МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Кафедра «Информационных систем»

Реферат

по дисциплине ТСПП

на тему:

«Методология разработки программного продукта Lean»

Выполнил:

ст. гр. ИСб-22о

Воронин И.Ю.

Проверил:

Строганов В.А.

Севастополь

2016

**Содержание**

Введение 3

1.История возникновения методологии 4

1.1 История развития 4

1.2 Основные публикации 6

2.Содержание методологии 7

2.1 Общее описание 7

2.2 Основные принципы 8

**2.2.1 Ликвидация потерь 8**

**2.2.1.1 Виды потерь 8**

**2.2.1.2 Анализ принципа 9**

**2.2.2 Встраивание качества 10**

**2.2.3 Создание знания 11**

**2.2.4 Откладывание необратимых решений 12**

**2.2.5 Быстрая доставка 13**

**2.2.6 Уважать людей 13**

**2.2.7 Оптимизировать целое 15**

3. Известные применения 16

Заключение 17

Список использованной литературы 18

Введение

В ходе разработки программного продукта крайне важно выбрать подходящую методологию, которая будет учитывать особенности исследуемой проблемы, состава команды разработчиков, сроки и имеющиеся ресурсы.

Каждая из методологий представляет совокупность принципов, которых необходимо придерживаться, а также чётко определённые этапы самой разработки. Также всегда учитываются преимущества и недостатки выбранной методологии.

Lean Software Development – методология разработки программного обеспечения, использующая методы концепции бережливого производства. Является результатом развития Agile-методологии.

На данный момент бережливый подход используется во многих отраслях промышленности. В IT-сфере применяется в том числе и организации стартапов и маркетинга.

1.ИСТОРИЯ ВОЗНИКНОВЕНИЯ МЕТОДОЛОГИИ

1.1 История развития

Зарождение методологии бережливого производство началось в октябре 1949 года в Японии на заводе семьи Тайода. Глава компании Кийтиро Тойода поставил перед собой задачу обогнать американских конкурентов, тем самым заняв свою нишу на мировом рынке. Однако используя американскую модель массового производства Toyota не могла быть конкурентоспособной, так как на Японию в послевоенное время были наложены санкции, ограничивающие производство легковых автомобилей. Массовое производство – это получение прибыли за счет увеличения объемов производства, однако в то время был дефицит сырья, заказов было мало, а требования к разнообразию ассортимента были высокие. В этих условиях получать прибыли, увеличивая объемы производства, было просто невозможно.

Основная идея Кийтиро Тойода состояла в том, что детали на сборочную линию должны поступать ‘‘точно вовремя’’, тогда, когда в них возникнет потребность. Иными словами, детали для сборки брались не со склада, а были произведены непосредственно накануне.

Важно заметить, что данный принцип полностью противоречил всем общепринятым представлениям того времени. Сопротивление усилиям руководителей и сторонников этой идеи было огромным, однако они блестяще предвидели, что перспективы кроются вовсе не в увеличении объемов производства, а в том, чтобы сохранять рентабельность при расширении ассортимента. При увеличении объема вдвое затраты сокращаются на 15-25% на единицу продукции; в то же время затраты растут на 20-35% каждый раз, когда ассортимент удваивается. Однако поставки ‘‘точно вовремя’’ снижают действие факторов, обеспечивающих основной вклад в рост затрат при расширении ассортимента. В действительности это единственная модель производства (среди существующих на сегодняшний день), которая позволяет иметь рентабельное производство при широком ассортименте.

В книге «Study of ‘Toyota’ Production System, Productivity Press» Сигаэо Синго, консультанта, который помогал Тайити Оно, рассматриваются две основные темы: производство без запасов и без контроля.

* Производство без запасов

Подход ‘‘точно вовремя’’ предполагает отказ от запасов (деталей, материалов и т.п.), которые неизбежны при стремлении получить экономический эффект за счет расширения производства. Основная идея в том, чтобы производить все мелкими партиями, а для этого нужно очень быстро перенастраивать оборудование с производства одной детали на производство другой. В разработке программного обеспечения запасами является незавершённая работа. В этом случае снизить объем незавершенной работы можно осуществляя частые релизы создаваемого программного продукта. Например, пользователи вправе ожидать обновления антивирусного ПО на их компьютерах через несколько часов после появления новой угрозы. В этом случае изменения ПО будут небольшими, а потому трудозатраты, связанные с вводом его в эксплуатацию, будут незначительны.

