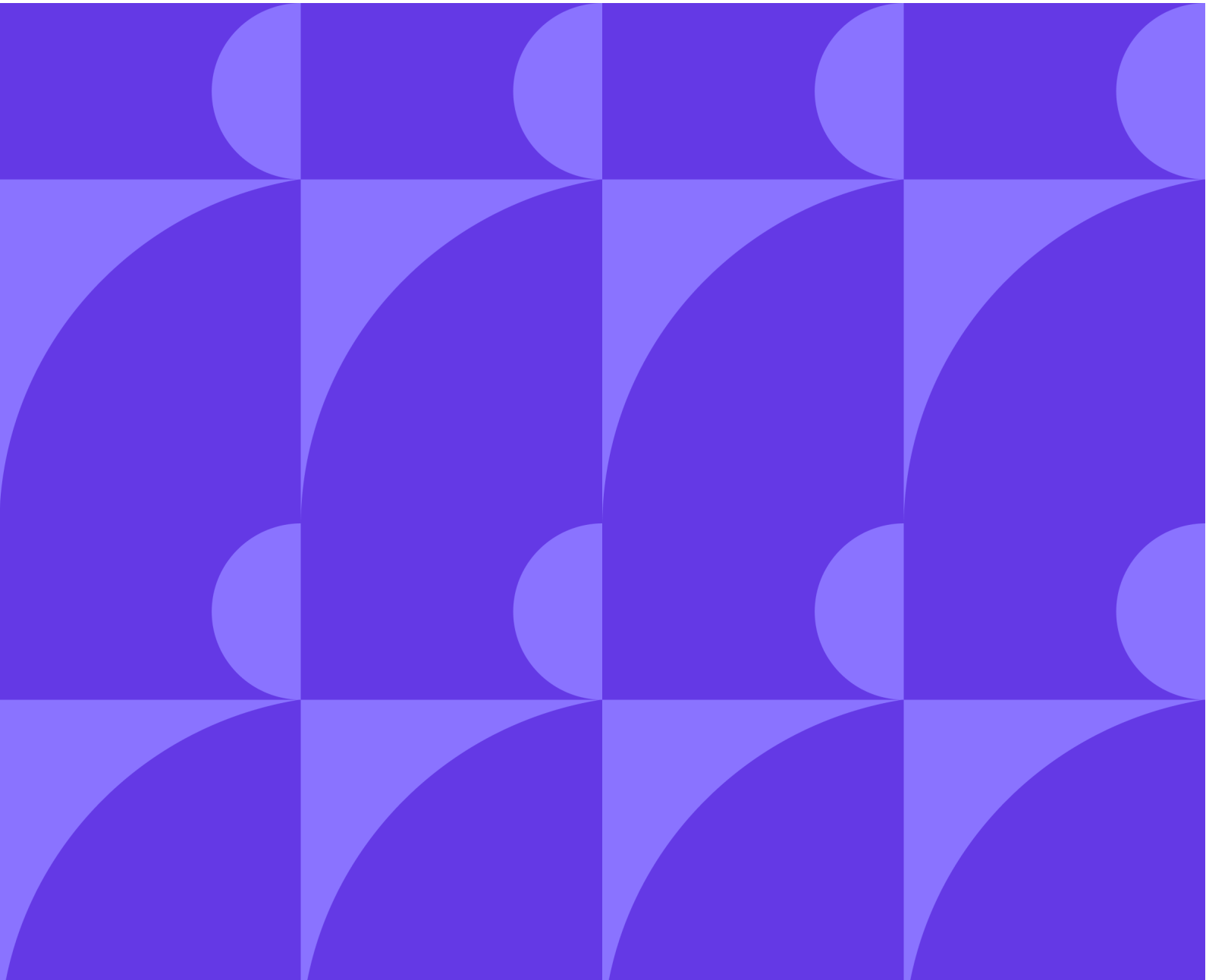
Гибкие методологии

Lean



На этом уроке

1. Узнаем, что такое Lean и как он появился.
2. Поговорим про особенности данного подхода к разработке.
3. Обсудим как строится процесс работы по Lean.
4. Рассмотрим плюсы и минусы применения Lean.
5. Узнаем, как внедрить данный подход в текущую работу команды.

Оглавление

Глоссарий

CI/CD (Continuous Integration, Continuous Delivery — непрерывная интеграция и доставка) — это технология автоматизации тестирования и доставки новых модулей разрабатываемого проекта заинтересованным сторонам (разработчики, аналитики, инженеры качества, конечные пользователи и др.).

MVP (Minimum Viable Product, «минимально жизнеспособный продукт») — это самая ранняя версия продукта, у которой есть минимальный набор функций, достаточный для презентации публике и проверке на первых потребителях.

Потеря – согласно Lean это всё, что не делает продукт ценнее.

