Курс лекций по дисциплине «Программная инженерия»

Лекция 11. "Гибкие" (agile) методы разработки

Ст. преподаватель кафедры ИС
Важдаев А.Н.

Общее

- "Гибкие" (agile) методы разработки ПО появились как альтернатива формальным и "тяжеловесным" методологиям, наподобие CMM и RUP. Талантливые программисты не желают превращения разработки ПО в рутину, хотят иметь максимум свобод и обещают взамен высокую эффективность. С другой стороны, практика показывает, что "тяжеловесные" методологии в значительном количестве случаев неэффективны. Основными положениями гибких методов, закрепленных в Agile Manifesto в 2007 году являются следующее:
- индивидуалы и взаимодействие вместо процессов и программных средств;
- работающее ПО вместо сложной документации;
- взаимодействие с заказчиком вместо жестких контрактов;
- реакция на изменения вместо следования плану.

Extreme Programming

- Самым известным гибким методом является Extreme Programming (известное сокращенное название – XP). Он был создан талантливым специалистом в области программной инженерии Кентом Беком в результате его работы в 1996-1999 годах над системой контроля платежей компании "Крайслер".
- Модель процесса по XP выглядит как частая последовательность выпусков (releases) продукта, столь частых, сколь это возможно. Но при этом обязательно, чтобы в выпуск входила новая целиковая функциональность.

Extreme Programming

- Планирование (Planning Game), основанное на принципе, что разработка ПО является диалогом между возможностями и желаниями, при этом изменятся и то и другое.
- Простой дизайн (Simple Design) – против избыточного проектирования.
- Метафора (Metaphor) – суть проекта должна умещаться в 1-2 емких фразах или в некотором образе.
- Рефакторинг (Refactoring) – процесс постоянного улучшения (упрощения) структуры ПО, необходимый в связи с добавлением новой функциональности.
- Парное программирование (Pair Programming) – один программирует, другой думает над подходом в целом, о новых тестах, об упрощении структуры программы и т.д.

Extreme Programming

- Коллективное владение кодом (Collective Ownership).
- Участие заказчика в разработке (On-site Customer) – представитель заказчика входит в команду разработчика.
- Создание и использование стандартов кодирования (Coding Standards) в проекте – при написании кода (создаются и) используются стандарты на имена идентификаторов, составление комментариев и т.д.
- Тестирование – разработчики сами тестируют свое ПО, перемежая этот процесс с разработкой. При этом рекомендуется создавать тесты до того, как будет реализована соответствующая функциональность. Заказчик создает функциональные тесты.
- Непрерывная интеграция. Сама разработка представляется как последовательность выпусков.
- 40-часовая рабочая неделя.

Scrum

- В 1986 японские специалисты Hirotaka Takeuchi и Ikujiro Nonaka опубликовали сообщение о новом подходе к разработке новых сервисов и продуктов (не обязательно программных). Основу подхода составляла сплоченная работа небольшой универсальной команды, которая разрабатывает проект на всех фазах. Приводилась аналогия из регби, где вся команда двигается к воротам противника как единое целое, передавая (пасуя) мяч своим игрокам как вперед, так и назад. В начале 90-х годов данный подход стал применяться в программной индустрии и обрел название Scrum (термин из регби, означающий - схватка), в 1995 году Jeff Sutherland и Ken Schwaber представили описание этого подхода на OOPSLA '95 – одной из самых авторитетных конференций в области программирования. С тех пор метод активно используется в индустрии и многократно описан в литературе. Scrum активно используется также в России.

Scrum


Создание требований к продукту
Планирова-ние итерации
Выполнение итерации (2-4 недели)
Анализ результатов, пересмотр требований

Scrum

- *Владелец продукта* (Product Owner) – это менеджер проекта, который представляет в проекте интересы заказчика. В его обязанности входит разработка начальных требований к продукту (Product Backlog), своевременное их изменение, назначение приоритетов, дат поставки и пр.
- *Scrum-мастера* (Scrum Master) обеспечивает максимальную работоспособность и продуктивную работу команды – как выполнение Scrum-процесса, так и решение хозяйственных и административных задач. В частности, его задачей является ограждение команды от всех воздействий извне во время итерации.

Scrum

- *Scrum-команда* (Scrum Team) – группа, состоящая из пяти–девяти самостоятельных, инициативных программистов. Первой задачей команды является постановка для итерации реально достижимых и приоритетных для проекта в целом задач (на основе Project Backlog и при активном участии владельца продукта и Scrum-мастера). Второй задачей является выполнение этой задачи во что бы то ни стало, в отведенные сроки и с заявленным качеством. Важно, что команда сама участвует в постановке задачи и сама же ее выполняет. Здесь сочетается свобода и ответственность, подобно тому, как это организовано в MSF. Здесь же "просвечивает" дисциплина обязательств.

Scrum

- *Sprint Planning Meeting*. Проводится в начале каждого Sprint. Сначала Product Owner, Scrum-мастер, команда, а также представители заказчика и пр. заинтересованные лица определяют, какие требования из Project Backlog наиболее приоритетные и их следует реализовывать в рамках данного Sprint. Формируется Sprint Backlog. Далее Scrum-мастер и Scrum-команда определяют то, как именно будут достигнуты определенные выше цели из Sprint Backlog. Для каждого элемента Sprint Backlog определяется список задач и оценивается их трудоемкость.

