§ 5 СИСТЕМНЫЙ АНАЛИЗ, ПОИСК, АНАЛИЗ И ФИЛЬТРАЦИЯ ИНФОРМАЦИИ

Голосовский М. С.

ИНФОРМАЦИОННО-ЛОГИЧЕСКАЯ МОДЕЛЬ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация: *Предметом исследования являются ранние стадии жизненного цикла программного обеспечения, от качества выполнения работ на которых существенно зависит качество результата разработки программного обеспечения. В результате анализа опыта практического применения широко используемых водопадной (каскадной), итеративной и инкрементной моделей жизненного цикла программного обеспечения показано, что они не в полной мере удовлетворяют потребностям практики. Вместе с тем, имеется возможность синтеза модели жизненного цикла программного обеспечения, объединяющей достоинства трех названных моделей. Методология исследования базируется на моделях жизненного цикла программного обеспечения, структурном системном анализе, программной инженерии и информационно-логическом моделировании. Основные выводы проведенного исследования заключаются в том, что разработана модель жизненного цикла программного обеспечения (для этапа его разработки), представленная в нотации UML-диаграммы, которая состоит из этапов инициации разработки, постановки инкремента, исполнения инкремента и завершения разработки. Практическая реализация разработанной модели обеспечивает сокращение времени, необходимого на разработку программного обеспечения и необходимой отчетной документации.*

Ключевые слова: *программное обеспечение, жизненный цикл программы, программная инженерия, информационно-логическое моделирование, модель жизненного цикла, разработка программного обеспечения, структурный системный анализ, стадии жизненного цикла, программные средства, системная инженерия*

На протяжении последних нескольких десятилетий актуальной является задача повышения продуктивности, качества и надёжности разработки программного обеспечения [1-12]. В жизненном цикле программного обеспечения, как правило, выделяют 5 стадий: инициация, планирование (проектирование), выполнение (разработка, испытания, серийное производство), контроль и мониторинг (сопровождение), завершение [4, 5, 7, 11]. Реализуемый

DOI: 10.7256/2305-6061.2015.1.14119

при этом системный подход к анализу, проектированию, оценке, реализации, тестированию, обслуживанию и модернизации программного обеспечения называют *программной инженерией* [7, 14-15]. Несмотря на большое число исследований, посвященных совершенствованию методов поддержки жизненного цикла программного обеспечения и методов программной инженерии, до 80% проектов по созданию программного обеспечения считают неудачными [4-7, 16]. При этом к числу наиболее проблемных вопросов разработки программного обеспечения относят трудности управления его жизненным циклом [4-7, 17-27]. Структуру, определяющую последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла программного обеспечения, называют *моделью жизненного цикла* программного обеспечения [4-7]. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система (в рассматриваемом случае – программное обеспечение) создается и функционирует.

Федеральным агентством по техническому регулированию и метрологии РФ 01.03.2012 г. принят стандарт ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств», идентичный международному стандарту ISO/IEC 12207:2008 «System and software engineering — Software life cycle processes». Этот стандарт не конкретизирует выбор модели жизненного цикла программного обеспечения, а лишь описывает структуру процессов жизненного цикла, не конкретизируя, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Модель жизненного цикла программного обеспечения включает: стадии, результаты выполнения работ на каждой стадии и ключевые события (точки завершения работ и принятия решений). На каждой стадии могут выполняться один или несколько из 43 процессов семи групп, определенных в стандарте, и наоборот, один и тот же процесс может выполняться на различных стадиях [4-7, 24-27]. Соотношение между процессами и стадиями разработки программного обеспечения определяется используемой моделью жизненного цикла программного обеспечения, что обусловливает актуальность ее совершенствования.

Базовые модели жизненного цикла программного обеспечения

Одной из наиболее часто применяемых моделей разработки программного обеспечения является *водопадная (каскадная) модель*, лежащая в основе ГОСТ 34-й серии [4-7, 21-27]. Согласно этой модели процесс разработки программного обеспечения выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

Однако в условиях неопределённости предметной области зачастую приходится дорабатывать и изменять облик разрабатываемого программного обеспечения в процессе разработки по следующим причинам:

- на начальных этапах допускаются ошибки в модели предметной области.
- производится проверка гипотез, которые могут оказаться несостоятельными;
- производится уточнение моделей на уровне человеко-машинного взаимодействия;
- отсутствует четкое понимание свойств и характеристик результирующего продукта;
- используются новые технологии и инструменты, которые могут оказаться неприменимыми в исследуемой области.

