УДК 004.94 **doi: 10.20998/2522-9052.2017.1.15**

С. Г. Семенов[1], Кассем Халифе[2], М. М. Захарченко[3]

[1] Национальный технический университет "ХПИ", Харьков, Украина
[2] Технический институт, Мараке, Ливан
[3] Харьковский национальный университет Воздушных Сил имени И. Кожедуба, Харьков, Украина

УСОВЕРШЕНСТВОВАННЫЙ СПОСОБ МАСШТАБИРОВАНИЯ ГИБКОЙ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Предметом изучения статьи является усовершенствованный способ масштабирования гибкой методологии разработки программного обеспечения. **Цель** - снижение потенциальных потерь, обусловленных рисками безопасности разработки и эксплуатации программного обеспечения на большинстве этапов его жизненного цикла. **Задачи**: анализ существующих методологий и подходов разработки программного обеспечения, исследование возможностей масштабирования методологий в рамках отдельных проектов, усовершенствование общей схемы цикла разработки программного обеспечения, разработка структуры управления разработкой программного обеспечения в рамках как отдельно взятого проекта, так и организации-разработчика в целом, разработка практических рекомендаций по повышению безопасности программного обеспечения на различных этапах жизненного цикла. Используемыми **методами** являются**:** системный анализ рисков, причинно-следственный анализ. Получены следующие **результаты.** Проведен анализ существующих гибких методологий разработки программного обеспечения, определены перспективные направления и подходы данной индустрии, выявлены возможности масштабирования гибких методологий. Усовершенствована схема жизненного цикла разработки программного обеспечения, отличительной особенностью которой является введение дополнительных подразделов и ролей, имеющих целью повышение безопасности программного обеспечения. Усовершенствована структура управления разработкой программного обеспечения, отличающаяся от известных учетом рисков безопасности в процессе разработки. Разработаны практические рекомендации использования усовершенствованного способа масштабирования гибкой методологии. **Выводы.** Реализация предложенного усовершенствованного способа масштабирования существующей методологии разработки ПО, отличается от известных включением и использованием в команде разработчиков дополнительных специалистов безопасности. Это может повлечь некоторое замедление выполнения кода и увеличение количества выявленных дефектов (багов) при альфа-тестировании, а, следовательно, увеличение времени жизненного цикла багов. Однако в перспективе эти локальные ухудшения позволяют добиваться лучшего конечного результата (повышения безопасности разработанного ПО) и обеспечивать как быстрый рост функционала, так и приемлемый уровень качества сервиса. А это в свою очередь будет привлекательным мотивом дальнейшего сотрудничества заказчика и фирмы-разработчика.

Ключевые слова: безопасность программного обеспечения, гибкие методологии разработки программного обеспечения, Agile, Scrum.

Введение

Концепция качества программного обеспечения (ПО) возникла как обобщение ряда похожих, но в то же время и имеющих определенные отличия концепций. Эти концепции были предложены рядом крупных специалистов в области менеджмента [1, 4], которых часто называют «учителя качества». Все они оказали огромное влияние на экономики целых стран и способствовали переходу к эпохе «всеобщего качества» (TQM). Их теории, в отличие от социально-экономических доктрин прошлых лет, проверены по критерию эффективности, они имеют комплексный, собирательный характер, анализируя и синтезируя все наиболее ценное в опыте различных компаний и даже стран [4]. Эти концепции демонстрируют возрастающую роль всех участников процесса в достижении успешного развития и конкурентоспособности компаний (изделий), при этом подчеркивают важность мотивации и непрерывного обучения, совершенствования персонала, его адаптации к изменениям конъюнктуры и «вызовам» внешних факторов.

Основываясь на уже известные постулаты менеджмента IT-индустрии, в последнее время разработчики программного обеспечения непрерывно ищут новые пути решения задач оптимизации процесса управления разработкой ПО и усовершенствуют существующие гибкие методологии. Это приводит к появлению усовершенствованных правил и способов разработки, новых подходов коммуникаций и управления и даже появлению новых профессий (например, DevOps). При этом вопросы, связанные с организацией работы, при неизменном качестве ПО, остаются. Кроме того в условиях возникновения новых рисков на всех этапах жизненного цикла программного обеспечения (в частности рисков безопасности ПО) эти вопросы усложняются и, как показали исследования, спектр нерешенных актуальных задач имеет тенденцию расширения.

