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 Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

РЕФЕРАТ

по дисциплине: «Технологии создания программных продуктов»

на тему «Lean – гибкая методология разработки программного обеспечения»

Выполнил

студент 2 курса группы ИС/б-23-о

Генералов Николай Николаевич

Отметка о зачете\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Проверил

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (фамилия, инициалы)

Севастополь

2018

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc514791794)

[ВВЕДЕНИЕ 3](#_Toc514791795)

[1. ИСТОРИЯ ВОЗНИКНОВЕНИЯ МЕТОДОЛГИИ 4](#_Toc514791796)

[2. СОДЕРЖАНИЕ МЕТОДОЛОГИИ 8](#_Toc514791797)

[ЗАКЛЮЧЕНИЕ 13](#_Toc514791798)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 14](#_Toc514791799)

# ВВЕДЕНИЕ

Разработка – это процесс, при котором начальные идеи, в результате человеческой деятельности, трансформируются в конечные продукты. Подходы к процессу трансформации, можно разделить на детерминистский и эмпирический.

Детерминистский подход, используется в классических методологиях создания программной продукции. Он подразумевает полное описание продукта и требований к нему, на первом этапе разработки. На следующем этапе, , на основе этого описания, происходит непосредственная разработка данного продукта.

Эмпирический подход, на начальном этапе предполагает создание концепции будущего продукта, а затем, уже в процессе разработки, приобретение свойств, подробно описывающих данный продукт, пока он не будет готов.

Lean Software Development – один из подвидов гибкой методологии, который основывается на эмпирическом подходе, позволяющем легко адаптировать продукт к изменения, как при разработке, так и в процессе его эксплуатации. Большая часть программного продукта разрабатывается, непосредственно после его первого релиза, отнеся последующие затраты к затратам на его поддержку.

Данная методология акцентирует свое внимание, на сохранении высокого морально-функционального состояния разработчиков, а так же минимизации «потерь», которые имеют особое определение в рамках данной методологии.

В основу методологии Lean положено семь основных принципов, которых рекомендуется придерживаться в совокупности, для достижения наиболее эффективных показателей при внедрении данной методологии. Данные принципы часто пересекаются с общими принципами гибких методологий, в частности с принципами экстремального программирования.

# ИСТОРИЯ ВОЗНИКНОВЕНИЯ МЕТОДОЛГИИ

В 1785 году, через полтора года, после завершения освободительной войны в Америке, потребность в оружии беспокоила многие государства. В это время в Париже, один из выдающихся оружейных мастеров провел демонстрацию, сборки ружейных замков. В процессе сборки обнаруживалось, что детали разных мушкетов, были совместимыми и легко заменялись деталями других. Была продемонстрирована возможность собирать ружья из взаимозаменяемых деталей.

На основе этого, инженер Эли Уитни, создал технологический процесс, позволяющий неквалифицированным рабочим, при помощи соответствующих инструментов, изготавливать шаблонные взаимозаменяемы детали, из которых затем собирались готовые изделия. Этот процесс был положен в основу американской производственной системы, что обусловило последующий экономический рост.

Адаптировал и применил эту идею к персоналу, Генри Форд, после запуска сборочного конвейера для Ford «Model T» в 1914 году. Подняв зарплаты своим работникам в 2 раза и сократив их рабочий день на треть, поспособствовал созданию среднего класса, имеющего свободное время и деньги для приобретения автомобилей.

Так как зарплата рабочих не увеличивалась от эффективности труда, предполагалось, что работник должен самостоятельно организовывать свою часть процесса. Изначально на сборку автомобиля требовалось около 12 часов, после внедрения нововведений требовалось 90 минут. Производство Форда испытывало небывалый подъем.

Однако система производства Генри Форда, была отлажена и эффективна в рамках одной модели. Обучение нового работника занимала около 10 минут, поэтому в центре данной индустрии оказались взаимозаменяемые работники, подобно ранее взаимозаменяемым деталям механизмов. Так как людей было легко заменить, исключалось и уважение к ним.

Возрастающие потребности среднего класса, не могли быть удовлетворены одной моделью автомобиля и эту нишу начали занимать другие производители.

