Steamus карма 56,4 рейтинг 5,6 363 голоса

9 сентября 2008 в 04:08

Панацея ли Scrum? Давайте рассуждать вместе, где он нам полезен

Управление проектами*

Начну я просто — поясню, что такое Scrum и зачем он нужен, что бы те люди, кто с ним пока не сталкивался, могли с интересом прочесть данную заметку и понять о чём собственно идёт речь.

Итак, Scrum, это популярная (модная, если хотите) сегодня методология ведения программных проектов. Другими словами, как управлять командой разработчиков, что бы программный проект завершился успешно. Что и как документировать, как, с кем и как часто обсуждать детали проекта, как ставить задачи людям и как контролировать результат. Всё это попадает под термин “методология управления программным проектом”.

Вам понятно? Отлично! А теперь... я честно скажу, что я и сам не знаю что такое скрам. Но я знаю, что все разработчики его очень любят. Почему? Ну, наверное, потому что скрам крайне неформальный подход к разработке, предполагающий минимум документирования и много общения. Вы часто видели разработчиков, которым нравится писать отдельную документацию на свой собственный код? Я – редко.

Помните такой старый анекдот – руководство некоего института решило провести эксперимент: предложить студентам самим принимать экзамены у своих сокурсников (вместо преподавателей). И что вы думаете? Эксперимент себя полностью оправдал – в зачётках студентов только хорошие и отличные оценки!

Вернёмся к скраму. Скрам есть одна из популярных методологий, относящихся к классу Agile. И, следующих принципам XP (Extreme Programming). Вот тут хотелось бы внятно пояснить, что такое XP, Agile и Scrum. Ну, разумеется, так как я сам это понимаю (да вы сильно не пугайтесь, некое количество букварей я из вежливости прочёл, а те, кто прочёл больше, я знаю, обязательно меня тактично поправят).

Итак – XP.

Когда возникла вселенная, и люди только начали писать компьютерные программы, они относились к этому очень просто. Примерно как к созданию самолёта. Дело плёвое. Взлетит, не взлетит… да куда он денется. Ну а если ёбн… не взлетит в смысле, чтож, значит ошиблись где-то в расчётах. Бывает. Делать просто, это в смысле – делать всё последовательно. То есть садятся опытные, как правило уже седые мужики (как я примерно), и тщательно планируют всё наперёд. Иногда, сдуру, садились неседые, но у них, обычно всё-таки получалось хуже. Да. И эти мужики планируют работу по принципу: мужик сказал – мужик сделал! А мужик сказал просто, сегодня мы выжимаем из этих грёбанных заказчиков всё что они про ЭТО знают (а фуля, это же им надо!). Завтра, наши аналитики тщательно ЭТО фиксируют в спецификациях, и затем, самые наши аналитичные аналитики предлагают лучшие подходы, что бы сделать всё как ИМ надо в лучшем виде. Строго и после затем, из моря выходят архитекторы и подбирают наиболее подходящую архитектуру. И, ну теперь-то всё понятно. Налетают мальчишки кодеры, быстро всё это надиктовывают на… да это уже и не важно то на чём. И на выходе получаем конфетку.

В чём особенность такого подхода? А в том, что всё делается строго последовательно. Как в армии: подъём, зарядка, завтрак, построение на плацу… То есть, мы медленно… медленно спустимся с горы и тщательно перее… Я хотел сказать, что проект неукоснительно и бесповоротно следует к финишу, как вода падает со скалы. Подход так и называется – водопад (waterfall). Всё спланировано, всё предопределено, сроки размечены наперёд с точностью до часа. Всё бы хорошо, но… Нашлись наглые и дерзкие заказчики, которые почему то не захотели называть результат конфетой. Вкус им, видите-ли, не понравился. Чёрт с ним как они это назвали (не к обеду будь упомянуто), но они и платить то не хотели. Что уже не очень и хорошо.

В общем, эта ерунда, видимо, стала повторяться с завидной регулярностью. Много лет, мы, разработчики это терпели, но затем пришёл конец. Лет десять назад самые мудрые из нас, набрались смелости и приняли другое решение, и постулировали иные принципы.

Принципы эти таковы:

— Заказчику спуску не даём! Мы сразу берём небольшую паузу на то, что бы сделать правильную декомпозицию объектов, спроектировать интерфейсы к программным модулям, забить их заглушками за недели три, и далее выкатить первые визуальные формы

— Мы держим заказчика за хвост и показываем ему эти формы хотя бы раз в неделю. Заказчик начинает

осознавать, что и как он действительно хотел и как ему в самом деле удобно. Его уже не пугает ничего. Он начинает мыслить!

