bishop3000 карма 184,9 465 голосов рейтинг 18,5

21 июля 2009 в 11:12

Канбан в IT (Kanban Development)

Разработка*

Я собираюсь написать несколько статей про новую методологию гибкой разработки Канбан (Kanban Development) в целях подготовки к Scandinavian Agile Conference 2009, где я буду делать один из докладов (кстати, заодно приглашаю всех на конференцию).
Сегодня публикую первую из статей.
Основная задача первой статьи — это как можно проще описать основы Канбан: что это такое, в чем отличие от других гибких методологий и зачем это нужно.
Также я хотел бы собрать как можно больше вопросов и сомнений в комментариях, чтобы ответить на них в следующих статьях, так что пишите всё, что вам непонятно, или что ещё вы хотели бы узнать про Канбан.
Я не то, чтобы большой специалист по этой новой методологии, но мы внутри команды пришли к Канбану самостоятельно и последовательно прошли все этапы мутации от SCRUM до Канбан, так что практический опыт есть.

Для начала напишу про происхождение термина **Канбан**.

Этот термин пришёл к нам из Японии благодаря широко известной в узких кругах производственной системе Тойота. Хотелось бы, чтобы как можно больше людей прочитало про эту систему и основные принципы, заложенные в неё — бережливое производство, постоянное развитие, ориентацию на клиента и т.п. Все эти принципы описаны в книге Тайити Оно Производственная система Тойоты, которая переведена на русский.

Термин Канбан имеет дословный перевод: “Кан” значит видимый, визуальный, и “бан” значит карточка или доска. На заводах Тойота карточки Канбан используются повсеместно для того, чтобы не загромождать склады и рабочие места заранее созданными запчастями. Например, представьте, что вы ставите двери на Тойоты Короллы. У вас около рабочего места находится пачка из 10 дверей. Вы их ставите одну за другой на новые машины и, когда в пачке остается 5 дверей, то вы знаете, что пора заказать новые двери. Вы берете карточку Канбан, пишете на ней заказ на 10 дверей и относите ее тому, кто делает двери. Вы знаете, что он их сделает как раз к тому моменту, как у вас закончатся оставшиеся 5 дверей. И именно так и происходит — когда вы ставите последнюю дверь, прибывает пачка из 10 новых дверей. И так постоянно — вы заказываете новые двери только тогда, когда они вам нужны.
А теперь представьте, что такая система действует на всём заводе. Нигде нет складов, где запчасти лежат неделями и месяцами. Все работают только по запросу и производят именно столько запчастей, сколько запрошено. Если вдруг заказов стало больше или меньше — система сама легко подстраивается под изменения.

Основная задача карт Канбан в этой системе — это уменьшать количество «выполняющейся в данный момент работы» (work in progress).
Например, на всю производственную линию может быть выделено ровно 10 карточек для дверей. Это значит, что в каждый момент времени на линии не будет больше 10 готовых дверей. Когда заказывать новые двери и сколько — это задача для того, кто их устанавливает. Только он знает свои потребности, и только он может помещать заказы производителю дверей, но он всегда ограничен числом 10.
Этот метод Бережливого производства (Lean manufacturing) был придуман в Тойоте и сейчас многие производственные компании по всему миру его внедряют или уже внедрили.

Но это всё относится к производству, а не к разработке программного обеспечения.
А что же такое Канбан разработка применительно к ПО, и чем она отличается от других гибких методологий, буть то SCRUM или XP?

Во-первых, нужно сразу понять, что Канбан — это не конкретный процесс, а система ценностей. Как, впрочем, и SCRUM с XP. Это значит, что никто вам не скажет что и как делать по шагам.
Во-вторых, весь Канбан можно описать одной простой фразой — **«Уменьшение выполняющейся в данный момент работы (work in progress)»**.
В-третьих, Канбан — это даже еще более «гибкая» методология, чем SCRUM и XP. Это значит, что она не подойдет всем командам и для всех проектов. И это также значит, что команда должна быть еще более готовой к гибкой работе, чем даже команды, использующие SCRUM и XP.

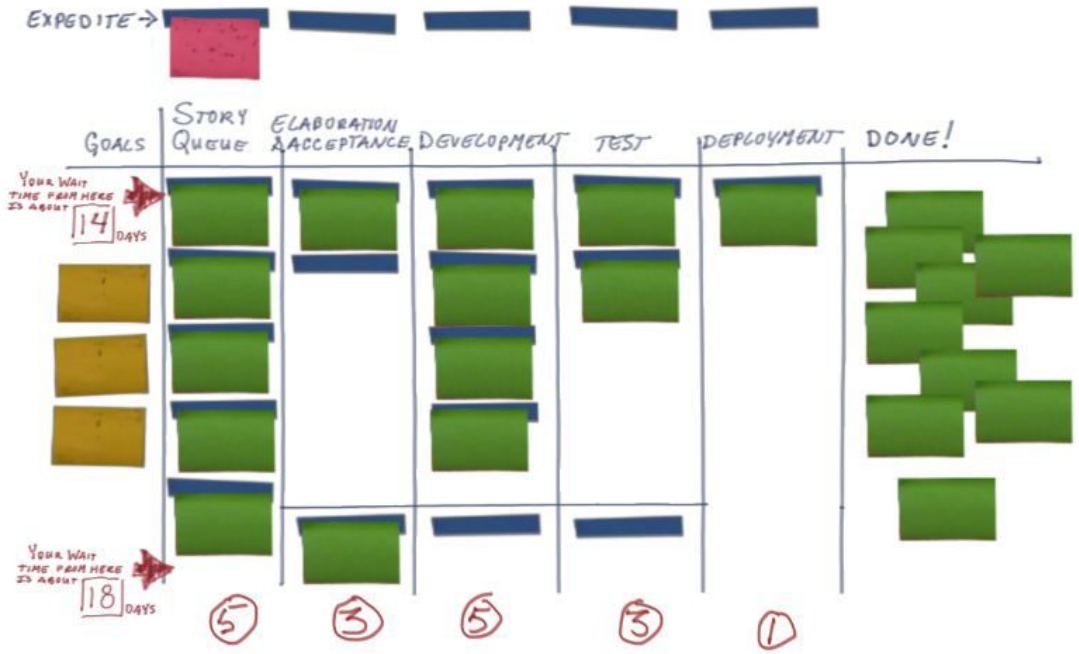
Разница между Канбан и SCRUM:
— В Канбан нет таймбоксов ни на что (ни на задачи, ни на спринты)
— В Канбан задачи больше и их меньше

— В Канбан оценки сроков на задачу опциональные или вообще их нет
— В Канбан «скорость работы команды» отсутствует и считается только среднее время на полную реализацию задачи

А теперь посмотрите на этот список и задумайтесь — что остается от гибкой методологии, если мы удаляем спринты, увеличиваем размеры задач и перестаем мерять скорость работы команды? Ничего?
Как вообще можно говорить о контроле за разработкой, если мы убираем основные инструменты контроля — сроки, скорость работы и спринты? Для меня этот вопрос является чуть ли не самым важным.
менеджеры всегда думают о контроле и пытаются его получить, хотя на самом деле никогда его не имеют.
Контроль разработки со стороны менеджера — это фикция. Если команда не хочет работать, то как ее не контролируй, она провалит проект.
Если команда получает фан от работы и работает с полной отдачей, то никакой контроль и не нужен, а только мешает, увеличивает издержки.
Например, общеизвестная проблема SCRUM — это большие издержки от обсуждений, встреч и большие потери времени на стыках спринтов (когда как минимум день уходит на закрытие одного спринта, а потом день на открытие нового. И если спринт — 2 недели, то 2 дня из 2 недель — это 20%, чертовски много). В итоге чуть ли не 30-40% времени при применении SCRUM тратится на поддержание самого процесса — на ежедневные митинги, на 5% workshop, на спринт ретроспектив и т.п. 30%!