Анализ проблемы и постановка задачи

Анализ литературы [1–13] а так же результаты проведенных исследований показали, что комплексное использование современных подходов разработки ПО (Agile, DevOps, Lean и др.), которые, благодаря современным средствам автоматизации, получили новый стимул к развитию, в совокупности с технологиями синтеза и адаптации новых участников проектов (в соответствии с требованиями минимизации рисков), могут позволить минимизи-

ровать временные затраты на разработку и обеспечить определенное заказчиком качество ПО.

Можно заметить, что итеративность, обратная связь и гибкость Agile-методов, основанных на взаимодействии команд разработчиков, дают возможность для постоянного и непрерывного выпуска ПО, а синтез и адаптация усовершенствованных подходов [5, 13] в Agile-методы позволяет устранить ряд недостатков, связанных с существующими рисками экономического, социального, правового и других направлений [12].

В то же время, мировые тенденции увеличения киберпреступности и повышенного внимания заказчиков к безопасности эксплуатации ПО, требуют от разработчиков усовершенствования и реализации новых подходов и способов масштабирования методологий разработки ПО. В связи с этим поставленная задача масштабирования гибкой методологии разработки программного обеспечения является актуальной.

Решение проблемы

Как отмечено выше, воспользоваться существующим опытом масштабирования, а так же разработать новые решения необходимо в условиях повышения требований к безопасности программного обеспечения. В таких условиях необходимо видоизменять некоторые положения менеджмента и разработки ПО, а также вносить новые элементы в уже существующую систему и жизненный цикл ПО. Для обеспечения качества ПО предлагается усовершенствованная схема цикла разработки ПО представленная на рис. 1.

Как видно из этой схемы в общий цикл разработки ПО рекомендуется включить следующие подразделы:

1. «Чистота» кода – введение безопасного кодирования;

2. Безопасный DevOps – включение дополнительных субъектов и инструментов безопасности;

3. Моделирование угроз – предполагает введение новой роли специалиста управления кибербезопасностью;

4. Анализ рисков безопасности – анализ приоритетности в отставания.

Кроме этого дополнительные требования безопасности ПО требует введения новой роли «Этичный» хакер.

