Java School #18

Opening

Saint Petersburg, 2016

T-Systems

Правила

- Все официальные публикации происходят в группе VK:
  https://vk.com/t_school
- Изменения в расписании публикуются там же
- Ноутбук нужен на всех лекциях *
- И он открывается только на время практики
- Сами лекции тут: https://bitbucket.org/tschool/javaschool
- Если что-то непонятно, то лучше спросить

- В любых непонятных случаях можно писать :
  Daniil.Shulgin@t-systems.ru или Andrey.Bulov@t-systems.ru

T · · Systems

Расписание

Лекция	Лектор	Дата	Аудитория	Время
Developer Tools, дизайн приложения	Шульгин/Булов	15.08.2016	0.3	16:00
DB/DB patterns/JDBC	Шульгин/Матвеев	18.08.2016	0.3	16:00
JPA/Hibernate	Круглов/Иванов	22.08.2016	0.3	16:00
Основы Web, Servlets	Брагин/Кудинов	25.08.2016	0.3	16:00
HTML	Губарев/Макаревич	29.08.2016	0.3	16:00
Javascript	Губарев/Кувшинов	01.09.2016	0.3	16:00
JSP/JSTL	Матвеев/Кудинов	05.09.2016	0.3	16:00
Test Frameworks, Exceptions, Logging	Матвеев/Макаревич	08.09.2016	0.3	16:00
Multithreading/Concurrency*	Дядыч/Есипов	12.09.2016	0.3	16:00
Показ работ	Все	**15.09.2016**	0.3	16:00
Enterprise Stack Review	Булов/Губарев	20.09.2016	0.3	16:00
Architecture*	Лукин/Губарев	22.09.2016	0.3	16:00
Spring Framework	Двинянин/Кузнецов	27.09.2016	0.3	16:00
EJB 3.x	Булов	29.09.2016	0.3	16:00
JSF 2	Шульгин/Булов	04.10.2016	0.3	16:00
Webservices	Урих/Никифорова	06.10.2016	0.3	16:00
Software Development Processes	Строкан	11.10.2016	0.3	16:00
Testing*	Варгин	13.10.2016	0.3	16:00
Показ работ	Все	**18.10.2016**	0.3	16:00

Кураторы

Студент	Куратор
Лебедев Сергей	Двинянин Андрей
Авдеев Игорь	Матвеев Владислав
Антипов Владимир	Журавлев Дмитрий
Платэ Алексей	Дядыч Павел
Медведев Илья	Урих Герман
Демьянов Дмитрий	Есипов Александр
Буевич Виктор	Кузнецов Кирилл
Книзе Кирилл	Губарев Илья
Карнов Артем	Шульгин Даниил
Алиев Мирза	Круглов Вячеслав
Ульяничев Кирилл	Булов Андрей

T · · Systems

Правила работы с куратором

- Куратор будет уделять вам от двух часов в неделю
- Если вы зависли на 4 часа и не знаете как решить проблему, это хороший повод написать куратору
- Куратор будет проверять ваш прогресс на еженедельной основе
- Вы должны получить допуск к показу у куратора

Сквозное задание

- Обязательные критерии успешности:
  - Работающее приложение в котором реализован **весь** функционал
  - Код документирован (javadoc)
  - Написаны unit-тесты
  - Technical solution description
  - Нет major'ов в checkstyle
- Плюсом будет:
  - Красивый UI
  - Selenium Auto Tests
  - Killer features

Вопросы?

T · · Systems

Java Lecture #1

Developer tools

T-Systems

Saint Petersburg, 2016

Agenda

- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

IDE

- Integrated Development Environment

- Попытка совместить весь необходимый инструментарий в одном приложении
- Расширяема за счет плагинов
- Потребляет очень много ресурсов

- Популярные IDE для Java-разработки
  - NetBeans
  - Eclipse
  - Intellij Idea

NetBeans

- IDE с открытым исходным кодом, первоначально разрабатывалась в Sun

- Достоинства
  - Отличный встроенный профайлер
  - Модульная структура
  - Подробная wiki: http://wiki.netbeans.org

- Недостатки
  - Огромное количество мастеров и помощников, скрывающих реально происходящие вещи
  - Медленная работа с remote-проектами
  - GUI-дизайнер Matisse
  - Open source – багфиксы происходят «по желанию»

