Содержание:

Введение

Гибкая методология разработки (англ. *Agile software development*) — это концептуальный каркас, в рамках которого выполняется разработка программного обеспечения. Существует несколько подобных методик.

Методология «скрам» была впервые представлена в 1990-х годах Кеном Швабером и Джефом Сазерлендом в виде четко задокументированного и формализованного «руководства по «скраму».

«Скрам»

- это гибкий и легкий процесс управления и контроля программным обеспечением и процесса разработки продукта в быстро меняющихся условиях». Данная методология устанавливает определенные правила управления IT-проектом (разработкой или внедрением информационной технологии), которые основываются на возможности постоянной корректировки требований и на внесении тактических изменений.

Перед началом проекта участники обсуждают глобальные цели и базовую стратегию, направленную на достижение этих целей. Лицо со стороны Заказчика проекта представляет интересы своей компании путем описания бизнес-функций внедряемого или настраиваемого программного обеспечения. Со стороны исполнителя определяется состав команды разработчиков и другие ресурсы, после чего сторонами обсуждаются примерные рамки проекта, а так же дата начала первой итерации. Итерация в «скраме» получила название спринт, и она длится две-четыре недели.

В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из

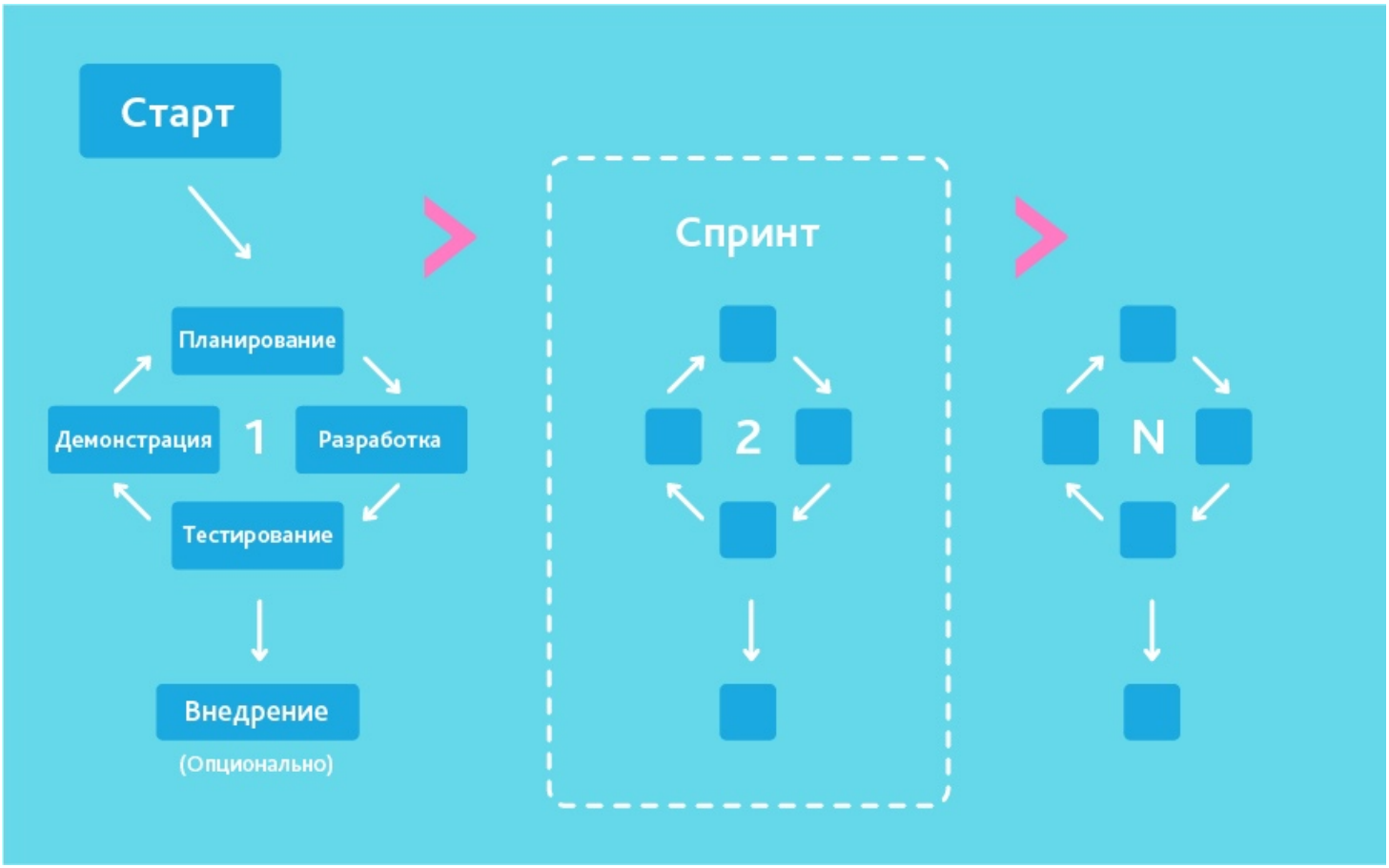
преимуществ гибкой модели. К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку. Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

