senpay карма 4,0 10 голосов рейтинг 0,4

22 июля 2014 в 18:13

Методология Kanban: введение

из песочницы

Agile*

Добрый день!

Одним из моих профессиональных интересов, как координатора команды тестировщиков, являются методологии разработки программного обеспечения. В настоящее время все большую популярность приобретают так называемые Agile-методологии, в особенности Scrum и Kanban. На «распиаренных» терминах играют недобросовестные «тренеры», семинары и сертификации («сертифицированный Scrum-мастер», «сертифицированный Product owner» и т.д.) растут как на дрожжах.

В большинстве недобросовестных статей и тренингах любая методология представляется как магическая серебряная пуля, которая мигом решит проблемы коммуникации, враз спасет от некомпетентности отдельных членов команды. В общем, поможет именно вам решить именно ваши проблемы. В текущем году я поступаю в магистратуру Белорусский Государственный Университет по специальности «технологии управления персоналом» и планирую рассмотреть подробно плюсы и минусы, а также ограничения применимости наиболее распространенных методологий разработки программного обеспечения.

В процессе работы я часто сталкивался с непониманием и неверным трактовкам инструментов методологий, применения модной методологии без учета контекста. После прочтения статьи я понял что проблема скорее глобальная, чем локальная. Предлагаю сегодня немного рассмотреть Kanban, его историю, основные принципы, и возможные границы применения.

История термина

Kanban – японский термин, который начали использовать применительно к производству в 60-х годах 20-го века в компании Toyota. В основу данного принципа положен конвейерный метод производства, а также различные скорости выполнения отдельных технологических операций на производстве. Попробую объяснить на пальцах. При любом производстве есть основное производство («главный конвейер») и дополнительное производство («дополнительные конвейеры»). Темп выпуска конечных изделий задает главный конвейер, в то время как дополнительные конвейеры не могут ускорить темп выпуска изделия, но могут замедлить его, в случае несвоевременного выпуска требуемых деталей.

Дополнительно, при производстве может произойти смена приоритетов. К примеру выяснилось что станция, которая производила левые зеркала произвела 20 шт., а станция производившая правые зеркала — 10 шт., в то время как на конвейере находятся 15 автомобилей и необходимо 15 штук зеркал обоих типов. Налицо конфликт метрики — количественно производство не упало (дополнительные конвейеры выпустили 30 изделий в срок), но производство все равно рискует остановится. Kanban призван помочь с этой проблемой.

В упрощенном варианте, Kanban включает в себя два простых правила:

- производственная станция имеет план производства деталей («backlog»). План отсортирован по приоритету, и может меняться в любое время (к примеру станция производящая слишком много левых зеркал должна иметь возможность переключиться на правые как можно скорее);
- количество задач, выполняемых на станции одновременно ограничено (т.е. производить не более заданного количества зеркал одновременно). Это ограничение необходимо для управления скоростью производства на станции, а также скоростью реагирования на изменения плана.

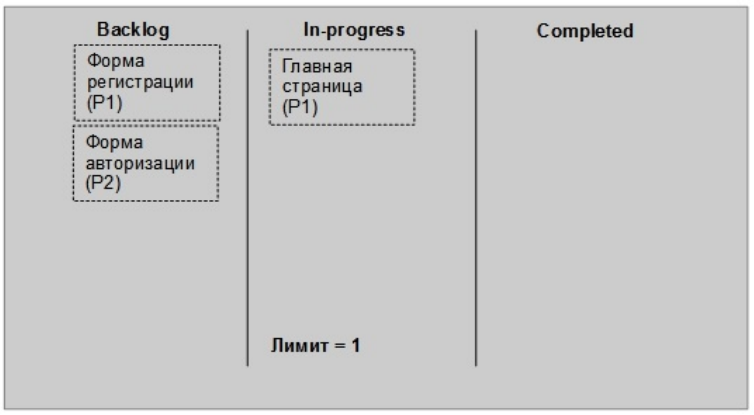
Настоящее время

В последнее время, Kanban набирает большую популярность в производстве программного обеспечения. Некоторые команды считают эту методологию исключительно полезной, некоторые используют по принципу «культа Карго». Основываясь на моем эмпирическом опыте чистый Kanban плохо работает для продуктовых команд (читай — «основной конвейер»), но отлично работает с командами поддержки, такими как:

- группы поддержки программного обеспечения, где не важен «план», но важна скорость реагирования на изменения;
- группы тестирования, работающие отдельно от групп разработки;

- службы поддержки;
- другие примеры «неосновных производств».

