испытаний*
2.4.2	*Верификация результатов испытаний*	*Разработчики Организации НС, ТС*	*Экспертное заключение*
3	Производство		
3.1	Постановка ПС на производство	Разработчики предприятия-изготовителя	Копии РКД, ПД и ЭД Технологическая линия производства ПС
3.2	Изготовление ПС	Разработчики предприятия-изготовителя	ПС с комплектом документации
3.3	*Контроль и приемка ПС*	*ОТК Потребитель*	*Протоколы и акты приемки ПС*
3.4	Поставка ПС потребителю	Разработчики предприятия-изготовителя	Акт о передаче ПС потребителю
4	Применение в составе системы		
4.1	Эксплуатация		
4.1.1	Опытная (экспериментальная) эксплуатация	Потребитель Разработчики	Отчет с результатами опытной эксплуатации
4.1.2	Функционирование ПС в составе системы	Потребитель	Результаты применения по назначению (расчеты, информационные документы и т. д.)
4.2	Сопровождение		
4.2.1	Организация сопровождения ПС	Разработчики предприятия-изготовителя	Договоры и акты о сопровождении ПС

Окончание табл. 2

№	Этапы, виды деятельности (процессы)	Исполнители	Артефакты (содержание и форма)
4.2.2	*Анализ функционирования ПС*	*Потребитель Разработчики предприятия-изготовителя*	*Замечания и рекламации. Предложения о доработке и модернизации*
4.2.3	Модернизация ПС	Разработчики	Модернизированное ПС
4.2.4	*Тестирование и верификация модернизированного ПС*	*Разработчики ОТК Потребитель*	*Протоколы и акты испытаний*
4.3	Прекращение эксплуатации		
4.3.1	Подготовка к снятию ПС с эксплуатации	Потребитель Разработчики	Обоснование целесообразности снятия ПС с эксплуатации
4.3.2	Прекращение эксплуатации ПС	Потребитель Заказчик	Документ о снятии ПС с эксплуатации

Важной особенностью представленной в табл. 2 модели ЖЦ является наличие точного указания места и роли процессов контроля качества ПС КВС, которые в таблице выделены курсивом. Результатом выполнения данных процессов является оценка степени соответствия артефактов, получаемых на каждом этапе, заданным или ожидаемым требованиям к создаваемому ПС.

Контроль качества ПС КВС

Качество ПО – это совокупность существенных свойств (характеристик) программного обеспечения, обусловливающих его пригодность для использования по назначению. Как видно из табл. 2, процессы верификации должны осуществляться по отношению ко всем значимым артефактам. Первым из них является ТЗ, в котором определяются требования к функциональным и эксплуатационным характеристикам ПС. Наиболее полная классификация характеристик ПС представлена в стандартах серии ISO/IEC 25000 [10]. В соответствии с этой классификацией все характеристики ПО сведены в 8 групп (рис. 1).

Каждая группа характеристик состоит из подхарактеристик, или атрибутов. Например, такая характеристика ПС, как «надежность», состоит из четырех подхарактеристик: отказоустойчивости, восстанавливаемости, завершенности и доступности [10]. Для оценивания степени, с которой определенная характеристика соответствует установленным требованиям, используются показатели качества. Показатель качества – это переменная или несколько переменных, значение которых характеризует меру качества ПО относительно одного или нескольких существенных свойств. Так, в качестве меры отказоустойчивости системы могут использоваться следующие показатели:

- вероятность безотказной работы $P(t)$;
- средняя наработка на отказ Tо;
- гамма-процентная наработка до отказа $T\gamma$;
- интенсивность отказов $\lambda(t)$;
- параметр потока отказов $\omega(t)$;
- средняя доля безотказной наработки $I(t)$;
- плотность распределения времени безотказной работы $f(t)$.