Поток создания ценности – последовательность шагов, осуществление которых необходимо для того, чтобы предоставить продукт или услуги пользователям.

Теория урока

1. Что такое Lean и как он появился
2. Принципы Lean и как их применить
3. Преимущества и нюансы Lean
4. Инструменты бережливой разработки
5. Пример внедрения Lean в ИТ
6. Выводы

Что такое Lean и как он появился

История возникновения Lean

Lean — это концепция, которую руководители предприятий используют, чтобы максимально избежать потерь на производстве. По сути дела, Lean – это бережное/бережливое производство.

Концепция lean production или бережливого производства появилась в Японии в 50—х годах XX века на заводе Toyota.

Toyota никак не могла продать свои автомобили из-за сложной экономической ситуации в стране. Как оказалось, дело было не только в этом. Слишком много денег компании уходило на закупку и хранение деталей, исправление дефектов. В результате больших трат на производство получались дорогие машины, которые в Японии были не нужны, а другие рынки уже давно осваивала американская компания Ford.

Стандартные методы решения проблем не помогали исправить ситуацию, поэтому Toyota решила полностью изменить подход к производству машин. Новый подход к управлению производством внедрили Тайити Оно (инженер и предприниматель) и Сигео Синго (промышленный инженер). Они научились быстро переключать автомобильные конвейеры на производство другой модели за несколько часов — другие производители тратили на это недели.

Toyota быстро превратилась из маленькой компании в производителя мирового уровня, и уже спустя 30 лет американские компании начали разбирать по кусочкам историю её успеха. Тогда-то Джон Крафчик в конце 80-х годов XX века и написал свою диссертацию и официально назвал такой подход «Lean production».

Концепция бережного производства прижилась сначала в автопроме и в производстве, а затем появились ответвления в разных отраслях, в том числе и ИТ.

Lean в ИТ

В начале XXI века разработчики Том и Мэри Поппендик решили применить систему Toyota для разработки программного обеспечения. Они были первопроходцами в использовании принципов бережливого производства применимо к ИТ-продуктам.

⚠️ **Внимание!**

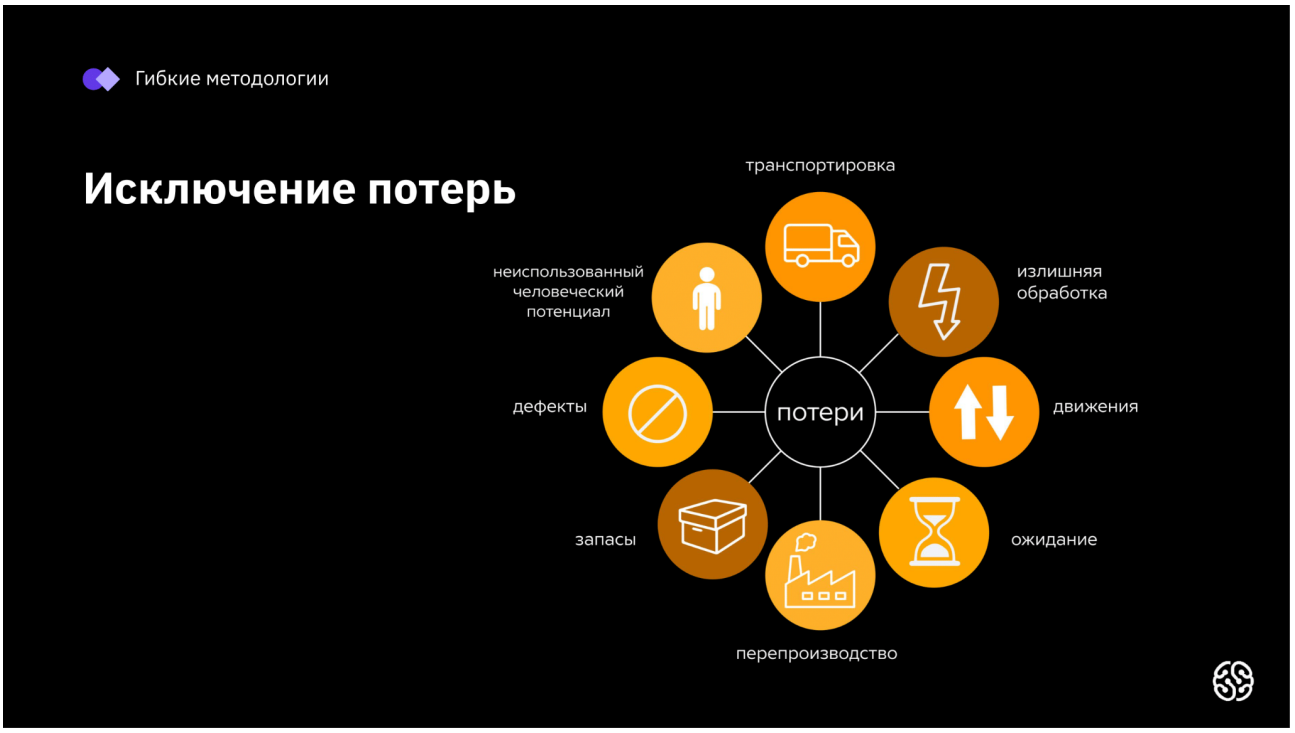
Lean ориентируется на клиента и его потребности, а, значит, при работе над созданием программного обеспечения необходимо сделать полезный продукт, но при этом сократить ресурсы, необходимые для его производства. Это позволит снизить себестоимость разработки.

