Семейство Agile

Если в материальном производстве ведущую роль играют станки, машины, оборудование, на которых работает в общем-то довольно легко заменяемый персонал, то совершенно иная ситуация в индустрии программного обеспечения (ПО). «Разработка ПО — это, возможно, первый случай массового применения мозгов в качестве основного средства производства во всем производственном цикле... Во многом опыт "общего управления" не применим в разработке ПО именно из-за этой особенности: IT-специалисты носят свои "станки" в голове. Чтобы "модернизировать производство", нужно не новый завод строить, а непрерывно обучать сотрудников и постоянно охотиться за лучшими мозгами. Если попытаться "оптимизировать затраты" привычными методами, есть риск в одночасье потерять весь производственный потенциал: "мозги" встанут и уйдут, а оставшиеся столы и компьютеры сами по себе средствами производства не являются». Это пишет на своем сайте Александр Орлов, программист с десятилетним стажем, а ныне — консультант по вопросам управления командами разработчиков и построения карьеры в software-индустрии. Его сайт [102] называется «Клуб успешных менеджеров-программистов», насчитывает около 2000 подписчиков и содержит много полезной информации.

Действительно, разработка программного обеспечения — относительно новый, характерный для постиндустриального, информационного общества вид человеческой деятельности. Масштабы этой отрасли впечатляют. Вот оценки по состоянию на 2010 г. В США объем рынка IT превышает 500 млрд долл. и составляет более 5% ВВП. Ориентировочный размер мирового рынка IT — 1,6 трлн долл. Если принять, что доля разработки и сопровождения ПО составляет 25% всего объема рынка, то получаем сумму 400 млрд долл. Для справки: размер ВВП России — около 1,3 трлн долл. Таким образом, мировой рынок разработки и сопровождения ПО составляет примерно 30% российского ВВП.

Чем же отличается разработка ПО от, скажем, строительства дома? Первое отличие сформулировал А. Орлов. Программирование — творческая работа, в которой основным сред-

ством производства является целостная личность человека: сочетание его профессиональных знаний, способностей с индивидуальной этической и эмоционально-волевой сферой. Личностью невозможно управлять: малейшее давление — и следует иррациональная (ненаблюдаемая, неподвластная внешнему контролю) реакция, которая тут же сказывается на качестве и количестве рационально выполняемой работы. В программировании это известно как закон Тома Листера: если надавить как следует, люди не будут думать быстрее, но любить свою работу будут меньше.

Поставить людей на первое место — серьезный шаг, для осуществления которого потребуется вся ваша решительность. Отношение к людям как к ресурсам глубоко укоренилось в бизнес-сознании, его истоки берут начало в «Научном управлении» Фредерика Тейлора. При управлении фабрикой этот подход, возможно, и оправдывает себя. Однако для такой творческой и профессиональной работы, какой я считаю создание программных продуктов, он не годится.

М. Фаулер

Второе отличие — особенность создаваемого продукта, его нематериальность. Продуктом является программный код, представляющий последовательность «фраз» на языке программирования, которую «понимает» компьютер. Программный код — гораздо более гибкая и изменяемая вещь, чем материальное изделие. Программный код тиражируем, чего нельзя сказать о физическом продукте.

Методологи разрабатывают сложные системы, в которых есть весьма изменчивые и нелинейные компоненты — люди. При этом им как-то удается вообще не замечать эти компоненты и то воздействие, которое они оказывают на проектируемую систему. После некоторого размышления такое положение дел кажется абсурдным... После пяти лет мучительных поисков мне стало ясно, что в моем уравнении не хватает одной переменной — влияния на методологию такого понятия, как «человек». Теперь я считаю, что люди — это главный, первоочередной двигатель проекта.

А. Коуберн

Третье отличие заключается в специфике этапов проектирования и производства. При постройке дома сначала проект-

ная организация создает комплект чертежей, по которым потом строители возводят здание. В программировании этапы проектирования и производства неразделимы. Первоначально есть только описание того, что должна делать программа. Такие требования к будущей программе часто имеют вид сценариев, «историй»: как должна реагировать программа на те или иные действия пользователя. В течение периода создания программы заказчик может активно менять требования к ней по двум группам причин. Взаимодействуя с внешней средой, он может, во-первых, получать информацию, которая меняет его представление о том, что и как должна делать программа; во-вторых, получать информацию от разработчиков, например о трудностях реализации тех или иных функций. Такая ситуация немыслима при строительстве дома. Непрерывное изменение требований к создаваемому продукту — характерная особенность программирования.