— Мы не паримся с тщательным названием переменных и методов кода. Мы выгоняем код быстро. Мы даже садимся рядом, чтобы другой чел не читал хабр в то время когда я кодирую

— Мы понимаем, что нагородим много плохо работающего кода. Посему самые умные из нас изначально пишут тесты на предмет проверки того, что бы проверить, что всё это работает так как вроде бы хотел заказчик. И мы всегда проверяем каждый пук (зачёркнуто), нашу работу этими тестами (test driven development)

— Мы, в душе, понимаем, что код должен хорошо выглядеть. Посему мы периодически его чистим. Не меняя его суть и не перепроектируя. По умному — делаем рефакторинг. Что бы в итоге не напартачить, на всякое изменение гоняем тесты.

То есть мы по полной, с первого дня, включаемся в гонку с заказчиком на предмет того, что бы как можно быстрее получать от него отдачу. Вначале просто формы, затем пошли расчёты, затем сценарий… Заказчик пищит, но он всегда под рукой… (хвост то один) Он уже не скажет, что ждал чего-то нового и интересного. Он уже офигел от нашего напора.

В чём отличие? А отличие в том, что мы не просто согласовываем с заказчиком как будет выглядеть деревянное изделие на выходе. Нет. Хотя и это тоже. Мы шаг за шагом снимаем стружку, пусть даже и не обрабатывая изделие наждачкой, но мы настойчиво суём его под нос заказчику с завидной регулярностью, что бы он проникся и не сказал потом, что этот выточенный хрен выглядит совершенно иначе, чем он представлял его в своих мечтах.

Мы спиралью последовательно умощняем наш проект, сохраняя его в рабочем состоянии и тихо, ночами, подменяем наши заглушки уже реальным кодом. Предварительно прогнав наши тесты.

Да. Вот это и есть XP. Набор технических принципов, как сделать разработку чувствительной к внешним требованиям. Это ещё не методология управления, а именно и только набор полезных технических подходов.

Грубо говоря, вместо того, что бы отслюнявить химический карандаш, провести линию и начать, отвернувшись, рубить по ней топором, вы организуете работу так, что бы тактично водить рубанком, показывая каждый срез стружки заказчику и учитывать все его замечания. Пусть смотрит и тащится.

Поскольку пиво закончилось, то продолжение последует позже. :-)

scrum xp проектирование

+60 2754 66 Steamus 5,6

Комментарии (62)

glader 9 сентября 2008 в 07:29 # +34

«Мы даже садимся рядом, чтобы другой чел не читал хабр в то время когда я кодирую»
Наконец-то честно сказали, зачем нужно парное кодирование

trix 11 сентября 2008 в 13:16 # ↑ +5

если налаживается хороший контакт, то во время сессии парного программирования оба разработчика читают сначала френдленту одного из них, затем оба читают френдленту другого.

Vodkin 11 сентября 2008 в 14:47 # ↑ +1

мог бы плюсовать — плюсанул :-)

Kuzma 13 сентября 2008 в 18:11 # ↑ 0

По идее уже можешь)

Vodkin 16 сентября 2008 в 14:46 # ↑ 0

Теперь Хабр не дает: Хабрахабр — Голосования: Время голосования прошло«…

XaocCPS 9 сентября 2008 в 08:06 # +19

Статья о Scrum а описывается в основном Extreme Programming, где раскрытие темы статьи? я так и не понял что такое Scrum, почему он панацея и где он полезен :(

croatian 9 сентября 2008 в 08:29 # ↑ +3

Пиво закончилось =P

XaocCPS 9 сентября 2008 в 08:35 # ↑ +3

ну это, конечно, все объясняет :)

porchini 9 сентября 2008 в 10:32 # ↑ +2

а он статью пишет тоже по scrum'y

Steamus 9 сентября 2008 в 13:36 # ↑ +1

Scrum, это Agile методология управления проектами, в основу которой положены принципы и подходы XP. Включая основной. Если это не оттенить, то говорить о Scrum сложно. Что бы не сказать — бесполезно.

alexshelkov 11 сентября 2008 в 00:32 # ↑ +4

Да какая разница, не принципиально, статья то хорошая =) Все бы так писали. Да и как указал автор нужно быть последовательным.
Радует меня такие статьи, статьи — в которых, с долей юмора, сам автор, описывает проблемную тему. Не копипаст выдержек из разных источников, а именно рассказ. В общем Спасибо, и шоу маст гоу он =)

GubkaBob 9 сентября 2008 в 10:25 # +8

Во первых не пойму для кого эта статья — для пм-ов или членов команды?

Комментарии:
качество кода и скрам — никак не связаны. точно так же налажать можно и в водопаде, и в итерациях, и в спринте. зря вы это смешали.

отсутствие документации — тоже не верно. мои проекты документируются. спецификации — нет, но она и не нужна, поскольку все и так уже описано в product requirements.

основной принцип скрама не в том, что — сначала сделаем, а потом посмотрим. а в том, что делается понемногу, минимальными итерациями — неделя-две. это значит что через неделю заказчик получает оттестированный продукт, в которой есть по крайней мере одна имплементированная фича и CR. Таким образом минимизируется изменения в работающих компонентах. Но команда тем не менее отвечает за качество и полноту кода.