Затем Мэри и Том Поппендик написали книгу «Lean SoftwareDevelopment», в которой осветили как применить традиционные принципы бережливого производства к разработке программного обеспечения.

Принципы Lean и как их применить

В основе Lean, как бережливой разработки лежат 7 ключевых принципов, которые применимы как для производства продукции на заводе, так и для создания ИТ-продукта (ПО, вэб-сайт, мобильное приложение и т.д.):

1. Исключение потерь



- **Потери из-за перепроизводства**

На производстве это, когда создаются большие запасы готовой продукции и их некуда продать, т.е. произвели больше, чем у нас готовы купить.

А в ИТ это избыточная функциональность. Например, при разработке ПО вы увлеклись и сделали пару фишек, которые оказались никому не нужны, ни заказчику, ни конечным пользователям. В итоге, они есть в нашем продукте, но ими никто не будет пользоваться, да и платить за разработку этих фишек заказчик не хочет, пусть даже они и очень классные.

- **Потери из-за ожидания**

Тут ситуация на производстве и при разработке ИТ-продукта очень схожа.

Это происходит при бесконечных согласованиях и обсуждениях продукта с клиентом. Вы никак не можете начать работать над продуктом – заказчик постоянно на связи, постоянно устраивает встречи, созвоны, что-то согласовывает. При таком подходе времени на саму разработку практически не останется.

- **Потери при ненужной транспортировке**

На производстве такое возникает, когда не отлажены процессы логистики запасных частей, комплектующих и прочих деталей, необходимых для сборки какого-то оборудования, машины и т.д.

А в ИТ потери при ненужной транспортировке – это передача данных между большим количеством людей и систем. Нужно проанализировать, правильно ли выстроены процессы передачи кода, каких-то данных, есть ли в этой цепочки ненужные звенья.

Основная цель – сделать передачу информации при разработке продукта быстрой без лишних этапов согласования, если они не несут в себе никакой ценности для разработки.

- **Потери из-за лишних этапов обработки**

На производстве это создание продукции или оказание услуг с теми качествами, которые потребителю не нужны, и за которые он не готов платить.

В ИТ это дополнительные работы по созданию продукта, например, документирование, согласование, планирование, составление отчётов.

- **Потери из-за лишних запасов**

На производстве это приобретение и хранение излишних объёмов материалов, которые пока не нужны. Они занимают место на складах и «замораживают» деньги.

В ИТ аналогом складских запасов выступает незавершённая работа, которая потребила ресурсы, но не принесла ценности клиенту. Код, который долго остаётся незавершённым, висит мёртвым грузом, устаревает и приходит в негодность Вложенные в его создание время, усилия и средства оказываются потраченными в пустую.

Основное правило бережливой разработки — это постоянный контроль за тем, чтобы в программе не было незавершённого кода, т.е. либо код дописан и работает, либо его не нужно писать совсем. Это касается и других операций при создании продукта – дизайн, вёрстка и т.д.

- **Потери из-за ненужных перемещений**

На производстве это ненужные движения людей из-за нерациональной организации или хаотичности расположения рабочего пространства.

В ИТ такие потери из-за ненужных перемещений происходят при передаче ответственности.

Например:

Вы ведёте разработку, распределили зону ответственности между членами команды, а затем начинаете передавать задачу или целый проект одного человека другому при этом без каких-то серьёзных причин на это.

Также такие потери возможны при многозадачности команды. Вы заставляете переключаться команду с одной задачи на другую. Естественно, сделать это быстро у человека не получится. Вы теряете время.

Ещё одним ярким примером потерь из-за ненужных перемещений является использование множества мессенджеров для обмена корпоративными данными. Работники переключаются с одного мессенджера на другой, обсуждая один тот же продукт в разных мессенджерах – в итоге теряется последовательность обсуждения, а по прошествии нескольких дней и вообще невозможно вспомнить, с чего всё начиналось. А это потеря времени, которую нужно исключить.

- **Потери из-за дефектов**

Здесь всё предельно понятно, при написании кода, отрисовки дизайна возникают баги, ошибки, которые нужно будет устранять, а это съедает ваши деньги и время.