Канбан разработка отличается от SCRUM в первую очередь ориентацией на задачи. Если в SCRUM основная ориентация команды — это успешное выполнение спринтов (надо признать, что это так), то в Канбан на первом месте задачи.
Спринтов никаких нет, команда работает над задачей с самого начала и до завершения. Деплоймент задачи делается тогда, когда она готова. Презентация выполненной работы — тоже. Команда не должна оценивать время на выполнение задачи, ибо это имеет мало смысла и почти всегда ошибочно вначале.
Если менеджер верит команде, то зачем иметь оценку времени? Задача менеджера — это создать приоритизированный пул задач, а задача команды — выполнить как можно больше задач из этого пула. Всё. Никакого контроля не нужно. Всё, что нужно от менеджера — это добавлять задачи в этот пул или менять им приоритет. Именно так он управляет проектом.

Команда для работы использует Канбан-доску. Например, она может выглядеть так (взял тут):



Столбцы слева направо:

Цели проекта:
Необязательный, но полезный столбец. Сюда можно поместить высокоуровневые цели проекта, чтобы команда их видела и все про них знали. Например, «Увеличить скорость работы на 20%» или «Добавить поддержку Windows 7».

Очередь задач:
Тут хранятся задачи, которые готовы к тому, чтобы начать их выполнять. Всегда для выполнения берется верхняя, самая приоритетная задача и ее карточка перемещается в следующий столбец.

Проработка дизайна:
этот и остальные столбцы до «Закончено» могут меняться, т.к. именно команда решает, какие шаги проходит задача до состояния «Закончено».

Например, в этом столбце могут находиться задачи, для которых дизайн кода или интерфейса еще не ясен и обсуждается. Когда обсуждения закончены, задача передвигается в следующий столбец.

Разработка:
Тут задача висит до тех пор, пока разработка фичи не завершена. После завершения она передвигается в следующий столбец.
Или, если архитектура не верна или не точна — задачу можно вернуть в предыдущий столбец.

Тестирование:
В этом столбце задача находится, пока она тестируется. Если найдены ошибки — возвращается в Разработку. Если нет — передвигается дальше.

Деплоймент:
У всех проектов свой деплоймент. У кого-то это значит выложить новую версию продукта на сервер, а у кого-то — просто закомитить код в репозиторий.

Закончено:
Сюда стикер попадает только тогда, когда все работы по задаче закончены полностью.

В любой работе случаются срочные задачи. Запланированные или нет, но такие, которые надо сделать прямо сейчас. Для таких можно выделить специальное место (на картинке отмечено, как «Expedite»). В Expedite можно поместить одну срочную задачу и команда должна начать ее выполнять немедленно и завершить как можно быстрее. Но может быть только одна такая задача! Если появляется еще одна — она должна быть добавлена в «Очередь задач».

А теперь самое важное. Видите цифры под каждым столбцом? Это число задач, которые могут быть одновременно в этих столбцах. Цифры подбираются экспериментально, но считается, что они должны зависеть от числа разработчиков в команде.
Например, если вы имеете 8 программистов в команде, то в строку «Разработка» вы можете поместить цифру 4. Это значит, что одновременно программисты будут делать не более 4-х задач, а значит у них будет много причин для общения и обмена опытом. Если вы поставите туда цифру 2, то 8 программистов, занимающихся двумя задачами, могут заскучать или терять слишком много времени на обсуждениях. Если поставить 8, то каждый будет заниматься своей задачей и некоторые задачи будут задерживаться на доске надолго, а ведь главная задача Канбан — это уменьшение времени прохождения задачи от начала до стадии готовности.
Никто не даст точный ответ, какие должны быть эти лимиты, но попробуйте для начала разделить число разработчиков на 2 и посмотреть, как это работает в вашей команде. Потом эти числа можно подогнать под вашу команду.
Под «разработчиками» я понимаю не только программистов, но и других специалистов. Например, для столбца «Тестирование» разработчики — это тестеры, т.к. тестирование — это их обязаность.

Задачи на такой доске — это не просто задачи, а то, что называется Минимальной Маркетинговой Фичей, то есть фича, которую можно «продать» клиентам.
Хорошая проверка для ММФ — это вопрос себе «А стал бы я писать про эту фичу в блоге компании?». Если нет — это не ММФ.

Что нового и полезного дает такая доска с лимитами?

Во-первых, **уменьшение числа параллельно выполняемых задач сильно уменьшает время выполнения каждой отдельной задачи.** Нет нужды переключать контекст между задачами, отслеживать разные сущности, планировать их и т.д. — делается только то, что нужно. Нет нужды устраивать спринт планнинги и 5% воркшопы, т.к. планирование уже сделано в столбце «очередь задач», а детальная проработка задачи начинается ТОЛЬКО тогда, когда задача начинает выполняться.

Во-вторых, **сразу видны затыки.** Например, если тестеры не справляются с тестированием, то они очень скоро заполнят весь свой столбец и программисты, закончившие новую задачу, уже не смогут переместить ее в столбец тестирования, т.к. он заполнен. Что делать? Тут время вспомнить, что «мы — команда» и решить эту проблему. Например, программисты могут помочь тестерам завершить одну из задач тестирования и только тогда передвинуть новую задачу на освободившееся место. Это позволит выполнить обе задачи быстрее.

В-третьих, можно вычислить время на выполнение усредненной задачи. Мы можем помечать на карточке дату, когда она попала в очередь задач, потом дату, когда ее взяли в работу и дату, когда ее завершили. По этим трем точкам для хотя бы 10 задач можно уже посчитать среднее время ожидания в очередь задач и среднее время выполнения задачи. А из этих цифр менеджер или product owner может уже рассчитывать всё, что ему угодно.

Весь Канбан можно описать всего тремя основными правилами:
1. **Визуализируйте производство**
— Разделите работу на задачи, каждую задачу напишите на карточке и поместите на стену или доску.

— Используйте названные столбцы, чтобы показать положение задачи в производстве.
2. **Ограничивайте WIP** (work in progress или работу, выполняемую одновременно) **на каждом** этапе производства.
3. **Измеряйте время цикла** (среднее время на выполнение одной задачи) и **оптимизируйте постоянно процесс**, чтобы уменьшить это время.