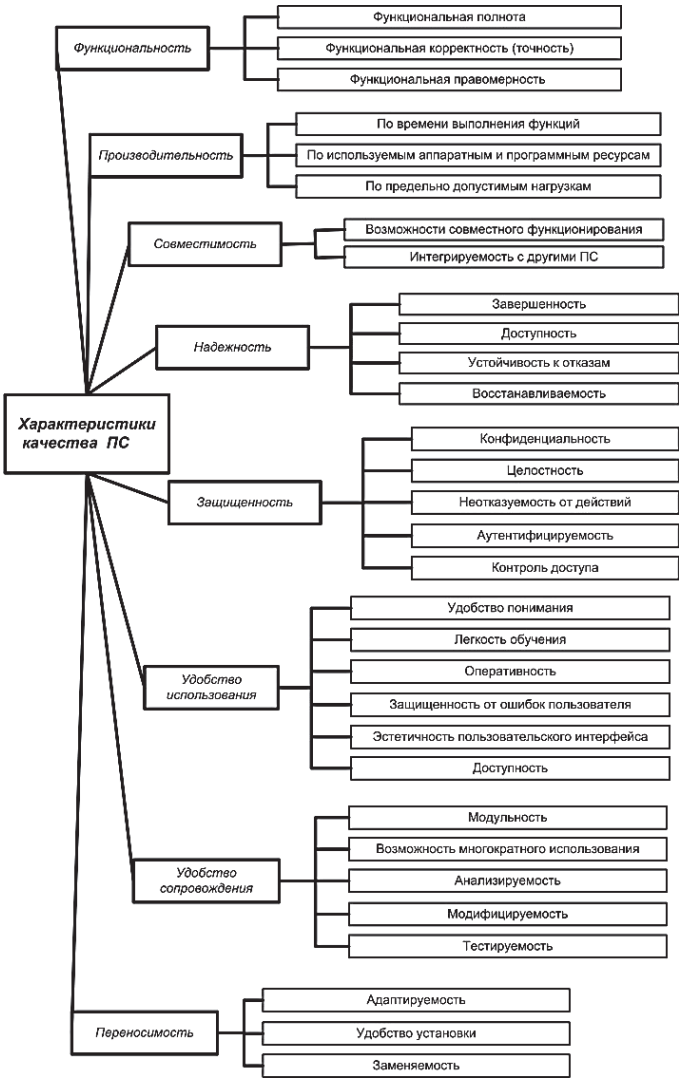


Рис. 1. Классификация характеристик качества ПС

Показатели качества можно спроецировать на основные группы их потребителей: разработчиков, заказчиков и конечных пользователей. В соответствии с этим принципом все показатели распределяются по трем группам [2, 4, 10]:

Показатель внутреннего качества – степень, с которой множество статических свойств ПС удовлетворяет заявленным или подразумеваемым требованиям к ПС при использовании в заданных условиях. Статические свойства имеют отношение к архитектуре, внутренней организации и корректности (безошибочности) кода ПС. Примерами показателей внутреннего качества являются количество ошибок спецификации, проектирования и кодирования, избыточность кода, отсутствие контроля вводимых данных и др. Эти показатели используют программисты и тестировщики ПС в ходе кодирования, отладки, тестирования.

Показатель внешнего качества – степень, с которой ПС позволяет функционированию системы удовлетворять предъявленным к ней требованиям при использовании в заданных условиях. Примерами таких показателей являются вероятность безотказной работы в течение определенного периода времени, время решения расчетных задач, пропускная способность каналов передачи данных, время восстановления работоспособности после сбоев и другие. Используется во

время различных испытаний ПС представителями заказчика, органов сертификации и конечными пользователями.

Показатель качества при использовании – степень, с которой ПС пригодно к использованию по назначению определенными пользователями в заданных условиях применения. Используются, прежде всего, конечными пользователями на этапах эксплуатации ПС.

В табл. 3 представлены методы верификации артефактов, получаемых в ходе разработки ПС КВС.