Я понимаю, что можно привести пример когда скрам или водопад заваливает проект. Тем не менее скрам интересен своей динамичностью, люди не сидят в одном и том же месте (коде) неделями, а могут оперативно меняться. Это кстати, одна из причин pair programming. Есть несколько основных принципов в agile (которые, например, можно узнать из книжки Бека): common ownership, pair programming, shared knowledge.

Как человек, поработавший со всем, чем можно могу сказать, что скрам «веселее», дает больше шансов проявиться, проекты как правило небольшие и не дают засиживаться. Но и выкладываться конечно в скрам-проекте на 100% придется. Так что любителям полазать в контакте не советую.

Steamus 9 сентября 2008 в 13:20 # ↑ +2

Перенёс текст сюда снизу.

Когда речь заходит о некоем наборе принципов, которые лягут в основу методологий, то вначале важно уловить именно ключевую суть, а не перечислить все сопутствующие признаки. Ключевая суть XP именно в том, что бы завязать проект в спираль, на каждом витке которой вы имеете работающую систему. И, тем самым, имеете что-то, отчего сразу можете отталкиваться в вашей коммуникации с заказчиком.

XP кстати, имеет и ещё одну жёстокую особенность, которая часто остаётся в тени. А именно — что бы построить проект таким образом, вам нужно крайне хорошо владеть принципами проектирования. Ибо никаким рефакторингом вы не заполируете тот факт, что, к примеру, на ранних этапах проектирования вы допустили серъёзные изъяны в декомпозиции системы.

Junior 10 сентября 2008 в 21:56 # ↑ +2

Иомайо, а я думал, же мне напоминает этот «многоэтапный план поставки продкта» из книги Макконнелла «Остаться в живых». Сенкю за прозрение :)

bishop3000 9 сентября 2008 в 11:40 # +9

Пара комментариев:
>> Почему? Ну, наверное, потому что скрам крайне неформальный подход к разработке, предполагающий минимум документирования и много общения. Вы часто видели
>> разработчиков, которым нравится писать отдельную документацию на свой собственный код? Я – редко.

Это общее заблуждение всех, кто про SCRUM и XP только читал. В книгах написано не «документации быть не должно», а «не надо создавать ЛИШНЕЙ документации». То есть, если заказчику и коменде не нужна полная UML схема проекта, то не надо ее делать. Если небольшой документации в wiki по каждому модулю достаточно — отлично, не надо писать целую книгу про программу. Не нужно заводить специальную должность, который будет писать документацию, если она не нужна.

>> — Мы не паримся с тщательным названием переменных и методов кода. Мы выгоняем код быстро. Мы даже садимся рядом, чтобы другой чел не читал хабр в то время
>>когда я кодирую

Про «не паримся с названием» — это вообще неправда и обман. В гибких технологиях качество кода — одна из важнейших составляющих успеха. У всех есть coding standards, code review и т.п. Парное программирование вообще позволяет забыть про страшный код, т.к. второй именно за этим и следит. Ну, также и чтобы на хабр не лезть — да. Это, кстати, производительность иногда в разы повышает :)

Но в целом про XP верно написано, если убрать негативный оттенок из текста :)
Смысл — Подцепить заказчика на крючок и держать его там до конца. Но это обоюдовыгодный крючок.

Steamus 9 сентября 2008 в 13:03 # ↑ +4

Нагативного оттенка нет. Я считаю, что именно на XP подходы в наше время и должны опираться методологии управления проектами. Обязательно ли это должна быть методология Scrum, это уже другой вопрос.

Насчёт документации. Вы абсолютно правы. Я также нигде не встречал, что некая методология вслух постулирует позицию вида — не пишите документацию. Конечно же нет. Но, если методология явно не настаивает на том, какие документы и где должны порождаться в процессе ведения проекта, то тем самым она оставляет это на откуп разработчика или руководителя проекта. И у них соблазн велик. Многие искренне полагают, что Скрам им развязал руки и терь можно только говорить и ничего не писать. Типа мы и раньше так делали, а тут и научная база подоспела. Это действительно заблуждение.

К примеру, если взять RUP, то там внятно перечислено достаточно много так называемых артефактов, которые должны (желательны) порождаться на каждой стадии проекта. Артефакт, это, грубо говоря, некий, как правило бумажный, выхлоп после прохождения стадии создания проекта. Называется он таким словом потому, что быть он может разным. Это и текст, и UML-диаграмма, и просто сведёные в таблицу кейсы, и… RUP использует UML и тщательно прописывает какого рода UML диаграммы вам желательно сделать в том или ином случае.