Всего 3 правила!
Например, в SCRUM — 9 базовых правил. В XP — 13, а в классическом RUP — аж более 120. Почувствуйте разницу.

На этом я закончу первую статью про Канбан.
Жду ваших отзывов и комментариев, а также пожеланий к следующим статьям.

Дополнительные ссылки (к сожалению всё только по-английски):

- Toyota production system
- The Poppendieks on Lean Software Development
- Software by Numbers
- David Anderson on Kanban
- Corey Ladas on Scrumban
- Arlo Belshee's Naked Planning
- Aaron Sanders Kanban ground rules
- InfoQ article from Kenji Hiranabe on Kanban
- My simple Powerpoint presentation on Kanban

методологии разработки, канбан, agile

+55 80442 308 bishop3000 18,5

Комментарии (91)

NikitaG 21 июля 2009 в 11:30 # +8

Вот из-за таких вот идей руковдство и покупает дурацкие доски с наклейками вместо настольного тениса и кофемашины.=)

bishop3000 21 июля 2009 в 12:05 # +3

У нас и доски, и кофемашины, и спортзал, и чего только нет :)
Если методология подходит команде и команда работает лучше, то она приносит больше денег фирме, а значит фирма тратит больше денег на всякие полезные мелочи.

zorba_buddha 5 ноября 2013 в 16:58 # 0

Полезная рекурсия!

askarmuk 25 июля 2009 в 05:00 # 0

Значит, руководство видит доску и не видит идею. Жаль.

genixg 16 апреля 2015 в 07:35 # 0

Вчера внедрял Kanban в нашей небольшой организации, доска + канцелярские принадлежности вышло в 528 рублей, да и то, стикеры можно было найти дешевле, а не покупать их за 220 рублей. А сама доска — лист пленки «оракал» 2 метра (220 р), наклеенной прямо на стену, + 2 кольца черной изоленты по 30р.
Теннисный стол стоит дороже)

neuymin 21 июля 2009 в 11:45 # +1

Круто. Почтав про Scrum, я сделал доску, выкинув то, что быстро и безболезнено не втыкалосб. Оказывается, я внедрил Канбан :-)

bishop3000 21 июля 2009 в 12:07 # +1

Так оно обычно и получается — дял реально гибких разработчиков даже Scrum недостаточно гибок, да и не для каждой работы он подходит, так что начинают его упрощать. А тут уже и Канбан.
Но основная идея Канбана — ограничивать число «работы в данный момент» на каждом этапе разработки. Без этого Канбан -не совсем Канбан.

ldmitry 21 июля 2009 в 19:20 # +3

"… для реально гибких разработчиков даже Scrum недостаточно гибок" — главное, чтобы под маской этой «гибкости» не скрывался хаос в проекте, что бывает в большинстве случаев.

 **da_elf** 25 июля 2009 в 09:17 # ↑ 0

Вы менеджер?

 **Elvalery** 21 июля 2009 в 12:15 # 0

Интересная методология.
Вопрос автору: если нет временнных отметок, как тогда продаются эти проекты?

 **bishop3000** 21 июля 2009 в 12:22 # ↑ +1

>> если нет временнных отметок, как тогда продаются эти проекты?

А как временные отметки (итерации, спринты) связаны с продажей? Итерации в Scrum или XP — это просто квант работы команды. А проект содержит результат работы нескольких команд и в течение нескольких итераций.
В случае с Канбан никаких итераций нет — есть задачи. Они постепенно выполняются, проект наполняется фичами. В какой-то момент менеджер или product owner может решить, что проект достаточно хорош и готов к релизу и «продать» его.
Либо же просто ждать, пока самые важные фичи не будут выполнены.

 **acerv** 21 июля 2009 в 16:59 # ↑ +3

Что вы говорите. Одно из понятий скрама — фокус-фактор, который говорит, как эффективно работает команда и сколько она способна решить задач за квант! Квант здесь — измеримое понятие, т.е. бизнес-людям можно оценить сверху, будут ли реализованы все важные запланированные задачи к сроку поставки, или же нужно сокращать функционал.

В канбане я этого не вижу. Т.е. с первого взгляда — это такая текучка, очень хорошая для постоянного процесса (как непрерывное производство машин, например), но для производства ПО — это как-то ооооочень гибко.

Раз уж решили сравнивать скрам, то я вот что скажу — скрам хорош тем, что он защищает разработчиков от оголтелого произвола менеджеров, или же наоборот, позволяет менеджерам видеть, кто саботажник или просто плохо работает. Это позволяет скорректировать планы и той, и другой стороне.

В канбане этого нет. Что я менеджеру скажу? Я не могу ему сказать, что ему надо убрать пять задач, потому что мы их тупо не успеем сделать — у нас статистикой доказано, что средний фокус фактор, например, 48%. Менеджер от меня ждет оценку (от менеджера этого требуют заказчики) — мне нужно дать ему оценку, пусть даже вы правы насчет тем, кто решает успех проекта.

В канбане нет фокус фактора, весь процесс производства — постоянная текучка, стало быть менеджер будет скакать вокруг меня и орать, чтобы мы работали быстрее, будет пытаться впихнуть больше задач единовременно или увеличить емкость семафора с задачами. Говорить ему при этом «Дядя, че ты ко мне лезешь, неужели ты не знаешь, что от тебя в разработке проекта ничего не зависит?», как было заявлено в статье, это моветон, меня дядя в 24 часа уволит за некомпетентность.

 **bishop3000** 21 июля 2009 в 17:21 # ↑ 0

Квант здесь — измеримое понятие, т.е. бизнес-людям можно оценить сверху, будут ли реализованы все важные запланированные задачи к сроку поставки, или же нужно сокращать функционал.

Так ли это по сути?
Вы можете рассчитать этот квант и соотношение запланированного к выполненому. Вы можете знать, сколько команда выполнит в следующем спринте, исходя из этих данных.
И это всё!

Что вы знаете про весь проект с этими данными? Ведь чтобы оценить весь проект, надо оценить каждую фичу в проекте, причем довольно точно. Если неточно оценил — грош цена такой оценке и это тоже самое, что и менеджер на коленке бы прикинул, что успеет сделаться, а что нет.
Собственно, гибкая разработка как раз о том, что нифига мы не можем прогнозировать точно и должны быть просто готовы зарелизить тогда, когда надо, а что не успели — отрезать.
Точные прогнозы — это к более тяжелым процессам, а не к agile.

В канбане все тоже самое — есть примерный скоуп проекта и команда его выполняет максимально быстро. product owner следит за скоупом и за временем выполнения одной задачи — из этого он делает выводы, сколько будет выполнено к дедлайну и меняет приоритеты, если что.
Контроль не хуже, чем в скраме — время выполнения задачи можно сравнивать и задача команды — сокращать это время. Если оно растет — что-то не так.

стало быть менеджер будет скакать вокруг меня и орать, чтобы мы работали быстрее, будет пытаться впихнуть больше задач единовременно или увеличить емкость семафора с задачами.

