УДК 004.413

ГИБКАЯ МЕТОДОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

© 2011 г. ***Д.В. Карпов***

Нижегородский госуниверситет им. Н.И. Лобачевского

dkarpov.career@gmail.com

Поступила в редакцию 20.01.2011

Освещаются основные вопросы концепции гибкой методологии разработки программного обеспечения, а именно: проблемы организации процесса разработки ПО, основные принципы методологии, применение практик, основанных на методологии, основные проблемы перехода на данную методологию.

Ключевые слова: гибкая методология, итерация, риск, программное обеспечение, управление проектом, команда проекта, программирование, требование, спринт, список задач, заказчик, скрам, скрам мастер.

Разработка программного обеспечения (ПО), как и любая другая техническая дисциплина, имеет дело со следующими основными проблемами: качество, стоимость и надежность. В связи с этим правильная организация процесса разработки программного обеспечения является основой достижения запланированного результата в ожидаемые сроки, с ожидаемым уровнем качества и с адекватным бюджетом.

Среди общераспространенных проблем процесса разработки программного обеспечения встречаются следующие:

1. Изменение требований непосредственно в процессе разработки.
2. Нечеткое распределение ответственности за выполняемую работу и ее результат.
3. Наличие непрерывного потока мелких, «быстрых», наваливающихся требований, отвлекающих разработчиков и менеджеров от основного направления работ.
4. Как следствие, срыв сроков, раздувание бюджетов, потеря качества.

Для решения задачи успешной организации процесса разработки ПО была создана гибкая методология разработки ПО.

Гибкая методология разработки (англ. Agile software development) – это набор принципов и правил, в рамках которого осуществляется разработка ПО.

Методология Agile – это семейство процессов разработки, а не единственный подход к разработке программного обеспечения [1–6]. Ценности и принципы Agile методологии закреплены в документе ‘Agile Manifesto’. Agile не включает конкретных практик, а определяет ценности и принципы, которыми руководствуются успешные команды.

Agile Manifesto разработан и принят 11–13 февраля 2001 года на лыжном курорте The Lodge at Snowbird в горах штата Юта, США. Манифест подписали представители следующих методологий:

- Extreme programming
- Scrum
- DSDM
- Adaptive Software Development
- Crystal Clear
- Feature-Driven Development
- Pragmatic Programming

Большинство гибких методологий нацелено на минимизацию рисков путём сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся одну-две недели. Каждая итерация сама по себе выглядит как программный проект в миниатюре и включает все задачи, необходимые для выдачи миниприроста по функциональности: планирование, анализ требований, проектирование, кодирование, тестирование и документирование. Хотя отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, подразумевается, что гибкий программный проект готов к выпуску в конце каждой итерации. По окончании каждой итерации команда выполняет переоценку приоритетов разработки.

Agile-методы делают упор на непосредственное общение лицом к лицу. Большинство

agile-команд расположены в одном офисе, иногда называемом bullpen. Как минимум она включает и «заказчиков» (англ. product owner). Это заказчик или его полномочный представитель, определяющий требования к продукту. Эту роль может выполнять менеджер проекта, бизнес-аналитик или клиент. Офис может также включать тестировщиков, дизайнеров интерфейса, технических писателей и менеджеров.

Основным результатом работы по agile-методологии является работающий программный продукт. Расценивая именно работающий программный продукт в качестве единственного показателя работы команды проекта за конечный период времени, создатели концепции agile сформулировали следующие ценности и принципы методологии.

Ценности Agile-методологии:

- личности и их взаимодействия важнее, чем процессы и инструменты;
- работающее программное обеспечение важнее, чем полная документация;
- сотрудничество с заказчиком важнее, чем контрактные обязательства;
- реакция на изменения важнее, чем следование плану.

Принципы Agile-методологии:

- удовлетворение клиента за счёт ранней и бесперебойной поставки ценного ПО;
- приветствие изменения требований, даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
- частая поставка рабочего ПО (каждый месяц или неделю или ещё чаще);
- тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
- проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
- рекомендуемый метод передачи информации – личный разговор (лицом к лицу);
- работающее ПО – лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределенный срок;
- постоянное внимание на улучшение технического мастерства и удобный дизайн;
- простота – искусство НЕ делать лишней работы;
- лучшие архитектура, требования и дизайн получаются у самоорганизованной команды;
- постоянная (частая) адаптация (улучшение эффективности работы) к изменяющимся обстоятельствам.

