SCRUM
ВЗГЛЯД РАЗРАБОТЧИКОВ

ИЛИ ДАЙТЕ МНЕ УЖЕ **ПОКОДИТЬ**!

АЛЕКСАНДР ГЕРГАРДТ

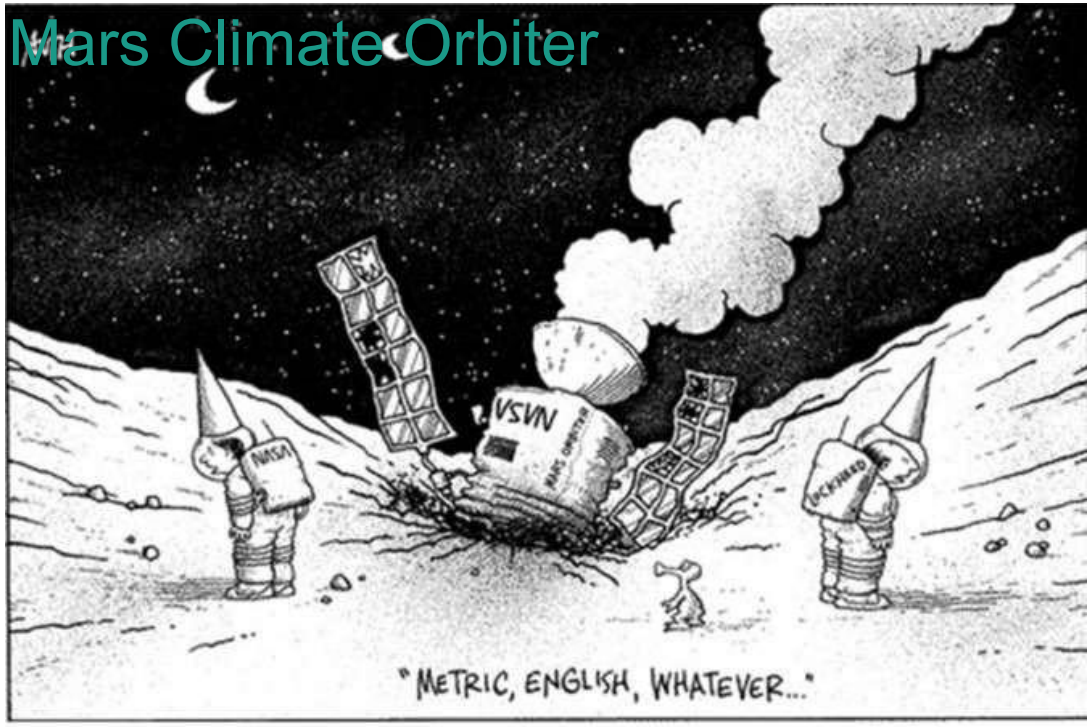
Mars Climate Orbiter

Один из лучших уроков, которые я освоил:

“по умолчанию” - точка наибольшего количества ошибок

ONIX

Mars Climate Orbiter



Remember the Mars Climate Orbiter incident from 1999?

ONIX

Команда разработки о своем понимании Scrum

Достали уже эти митинги, фасилитации и так далее... Можно меня, простого инженера, не смешивать с этим?

ONIX

Мое представление и мечты

Мы используем Scrum потому что:

1. Его просто понять и использовать
2. Помогает сохранить время, деньги и ~~нервы~~ всех участников процесса
3. Повышает мотивацию и ответственность команды
4. Это круто, модно, молодежно. Все компании в вакансиях хотят “Scrum experience” ....

ONIX

Точка отсчета

Обычно внедрять Скрам начинают, когда ещё очень мало людей прошли обучение, ещё меньше людей, действительно понимающих, что такое Скрам, но много тех, которые уверены, что понимают, что нужно делать.

