

Лекция №20 Процесс разработки, методологии и инструменты



○ Ознакомиться с методологиями waterfall, Agile/SCRUM

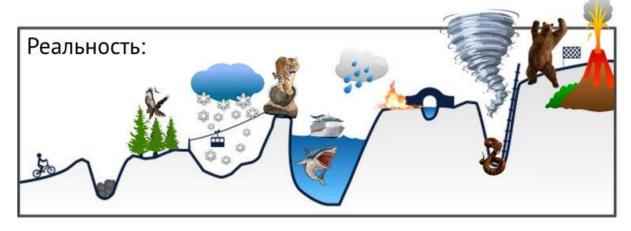
Изучить методологию SCRUM

Ознакомиться с инструментами цикла разработки

# ПРОЦЕСС РАЗРАБОТКИ







# МЕТОДОЛОГИИ/МОДЕЛИ РАЗРАБОТКИ



• Существуют две основных методологии разработки:

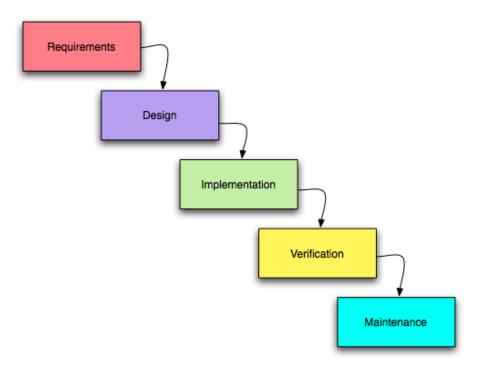
• Водопад (каскадная)

• Agile/SCRUM – гибкая методология разработки

### ВОДОПАД – КАСКАДНАЯ МЕТОДОЛОГИЯ



Модель процесса разработки ПО, за основание берут 1970г. – статью опубликованную У. У. Ройсом Фазы жизненного цикла:



- Определение требований
- Проектирование/дизайн
- Реализация
- Тестирование и отладка (также «верификация»)
- Инсталляция
- Поддержка/обслуживание

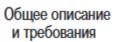
# ВОДОПАД – УНИФИЦИРОВАННЫЙ ПРОЦЕСС (UP)



UP основывается на исследованиях процессов, проводимых в Ericsson (метод Ericsson, 1967), Rational (Rational Objectory Process, 1996–1997) и других ведущих компаниях.

Унифицированный процесс компании Rational (Rational Unified Process, RUP) — это коммерческая версия UP от IBM, которая поглотила Rational Corporation в 2003 году.