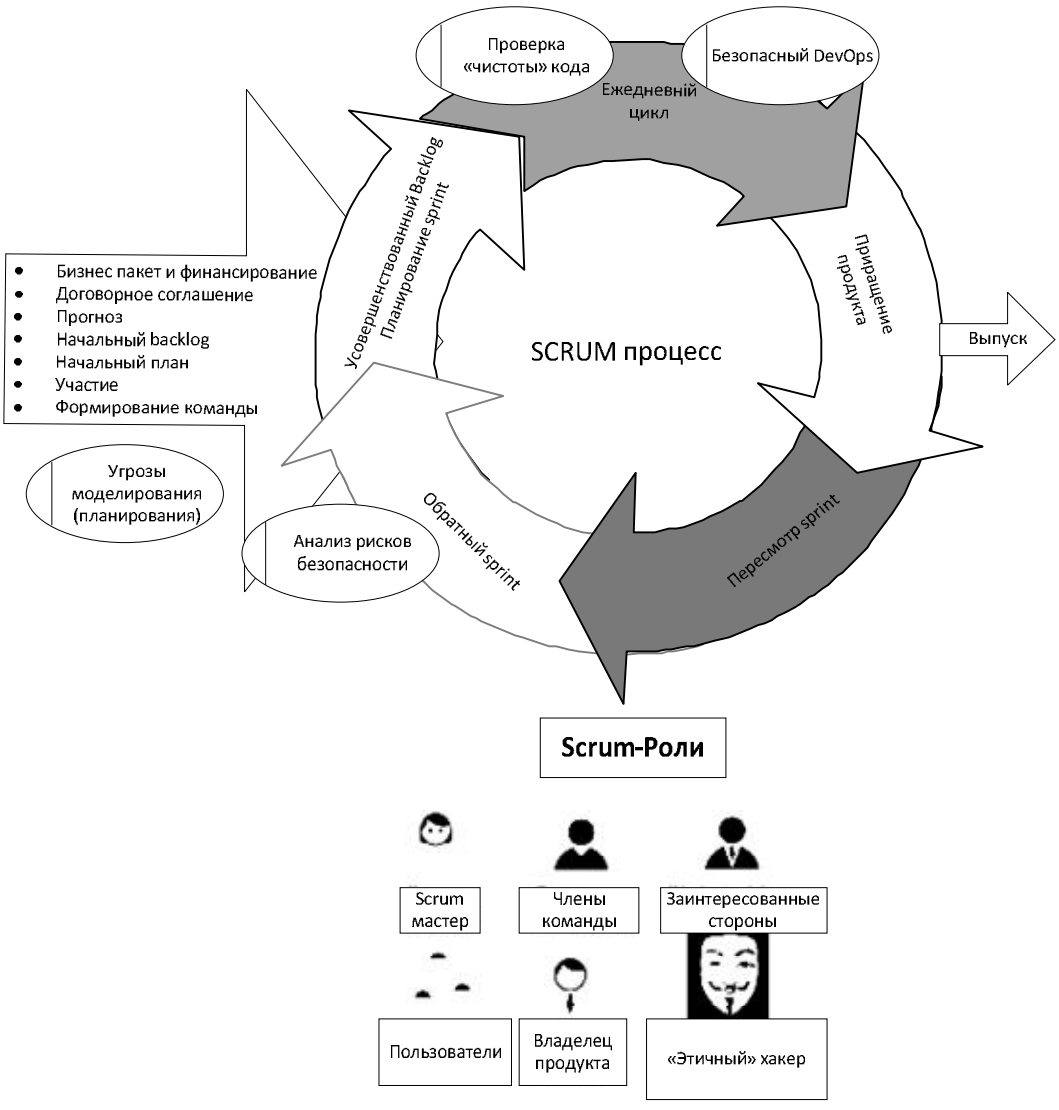

Рис. 1. Усовершенствованная схема цикла разработки ПО

Следует заметить, что внедрение в команду на разных этапах разработки программного обеспечения специалиста кибербезопасности, безопасного программирования и особенно тестирования безопасности («этичного хакинга») [2б 13], является одним из требований современного рынка разработки программного обеспечения. Принципы такого внедрения на данном этапе развития индустрии разработки программного обеспечения должны лежать в рамках «горизонтального» (межкомандного) управления. Данный подход полностью соответствует принципам бережливой разработки (Lean), которая в свою очередь, обеспечивает минимизацию издержек при выпуске новых версий программного обеспечения.

С учетом указанных основных принципов усовершенствованную обобщенную структуру управления разработкой программного обеспечения можно представить в виде схемы, показанной на рис. 2.

