Аналитик в Agile:
архаизм или необходимость?

Бибичев Андрей
(andrew@custis.ru)

22 октября 2008 года

I. Введение

Гибкие методологии набирают популярность в России и странах СНГ. Как правило, при рассмотрении возможности перехода на них (и даже в процессе перехода или начального использования), почти неизбежно встает целый ряд вопросов, если не сказать, сомнений. Наиболее частые из них:

1. Как быть с fix-price контрактами в Agile?
2. Какова роль менеджера в Agile, и как эта роль соотносится с понятием Product Owner?
3. Нужны ли аналитики в Agile, и если да, как должно быть организовано взаимодействие с ними?

Что касается первых двух вопросов, то ответы на них можно найти в великолепных презентациях двух главных популяризаторов методологии Scrum — Джефа Сазерленда (Jeff Sutherland) и Хенрика Книберга (Henrik Kniberg):

- Money for Nothing and Your Change for Free: Agile Contracts;
- The Manager's Role in Scrum.

Кроме того, весьма полезным в этом смысле может оказаться и знакомство с докладом Стаси Бродерик (Stacia Broderick), в котором она объясняет, как перевернуть с головы на ноги классический треугольник сроки/объем/бюджет:

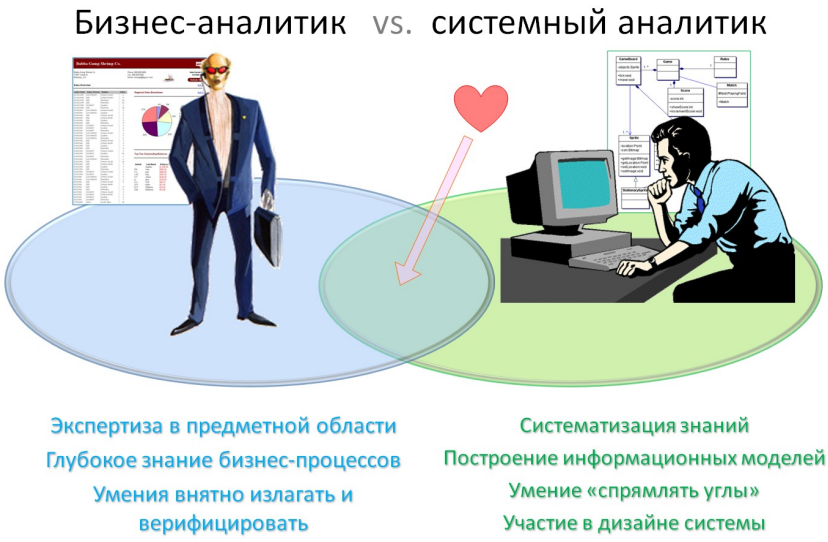
- Introduction to Agile for Traditional Project Managers.

А вот найти готовый ответ на третий вопрос, как ни странно, не так-то просто. Данный материал призван попытаться хотя бы частично заполнить этот пробел.

II. Необходимые оговорки

Бизнес- и системные аналитики

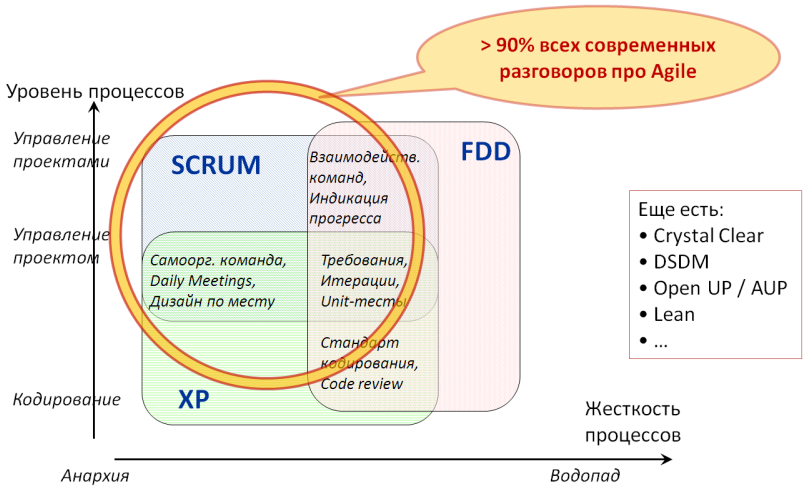
Достаточно распространенным является деление аналитиков на бизнес-аналитиков и системных аналитиков. Первые больше концентрируются на вопросах и тонкостях функционирования автоматизируемого бизнеса, являются экспертами в предметной области, но далеки от каких-либо системных «заморочек». А вторые — на системном анализе, построении информационных моделей, имеют представление об ограничениях и особенностях используемых технологий, участвуют в архитектурном дизайне.



В данном докладе столь четкого деления не используется и везде употребляется термин «аналитик» без дополнительных уточнений. При этом в большинстве случаев предполагается, что это, скорее, бизнес-аналитик, но которому не чужды системное мышление и примерные представления о технологических особенностях разрабатываемой системы, т.е. кроссфункциональный аналитик (что вполне в духе Agile) с уклоном в бизнес-область.