Существуют методологии, которые придерживаются ценностей и принципов, заявленных в Agile Manifesto. Одной из наиболее распространенных является методология разработки Scrum, которую, пожалуй, можно считать набором конкретных практик, используемых в процессе разработки ПО. Термин Scrum ассимилирован в русский язык без перевода.

Scrum (в переводе с англ. «митинг», «свалка», «драка») чётко делает акцент на качественном контроле процесса разработки.

Scrum – это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные небольшие промежутки времени (**спринты** от 2 до 4 недель) предоставлять конечному пользователю работающее ПО с добавленными возможностями, для которых определён наибольший приоритет. Требуемый к реализации функционал в очередном спринте определяется до его начала на этапе планирования и не может изменяться на всём протяжении спринта. При этом строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость. Scrum является одним из наиболее общераспространенных «последователей» гибкой методологии разработки ПО.

Данный подход впервые описали специалисты Хиротака Такеути и Икудзиро Нонака в 1986 г. Они отметили, что проекты, над которыми работают небольшие, кросс-функциональные команды, обычно систематически производят лучшие результаты, и объяснили это как «подход регби».

В рамках методологии Scrum на каждый спринт выделяется следующий **состав команды спринта**:

1. Скрам Мастер (Scrum Master).
2. Заказчик/Владелец продукта (Product owner).
3. Команда (Team).

Рассмотрим каждую группу участников более подробно.

1. **Скрам Мастер** (Scrum Master) – самая важная роль в методологии. Скрам Мастер отвечает за успех Scrum (как принципа организации работы команды) в проекте. По сути, Скрам Мастер является интерфейсом между менеджментом и командой. Как правило, эту роль в проекте играет менеджер проекта.

Основные **обязанности Скрам Мастера** таковы:

- создает атмосферу доверия,
- устраняет препятствия,
- делает проблемы и открытые вопросы видимыми,
- отвечает за соблюдение практик и процесса в команде.

Скрам мастер ведет ежедневное собрание команды спринта (Daily Scrum meeting) и отслеживает прогресс команды при помощи Списка задач спринта (Sprint Backlog), отмечая статус всех задач в спринте. Скрам мастер может также помогать Заказчику в создании списка задач спринта для команды.

2. **Заказчик/Владелец продукта** (Product Owner) – это человек, отвечающий за разработку продукта. Как правило, это менеджер продукта для продуктовой разработки, менеджер проекта для внутренней разработки и представитель заказчика для заказной разработки. Заказчик – это единая точка принятия окончательных решений для команды в проекте, именно поэтому это всегда один человек, а не группа или комитет.

Обязанности Заказчика таковы:

- отвечает за формирование видения продукта,
- управляет рентабельностью,
- управляет ожиданиями заказчиков и всех заинтересованных лиц,
- координирует и приоритизирует список задач спринта,
- предоставляет понятные и тестируемые требования команде,
- взаимодействует с командой и заказчиком,
- отвечает за приемку кода в конце каждой итерации.

Заказчик ставит задачи команде, но он не вправе ставить задачи конкретному члену проектной команды в течение спринта.

3. Команда (Team)

В методологии Scrum команда является самоорганизующейся и самоуправляемой. Команда берет на себя обязательства по выполнению объема работ на спринт перед Product Owner. Работа команды оценивается как работа единой группы. В Scrum вклад отдельных членов проектной команды не оценивается, так как это разваливает самоорганизацию команды.

Обязанности команды таковы:

- отвечает за оценку элементов баклога,
- принимает решение по дизайну и имплементации,
- разрабатывает софт и предоставляет его заказчику,
- отслеживает собственный прогресс (вместе со Скрам Мастером),
- отвечает за результат перед Product Owner.

Размер команды ограничивается размером группы людей, способных эффективно взаимодействовать лицом к лицу. Типичные размер команды – 7±2.

Команда в Scrum кроссфункциональна. В нее входят люди с различными навыками – разработчики, аналитики, тестировщики. Нет заранее определенных и поделенных ролей в команде, ограничивающих область действий членов команды. Команда состоит из инженеров, которые вносят свой вклад в общий успех проекта в соответствии со своими способностями и проектной необходимостью. Команда самоорганизуется для выполнения конкретных задач в проекте, что позволяет ей гибко реагировать на любые возможные задачи.

Для облегчения коммуникаций команда должна находиться в одном месте, Предпочтительно размещать команду не в отделенных друг от друга помещениях, а в одной общей комнате для того, чтобы уменьшить препятствия для свободного общения. Команде необходимо предоставить все необходимое для комфортной работы, обеспечить досками и флипчартами, предоставить все необходимые инструменты и среду для работы. От всех этих условий зависит результат спринта.