Процесс разработки программного обеспечения

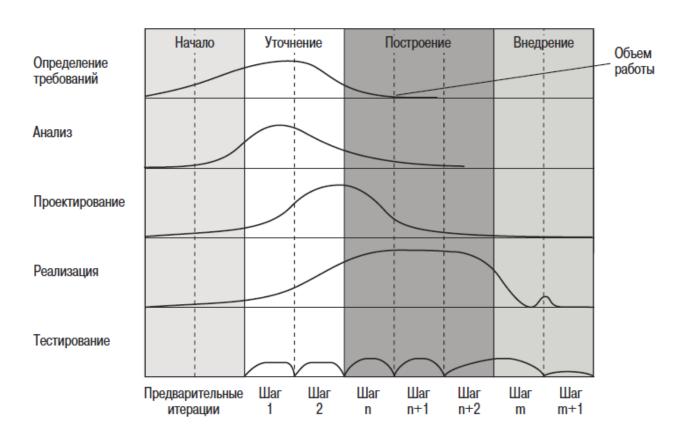


Программное обеспечение

# УНИФИЦИРОВАННЫЙ ПРОЦЕСС (UP/RUP)



# Фазы жизненного цикла и рабочие потоки проекта



### ДИАГРАММЫ ГАНТА



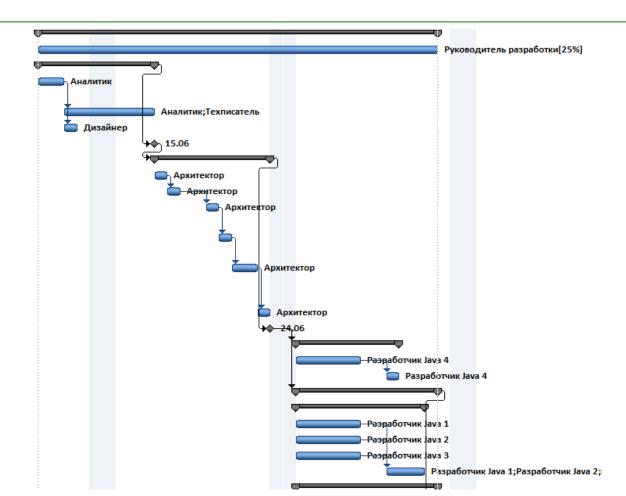
Диаграмма Ганта является одним из методов планирования проектов.

Задачи проекта по водопадной методологии органично отображаются этой диаграммой.

Используется для иллюстрации плана, графика работ по какомулибо проекту

# ДИАГРАММА ГАНТА В MS PROJECT





# ВОДОПАД ПЛЮСЫ



- Детальная документация.
- Согласованные и подписанные требования.
- На проект можно нанять менее профессиональных разработчиков.
- Хорошо выраженные точки входа и выхода для каждой фазы, что позволяет легко оценить прогресс.

# ВОДОПАД МИНУСЫ



- Медленный старт.
- Фиксированные условия, которые трудно изменить.
- Невозможность клиентской оценки ПО до окончания разработки.
- Невозможность смены направления из-за недостатка гибкости.
- На начальном этапе клиент зачастую не может четко сформулировать свои требования.

### AGILE – ГИБКАЯ МЕТОДОЛОГИЯ



Agile-методы делают упор на непосредственное общение лицом к лицу. Большинство agile-команд расположены в одном офисе.

# Основные идеи:

- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

### AGILE ОРГАНИЗАЦИЯ



- Самоорганизующиеся кросс-функциональные команды
- Максимальные полномочия команды для удовлетворения потребности клиента
- Гибкие методы разработки продукта при максимальной автоматизации внедрения
- Гибкость поддерживающих процессов

# SCRUM (CXBATKA) – ГИБКАЯ МЕТОДОЛОГИЯ УПРАВЛЕНИЯ ПРОЕКТАМИ

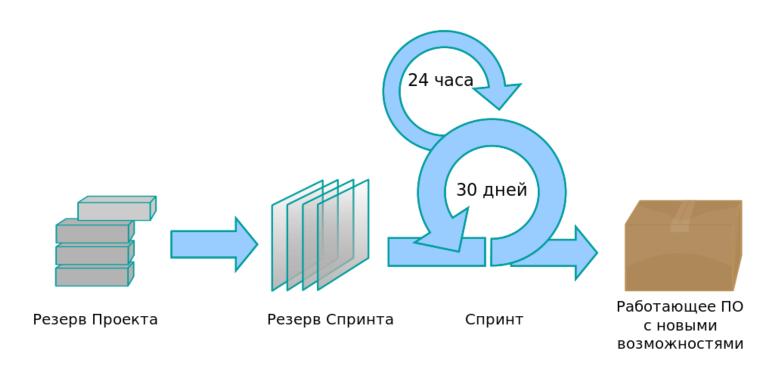


Рождение методологии — статья Хиротака Такэути и Икудзиро Нонака в Гарвардский Деловой Обзор, январь-февраль 1986.

Проекты, над которыми работают небольшие команды из специалистов различного профиля, обычно систематически производят лучшие результаты, и объяснили это как «подход регби»

В настоящее время, Scrum является одной из наиболее популярных «методологий» разработки ПО. Согласно определению, Scrum — это каркас разработки, с использованием которого люди могут решать появляющиеся проблемы, при этом продуктивно и производя продукты высочайшей значимости





# SCRUM – КОМАНДА И РОЛИ



### 3 базовых роли (Свиньи):

- Product owner
- Scrum master
- Команда разработки (Development team)

DT должны обладать следующими качествами и характеристиками:

- самоорганизующейся. Никто (включая SM и PO) не может указывать команде каким преобразовать Product Backlog в работающий продукт
- многофункциональной, обладать всеми необходимыми навыками для выпуска работающего продукта
- За выполняемую работу отвечает вся команда, а не индивидуальные члены команды