- **Потери из-за нереализованного потенциала сотрудников**

Нет ничего хуже, чем нереализованный потенциал сотрудника. Каждый человек должен быть на своём месте и постоянно совершенствоваться. Когда Вы нагружаете человека длительное время только однотипными задачами, он может загрустить и потерять энтузиазм к работе. Это приведёт к замедлению сроков выполнения им своих задач и отразится на качестве разрабатываемого продукта.

2. Акцент на обучении

Чтобы создать качественный продукт, каждому члену команды нужно самосовершенствоваться, повышать свои навыки, проходить обучение, курсы повышения квалификации. При этом, руководитель должен обеспечивать команду временем и ресурсами для этого, дабы не тратить время потом в активной фазе разработки продукта.

По сути, это своеобразные инвестиции в команду, которые окупятся при работе над продуктом.

3. Предельно отсроченное принятие решений

Полагаться только на факты, никаких прогнозов и предположений. Иногда поздно принятое решение может испортить всю проделанную работу, но для Lean откладывать принятие решения до последнего – это способ собрать как можно больше информации о вопросе. А значит, быть уверенным в его правильности и избежать ошибок.

Значит, пока Вы не будете обладать полной картиной всех обстоятельств, не спешите делать выводы и менять ранее принятые решения.

4. Предельно быстрая доставка заказчику

Этот принцип предполагает, что необходимо получить как можно раньше обратную связь от заказчика. Чем раньше команда покажет свои наработки заказчику, тем быстрее получит от

него обратную связь. Разработчики будут уверены, что все делают так, как хочет клиент, или смогут все изменить при необходимости.

5. Мотивация команды

Человеческое отношение, интересные задачи, внеплановые развлекательные мероприятия, тимбилдинги — это всё просто необходимо, чтобы заряжать команду и создавать благоприятный эмоциональный фон для работы над продуктом.

6. Интегрирование

Информационный поток должен быть постоянным в обоих направлениях — от заказчика до разработчиков и обратно, чтобы вываливание большого объёма информации не вызвало шок.

Опять же, тут идёт отсылка к быстрой поставке частей продукта – сделали что-то, покажите заказчику, при необходимости скорректируйте и идите дальше. Не копите свои наработки и не сбрасывайте их в конце разработки на заказчика как снежный ком.

7. Целостное видение

Убедитесь, что каждый человек в команде понимает и видит продукт так, как это должно быть — введите своеобразную стандартизацию. Тут уже необходимо поработать руководителю команды, потому что именно он должен будет донести до команды все требования и наглядно, даже при необходимости, визуально показать, что хочет заказчик и как это должно работать. Таким образом, нужно привести все видения членов команды в единую плоскость, синхронизировать работу над продуктом.

Исходя из вышеприведённых примеров Lean обладает своими преимуществами и нюансами. Именно нюансами, а недостатками. Давайте поговорим про это на примерах.

Преимущества и нюансы Lean

Очевидным преимуществом будет то, что в проекте Вы точно быстро получите MVP. При этом, расходы получатся меньше, а команда захочет сделать MVP идеальным.

Но есть следующие нюансы:

Чтобы избежать всех потерь и придерживаться принципов, ваша документация должна быть абсолютно точной, а сотрудники высококвалифицированными, с огромным багажом в своей области, ведь обучение на ходу неприемлемо — это поставит под угрозу проект.

Lean как методологию очень хорошо использовать при обновлении давно работающего продукта. Переписать движок с использованием новых технологий или поколдовать над

устаревшим интерфейсом — с этим Lean справится. Тут даже сама история этой методологии нам говорит, почему это так.

А вот, когда Вам нужно разработать что-то новое, сделать проект с нуля, Lean нужно будет совместить с одной из других гибких методологий, иначе ничего не получится.

Например:

Вы с командой разрабатываете программное обеспечение. Используя изначально Lean наряду с другой гибкой методологией, Вы сможете исключить потери времени и ресурсов, а, значит, ускорить получение первой работоспособной версии своего будущего продукта.

Для того, чтобы реализовать на практики все преимущества Lean, есть определённые инструменты. Они так и называются – Инструменты бережливой разработки продукта.

Инструменты бережливой разработки

Бережливая разработка включает в себя набор инструментов, которые:

- делают рабочий процесс более прозрачным;
- помогают снизить затраты на производство продукта, не теряя в качестве;
- снижают риск потери ресурсов;
- повышают уровень удовлетворения пользователей;
- способствуют мотивированности сотрудников и их вовлеченности в рабочий процесс.

Потери могут возникнуть из-за неравномерной рабочей загрузки, переработок или ошибок в распределении финансов.

⚠️ **Внимание!**

Чтобы эффективно внедрить и использовать бережливую разработку продукта, важно определить наиболее приоритетное направление, которое Вы хотите проработать и усовершенствовать.

Бережливая разработка ИТ-продукта сконцентрирована на максимально быстром получении программного продукта, но добивается этого не за счёт особых техник и методик, как например Scrum, а благодаря сокращению затрат труда, времени и прочих ресурсов.