Eclipse

- Бесплатная IDE с открытым исходным кодом
- Достоинства
  - Возможность хранения Workspace – независимой от проекта конфигурации IDE
  - Огромная коллекция плагинов для самых разных технологий
  - Специализированные сборки IDE: STS, JBoss Tools
- Недостатки
  - На каждый “чих” нужен плагин
  - Средства рефакторинга
  - Часто не видит изменений, сделанных не в IDE
  - Документация оставляет желать лучшего

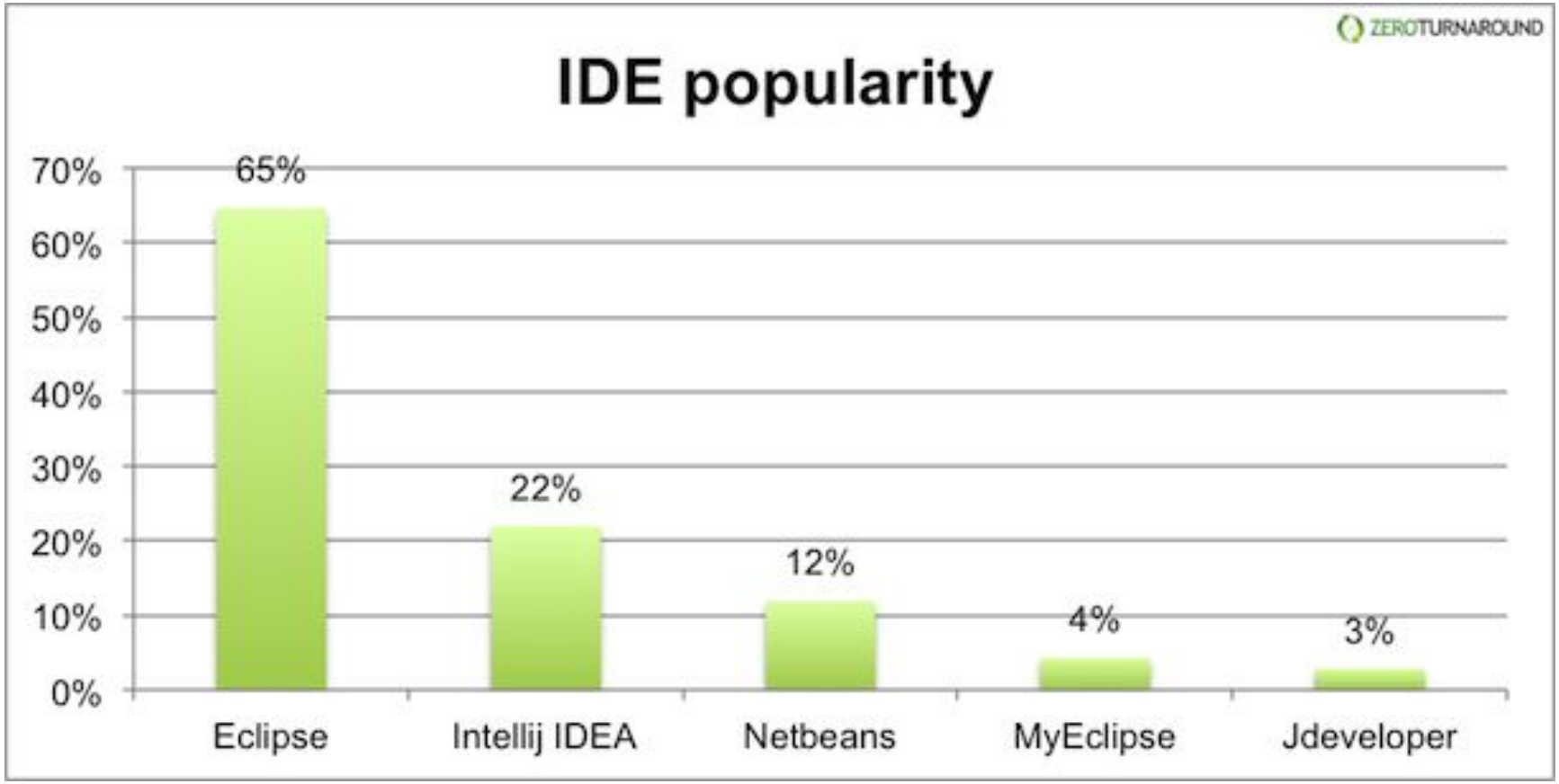
Intellij Idea

- Community - версия бесплатна, нет поддержки EE/Web-разработки
- Ultimate - платная

- Достоинства
  - Отличные инструменты для рефакторинга из коробки
  - Сборки для других языков на том же ядре (PyCharm, RubyMine)
  - Хорошо умеет менять структуру проекта без нарушения его целостности
  - «The most intelligent Java IDE»
- Недостатки
  - Введение модульной архитектуры не пошло на пользу стабильности
  - Многие «неофициальные» плагины просто неработоспособны
  - Документации по плагинам практически нет