В связи с этим классическая водопадная модель оказывается неприменимой, и воз-

DOI: 10.7256/2305-6061.2015.1.14119

никает необходимость использования модели, ориентированной на частые изменения разрабатываемой системы.

При разработке программного обеспечения в условиях изменяющихся требований и высокой неопределённости предметной области применяют итеративные гибкие процессы (методологии) разработки семейства Agile (семейство «гибкие процессы»): scrum, kanban, OpenUP и др. [21-27]. Основной идеей, лежащей в основе этих процессов, является частая смена итераций с получением на каждой итерации работающего программного образца, пусть и реализующего не весь функционал, заложенный на этапе постановки задачи и анализа требований. Основным достоинством этого подхода является возможность быстро реагировать на возникающие изменения и изменять направление разработки. Тем не менее, внедрение процессов семейства Agile сопряжено со следующими трудностями:

- требуется полный комплект документации на разработанное программное обеспечение, что противоречит основному принципу гибких процессов разработки: «Работающий программный продукт важнее полной документации»;
- в тех случаях, когда невозможно получить новую версию программного продукта за короткую итерацию, происходит снижение его качества;
- проекты, как правило, выполняются в жесткой функциональной структуре организации, что препятствует использованию принципа самоорганизующихся команд.

Комбинированная (инкрементно-итеративная) модель жизненного цикла программного обеспечения

Для решения указанных проблем предложена модель разработки программного обеспечения, основанная на смеси инкрементной и итеративной разработки с параллельным формированием программной документации. Основное различие между итеративной и инкрементной разработкой программного обеспечения состоит в том, что итеративная разработка подразумевает создание системы (пусть с ограниченным функционалом) и её последующее улучшение в течение нескольких итераций [23, 25]. Достоинством итеративной разработки является возможность получения работающего образца программного продукта на каждой итерации, с возможностью определения его соответствия требованиям и целесообразности продолжения дальнейшей работы над программным образцом (модулем), в случае если необходимо продолжение разработки, то всегда можно выбрать: продолжать работу с текущей итерацией или можно развивать альтернативное направление, с одной из более ранних итераций. В этом кроется один из недостатков итеративного подхода – сложность управления итеративными проектами высока, в связи с тем, что не всегда можно предсказать количество итераций, которое может потребоваться для разработки программного образца [4-7, 22-26].

При инкрементной разработке программный продукт разбивают на несколько законченных функциональных частей, и в рамках каждого инкремента производят разработку одной функциональной части и её тестирование. Финальным инкрементом является интеграция всех модулей и тестирование всего программного продукта. Достоинствами инкрементной стратегии разработки являются легкое управление сроками и расписанием процесса разработки за счет разбиения системы на части (количество частей известно, что позволяет отслеживать разработку каждой части системы) [5, 26].

DOI: 10.7256/2305-6061.2015.1.14119

Недостатками инкрементной разработки являются [5, 23, 26]:

- возможные ошибки в определении количества модулей на начальном этапе разработки, что может привести к возможному срыву плана разработки;
- отсутствие возможности точного прогнозирования качества результирующего продукта, полученного в результате сборки исходных частей.

Модель разработки программного обеспечения в нотации диаграммы деятельности языка UML представлена на рис. 1.