«Мы говорим Agile, подразумеваем Scrum»

Сейчас методология Scrum настолько популярна и распространена, что в большинстве современных докладов и статей, посвященных Agile, как правило, основное внимание уделяется именно Scrum. Это похоже на доминирование Internet Explorer-а, которое было на рынке Web-браузеров несколько лет назад.



Перефразируя известный лозунг о партии и Ленине: «Мы говорим Agile, подразумеваем Scrum! Мы говорим Scrum — подразумеваем Agile!»

Данный материал не является исключением из этой тенденции, предполагая использование именно Scrum в качестве Agile-методологии. Но следует понимать, что это не является серьезным ограничением на применимость описанных подходов в сочетании с другими гибкими методологиями — такими как eXtreme Programming (XP), Feature

Driven Development (FDD), Open Unified Process (OpenUP) и т.п. — изменятся только терминология и некоторые несущественные нюансы.

Читателя, недостаточно знакомого с методологией Scrum, хочется адресовать к широко известному в узких кругах практическому введению Хенрика Книберга:

- Scrum and XP from the Trenches.

На чем основан материал

В основу легли отнюдь не виртуальные эксперименты или философские разговоры в курилке, а реальный опыт перевода целой компании на методологию Scrum. Подробная информация об этом опыте была представлена в докладе на конференции РИТ-2008:

- Практика внедрения Scrum: трудности и пути их преодоления (презентация);
- Практика внедрения Scrum: трудности и пути их преодоления (статья).

III. Мифы Agile

Нам не дано предугадать,
Как слово наше отзовется.

Эти две строки из стихотворения Тютчева как нельзя лучше характеризуют ситуацию вокруг многих тезисов и принципов, декларируемых как в рамках Agile в общем, так и в рамках отдельных представителей гибких методологий, таких как Scrum, XP или Lean. Вот некоторые из них:

- наиболее эффективные команды — это кроссфункциональные команды;
- вначале итерации команде не должно требоваться наличие подробных спецификаций для решения поставленных задач;
- не следует создавать «лишние» артефакты (например, write-only документацию): работающий программный код — единственный артефакт, необходимость которого не вызывает сомнений (а XP почти полностью отрицает необходимость ведения внутренней проектной документации);
- у разработчиков должна быть возможность общаться с пользователями и представителями бизнеса напрямую, т.е. без лишних посредников (а в XP еще требуется присутствие представителя заказчика непосредственно в команде разработчиков).

Список можно продолжать, но даже этого достаточно, чтобы мнение типа «аналитику нет места в Agile, либо он должен уметь программировать» получило столь широкое распространение, что с этим уже приходится считаться.

Очень часто приходится слышать вопросы:

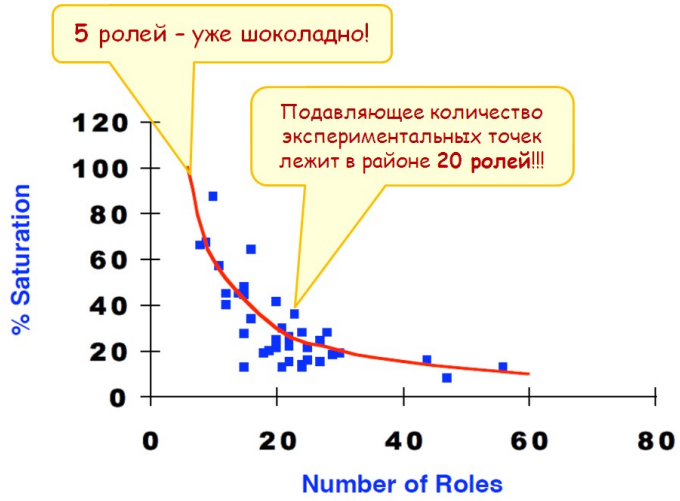
- И что, при переходе на Scrum нам нужно учить аналитиков программировать?
- Product Owner — это руководитель проекта, выполняющий еще и функции аналитика?
- Теперь вообще не следует предварительно прорабатывать задачу или требование перед постановкой их реализации в итерацию?

Давайте попробуем разобраться по порядку:

Кроссфункциональность

По сути, это тот горизонт, к которому, по возможности (и в рамках разумного), следует стремиться, но который вряд ли когда-либо будет достигнут.

Когда идеологи Agile говорят о кроссфункциональности, нужно понимать, чему они ее противопоставляют:

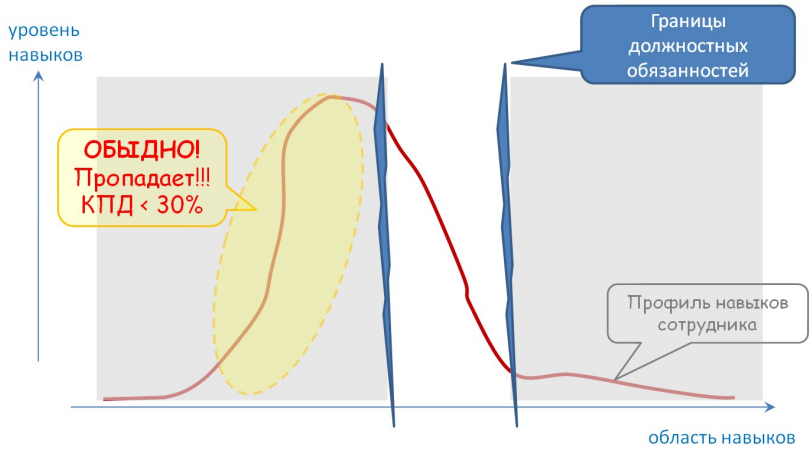


Источник: *Organizational Patterns of Agile Software Development by Coplien and Harrison (2004)*

На этом графике приведены результаты исследования зависимости условной комфортности командной работы от количества ролей в команде. Видно, что большинство экспериментальных точек лежит в окрестности 20 ролей! Двадцати! При этом весьма благоприятные условия наступают уже при количестве ролей, меньшем 10. А если в команде около 5 ролей — то это почти безграничное счастье каждого члена команды.

