ТЕХНОЛОГИЯ И ПРОЦЕСС РАЗРАБОТКИ ПО (Л-4)

к.п.н., доцент Касаткин Д. А.
e-mail: kasatkinda@cfuv.ru

Литература

Lean Software Development

История и принципы бережливого производство ПО

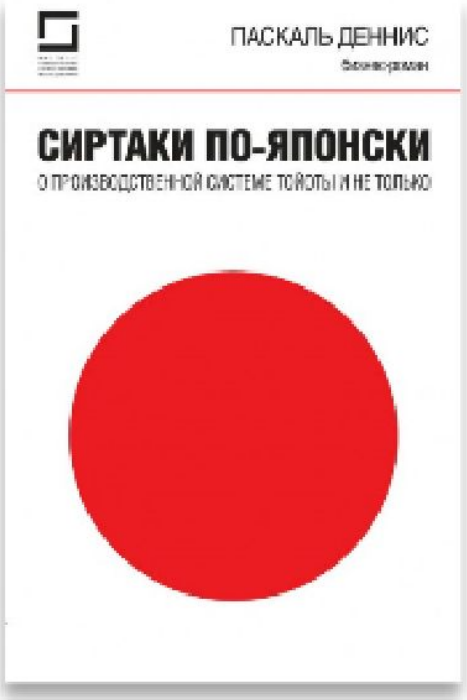
5S для рабочих: как улучшить

Догнать зайца: как лидеры

Изучение производстве

Литература
Lean Software Development

Производственна
я система



Сиртаки по-
японски: о

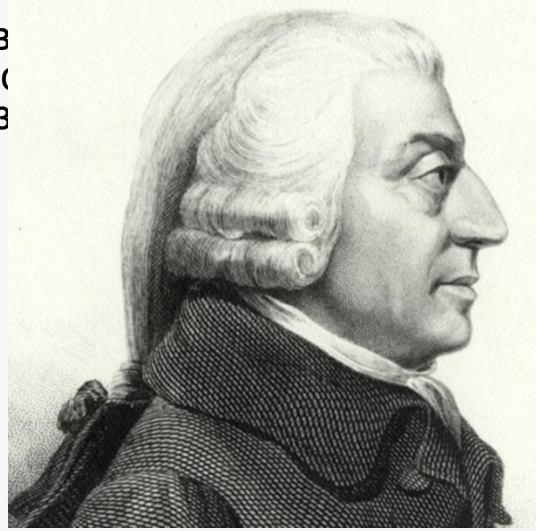
Сломай
стереотип!

История
Lean Software Development

- Adam Smith (1732)
- Eli Whitney (1765)
- Frederick Taylor (1856)
- Henry Ford
- Kiichiro Toyoda (1894)
- Taiichi Ohno (1912)

История
Lean Software Development

- Ада́м Смит родился 16июня 1723, Керколди, Шотландия, Королевств Великобритания — 17 июля 1790, Эдинбург, Шотландия, Королевство Великобритания) — шотландский экономист, философ-этик; один из основоположников современной экономической теории.



1. Свободная торговля.
2. Принцип невмешательства.
3. Разделения труда.

Для увеличения производительности сложную задачу можно разбить на небольшие этапы, на каждом этапе поставить людей, которые будут отлично делать именно этот участок работы.

История
Lean Software Developm

- Eli Whitney (1765)

1. Catton GIN
2. Разделение труда
3. Принцип взаимозаменяемости деталей при сборке.
4. Американская производственная система.

Франция 1785 г. **Honore Blanc (ружья)** – Thomas Jefferson – Eli Whitlney Идем к снижению квалификации работников.

История
Lean Software Development

- Frederick Taylor (1856)

1. Принципы научного управления.
2. Партнерство предпринимателя и работника.
3. Взаимозамещение людей.
4. One best way.

Любой труд может быть проанализирован, систематизирован и передан в процессе обучения любому человеку

Им были недовольны и профсоюзы, и владельцы предприятий.

История
Lean Software Development

- Henry Ford

Ге́нри Форд (30 07 1863 — 7 04 1947)

1. Промышленный конвейер.
2. Стандартные запчасти.
3. Массовое производство.

...время, необходимое для выпуска модели Т сократилось на 12 часов до 2 часов.

История
Lean Software Development

Kiichiro Toyoda (1894)

1. Текстильная индустрия.
2. Автоматических ткацких станков
3. Toyota Motors
4. Догнать Америку, но не моделью массового производства

Один высококвалифицированный рабочий наблюдает за десятком машин. Машины могут работать круглосуточно.