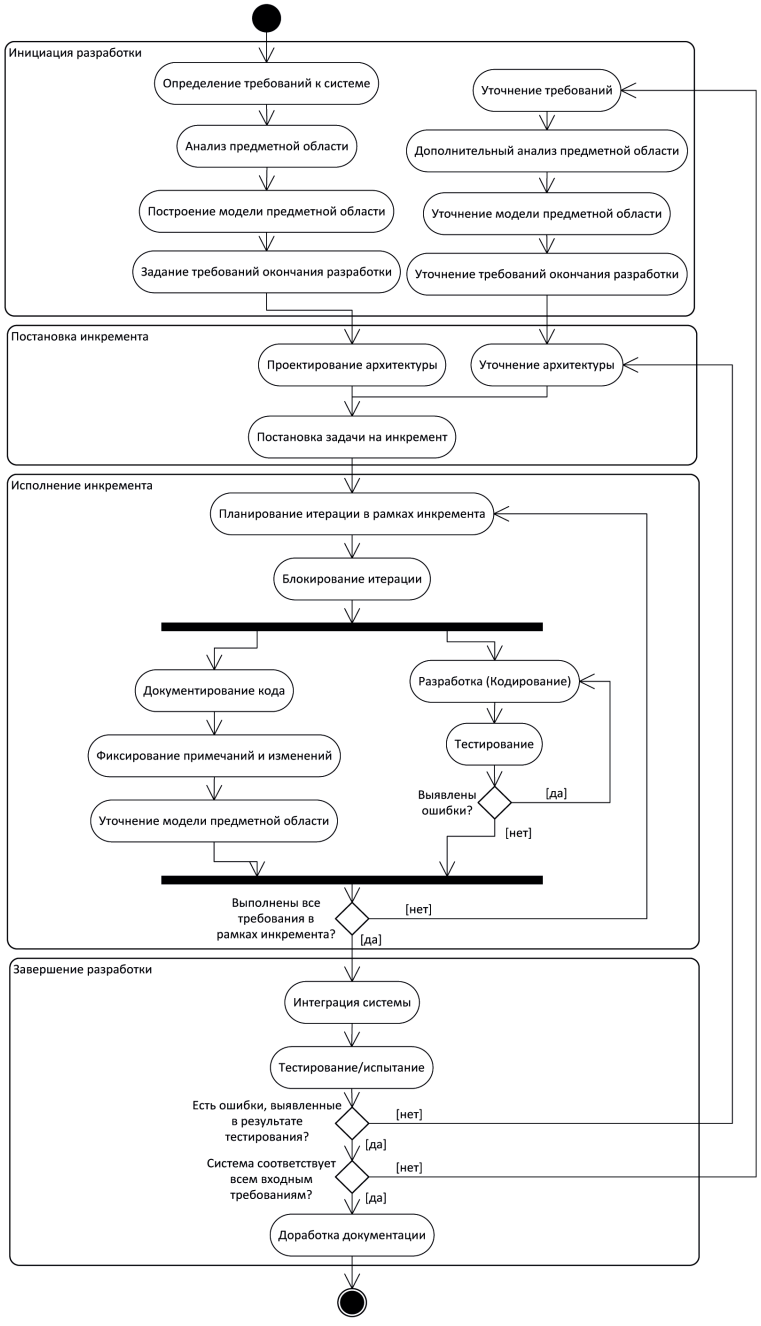

Рисунок 1 – Модель жизненного цикла разработки программного образца.

DOI: 10.7256/2305-6061.2015.1.14119

Она состоит из четырех основных этапов (рис. 1): инициация разработки, постановка инкремента, исполнение инкремента и завершение разработки.

На этапе инициации разработки производят определение требований к разрабатываемой системе, анализ предметной области, на основе результатов которого строят модель предметной области, задают требования и критерии завершения разработки.

На этапе постановки инкремента с использованием модели предметной области разрабатывают общую архитектуру системы, выделяют модули, разрабатываемые в отдельных инкрементах и определяют их интерфейсы для связи между собой и последующей интеграции. Если систему нельзя разбить на несколько слабо связанных между собой частей (модулей), модель выродится в итеративную. В завершении этапа постановки разрабатываются критерии достижения цели, определяющие в каком случае можно считать инкремент завершенным.

Следующим этапом является этап исполнения. На этом этапе разрабатывают работающий образец по итеративной стратегии. Начинается этап с планирования итерации. При планировании описывают возможные пути решения задачи и необходимые для их реализации инструменты и методы. На этой стадии можно применять различные методы коллективной генерации идей.