Ну, если менеджер — обезьяна, то тут и скрам не поможет. Обезьяна все равно скакать вокруг постоянно будет :)

 **acerv** 21 июля 2009 в 17:52 # ↑ 0

>Ну, если менеджер — обезьяна, то тут и скрам не поможет.

Именно что поможет. Наша постоянно лезла в разработку, «управляла». Нейтрализовало на 100%.

>Что вы знаете про весь проект с этими данными?
Зависит от срока и размера проекта. Если проект мелкий — средний, то всю картину можно разбить на бизнес-задачи. Если он очень большой (как у нас сейчас, например), то берется некоторый актуальный зазор задач, определяемый на встречах с заказчиком — своего рода «мелкий» проект, без особых четко очерченных границ. В любом случае, я хочу сказать, что «знать» про проект — можно. Составление бэклога с последующей приоретизацией и межкомандным согласованием — это очень важная задача. Блин, да что я вам говорю, про это классики пишут в почти каждой книжке.

>когда надо, а что не успели — отрезать.
Вы знаете, получается что в канбане так — ВДРУГ всем стало ясно, что не успели. Работали работали, и не успели. Итерация дает обратную связь, позволяет изменять процесс и делать его прозрачным.

>Контроль не хуже, чем в скраме — время выполнения задачи можно сравнивать и задача команды — сокращать это время. Если оно растет — что-то не так.

Вот этого я не понимаю. Спринт закончился — если остались задачи, спринт провалился, устраивается разбор полетов. В канбане задача может провисеть мертвой некоторое время, пока кто-нибудь не спохватится — «ой, что это у нас тут».

>и за временем выполнения одной задачи — из этого он делает выводы, сколько будет выполнено к дедлайну
По-моему, это некорректно. Задачи ведь неравномерны по времени своего исполнения!

>Если оно растет — что-то не так.
В статье писалось, что такого аспекта как замеры времени в процессе не существует:)))

Ладно, хватит придирок. Дело в том, что если сравнивать скрам с канбаном, если вы признаете последний набором методик, а не процессом разработки, некорректно. Если же брать канбан как процесс в том виде, в котором он приведен в статье — то канбан нежизнеспособен в жестких условиях поставки.

 **bishop3000** 21 июля 2009 в 18:02 # ↑ 0

Наша постоянно лезла в разработку, «управляла». Нейтрализовало на 100%.

Значит все-таки не обезьяна :)

Блин, да что я вам говорю, про это классики пишут в почти каждой книжке.

Все зависит от команды и проекта. Для каких-то проектов планирование и согласование — важнейшая часть проекта. Для других это ненужная трата времени. Есть отличная статья недавняя Тома Демарко — советую прочитать. И учтите, что он был одним из апологетов контроля за разработкой.

Дело в том, что если сравнивать скрам с канбаном, если вы признаете последний набором методик, а не процессом разработки, некорректно. Если же брать канбан как процесс в том виде, в котором он приведен в статье — то канбан нежизнеспособен в жестких условиях поставки.

Всё верно написано. В жестких условиях может не подойти. И это не процесс, а надстройка над процессом — над тем же скрамом или xp.

 **vzzvzz** 23 июля 2009 в 19:38 # ↑ 0

Комментарий заставил задуматься над тем как я работаю со своей командой, спасибо…
Фокус-фактор штука мощная, когда она есть. Но, если в команде работают люди с сильно разной квалификацией, он превращается в среднюю температуру по палате.
Вопрос «будут ли ВСЕ важные запланированные задачи готовы к сроку поставки» не всегда является центральным. Подход может быть другим, например: «Вот задачи A, B, C. Скажите нам, что из них когда мы будем продавать». Получается поток задач и это не плохо само по себе. Всего лишь вопрос выбора бизнес-модели.
Впихивания задач в канбане не произойдет, если менеджер (мы кстати о каком менеджере говорим? о менеджере продаж надеюсь?) принимает условие, что количество открытых задач в потоке конечно, и чтобы что то добавить — надо что то убрать. Как оценить это количество? Тут фокус-фактор сам по себе меня никогда не спасал. На самом деле эту оценку всегда приходится проводить и доказывать заново, как только кому то надо подвинуть сроки или расширить объем работ. Можете считать индивидуальный фокус-фактор для каждого члена команды, и давать оценки в зависимости от того, кто конкретно будет делать работу.

 **acerv** 23 июля 2009 в 19:51 # ↑ 0

Что-то вы усложняете. Какая еще средняя температура по больнице? Есть спринт в днях, есть задачи, есть их оценка в идеальных днях. Устанавливаете фокус-фактор, умножаете и отрезаете те задачи, которые не вмещаются в спринт. Если сделали все задачи — угадали с фокус-фактором, не сделали — значит, фокус фактор был ниже и на следующий спринт его надо скорректировать. Сделали больше — круто, значит, фокус-фактор на следующий спринт можно как-то увеличить.

Здесь квалификация отдельного человека не причем, идет оценка работы всей команды как единого целого.

 **vzzvzz** 23 июля 2009 в 20:55 # ↑ 0

Может и усложняю, но я полагал что фокус-фактор — отношение идеальных дней к реальным в последнем спринте. На практике в команде могут быть настолько по разному подготовленные люди, и задачи можно раздать этим исполнителям так, что получим довольно разные значения фокус-фактора для одного и того же спринга. Вплоть до нуля.

acerv 23 июля 2009 в 23:04 # ↑ 0

В общем, я вам объяснять не буду, вы просто не совсем четко понимаете, как это все работает.

Вам следует прочитать вот эту книгу:
scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf

Там всего 100 страниц и все на русском. Не поленитесь, это очень полезно.

vzzvzz 26 июля 2009 в 00:59 # ↑ 0

все 100 на русском? :-) а картинки перевести забыли, без картинок неподготовленным читателям тяжко разобраться

Фокус-фактор — во многих случаях весьма приблизительный инструмент. Есть много факторов посерьезнее болезней и отгулов, которые не позволят оценить эффективность комманды с помощью этого инструмента. Наверное, есть такие проекты, где ситуация остаается предсказуемой достаточно долгое время, чтобы можно было стабилизировать фокус-фактор. Но точно есть и другие проекты.

acerv 27 июля 2009 в 10:52 # ↑ 0

Картинки там переведены.

В книжке рассмотрены приводимые вами ситуации. Вы бы почитали сначала, а потом умничали.

vzzvzz 27 июля 2009 в 11:39 # ↑ 0

Прочитал, рецепты из книжки не подходят.

Napolsky 21 июля 2009 в 12:49 # +8

Вот из-за таких статей я и люблю хабр! Все по делу, грамотно, профессионально, интересно, актуально. Автор — молодец. Спасибо за статью!

alexsun 21 июля 2009 в 13:26 # +5

Почитайте оригинальные книги по канбан 70-ых годов и вы поймёте, что канбан притянут за уши в разработку программного обеспечения. Как минимум по двух причинам. Одно из обязательных условий для того, чтобы TPS работал является постоянность временных интервалов связанные с выполнением определённых процедур. Именно поэтому в разрезе TPS планирование — это «waste» (потеря, лишняя трата времени).
Это один из китов на котором основан ТПС и Канбан. А так как такое требование никогда не может быть выполнены в разработке (вы в 30-40 процентов никогда не скажете точно сколько времени уйдёт на исправление того или иного бага), то мы приходим к выводу, что это всего лишь очередная маркетинговая замануха.

mikedin 21 июля 2009 в 13:57 # ↑ 0

плюс еще если речь идет о стартапе, то маркетинговые вещи должны начинаться заранее, а не отсчитывая от конца разработки, иначе теряется время. для автомобилей, причем устоявшейся линейке эти процессы могут идти параллельно, а в стартапе и вообще разработке тут же получим просос.