Отдельно необходимо отметить, что Kanban хорошо работает в стартапах, не имеющих четкого плана, но активно работающих над разработкой. Предлагаю рассмотреть пример использования Kanban в разработке программного обеспечения. Заранее прошу простить за некрасивые иллюстрации. Давайте представим себе команду из одного разработчика, работающего над небольшим проектом. План разработки (backlog) отсортирован в порядке приоритета кусков работы, лимит команды на задачи в процессе — 1 шт.



Для управления процессом руководитель проекта может:

1. изменить лимит на количество задач в работе;
2. добавить задачу с более высоким приоритетом (к примеру p0) для того чтоб она была взята как можно скорее;

В процессе работы может так произойти, что работа заблокирована (сломался хостинг, не скачан нужный framework и т.д.). В общем случае, заблокированная работа возвращается в backlog, и выбирается новая задача, с максимальным приоритетом. В зависимости от характера задач и типа команды лимит может быть увеличен или уменьшен. К примеру, наш разработчик может одновременно рисовать форму регистрации и смотреть за процессом развертывания нового сервера. Тем не менее, если время завершения задач будет меньше требуемого, руководитель проекта может уменьшить лимит, или увеличить команду. Таким образом, при грамотном руководстве, Kanban обеспечивает максимально возможную для данной команды скорость работы, максимальную скорость реагирования на изменения и в то же время сократить «расходы» на поддержку методологии. В общем все! Kanban — это не просто, просто. Это очень просто!

К ограничениям Kanban'а при использовании его в продуктовых командах можно отнести:

- данная методология плохо работает с большими командами (больше 5 человек);
- в чистом виде, Kanban плохо работает с кросс-функциональными командами. Т.е. в отличие от Scrum, тяжело совместить тестирование и разработку в одной команде. Более удачной мыслью является разбить процесс на «станцию» разработки и «станцию» тестирования с отдельными руководителями и backlog-ами;
- ввиду своей истории и специфики, Kanban не предназначен для долгосрочного планирования.

Заключение

В заключение, хочу добавить, что сравнение любых методологий по принципу «кто круче» не продуктивно и контр-конструктивно (капитан очевидность). Каждая, более-менее распространенная, методология имеет свои плюсы, минусы и границы применения. Дополнительно, Agile-методологии в принципе накладывают большие требования на сработанность и опыт членов команды.

В случае возникновения интереса к теме продолжу рассмотрение Kanban'а подробнее. В последствии, предлагаю разобрать по полочкам и картинкам Scrum и RUP.

Более подробно, и наглядно можно посмотреть в:

- статье, посвященная Kanban на Wikipedia.org;
- отличной книге «SCRUM и KANBAN: выжимаем максимум», Хенрик Книберг и Маттиас Скарин.

Kanban, Scrum vs Kanban

Комментарии (50)

linjan 22 июля 2014 в 18:18 (комментарий был изменён) # +3

Заинтересованным советую попробовать канбан в деле.

SLY_G 22 июля 2014 в 21:13 # ↑ +1

Пробую.
Как я понял, если расценивать карты как отдельные задачи, то можно назначать их определённым исполнителям. И можно смотреть, какие карты/задачи есть у исполнителя. Но при этом довольно трудно отслеживать скорость его работы, т.е. количество задач, сделанных за неделю, или тем более назначать задачам сложность или ресурсоёмкость и измерять эффективность работника.
Для таких целей какие инструменты используются?

huze 23 июля 2014 в 13:55 # ↑ 0

Или pintask.me (статья).

realno 22 июля 2014 в 18:36 # +2

тему как будто палочкой потыкали :) пишите смелее

senpay 22 июля 2014 в 21:40 # ↑ 0

Хорошо, соберусь с силами и напишу длинное продолжение :)

misato 22 июля 2014 в 20:40 # +1

«разпиаренных» => распиаренных
«серебренная» => серебряная

Скажу от себя, что Канбан плохо подходит для клиентской поддержки. Казалось бы, он даёт инструмент для быстрого реагирования, но зато полностью отбирает возможность получить какую-либо определённость в сроке исполнения задачи, потому что любая работа может быть сдвинута на неопределённый срок другими задачами.