Так что пропаганда кроссфункциональности — это попытка объяснить и донести до широких слоев, что не должно быть *избыточного* или *искусственного* деления по ролям, хотя при этом наличие разумной специализации и различия в квалификации вовсе не отрицаются.

Здесь весьма важными являются выделенные слова: например, если требуется участие аналитика в ручном тестировании пользовательского интерфейса, то аналитик не должен от этого отказываться только на том основании, что это не входит в его обязанности:



Помните о золотой середине!

Нет подробным спецификациям

Это противопоставление водопаду (waterfall) и тяжеловесным (heavy-weight) процессам, в которых сильна вера в магию подробных технических заданий и формальных спецификаций на компоненты системы: умные люди долго думают и тщательно пишут хорошие и подробные бумажки, после дело остается за малым — перевести это всё в машинный код при помощи труда кодеров (даже не программистов, и уж тем более не кроссфункциональных разработчиков).

Эта магия настолько часто не срабатывает, что в большинстве случаев даже в успешных проектах разработчики (если таковые оказываются среди кодеров) почти не используют эти технические задания и спецификации или настолько отходят от них (для того чтобы система оказалась жизнеспособной), что ценность этих самых вымученных документов сводится к нулю.

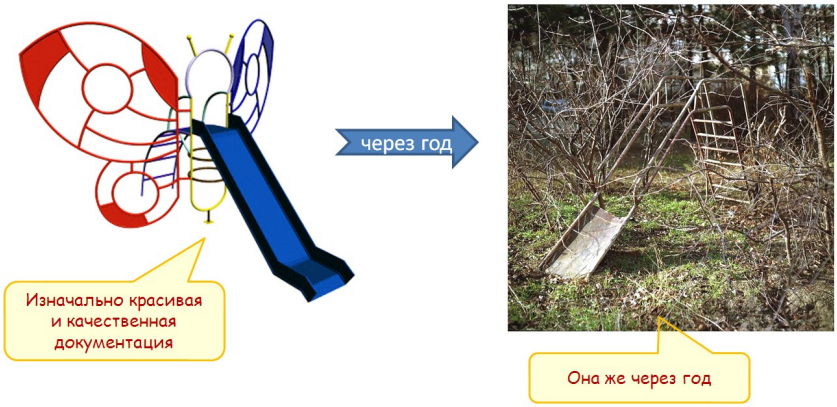
В спецификациях есть еще одна уязвимость: чем более подробными и/или низкоуровневыми они оказываются, тем меньше шансов, что заказчик или потенциальный пользователь смогут их прочитать, причем о понимании и верификации, как правило, и речи нет.

Однако, это не повод совсем не прорабатывать задачу или требование перед тем, как ставить их в итерацию. Опять же, важен разумный компромисс: нужно проработать на столько, чтобы команда в рамках итерации могла сконцентрироваться на реализации соответствующей функциональности, а не бесконечных выяснениях «а что же, собственно, нужно», при этом могут произойти какие-то уточнения и даже небольшие изменения в начальном в**и**дении (vision) задачи.

Минимум документации

С развитием интернета и блогосферы всё больше ощущается проблема не с писателями, а с читателями — пишут миллионы, а читают единиц. С проектной документацией во многом часто та же ситуация — если её много, то вряд ли её кто осилит целиком, а если и найдутся уникумы, то весьма сомнительно, что они смогут всё должным образом усвоить, отличив при этом главное от второстепенного.

Кроме того, проектная документация имеет тенденцию к очень быстрому устареванию, если не сказать «протуханию» — в особенности в сочетании с Agile, где изменения — норма, а не форс-мажор. Это усугубляется тем обстоятельством, что консистентность документации, в отличие от работающей системы, очень тяжело проверить.



В связи с этим, в Agile-методологиях всячески советуется минимизировать документацию:

- избегать write-only документации, т.е. той, которую заведомо никто не прочитает;
- писать лаконично, выделяя главное и опуская излишние детали, ибо детали меняются чаще;
- использовать больше визуальных образов, схем, графических нотаций.

Но это вовсе не значит, что минимизировать надо все и до нуля. При полном отсутствии документации, скорее всего, возникнут следующие проблемы:

- тяжело вводить новых людей в курс проекта;
- велика вероятность утери общей концепции и видения (как проекта в целом, так и отдельных его частей);
- сложно осуществлять контроль качества, так как не понятно с чем сверять (в особенности это касается полнофункционального регрессионного тестирования, которое полностью автоматизировать очень тяжело);
- сложно сопровождать и развивать старую функциональность, так как все уже забыли почему и зачем именно так сделали;
- и т.д.