очень конечно красиво, но думать о внедрении любой методологии тоже надо. причем _до_ внедрения и оценивать не красивость\простоту\что-еще-там идеи, а то, как она реально наложится на существующие процессы :)

bishop3000 21 июля 2009 в 14:01 # ↑ +1

Я не говорю про TPS, я говорю про Канбан. Никто не пытается перенести всю систему работы в Тойоте в IT — это невозможно и не нужно. Да и само слово «Канбан» взято из TPS скорее потому, что красиво звучит :)

Основа у Канбан та же, что и у TPS — уменьшение work in progress, но это и всё. Что еще взято из TPS?

Канбан — это просто адаптация хорошей идеи к реалиям разработки ПО.

alexsun 21 июля 2009 в 14:30 # ↑ +1

Канбан, вырванный из TPS — это всего лишь практика и никак не методология.
Даже Хенриг Книберг в своём сравнении скрама и канбана (вдруг не читали, blog.crisp.se/henrikkniberg/2009/05/29/1243594140000.html) скатывается к откровенно премитивному выводу — вся суть «методологии» канбан — это контроль WIP-а, а всё остальное по вашему усмотрению. Опытному менеджеру и разрабочки при этом становится просто смешно.
Приведу вам хороший пример использования практики (именно практики, а не методологии) канбан.
«Не разрабатывайте фичей больше чем можете протестировать».

creadone 21 июля 2009 в 13:39 # 0

Спасибо, мы давно так не смеялись. Это удивительная методология! Шедевральо почти все, но особенно порадовало: «Например, в SCRUM — 9 базовых правил. В XP — 13, а в классическом RUP — аж более 120. Почувствуйте разницу».

creadone 21 июля 2009 в 13:40 # ↑ +1

Из зала добавили: копаем от забора и до заката

turbo_exe 21 июля 2009 в 13:43 # –2

я один неправильно прочитал слово канбан? оО

DmitryKoterov 21 июля 2009 в 15:32 # ↑ –2

Нет, я тоже прочитал его раза 4 неправильно, прежде, чем понял.

arilou_camper 21 июля 2009 в 14:28 # +1

1) Как данная методология решает основной вопросы на уровне топ-менеджмента: Когда в продакшене будет следующая стабильная версия системы?

Решение вижу только одно: манагер все равно должен трэкать частоту закрытия ММФ в период времени, получать project velocity, смотреть на burn-down, отчитываться перед руководством. Поэтому имею мнение, что:

— Методология подходит только для проектов с дедлайнами типа «when it's done» (таких меньшинство) -ИЛИ-
— Оценка временнЫх факторов решения задач все равно присутствует в неявной форме в ежедневной работе менеджера

2) Частично согласен, что кол-во параллельных задач в работе определяется кол-вом программистов. Частично, потому что это кол-во еще определяется индивидуальными качествами программистов, маркетинговыми потребностями, техническими взаимосвязями в задачах.

3) Несогласен, что контроль — это фикция. Либо вы как-то по-своему понимаете значение этого слова. В моем понимании, контроль — это _управляемость_ процессом разработки, т.е. возможность направлять его в нужное на данный момент русло. «Данный момент» в контексте бизнеса — недельный срок вполне приемлем.

В-общем, если резюмировать — agile с недельными спринтами и оценками задач при планировании более предсказуем для команды, которая уже хотя бы один проект по такой методике имеет за плечами.

bishop3000 21 июля 2009 в 14:55 # ↑ +1

Как данная методология решает основной вопросы на уровне топ-менеджмента: Когда в продакшене будет следующая стабильная версия системы?

А как любая другая методология решает этот вопрос? Фактически никак — ставится срок на основе каких-то предварительных оценок и проект движется к этому сроку. Что успели сделать — то входит в состав проекта. Что не успели — не входит, либо срок передвигается.
Какую методологию не используй, точно спрогнозировать вначале не сможешь. Можно только буферы задать побольше, чтобы подстраховаться.
В Канбане тоже самое, только измерение времени идет по скорости разработки одной задачи. Если на проект есть 100 задач и мы знаем, что команда в среднем делает задачу за неделю и параллельно может делать до 5 задач, то можем считать, что через 20 недель все будет готово.

— Методология подходит только для проектов с дедлайнами типа «when it's done» (таких меньшинство)

Не только. Но для работы над проектами с жестким дедлайном и жестким набором фич эта методология плохо подходит — факт.

— Оценка временнЫх факторов решения задач все равно присутствует в неявной форме в ежедневной работе менеджера

Конечно присутствует. Иначе зачем мерять время на выполнение одной задачи?
Менеджер (product owner) по-прежнему должен общаться с маркетингом и другими заинтересованными лицами и должен следить за конечным сроком. Инструменты для этого в Канбан есть — это время на выполнения 1 задачи и число задач, выстроенных по приоритетам.

2) Частично согласен, что кол-во параллельных задач в работе определяется кол-вом программистов. Частично, потому что это кол-во еще определяется индивидуальными качествами программистов, маркетинговыми потребностями, техническими взаимосвязями в задачах.

Да, ограничение на число параллельно выполняемых задач не жесткое и зависит от команды и других условий. Команда вольна сама менять эти ограничения, если видит, что может работать эффективнее, увеличив или снизив их.

В моем понимании, контроль — это _управляемость_ процессом разработки, т.е. возможность направлять его в нужное на данный момент русло.

В моем тоже. И тут Канбан как нельзя лучше подходит к проектам, где изменения происходят очень часто. Например — поддержка (maintenance). Или любая другая разработка, где 2 недели на спринт — это слишком много, т.к. изменения в

планы приходится вносить чуть ли не каждый день.

В-общем, если резюмировать — agile с недельными спринтами и оценками задач при планировании более предсказуем для команды, которая уже хотя бы один проект по такой методике имеет за плечами.

Не согласен. Чистый Agile (Scrum) с недельными спринтами будет иметь слишком большой оверхед на поддержку методологии — слишком много «обязательных» артефактов и обсуждений. Хотя опять же все зависит от команды и условий проекта, так что тут нет общего решения.

 **arilou_camper** 21 июля 2009 в 15:06 # ↑ +1

Ну в целом я с Вами согласен =) Каждому проекту своя методология.

 **vzzvzz** 23 июля 2009 в 21:02 # ↑ 0

И тут Канбан как нельзя лучше подходит к проектам, где изменения происходят очень часто.