По поводу качества кода. Тут опять же — ни одна здравомыслящая методология не предлагает писать грязный код или код не в соответствии с принятыми в компании стандартами. Ни в коем разе.
Но, XP прямо и честно говорит — и в коде не старайтесь всё учесть сразу. Не тратьте на первых этапах свое время на то, что бы придумать максимально точное название. Не тратьте время на то, что бы сразу и окончательно решить будет эта переменная локальной или членом класса. Не тратьте сразу время на то, что бы окончательно решить что будет отдельным методом класса и какая у него область видимости. Не потому, что это не важно, а потому, что вы пока этого окончательно не знаете. Признайте факт того, что с высокой вероятностью это может измениться и сейчас сделайте так как вам удобнее. Затем, когда некая часть кода устоится (или просто появится больше определённости), вы обязательно зарефакте эту часть, что бы окончательный код был действительно хорош и удобен. Вот где-то так примерно.

Steamus 9 сентября 2008 в 13:19 # ↑ +1

Когда речь заходит о некоем наборе принципов, которые лягут в основу методологий, то вначале важно уловить именно ключевую суть, а не перчислить все сопутствующие признаки. Ключевая суть XP именно в том, что бы завязать проект в спираль, на каждом витке которой вы имеете работающую систему. И, тем самым, имеете что-то, отчего сразу можете отталкиваться в вашей коммуникации с заказчиком.

XP кстати, имеет и ещё одну жёстокую особенность, которая часто остаётся в тени. А именно — что бы построить проект таким образом, вам нужно крайне хорошо владеть принципами проектирования. Ибо никаким рефакторингом вы не заполируете тот факт, что, к примеру, на ранних этапах проектирования вы допустили серъёзные изъяны в декомпозиции системы.

ness 10 сентября 2008 в 20:33 # +1

Все используем, но по поводу тестов: для многих функций мы не пишем тесты только потому, что написание теста занимает столько времени, за сколько можно написать еще пару-тройку функций :)

В основном, если пишем тесты, то для, скажем так, «вычислительных функций», в которых зашит некий алгоритм с достаточно сложной логикой.

Т.е. если функция выбирает данные с базы и возвращает некий объект, то для такой функции тесты, по-моему, излишни.

Хотя, с точки зрения TDD это, может и неверно, но все-таки надо применять гибкий подход: иногда писать тесты, а иногда и нет :)

VasilioRuzanni 10 сентября 2008 в 20:39 # ↑ 0

Тесты не только для того, чтобы проверить расчеты. Они, в частности, для того, чтобы держать всю систему в согласованном состоянии и быть уверенным, что разные блоки системы нормально стыкуются друг с другом и ничего «друг у друга не ломают». Это принципы постоянной интеграции (CI) — в маленьком проекте это может быть не так критично, но для продуктов, поддерживаемых и дорабатываемых (и расширяемых) в течение длительного времени — это просто панацея от большинства багов (в особенности, функциональных и интеграционных).

ness 10 сентября 2008 в 20:48 # ↑ +1

Да, сама концепция «запустил набор тестов-получил ответ, где баги» мне нравится.

Во многих случаях действительно получалось, что изменение в одной части системы нарушало что-то в другой, но становилось это видно только тогда, тогда добираешься до проверки той другой части.

Вот только подскажите, как бы ее начать применять и не забить на полпути? :)

Еще одна проблема, которая возникает при внедрении TDD — непонятность с объемом:
если есть 25 объектов, и у каждого по 6-7 функций, то выходит, что нужно минимум 150 тестов, не учитывая комплексных, которые работают с несколькими объектами или функциями.

Как быть?

Или где об этом почитать :)? (не теорию, а именно практические аспекты применения)

VasilioRuzanni 10 сентября 2008 в 23:06 # ↑ +1

Да, совершенно верно. Придется писать эти, грубо говоря, 150 тестов. Более того, тесты могут быть достаточно сложными (если используется подход mocking-а объектов и тестирование поведений — один из двух стилей TDD). Но это с лихвой (в частности, у нас) окупается отсутствием длительного времяпрепровождения в дебаггере и в простоте дальнейшей поддержки и беспрепятственным развитием нового функционала.

CI — внедряется совсем не сложно. Не буду описывать здесь всю концепцию, однако в целом процесс выглядит так:
— Задачи делятся на очень атомарные (то есть, программист не должен выполнять одну задачу, скажем, неделю, максимум — день);
— Все всегда хранится в репозитории контроля версий;
— Кусок кода (рабочая копия), полностью покрытый тестами и работающий помещается в репозиторий;
— Параллельно, при загрузке, может возникнуть необходимость соединения кусков файла когда разные разработчики работали с одним и тем же файлом);
— На сервере стоит CI-сервер (мы используем CruiseControl.NET), который автоматически после помещения разработчиком его куска кода, с помощью NAnt собирает общий билд решения и запускает необходимые тесты на выполнение (также запускается NCover для проверки уровня покрытия кода тестами и несколько других полезных и не очень утилит);
— Если все тесты, запущенные CI-сервером выполняются, значит **интеграция прошла успешно**.
— Если часть тестов не выполнилась — разработчик, запустивший свой код в репозиторий последним, устраняет ошибки, переписывая часть кода (причем, **не только своего**!), чтобы все тесты всего решения выполнялись;

В итоге получаем, что решение в репозитории всегда работающее, а ошибки не живут, в общем случае, больше суток. Причем, это не зависит от размера проекта. В итоге, мы не проводим перед релизом (или после его предполагаемой даты :), интеграцию, в течение трех недель вылавливая ошибки и пытаясь понять «кто и где напортачил».