Если они не возникли, считайте, что вам повезло. Но если вы думаете о будущем проекта — лучше не полагаться на везение.

Общение разработчиков с пользователями

Такую возможность действительно нужно обеспечить, т.к. это бывает очень полезно:

- повышает качество (меньше искажений в передачи информации, выше мотивация при прямом контакте с пользователями);
- выше вероятность сделать то, что нужно, а не то, что просили («спасибо, вы сделали то, что я просил, но это не то, что мне нужно»);
- ускоряет процесс (короче цепочка передачи информации).

Как обычно, в этой идиллии есть существенные «но». Во-первых, разработчики больше обращают внимание на техническую шелуху, а пользователи и представители заказчика — на бизнес-шелуху, из-за чего им иногда тяжело договориться, а зачастую и понять друг друга. Во-вторых, очень часто встречается ситуация, которую принято описывать термином «вязкий заказчик»: на уточнение и прояснение требований приходится тратить большое количество усилий и времени. Как правило, это связано с большой бюрократизацией у заказчика или с экстремально высокой занятостью ключевого

персонала у заказчика, а ведь всякие нетривиальные моменты и важные решения приходиться выяснять и согласовывать именно с ключевыми фигурами в бизнес-процессе, а не линейными исполнителями, которые редко когда видят дальше своих повседневных рутинных обязанностей.

В связи с этим возникает идея, чтобы был кто-то, помогающий разработчикам и пользователям находить общий язык и договариваться, а кроме того, преодолевать вязкость среды своими личными усилиями и навыками.

IV. Возможные функции аналитика

Итак, в Agile нет антагонизма к роли аналитика, как таковой. Но действительно ли аналитик может быть полезен и в чем именно?

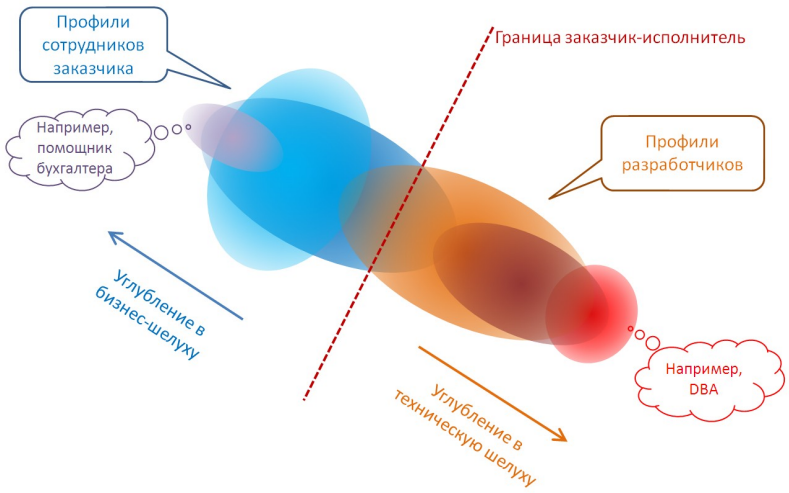
Связующее звено между разработчиками и заказчиками

В отличие от классической интерпретации функций аналитика, в Agile именно обеспечение эффективной связи между заказчиками (пользователями) и командой разработчиков играет, по сути, ключевую роль. Основные причины описаны в предыдущем разделе.

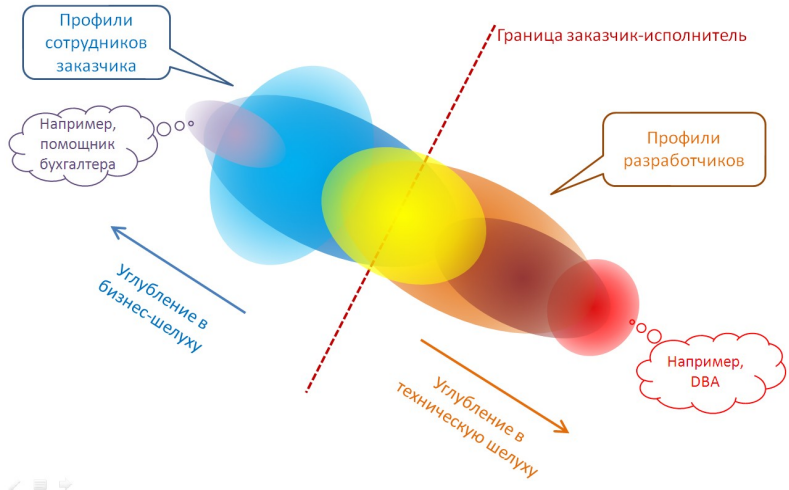
Т.е. аналитик оказывается тем парнем (на самом деле, чаще девушкой), которому(ой) доверяют и пользователи, и разработчики:

- Если возникают проблемы в формулировании или интерпретации требований, то необходимо, чтобы кто-то организовал общее совещание, на котором бы оказались все вовлеченные стороны (и от разработки, и от бизнеса), при этом нужно управлять ходом дискуссии, помочь сформировать общее решение и сформулировать его таким образом, чтобы оно было понятно всем заинтересованным лицам.
- Если пользователи страстно хотят одного, а разработчики упрямо настаивают на другом, то кто-то должен помочь найти компромисс или убедить разработчиков сделать так, как просят заказчики и пользователи.
- Если для решения задачи требуется прояснить ряд существенных деталей, а представители заказчика уходят в тину или устраивают круговую поруку (один ссылается на другого, тот на третьего и так далее, пока не замкнется в круг) или противоречат друг другу, то нужен кто-то, кто сможет эффективно выйти из этой ситуации.
- Если заказчики активно используют свой внутренний жаргон (бизнес), а разработчики — свой (технический), то нужен кто-то, кто сможет быть переводчиком.