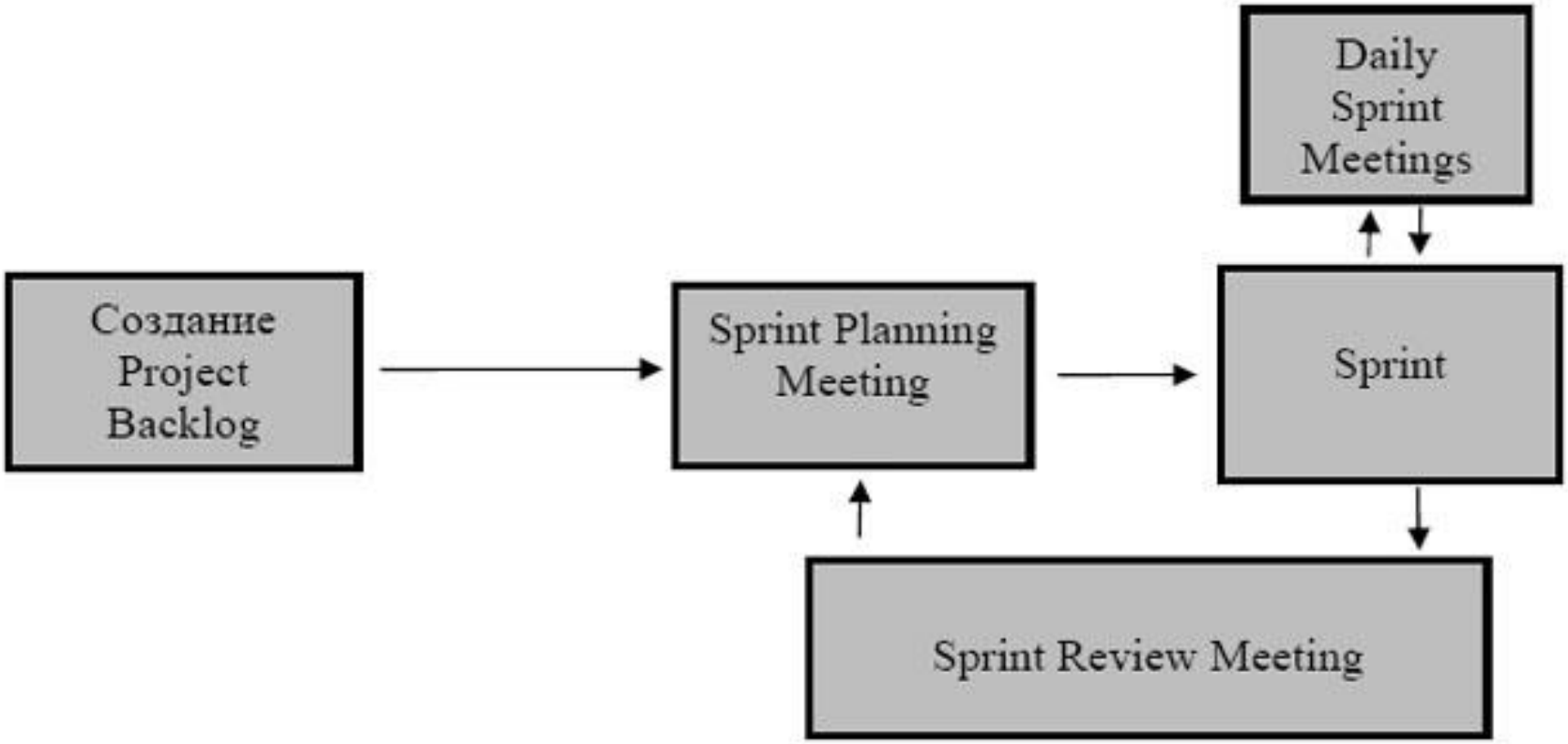
Scrum

- *Daily Scrum Meeting* – пятнадцатиминутное каждодневное совещание, целью которого является достичь понимания того, что произошло со времени предыдущего совещания, скорректировать рабочий план сообразно реалиям сегодняшнего дня и обозначить пути решения существующих проблем. Каждый участник Scrum-команды отвечает на три вопроса: что я сделал со времени предыдущей встречи, мои проблемы, что я буду делать до следующей встречи? В этом совещании может принимать участие любое заинтересованное лицо, но только участники Scrum-команды имеют право принимать решения. Правило обосновано тем, что они давали обязательство реализовать цель итерации, и только это дает уверенность в том, что она будет достигнута.

Scrum

- *Sprint Review Meeting*. Проводится в конце каждого Sprint. Сначала Scrum-команда демонстрирует Product Owner сделанную в течение Sprint работу, а тот в свою очередь ведет эту часть митинга и может пригласить к участию всех заинтересованных представителей заказчика. Product Owner определяет, какие требования из Sprint Backlog были выполнены, и обсуждает с командой и заказчиками, как лучше расставить приоритеты в Sprint Backlog для следующей итерации. Во второй части митинга производится анализ прошедшего спринта, который ведет Scrum-мастер. Scrum-команда анализирует в последнем Sprint положительные и отрицательные моменты совместной работы, делает выводы и принимает важные для дальнейшей работы решения. Scrum-команда также ищет пути для увеличения эффективности дальнейшей работы. Затем цикл повторяется.

Scrum


Создание
Project
Backlog
Sprint Planning
Meeting
Sprint
Daily
Sprint
Meetings
Sprint Review Meeting