Факт! И таких проектов не так уж мало. Когда в разгар работы приходит менеджер по продажам и говорит что нашел кучу денег, но надо срочно делать прототип — моя команда садится и делает. И приходится забыть про текущий спринг, рассчитанный фокус-фактор… И с кураторами проектов мне проще торговаться за то, какие задачи сейчас отложить, чтоб сделать что то внезапно-срочное, чем доказывать что сначала надо закончить текущий спринг. И зачастую это единственнный выход!

 **alexsun** 21 июля 2009 в 14:32 # +1

Кстати, 8 месяцев назад общались мы с Robin Dymond (CST), которые сейчас читает также тренинги по лин и канбану. Так вот спросили его потом за бутылочкой пивка, а ты хоть раз в жизни видел проект, который бы работал использую Канбан. Его ответ? «Нет»)))

 **bishop3000** 21 июля 2009 в 14:56 # ↑ 0

Я работал прошлые полгода с использованием Канбан. Проект вышел достаточно успешный и все остались довольны :)

 **alexsun** 21 июля 2009 в 15:08 # ↑ 0

То есть вы никогда не планировали? Просто сидели и что-то потихоньку делали?
Или как там из зала добавляли «копали от забора и до заката»?

 **bishop3000** 21 июля 2009 в 15:14 # ↑ 0

Да, так и было. На спринт планнинге мы не планировали конкретные работы, а просто общались с product owner-ами и выясняли, сколько мы должны времени потратить на работу на каждого из них (т.к. у нас было несколько product owner-ов с разными целями и задачами).
Далее каждый из них имел свой бэклог с приоритизированными задачами и мы просто брали оттуда задачи, когда заканчивали очередную и делали их. Планирования сроков никакого не было — только обсуждение технической стороны задачи, когда ее берешь.

Опять же повторюсь — если команда работает и хочет работать хорошо, то какое еще планирование нужно? Если в бэклоге 50 задач, то мы должны все их сесть и оценить? Или мы должны, как в Scrum, оценивать только на 2 недели вперед? Так через день придет новый баг и его надо исправлять. В Scrum это решается планированием «непредвиденных задач», но если непредвиденных — 50%, то это уже глупо становится.

 **sse** 21 июля 2009 в 15:22 # ↑ 0

А в канбан непредвиденных не бывает, получается?
Имхо если 50% непредвиденных — то это уже даже не управлятор рисками дал сбой, а в понимании задачи командой что-то не то, или команда представляет из себя не то, что думает лид. И методология тут уже ИМХО влияет мало.

 **bishop3000** 21 июля 2009 в 15:26 # ↑ 0

В Канбан нет непредвиденных задач, т.к. никто не планирует вперед. Задача обсуждается и принимается в работу только когда для нее появляется место. Поэтому если появляется новая задача — ее приоритизирует product owner и помещает в буфер задач, а оттуда она уже будет взята в разработку, когда команда доберется. А если приоритет очень высок, то можно и вверх списка поместить и получить выполненную задачу уже очень скоро.

Имхо если 50% непредвиденных — то это уже даже не управлятор рисками дал сбой, а в понимании задачи командой что-то не то, или команда представляет из себя не то, что думает лид. И методология тут уже ИМХО влияет мало.

В техподдержке крупной системы только так люди и работают — каждый день что-то новое :)
В разработке компьютерных игр или других сильно «креативных» программ тоже изменений очень много

постоянно.

 **sse** 21 июля 2009 в 15:52 # ↑ 0

>>В Канбан нет непредвиденных задач
Как это нет — поясните, пожалуйста? Он живет в другой реальности? Или имеется в виду, что в ваших терминах новая != непредвиденная?

>>никто не планирует вперед
В моем понимании в таком случае каждая задача — непредвиденная

>> вверх списка поместить и получить выполненную задачу уже очень скоро.
Чем это отличается от бэклога в scrum?

>>В разработке компьютерных игр или других сильно «креативных» программ тоже изменений очень много постоянно.
Если я правильно понимаю, то это — не очень хорошо и вряд ли позитивно сказывается на качестве.

 **bishop3000** 21 июля 2009 в 15:57 # ↑ 0

Или имеется в виду, что в ваших терминах новая != непредвиденная?

Да, новая != непредвиденная. Непредвиденные задачи — это когда у тебя распланирован спринт и все заполнено и вдруг приходит новая срочная задача. А ты сопротивляешься ей, т.к. внутри спринта задачи нельзя новые добавлять.

 **sse** 21 июля 2009 в 16:19 # ↑ 0

Я считаю, что это — хорошо, когда проект сопротивляется внедрению фич. Каждая фича должна доказать свою необходимость, ее необходимость будет выражена в приоритете. Тут же я воздержусь от спора, потому что мне пока непонятно, кто ставит приоритеты задачам в канбан.

 **bishop3000** 21 июля 2009 в 16:23 # ↑ 0

Приоритеты ставит product owner. И он же должен и фичам сопротивляться — он владелец проекта и он лучше всех остальных должен знать, что хорошо, а что плохо.

 **sse** 21 июля 2009 в 18:24 # ↑ 0

В случае IT, похоже, в канбан предлагается просто иметь пул задач, куда они все валятся — первый освободившийся берет задачу и делает. Освободился от текущей — делай следующую.

Это хорошо для Toyota, где технические аспекты не рассматриваются — есть хорошо известный процесс («приделывать двери»), который невозможно испортить (разве что делать медленно) и где не нужно экспериментировать, и вся затыка — только в ресурсах.

Кто будет думать над hi-lvl архитектурой, когда product owner лично назначает приоритеты? Где в данной схеме место тех.лида?

 **bishop3000** 21 июля 2009 в 23:11 # ↑ 0

>> Кто будет думать над hi-lvl архитектурой, когда product owner
>> лично назначает приоритеты? Где в данной схеме место
>> тех.лида?

Я могу сказать только про наш опыт. Мы разрабатывали архитектуру по большей части сами, то есть и тех. лид и архитектор находились в команде.
Когда были совсем высокоуровневые вопросы, то у нас в компании есть должность главного архитектора — шли к нему и обсуждали. Никаких проблем с архитектурой замечено не было.

 **bishop3000** 21 июля 2009 в 15:59 # ↑ 0

>> вверх списка поместить и получить выполненную задачу уже очень скоро.
Чем это отличается от бэклога в scrum?

Бэклог в скраме трогается только при планировании нового спринта и если туда что-то помещается, то в работу это пойдет только в след. спринте. А если спринт — 1 месяц, то это ой как нескоро. Да и 2 недели тоже.

>>В разработке компьютерных игр или других сильно «креативных» программ тоже изменений очень много постоянно.
Если я правильно понимаю, то это — не очень хорошо и вряд ли позитивно сказывается на качестве.

В целом — да. Но там по-другому нельзя — слишком много креатива, проб и ошибок.

 **sse** 21 июля 2009 в 16:21 # ↑ 0

Т.е. канбан — это скрам со сверхкраткими спринтами длиной иногда и в один день, с удалением всех активностей, которые не влезли в спринт, типа планирования/проектирования?

bishop3000 21 июля 2009 в 16:29 # ↑ 0

Канбан может быть использован вместе с Scrum или XP.
Канбан-команда может использовать любые артефакты из других методологий, т.к. Канбан — это не методология по сути, а набор ценностей, надстройка над методологией.