Рассмотрев основополагающие принципы гибкой методологии (и набора практик Скрам), необходимо прорезюмировать, что методология Agile означает:

- гибкость, адаптируемость, снижение рисков,
- масштабируемость, широта применения,
- ориентированность на эффективную командную работу,
- более вероятные и прогнозируемые сроки предоставления продукта заказчику.

Однако многие компании сталкиваются с серьезными проблемами в попытке перейти к реальному использованию Agile и, в частности, к Scrum.

Основные проблемы перехода на гибкие методологии:

1. **Непонимание роли руководства при внедрении методологии.** Переход на многие гибкие методологии подразумевает кардинальную смену задач и методов работы руководителей. В двух словах этот переход можно сформулировать как переход от управления к направлению, переход от приказов и указаний к

рекомендациям. Первая ошибка, которая допускается при таком переходе – подсознательное стремление сохранить за собой власть, ведущее к всё тому же управлению. Вторая не менее серьёзная ошибка прямо противоположна – неправильное понимание этого перехода может привести к устранению менеджеров от руководящих функций, когда руководитель перестаёт давать указания, но так и не начинает давать рекомендации. Роль руководителя при этом сводится к чисто формальным, секретарским функциям. Причиной этого может быть и неготовность руководителей давать аргументацию своему мнению (в старой модели им этого зачастую делать не приходилось) или боязнь невыполнения высказанных рекомендаций.

2. **Построение «системы», не обладающей необходимой гибкостью**. Отсутствие опыта работы по новой методологии ведёт к тому, что новый процесс внедряется по инструкциям, буква к букве, что ведёт к негибкости и бюрократизации.

3. **Начало внедрения не с «основ».** При переходе на новые методологии руководство преследует конкретные выгоды, которые должна обеспечить методология: высокая производительность, качество и т.п. При этом некоторые практики нового процесса более очевидно ведут к этим выгодам, а некоторые – совсем не очевидно. В связи с этим возникает соблазн внедрить сначала более «выгодные» практики, а потом уже остальные. При этом не учитывается, что некоторые практики являются базовыми по отношению к другим, и внедрение вторых без первых невозможно.

4. **Изменяются рабочие места, но не меняются привычки.** Провозглашение новых правил и следование этим правилам – это принципиально разные вещи. Недостаточное понимание новых идей **всеми членами команды** или слабое осознание выгод от перехода на новый процесс, слабая мотивация, отсутствие переходного периода с явно выраженными послаблениями в графике работ и количестве задач – это далеко не полный список ошибок, способных привести к тому, что новый процесс может соблюдаться лишь формально «для видимости», а в действительности саботироваться членами команды.

5. **Все измерять (собирать данные), но ни на что не реагировать. Бесконечный анализ ситуации, вместо непрерывных улучшений.** Большинство гибких методологий подразумевают сбор данных и обсуждение ошибок, совершённых на предыдущем этапе, перед началом следующего. Распространённые ошибки в этой сфере – сбор данных без их последующего анализа или неверная, слишком поверхностная интерпретация собранных данных.

6. **Обходиться без поддержки.** Отсутствие опыта работы команды по новой методологии и внедрение по букве инструкций таит много ошибок, неверных интерпретаций и недопонимания. Только участие в процессе перехода опытного инструктора, имеющего непосредственных опыт работы по новому процессу, может избавить команду от множества неверных поворотов и тупиков или в отдельных случаях даже от построения совершенно неверной методологии.

Список литературы

1. Thomas D., Hansson D. H. Agile Web Development with Rails. 2007. P. 17–64.

2. Cohn M. Succeeding with Agile: Software Development Using Scrum. 2008. P. 53–81.

3. http://en.wikipedia.org/wiki/SCRUM (дата обращения: 03.09.2010)

4. http://agilemanifesto.org/ (дата обращения: 04.09.2010)

5. http://citforum.ru/SE/project/scrum/ (дата обращения: 02.09.2010)

6. http://agileguru.ru/AgileWiki/Scrum (дата обращения: 02.09.2010)

AGILE METHODOLOGY OF SOFTWARE DEVELOPMENT

D.V. Karpov

The article highlights the concept of agile software development as a philosophy and a methodology; namely, organizing the software development process, the methodology's key points, the use of methodology-based practices, major issues and benefits around agile transition.

Keywords: agile methodology, iteration, risk, software, project management, project team, programming, requirement, sprint, backlog, Product Owner, Scrum, ScrumMaster.