После планирования проводят блокирование итерации, при котором выбирают один, наиболее подходящий вариант плана, методов и инструментов. При необходимости для выбора применяют методы экспертных оценок (голосование, ранжирование, МАИ и др.) [1, 9, 10, 15, 17-20]. В общем случае в литературе по гибким процессам разработки рекомендуется применение как коротких итераций: 2-3 дня, так и итераций продолжительностью около месяца (25-35 дней).

На практике наиболее эффективной длительностью итерации оказалась одна неделя (5 рабочих дней). При этом завершение итерации совпадает с еженедельным подведением итогов. В связи с тем, что панируемая длительность итерации оказывается равной 5 дням, на планирование и блокировку итерации отводят порядка 4 часов. На выходе процесса блокировки итерации формируют план выполнения, содержащий задачи на итерацию, список исполнителей и их распределение по задачам, приоритеты или последовательность выполнения задач.

После выполнения блокировки итерации выполнение идет по двум параллельным ветвям: разработка программного образца и написание документации. При разработке программного образца рекомендуется применять стандарты кодирования, единые в рамках организации, а так же выполнять комментирование программного кода. Для быстрого получения документированного программного кода рекомендуется использовать системы документирования исходных текстов Doxygen или PHPDocumentor [5-7, 24-26].

Также в процессе разработки часто возникают идеи: как можно сделать эту итерацию другими способами или улучшить разрабатываемый программный образец в рамках итерации. При выполнении этой модели должно выполняться основное правило: лю-

DOI: 10.7256/2305-6061.2015.1.14119

бые изменения в рамках итерации запрещены. Это связано с тем, что новые идеи могут возникать довольно часто, при этом частые изменение состава задач и инструментов в течение итерации могут привести к тому, что в результате итерации ничего не будет реализовано. Но возникающие идеи могут нести рациональное зерно, к тому же в результате разработки могут обнаруживаться ранее неучтенные «пробелы» в предметной области. Все предложения фиксируют в журнале предложений и изменений, там же фиксируют все выполненные за время итерации запланированные изменения. В случае необходимости вносят изменения в модель предметной области. Ветка разработки (написания кода) завершается тестированием: если программный код не проходит тестирование, он возвращается на этап разработки. В процессе тестирования следует использовать технологии TDD [23] или модульное тестирование (Unit Testing) [21].

После прохождения тестирования, проверяют – все требования, поставленные на инкремент, выполнены или нет, в случае, если выполнены не все требования, производят новую итерацию, но при планировании и блокировке итерации учитывают предложения и изменения, полученные на предыдущей итерации.

За этапом исполнения инкремента следует этап завершения разработки. На этом этапе производят интеграцию модулей, полученных в рамках предыдущих инкрементов. По завершении интеграции производят тестирование и проверку всей системы на наличие ошибок. В случае выявления ошибок, полученные в результате интеграции, программный продукт отправляют на доработку, начиная с этапа постановки инкремента, на котором производят уточнение архитектуры системы и выполняют одну или несколько итераций по доработке.

Тестирование на соответствие исходным требованиям (проведение испытаний), производят после устранения ошибок, возникших в результате интеграции модулей. Недостатки и несоответствия, выявленные в процессе испытаний, могут носить более глобальный характер и потребовать проведения уточнения требований, проведения дополнительного анализа предметной области, уточнения требований окончания разработки. По результатам выполненных испытаний формируют акт испытаний. Заключительным процессом этапа завершения разработки является процесс доработки документации.

Заключение

Предложенная модель жизненного цикла разработки программного обеспечения была применена при разработке пяти программных комплексов: качественные (удовлетворяющие требованиям заказчика) программные продукты созданы с сокращением времени на разработку отчетной и программной документации, по сравнению с ранее выполненными сопоставимыми по сложности и трудоемкости разработками программного обеспечения, на 15-25%.