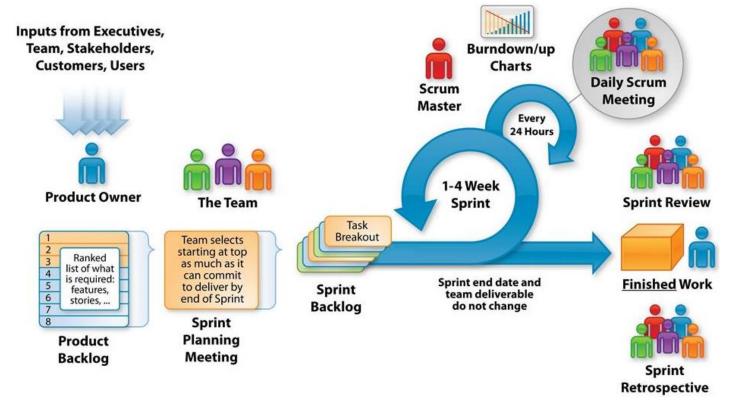
# Дополнительные роли (Куры):

- Пользователи (*Users*)
- Клиенты, Продавцы— лица, которые инициируют проект и для кого проект будет приносить выгоду. Они вовлечены в скрам только во время обзорного совещания по спринту (Sprint Review).
- Управляющие (*Managers*) люди, которые управляют персоналом.
- Эксперты-консультанты (Consulting Experts)

### SCRUM – ЦИКЛЫ И ИТЕРАЦИИ



# Цикл Деминга – PDCA (Plan-Do-Check-Act, Планировать-Действовать-Проверять-Корректировать)



# ДИАГРАММА СГОРАНИЯ ЗАДАЧ (BURNDOWN CHART)





#### SCRUM ПЛЮСЫ



- Ориентирован на клиента
- Адаптивен
- Прост в изучении
- Упор на самоорганизующуюся, многофункциональную команду

#### SCRUM МИНУСЫ



- Жесткие ограничения на работу команды в рамках спринта
- Сложен во внедрении
- Высокие затраты на формирование команды
- Неопределенность. Количество спринтов неограниченно, поэтому сложно поставить конечную дату в проекте.

#### РАЗРАБОТКА ПО ШАГИ



- 1. Выбор решения
- 2. Проработка архитектуры
- 3. Создаем проекты
- 4. Описание интерфейсов
- 5. Пишем код
- 6. Пишем тесты
- 7. Pull Request
- 8. Релиз/Документация
- 9. Установка

#### РАЗРАБОТКА: НАЧАЛО



# Качаем шаблон:

Тесты

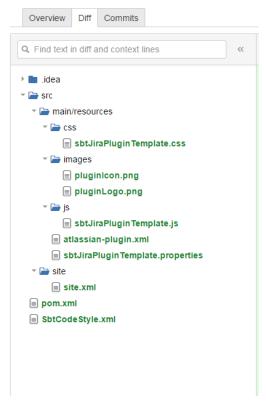
Покрытие

Структура

Maven Site

C

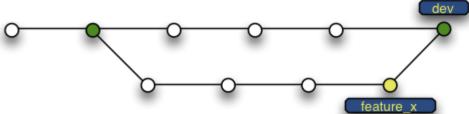
#### [template] add plugin template





# Создаем проект в Stash:

Делаем branch

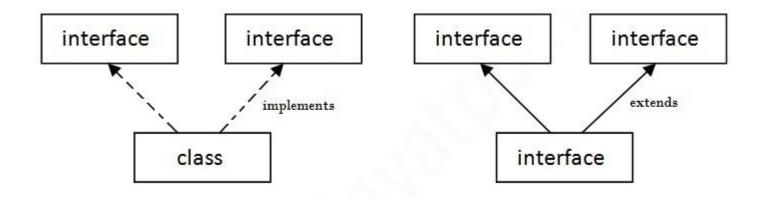






# Проектируем интерфейсы:

- Отвязываемся от реализации
- Пишем параллельно



### РАЗРАБОТКА: ЧТО УЖЕ МОЖНО ДЕЛАТЬ



- Создавать СІ сборку
- Писать back-end
- Писать frontend
- Настраивать Stash
- Создавать проект в Sonar Qube