ONIX

SCRUM

People Can Address
Complex Adaptive Problems

THE HIGHEST
POSSIBLE VALUE

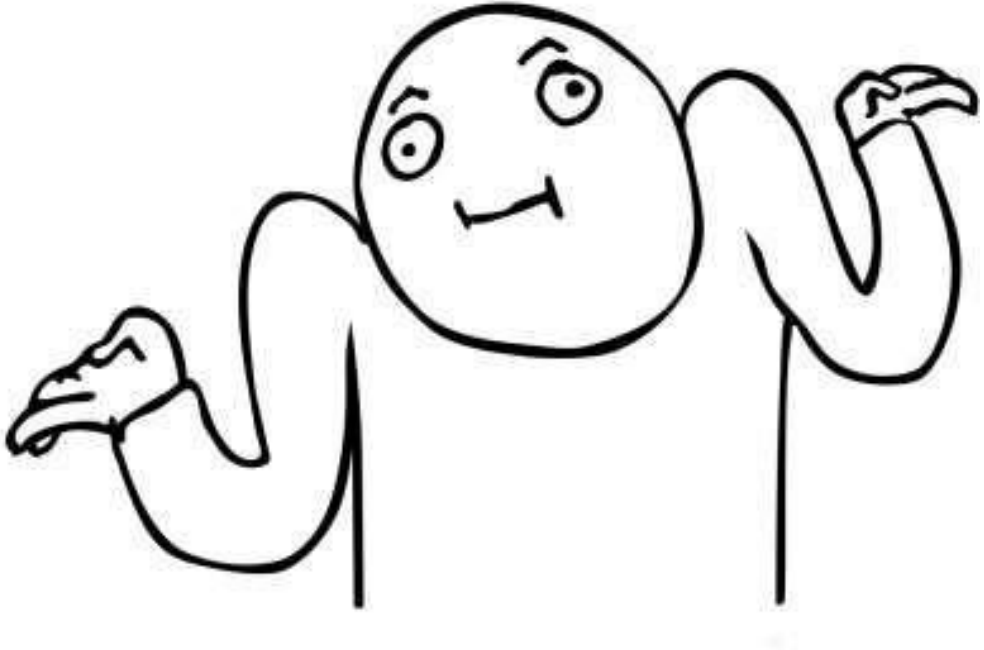
ONIX

История “успеха”

Весело, задорно, будучи классной командой запросто можно напилить кучу... “кода”...

ONIX

Ни панятна… Что не так?



ONIX

Добавим немного магии

В “пустую коробку”
Scrum добавим

IT контекст, практики
и идеи

ONIX

Команда разработки о своем понимании Scrum

Со всеми этими планированиями да стрижками собак (grooming) мне писать код некогда!!!

r

ONIX

Planning - Зачем?

Это возможность для **всей!** команды сесть вместе и понять:

1. Что (какие стори) они смогут точно поставить в течении спринта? От начала и до конца
2. Как именно (через какие задачи) команда придет к этому результату?

ONIX

Planning - Что делать?

1. Подготовить все необходимое для проведения Планирования **ДО** его начала
2. Создать и оценить задачи, с помощью которых команда будет реализовывать цель Спринта. Не забывайте о технических задачах!
3. “Примерить” планы команды к team capacity
4. Сформировать, по итогу, Sprint Backlog

ONIX

Planning - Идеи

Вопросы, которые можно задать ребятам
во время планирования:

1. Все ли зависимости они учли?
2. Раньше с подобными задачами работали? Что тогда пошло не так?
3. После того как сторя будет реализована как поменяется система?
4. Какие могут быть сложность с проверкой данной функциональности?

ONIX

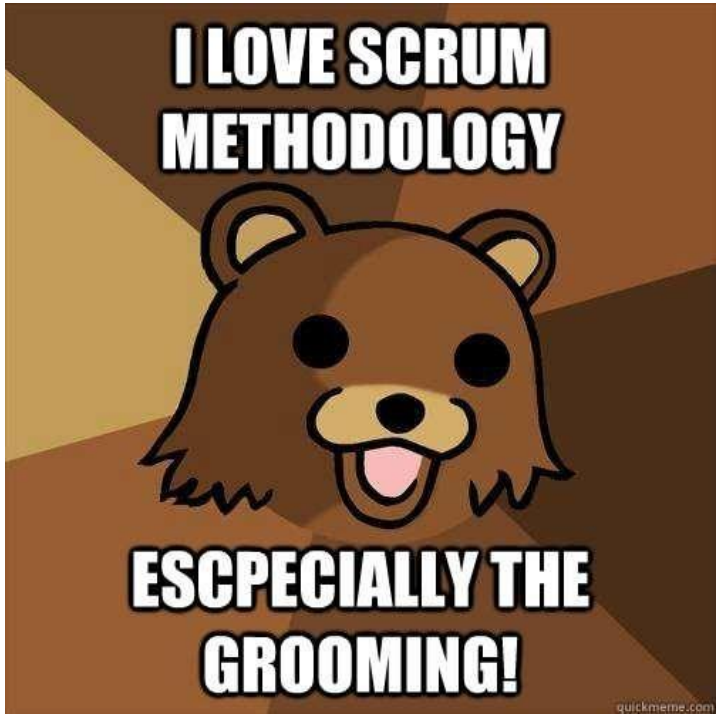
Grooming - Зачем?

aka Product Backlog Refinement (PBR)

В первую очередь это время для того, чтоб найти “идеальный” баланс между потребностями бизнеса и техническими возможностями системы/команды

ONIX

Product Backlog Refinement (PBR) - Что делать?