DOI: 10.7256/2305-6061.2015.1.14119

Библиография :

1. Богомолов А.В., Зуева Т.В., Чикова С.С., Голосовский М.С. Экспертно–аналитическое обоснование приоритетных направлений совершенствования системы предупреждения биологических террористических актов // Информатика и системы управления. 2009. № 4. С. 134–136.
2. Богомолов А.В., Майстров А.И. Технология анализа системных причинно-следственных связей на основе диаграмм Исикавы // Системный анализ в медицине (САМ 2014): Материалы VIII международной научной конференции. Благовещенск, 2014. С. 13-16.
3. Богомолов А.В., Чуйков Д.С., Запорожский Ю.А. Средства обеспечения безопасности информации в современных автоматизированных системах // Информационные технологии. 2003. № 1. С. 2.
4. Брукс Ф. Мифический человеко-месяц, или Как создаются программные системы. СПб.: Символ-Плюс, 2010. 304 с.
5. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс. 2008. 720 с.
6. Виноградов А.Н., Макаренков С.А., Чиров Д.С. Применение методов data mining для формирования базы знаний экспертной системы классификации радиосигналов // T-Comm: Телекоммуникации и транспорт. 2010. Т. 4. № 11. С. 61-64.
7. Голосовский М.С. Модель жизненного цикла разработки программного обеспечения в рамках научно-исследовательских работ // Автоматизация и современные технологии. 2014. № 1. С. 43-46.
8. Голосовский М.С., Шашин А.Е. Технология адаптивного синтеза системы тестового контроля качества автоматизированного обучения // Материалы VIII международной научной конференции «Системный анализ в медицине» (САМ 2014). Благовещенск, 2014. С. 71-74.
9. Есев А.А., Мережко А.Н., Ткачук А.В. Технология квалиметрии технического уровня сложных систем // Вестник компьютерных и информационных технологий. 2014. № 7 (121). С. 28-34.
10. Есев А.А., Ткачук А.В., Зыкин А.П. Методическое обеспечение исследования технического уровня образцов вооружения и военной техники // Двойные технологии. 2014. № 1 (66). С. 59-64.
11. Кон М. Scrum: гибкая разработка ПО = Succeeding with Agile: Software Development Using Scrum. М.: Вильямс, 2011. 576 р.
12. Кукушкин Ю.А., Богомолов А.В., Ушаков И.Б. Математическое обеспечение оценивания состояния материальных систем // Информационные технологии. 2004. № 7 (приложение). 32 с.
13. Куликов Г.В., Соснин Ю.В., Непомнящих А.В., Нащекин П.А. Моделирование процесса защиты информации при реализации несанкционированного доступа к ней // Вестник компьютерных и информационных технологий. 2014. № 4 (118). С. 45-51.
14. Ларман К., Базилию В. Итеративная и инкрементальная разработка: краткая история // Открытые системы. 2003. № 9. С. 43-53.
15. Максимов И.Б., Столяр В.П., Богомолов А.В. Прикладная теория информационного обеспечения медико-биологических исследований. М.: Бином, 2013. 312 с.
16. Северцев Н.А., Шипилов В.В. Моделирование процесса управления инновационными проектами организаций в условиях неопределенности // Вопросы теории безопасности и устойчивости систем. 2014. № 16. С. 3-15.

DOI: 10.7256/2305-6061.2015.1.14119

17. Чиров Д.С., Терешонок М.В., Елсуков Б.А. Метод и алгоритмы оптимизации технических характеристик комплексов радиомониторинга // T-Comm: Телекоммуникации и транспорт. 2014. Т. 8. № 10. С. 88-92.
18. Харитонов В.В., Мережко А.Н., Есев А.А., Зыкин А.П. Структурный системный анализ процессов управления затратами на летные испытания авиационного вооружения и военной техники // Известия Института инженерной физики. 2013. Т. 2. № 28. С. 32-35.
19. Шипилов В.В. Об эпиморфном преобразовании многомерных данных в задачах построения сложных технических систем // Вопросы теории безопасности и устойчивости систем. 2014. № 16. С. 27-39.
20. Шипилов В.В., Сахаров О.В. Планирование вариантов групп исполнителей для обеспечения выполнения этапов проекта // Нелинейный мир. 2014. Т. 12. № 7. С. 84-86.
21. Agile-манифест разработки программного обеспечения. [Электронный ресурс] URL: http://agilemanifesto.org/iso/ru/ (дата обращения 24.12.2014).
22. Beck K. Embracing Change with Extreme Programming // Computer, № 32 (10), 1999. PP. 70-77.
23. Beck K. Test Driven Development: by example. Addison-Wesley Professional 2002. 252 p.
24. Brooks F. No Silver Bullet — Essence and Accident in Software Engineering // Proceedings of the IFIP Tenth World Computing Conference: 1069–1076. 1986.
25. Cockburn L. Using both incremental and iterative development // Cross Talk: the journal of defense software Engineering, №5, 2008. PP. 27-30.
26. Larman C. Agile and Iterative Development: A Manager's Guide // Addison-Wesley, 2004. 27 p.
27. Royce W. Managing the Development of Large Software Systems// TRW, August 1970. PP. 328-338.
28. Кубашева Е.С., Гаврилов А.Г. Методика оценки качества веб-приложений // Программные системы и вычислительные методы.-2013.-1.-С. 28-34. DOI: 10.7256/2305-6061.2013.01.2.
29. Катасёв А.С., Емалетдинова Л.Ю. Нечетко-продукционная каскадная модель диагностики состояния сложного объекта // Программные системы и вычислительные методы.-2013.-1.-С. 69-81. DOI: 10.7256/2305-6061.2013.01.6.