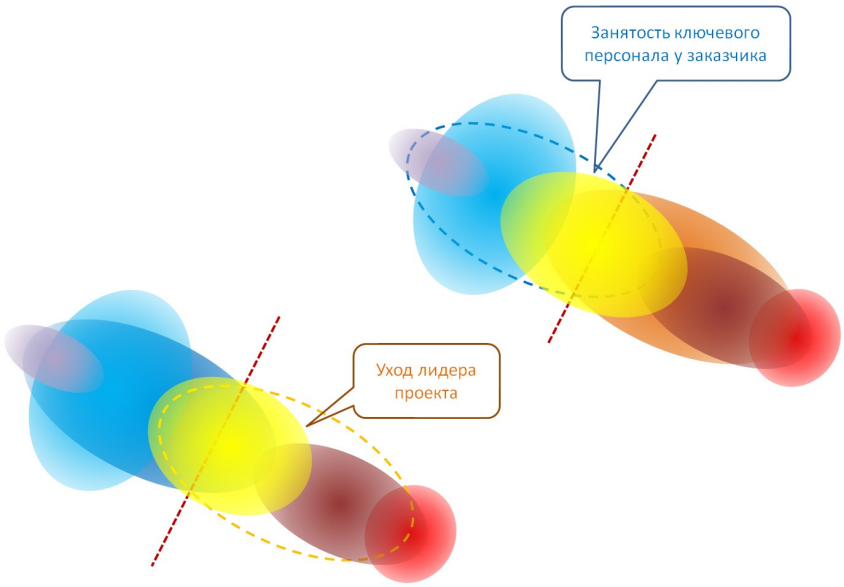
В большинстве случаев этот кто-то — это аналитик, которому помогают менеджеры, когда требуется административный ресурс, и ключевые эксперты проекта или даже компании, когда требуется генерация или верификация нестандартных решений.



Взаимное проникновение зон компетенций и интересов персонала со стороны заказчика и со стороны исполнителя



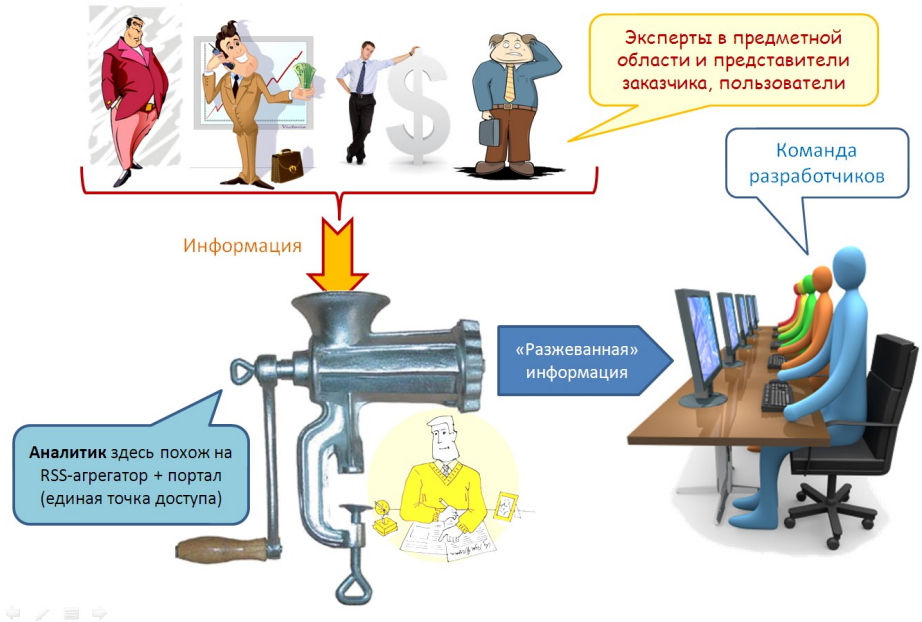
Видно, что аналитик (желтое «пятно») – хорошее усиление «взаимного проникновения», а значит и связи



А в подобных ситуациях аналитик оказывается жизненно необходим

Экспертиза в предметной области

Это достаточно стандартная функция для бизнес-аналитиков. Даже если у заказчика есть внятные эксперты в предметной области, умеющие доходчиво изложить свои знания, полезно иметь в команде (или «рядом» с ней) кого-то, кто будет накапливать эту экспертизу. Это позволит не зависеть от внешних экспертов при работе с другими заказчиками на том же вертикальном рынке или при переходе от штучного заказного решения к тиражируемому (коробочному), ведь, как известно, зачастую покупается не продукт, а экспертиза и опыт.