1. Задать все необходимые уточняющие вопросы касательно требований
2. Дать обратную связь PO, касательно сложности и технической возможности реализации
3. Выделить время на исследования, построение и проверку гипотез
4. Просите о декомпозиции или изменении приоритетов (если нужно)

ONIX

Product Backlog Refinement (PBR) - Идеи

1. Учитывайте зависимости на текущее состояния системы, ее ограничения и возможности
2. Уточняйте, пока можно, какие требования must a какие - nice to have…
3. Не забывайте про обратку негативных сценариев

Помните о том, что цель команды - построить качественный! продукт

ONIX

Команда разработки о своем понимании Scrum

Делал дело, буду делать дело,
проблем нет…

ONIX

Daily - Зачем?

ONIX

Daily - Что делать?

1. Проверить текущее состояние системы. Если билд разломан - первый приоритет починить его
2. Понять какие задачи необходимо решить команде сегодня, чтоб продвигаться к цели
3. Озвучить проблемы/сомнения - все, что может помешать команде достичь “сегодняшней цели”
4. Обновить burn-down chart!

ONIX

Daily - Идеи

1. Организовывайте и фасилитируйте (если надо) follow-up встречи
2. Возникла проблема рассинхрона команды (касательно целей, приоритетов, стандартов написания кода, тестирования и тд) - обсуждайте тут же, не ждите лучших времен (ретроспективы)

ONIX

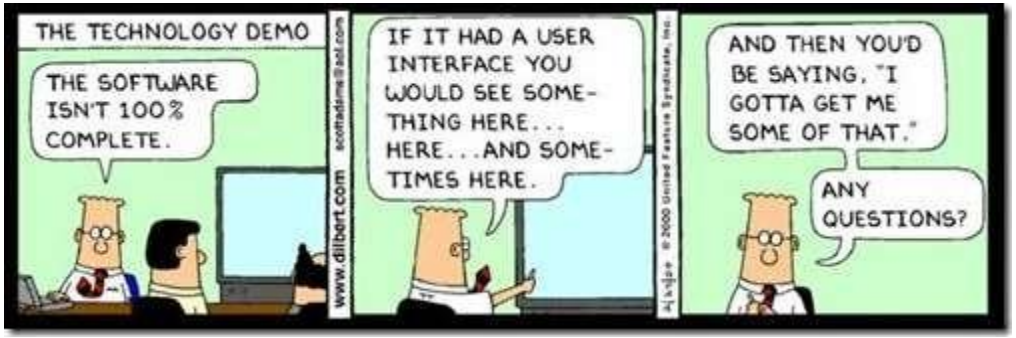
Команда разработки о своем понимании Scrum

про Review

Три минуты позора и расходимся

ONIX

Review - Зачем?



Review: Increment+Current Business Conditions+Product Backlog = Updated Backlog

Чтоб иметь возможность принимать верные технические решения, касательно развития системы - необходимо понимать ее текущее состояние и планы развития

ONIX

Review - Что делать?

1. Презентовать готовый функционал, поясняя кратко какие методы были использованы для его реализации
2. Задавать уточняющие вопросы, с целью получения обратной связи
3. Получить описание будущих планов и их мотивации

ONIX

Review- Идеи

Чтоб получить максимально широкую и четкую обратную связь:

За какое-то время **ДО начала встречи** отправьте Заказчику список реализованных элементов и билд, где можно их посмотреть, “пощупать руками”

ONIX

Команда разработки о своем понимании Scrum

О ретроспективе:

Во! Опять сейчас тошнить начнет: чего хорошо, плохо...

ONIX

Retrospective - Зачем?

Scrum команда проверяет как прошел Sprint с целью:

1. Повысить командую производительность
2. Уменьшить размер технического долга *(all unDone work - technical dept)*
3. Сделать DoD более жестким *(если возможно)*

Если надо было бы определить единственную цель Scrum это было создание DONE инкрементов!

ONIX

Retrospective - Что делать?