Годами ранее компания Toyoda Automatic Loom Works, начала внедрять американскую производственную систему в производство своих ткацких станков. До внедрения данной системы, станки Toyoda были громоздкими, сложными и трудно обслуживаемыми. После внедрения, владельцу компании, Сакичи Тойода, удалось реконструировать и усовершенствовать процесс производства, используя взаимозаменяемые детали для автоматических ткацких станков.

В 1927 году, отец и сын, Сакичи и Кийтиро Тойода, провели презентацию прядильно-ткацкого цеха, насчитывающего 520 одновременно работающих ткацких станка, для обслуживания которых требовалось всего 20 ткачей. И хотя в производстве станков использовались взаимозаменяемые детали, в нем не было места для взаимозаменяемых работников.

Презентация имела огромный успех и позднее Кийтиро, использовал полученные прибыли для освоения нового бизнеса. Посетив Детройт, он начал изучать, технологии изготовления двигателей.

В послевоенные годы, после отмены на производство легковых автомобилей, у компании Тойода была поставлена задача «догнать Америку», но используя американскую систему производства, которая основывалась на получении прибыли за счет увеличения производства, не было осуществимо, за счет дефицита заказов и сырья. Поэтому Кийтиро Тойода начал разрабатывать собственную идею, сделав упор на расширение ассортимента, при этом избежав проблем с поставками деталей. Позднее эта идея получила название «точно вовремя».

Идея «точно вовремя» заключалась в том, чтобы все детали необходимые для сборки автомобилей, производились мелкими партиями накануне. При таком подходе удавалось избежать накопления и устаревания неиспользованных деталей на складах. В то время эта идея была противоречащей всем устоявшимся принципам. После отставки Кийтиро Тойода, эту идею начал продвигать генеральный менеджер компании Toyota – Тайити Оно.

Тайити Оно, так же внедрил идею «производства без проверок», основанную на том, что система должна быть с созданной защитой от ошибок. Предполагалось, что любая ошибка, которая может быть совершенна, рано или поздно произойдет, поэтому следует сделать, совершение этой ошибки невозможным. Эти идеи были заложены в основу системы, названной «Toyota Production System».

К 80-м годам японские автомобили захватывали мировой рынок, вызывая беспокойство у американских и европейских автопроизводителей, у которых разработка автомобиля занимала в 3 раза больше инженерных часов и на 50% больше времени на производство. Этот фактор вызвал большой интерес в научных кругах, а производственная система Toyota, позже легла в основу принципов бережливого мышления и бережливого производства.

Основателями «Lean Software Development» являются Мэри и Том Поппендик. В 80-ые годы Мэри работала над крупными проектами, включающих создание и запуск сложных технологических линий в компании по производству магнитной ленты, создавая встраиваемое программное обеспечение. Став менеджером информационной системы, она активно использовала точные, легко настраиваемые процессы для управления сложными проектами по разработке программного обеспечения, построенные на принципах схожих с принципами бережливого производства. Однако позже, при работе над государственным проектом, произошло столкновение с «водопадной» (каскадной) методологией, налагающей жесткие ограничения на весь процесс разработки, и впоследствии оказавшимся неуспешным даже для проектов небольших размеров. Тогда было положено начало развития идеи «Бережливой разработки программного обеспечения».

В 1999 году Кент Бэк, в своей книге «Extreme Programming Explained», предлагал совершенно новый подход к разработке программного обеспечения. Принципы и конкретные практики, представленные в книге, были нацелены на устранение недостатков в классических методологиях разработки. Одним из таких недостатков была практика передачи больших запасов информации между этапами проектирования, разработки, тестирования и развертывания, накапливаемой на каждом из них. Существующие методологии предполагали тестирование и интеграцию модулей в конце цепочки разработки, поэтому дефекты могли быть не обнаруженными на протяжении долго времени. В 2001 году, ранее представленному подходу к разработке программного обеспечения, Кент Бэк дал название «Agile».

В 2003 году бережливые принципы производства и практики Agile методологии были объединены в книге, написанной Мэри и Томом Поппендик – «Lean Software Development: An Agile Toolkit». В данной книге были представлены семь основополагающих принципов бережливой разработки программного обеспечения.