Практические шаги к внедрению Lean Software Development

Чтобы организовать процесс бережной разработки ИТ-продукта нужно:

1. Стандартизировать программный код

При разработке нужно использовать определенные заранее стандарты в наименовании переменных и классов, создании кода, файловых структурах и т. д.

2. Использовать парное программирование

Парное программирование — практика, когда два человека пишут код на одном компьютере. Оно подразумевает много совместных усилий в работе и постоянной коммуникации. Когда пара разработчиков вместе работает над задачей, они не только пишут код, но и вместе планируют и обсуждают свою работу. Они проясняют всё «на ходу», обсуждают варианты и находят лучшие решения. Такая практика уменьшает количество ошибок – как говорится: «Одна голова хорошо, а две лучше».

3. Встраивать защиту от ошибок

Необходимо сразу же создавать код и пользовательский интерфейс таким образом, чтобы исключить возможность человеческой ошибки, т.е. чтобы кто-то случайно не совершил опасные некорректные действия, так называемая «защита от дурака».

4. Автоматизировать рутинные задачи

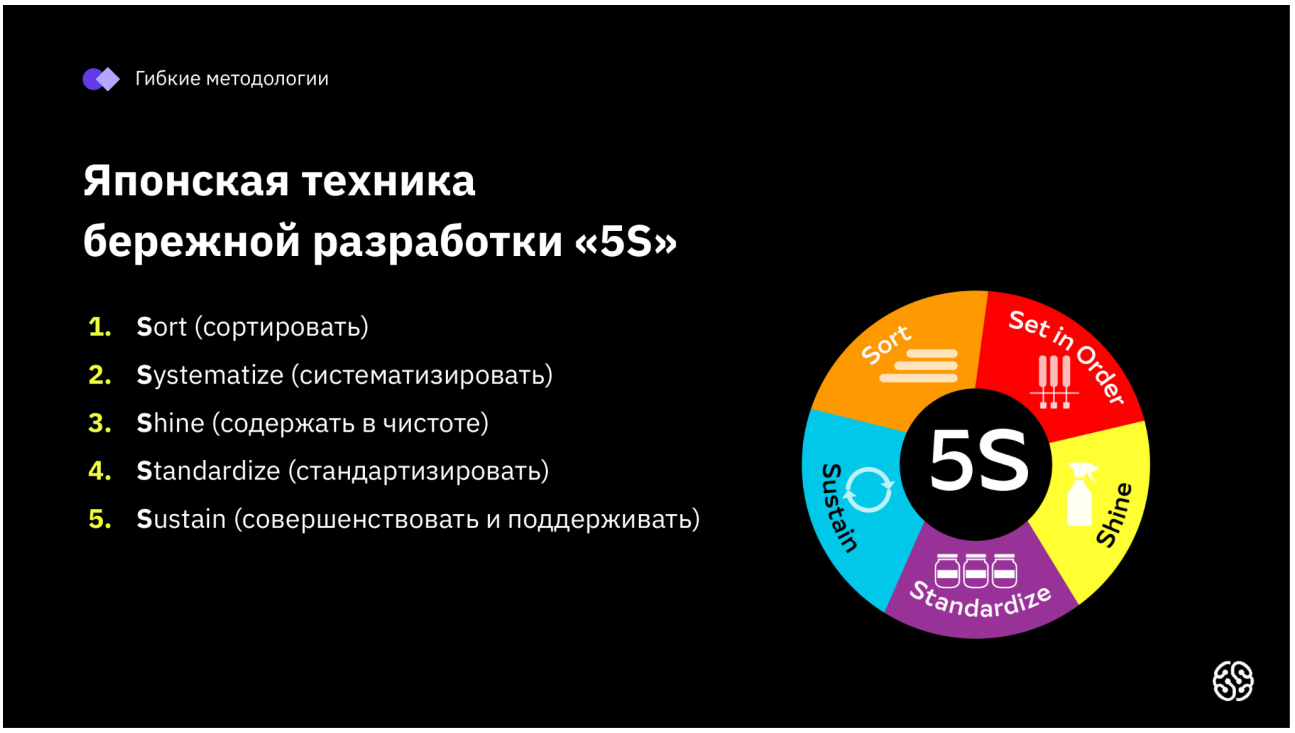
Необходимо создавать скрипты для автоматической сборки новых версий, выполнять сборки по расписанию и, в целом, автоматизировать все задачи, которые только возможно, чтобы избежать потери времени сотрудников на рутину, которую может выполнять компьютер.

5. Практиковать непрерывную интеграцию

На практике это выглядит следующим образом: написали или скорректировали код и сразу же добавили его в общее пространство. Это позволяет каждому разработчику постоянно иметь доступ к наиболее свежей версии каждого фрагмента кода, а также сразу проверять, насколько хорошо работает новый код в общем пространстве.