Проанализировать:

1. DoD (Definition of Done)
2. Процессы
3. Инструменты, которые использовала команда
4. Коммуникации и отношения в команде

Дополнительно можно использовать:

1. RCA (root cause analysis)
2. Code analysis data

ONIX

Retrospective - Идеи

Пример “жесткого” DoD *(может служить источником для идей или целью)*

1. Performance testing
2. Stability testing
3. Refactoring
4. Release notes
5. User acceptance testing
6. User documentation
7. Regression testing
8. Code reviews

ONIX

Retrospective - Идеи

Technical Debt forms:

1. Lack of automated build or deployment
2. High code complexity
3. Lack of unit or and acceptance tests
4. Highly coupled code
5. High cyclomatic complexity
6. Duplicated code or modules
7. Unreadable names or algorithms

Paying Back Technical Debt:

1. Stop creating debt
2. Make a small payment each Sprint

ONIX

Retrospective - Идеи

Чтоб помочь команде взглянуть на проблему под иным углом можно использовать следующие вопросы:

1. Эти баги были бы если бы мы использовали TDD?
2. Если бы в течении планирования и детализации мы проверили свои гипотезы с помощью Spike - нам бы это помогло?
3. Если бы тестовое покрытие кода было лучше - что нам бы это дало?

ONIX

Continuous process

1. Continuous integration
2. Code refactoring
3. Small releases
4. Sustainable pace

Shared understanding

1. Test-driven development
2. Coding standards
3. Collective code ownership
4. Simple design

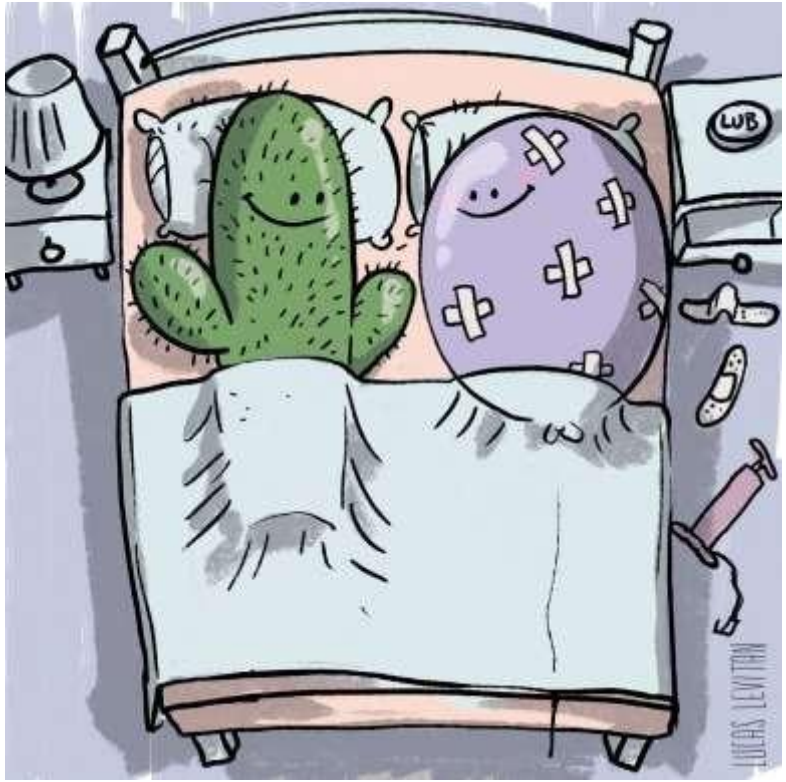
Заключение

Добавление XP должно стать естественным путем для команды, которая начала работать по Scrum и стремиться стать профессиональной Scrum командой. Из моего опыта, команды, которые используют XP - гипер-продуктивны.

© статья Джошуа Партоджи (Joshua Partogi) “Scrum and eXtreme Programming”

ONIX

Take away notes



Здоровые отношения между командой и всем остальным миром - крайне важная цель!

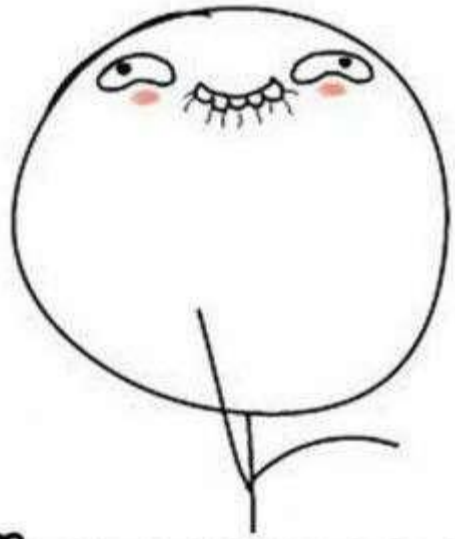
ONIX

Take away notes

Попытаться достичь их можно помня что:

1. Фраза “так это ж понятно по умолчанию” - привела к крушению Mars Orbiter
2. Scrum - пустая коробка, которую необходимо наполнять контекстом
3. Инженерные практики (XP) - хороший кандидат для такого дополнения

ONIX



ОЙ СПАСИБО СПАСИБО
ПРИСПАСИБО)

ONIX