Да и в целом, я бы не стал называть Канбан методологией. Это скорее удобная практика, которую можно применять в подходящих случаях.

dimaniko 22 июля 2014 в 21:09 # ↑ 0

Как это отбирает определенность? Вы же замеряете lead time для классов задач. По каждому классу прогноз и будет очень точный. При достаточной статистике, конечно. Не через неделю, после того, как доску повесили.

misato 22 июля 2014 в 21:13 # ↑ 0

Если я правильно понимаю вашу терминологию, lead time — это статистика, показывающая, в среднем, как долго задача добирается до статуса (in-progress? completed?). В любом случае, статистика — это вероятностная величина. Её нельзя пообещать клиенту, которому, например, надо запустить лэндинг к четвергу, потому что у него стартует реклама по телевизору, которая стоит дороже чем вся ваша команда с её канбаном.

dimaniko 22 июля 2014 в 21:36 # ↑ 0

Lead time — время, за которое задача проходит через систему, от приема в работу, до завершения. Не знаю, откуда вы знаете, сколько стоит моя команда, но если вы просто поглядываете, сколько там у вас в среднем на задачу уходит и ничего не предпринимаете для повышения предсказуемости процесса, то это не канбан, а… хроника пикирующего бомбардировщика.

А если вы беклог формируете из «лендингов к четвергу», то вы сам себе злобный буратино и никакой канбан со скрамом вам не помогут.

misato 22 июля 2014 в 23:16 # ↑ 0

Когда я говорил «ваша команда», то имел в виду не вас, а некоторую абстрактную команду, использующую канбан, равно как и пример заказчика с телевизором тоже выдуман из головы, не принимайте на свой счёт.

Лично я, пока участвовал в организации работы отдела разработки, предпринял для повышения предсказуемости следующее — предложил избавиться от канбана и внедрить старое-доброе планирование работы по дням недели, что впоследствии очень хорошо сказалось как на предсказуемости результатов, так и на производительности труда, и на прозрачности всего процесса в целом.

dimaniko 23 июля 2014 в 12:56 # ↑ 0

Вы спрашиваете, что такое lead time, сомневаетесь в WIP — извините, но кажется у вас не было канбана.

misato 23 июля 2014 в 13:14 # ↑ 0

Вы почему-то не настроены на продуктивную беседу. Возможно я не владею нативной терминологией этой практики, но это не мешает рассуждать в общем.

В компании, куда я пришёл заниматься организацией деятельности, использовалось нечто, что они называли канбаном. У них были карточки с ограничением количества в работе и прочие вещи. А компании нужна была прозрачность и понимание того, в какой день какая задача будет готова. Всю эту систему мы заменили на планирование по дням, потому что, как мне кажется, система подобная канбану такого дать не может, у неё другие преимущества.

Вы мне пытаетесь доказать, что в канбане можно гарантировать исполнение какой-то задачи в конкретный день. Я и спрашиваю вас, как? Статистические значения здесь не годятся, потому что сроки бывают жёсткими.

dimaniko 23 июля 2014 в 13:44 # ↑ 0

Вы пришли на новое место, где система была «установлена», но, я так понял, задачи не решала. Вы систему заменили на свою. И это понятно — вам нужны были результаты, но добиться их установленной не вами системой вы не могли. Если меня попросят свалить дерево и дадут бензопилу и топор — я начну рубить топором, потому что бензопилу не разу в жизни не заводил и не знаю сколько времени на запуск уйдет, и запустится ли она вообще.

Но это не дает вам права заявлять, что канбан не подходил для вашей компании. Он не подошел лично вам. Никто не показал вам, как пользоваться бензопилой, а на чтение инструкций времени, скорее всего, не было. Но бензопила не стала от этого хуже — даже начинающий пильщик обгонит ваш топор. И, продолжая аналогию, чем больше дерево, тем явнее преимущество.

dimaniko 23 июля 2014 в 14:03 # ↑ 0

Про задачи с дедлайнами. Как вы сейчас определяете, будет ли задача сдана в срок?

misato 23 июля 2014 в 14:41 # ↑ 0

Сейчас в этой компании задачи от клиента формализованы менеджером или разработчиком и оценены разработчиком в часах. Планирование по дням позволяет запланировать работу по задаче на определённый день. Оставшийся риск в этой схеме — это риск неправильной оценки задачи разработчиком. (Практики снижения этого риска уже выходят за рамки обсуждения, конечно они тоже применяются.) Таким образом, мы проблему решили: существует план, по плану задача в четверг, значит она будет в четверг, и любые движения без предварительных согласований невозможны.