* Производство без проверок

Идея, лежащая в основе автоматизации, состоит в том, что система должна быть создана с защитой от ошибок. Защита от оплошности предполагает, что любая ошибка, которая может быть совершена, в конце концов непременно будет совершена, поэтому изначально следует сделать так, чтобы данная оплошность была невозможна. Во время работы никто не будет искать поломки в машине или проверять качество продукции. Система со встроенной защитой от ошибок не нуждается ни в каких проверках. Например, вам не удастся подключить разъем кабеля монитора к компьютеру перевернутым. И нет необходимости в проверке правильности подключения, так как подключить его неправильно просто невозможно.

В 1990 году в книге The Machine That Changed the World (‘‘Машина, изменившая мир’’) было предложено новое имя для того, что до этого было известно, как поставки ‘‘точно вовремя’’, или производственная модель Toyota (TPS, она же Toyota Production System). С этого момента данный принцип стал называться бережливым производством (lean production). В течение нескольких последующих лет многие компании старались внедрить у себя бережливое производство, однако это оказалось удивительно сложным делом. Как и со всеми предыдущими новыми моделями производства, сопротивление со стороны тех, кто инвестировал в старую модель, было яростным.

В 1993 году бережливый подход к разработке ПО не пользовался большой популярностью ни в одной из отраслей. Методология Scrum явилась первой конкретной реализацией бережливого мышления для разработки ПО, позволившей различным компаниям организовать работу коллективов разработчиков на основе принципов бережливости и с использованием легкой в понимании стандартной модели. Основанная проблема была в том, что трудно было объяснить, как реализовать эту модель, чтобы добиться стабильного повышения качества и производительности

Многие сочли данную концепцию противоречащей интуиции и недостаточно убедительной для того, чтобы менять укоренившиеся привычки. Очень часто система внедрялась частично, например, поставки ‘‘точно вовремя’’ без второй составляющей, известной как ‘‘останов линии’’. Однако в этих компаниях не учли, что по-настоящему бережливым будет являться производство ‘‘... где максимальное количество задач и ответственности передаются рабочим, которые по сути и создают автомобиль и поощряет внедрение системы выявления брака, которая быстро определяет источник обнаруженной проблемы. ’’

Но несмотря на все трудности, связанные с внедрением данного принципа, многие попытки оказались чрезвычайно успешными. Были созданы по-настоящему бережливые производства, которые неизменно процветали.

1.2 Основные публикации

Впервые данная методология была в деталях описана в книге «Lean Software Development» Мэри Поппендик и Toма Поппендика. Были представлены традиционные принципы бережливого производства применительно к разработке программного обеспечения, также набор из 22 инструментов (практик) и их сравнение с гибкой методологией разработки. Мэри и Том участвовали в ряде различных конференций, посвящённых методикам Agile, что объясняет известность концепции бережливого производства среди сообщества гибкой методологии разработки.

Также известна книга под названием «Lean Solutions» (‘‘Бережливые решения’’) авторы которой Джеймс Вумэк (James Womack) и Дениэл Джонс (Daniel Jones)

2.СОДЕРЖАНИЕ МЕТОДОЛОГИИ

2.1 Общее описание

Бережливая разработка программного обеспечения (Lean Software Development) — это методология разработки программного обеспечения, использующая методы концепции бережливого производства.

Бережливый подход к созданию ПО рассматривает все гибкие (agile) методы в качестве эффективных, доказавших свою пригодность применений бережливого мышления.

Однако данный подход предоставляет более широкую перспективу, позволяющую гибким методам быстро развиваться. Во-первых, бережливый подход позволяет держать под контролем всю цепочку создания ценности (от концепции до прибыли), а также выявлять все случаи потерь и задержек, имеющих место до и после разработки кода.

Во-вторых, он создает менеджмент-среду, делающую возможным развитие гибких методов создания программного обеспечения.

В-третьих, бережливый подход предоставляет набор доказавших свою эффективность принципов, которые каждая организация может использовать для внедрения предлагаемых средств, приемов и методов в своей уникальной среде и с учетом своих возможностей.