Подробнее об этих отличиях можно почитать, например, у М. Фаулера [73]. Как правило, над одной программой трудятся целые коллективы разработчиков. И указанные особенности совершенно по-новому поставили перед ними задачу организации работы по созданию программных продуктов. По мере развития отрасли ПО появлялись различные технологии организации процесса разработки компьютерных программ. Как правило, такие методологии вырастали из среды программистов-практиков, без особого участия психологов, социологов, специалистов по управлению организацией.

Примерно к 2000 г. создалась ситуация, когда оформилось около десятка методологий разработки программного обеспечения, никак не связанных между собой, внешне непохожих друг на друга, но обладающих внутренним родством. Их объединяло понимание человека как главного звена процесса разработки и понимание коллектива разработчиков как системы. Результатом «объединительного» процесса стал Манифест *Agile* — короткий документ, созданный группой известных специалистов в области организации разработки программного обеспечения. *Agile* в переводе с английского означает «живой», «подвижный», «гибкий». Двенадцать принципов манифеста основываются на четырех исходных базовых положениях:

- люди и их взаимодействие важнее, чем процессы и средства;
- работающее ПО важнее, чем исчерпывающая документация;
- сотрудничество с заказчиком важнее, чем обсуждение условий контракта;
- реагирование на изменения важнее, чем следование плану.

Внимательный анализ манифеста *Agile* приводит к выводу: преодолеть проблемы организации разработки ПО можно на пути отказа от администрирования и рассмотрения процесса разработки в системной парадигме, т.е. в признании:

- свободной творческой активности как высшей ценности;
- изменчивости постановки задачи, продукта, команды;
- важности постоянных коммуникаций, обратных связей;
- важности постоянных связей с заказчиком.

Две наиболее популярные методологии семейства *Agile* — *XP* и *Scrum*. Вот некоторые правила и процедуры из этих методологий, касающиеся организации процесса разработки программного обеспечения.

***XP, Extreme Programming*, экстремальное программирование.** «XP стоит на четырех китах: Коммуникации, Обратной связи, Простоте и Смелости. На них основаны двенадцать практик, которым должны следовать проекты, использующие XP» [73]. В XP большую роль играет прямое общение, поэтому команда не должна быть разбита на несколько частей — внедрение XP в распределенной географически команде будет крайне рискованным мероприятием. По той же причине возможный размер команды ограничен сверху — числом в 10–15 человек. В проектной команде должен постоянно работать так называемый представитель заказчика — он обладает детальной информацией о необходимой функциональности, определяет приоритеты отдельных требований, оценивает качество создаваемой программы. В *XP* не рекомендуется тратить много времени на планирование; сам процесс планирования называется игрой (*planning game*). Архитектура программы должна быть максимально простой. *XP* не рекомендует проектировать в расчете на будущее раз-

витие программы. Причина — большая вероятность изменения требований.

Люди, качество взаимоотношений между нами, коммуникации — вот главный фактор успеха наших проектов. Именно это мы имеем в виду, когда говорим, что люди важнее процессов и стандартов. Сплоченные, внутренне мотивированные, самоорганизованные команды — основа нашей общей эффективности. Команды, которым не нужен менеджер. Мы верим, что каждый из нас личность, которой не нужен контроль. Мы стремимся, чтобы команда могла сама ставить себе задачи и сама оценивала свою работу.

Д. Войханский

В *XP* поощряется коллективное владение кодом. Увидев возможность улучшения в любом компоненте, разработчик может сделать эти улучшения вне зависимости от того, кто является основным разработчиком компонента. Возможные ошибки должны выявляться автоматическими тестами. Еще один «плюс» коллективного владения кодом — взаимная заменяемость, когда выбытие из процесса специалиста не парализует работу.

Парное программирование: за одним компьютером одновременно работают два человека, программируя по очереди. Такой прием ведет к активному обмену опытом, улучшению взаимопонимания, способствует нахождению более продуманных решений, уменьшает количество ошибок. По оценкам специалистов, это «самая противоречивая» практика *XP*, поскольку физическое присутствие рядом партнера может стать помехой индивидуальному творческому процессу. Продолжительность рабочей недели не должна превышать 40 часов. По сравнению с обычной практикой постоянных переработок в средне- и долгосрочной перспективе это повышает производительность проектной команды за счет уменьшения стресса и переутомления.