В ИТ есть специальный практики разработки продукта для этого: CI (Непрерывная интеграция / Continuous Integration) и CD (Непрерывная поставка /Continuous Delivery). Они предполагают развёртывание кода в тестовом пространстве уже работающей системы и позволяют обеспечить последовательную сборку приложений.

5S



Помимо вышеописанных пяти шагов есть специальная японская техника бережной разработки, которая на производстве нацелена на обустройство рабочего пространства сотрудников – это «5S»:

1. sort (сортировать)
2. systematize (систематизировать)
3. shine (содержать в чистоте)
4. standardize (стандартизировать)
5. sustain (совершенствовать и поддерживать)

Давайте посмотрим, как это работает в ИТ.

Эти пять правил позволяют обеспечить идеальный порядок на рабочем месте программиста, в его рабочем компьютере, на серверах и в программном коде.

1. Сортировать

Посмотрите на свой стол и проверьте свой жёсткий диск. Удалите всё ненужное: старые версии программы, неиспользуемый софт, устаревшие отчёты и документы, одним словом, всё то, что будет мешать искать быстро необходимую информацию для работы на текущем

проекте. При необходимости, Вы, конечно же, можете, создать резервную копию всего удалённого.

2. Систематизировать

Затем разложите всё «по полочкам», в случае с ИТ можно сказать по папкам, хранилищам и т.д. – главная цель: навести порядок.

При этом, важно создать удобную и логично устроенную структуру папок, которая позволит быстро находить нужные файлы. Ярлыки часто используемых программ и документов разместите на рабочем столе или закрепите в панели инструментов, редко используемые — переместить в меню «Пуск» или соответствующие подпапки. Придерживайтесь одной системы на всех компьютерах. Рабочая среда должна быть такой, чтобы каждый специалист из команды мог найти в ней необходимое, войдя в систему с любого компьютера.

3. Содержать в чистоте

Мусор всегда отправляйте в корзину – такой подход используйте как для физического мусора, так и для цифрового. Наброски алгоритмов и кода, сделанные на клочках бумаги, стоит перенести в файлы, а бумагу выбросить или подшить в папку (если нужно).

4. Стандартизировать

Нужно обеспечить единую конфигурацию программных средств на всех рабочих станциях. У команды разработчиков, трудящихся над одним проектом, должна быть установлена одна версия среды разработки и программных средств — чтобы не было проблем с совместимостью. На всех машинах должен быть идентичный набор программного обеспечения.

5. Совершенствовать и поддерживать

Теперь соблюдайте такой порядок на протяжении всей разработки. Там, где это необходимо, совершенствуйте наведённый вами порядок.

Картографирование потоков создания ценности

Навели порядок – это хорошо, но этого мало. Теперь нужно найти потери, о которых мы говорили выше. А чтобы это сделать нужно нарисовать (визуализировать) процессы разработки. Это называется Картографирование потоков создания ценности (Value stream mapping, VSM).

Поток создания ценности – последовательность шагов, осуществление которых необходимо для того, чтобы предоставить продукт или услуги пользователям.

Соответственно, нам нужно нарисовать карту потоков создания ценности, которая нам покажет целостную картину того, как работа протекает по всей системе при разработке ИТ-продукта.

Зачем это нужно? Порой членам команды кажется, что они и так знают как устроены процессы и ничего рисовать не нужно.

Очень часто бывает именно так, как на этой картинке:

Создание такой карты потока ценности делится на 3 блока:

1. Производственный или процессный поток — традиционная блок-схема, в которой слева направо фиксируется путь создания ценности продукта. Если кроме основного процесса существуют дополнительные или вспомогательные, они наносятся под основным. Таким образом основные задачи отделяется от второстепенных.
2. Информационный или коммуникационный поток — в верхней части карты потока создания ценности стрелками изображаются потоки информации, которые происходят параллельно с разработкой продукта. Учитывается и формальный и неформальный обмен данными. Информационные потоки наносятся на карту в свободной форме, так, как они протекают в действительности.

3. Timeline и расстояния — линии, которые рисуются в нижней части карты. Линия времени делится на верхнюю и нижнюю части. Сверху отображается Lead time — время ожидания. Снизу наносится продолжительность цикла – Cycle time.

Cycle Time – это время, которое тратится на весь внутренний процесс (сборка и т.п.), а Lead Time – это время, требуемое на весь процесс, т.е. от получения заказа клиента и до доставки конечного продукта.