В канбане это гораздо сложнее устроить. Если тебе нужно, чтобы задача была во что бы то ни стало сделана к конкретному дню, тебе нужно постоянно играть с приоритетами, повышая их, по мере приближения дедлайна. И это всё равно не гарантирует выполнения в срок (потому что всё, что уже висит на доске, может затянуться).

Плюс ко всему, когда в компании несколько инициаторов задач и несколько бэклогов, а отдел разработки один, тогда ещё появится и проблема дележа приоритетов (или ресурсов).

dimaniko 23 июля 2014 в 15:08 # ↑ 0

> В канбане это гораздо сложнее устроить. Если тебе нужно, чтобы задача была во что бы то ни стало сделана к конкретному дню, тебе нужно постоянно играть с приоритетами, повышая их, по мере приближения дедлайна.

Это совсем не так. ;] Канбан вообще ничего почти не предписывает, он накладывается на ваш существующий процесс. Это просто инструмент для анализа и изменения существующего процесса. Следуй минимуму правил, а дальше используй «плагины» к канбану по необходимости. Зачем «играть» какими-то «приоритетами»? Если у вас есть задачи со сроком поставки — крупно дедлайн на каждой надпишите да выделите их в отдельный класс, поставьте лимит и применяйте к ним свои правила (policy). Ну, может, покрасьте их карточки в какой-нибудь оранжевый для наглядности.

> И это всё равно не гарантирует выполнения в срок (потому что всё, что уже висит на доске, может затянуться).

А если задачу не повесить на доску — она не затянется? Почему?

> Плюс ко всему, когда в компании несколько инициаторов задач и несколько бэклогов, а отдел разработки один, тогда ещё появится и проблема дележа приоритетов (или ресурсов).

А сейчас почему проблемы нет?

misato 23 июля 2014 в 19:24 # ↑ 0

Задачи со сроками могут быть не крупными проектами, а небольшими работами на три-четыре часа. Выделять под это собственный канбан и резервировать людей бессмысленно. Можно написать крупно дедлайн, но кому будет до него дело, если люди смотрят на те задачи, которые висят «в процессе».

По поводу «висят на доске» — видимо мы друг друга не поняли. Я имел в виду, что даже если вы ставите задачу в бэклог с высоким приоритетом, до неё может не дойти дело, поскольку исполнители уже заняты задачами «в процессе», которые там двигаются в тестирование и обратно. Вот эта неопределённость и мешает.

Почему планирование по дням помогает решить проблемы с дележом ресурсов? Потому что при таком подходе проще отследить, кто именно загрузил отдел работой в прошлом, в настоящее время, и в будущем. Это наглядно видно по календарю, а значит с этим можно работать. Менеджеры могут буквально до часов делить ресурсы отдела на следующую неделю и это легко контролируется.

dimaniko 23 июля 2014 в 20:02 # ↑ 0

Мы действительно во многом друг друга не понимаем. Как вам удается с точностью до часов распланировать работу команды, но при этом испытывать «неопределенность» по поводу сроков завершения работы «прогрессе»?

Если вы заранее способны предсказать человеко-часы на задачу, что вам мешает посчитать когда закончится задача в прогрессе и команда возьмет сверху стопочки задачу с дедлайном?

Если вы не знаете, сколько времени уйдет на каждую задачу (что ближе к истине), то на чем основывается почасовой план распределения ресурсов? И выдерживается ли он вообще?

misato 23 июля 2014 в 22:51 # ↑ 0

Посчитать все задачи в процессе можно, но когда задач много, это неудобно. Всякий раз пересчитывать, прикидывать, успеют или нет… Это непрозрачно.

План по часам в целом выдерживается. Не вижу ничего сверхъестественного в том, что человек способен оценить время, необходимое на работу, тем более что сложные блоки раскладываются на более простые, которые уже и оцениваются. Исследовательские задачи в которых вообще ничего непонятно можно искусственно ограничивать оценкой (т.е., время в таком случае важнее результата). Это даёт управляемость процесса, нет такого, что все люди вдруг заняты, и некому сделать что-то срочное.