# CI - CONTINUOUS INTEGRATION







Непрерывная интеграция (англ. Continuous Integration)

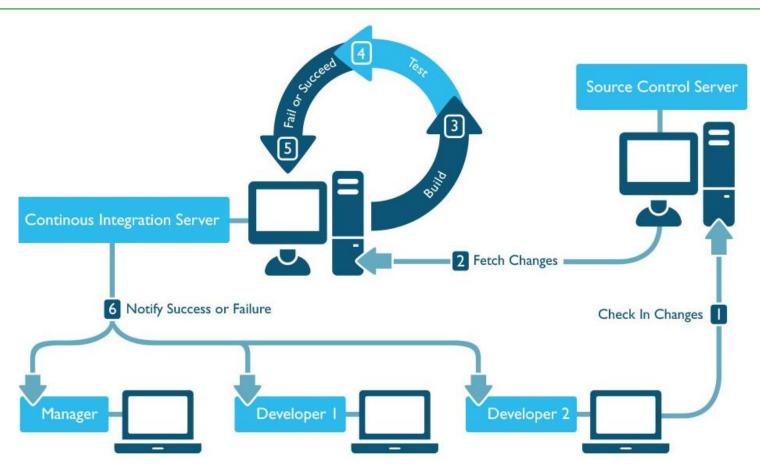
— это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.



CIS – это некая программа, которая следит за вашим Source Control, и при появлении там изменений автоматически стягивает их, билдит, выполняет тесты (конечно, если их пишут) и возможно делает кое-что ещё. В случае же неудачи она дает об этом знать всем заинтересованным лицам, в первую очередь – последнему коммитеру.

# CIS – CI SERVER CXEMA ВЗАИМОДЕЙСТВИЯ





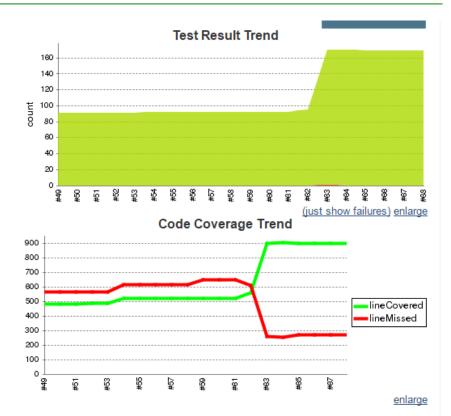


# Jenkins – экосистема для различных CI процессов





- Сборки
- Оповещения
- Интеграция со всем
- Ссылки на ресурсы
- Графики, статистика



#### CIS - JENKINS

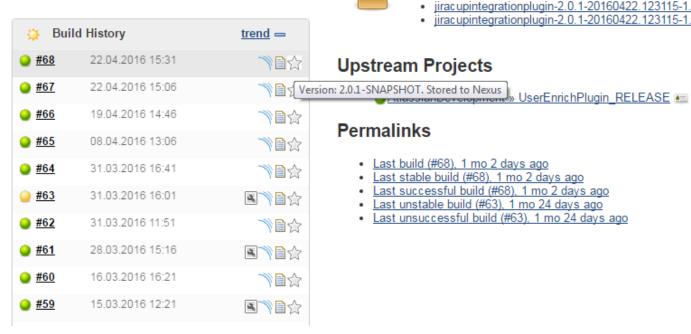
















#### Latest Maven Deployments:

- jirac upintegrationplugin-2.0.1-20160422.123115-1.jar
- jiracupintegrationplugin-2.0.1-20160422.123115-1.pom

#### **Upstream Projects**

#### **Permalinks**

- Last build (#68), 1 mo 2 days ago
- Last stable build (#68), 1 mo 2 days ago
- Last successful build (#68), 1 mo 2 days ago
- Last unstable build (#63), 1 mo 24 days ago
- Last unsuccessful build (#63), 1 mo 24 days ago

#### **ШАГИ РАЗРАБОТКИ: PULL REQUEST**



PR - «запрос на включение (сделанных вами изменений)»
Посылая pull request, вы говорите автору изначального репозитория (и всем заинтересованным лицам): «Смотрите, что я сделал, не хотите ли принять мои изменения и влить их в проект?»