В свою очередь Lean занял следующее чёткое положение в методологиях разработки ПО. (рис.1)

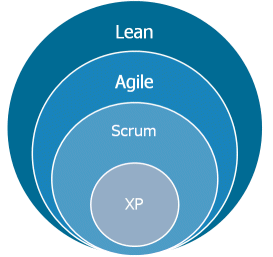


Рисунок 1 – Иерархия методологий разработки

2.2 Основные принципы

**2.2.1 Ликвидация потерь**

**2.2.1.1 Виды потерь**

Большое внимание в Lean отводится в определении и устранении потерь и причин их возникновения. Изначально данные виды потерь в своей спецификации определялись сугубо производством, однако в программировании можно выделить следующие типы:

1. Перепроизводство

* Пример: Избыточные функциональные возможности.

2. Излишние запасы

* Пример: Частично выполненная работа.

3. Излишняя обработка

* Пример: Повторное приобретения знания, дополнительные шаги разработчика.

4. Ненужные перемещения

* Пример: Переключения между задачами.

5. Выпуск дефектной продукции

* Пример: Баги, которые не были выявлены при тестировании.

6. Ожидание

* Пример: Задержки, связанные с ожиданием решений или клиентов.

7. Ненужная транспортировка

* Пример: Передача проекта, требований, знаний, развертывание систем.

**2.2.1.2 Анализ принципа**

Все сводится к совершенствованию графика выполнения работ с момента получения заказа и до получения оплаты за выполненный заказ. Процесс разработки программного обеспечения, основанный на принципах бережливости, фокусируется на сокращении сроков выполнения заказа путем ликвидации всех потерь

Для ликвидации лишних потерь прежде необходимо их выявить.

Первый шаг в ликвидации подобных затрат состоит в том, чтобы понять, что собой представляет потребительская ценность. Интуитивно понятно, что ценность — это то, от чего горят глаза у ваших клиентов и что в конечном счете приносит деньги вашей компании.

Запасами в процессе разработки программного обеспечения является незавершенная работа. Неиспользуемый программный код теряется, устаревает, содержит проблемы с качеством и ‘‘омертвляет’’ вложенные средства. И как раз таки в незавершённой работе кроется большинство рисков, связанных с созданием программного продукта. Данный код также будет нуждается в тестировании, документировании и поддержке. Он повышает неустойчивость всего программного кода, трудность его понимания и совершенствования в будущем.

Однако значительно больший источник непроизводительных затрат при разработке ПО - это избыточные функциональные возможности. Только около 20% функциональных возможностей продукта используется регулярно, и около двух третей возможностей используется редко. Речь идет о функциях, особой необходимости в которых нет изначально. Добавление избыточных функциональных возможностей в программное обеспечение обходится чрезвычайно дорого. Они увеличивают сложность программного кода, затрудняя его последующее совершенствование, и тем самым сокращают ‘‘срок его жизни’’.

Важно, чтобы заказчик при составлении ТЗ соблюдал конкретные рамки. Часто в этот список требований включают все, что только можно. И очень часто меры, призванные ограничивать масштабы создаваемого программного обеспечения, дают противоположные результаты.

**2.2.2 Встраивание качества**

С самого начала целью должно быть встраивание качества в программный код (build quality into the code), а не тестирование этого кода после его создания. Необходимо прилагать все усилия, чтобы дефекты просто не попадали в код. Однако такое возможно только при наличии высокого уровня дисциплины в компании.

Согласно Сигаэо Синго, возможны две разновидности контроля:

* контроль с целью обнаружения дефекта (когда он уже имеется)
* контроль с целью его предотвращения.

Для достижения наилучшего качества необходимо предотвратить появление дефектов, а не устраивать контроль (или тестирование) после того как факт имел место. А если это не представляется возможным, необходимо будет проверять продукт после каждого небольшого этапа разработки с тем, чтобы дефекты выявлялись немедленно после их появления. По обнаружении дефекта вы ‘‘остановите производственную линию’’, выясните его первопричину и немедленно устраните дефект.

Стоит отметить, что целью процесса тестирования должно быть предотвращение дефектов, а не поиск их. В организации, где качеству уделяется должное внимание, работа должна быть ориентирована на удаление дефектов из программного кода в процессе его создания (если угодно, это можно назвать встраиванием качества в код), а не на проверку качества кода позже. Тем не менее заключительная проверка кода не будет излишней. Просто выявление дефектов во время заключительной проверки должно быть исключением, а не правилом.

**2.2.3 Создание знания**

Один из аспектов ‘‘водопадной’’ модели разработки состоит в том, что некоторое знание в форме ‘‘технических условий’’ создается до и отдельно от создания программного кода.