Типичный гибкий подход к разработке программного обеспечения, большую часть времени, пытался оптимизировать процесс разработки, сосредотачивая внимание на отсутствие формальных отношений, в свою очередь бережливый подход работал над оптимизацией эффективности процесса разработки целиком – «от концепции к прибыли», что было отраженно в следующей книге Мэри и Тома - «Implementing Lean Software Development: From Concept to Cash» (Поппендик, 2006).

# СОДЕРЖАНИЕ МЕТОДОЛОГИИ

На развитие бережливого подхода к разработке программного обеспечения, оказали действие гибкие методологии, в частности экстремальное программирование. Как будет видно из описания основных принципов, они часто пересекаются.

Принцип №1. Ликвидация потерь.

Процесс разработки, концентрируется на снижении времени разработки заказа, путем ликвидации потерь. При этом первоочередной задачей является выявление этих самых потерь.

В методологии Lean, «потери» или «затраты» наделены особым смыслом. Эти понятия охватывают все, что не помогает предоставить заказчику то, что ему необходимо. Сюда же относятся любые задержки по выполнению заказа.

Как на примере бережливого производства, источником затрат являются запасы, которые необходимо хранить, вести учет, перемещать, а так же они могут скрывать дефекты и устаревать, что снижает их стоимость. При разработке программного обеспечения, роль запасов играют не законченные проекты, которым свойственны все те же недостатки. Часто, из-за долгих релизов программного обеспечения, у заказчиков меняются требования, что может привести к переделыванию уже созданных модулей. Одним из наиболее часто встречающихся видов потерь являются избыточные функциональные возможности. Данный вид потерь является чрезвычайно дорогим, так как он увеличивает сложность программного кода, затрудняя его дальнейшую модификацию и поддержку.

Принцип №2. Встраивание качества.

Большую роль при бережливом подходе к разработке ПО играет дисциплина. Ее цель является внедрение качества в программный код, а не тестирование кода после его создания. Следует проводить контроль кода от дефектов.

Условно контроль можно разделить на контроль с целью обнаружения дефекта и контроль с целью предотвращения дефекта. В бережливом подходе имеет место второй вид, согласно которому системы, обнаруживающие дефекты, способствуют накоплению работ, которые нуждаются в исправлении и являются источниками непроизводственных затрат. В данной концепции целью является устранение дефектов в процессе создания кода. Интеграция кода производится совместно с тестами, и если тест не будет пройден, новый код не создается. Данный принцип пересекается с принципом TDD (разработка через тестирование) из методологии экстремального программирования.

Авторы методологии не призывают отказываться от заключительного тестирования, а пытаются донести, что выявление дефектов при заключительной проверке должно быть исключением, а не правилом.

Принцип №3. Создание знаний.

В классических методологиях создания ПО, архитектура создаваемого продукта должна быть определена на начальном этапе, до создания кода, но правильность этой архитектуры подтверждается только после. Так как при предварительном проектировании архитектуры сложно предусмотреть все возможные проблемы, которые могут возникнуть в процессе разработки, данный принцип, предполагает, что архитектура определяется в процессе создания кода.

Компании, создают знания, в процессе разработки программного обеспечения, сохраняют в удобном виде и накапливают, для последующего доступа. Это касается не только решений, которые уже были успешно применены, а так же выявления проблем в процессе разработки и средств их устранения.

Принцип №4. Предельно отсроченное принятие решений.

При разработке программного продукта, следует избегать необратимых или трудно обратимых решений. Одним из возможных подходов при возникновении ситуации требующих сложного выбора, является быстрое принятие решения, чтобы сразу уменьшить неопределенность.

Данный принцип заключается в том, чтобы провести эксперименты с различными вариантами, откладывая принятие решения до последнего момента. Конечно, это требует наличия определенных способностей у коллектива или руководителя, для оценки ситуации, и определения подходящего времени для принятия критического решения.

Принцип №5. Быстрая доставка.

Компании, с высоким темпом производства и доставки своей продукции, как правило имеют большое преимущество, перед своими конкурентами, и это касается не только производства физических товаров. Доставка продукта в кратчайший срок, способствует снижению вероятности отказа от него и сохранению прибыли.

При производстве программного обеспечения, данный принцип предлагает частый релиз версий и доставку программного обеспечения настолько быстро, чтобы у заказчика не оставалось времени на то, чтобы передумать.