### ШАГИ РАЗРАБОТКИ: PULL REQUEST ПРАВИЛА



# Правила создания PR:

- PR должен быть хорошо оформлен и содержать исчерпывающее описание.
- Обычное правило, один баг один PR, одна фича один PR. Не нужно пытаться впихнуть сразу кучу всего.
- Очень важно соблюдать Code Style того проекта, для которого вы делаете PR. Пусть даже он кажется вам противоестественным (например вы всегда делаете отступы в виде 4 пробелов, а в проекте табы).



# Шаги перед принятием PR:

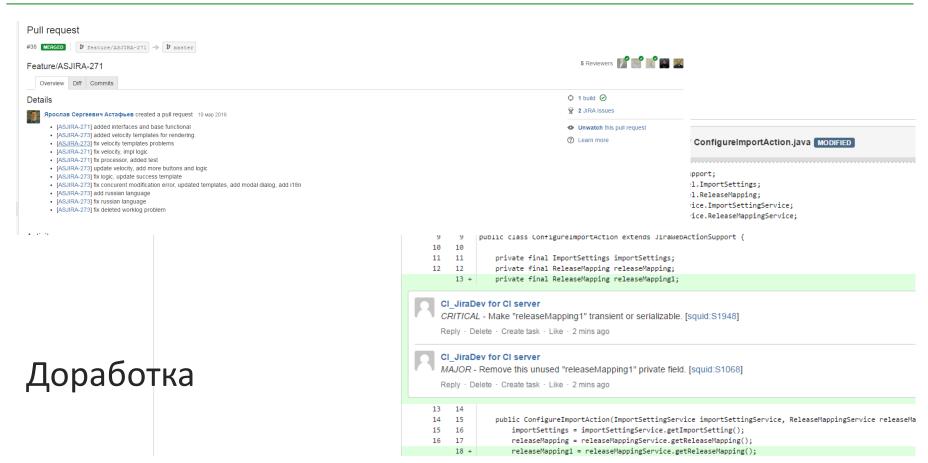
- Код ревью
- Автозапуск тестов
- Sonar Qube
- Code Style





#### **РАЗРАБОТКА: CODE REVIEW**





## **РАЗРАБОТКА: STASH CODE REVIEW**



¢
V
1
đ
₽

	Ярослав Сергеевич Аст	17f72e7262d	[ASJIRA-332] remove unused dependent	31 мар 2016	ASJIRA-332	<b>⊘</b>
	Ярослав Сергеевич Аст	19fe8337755	[ASJIRA-332] increase heap	31 мар 2016	ASJIRA-332	
	Ярослав Сергеевич Аст	d487c6551a3	[ASJIRA-332] increase heap	31 мар 2016	ASJIRA-332	(!)
9	Ярослав Сергеевич Аст	9daf085243d	[ASJIRA-332] remove useless tests	31 мар 2016	ASJIRA-332	(!)
	Ярослав Сергеевич Аст	92a8c52896a	[ASJIRA-332] remove useless test	31 мар 2016	ASJIRA-332	
	Ярослав Сергеевич Аст	291bb3dd058	[ASJIRA-332] fix PR bugs	31 мар 2016	ASJIRA-332	<b>②</b>
9	Вадим Андреевич Гаузяк	449daebf896 M	Merge pull request #40 in AD/sbt-jira-cup	31 мар 2016	ASJIRA-328	
•	Вадим Андреевич Гаузяк	9b37b0508aa	[ASJIRA-328] update test	30 мар 2016	ASJIRA-328	<b>⊘</b>
•	Вадим Андреевич Гаузяк	e0826465312	[ASJIRA-328] fix build	30 мар 2016	ASJIRA-328	<b>②</b>
•	Вадим Андреевич Гаузяк	f0abde717e3	[ASJIRA-328] added test	30 мар 2016	ASJIRA-328	(!)

## РАЗРАБОТКА: PULL REQUEST MERGE



- Resolved merge Conflicts
- Успешная сборка
- 2 approve
- Покрытие тестами > 70%



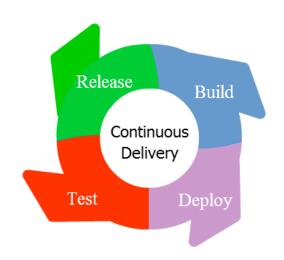


CD – непрерывная поставка, как естественное продолжение (расширение) практики Cl

Обеспечивает автоматизированную поставку продукта с учетом всех функциональных изменений и исправлений ошибок конечному пользователю, быстро и безопасно на устойчивой основе.