Более подробно и детально процесс описан в статье Мартина Фаулера Continuous Integration.

VasilioRuzanni 10 сентября 2008 в 23:10 # ↑ 0

Кстати, при использовании CI, то, что разработчики правят код друг друга приводит к реализации другой хорошей практике Agile — коллективное владение кодом. При грамотном использовании ротации и рецензировании кода, каждый сотрудник в проекте, помимо понимания всех частей системы, становится также в значительной степени заменяемым — нет страха того, что если он заболеет/уйдет/уедет в отпуск, некому будет поддерживать и развивать код, написанный им.

Steamus 10 сентября 2008 в 23:13 # ↑ +1

Концепция тестов в XP чуть шире. Что бы просто найти баги, достаточно один раз всё аккуратно проверить и поправить. В XP же предполагается, что вы и код будете итеративно менять (проводить рефакторинг), что бы он был вылизан и удобен для последующего сопровождения.

Меняя код, всегда есть риск внести ошибку. Посему, помимо поиска ошибок, программыне тесты (юнит-тесты) нужны для быстрой механической проверки уже работающих кусков кода после рефакторинга.

VasilioRuzanni 10 сентября 2008 в 20:36 # 0

Здесь стоит провести небольшую грань между Scrum и XP. Сами создатели Скрама говорят, что это больше «организационная» (то есть, о том, как организовать сам проект) методология, и никак не противоречит использованию «инженерных» (то есть, как организовать работу над реализацией проекта, включая технические аспекты) методологий вроде XP.

Объединяет же их то, что они обе «agile» — то есть это адаптивный подход, связанный с реальностью — для сложной системы никогда не известно заранее, что должно получиться и лучше не «гадать на кофейной гуще», а разрабатывать постепенно то, что уже известно.

Инженерные практики типа TDD здесь просто в помощь и адресованы для решения вполне конкретных проблем.

Сами используем и XP и Scrum, чему несказанно рады. Люди уже просто не представляют как можно работать без таких техник, как Test-Driven Development, CI (Continuous Integration) и других — ведь это в разы повышает качество и предсказуемость кода, чем сильно упрощает жизнь.

Вообще, прийти к Alige заставляет понимание того, что процесс разработки должен быть как-то адаптирован к **постоянным** запросам на изменение от заказчиков, а классический процесс не адаптирован к этому.

 **afitiskin** 10 сентября 2008 в 20:46 # +1

Scrum подразумевает работу в команде, а это немного отличается от парного программирования тем, что 2 человека не тыкают по очереди по одной клавиатуре, а они садятся и обсуждают каким образом модули, которые они будут реализовывать будут соединены воедино, и смысл в том что именно это обсуждение заменяет документацию (уменьшает ее объем и время потраченное на нее).

 **hemule** 10 сентября 2008 в 20:46 # +2

Никакая методика не является панацеей, если методика позиционирует себя как панацея — это повод лишний раз ее внимательно проверить на адекватность ситуации, к которой ее предлагается приложить. Панацеей является только желание думать и выбирать методику или набор методик, которые подходят к конкретной ситуации.

 **corvus** 10 сентября 2008 в 21:09 # +1

Я бы вынес описание сабжа ДО хабраката. Это позволит человеку понять о чем идет речь, не открывая страницу — простое правило хорошего тона.

Steamus 10 сентября 2008 в 22:16 # ↑ +1

Согласен. Спасибо.

 **lalaki** 10 сентября 2008 в 21:40 # +1

невероятная профанация — такое ощущение, что автор далек от управления проектами. Это, наверное, плохо и, возможно, невежливо, но копаться в тексте и расписывать каждое некорректное место не буду — не стоит того. Просто, наткнувшись на фразу «Лет десять назад самые мудрые из нас, набрались смелости и приняли другое решение, и постулировали иные принципы.» — типа, водопад, это плохо, а итеративная/спиральная разработка лучше? Очевидно же, что для разных проектов подходят разные методологии. И есть, и долго еще будет большое количество проектов, для которых подходит только «водопад», пусть даже с большими затратами. После пары таких высказываний и откровенно небрежных выводов нет желания что-либо пытаться улучшить в этой статье. Соответственно, такие высказывания заставляют задуматься: то ли Автор так не уважает читателей, что позволяет себе писать откровенную ерунду, то ли он сам в этом слабо разбирается.

