Масштабирование гибкой методологии разработки программного обеспечения с учетом требований безопасности

Семенов С. Г., Кассем Халифе, Змиевская В. Н.
Кафедра вычислительной техники и программирования
Национальный технический университет «Харьковский политехнический институт»
Харьков, Украина
s_semenov@ukr.net
en_kh_kassem@hotmail.com
vitalinavtp@gmail.com

Scaling dynamic software development methodology with security considerations

Serhii Semenov, Kassem Khalifeh, Vitalina Zmievskaya
Deportment Computers and Programming
National Technical University "Kharkiv Polytechnic Institute"
Kharkiv
s_semenov@ukr.net
en_kh_kassem@hotmail.com
vitalinavtp@gmail.com

***Аннотация -п*роведен 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.**

Ключевые слова: безопасность программного обеспеяения, гибкие методологии разработки программного обеспечения, agile, scrum.

Abstract - 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. 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.

ВЕДЕНИЕ

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 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 минимизировать 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.

СХЕМА ЦИКЛА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Как отмечено выше, воспользоваться существующим опытом масштабирования, а так же разработать новые решения необходимо в условиях повышения требований к безопасности ПО. 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.

Следует заметить, что внедрение в команду на разных этапах разработки ПО специалиста кибербезопасности, безопасного программирования и особенно тестирования безопасности («этичного хакинга») [2, 13], является одним из требований современного рынка разработки ПО. Принципы такого внедрения на данном этапе развития индустрии разработки ПО 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 ПО.

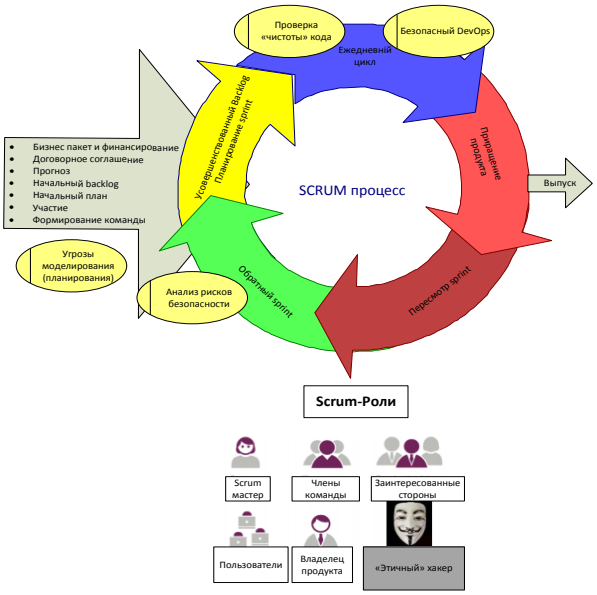

Рисунок 1 – Усовершенствованная схема цикла разработки ПО

СПОСОБ МАСШТАБИРОВАНИЯ ГИБКОЙ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Иллюстрация интервалов времени на этапах разработки, а также объемов выполнения указанных функций безопасности представлена на временной диаграмме 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 ПО 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 ПО, 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 ПО.

Выводы

В целом следует заметить, что реализация предложенного усовершенствованного метода 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.

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

Література References

[1] Barry W. Boehm and Richard Turner (2004) «Balancing Agility and Discipline - A Guide for the Perplexed», Addison-Wesley, 2004

[2] Demarko T., Lister T. (2005) Chelovecheskiy faktor: uspeshnyye proyekty i komandy. – SPb.: Simvol-Plyus, 2005. – 256s.

[3] Ruby S., Thomas D., Hansson D.H. (2013) Agile Web Development with Rails 4 / Pragmatic Programmers, LLC., 2013. — 439 p.

[4] Guru menedzhmenta kachestva i ikh kontseptsii: E.Deming, Dzh.Dzhuran, F.Krosbi, K.Isikava, A.Feygenbaum, T.Taguti (2001) / available at: http://www.management.com.ua/qm/qm009.html (last accessed June 10, 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=SecurityPractitionerPerspectiveonDevOpsforBuildingSecureSolutions (last accessed June 10, 2017).

[6] Highsmith J. (2006) Agile Software Development Ecosystems. – Boston: AddisonWesley, 2006. – 448p.

[7] Maykl Kuzumano, Meri Poppendik (2012) Berezhlivaya razrabotka programm // Otkrytyye sistemy. SUBD 2012 № 08/ available at: https://www.osp.ru/os/2012/08/13019237/ (last accessed June 10, 2017).

[8] Makhmetov G.Ye. (2017) Kogda «agile» (ne) k mestu / available at: https://makhmetov.ru/articles/agile.html (last accessed June 01, 2017).

[9] Mark Sherman (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 June 10, 2017).

[10] Mark Sherman, Robert Schiela (2016) From Secure Coding to Secure Software / available at: http://www.sei.cmu.edu/webinars/view_webinar.cfm?webinarid=483646 (last accessed June 10, 2017).

[11] Putu Adi Guna Permana (2015) 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] Shvachich G.G., Semenov S.G. Glavchev M.I., Kassem Khalife (2017) Model' rascheta vremennykh granits proyektov razrabotki programmnogo obespecheniya / Sistemi upravlínnya navígatsíÿ ta zv'yazku №1(41) 2017 s. 43-49.

[13] William Klieber, William Snavely (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 June 10, 2017).