Таблица 3

Артефакты ЖЦ ПС КВС и методы их верификации

Артефакт	Метод верификации	Исполнитель
ТЗ на разработку ПС	Экспертиза (общая и техническая)	Заказчики Потребители Организации НС, ТС
Архитектура ПС (проект)	Экспертиза техническая	Аналитики Проектировщики Организации НС, ТС
Программный код ПС	1) Статический анализ исходного кода; 2) динамический анализ выполнения программы; 3) регрессионное тестирование; 4) тестирование на стенде	Программисты Тестировщики
РКД, ПД и ЭД ПС	Экспертиза (общая и техническая)	ОТК ПЗ Организации НС, ТС
Результаты испытаний	1) Экспертиза (общая и техническая); 2) инспекция	Заказчики ПЗ Организации НС, ТС
Программный продукт (на этапе производства и приемки)	1) Экспертиза (общая и техническая); 2) тестирование на стенде	Разработчики ОТК ПЗ Организации НС, ТС

Как показал анализ, основными методами верификации являются экспертиза требований, проектных решений и программного кода, статические и динамические методы тестирования ПС, аудит, регистрация и анализ результатов эксплуатации.

Наряду с совершенствованием нормативно-методической базы важным направлением совершенствования процессов управления качеством ПС является использование для верификации и валидации артефактов ЖЦ ПС КВС формальных процедур и поддерживающих их инструментальных средств. Для создания таких средств необходимо разработать формальную модель и язык описания всех артефактов. Схема алгоритма верификации представлена на рис. 2.

В настоящее время наиболее подходящими для этого средствами являются унифицированный язык моделирования UML (UnifiedModellingLanguage) [11], язык описания метамоделей MOF (MetaObjectFacility) [12] и протокол обмена метаданными XMI (XML MetadataInterchange) [13].

В результате совместного использования этих средств можно построить абстрактную метамодель ЖЦ ПС КВС, разработать средства управления и обмена данными (моделями) с целью их трансформации в поддерживаемые технологии программирования и реализации автоматизированной

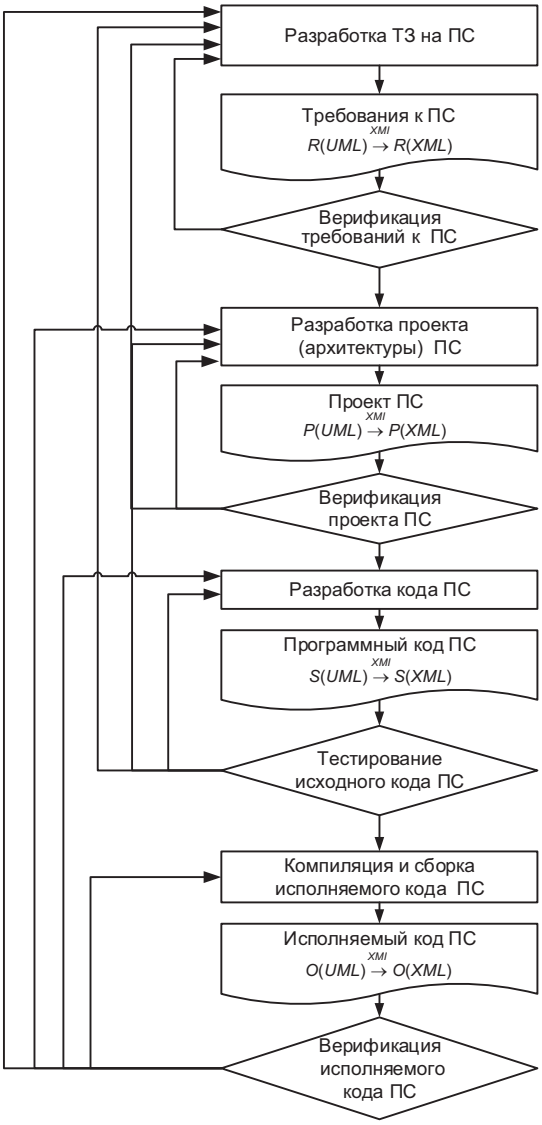


Рис. 2. Алгоритм контроля качества ПС КВС

верификации. Ярким примером таких средств является линейка продуктов MATLAB: Simulink, SystemTest, Simulink Design Verifier, Simulink Verification and Validation [14].