поэтому вопрос, который с первых строк возник — для чего Автор написал эту статью? он столкнулся с соответствующей проблемой и решил обсудить ее? или что-то для себя понял и решил просветить начинающих? другой вариант? поэтому и интересно, какое отношение сам Автор имеет к УП? (несмотря на первый «негативный» абзац, это не попытка перехода на личности — вопрос возник сразу же, при прочтении вступления — всегда лучше воспринимаю статью, если понимаю мотивацию автора, или хотя бы то, что он за нее выдает).

Steamus 10 сентября 2008 в 21:52 # ↑ +2

Автор имеет некоторый опыт управления проектами. :-)
Также, автор не имеет ничего против водопада. Также как и Рупа и прочих. Автор также полагает, что XP подходы были достаточно революционны в свое время и заметно отличались от других. С проблемами управления автор сталкивался часто. Читателей он уважает. Иначе не стал бы писать.

 **lalaki** 10 сентября 2008 в 23:12 # ↑ +1

Спасибо.
Собственно, остается вопрос: цель этой статьи? И, конечно, когда будет раскрыта собственно тема статьи? Пока что есть только некое описание трех методологий, но тема выбора не затронута

Steamus 10 сентября 2008 в 23:43 # ↑ +1

Цель — перевести обсуждение Скрам в критическую плоскость. Не потому, что он плох, а ради понимания того, что и где он навязывает, когда даёт свободу и какие важные вещи действительно остаются на откуп разработчикам или просто остаются за кадром. Что бы применять его грамотно и точно, не сводя возможную неудачу проекта к недостаткам методологии.

 **lalaki** 11 сентября 2008 в 01:08 # ↑ 0

хорошая цель)

довольно интересный вопрос, хоть и выводящий за пределы привычной сфре скрама — в каких непрограммистских/неайтишных проектах его можно применять. Я на такой вопрос не могу ответить с достаточной долей уверенности.

если брать айти — интересно рассмотреть проекты с «внутренним» заказчиком. например, в проектах разработки и внедрения внутрикорпоративного продукта. Как правило, такие проекты затрагивают не одно подразделение, соответственно, много руководителей так или иначе участвует в проекте или пытается на него влиять. Тут, по моему мнению и опыту, скрам позволяет максимально защитить и команду проекта, и сам проект от множества векторов влияния, продиктованных, к тому же, самыми разнообразными причинами. Но для этого (опять же по опыту), очень сплоченно должны действовать куратор и руководитель проекта (т.е. Product Owner и Scrum Master) — им вместе придется стандартные совещания начальства стараться сделать максимально продуктивными для ведения баклога проекта, и вместе же придется отстаивать такие вещи, как неприкосновенность плана текущего спринта. Кроме того, могут быть проблемы, если участники команды структурно находятся в разных подразделениях — соответственно, над ними, кроме руководителя проекта, есть еще линейные руководители, опять же имеющие свою точку зрения на

проект, не всегда совпадающую с утвержденным баклогом (такая ситуация может быть в айтишных фирмах). Другие интересные варианты проектов с «внутренним» заказчиком — разработка принципиально нового продукта/разработка продукта, уже имеющего аналоги, в т.ч. новой версии существующего продукта. Отличаются, по моему, в первую очередь агрессивностью внешней среды — во втором случае добавляются такие факторы давления, как ожидания/претензии существующих пользователей, учет проекта в планах продаж, соответственно, различные публичные обещания руководства, маркетологов и т.п. Первый случай — это, фактически, стартап, т.е. проект, не встраивающийся в существующий рынок, а пытающийся этот рынок изменить/перевернуть/создать новый, и не имеющий описанной «наследственности» — существующих пользователей и т.п. Несмотря на то, что проекты а ля стартапы имеют тенденцию сильно меняться в ходе своего развития, все-таки постоянного сильного давления на них нет. К проектам первого типа скрам, по-моему, вообще очень хорошо подходит — позволяет двигаться даже без четкой дорожной карты. К проектам второго, в принципе, не хуже — опять же, внешняя среда очень изменчива.

проекты с внутренним заказчиком интересны прежде всего потому, что они заведомо рассчитаны на «продолжение» — после формального запуска проекта так или иначе его нужно продолжать разрабатывать, и, обычно, силами той же команды. То есть у них заведомо есть установка — сделать успешный продукт. В то время как в проектах «для дяди» основная цель — сделать продукт, которым будет доволен заказчик (успешно сдать продукт). А некоторая разница между этими целями есть)

Важный момент: под разработчиками Вы понимаете программистов/проектировщиков/архитекторов или всех членов команды? потому что говорить, что скрам отдает на откуп программистам какую-либо задачу, обычно не свойственную им, например, аналитику, — неверно в принципе: скрам позволяет команде самой задавать распределение задач, но происходит это в соответствии в том числе с компетенцией. И даже если, скажем, программист занимается аналитикой — это не потому, что скрам ему ее на откуп отдал, а потому, что программист в данном спринте играет еще и роль аналитика, т.к. команда нашла его достаточно компетентным. Т.е. все равно аналитикой занимается аналитик.