История
Lean Software Development

Taiichi Ohno (1912)

1. Создатель Toyota Production System
2. Рентабельность при широком ассо
3. Отсутствие потерь
4. Just in Time
5. Jidoka (умная автоматизация)

«…смотрим на время от момента получения заказа, до момента получения денег. Мы уменьшаем это время, путем удаления потерь …..»

Виды потерь Lean Software Development

Тайити Оно (1912—1990), один из главных создателей производственной системы компании Toyota, выделил 7 видов потерь:

- **потери из-за перепроизводства;**
- **потери времени из-за ожидания;**
- **потери при ненужной транспортировке;**
- **потери из-за лишних этапов обработки;**
- **потери из-за лишних запасов;**
- **потери из-за ненужных перемещений;**
- **потери из-за выпуска дефектной продукции.**
- Тайити Оно считал перепроизводство основным видом потерь, в результате которых возникают остальные. Джеффри Лайкер, исследователь производственной системы Toyota (наряду с Джеймсом Вумеком и Дэниелом Джонсом), в книге «Дао Тойота» добавил ещё один вид потерь:
- **нереализованный творческий потенциал сотрудников.**
- Также принято выделять ещё два источника потерь — muri (яп. 無理 *му́ри*), — перегрузка рабочих, сотрудников или мощностей при работе с повышенной интенсивностью и mura (яп. 斑 *му́ра*) — неравномерность выполнения операции, например, прерывистый график работ из-за колебаний спроса.

Kanban

1. 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.

EXPEDITE →
GOALS
STORY QUEUE
ELABORATION & ACCEPTANCE
DEVELOPMENT
TEST
DEPLOYMENT
DONE!
YOUR WAIT TIME FROM HERE IS ABOUT 14 DAYS
YOUR WAIT TIME FROM HERE IS ABOUT 18 DAYS
5
3
5
3
1

7 принципов Lean

1. Ликвидировать потери
2. Выстраивать качество
3. Создания знания.
4. Откладывать необходимые решения.
5. Доставлять быстро.
6. Уважать людей.
7. Оптимизировать целое

Ликвидировать потери принципы lean

Проблемы:

- Изменение требований и технический долг.
- Отложенная интеграция
- Отложенное тестирование
- Избыточные возможности (борьба за простоту легкая архитектура).
- Подробные ТЗ

МИФ: Созданная заранее спецификация сокращает потери.

Выстраивать качество принципы lean

- Позднее выявление дефектов
- Очереди дефектов на исправление
- Полностью ручное тестирование
- Ручной выпуск новых версий

Миф: Цель тестирования выявление дефектов.

Создания знания. принципы lean

Проблемы:

- Полное ТЗ до создания прототипа.
- Big Design Up Front
- Поздние релизы, плохая обратная связь.
- Негибкая команда
- Следование плану

МИФ: Прогнозы обеспечивают предсказуемость.

Откладывать необходимые решения

- Откладывание важных решений до полной ясности
- Принимаем решения слишком рано.

МИФ: План – это обязательство!!!

Доставлять быстро

Проблемы:

- Низкая скорость разработки
- Боязнь релиза
- Планы стандарты и спецификация для разработчиков.

МИФ: Спешка ведет к браку.

Уважать людей

- Восприятие разработчиков, как взаимозаменяемые инструменты
- Забираем ответственность у тех кто выполняет реальную работу.
- Микроменеджмент

МИФ: Существует наилучший метод

7 основных потерь по lean

- Недоделанная работа
- Лишняя функциональность
- Повторное изучение
- Передача работы глухой телефон
- Переключение между задачами
- Задержки
- Дефекты

Сравнение процессов Toyota и Microsoft

Бережливое производство Toyota (ручное «вытягивание» в зависимости от спроса с использованием карточек «Канбан»)	Скорая разработка Microsoft 1990-х (ежедневные сборки с совершенствованием функциональности)
JIT-выпуск небольших партий	Разработка малых функций
Минимум промежуточных запасов	Короткие циклы и интервалы между вехами
Географическая концентрация — производство	Географическая концентрация — разработка
Равномерное производство	Планирование по функциям и вехам
Быстрая наладка	Автоматизированные средства сборки и быстрые тесты
Оптимизация оборудования и линий	Ориентация на малые, мультифункциональные команды
Стандартизация работы	Стандарты проектирования, кодирования и тестирования
Простые в использовании средства автоматизации	Сборки и непрерывное интеграционное тестирование
Рабочие-«многостаночники»	Несколько обязанностей у каждого участника
Избирательная автоматизация	Автоматизированные средства, но без генераторов кода
Непрерывное улучшение	Анализ по методу «посмертного вскрытия», развитие процесс