References:

1. Bogomolov A.V., Zueva T.V., Chikova S.S., Golosovskii M.S. Ekspertno–analiticheskoe obosnovanie prioritetnykh napravlenii sovershenstvovaniya sistemy preduprezhdeniya biologicheskikh terroristicheskikh aktov // Informatika i sistemy upravleniya. 2009. № 4. S. 134–136.
2. Bogomolov A.V., Maistrov A.I. Tekhnologiya analiza sistemnykh prichinno-sledstvennykh svyazei na osnove diagramm Isikavy // Sistemnyi analiz v meditsine (SAM 2014): Materialy VIII mezhdunarodnoi nauchnoi konferentsii. Blagoveshchensk, 2014. S. 13-16.
3. Bogomolov A.V., Chuikov D.S., Zaporozhskii Yu.A. Sredstva obespecheniya bezopasnosti informatsii v sovremennykh avtomatizirovannykh sistemakh // Informatsionnye tekhnologii. 2003. № 1. S. 2.
4. Bruks F. Mificheskii cheloveko-mesyats, ili Kak sozdayutsya programmnye sistemy. SPb.: Simvol-Plyus, 2010. 304 s.
5. Buch G. Ob"ektno-orientirovannyi analiz i proektirovanie s primerami prilozhenii. M.: Vil'yams. 2008. 720 s.
6. Vinogradov A.N., Makarenkov S.A., Chirov D.S. Primenenie metodov data mining dlya formirovaniya bazy znanii ekspertnoi sistemy klassifikatsii radiosignalov // T-Comm: Telekommunikatsii i transport. 2010. T. 4. № 11. S. 61-64.