На практике реальная архитектура программного обеспечения всегда материализуется в процессе разработки, так как при предварительном составлении невозможно учесть ни всей сложности, с которой приходится сталкиваться в процессе ее реализации, ни обратной связи с процессом разработки. Что еще хуже, заданная предварительно и подробно архитектура не чувствительна к обратной связи с инвесторами и заказчиками. Процесс разработки, направленный на создание знания, предполагает, что архитектура программного обеспечения возникнет во время создания программного кода, и исключает затраты на его предварительное определение.

Важно чтобы процесс разработки поощрял систематическое и целенаправленное приобретение опыта, но и также не стоит забывать совершенствовать сам этот процесс. В организации, где внедрены принципы бережливости, знают, что необходимо постоянно совершенствовать используемые процессы, поскольку в комплексной среде проблемы неизбежны. Усилия по совершенствованию процесса разработки должны стать стремлением и ответственностью коллектива разработчиков, и в каждом коллективе должно выделяться время специально для работы над совершенствованием этого процесса на регулярной основе.

Разработка программного обеспечения известна своей непредсказуемостью, поэтому на соответствующие организации постоянно оказывается давление с целью сделать их более предсказуемыми.

Для увеличения предсказуемости результатов требуется уменьшить количество прогнозов, на основе которых и принимаются решения. Решения, основанные на фактах, а не на прогнозах, ведут к наиболее предсказуемым результатам. По этой же причине эмпирический подход даёт более предсказуемые результаты, чем детерминистский. В целом, команда, быстро и корректно реагирующая на события, даст более предсказуемый результат, чем та, в которой пытаются предсказать будущее.

**2.2.4 Откладывание необратимых решений**

Аналогично принципу создания знания, та же логика применима для необратимых решений, которые приходится принимать в процессе создания программного обеспечения, а именно откладывать принятие подобного решения на последний момент, прежде чем будет слишком поздно. Собственно, не стоит вообще торопиться с любыми решениями, а лучше при этом всём сделать так, чтобы все решения были обратимы. Из чего следует, что определяя заранее свойства системы, по возможности необходимо избегать принятия решения, которые трудно было бы изменить. Нет необходимости в абсолютной гибкости продукта, однако необходимо, чтобы в определенных точках процесса (где наиболее вероятны изменения) сохранялась возможность выбора того или иного варианта.

Многие предпочитают принимать решения быстро с тем, чтобы сразу уменьшить неопределенность. Однако при неопределенности, особенно когда она сопровождается сложностью, наиболее успешный подход заключается в том, чтобы экспериментировать с различными вариантами, откладывая принятие решения до последнего момента. В действительности многие лучшие стратегии создания программного обеспечения изначально нацелены на то, чтобы оставлять возможность выбора одного из нескольких вариантов как можно дольше, чтобы необратимые решения принимались в последний момент.

**2.2.5 Быстрая доставка**

Аналогично компаниям быстрой доставки, программный продукт должен поставляться настолько быстро, чтобы у заказчика не оставалось времени на то, чтобы передумать. В это время **рынок не успевает изменится в большой степени и это даёт возможность соответствовать столь часто меняющимся требованиям.**

Такие компании часто имеют существенное преимущество также и в ценах на свои услуги, так как они смогли ликвидировать большую часть непроизводительных затрат, что и принесло им существенную экономию. Кроме того, у них очень низкий процент брака, поскольку скорость невозможна без высокого качества.

Более того, невозможно реализовать быстрое выполнение заказа без хорошего понимания требования самого заказчика.

Долго считалось, что для достижения высокого качества необходимо снизить темпы и проявлять осторожность. Однако, отказавшись от данного стереотипа компания может добиваться высокой конкурентоспособности. Примером могут служить Google и PatientKeeper, программное обеспечение которых всегда поставляется быстро и с высоким качеством.

**2.2.6 Уважать людей**

**Хорошим примером для объяснения данного принципа есть правило, которое негласно соблюдается в компании Microsoft. Звучит оно следующим образом:**

«Если вам поручено отвечать за ту или иную работу, то, следовательно, вы Бог в том, что вам поручено, и никто не имеет права становится на вашем пути.» Никто не имеет кодировать лучше программиста и проектировать лучше проектировщика. Поэтому стоит уважать людей, которые выполняют реальную работу.

Также важны следующие особенности организации структуры компании:

1. Руководитель

В компаниях, где развито уважение к людям, должное внимание уделяется воспитанию творческих руководителей. Подобные руководители затем возглавляют коллективы, и одной из их задач является поощрение заинтересованных, думающих людей и направление их усилий на создание выдающихся продуктов.