В то же время эта система защищает разработчиков от перегруза, когда всем всё надо сделать срочно. Есть план — в нём и разбирайтесь, кому из вас нужнее. Это хорошо влияет на качество.

Естественно, в этой схеме предусмотрены действия, что делать если человек ошибся с оценкой и как максимально безболезненно осуществлять перепланировку.

В целом я не пытаюсь доказать, что такая система однозначно лучше канбана. Она просто лучше в определённых условиях, когда есть много проектов, много задач, требуется быстрое реагирование и соблюдение каких-то локальных сроков.

dbondarev 23 июля 2014 в 14:31 # ↑ +1

А как вы вообще можете гарантировать испольнение какой-то задачи в конкрентый день? Приоритезация и дисциплана — без этого при любом подходе сроки будут продалбываться

misato 23 июля 2014 в 19:26 # ↑ 0

Дисциплина — это исполнение правил. Если правила разрешают возить задачу из тестирования в разработку и обратно, в то время, пока в бэклоге лежит срочная задача — то никакая дисциплина не поможет.

dbondarev 23 июля 2014 в 19:44 # ↑ +1

а еще есть приоретизация и здравый смысл.

в бизнесе всегда все срочно и важно. если бездумно хватать задачи из беклога — легко оказаться в ситуации когда все почти готово, но из-за «почти» — ничего еще не работает.
в этом плане канбан очень хорошо показывает что «окей, мы сейчас вот эту срочную задачу возьмем, но тогда этунадо одну из этих вернуть в начало». и в этом момент может оказаться, что для бизнеса закончить ту задачу, которую возят из разработки в тестирование — важнее, чем взять в разработку еще одну новую.

misato 23 июля 2014 в 22:56 # ↑ 0

К сожалению, здравый смысл — это внесистемное решение.

dimaniko 23 июля 2014 в 20:10 # ↑ 0

Так поменяйте правила-то. Договоритесь, например, что задача с дедлайном требует переключить на себя внимание. Вдобавок WIP составной сделайте — 3 обычных и 1 со сроком.

senpay 22 июля 2014 в 21:46 # ↑ 0

Ошибки поправил.

По поводу того, что любая задача может быть сдвинута — в классическом Kanban такого происходить не должно. Возвращаясь к аналогии с производством автомобилей — производственная станция не может прекратить производство зеркала без существенных потерь времени, поэтому из «in progress» путь в backlog должен быть только один — задача заблокирована и не может быть выполнена.

Правомерность использования термина «методология» применительно к Kanban я планирую рассмотреть в продолжении статьи.

funca 22 июля 2014 в 21:41 # +3

Везде пишут, что канбан был придуман для управления manufacturing, в смысле серийным производством, где про выработку каждой отдельной штуковины известно все — время, цена, и качество. Управляемой величиной является объем производства (или заказов наружу) штуковин, необходимых для удовлетворения нужд конвейера.

В процессах разработки (development), примеры из которого приводятся в статье, время, качество и цена штуковин являются сильно переменными величинами. Т.е. ставя задачу Васе сделать формы авторизации и регистрации, вы переживаете совсем не о том, что он сделает 20 форм регистрации, а форм авторизации всего 10, в то время как на конвейере у вас 15 сайтов. Скорее всего это один сайт, одна форма регистрации и одна форма авторизации, а то чем вам приходится управлять — это неопределенность, что же на самом деле сделает Вася, когда, сколько вам это будет стоить. При попытке притянуть сюда канбан, какая-то методология тоже получается, но это уже (имхо) совсем не похоже на канбан. Может просто пример неудачный или я совсем запутался.

senpay 22 июля 2014 в 21:51 (комментарий был изменён) # ↑ 0

Согласен, пример натянутый. В реальной жизни в данной ситуации возможно более подойдет другая методология. В продолжении я планирую расписать Kanban применяемый на Toyota, «книжный» вариант для разработки ПО, и примеры из реальной жизни, где Kanban пришелся к месту. (В основном — это команды автоматизации тестирования, «обслуживающие» несколько отдельных групп тестирования)

misato 22 июля 2014 в 23:20 # ↑ 0

Всё верно.