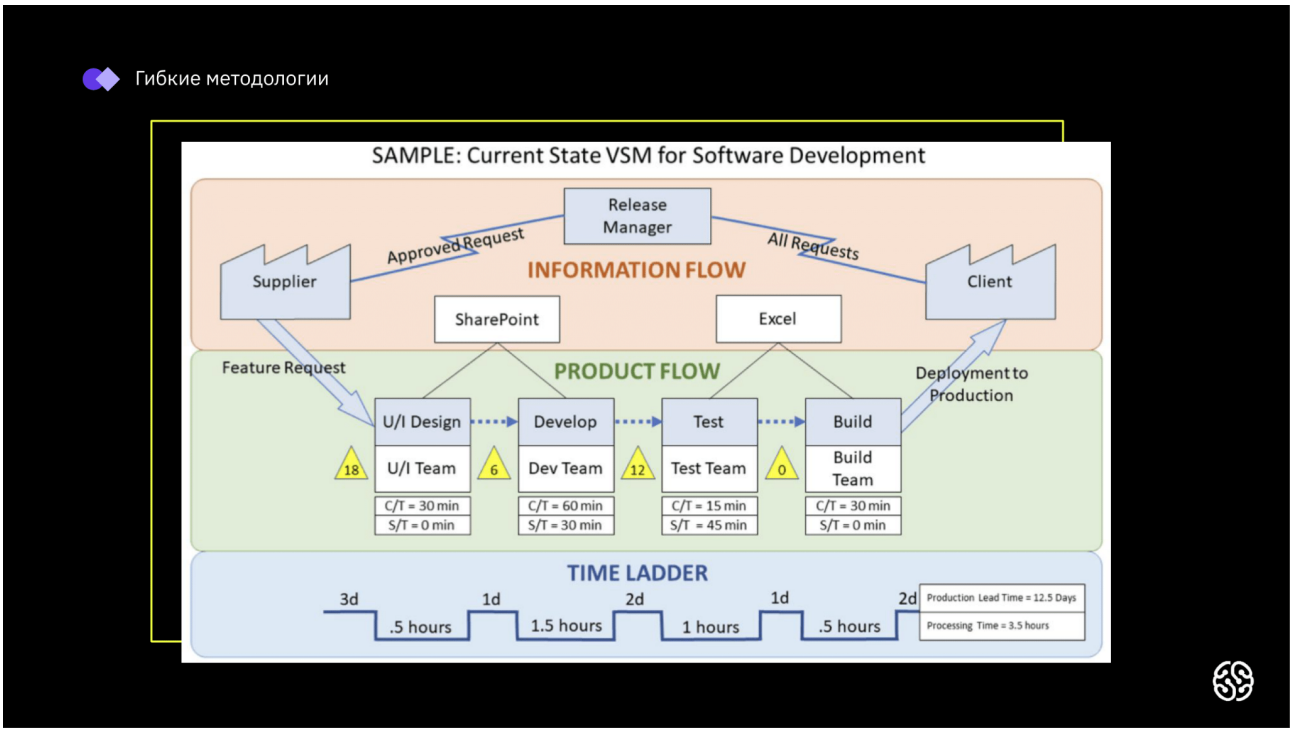
Под линией времени может находиться еще одна линия, в самом низу, показывает расстояния, по которым продукт или персонал движутся внутри процесса.

Для того, что нарисовать такую карту соберите команду – её ещё называют группа VSM.

В команду должны входить те люди, которые могут напрямую вносить изменения в поток. Например, вы захотите избежать ситуации, когда ваша команда состоит только из разработчиков и тестировщиков, если понятно, что необходимы изменения, требующие поддержки со стороны эксплуатации, или если определенные изменения носят организационный характер и требуют участия высшего руководства. Точно также вы не захотите, чтобы группа старшего руководства участвовала в этом упражнении, если цель состоит в улучшении тактических процессов низкого уровня. При выборе команды важно не делать команду по составлению карт слишком большой. Группа из 6-10 человек — самый оптимальный вариант.

Например, группа картографирования может состоять из владельца продукта, мастера разработки, инженера конвейера, архитектора, менеджера по разработке, менеджера по тестированию, менеджера тестовых сред и менеджера по выпуску.

Давайте посмотрим на пример такой карты потока ценности:



Обратите внимание, что в разделе «Информационный поток» группы по разработке и разработке U/I используют SharePoint, а Excel — группы тестирования и сборки.

Возможное решение:

После обсуждения группа VSM решает, что унифицированное решение для обмена информацией увеличит скорость перехода продукта от одной команды к другой и повысит наглядность «конвейера» производства продукта. Обратите внимание, что изолированные информационные «бункеры» (не интегрированные средства хранения и обработки информации), подобные показанным здесь, могут вызывать неэффективность процесса, способствовать образованию узких мест, вносить вклад в плохую коммуникацию между группами и специалистами.

Пример внедрения Lean в ИТ

Учитывая всё вышесказанное хотелось бы привести следующий пример применения Lean при разработке ИТ-продукта:

В компании есть ИТ-отдел для обслуживания внутренних подразделений компании. К нему регулярно возникают претензии, потому что он медленно реагирует на инциденты. Руководство собрало группу VSM, нарисовали карту потока ценности. И при анализе работы отдела стало ясно, что его сотрудники вынуждены постоянно оперативно решать возникающие проблемы. Их работа напоминала тушение пожаров.