Систематизация и построение моделей

В Agile задачи систематизации представлений и требований, построение моделей предметной области и отдельных элементов системы принято решать при помощи группового интеллекта — на общекомандных дизайн-сессиях, сессиях планирования и т.п. И это оказывается очень эффективным — во время таких общих обсуждений рождаются наиболее адекватные модели и качественный дизайн. Поэтому аналитик в Agile не обязан уметь делать это самостоятельно, но практика показывает, что если аналитик способен предложить «начальное приближение», то скорость и эффективность общего обсуждения резко возрастают. В отличие от «водопада» такая ситуация достаточно комфортна — не страшно сделать ошибку или что-то не учесть — коллеги помогут это исправить.

Если же аналитик не обладает достаточным уровнем системности мышления или навыками моделирования (составления моделей предметной области), то эти функции возьмет на себя команда, возможно даже с привлечением экспертов из других проектов.

Контроль качества

Традиционно считается, что контролем качества занимаются специальные люди (или роботы?), которые выполняют приемочные испытания по описанным методикам и программам. В Agile говорят, что этого не достаточно и к этому, как минимум, прибавляют:

- регулярные, 1–2 раза в месяц, демонстрации представителям заказчика для получения неотложной обратной связи;
- unit-тесты для максимальной автоматизации и упрощения процесса регрессионного тестирования, локализации возникающих ошибок, формализации спецификаций на части системы до уровня проверяющего кода;
- непрерывную интеграцию для раннего обнаружения проблем.

Это всё здорово, полезно, действительно повышает качество и минимизирует некоторые риски. Однако кто проверит, что сделали то, что нужно, и что пользоваться удобно? Пользователи и заказчики на демонстрации — это, конечно, вариант, но, во-первых, не факт, что среди них окажутся заинтересованные именно в этой функциональности, а во-вторых, так можно потерять лицо (демонстрация превратиться в сессию тестирования и отладки, причем на глазах у изумленной публики).

Генрих Книберг в своей презентации «10 способов напортачить со Scrum и XP» говорит о критерии «сделано» (DoD, Definition Of Done). Согласно этому критерию, задача не считается до конца выполненной до тех пор, пока соответствующая функциональность не пройдет функциональные тесты и пока Product Owner не согласится с тем, что сделано то, что нужно. Т.е. получается, что Product Owner участвует в контроле качества — когда команда считает, что задача успешно решена, в этом должен еще убедиться и Product Owner.

Достаточно очевидно, что вместо (или совместно с) Product Owner-а (ом) эта почетная обязанность может быть возложена на аналитика.

При этом лучше не втягивать аналитиков в ручное регрессионное тестирование или другие рутинные процедуры — только в проверку правильности реализации новой функциональности.

Участие в пилотных внедрениях

Во время первых, «пилотных» внедрений системы, когда процедура еще не отлажена до формальных регламентов, возникает много проблем и нетривиальных трудностей:

- обучение пользователей при нехватке или даже отсутствии учебных материалов и инструкций;

- помощь пользователям в освоении системы («живая справка»);
- исправление ошибок, возникших в результате некорректных действий неопытных пользователей или сбоев в системе;
- фиксация всех узких мест, дополнительных пожеланий и требований, ошибок и неточностей (дабы донести до разработчиков и Product Owner-а);
- перегрузка и выверка данных (в особенности справочников);
- начальная настройка системы.

Привлечение аналитика к их решению существенно помогает, так как это человек, который хорошо знаком не только с пользователями и характером их работы, но и с самой системой.

VIP-сопровождение

Если у вас есть VIP-клиенты (крупные заказчики, первые внедрения и т.п.), то к процессу сопровождения и внедрения почти неизбежно придется привлекать высококвалифицированный персонал. Ведь у таких клиентов и проблемы нестандартные, и пожелания сложные, а bug-репорты зачастую выливаются в новый пласт функциональности или объемные доработки — да, грань между feature request и bug-ом в подобной ситуации почти условна.

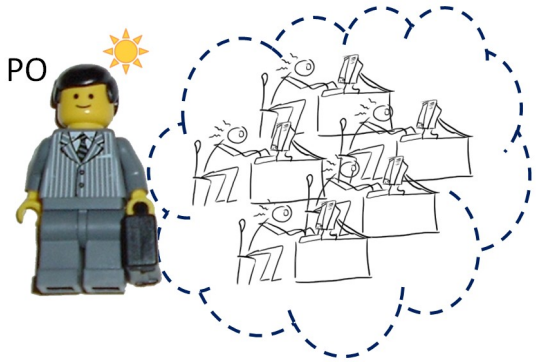
И здесь «опять на помощь приходят они»: те, кто хорошо знает и бизнес, и систему и к тому же умеет понимать пользователей и объяснять разработчикам.

V. Схемы взаимодействия аналитик – команда

В Agile большое внимание уделяется командной работе, самоорганизации команды (уход от микро-менеджмента). Как организовать взаимодействие аналитика с командой с учетом возлагаемых на него функций? Однозначного ответа на этот вопрос нет. Существуют разные варианты, причем в зависимости от обстоятельств эффективными оказываются те или иные. Ниже рассмотрены те схемы, с которыми приходилось сталкиваться на личном опыте.

Product Owner – аналитик

Это самый простой и очевидный случай. Product Owner отвечает за продукт, за сбор и приоритизацию требований, является своеобразным представителем заказчика, но на стороне исполнителя, отвечает или помогает ответить на уточняющие вопросы. Всё это тесно переплетается с функциями аналитика, обсуждаемыми выше.