Scrum. Это слово в переводе с английского означает «схватка вокруг мяча» — ключевой момент в регби. За продукт отвечает владелец продукта. Он определяет концепцию продукта и контролирует его функциональность: перечень требуемых свойств, расставленных по приоритетам. Приоритеты обсуждаются с группой заинтересованных лиц, включая

команду. Эти люди формируют список свойств (функций), подлежащих реализации в следующем спринте (итерации). Спринт — промежуток времени, обычно месяц, в течение которого команда работает над очередной готовой версией продукта с новыми функциями, которых не было в предыдущей версии. Спринт заканчивается демонстрацией такой версии, которая служит отправной точкой для планирования следующего спринта с новым приростом функциональности. В методологии Scrum всего три роли: скрам-мастер, владелец продукта и команда.

Скрам-мастер — самая важная роль в методологии. Он отвечает за соблюдение командой практик гибкой разработки и помогает команде придерживаться принятых ею решений. Важно подчеркнуть, что скрам-мастер не раздает задачи членам команды. Он обязательно должен быть членом команды. Как правило, он совмещает работу скрам-мастером с проектной ролью (как разработчик, тестировщик, аналитик и т.д.). Его обязанности — создание атмосферы доверия, вскрытие проблем, участие в митингах в качестве фасилитатора (модератора), устранение внешних препятствий.

Владелец продукта отвечает за разработку продукта. Владелец продукта — это единая точка принятия окончательных решений по продукту. Владелец продукта ставит задачи команде, но он не вправе ставить задачи конкретному члену проектной команды. Владельцем продукта может быть представитель заказчика.

В методологии *Scrum* команда является самоорганизующейся и самоуправляемой. Команда берет на себя обязательства по выполнению объема работ на спринт перед владельцем продукта. Работа команды оценивается как работа единой группы. В *Scrum* вклад отдельных членов команды не оценивается, так как это разваливает самоорганизацию команды. Размер команды ограничивается размером группы людей, способных эффективно взаимодействовать лицом к лицу. Идеальный размер команды — 7 ± 2. Команда в *Scrum* кросс-функциональна. В нее входят люди с дополняющими навыками — разработчики, аналитики, тестировщики. Нет заранее определенных и поделенных ролей. Команда сама решает, кто и что будет делать в течение спринта, для принятия таких решений есть специальные приемы.

Важнейшей практикой этой методологии является короткое ежедневное утреннее совещание, которое так и называется — «скрам» (*Daily Scrum Meeting*). Оно предназначено для того, чтобы все члены команды знали, кто и чем занимается в проекте, у кого какие проблемы. Длительность этого совещания строго ограничена и не должна превышать 15 мин. Цель совещания — поделиться информацией. Оно не предназначено для решения проблем в проекте. Все требующие специального обсуждения вопросы должны быть вынесены за его пределы.

Вот несколько отечественных сайтов по *Agile*: [93, 94, 97]. Понятно, что гибкие методологии имеют свою область применения. Они хороши для относительно небольших команд и для задач, когда требования к программе не могут быть детально сформулированы и меняются по ходу разработки. Чем больше профессиональная квалификация и опыт членов команды, тем неуместнее попытки администрирования.

Виктор: А как много компаний используют эти гибкие методологии?

— Ты задал провокационный вопрос. Поскольку я не специалист в этой области, пусть за меня ответит Асхат Уразбаев: «*Scrum* и другие разновидности гибких методологий используют практически все продуктовые компании на Западе, в том числе и такие крупные и традиционно менее склонные к изменениям организации, как Oracle и Microsoft» [97].

Но, Виктор, разве распространенность имеет значение? Вспомни историю с коперниканской революцией из гл. 6. Представь, что ты — Галилей на допросе у церковников. И они тебя спрашивают: «А как много людей поддерживают гелиоцентрическую систему мира?»

Виктор: Если я понял, уверен и мне очевидно, что Земля вращается вокруг Солнца — тогда мне неважно, сколько людей поддерживают такую точку зрения.

— То-то же. И еще одна мысль. Здесь опять работает механизм самоисполняющегося пророчества. Если ты веришь в такие подходы и начинаешь действовать в соответствии с ними, то они могут начать работать. Но если не веришь, но по каким-то причинам пытаешься им следовать, то точно ничего не получится.

Виктор: Слушай, Евгений. А можно эти *Agile* перенести из сферы разработки программного обеспечения в другие предметные области?

— Хороший вопрос. Я полагаю, что можно. Но это — тема для другой книги.