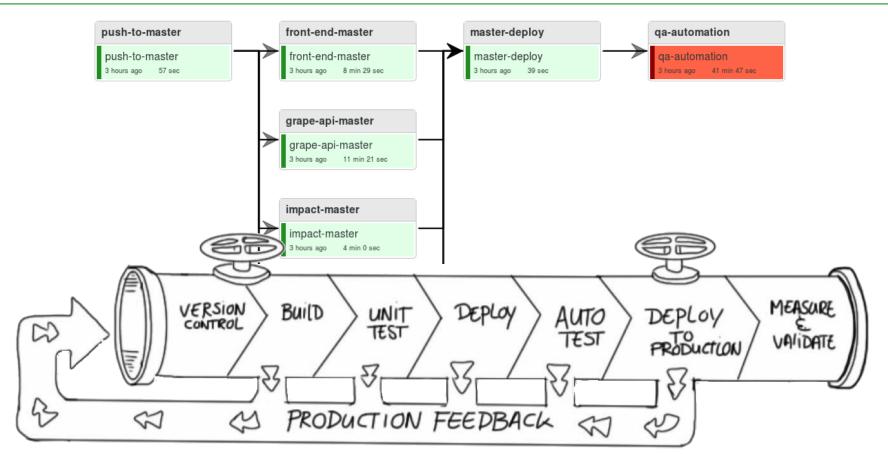


- 1. Release Notes
- 2. Test Results Publish
- 3. Code Analyses Publish
- 4. Deploy
- 5. Configuration management
- 6. Test& Publish
- 7. Delivery Pipeline
- 8. Supervisor integration



### **CD: PIPELINES**

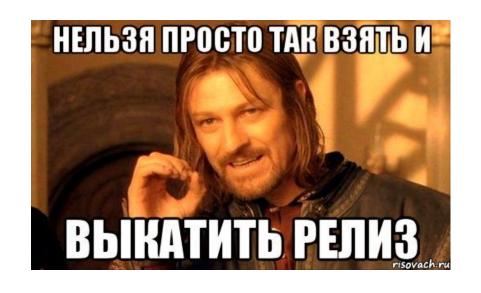




### **РАЗРАБОТКА: BUILD & STORE**

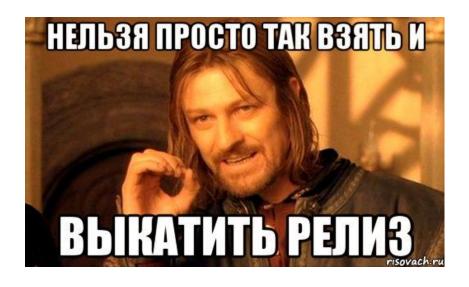


- Создаем RC
- Формируем релиз
- Ставим на dev
- Тестируем
- Дорабатываем
- Выпускаем релиз