DOI: 10.7256/2305-6061.2015.1.14119

7. Golosovskii M.S. Model' zhiznennogo tsikla razrabotki programmnogo obespecheniya v ramkakh nauchno-issledovatel'skikh rabot // Avtomatizatsiya i sovremennye tekhnologii. 2014. № 1. S. 43-46.
8. Golosovskii M.S., Shashin A.E. Tekhnologiya adaptivnogo sinteza sistemy testovogo kontrolya kachestva avtomatizirovannogo obucheniya // Materialy VIII mezhdunarodnoi nauchnoi konferentsii «Sistemnyi analiz v meditsine» (SAM 2014). Blagoveshchensk, 2014. S. 71-74.
9. Esev A.A., Merezhko A.N., Tkachuk A.V. Tekhnologiya kvalimetrii tekhnicheskogo urovnya slozhnykh sistem // Vestnik komp'yuternykh i informatsionnykh tekhnologii. 2014. № 7 (121). S. 28-34.
10. Esev A.A., Tkachuk A.V., Zykin A.P. Metodicheskoe obespechenie issledovaniya tekhnicheskogo urovnya obraztsov vooruzheniya i voennoi tekhniki // Dvoinye tekhnologii. 2014. № 1 (66). S. 59-64.
11. Kon M. Scrum: gibkaya razrabotka PO = Succeeding with Agile: Software Development Using Scrum. M.: Vil'yams, 2011. 576 r. 12.
12. Kukushkin Yu.A., Bogomolov A.V., Ushakov I.B. Matematicheskoe obespechenie otsenivaniya sostoyaniya material'nykh sistem // Informatsionnye tekhnologii. 2004. № 7 (prilozhenie). 32 s.
13. Kulikov G.V., Sosnin Yu.V., Nepomnyashchikh A.V., Nashchekin P.A. Modelirovanie protsessa zashchity informatsii pri realizatsii nesanktsionirovannogo dostupa k nei // Vestnik komp'yuternykh i informatsionnykh tekhnologii. 2014. № 4 (118). S. 45-51.
14. Larman K., Baziliyu V. Iterativnaya i inkremental'naya razrabotka: kratkaya istoriya // Otkrytye sistemy. 2003. № 9. S. 43-53.
15. Maksimov I.B., Stolyar V.P., Bogomolov A.V. Prikladnaya teoriya informatsionnogo obespecheniya mediko-biologicheskikh issledovanii. M.: Binom, 2013. 312 s.
16. Severtsev N.A., Shipilov V.V. Modelirovanie protsessa upravleniya innovatsionnymi proektami organizatsii v usloviyakh neopredelennosti // Voprosy teorii bezopasnosti i ustoichivosti sistem. 2014. № 16. S. 3-15.
17. Chirov D.S., Tereshonok M.V., Elsukov B.A. Metod i algoritmy optimizatsii tekhnicheskikh kharakteristik kompleksov radiomonitoringa // T-Comm: Telekommunikatsii i transport. 2014. T. 8. № 10. S. 88-92.
18. Kharitonov V.V., Merezhko A.N., Esev A.A., Zykin A.P. Strukturnyi sistemnyi analiz protsessov upravleniya zatratami na letnye ispytaniya aviatsionnogo vooruzheniya i voennoi tekhniki // Izvestiya Instituta inzhenernoi fiziki. 2013. T. 2. № 28. S. 32-35.
19. Shipilov V.V. Ob epimorfnom preobrazovanii mnogomernykh dannykh v zadachakh postroeniya slozhnykh tekhnicheskikh sistem // Voprosy teorii bezopasnosti i ustoichivosti sistem. 2014. № 16. S. 27-39.
20. Shipilov V.V., Sakharov O.V. Planirovanie variantov grupp ispolnitelei dlya obespecheniya vypolneniya etapov proekta // Nelineinyi mir. 2014. T. 12. № 7. S. 84-86.
21. Agile-manifest razrabotki programmnogo obespecheniya. [Elektronnyi resurs] URL: http://agilemanifesto.org/iso/ru/ (data obrashcheniya 24.12.2014).
22. Beck K. Embracing Change with Extreme Programming // Computer, № 32 (10), 1999. RR. 70-77.
23. Beck K. Test Driven Development: by example. Addison-Wesley Professional 2002. 252 p.
24. Brooks F. No Silver Bullet — Essence and Accident in Software Engineering // Proceedings of the IFIP Tenth World Computing Conference: 1069–1076. 1986.
25. Cockburn L. Using both incremental and iterative development // Cross Talk: the journal of defense software Engineering, №5, 2008. RR. 27-30.

DOI: 10.7256/2305-6061.2015.1.14119

26. Larman S. Agile and Iterative Development: A Manager's Guide // Addison-Wesley, 2004. 27 r.
27. Royce W. Managing the Development of Large Software Systems// TRW, August 1970. RR. 328-338.
28. Kubasheva E.S., Gavrilov A.G. Metodika otsenki kachestva veb-prilozhenii // Programmnye sistemy i vychislitel'nye metody.-2013.-1.-C. 28-34. DOI: 10.7256/2305-6061.2013.01.2.
29. Katasev A.S., Emaletdinova L.Yu. Nechetko-produktsionnaya kaskadnaya model' diagnostiki sostoyaniya slozhnogo ob"ekta // Programmnye sistemy i vychislitel'nye metody.-2013.-1.-C. 69-81. DOI: 10.7256/2305-6061.2013.01.6.

DOI: 10.7256/2305-6061.2015.1.14119