В командах, разрабатывающих ПО, практика канбана чаще сводится к тому, что это, дескать, очень наглядно: в любой момент видно, как много задач мы сделали, и как много ещё предстоит сделать. Хотя очевидно, что если команда использует хоть что-то похожее на учёт задач, то всё то же самое достигается одним простейшим отчётом.

dbondarev 23 июля 2014 в 07:03 # ↑ 0

это просто от того, что команды плохо понимают, зачем им это надо.

канбан — это жесткое ограничение числа одновременно выполняемых задач. если на это правило забивать — он не работает :)
простой отчет этого делать не заставит, только самодисциплина.

S_A 23 июля 2014 в 11:20 # +3

О как раскритиковали. Давайте теперь посмотрим на канбан с другой стороны. Вообще-то канбан — это бирка-карточка, которая крепится к детали. Движение карточек по сути отражает потребности производства.

Основная идея канбана (производственного) в том, что канбан возвращается по мере необходимости, чем достигается не только избавление от затоваривания ненужным, но и скорость отоваривания нужным. Следует отметить, что поставщики вынуждены затовариваться чтобы соответствовать Тойоте, в то время как она сама работает практически без муды на гембе (без лишнего на площадке). То есть система заказ-поставка перевернута в поставку-заказ.

Почему ИТ-канбан всё-таки канбан (методологией вообще громко что-то называть, неважно канбан это, скрум или даже RUP — все это сборники практик). Представьте себе стандартный усредненный процесс разработки

```
Сбор требований -> Анализ и проектирование -> Разработка -> Тестирование -> Развертывание
```

Это не конвейер, но канбан — карточка — и не требует конвейера, канбан это средство коммуникации в производстве. В чем проблема приведенного процесса? В том что ошибки накапливаются как снежный ком, при переходе с этапа на этап. Самый простой способ это устранить — это возвращать *канбан* в случае чего его отправителю, то есть сделать его таким

```
Сбор требований <--> Анализ и проектирование <--> Разработка <--> Тестирование <--> Развертывание
```

Вот вам и канбан. Благодаря обратным стрелкам заказ-поставка превращается… в поставку-заказ.

Почему ограничивают число задач в работе? Для того, чтобы они всё-таки проходили до конца. Если не ограничивать число задач, то тогда lead time будет непредсказуем — просто потому что задачи могут гулять бесконечно между статусами, всё новые и новые. Конечно, поскольку число задач конечно, процесс выйдет из тупика :) но за какое время — как раз-таки и не предскажешь. А при наличии лимитов lead time более-менее надежен, а его значения — путь к оптимизации (третья «заповедь» ИТ-шного канбана).

> - данная методология плохо работает с большими командами (больше 5 человек);
> - в чистом виде, Kanban плохо работает с кросс-функциональными командами. Т.е. в отличие от Scrum, тяжело совместить тестирование и разработку в одной команде. Более удачной мыслью является разбить процесс на «станцию» разработки и «станцию» тестирования с отдельными руководителями и backlog-ами;
> - ввиду своей истории и специфики, Kanban не предназначен для долгосрочного планирования

Это почему плохо она работает с командами больше 5 человек? Работаем вдесятером по канбану. Чтобы визуализация не захламлялась, как раз лимиты годятся. Это почему плохо она работает с кросс-функциональными командами? Как раз наоборот. Трекается как фича, так и её переходы. Канбан не годится для долгосрочного планирования точно так же как и любой другой agile-процесс, потому что это **способ организации процесса**. Однако прогнозы на его основе возможны (как и в скраме в том же).

Вобщем не согласен с тем, что канбан — это вспомогающий процесс для очень мелких команд. Канбан — это процесс, который позволяет иметь высокое качество при самых быстрых сроках (в том же скраме например исправления могут уйти только на следующую итерацию, а в канбане — мгновенно). И еще — канбан всем понятен. Ладно, больше не буду уже писать, и так много уже сказал.

misato 23 июля 2014 в 11:50 # ↑ 0

То есть, вы говорите, что главный профит здесь заключается в том, что суммарное количество карточек на всех этапах процесса ограничивается, таким образом, вы не можете окончательно завалить отдельный этап работой, из-за которой всё станет совсем непредсказуемо. Правильно?

То есть, если, например, в приведённой схеме «Сбор требований <--> Анализ и проектирование <--> Разработка <--> Тестирование <--> Развертывание» этап тестирования перегружен, то аналитик не может взять себе новых задач и будет сидеть без дела?