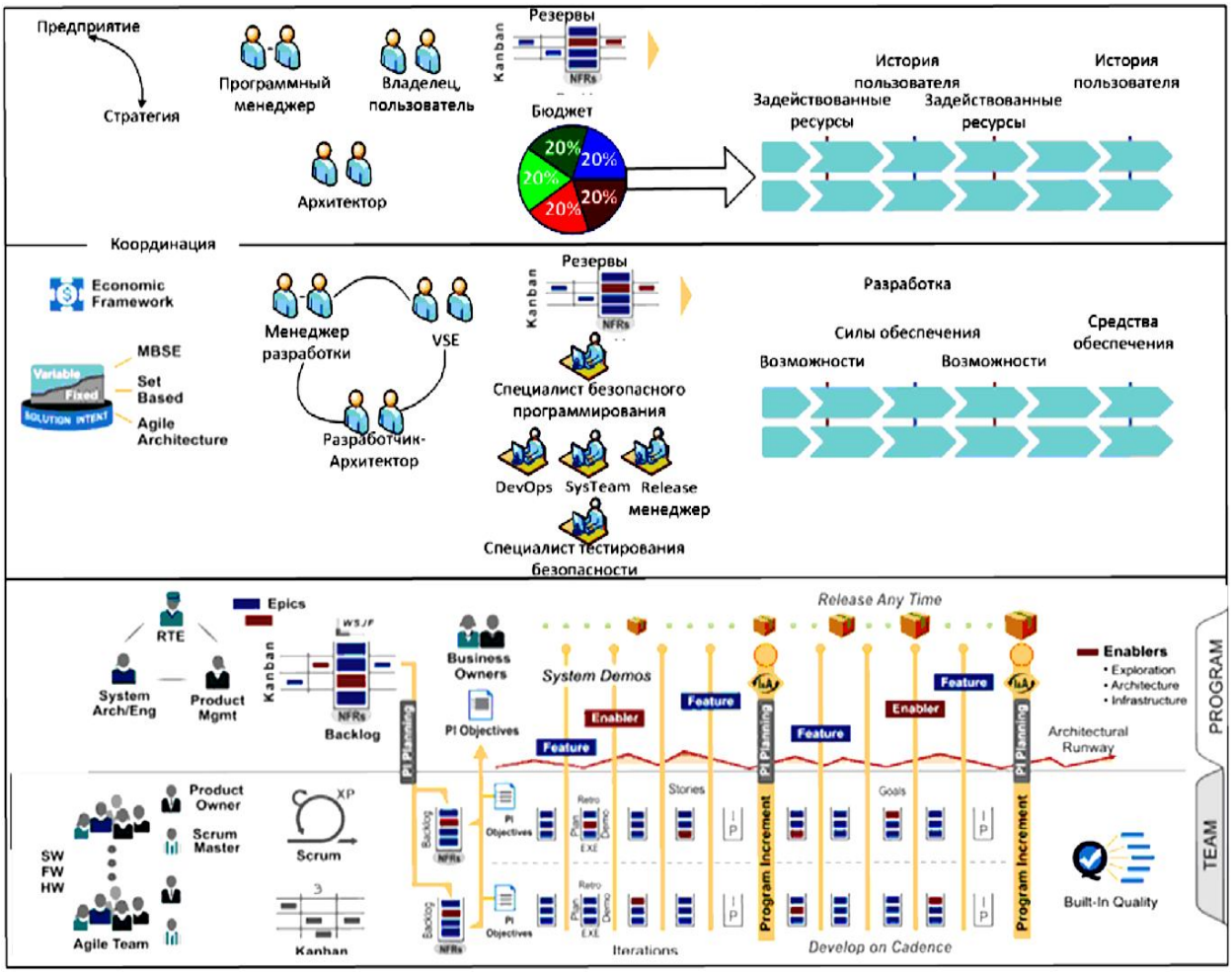

Рис. 2. Усовершенствованная обобщенная структура управления разработкой ПО

Следует заметить, что предложенная схема имеет своей целью оптимизацию и поддержку крупномасштабных разработок, т.е., работа большой команды идет не только по спринтам (двухнедельным циклам), как в Scrum, а с дополнительной итерацией планирования (program increment) длительностью десять недель.

В качестве примера рассмотрим случай, когда от бизнеса (заказчика) поступает запрос на новый функционал.

Бизнес-аналитик его рассматривает и преобразует в задания для разработчиков, которые создают или изменяют код и помещают его в хранилище.

В этот момент возможно подключение специалиста безопасного программирования, который на принципах «смоки» тестирования может оценить уровень безопасности кода.

В среде непрерывной интеграции (CI tools) код автоматически компилируется и сохраняется в нужной версии в репозитории.

Автоматически строится тестовая среда с необходимой конфигурацией для запуска последней версии кода. Автоматически осуществляется регрессионное тестирование и тестирование безопасности.

В случае успеха ресурсы тестовой среды высвобождаются и автоматически создается среда для интеграционного тестирования, а в системе ITSM формируется задание на внесение изменения в промышленную среду. Автоматически выполняется интеграционное тестирование. В случае

успеха и после согласования изменения в системе ITSM код автоматически разворачивается в промышленной среде.