Возвращаясь к статье: обсуждать методологию адекватно могут только те, кто, вне зависимости от специализации, достаточно опытен, чтобы вообще осознавать суть и назначение методологий. Обсуждать же пригодность методологии к тем или иным видам проектов, по-моему, вообще могут только люди, опытные в УП — менеджеры проектов да ведущие разработчики. А для таких людей статья, в принципе, ничего полезного не несет — возможно, хватило бы просто вопроса? Все равно в статье вопрос не задан более развернуто, чем в названии))

Steamus 11 сентября 2008 в 03:59 # ↑ 0

Трудно вычислить насколько я осознаю суть и назначение методологий. Это всё субъективно. Программные проекты я веду более 20 лет. Сказать что всё было успешно, и что я всё правильно осознаю? Не уверен. Не всё успешно. Но многое. Полагаю, что, что-то видимо я уже осознаю. :-)

Статья носит общеобразовательный характер. Повод для дискуссии. Осмысление вроде бы тривиальных вещей. Я не считаю что я во всём прав. Но какие-то знания определённо есть. В конце концов. я не навязываю свой опыт. Каждый вправе пользоваться любой, удобной ему информацией.

belonesox 6 февраля 2009 в 23:24 # ↑ 0

>Т.е. все равно аналитикой занимается аналитик.
Время для дискуссий наверно уже вышло, и все же замечу, что при переходе на SCRUM или иной вид кроссфункциональной организации часто возникает проблема, куда девать выделенных аналитиков.
Товарищ делал интересный доклад на эту тему: «Аналитик в Agile: Архаизм или необходимость?», возможно заинтересует.

paurock 13 сентября 2012 в 19:48 # ↑ 0

да хорошая статья для новичков. я вот только вхожу в курс дела. и обычно делаю это по принципу от общего к частному. поэтому статья мне дает некоторые представления вообще что за скрум хр и с чем их едят.
по моему, сложные вещи описать простым понятным языком много стоит. плюс, читая коммены, например, ваш, начинаешь глубже вникать в детали.

lalaki 13 сентября 2012 в 22:05 # ↑ 0

Ничего себе привет из прошлого) — это я про ответ на 4-летний коммент.

Если новичок, советую для начала почитать что-нибудь про принципы организации бизнес-процессов в целом, например, «Дао Toyota» — все программистские «новые методологии» — по сути, адаптация опыта других сфер в сочетании со здравым смыслом к отрасли разработки ПО, поскольку базовые законы везде одинаковы.

paurock 14 сентября 2012 в 09:52 # ↑ 0

=) да и тоакое бывает,
спасибо, за рекомендацию

lalaki 11 сентября 2008 в 01:21 # ↑ 0

извиняюсь за многа букав)
подумал, пока писал предыдущий пост: имеет больше смысла отталкиваться от типа создаваемого продукта, как, например, это делает Джоэл — www.joelonsoftware.com/articles/FiveWorlds.html. Возможно, кто-то не согласится с классификацией. Но даже по ней видно, что проекты из первого и второго миров как раз подходят под Agile в целом

— изменчивая внешняя среда, возможно, отсутствие полной дорожной карты проекта.
Вообще, гораздо проще понять, куда подходит/не подходит Agile, чем сформулировать критерии выбора внутри Agile, не опускаясь в личные предпочтения. Хотя, если XP брать как жестко заданный набор практик, можно понять, где его нельзя использовать) Скрам же все-таки проще, менее детализирован — в этом его «прелесть»

PDmitriy 10 сентября 2008 в 21:48 # −1

вербальная диарея… тот случай, когда копипаст предпочтительнее авторской статьи
www.osp.ru/os/2007/04/4220063/
www.citforum.ru/SE/project/scrum/
classicpm.wordpress.com/2006/09/14/scrum-part-1/

Steamus 10 сентября 2008 в 21:58 # ↑ +1

В принципе моей целью не было скопипастить чужие статьи. Цель несколько иная. Пояснив кратко принципы XP и Скрама, попытаться, к третьей части вынести дискуссию в критическую плоскость. Мне показалось нелогичным давать ссылку на чужую статью и затем критиковать. Я решил, что будет уместнее вначале изложить основы своими словами.

PDmitriy 10 сентября 2008 в 22:05 # ↑ 0

Два топика — это не кратко.

Steamus 10 сентября 2008 в 22:14 # ↑ +2

Я в общем три планировал. Похоже, что я измеряю информацию не количеством топиков. Но с другой стороны, вроде как и читать насильно я никого не заставляю. Тем более я никого не заставляю оскорблять других. Дело в том, что на написание заметки я трачу исключительно и только своё собственное время. :-)

subecho 11 сентября 2008 в 00:25 # ↑ +1

Не понимаю чего вы прицепились. Лично мне, немного знакомому с методологией, подобное не-академическое изложение понравилось — а это важно потому что читать сухие тексты по управлению проектами не каждый разработчик осилит :)

PDmitriy 11 сентября 2008 в 00:29 # ↑ 0

Да я вообще-то тоже малознаком с методологией. Именно поэтому я и обрадовался заголовку — хотел узнать больше. А что прочитал? Ноль по сути. И много красивых оборотов.