В основе такого типа — непродолжительные ежедневные встречи — «Scrum» и регулярно повторяющиеся собрания (раз в неделю, раз в две недели или раз в месяц), которые называются «Sprint». На ежедневных совещаниях участники команды обсуждают:

1. отчёт о проделанной работе с момента последнего Scrum’а;
2. список задач, которые сотрудник должен выполнить до следующего собрания;
3. затруднения, возникшие в ходе работы.

Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка. Соответственно, в процессе реализации требования изменяются. Стоит вспомнить класс творческих людей, которым свойственно генерировать, выдавать и опробовать новые идеи еженедельно или даже ежедневно. Гибкая разработка лучше всего подходит для этого психотипа руководителей.

Методология разработки ПО – это система, определяющая порядок выполнения задач, методы оценки и контроля. Модели разработки ПО выбирают, исходя из направления проекта, его бюджета, сроков реализации конечного продукта, а также внимание стоит обратить и на характер и темперамент руководителя проекта и его команды.



Модель методологии «скрам»

состоит из трех главных компонентов: ролей, артефактов и мероприятий. В данной методологии присутствуют три роли: Владелец продукта, Скрам-мастер и Скрам-команда.

- Владелец продукта чаще всего является заказчиком, который поддерживает в актуальном состоянии список требований к проекту. Чем лучше и четче владелец продукта опишет требования, тем меньше будет вопросов у разработчиков, тем реже будет изменяться функционал программного обеспечения с течением времени. Именно Владелец продукта определяет приоритет требований, то есть сам заказчик решает, какие функции программного обеспечения являются наиболее срочными и важными для компании на какой-то конкретный момент;
- Скрам-мастер ответственен за понимание сущности всего «скрам»-процесса другими участниками. Скрам-мастер чем-то похож на традиционного менеджера проекта, но у него есть одно отличие: он не отдает явных приказов и не устанавливает сроки разработчикам, он, напротив, помогает им при

возникновении каких-либо трудностей и следит, чтобы их работа была слаженной. Именно от Скрам-мастера зависит атмосфера в команде разработчиков, а, как следствие, и их инициативность, удовлетворенность и общий результат. Скрам-мастер решает любые проблемы, которые могут как-то помешать команде, будь то сломанное оборудование или внешнее давление со стороны заказчика. Скрам-мастер также следит, чтобы все события «скрама» (о которых будет написано ниже) происходили регулярно и с максимальной эффективностью;
- Скрам-команда «…состоит из профессионалов, выполняющих работу по разработке потенциально «готового» к выпуску новой версии продукта в конце каждого «спринта. Обычно количество разработчиков и программистов не превышает восьми, и все они являются заинтересованными в успешной реализации IT-проекта. Перед началом каждой итерации Скрам-команда ставит достижимую и значимую для заказчика цель, а во время итерации направляет все свои усилия на достижение этой цели в срок с заявленным качеством. Достижение цели характеризуется наличием запланированного кода, который впоследствии был отлажен, а возможные дефекты итерации устранены. Скрам-команда сама устанавливает себе сроки (обсудив их со Скрам-мастером и Владельцем продукта), сама оценивает свои возможности и распределяет время.