Таким образом, от постановки требований до перевода новой версии в промышленную эксплуатацию проходят часы, а не месяцы, как в традиционных информационных технологиях. Иллюстрация интервалов времени на этапах разработки, а также объемов выполнения указанных функций безопасности представлена на временной диаграмме фаз и этапов разработки программного обеспечения (рис. 3).

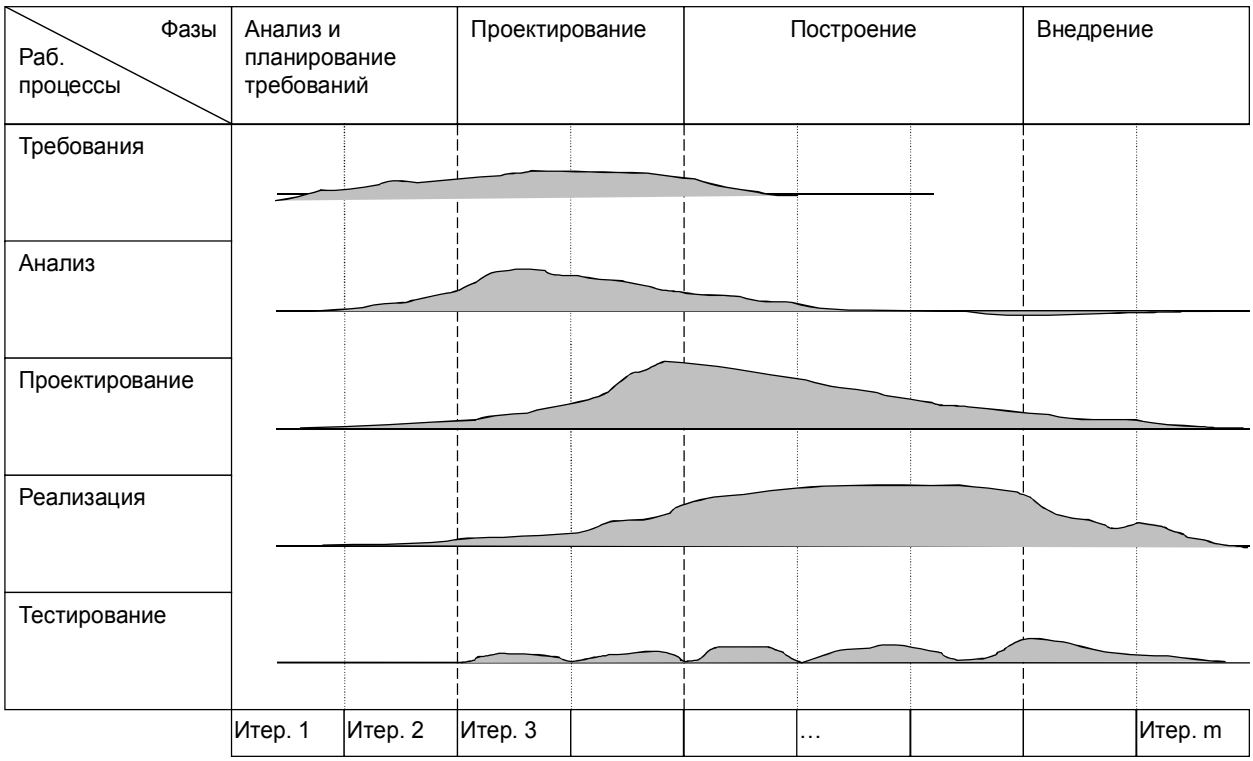

Рис. 3. Временная диаграмма фаз и этапов разработки ПО

Опишем более подробно функции безопасности на каждом этапе.

На этапе анализа и планирования требований идея превращается в концепцию готового продукта. При этом создаются:

- бизнес-план разработки;
- упрощенная модель вариантов использования;
- пробный вариант архитектуры.

На этом же этапе выявляются и оцениваются риски, расставляются приоритеты и выполняется грубая оценка проекта.