2. Технические специалисты высшей квалификации.

Компания, рассчитывающая добиться преимущества над конкурентами в определенной области, должна воспитывать нужных ей специалистов. Если же приглашать сторонних специалистов или же вообще не уделять этому должного внимания, то это непременно приведёт к низкой или средней конкурентоспособности. В компаниях, где преобладает мудрость, уделяют внимание воспитанию своих специалистов и заботятся, чтобы все коллективы состояли из работников, способных достичь поставленных целей.

3. Планирование и контроль, основанные на ответственности. Уважение к людям означает, что коллективу дается общий план и определяются реальные задачи, а дальше коллективу предоставляется свобода действий в их выполнении.

Не бывает ‘‘наилучших методов’’, поскольку не существует процесса, который нельзя было бы усовершенствовать. Но сами процессы должны совершенствоваться теми, кто выполняет соответствующую работу.

**2.2.7 Оптимизировать целое**

Бережливое производство позволяет оптимизировать поток создания ценности с момента принятия заказа и до поставки готового продукта и удовлетворения запросов заказчика. Если организация сосредотачивается на чем-то меньшем, чем поток создания ценности в целом, а точнее оптимизирует отдельные его этапы, можно почти гарантировать, что этот самый поток в целом пострадает. Когда обнаруживается большая задержка выполнения заказа, это объясняется ответственностью, которую никто не хотел на себя брать, одно подразделение организации пыталось переложить ее на другое и об интересах заказчиков в подобных случаях никто не заботится.

3. ИЗВЕСТНЫЕ ПРИМЕНЕНИЯ

За последние десять лет Lean-методологии пережили небывалый рост популярности. В последние годы деятельность компаний Toyota и Honda на рынке США была очень успешной, в то время как автопроизводители из Детройта были заняты реструктуризацией. Например, прибыли Toyota возросли с 8 млрд. долл. в фискальном году, который закончился 31 марта 2003 года, до более чем 10 млрд. долл. в 2004 году, 11 млрд. долл. в 2005 году и 12 млрд. долл. в 2006 году. Все это заставило многие компании еще раз присмотреться к концепции бережливости, чтобы попытаться понять, что стоит за таким устойчивым и непрерывным успехом.

По данным исследования Института комплексных стратегических исследований (ИКСИ) о распространении бережливого производства в России в марте-апреле 2006 года из 735 опрошенных российских промышленных предприятий 32 % использовали японский опыт. В марте-апреле 2008 года был проведён повторный опрос. Применение Lean Manufacturing на промышленных предприятиях России в 2006—2008 гг. на III Российском Лин-форуме «Бережливая Россия». Предприятия, которые первыми начали применять методы бережливого производства: Горьковский автомобильный завод (Группа «ГАЗ»), РУСАЛ, ЕвразХолдинг, Еврохим, ВСМПО-АВИСМА, ОАО «КУМЗ», Челябинский кузнечно-прессовый завод (ОАО «ЧКПЗ»), ОАО «Соллерс» («УАЗ», «ЗМЗ»), КАМАЗ, НефАЗ, Сбербанк России ОАО и др.

ЗАКЛЮЧЕНИЕ

Методология бережливого производства была крайне удачно адаптирована в сфере программирования в виде Lean Software Development (Бережливая разработка ПО). Она включает в себя большое количество методологий, в том числе Agile-методологий, которые в той или иной степени могут быть дополнены и улучшены за счёт данной методологии.

Lean-методология позволяет избежать большого количества производственных и непроизводственных потерь. Сам процесс разработки будет адаптивен к неожиданным изменениям, что также экономит ресурсы. Как и в гибких методологиях, всем сотрудникам уделяется внимание, что улучшает слаженность и оптимизируется вся работа команды в целом.

Результатом работы будет ПО, которое будет соответствовать актуальным требованиям рынка или заказчика; будет простым в понимании его исходного кода, что позволит его легко сопровождать, а также предавать командам сторонних разработчиков; опыт нахождения дефектов будет крайне полезен при последующих проектах.

Данная методология находит себе много новых реализаций и постоянно дополняется опытом компаний, которые её практикуют. Тем самым Lean стал полезным инструментом, который можно применить к любому проекту, дополнив, но не нарушив его внутреннюю организацию.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. П58 Бережливое производство программного обеспечения: от идеи до прибыли. : Пер. с англ. - М. : ООО ‘‘И.Д. Вильямс’’, 2010. - С. 19-85.