Заключение

Многочисленные исследования в области оценивания стоимости разработки ПО, в том числе стоимости обнаружения и исправления ошибок, показали, что удельная стоимость исправления дефектов возрастает по экспоненциальному закону распределения по мере продвижения продукта к стадии эксплуатации [15–19]. Представленный на рис. 3 график показывает [20, 21], что затраты на исправление дефекта на этапе тестирования в четыре раза превышают затраты на его устранение на этапе проектирования и в 20 раз, если бы он был обнаружен и устранен на этапе обоснования требований. Стоимость исправления дефектов на этапе эксплуатации превышает затраты на этапах проектирования и тестирования в десятки раз.

В связи с этим становится очевидным, что применение данного подхода на практике позволит не только повысить



Рис. 3. Алгоритм контроля качества ПС КВС

качество разрабатываемого ПО для КВС, но и существенно снизить затраты на поиск и устранение дефектов в нем.

Литература

1. http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf (дата обращения 05.07.2015).

2. Кулямин В. В. Методы верификации программного обеспечения / В. В. Кулямин. – М. : Ин-т системного программирования РАН, 2008. – 111 с.

3. Генельт А. Е. Управление качеством разработки программного обеспечения : учеб.-методич. пособие / А. Е. Генельт. – СПб. : ИТМО, 2007. – 187 с.

4. ISO/IEC 9126 Software engineering – Product quality. – Part 1–41, 2001.

5. ГОСТ Р 51189-98. Порядок разработки программных средств систем вооружения (введ. 01.07.1999). – М. : Стандартинформ, 2010. – 16 с.

6. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания (введ. 01.01.1992). – М. : Стандартинформ, 2009. – 6 с.

7. ГОСТ Р ИСО/МЭК 12207-2010. Процессы жизненного цикла программных средств (введ. 01.03.2012). – М. : Стандартинформ, 2011. – 105 с.

8. ГОСТ 28195-89. Оценка качества программных средств. Общие положения (введ. 01.07.1990). – М. : Стандартинформ, 2001. – 31 с.

9. ГОСТ Р ИСО/МЭК 15288-2005. Системная и программная инженерия. Процессы жизненного цикла систем (введ. 01.01.2007). – М. : Стандартинформ, 2006. – 57 с.

10. http://iso25000.com (дата обращения 27.05.2015).

11. http://www.uml.org (дата обращения 15.06.2015).

12. http://www.omg.prg/mof (дата обращения 17.11.2015).

13. http://www.omg.prg/spec/XMI (дата обращения 07.10.2015).

14. http://matlab.ru/products/simulink (дата обращения 20.09.2015).

15. Fairley R. E. Managing and Leading Software Projects. – Wiley-IEEE Comput. Soc. Press, 2009. – 512 p.

16. Naveda J. F., Seidman S. B. A Self-Study Guide for Today's Software Professional // IEEE Comput. Soc. Real-World Softw. Eng. Probl. – Wiley-IEEE Comput. Soc. Press, 2006. – 328 p.

17. http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html (дата обращения 20.10.2015).

18. http://www.softserveinc.com/en-us/services/software-testing (дата обращения 20.10.2015).

19. http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools (дата обращения 20.10.2015).

20. Boehm B., Basili V. Software Defect Reduction Top 10 List // IEEE Comput., IEEE Comput. Soc. – 2001. – Vol. 34, No.1. – P. 135–137.

21. Selby R. W. Software Engineering: Barry W. // Boehm's Lifetime Contrib. Software Dev., Manage. Res. – Wiley-IEEE Computer Society Press, 2007. – 832 p.