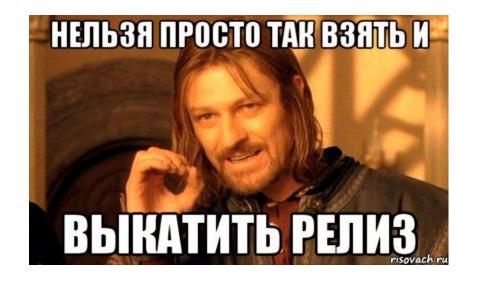


• Обновляем документацию



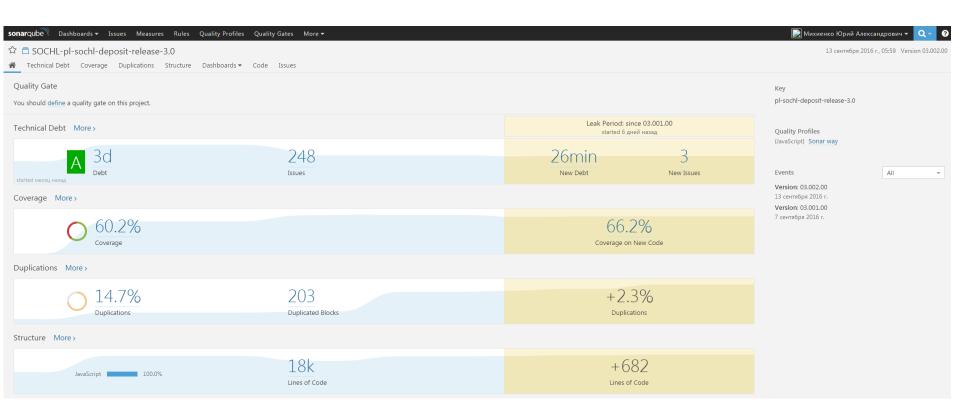


- Деплой на прод
- Эксплуатация
- profit



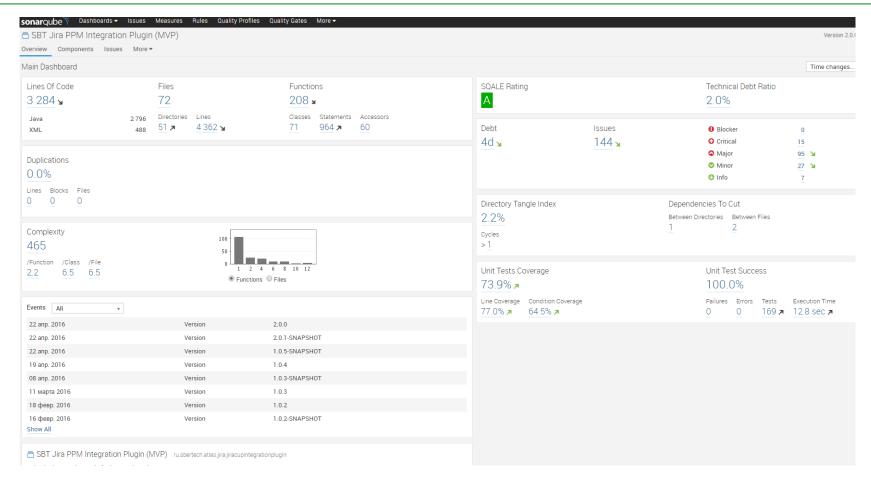
# SONAR: АВТОМАТИЗИРОВАННЫЙ КОД-РЕВЬЮ



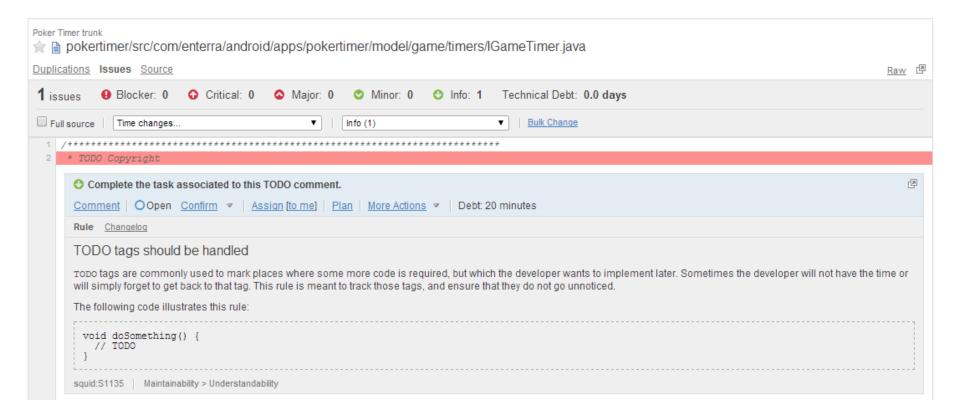


# SONAR: АВТОМАТИЗИРОВАННЫЙ КОД-РЕВЬЮ











- Статический анализ
- «Нотификации Stash»
- Быстрые ссылки на проблему
- Циклические зависимости
- Машина времени
- Profiles

### **ЛИТЕРАТУРА**



- Джефф Сазерленд SCRUM
- Джим Арлоу Айла Нейштадт UML2 и унифицированный процесс
- https://jenkins.io/

# ДОМАШНЕЕ ЗАДАНИЕ



- Установить SONAR
- Снять статистику со своего проекта, сформировать скриншоты с пояснениями