S_A 23 июля 2014 в 11:59 # ↑ 0

Где я такое сказал. Профит в том, что карточки ходят туда-сюда, это и наглядно, и контролируется лимитами. Ограничивается вообще-то каждый этап, а не все вместе.

misato 23 июля 2014 в 13:17 # ↑ 0

Ну фактически так и будет. Пробка накопится, по цепочке: мы завалим тестирование, потом остановится разработка, потому что она не сможет ничего передать вперёд, потом аналитика, потому что ей тоже некуда передавать задачи вперёд и всё, все сидят и ждут.

S_A 24 июля 2014 в 05:41 # ↑ +1

Тут ниже уже фактически ответили, я добавлю. Такое может случиться в любом процессе. И при обычном последовательном планировании разработчики тоже могут курить, пока идет тестирование.

Канбан точно так же как и обычный проект нужно организовывать. Если каждая задача будет дефектна, то тут ни канбан, ни что-либо еще не поможет. Если для 10 разработчиков взять 1 тестировщика, то тут ни канбан, ни что-либо еще не поможет.

Чтобы такого не происходило, следует подбирать лимиты не только исходя из скоростей, но и исходя из уровня качества на выходе каждого этапа. Если такое произошло, следует не только поменять лимиты, но и разобраться с технологией — почему столько дефектов, ведь никто управление в случае канбана не отменяет (и уже на основе результатов разборок менять лимиты).

Сами скорости для установки лимитов даже не очень обязательно знать. Важно знать относительные скорости. Например, аналитик разбирается с фичей в два раза быстрее, чем она делается, точно так же и тестирование фичи занимает половину времени разработки. Тогда исходя из количества имеющихся человек можно выставлять лимиты. В случае возврата карт, они будут делаться первыми, так как у них приоритет был изначально выше.

dimaniko 23 июля 2014 в 12:34 # ↑ 0

Именно что без дела. Чтобы вам видно было, где у вас боттлнек, вокруг которого оптимизировать надо.

misato 23 июля 2014 в 13:18 # ↑ 0

Это хорошая система на производстве, но в проектах ПО может так оказаться, что в один момент бутылочное горлышко будет на тестировании (делаем что-то, сложное в тестировании), а через месяц уже в разработке.

senpay 23 июля 2014 в 13:42 (комментарий был изменён) # ↑ 0

Поэтому удобнее разделить группы по специализации. Получается своеобразный аналог производственной линии с гибкой связью. К примеру, есть группа разработки, которая производит фичи со скоростью 6 фич в неделю. Эти фичи переходят в группу тестирования, которая тестирует со скоростью 4 фичи в неделю. На выходе «главный конвейер» (к примеру билд-инженеры) ожидают 5 фич в неделю. «Затык» становится сразу очевиден (по каким-то причинам тестирование идет медленнее чем ожидается, а разработка быстрее) и эту проблему можно решить достаточно оперативно.

Я встречал «жесткий» Канбан, в котором состояние «в разработке» разбито на состояния «в разработке», «в

тестировании», однако такой подход вносит следующие трудности:
— как задавать «лимит»? для каждого состояния в отдельности, или для всех одинаковый? В любом случае, возможна ситуация что разрабатывать будут быстрее чем тестировать (или тестировать быстрее чем деплоить, и т.д.), что в итоге лишь усложнит процесс управления.
— в идеале, все задачи должны быть разбиты на «равные куски» работы, с каким-то определенным временем на выполнение. И для каждого этапа работы обычно эти куски различные. С возрастанием количества состояний падает прозрачность процесса.
Возможно это субъективные критерии, но мой опыт показал что разделение команд на разные Канбан-доски принесло позитивные изменения.

Любая методология должна прежде всего упрощать процесс разработки, а не поддерживать сама себя. Если какая-то методология не помогает — значит что-то не так :)

S_A 24 июля 2014 в 05:51 # ↑ 0

Заведение лишней доски — это создание дополнительного проекта. Это означает, что чтобы фича прошла из бэклога в деплой, необходимо знать два лид тайма, и управлять двумя наборами лимитов. И это не решает вобщем-то вопрос с пробками, пробки могут образовываться и в этом случае так же. Большей управляемости переходом разработка-тестирование (и наоборот) можно достичь и дополнительными статусами. Прозрачность конечно падает при искусственных состояниях, но мы тут рассматриваем какой-то совсем плохой случай — когда команда не справляется. Чтобы команда справлялась, необходим как раз менеджер, который выстроит процесс.

senpay 24 июля 2014 в 17:13 # ↑ 0

Да, фактически это и есть создание дополнительного проекта. Возможно мое мнение предвзято — я привык работать на больших проектах, которые было бы удобнее «разбить» на несколько поменьше.

