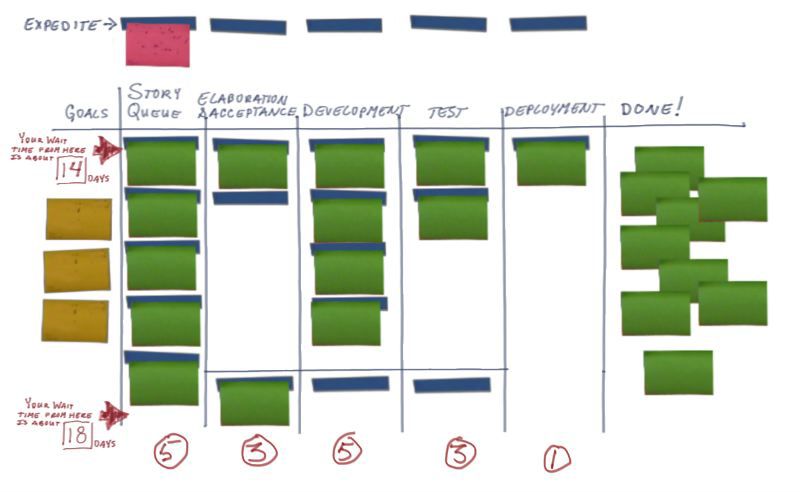
Для начала напишу про происхождение термина **Канбан**.  
  
Этот термин пришёл к нам из Японии благодаря широко известной в узких кругах [производственной системе Тойота](http://en.wikipedia.org/wiki/Toyota_Production_System).

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](http://www.agileproductdesign.com/blog/2009/kanban_over_simplified.html)):  
  
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.