Но гораздо эффективнее было бы думать проактивно – на опережение, соответственно нужно не тушить пожары, а предотвращать «пожароопасные ситуации».

Поэтому было решено проводить регулярные встречи ИТ-отдела с подразделениями компании и проводить обучения людей ИТ-грамотности. По мере устранения этих потерь стало видно, что необходимо разделить техподдержку по направлениям и внедрить системы общего информирования. Затем была создана база знаний и впоследствии реализована система автоматического управления заявками. Это позволило исключить потери времени на поиск нужного специалиста для решения вопроса.

С помощью такого подхода, когда внедрили все вышеперечисленные практики, количество претензий к ИТ-отделу уменьшилось. Загрузка его сотрудников снизилась, и они смогли сконцентрироваться на том, чтобы предупреждать запросы своих коллег.

Из данного примера видно, что внедрить Lean в ИТ-проект помогут следующие составляющие:

- Управление базой знаний

Необходимо чаще актуализировать информацию в любых базах данных – тогда вашим членам команды будет легко её находить при необходимости.

- Анализ первопричин

Если понять и устранить первопричину ошибки, то не придётся исправлять её раз за разом. Всегда ищете корневую причину потери, работайте с причинно-следственной связью возникновения проблемы при разработке.

- Масштабирование ИТ-системы

Если растёт компания, то должна расти и ИТ-служба. Это можно делать без затрат на оборудование и персонал – за счёт оптимизации уже имеющихся процессов.

Выводы

Lean – это набор принципов и подходов, которые позволят вам ускорить разработку продукта, применяя подход исключения потерь. При этом, отдельно от другой методологии Lean не работает, его нужно встраивать в текущие методологии управления работой команды.**.**При внедрении Lean в процесс разработки продукта, важно помнить, что это не только технические практики и процессы, но и организационная работа с людьми.

⚠️ **Внимание!**

Качественная разработка ИТ-продуктов начинается не с внедрения технических инструментов и различного ПО, а с изменения человеческого мышления, к примеру, понимания ценности продукта для конечного потребителя и отбрасывании лишнего.

Для быстрого создания качественного продукта необходимо избавиться от всех потерь.

Применяя подход исключения потерь, Вы устраняете мелкие повторяющиеся ошибки, исключаете задачи, которые не принесут клиенту пользы, и каждый раз анализируя, избавляетесь от всего, что не нужно и не приводит к результату.

Для того, чтобы плавно подойти к этому в своей работе, есть практические подходы внедрения Lean в работу своей команды:

1. «Зажгите Andon» – осветите существующую проблему
   В японской культуре Andon – это такая лампа, которую включали, потянув за шнурок. Принцип основан на том, что нужно «осветить» проблему, которая отрицательно влияет на качество продукта или процесса.
   Например:
   Если при внедрении нового ПО несколько раз происходит одна и та же ошибка, то необходимо остановить весь процесс и устранить проблему один раз, чтобы не сталкиваться с ней постоянно.

⚠️ **Внимание!**
Зажечь светильник в терминах бережливого производства означает «осветить» проблему, которая влияет на качество результата.

2. «Генчи Генбуцу и Гемба» – решайте проблему на месте
   В Японии была такая практика: молодых инженеров приводили в цех, ставили в нарисованный на полу мелом круг и просили наблюдать за процессом из него. Предполагалось, что когда инженеры «пойдут и начнут работать» (Генчи Генбуцу), то при возникновении проблемы они самостоятельно изучат и/или изменят процесс или место проведения работ (Гемба), чтобы найти истоки проблемы.

⚠️ **Внимание!**
Если есть трудно диагностируемый инцидент, то посмотрите, как выполняется работа, на каком этапе возникает проблема.

3. «Немаваси» – подготовьте почву для изменений

«Немаваси» переводится как «подготовить почву для посадки». Согласно этому принципу, перед большим и важным собранием нужно провести индивидуальные встречи со всеми участниками. Так они придут подготовленными, со своим мнением, аргументами и идеями.

⚠️ **Внимание!**

Серьёзные изменения возможны только тогда, когда все поддержат предложенную идею. До этого момента необходимо работать с каждой заинтересованной стороной, пока вы не придете к согласию.

Используемые источники

1. Джеймс П. Вумек, Дэниел Джонс «Бережливое производство: Как избавиться от потерь и добиться процветания вашей компании»
2. Мэри и Том Поппендик «Lean Software Development»
3. Сигео Синго «Изучение производственной системы Тойоты с точки зрения организации производства»
4. https://www.techtarget.com/searcherp/definition/lean-production
5. https://www.projectmanager.com/blog/what-is-lean-manufacturing