По поводу применимости Канбана к большим проектам, возможно все упирается в психологию малых групп — формальная группа численностью более 8 человек стремится к разбиению на две неформальных. Ну или принцип Паркинсона — кабинет численностью более 5 человек теряет эффективность :)

Конечно, это не значит что большой проект нежизнеспособен. Просто с моей точки зрения, управлять двумя небольшими прозрачнее и удобней чем одним большим.

dimaniko 23 июля 2014 в 14:12 # ↑ 0

Ну так это и есть сигнал вам — ваша система несбалансирована для такой сильной вариации задач. Либо вы смиряетесь с этим, увеличив WIP. Либо, если вам важно повысить производительность (а это не всегда самое важное), принимаете меры. В вашем случае — тестировать силами девелопмента при необходимости, например. Automation, все дела.

senpay 23 июля 2014 в 14:45 # ↑ 0

Если система не сбалансирована для такой вариации задач, почему не выбрать вариант «переработать систему»? Разделение на специализированные команды — один из вариантов, и он работает. Automation не решает проблем тестирования, и в случае если тест повторяется не более 3 раз (примерная эмпирическая цифра), то усилия на автоматизацию могут не окупиться.

Дополнительный пример из жизни это специализированные задачи де-факто, которые не требуют тестирования или девелопмента. Пример для задач которые не требуют девелопмента — интеграционное тестирование с новой версией сервиса, от которого зависит наше приложение. Такая задача «минует» фазу разработки и не будет отвлекать людей от их дел :)

Конечно, этот вариант не единственный. Возможно, специфика проектов, на которых я работал не типична для большинства других проектов. Но пока единственной слабой стороной такого решения мне видится необходимость заведения лишней доски :)

dbondarev 23 июля 2014 в 14:50 # ↑ +1

Тут зависит от того, как поставить процесс «сидения без дела».
Если одно звено плюет в потолок, пока остальные пашут — это одно.
Но ведь всегда есть разные вещи, которые откладываются потому что на них времени нет.
У тестеров завал? Значит разработчики пока могут технический долг отдать. Или юнит-тесты написать.
Главное принять тот факт, что 100% загрузка всех — это не всегда хорошо

dimaniko 23 июля 2014 в 15:10 # ↑ 0

Воистину.

misato 23 июля 2014 в 19:28 # ↑ 0

Это хорошие, правильные рассуждения, но многое зависит от бизнес-модели. Где-то такая вот работа по техническому долгу, рефакторинг, увеличивает издержки по проекту, которые не будут покрыты заказчиком, и это очень печалит менеджеров, которые получают премию за рентабельность :)

dbondarev 23 июля 2014 в 19:58 # ↑ +1

на эту тему буквально на днях Максим Дорофеев свой доклад выкладывал.
там очень простой и наглядный пример, где желание менеджера загрузить всю команду на 100% прибыль отрицательной. Поэтому рентабельность надо еще правильно просчитать :)

dimaniko 23 июля 2014 в 20:13 # ↑ 0

Это он немного про другое — про утилизацию. А техдолг и рефакторинг вполне можно с некоторым усилием в деньги перевести и позакомить с начальством.

misato 23 июля 2014 в 23:00 # ↑ 0

Фактически, когда вы предлагаете занять разработчиков рефакторингом или юнит-тестами — это тоже просто способы занять их на 100%. Вопрос только в том, кто за это заплатит в конечном счёте :)

podluzny 23 июля 2014 в 13:26 # 0

Хорошая онлайн доска http://leankit.com/

clockworkbird 28 июля 2014 в 10:39 # 0

Ограничиваться только Канбаном в управлении командой веб-разработчиков, довольно сложно, и даже рисковано. Хотя, конечно, все зависит от команды.
На мой взгляд он всего лишь один из инструментов, довольно высокоуровневый, предназначенный, в первую очередь, для управления самим процессом «производства» и хорошо подходит для команд, построивших конвеер разработки (например, разработка сайтов на потоке).

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.