Вы можете по-прежнему делать TDD, парное программирование, daily митинги, даже может быть итерации — все, что угодно, что помогает вам двигать задачи по доске Канбан как можно быстрее.
Про конкретные методы и приемы разработки Канбан вообще ничего не говорит.

eugenius_nsk 22 июля 2009 в 17:44 # ↑ 0

Кстати, что происходит в канбан (ну, или в вашей интерпретации канбан), когда одна критическая высокоприоритетная задача уже есть, появляется ещё одна такая же — а очередь задач уже забита?

bishop3000 23 июля 2009 в 11:11 # ↑ 0

В классическом Канбан Product Owner должен решить, какую из задач в очереди убрать и куда поместить новую высокоприоритетную. Но он не может ее впихнуть в производство, т.к. только одну может.

У нас же проще — мы просто берем вторую в производство. Третью уже можем и не взять — тогда это задача product owner-а поменять список в очереди задач.

vzzvzz 23 июля 2009 в 21:13 # ↑ 0

Еще в канбан можно отдельно ввести ограничение на количество высокоприоритетных задач, такой «блэклог высокоприоритетных задач».Тут конечно могут быть трудности с такими product owner, которые считают все свои задачи приоритетными. Но штука полезная, препятствует бардаку в приоритетах, который может взорвать канбан

bishop3000 23 июля 2009 в 21:23 # ↑ 0

У меня сейчас в проекте намечается момент, когда канбан может взорваться как раз из-за множества высокоприоритетных задач, которые берутся в работу немедленно. Из-за этого могут откладываться старые задачи.

Пробую с этим справиться, но пока что таким образом, что product owner-у не отказываю, но людей со старых задач стараюсь не снимать — пытаюсь хэндлить их самостоятельно. То есть как бы есть специальный выделенный человек на «неожиданности».

alexsun 21 июля 2009 в 15:26 # ↑ 0

То что вы только что описали, уже как лет 10-20 имеет очень чёткое определение в мире управления разработкой программного обеспечения. И имя ему «Ad hoc development». По склассификации CMMI относится к уровню 2 и имеет крайне низкий уровень предсказуемости и управляемости.
www.ctg.albany.edu/publications/reports/survey_of_sysdev?chapter=4
А то что для старых дел придумали новомодное очередное название, ещё раз говорит о том, что это всего лишь наживка для армии agile-тренеров.

alexsun 21 июля 2009 в 15:29 # ↑ 0

извините, последнюю фразу нужно читать как «наживка, созданная армией agile-тренеров»

bishop3000 21 июля 2009 в 16:01 # ↑ 0

Это не совсем Ad hoc, т.к. есть бэклоги, приоритизация, оценки времени на выполнение задач, ограничение работы в прогрессе и т.д.
А так в целом — да, похоже на Ad hoc.

alexsun 21 июля 2009 в 16:09 # ↑ 0

В канбан — ничего не сказано про приоритезацию. Там сказано — делайте как хотите, можете вообще не приоритезировать. Про оценку времени тоже ничего не сказано. Она не нужна, оценка — это трата времени. Когда будет сделано, тогда и будет сделано.

Всё что есть в канбан — это ограничение WIP и всё.

bishop3000 21 июля 2009 в 16:12 # ↑ 0

Про приоретизацию в трех основных правилах не сказано, да. Наверное потому, что это не задача команды, а задача product owner-а, который является внешней силой.
Про оценку времени сказано в третьем правиле.

alexsun 21 июля 2009 в 16:15 # ↑ 0

«В Канбан оценки сроков на задачу опциональные или вообще их нет» — это ваши слова.
Дальше — в канбан нет роли PO. В канбан нет вообще ролей. И в отличае от скрама в канбан не говорится что именно он должен делать приоритезацию.
Такие вот «мелочи» и отличают настоящую методологию от набора «полезных советов»

bishop3000 21 июля 2009 в 16:19 # ↑ 0

А вот еще мои слова, подтверждающие ваши: :)
Во-первых, нужно сразу понять, что Канбан — это не конкретный процесс, а система ценностей. Как, впрочем, и SCRUM с XP. Это значит, что никто вам не скажет что и как делать по шагам.

alexsun 21 июля 2009 в 16:24 # ↑ 0

Так ты всё таки определись, что это. Потому что самое первое твоё предложение звучит так «Я собираюсь написать несколько статей про новую методологию гибкой разработки Канбан». Это во-первых, а во-вторых Канбан не может быть системой ценностей, потому что это место уже забито таким понятием как «lean».
Scrum и XP — это методологии.

alexsun 21 июля 2009 в 16:24 # ↑ 0

Читайте первоисточники.
jeffsutherland.com/oopsla/schwapub.pdf

Our new approach to systems development is based on both defined and black box process management. We call the approach the SCRUM methodology (see Takeuchi and Nonaka, 1986), after the SCRUM in rugby — a tight formation of forwards who bind together in specific positions when a scrumdown is called.

bishop3000 21 июля 2009 в 16:33 # ↑ 0

Да, первое предложение не очень-то верно.
Канбан — это не полноценная методология разработки. Это скорее надстройка над другими методологиями. Канбан добавляет в то, над чем он построен, некоторые плюсы, и это методология, но не полная.

А «lean» и канбан — это немного понятия разного уровня. lean — это высокоуровневая система ценностей, в которую канбан входит, как составная часть.

vlaskin 21 июля 2009 в 14:32 # 0

Перенесли бы в «Agile: Живая разработка» (agile)
ИМХО там оно уместнее :)

garex 21 июля 2009 в 14:52 # 0

Минус идее.

Канбан и прочее — для конвейера.

IT чуть менее чем полностью конвейер. Хотя в небольшой доле он конвейер.

И по-мойму изобрели его не в Японии, а в США. Японцэ его внедрили.

bishop3000 21 июля 2009 в 14:58 # ↑ 0

Вы похоже не поняли идею применения Канбан к IT. Конвеер — это в TPS (toyota production system), а в IT Канбан совсем о другом — о разработке новых больших фич (ММФ) как можно быстрее. Никакого конвеера нет и в помине.

garex 21 июля 2009 в 15:04 # ↑ 0

Новая большая фича = качество?
Быстрее = скорость?

=> Затраты стремятся к бесконечности.

| Контроль разработки со стороны менеджера — это фикция.
| Если команда не хочет работать, то как ее не контролируй,
| она провалит проект.

Вообще-то здесь управленца надо отвазелинить, который не знает такого слова, как мотивация. Надо различать инструменты.

Контроль — это пассивная штука, а мотивация — активная.

Контроль — это панель приборов, а мотивация — педаль.

 **GrigoryPerepechko** 11 июля 2012 в 16:04 # ↑ 0

IT чуть менее чем полностью конвейер. Хотя в небольшой доле он конвейер.