Так что можно решить, что функции аналитика исполняет Product Owner. Или, если угодно, наоборот — аналитик исполняет роль Product Owner-а.

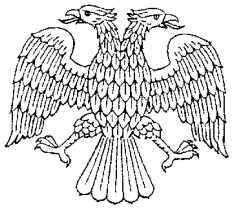
Среди плюсов такой схемы: простота, минималистичность, относительное удобство и для заказчика, и для разработчиков — и те и другие всегда знают к кому именно нужно обратиться со своими вопросами.

Недостатки:

- Получается, что слишком многое зависит от одного человека — большая нагрузка и связанные с этим риски «бутылочного горлышка».
- Экстремальная незаменимость — а если в отпуске, а если заболел (дай Бог ему здоровья!).
- Ситуация на рынке труда такова, что и аналитика не просто найти, и Product Owner-а тоже. А и то, и другое в одном лице — на порядки сложнее.
- Вряд ли получится плотно привлечь такого Product Owner-а к сопровождению системы или активному участию в пилотных внедрениях.
- Есть вероятность, что Product Owner будет откладывать решение каких-то задач не потому, что у них низкий приоритет, а потому, что он пока еще не успел как следует проработать постановку. Т.е. убивается вся идея приоритизации работ на основании потребностей бизнеса, а не исходя из внутренних или технических обстоятельств.

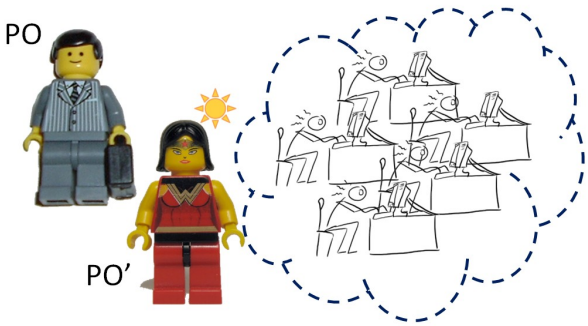
Аналитик – помощник Product Owner-а

Большинство недостатков предыдущей схемы можно преодолеть, если поделить обязанности Product Owner-а и аналитика между двумя людьми (и это не чуждо нашей культуре — см. рис. с двуглавым орлом). Это достаточно распространенная практика. Как правило, при этом «главный» решает, что в какую очередь делать, и дополнительно выполняет менеджерские функции (и/или отвечает за клиентские отношения). А помощник больше концентрируется на содержании и деталях работ, т.е. играет роль аналитика.



«Двуглавый» Product Owner

На конференции QCon 2008 London в докладе «Agile Mashups» Рейчел Дэвис (Rachel Davies), ссылаясь на свою обширную практику консультанта, утверждала, что это самая распространенная схема.



Однако, и у нее всё же присутствуют недостатки:

- Деятельность аналитика недостаточно прозрачна для команды, поскольку аналитик в нее не входит.
- Велика вероятность, что команда будет воспринимать аналитика, как Product Owner-а (или даже руководителя), т.е. видеть в нем не помощника и эксперта, а начальника, что почти наверняка убьёт критичность восприятия предложений и моделей, предлагаемых аналитиком — их будут воспринимать не как начальное приближение и дополнительную информацию, а как инструкции к действию.
- Опять же, не так-то просто привлечь такого аналитика к сопровождению.

Аналитик внутри команды

Можно пойти еще дальше и поместить аналитика внутрь команды. Что это значит:

- аналитик сидит вместе со всеми, т.е. в одной комнате с разработчиками;
- аналитик участвует в Scrum-митингах наравне с остальными (рассказывает, что делал вчера и что собирается делать сегодня);
- работа аналитика учитывается при планировании итерации;
- аналитик может «делиться» своей работой с другими (по общему согласию);
- и наоборот, аналитик может привлекаться к нехарактерным для себя работам, чтобы помочь остальной команде в непростую минуту — например, подготовить тестовые данные, прогнать часть ручных тестов.

Звучит здорово. И работает здорово! Однако бывают обстоятельства, при которых данная схема не подходит:

- одной предметной областью (или сильно похожими) занимается несколько команд разработчиков, так что держать по аналитику в каждой команде нет особого смысла;
- реализация больших или технически сложных проектов в методологии Scrum-of-Scrum;
- в проекте так много технических и технологических тонкостей и сложностей, что команда в основном сконцентрирована на них, а аналитик в такой команде оказывается инородным телом;
- нехватка квалифицированных аналитиков.

Часть из этих недостатков можно преодолеть, если сделать одного аналитика «по совместительству» сразу для нескольких команд. Т.е. аналитик, участвующий в работе не одной, а нескольких команд. Здесь под словом «нескольких» в 99% случаев понимается *двух* – вряд ли аналитик сможет потянуть больше:

- придется участвовать в N Scrum-митингах каждый день;
- придется участвовать в N сессиях планирования каждые несколько недель;

- придется участвовать в N демонстрациях каждые несколько недель;
- придется участвовать в N ретроспективах каждые несколько недель.