Organizatsiya avtomatizirovannogo kontrolya kachestva v zhiznennom tsikle programmnyh sredstv kriticheski vazhnyh sistem

Markovsky A. S., Samonov A. V., Burova I. O.
lexx26@list.ru, a.samonov@mail.ru, burova@mamaev.spb.ru
Military Space Forces Academy
Sankt-Petersburg, Russian Federation

Abstract. **The article presents the results of the analysis of the state and perspectives in the development of software of automated systems of critical infrastructure. It is established that the main directions in solving the field problems are the improvement of normative-methodical base, introduction of modern technologies for the development and implementation of end-to-end control quality software at all stages of their life cycle (LC). It is offered the improved model of LC software complying with the modern technology of software development life cycle, in which precisely defined the place and role processes of quality control. We describe the classification of quality characteristics software. The method and implementation technology for automated end-to-end quality control at all stages of the life cycle software are represented.**

Keywords: **quality control, quality software tools, software tools of weapons system, software development life cycle, model of software development life cycle, verification**

References

1. http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf (accessed 5 July 2015).

2. Kulyamin V. V. *Metodi verifikatcii programmnogo obespecheniya* [Methods of verification of the software], Moscow, Institut sistemogo programmirovaniya RAN, 2008, 111 p.

3. Genelt A. E. *Upravlenie kachestvom razrabotki programmnogo obespecheniya: uchebno-metodicheskoe posobiye* [Quality management of development of the software], St. Petersburg, ITMO, 2007, 187 p.

4. ISO/IEC 9126 Software engineering – Product quality. Part 1-41, 2001.

5. GOST R 51189-98. *Poryadok razrabotki programmnyh sredstv system vooruzheniya* [Order of development of software of systems of arms], 1999-07-01, Moscow, Standartinform, 2010, 16 p.

6. GOST 34.601-90. *Avtomatizirovannye sistemy. Stadiisozdaniya* [The automated systems. Development stages], 1992-01-01, Moscow, Standartinform, 2009, 6 p.

7. GOST RISO/MEK 12207-2010. *Protsessy zhiznennogo tsikla programmnyh sredstv* [Processes of life cycle of software], 2012–03–01, Moscow, Standartinform, 2011, 105 p.

8. GOST 28195-89. *Otsenka kachestva programmnyh sredstv. Obshchie polozhenya* [Assessment of quality of software. General provisions], 1990-07-01, Moscow, Standartinform, 2001, 31 p.

9. GOST R ISO/MEK 15288-2005. *Systemnaya i programmnaya inzheneriya. Protsessy zhiznennogo tsikla sistem* [Systems and software engineering. System life cycle processes], 2007–01–01, Moscow, Standartinform, 2006, 57 p.

10. http://iso25000.com (accessed 27 May 2015).

11. http://www.uml.org (accessed 15 June 2015).

12. http://www.omg.prg/mof (accessed 17 Nov. 2015).

13. http://www.omg.prg/spec/XMI (accessed 07 Oct. 2015).

14. http://matlab.ru/products/simulink (accessed 20 Sept. 2015).

15. Fairley R. E. Managing and Leading Software Projects. Wiley-IEEE Comput. Soc. Press, 2009, 512 p.

16. Naveda J. F., Seidman S. B. A Self-Study Guide for Today's Software Professional, *IEEE Comput. Soc. Real-World Softw. Eng. Probl.* Wiley-IEEE Comput. Soc. Press, 2006, 328 p

17. .http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html (accessed 20 Oct. 2015).

18. http://www.softserveinc.com/en-us/services/software-testing (accessed 20 Oct. 2015).

19. http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools (accessed 20 Oct. 2015).

20. Boehm B., Basili V. Software Defect Reduction Top 10 List, *IEEE Comput., IEEE Comput. Soc.*, 2001, Vol. 34, no. 1, pp. 135-137.

21. Selby R. W. Software Engineering: Barry W. Boehm's Lifetime Contrib. Software Dev., Manage. Res., Wiley-IEEE Computer Society Press, 2007. 832 p.