Steamus 11 сентября 2008 в 00:42 # ↑ 0

Я видимо закончу сегодня чуть детальнее про непосредственно Скрам. Мне не хотелось описывать технологические моменты до пояснения принципов. Будет просто неясно, зачем придуманы те или иные ухищрения.

ClintEastwood 11 сентября 2008 в 11:03 # ↑ 0

Может стоит задуматься над «красивыми оборотами»: где их можно применить, как их можно применить для своего проекта; а не ожидать что все разжуют и покажут как надо делать? ;)
Имхо, информация именно в таком неформальном виде наимолее «усваяемая».

subecho 11 сентября 2008 в 14:47 # ↑ 0

Кстати не встречал еще здесь упоминания про ПО для автоматизации процесса Scrum-разработки. Я лично знаком со ScrumWorks. Есть фришная версия. Подробности здесь: danube.com/scrumworks/basic

wayly 11 сентября 2008 в 23:14 # ↑ 0

Так читайте комментарии. Весьма интересная дискуссия… ТС, думаю, не будет против мысли, что обсуждение дает бОльшую пользу, нежели статья? :)

НЛО прилетело и опубликовало эту надпись здесь

ksurent 11 сентября 2008 в 00:07 # +1

Панацея ли Scrum?
Панацеи нет

Castro 11 сентября 2008 в 01:05 # 0

То, что Вы описали очень напоминает принцип «Quick and Dirty».
Работаю в проекте управляемом SCRUM уже 10 месяцев и статьи по нему читал, но пока конкретно о нём не писалось — подожду до следующей статьи с коментариями.

А у бывалых SCRUM-щиков и SCRUM-мастеров хотел спросить следующее.
Совместим ли SCRUM со свободным графиком работы? Возьмем книжную длительность спринта в 30 дней(27 полных дня +3 дня на риски). Реально ли огранизовать работу в офисе для небольшой команды со свободным графиком и какие принципы могут помочь, для эффективной работы? Как производить планирование в этом случае? Или для такой динамичной технологии управления свободный график губителен?

juks 11 сентября 2008 в 01:40 # 0

Скрам + нагруженный проект = коллапс

beskov 11 сентября 2008 в 02:23 # ↑ +3

А, так вот почему Хабр периодически nginx 500 выдаёт! Всё дело в волшебном Скраме!

juks 11 сентября 2008 в 12:36 # ↑ 0

Смех смехом, товарищи, но любым кодом кроме 200 и 301 ресурс очень сильно обязан этому самому скраму. К сожалению, у меня на хватает времени и идей, как этому лучше противостоять.

Если говорить о «внутреннем заказчике» то если взять зачаточный проект с сомнительным будущим, где квалификация и техническое руководство не играют роли, когда надо в длинный срок что-то хоть как-то сделать, а работать ой как не хочется, то скрам здесь очень кстати.

Скраму не место там где работают опытные, ответственные люди, вот моё мнение. Не место ему и там где случаются стрессовые ситуации. А рефакторинг в смысле скрама это такая несбыточная сказка или же попросту двойной труд.

Steamus 11 сентября 2008 в 03:17 # +2

Тут, парни, появилась вторая часть:

habrahabr.ru/blogs/pm/39534/

Налетаем, критикуем, пинаем ногами. Скрам, значит скрам.

Но вежливо. Что бы, бля, не поранить мою тонкую душевную организацию!

mrTempl 11 сентября 2008 в 12:35 # 0

А если не получается так часто встречаться с заказчиком? отправить письмом? а кто сказал что он там в течении дня ответить? названивать и просить посмотреть проект? потом сказать свой отзыв? опять же не всегда так может получиться.

и вот прос заклушки, заменяемые потом реальным кодом не очень понтяно.

veter 11 сентября 2008 в 17:32 # ↑ +1

А если у меня шурупы с плоской шляпкой и без резьбы, как их закручивать отверткой?

Инструмент, как и методологию, надо выбирать соответственно проекту и его условиям. И если заказчик методологию не приемлет, то ничего у вас не получится. Навязать ему определенное поведение вряд ли возможно.

egorinsk 12 сентября 2008 в 00:00 # 0

А не страдает ли при таком подходе качество кода и не получается ли продукт, попросту говоря, тормозным (и глючным)?

egorinsk 12 сентября 2008 в 00:01 # ↑ 0

К тому же автоматизированными тестами не меряют юзабилити и производительность.

toxa_xxl 13 сентября 2008 в 11:16 # 0

Поправьте пожалуйста теги. У Вас три тега слились в один.

Anna_Avvakumova 19 мая 2011 в 18:44 # 0

В общем-то, комментарии интересней, чем сама статья.
Вот и ответ на все вопросы о полезности поста.

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.