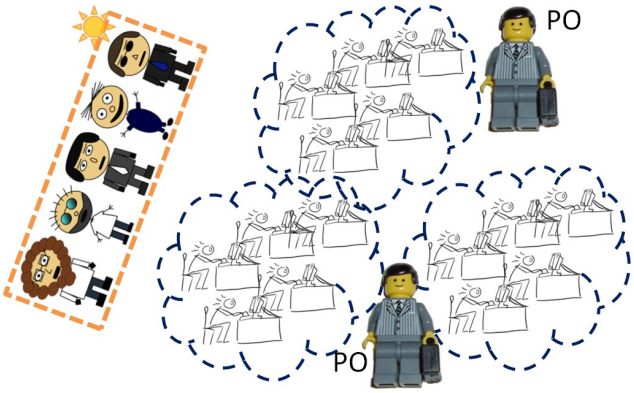
То есть получаются сплошные «накладные» расходы на Scrum-процесс, а надо же еще и «работу работать», да еще и на N команд…

Внешний отдел аналитиков

Это то, что больше характерно для водопадных и тяжеловесных методологий, но может работать и в Agile — например, в обстоятельствах, описанных в предыдущем пункте.

Из дополнительных плюсов: простота и естественность обмена информацией, навыками и практиками между аналитиками.

Основной недостаток: *экстремально* высокая отдаленность от команды разработчиков, что может привести к взаимному недоверию. Кроме того, вырастает вероятность отката к прежним практикам и прежним проблемам.



VI. Особенности аналитика в Agile

Так есть разница между аналитиком в Agile и не-Agile (например, в водопаде или другой тяжеловесной методологии)? Однозначный ответ – есть!

В тяжеловесных методологиях аналитик похож на слабопроницаемую стену между командой разработчиков и представителями заказчика/бизнеса. Чтобы сделать хороший продукт, приходится тратить кучу сил на «ковыряние дырок» в этой стене. Кроме того, ужасно высоки риски, связанные с ошибками аналитика. Команде разработчиков при этом отводится роль второго плана.

В Agile же аналитик играет роль связующего звена между разработчиками и заказчиками – своего рода магнит, который не дает им разбежаться по разным углам и тихо там что-то делать без ведома друг друга. При этом команде разработчиков отводится весьма значимая роль. Благодаря этому снижаются риски, связанные с ошибками аналитика – если что, команда уточнит/поправит (а если не поправит, то на демонстрации заказчики поправят). Т.е. у аналитика в Agile есть «лекарство от страха».

Аналитик в Agile – это золотая середина между следующими крайностями:

Одна крайность	Золотая середина	Другая крайность
Команду **не** допускают к аналитической работе	Agile	Разработчикам самим приходится **полностью** прояснять что же нужно

Одна крайность	Золотая середина	Другая крайность
Аналитик **мало** общается с заказчиком	Agile	Аналитик **всё время** проводит у заказчика

Одна крайность	Золотая середина	Другая крайность
Подробные спецификации перед началом итерации	Agile	Отсутствие **какой-либо проработки** требований до постановки их в итерацию

Одна крайность	Золотая середина	Другая крайность
Аналитик **не рисует** никаких диаграмм и схем	Agile	Аналитик **не пишет текст** – исключительно рисует диаграммы и схемы

Одна крайность	Золотая середина	Другая крайность
Команда с «придыханием» относится к постановкам аналитика	Agile	Команда **не доверяет** результатам работы аналитика (не использует их)

Одна крайность	Золотая середина	Другая крайность
Аналитик **не** участвует в тестировании (QA)	Agile	Аналитик вынужден **постоянно** «протыкивать» много **старых** интерфейсов

Одна крайность	Золотая середина	Другая крайность
Команда воспринимает аналитика **как руководителя**	Agile	Аналитик для команды – мальчик/девочка **«на побегушках»**

Одна крайность	Золотая середина	Другая крайность
Аналитик взаимодействует с командой **исключительно** при помощи **документации** и Bug-трекера	Agile	Аналитик взаимодействует с командой **исключительно** посредством **устных коммуникаций**

Одна крайность	Золотая середина	Другая крайность
С заказчиком и пользователями **общается только аналитик**	Agile	Все члены команды **без исключения вынуждены плотно** общаться с заказчиками и пользователями

Можно продолжать, но я остановлюсь…

VII. Смежные вопросы

Хотелось бы здесь еще рассмотреть такие сопутствующие вопросы как:

- Что такое Agile-спецификации? В каком виде следует формулировать и оформлять задачи на итерацию?
- Внутренняя проектная документация: как и в чём вести, какой инструментарий использовать?
- Что такое Domain Driven Design? И почему модели предметной области снова в моде? Как это связано с популярностью Agile?
- Как выращивать аналитиков внутри компании?

Но нужно что-то оставить и для будущих публикаций/заметок/докладов.

VIII. Заключение

Пожалуйста, не превращайте Agile в очередной карго-культ! Пытайтесь осознать основные идеи и следовать им — попытки использовать материалы по Agile-методологиям, как исчерпывающие инструкции или догматы, приведут вас к чему угодно, кроме того, что изначально вкладывалось в понятие Agile.

Описанные варианты, способы взаимодействия и подходы к проблемам бизнес-анализа не являются исчерпывающими или строго обязательными — в вашем конкретном случае другие приемы могут оказаться более эффективными и оправданными. Было бы очень интересно узнать об альтернативах — не замалчивайте их!

Буду рад любым откликам!