Специалистам по безопасному программированию и тестированию безопасности очень важно на этом этапе предоставить достоверную информацию о существующих рисках безопасности и представить общую концепцию (архитектуру) информационной защиты проекта.

На этапе проектирования выполняется детальное описание вариантов использования, формируется архитектура в виде представлений всех моделей и разрабатывается план действий и оценка ресурсов.

В этом случае, как и на первом этапе, очень важно избежать возможных ошибок безопасности и «лазеек» в архитектуре программного обеспечения и рисков их недооценки.

На этапе построения производится уточнение базового уровня архитектуры и реализуются все варианты использования. На этом этапе специалисты безопасного программирования консультативным путем и путем «смоки» тестирования должны обеспечить качество кода программного обеспечения, а специалисты тестирования должны выполнить полный набор тест-кейсов, имеющих отношение к безопасности и информационной защите.

На этапе внедрения осуществляется бета-тестирование, выполняются тренинги сотрудников заказчиков, а также устраняются выявленные дефекты.

Действия рассматриваемых специалистов на этом этапе аналогичны их действиям на предыдущем этапе, с той разницей, что специалисты безопасного программирования должны осуществлять контроль устранения уже выявленных ошибок безопасности программного обеспечения.

Выводы

В целом следует заметить, что реализация предложенного усовершенствованного способа масштабирования существующей методологии разработки программного обеспечения, отличается от известных включением и использованием в команде разработчиков дополнительных специали-

стов безопасности. Это может повлечь некоторое замедление выполнения кода и увеличение количества выявленных дефектов (багов) при альфа-тестировании, а, следовательно, увеличение времени жизненного цикла багов.

Однако в перспективе эти локальные ухудшения позволяют добиваться лучшего конечного результата (повышения безопасности разработанного программного обеспечения) и обеспечивать как быстрый рост функционала, так и приемлемый уровень качества сервиса.

А это в свою очередь будет привлекательным мотивом дальнейшего сотрудничества заказчика и фирмы-разработчика.

Количество дополнительных сил, которые необходимы в том или ином проекте следует оценивать исходя из его сложности (масштаба) а так же уровня требований безопасности заказчика.

СПИСОК ЛИТЕРАТУРЫ

1. Barry W. Boehm, Richard Turner. Balancing Agility and Discipline - A Guide for the Perplexed. New York : Addison-Wesley, 2004. 266 p.
2. Демарко Т., Листер Т. Человеческий фактор: успешные проекты и команды. Санкт-Петербург : Символ–Плюс, 2005. 256 p.
3. Ruby S., Thomas D., Hansson D.H. Agile Web Development with Rails 4. Pragmatic Programmers, LLC., 2013. 439 p. ISBN: 978-1-93778-556-7.
4. Гуру менеджмента качества и их концепции [Электронный ресурс]. [Э. Деминг, Дж. Джуран, Ф. Кросби, К. Исикава, А. Фейгенбаум, Т. Тагути]. Режим доступа: http://www.management.com.ua/qm/qm009.html (last accessed April 10, 2017).
5. Hasan Yasar. Security Practitioner Perspective on DevOps for Building Secure Solutions. [Электронный ресурс]. 2016. Режим доступа: http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid=474101&gaWebinar=Security PractitionerPerspectiveonDevOpsforBuildingSecureSolutions (last accessed April 10, 2017).
6. Highsmith J. Agile Software Development Ecosystems. Boston : AddisonWesley, 2006. 448 p.
7. Кузумано Майкл, Поппендик Мэри. Бережливая разработка программ [Электронный ресурс]. Открытые системы : СУБД. 2012. № 08/ Режим доступа: https://www.osp.ru/os/2012/08/13019237/ (last accessed April 10, 2017).
8. Makhmetov G. Ye. Kogda «agile» (ne) k mestu [Электронный ресурс]. Режим доступа: https://makhmetov.ru/ articles/agile.html (last accessed April 01, 2017).
9. Sherman Mark. Building Secure Software for Mission Critical Systems. [Электронный ресурс]. Режим доступа: http://resources.sei.cmu.edu/asset_files/Presentation/2017_017_001_495865.pdf (last accessed April 10, 2017).
10. Sherman Mark, Schiela Robert. From Secure Coding to Secure Software [Электронный ресурс]. Режим доступа: http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid=483646 (last accessed April 10, 2017).
11. Putu Adi? Guna Permana Scrum Method Implementation in a Software Development Project Management/ (IJACSA) International Journal of Advanced Computer Science and Applications. 2015. Vol. 6, № 9. P. 199-205.
12. Швачич Г. Г., Семенов С. Г., Главчев М. И., Халифе Кассем. Модель расчета временных границ проектов разработки программного обеспечения. Системи управління, навігації та зв'язку. Полтава : ПНТУ, 2017. Випуск 1 (41) . С. 43-49.
13. Klieber William, Snavely William Automated Code Repair Based on Inferred Specifications [Электронный ресурс]. Режим доступа: http://resources.sei.cmu.edu/asset_files/ConferencePaper/2016_021_001_483599.pdf (last accessed April 10, 2017).