У вас лурчанка головного мозга

 **Urn** 21 июля 2009 в 15:22 # 0

очень понравилась статья.
обьясните пожалуйста, канбан является одной из методик agile разработки?

 **bishop3000** 21 июля 2009 в 15:27 # ↑ 0

Да, конечно. В самом начале написано: *новую методологию **гибкой** разработки Канбан*

 **acerv** 21 июля 2009 в 17:38 # ↑ +1

Статья поначалу вызывает, восторг, да. Пока думать не начинаешь:) Почитайте комменты оппонентов.

 **seriyPS** 21 июля 2009 в 15:41 # +1

Кстати, в макдоналдсе тоже что-то похожее видел: кассиры такие пластмасски с циферками на кухню запускают и им по ним уже делают и возвращают всякие гамбургеры-чизбургеры

 **vzzvzz** 23 июля 2009 в 21:33 # ↑ 0

Я вам больше скажу, Таичи Оно придумал канбан для тойоты, когда посетил американский супермаркет.

 **belonesox** 21 июля 2009 в 15:49 # +3

Полмесяца назад была встреча «SCRUM против Kanban» сообщества AgileRussia.
Там все кости и первому и второму перетерли.

Есть видео в достойном качестве.
team.custis.ru/2009/07/kanban-vs-scrum.html

 **Arion** 21 июля 2009 в 15:57 # +1

Очень напрягали большие потери времени на планирование в SCRUM. Тут прям сказка, нужно попробовать как на практике

 **mx2000** 22 июля 2009 в 13:44 # ↑ 0

только не забывайте про закон Паркинсона ;-)

 **tenshi** 21 июля 2009 в 16:38 # 0

что делать с мелкими фичами с точки зрения маркетинга?

 **bishop3000** 21 июля 2009 в 16:40 # ↑ 0

С какими, например?
Общий случай — включать их в более крупные фичи.

 **tenshi** 21 июля 2009 в 16:58 # ↑ 0

например, рефакторинг. если включить его вкрупную маркетинговую фичу, то у тестеров добавится мороки при её тестировании, а неизбежно возникающие при рефакторинге ошибки будут тормозить релиз той фичи…

 **bishop3000** 21 июля 2009 в 17:14 # ↑ 0

Про рефакторинг — это целый отдельный разговор. Кто-то делает его постоянно про разработке любой фичи, считая, что тронул код — прорефактрь его.
Кто-то выделяет специально время на него — это уже может быть вне канбана. Просто раз в полгода на 2 недели все делают рефакторинг.
А можно и фичу такую завести и назвать ее как-нибудь типа «Улучшение надежности, стабильности и поддерживаемости кода».

 **Leningradez** 18 апреля 2013 в 10:54 # ↑ 0

Вопрос по bug-fixing:
— какое место во всей этой цепочке (по вашему мнению), занимают Bugs?
— Какой процесс доработки «некачественного» функционала?

 **waldfalke** 21 июля 2009 в 17:02 # +1

Спасибо за ссылки. Почитаю.

 **antirek** 21 июля 2009 в 17:52 # +1

1. Визуализируйте производство
— Разделите работу на задачи, каждую задачу напишите на карточке и поместите на стену или доску.
— Используйте названные столбцы, чтобы показать положение задачи в производстве.
2. Ограничивайте WIP (work in progress или работу, выполняемую одновременно) на каждом этапе производства.
3. Измеряйте время цикла (среднее время на выполнение одной задачи) и оптимизируйте постоянно процесс, чтобы уменьшить это время.

Да это в любой деятельности работает: в логистистике, в производстве, в делопроизводстве и так далее…

 **dummy** 22 июля 2009 в 02:28 # +3

не вступая в дискуссию по процессам (хотя я думаю, что канбан — это больше философия)
поделюсь следующей информацией.
Широко известный в узких кругах Nate Kohari — автор ninject.org, mvp, ex-Telligent, вместе со своей женой phd по industrial/organizational psychology запустил новый стартап</>. Речь идет об инструменте ведения проектов по Lean Kanban. На сайте можно попробовать канбан инструменты, хотя бесплатно там только 1 проект и 1 пользователь. Также вот здесь есть хороший подкаст Nate Kohari with Steve Hanselman на тему практического применения канбан и того инструмента, который сделал Nate.

 **ewro** 22 июля 2009 в 12:38 # –1

>>В-третьих, можно вычислить время на выполнение усредненной задачи.
Да уж, средняя температура по больнице.

>>А из этих цифр менеджер или product owner может уже рассчитывать всё, что ему угодно.
С этими цифрами ничего нельзя сделать.

 **Arekus** 22 июля 2009 в 14:41 # +1

Использую (веб-дев) близкую к канбану технику. Отличия:
1) фичи сгруппированы по сущностям системы
2) У каждой фичи есть предварительная оценка. Если оценка сильно различается с реальным временем — проводится анализ.
3) Написание теста перед разработкой. Собственно окончание разработки — успешное прохождение теста. Нормально, если перед разработкой фича может быть возвращена в область тестирования (добавить-убрать)
4) вместо доски — сайт с тикетами, за счет автоматизации идет контроль кто, сколько, чего делает, ну и еще несколько приятных фич для разработчиков.
5) Ежедневный и недельные циклы деплойментов.

В разработке лежит проект для веба, может когда-нить и доделаю.

 **vodka_ru** 16 августа 2009 в 17:59 # 0

1. >«Если команда не хочет работать, то как ее не контролируй, она провалит проект.»
Метлой к чертям и набираем команду, которая получает фан) В действительности, по-моему, команда для менеджера — это тыл, если он слаб или в тылу предатели — нужно срочно поправлять дело.

2. И еще я думаю, нужно принять во внимание внутреннюю дисциплину японцев, капитализм запада, расхлябанность и лень русского народа с исторически сложившимися восприятием барин-холоп в вперемешку с моделью социализма. К примеру, модель Йордона по мотивации, удержанию и повышению результативности сотрудников, которой он делился на семинаре с русскими IT управленцами в 2008, во многом сомнительна для русских IT компаниях.

 **Saigo** 26 марта 2010 в 22:51 # 0

> Если менеджер верит команде, то зачем иметь оценку времени?

Можно верить команде, то есть быть уверенным, что разработчики и тестеры будут работать на максимуме способностей и выполнять задачи очень-очень быстро, но как это избавляет от необходимости знать хотя бы приблизительные сроки завершения работы по конкретной задаче? Это ведь не для формального контроля нужно, а для выполнения менеджером своих прямых обязанностей: общения с заказчиками, которых мало волнуют особенности внедрённых технологических процессов; или подготовки к публичному, маркетинговому запуску новой фичи.

 **jurikolo** 5 апреля 2011 в 15:16 # 0

Отличная статья, благодарю!
Наконец-то получил «волшебный пинок» для того, чтобы повесить доску в доступное место и визуализировать процесс.
Осталось подобрать приложение для того, чтобы удалённые разработчики видели похожую доску, а именно — список задач.
Пока используется agilo, но есть мысли уйти от него.

 **evra** 12 апреля 2013 в 17:14 # 0

простейший канбан

trello.com

канбан с настройками

leankit.com

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.