Для организации этого принципа, требуется организовать работу так, чтобы работники без указаний знали, что им необходимо делать в той или иной ситуации, и решали проблемы, не дожидаясь разрешений.

Принцип №6. Уважение к людям.

Для более глубокого понимания, авторы книги приводят принципы, касающиеся людей, системы разработки продукции Toyota:

Руководитель-предприниматель.

В компаниях, где развито уважение к людям, зачастую уделяется много внимания для развития творческих способностей руководителей.

Технические специалисты высшей квалификации.

В данной системе большое внимание уделяется воспитанию собственных специалистов, подразумевая, что специалисты дают конкурентное преимущество. Так же внимание уделяется заботе о том, чтобы рабочие коллективы, были способны достичь поставленных целей.

Планирование и контроль, основанные на ответственности.

Данный принцип предполагает постановку общего плана и определение основных задач, а коллективу предоставляется свобода действий для их достижения.

Принцип №7. Оптимизация целого.

Как правило, в процессе разработки ПО, оптимизируются отдельные части системы или стадии процессов. Бережливое производство акцентирует свое внимание на том, что необходимо оптимизировать весь поток создания ценности от принятия заказа, до передачи готового продукта заказчику и удовлетворения его запросов, при сопровождении. Для этого требуется глубокое понимание поставленной задачи.

Структурирование контрактов, привлечение внешних подрядчиков и взаимодействие с различными компаниями должно проходить при совместных стимулах – оптимизировать целое.

Отличительной особенностью бережливого подхода, является анализ производственных потерь на каждом этапе разработки ПО, а так же практика, нацеленная на оптимизацию процесса разработки, называемая «Канбан».

Система «Канбан» представляет собой поток создания ценности и отображается в виде диаграммы, состоящей из столбцов, каждый из которых соответствует своему этапу из потока. Маркеры диаграммы, размещаются на столбцах, отражающих текущую стадию разработки. Когда достигнуты определенные критерии, означающие завершение текущего этапа, маркер перемещается на следующий столбец, со временем пройдя поэтапно весь поток. Главной особенностью является ограниченность системы, определенным объемом работ.

Использование «Канбан» системы позволяет определить и изучить узкие места разработки, что впоследствии поможет при оптимизации потока создания ценностей в целом.

Основными недостатками бережливой методики разработки ПО, может оказаться то, что очередной выпуск разрабатываемого продукта, будет отложен в связи с возникновением ошибки, которая не встречалась ранее и потребуется время для её анализа, выявления факторов повлиявших на её появление, решения и документирования. Так же в борьбе за дисциплину и скорость разработки, требуется выработка стандартизированного процесса, который способствует появлению работников, обладающих высокой квалификацией, необходимых для правильного функционирования системы в целом.

# ЗАКЛЮЧЕНИЕ

Бережливый подход, является одним из гибких подходов к разработке ПО. Он имеет тесную связь с методикой экстремального программирования, а его отличительной особенностью является ликвидация потерь, во всех их проявлениях начиная от непредвиденных задержек и заканчивая излишней функциональностью, которая не требуется заказчику для удовлетворения его нужд. Так же к особенностям данного подхода можно отнести концепцию, принятую от бережливого производства «точно в срок» и недопущение слишком большой нагрузки на сотрудников, выполняющих работу.

Для того чтобы избежать посредственного результата, при использовании бережливого подхода к созданию ПО, рекомендуется проводить внедрение всех принципов данного подхода в совокупности.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Поппендик М. Бережливое производство программного обеспечения. От идеи до прибыли. / М. Поппендик, T. Поппендик. – М.: ООО ‘‘И.Д. Вильямс’’, 2010. – 256 с. ISBN 978-5-8459-1538-2 (рус.)
2. Poppendieck M. The Lean Mindset: Energized Workers, Delighted Customers, Genuine Efficiency, Breakthrough Innovation / [Электронный ресурс] // www.poppendieck.com : Lean Software Development: The Backstory. [2010 – 2015]. URL: http://www.leanessays.com/2015/06/lean-software-development-history.html (дата обращения: 21.05.2018).
3. Mary, Tom Poppendieck. Lean Software Development: An Agile Toolkit. / Mary, Tom Poppendieck – Addison-Wesley Professional, 2003. – 240 с. ISBN: 0-321-15078-3