REFERENCES

1. Barry, W. Boehm and Richard, Turner (2004), *Balancing Agility and Discipline* - A Guide for the Perplexed, Addison-Wesley, New York, 266 p.
2. Demarko, T. and Lister, T. (2005), Chelovecheskiy faktor: uspeshnyye proyekty i komandy, Simvol-Plyus, Sankt-Peterburg, 256 p.
3. Ruby, S., Thomas, D. and Hansson, D.H. (2013), Agile Web Development with Rails 4, Pragmatic Programmers, LLC, 439 p., ISBN: 978-1-93778-556-7.
4. Deming, E., Dzhuran, Dzh., Krosbi, F., Isikava, K., Feygenbaum, A. and Taguti, T.(2001), Guru menedzhmenta kachestva i ikh kontseptsii, available at : http://www.management.com.ua/qm/qm009.html (last accessed February 1, 2017).
5. Hasan, Yasar (2016), Security Practitioner Perspective on DevOps for Building Secure Solutions, available at : http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid=474101&gaWebinar=SecurityPractitionerPerspectiveon DevOpsforBuildingSecureSolutions (last accessed February 1, 2017).
6. Highsmith, J. (2006), Agile Software Development Ecosystems, Boston : AddisonWesley, 448 p.
7. Kuzumano, Maykland and Poppendik Meri (2012), “Berezhlivaya razrabotka program”, *Otkrytyye sistemy. SUBD*, No. 08, available at : https://www.osp.ru/os/2012/08/13019237/ (February 1, 2017).
8. Makhmetov, G.Ye. (2017), Kogda «agile» (ne) k mestu, available at : https://makhmetov.ru/articles/agile.html (last accessed February 1, 2017).
9. Sherman, Mark (2017), Building Secure Software for Mission Critical Systems, available at : http://resources.sei.cmu. edu/asset_files/Presentation/2017_017_001_495865.pdf (last accessed February 1, 2017).

10. Sherman, Mark and Schiela, Robert (2016), From Secure Coding to Secure Software, available at : http://www.sei. cmu.edu/webinars/view_webinar.cfm?webinarid=483646 (last accessed February 1, 2017).
11. Putu, Adi and Guna, Permana (2015), "Scrum Method Implementation in a Software Development Project Management", *International Journal of Advanced Computer Science and Applications*, Vol. 6, No. 9, pp. 199–205.
12. Shvachich, G.G., Semenov, S.G. Glavchev, M.I. and Kassem, Khalife (2017), "Model' rascheta vremennykh granits proyektov razrabotki programmnogo obespecheniya", *Sistemi upravlínnya navígatsíí ta zv'yazku*, PNTU, Poltava, No. 1 (41), pp. 43-49.
13. Klieber, Williamand and Snavely, William (2016), Automated Code Repair Based on Inferred Specifications, available at : http://resources.sei.cmu.edu/asset_files/ConferencePaper/2016_021_001_483599.pdf (last accessed February 1, 2017).