Помимо спринта в методологии «скрам» существуют еще четыре мероприятия: планирование спринта, ежедневное совещание, обзор итогов спринта и ретроспективное совещание. Перед началом спринта происходит его планирование. Сначала Владелец продукта демонстрирует журнал продукта, в котором находятся требования к программному обеспечению, отсортированные по приоритету на конкретный момент. При этом Владелец продукта обсуждает со Скрам-командой возможные нюансы задач, тем самым, минимизируя возможность появления субъективизма в разработке. После этого команда выбирает наиболее важные задачи из журнала продуктов, которые помещаются в журнал спринта. Ключевым момент здесь является определение необходимого времени для выполнения этих задач самими разработчиками. Так как в данном случае они все решают сами, без внешнего принуждения от менеджера проекта, то вероятность к неоправданному завышению времени минимальна. После того, как задачи на следующий спринт назначены, Скрам-команда начинает между собой обсуждать каждую из них более подробно. Для успешного выполнения задачи, она может детализироваться понятным разработчикам образом на мини-задачи. Кроме этого, Скрам-команда обсуждает сам процесс разработки кода, взаимозависимость между

различными компонентами и так далее. Иными словами, разработчики решают, как они будут реализовывать требование заказчика.

Несмотря на «гибкость» методологии «скрам», команда должна быть уверена в неизменности и стабильности задач в рамках конкретного спринта. Если Владелец продукта хочет поменять требования, то для этого ему стоит дождаться планирования следующего спринта. Такой подход одновременно «гибок» и одновременно защищает проект от возможного беспорядка и разногласий. После окончания планирования начинается спринт. Каждый рабочий день Скрам-мастер организует короткое совещание, не превышающее по длительности пятнадцати минут. Целью ежедневного совещания является предоставление возможности участникам команды поделиться собственным прогрессом и возникшими трудностями. Каждый участник команды отвечает на три вопроса:

1. Что было сделано с момента прошлого такого совещания?
2. Что планируется сделать к следующему такому совещанию?
3. Какие существуют сложности, тормозящие процесс выполнения задания?

При возникновении каких-то проблем, Скрам-мастер делает все возможное, чтобы устранить их после окончания ежедневного совещания. Таким образом, пятнадцатиминутная встреча помогает держать в курсе участников Скрам-команды о том, что делают их коллеги и на какой стадии находятся задачи из журнала продукта. Владелец продукта тоже может присутствовать на этой встрече, однако никаких дискуссий между ним и разработчиками быть не должно: встреча организуется только для краткой отчетности

После завершения спринта проводится обзор его итогов, где демонстрируется основная функциональность, появившаяся в проекте за этот спринт. На этой встрече может присутствовать любой желающий. Важно отметить, что обзор итогов спринта не предполагает под собой организацию и планирование презентации, на деле должна быть непосредственная демонстрация того, что было сделано в прошедшем спринте. Владелец продукта определяет полноту и качество выполненного задания, тем самым налаживая обратную связь с разработчиками.

Впоследствии Владельцем продукта корректируется журнал продукта и делаются некоторые выводы касательно дальнейших сроков проекта. Результатами встречи являются скорректированный журнал продукта и планы на дальнейший спринт.

В целом говоря общий процесс работы скрам может быть представлен, как:

Заключение

Подводя итог, можно отметить, что благодаря постоянному анализу выполненной работы и возможностям осуществлять корректировки направления проекта между итерациями (спринтами) методология «скрам» позволяет более продуктивно достичь результатов и, следовательно, более качественно разработать программное обеспечение.