Надійшла (received) 02.03.2017
Прийнята до друку (accepted for publication) 06.06.2017

Удосконалений спосіб
масштабування гнучкої методології розробки програмного забезпечення

С. Г. Семенов, Кассем Халіфе, М. М. Захарченко

Предметом вивчення статті є удосконалений спосіб масштабування гнучкою методології розробки програмного забезпечення. **Мета** - зниження потенційних втрат, зумовлених ризиками безпеки розробки та експлуатації програмного забезпечення на більшості етапів його життєвого циклу. Завдання: аналіз існуючих методологій і підходів розробки програмного забезпечення, дослідження можливостей масштабування методологій в рамках окремих проектів, удосконалення загальної схеми циклу розробки програмного забезпечення, розробка структури управління розробкою програмного забезпечення в рамках як окремо взятого проекту так і організації-розробника в цілому, розробка практичних рекомендацій щодо підвищення безпеки програмного забезпечення на різних етапах життєвого циклу. Використовуваними **методами** є: системний аналіз ризиків, причинно-наслідковий аналіз. Отримані наступні **результати.** Проведено аналіз існуючих гнучких методологій розробки програмного забезпечення, визначені перспективні напрямки і підходи даної індустрії, виявлені можливості масштабування гнучких методологій. Удосконалено схему життєвого циклу розробки програмного забезпечення, відмінною рисою якої є введення додаткових підрозділів і ролей, що мають на меті підвищення безпеки програмного забезпечення. Удосконалено структуру управління розробкою програмного забезпечення, що відрізняється від відомих урахуванням ризиків безпеки в процесі розробки. Розроблено практичні рекомендації використання вдосконаленого способу масштабування гнучкою методології. **Висновки.** Реалізація запропонованого вдосконаленого способу масштабування існуючої методології розробки ПО, відрізняється від відомих включенням і використанням в команді розробників додаткових фахівців безпеки. Це може спричинити деяке уповільнення виконання коду і збільшення кількості виявлених дефектів (багів) при альфа-тестування, а, отже, збільшення часу життєвого циклу багів. Однак в перспективі ці локальні погіршення дозволяють добиватися кращого кінцевого результату (підвищення безпеки розробленого ПО) і забезпечувати як швидке зростання функціоналу, так і прийнятний рівень якості сервісу. А це в свою чергу буде привабливим мотивом подальшої співпраці замовника і фірми-розробника.

Ключові слова: безпека програмного забезпечення, гнучкі методології розробки програмного забезпечення, Agile, Scrum.

Advanced method
of scaling the flexible methodology of software development

S. Semenov, Kassem Khalifeh, M. Zakharchenko

The **subject** of the article is an improved way to scale flexible methodology of software development. The goal is to reduce the potential losses caused by the security risks of software development and operation at most stages of its life cycle. **Objectives:** analysis of existing methodologies and approaches to software development, exploring the possibilities for scaling methodologies within individual projects, improving the overall design of the software development cycle, developing a software development management framework for both the individual project and the development organization as a whole, developing practical Recommendations to improve the security of software at various stages of the life cycle. The **methods** that are used: system analysis of risks, cause-and-effect analysis. The following **results** are obtained. The analysis of existing flexible software development methodologies has been carried out, prospective directions and approaches of this industry have been determined, and the opportunities for scaling flexible methodologies have been identified. The scheme of the life cycle of software development is improved, the distinctive feature of which is the introduction of additional subsections and roles aimed at increasing the security of software. The structure of software development management is improved, which differs from the known ones taking into account the security risks in the development process. Practical recommendations for using an improved method of scaling a flexible methodology have been developed. **Conclusions.** The implementation of the proposed improved method of scaling the existing software development methodology differs from those known by the inclusion and use of additional security specialists in the development team. This may entail some slowdown in code execution and an increase in the number of detected defects (bugs) during alpha testing, and, therefore, an increase in the life time of bugs. However, in the future, these local impairments can achieve a better end result (improving the safety of the developed software) and provide both rapid growth of functionality and an acceptable level of service quality. And this, in turn, will be an attractive motive for further cooperation between the customer and the developer.

Keywords: software security, flexible software development methodologies, Agile, Scrum.