Популярность IDE

2011



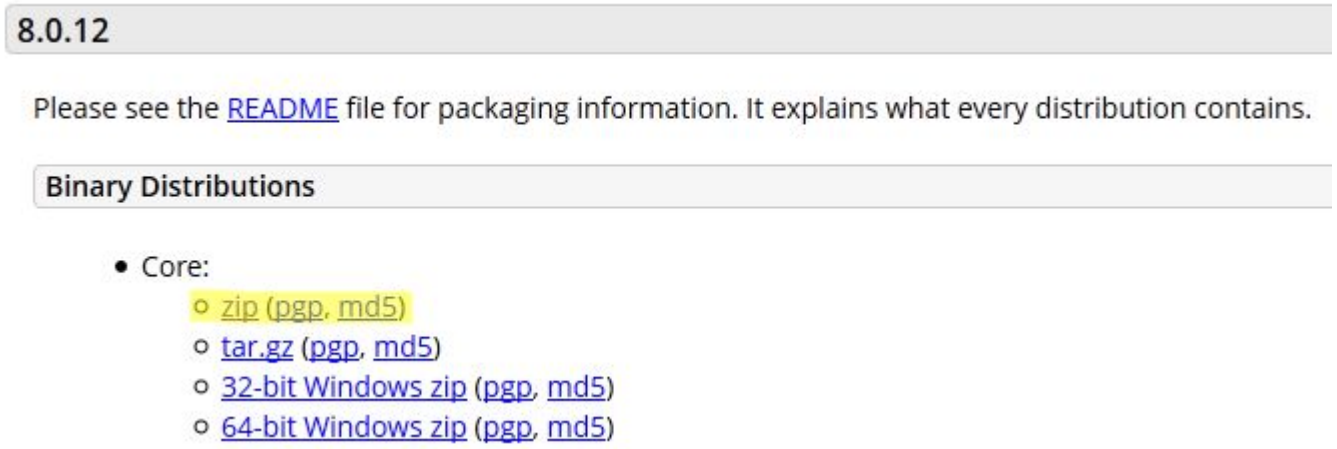
http://zeroturnaround.com/rebellabs/java-ee-productivity-report-2011/#ides

T · · Systems

Practice #1 – создание проекта

- Скачать и установить JDK 8 (если еще нет)
- Скачать и установить **Eclipse IDE for Java EE Developers**

- Скачать Tomcat 8 и распаковать его



Agenda

- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

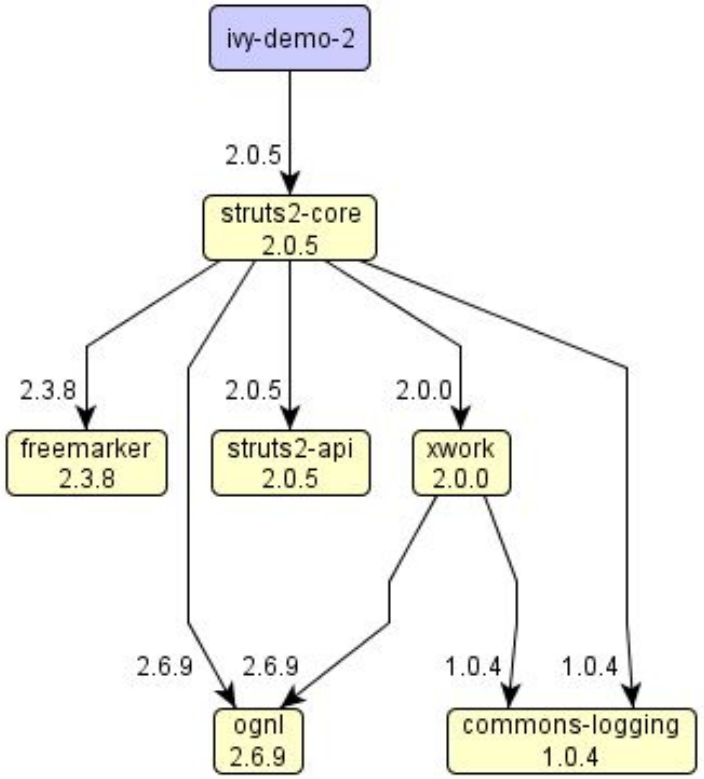
Build automation

- Процесс сборки и развертывания сложных Java-приложений может быть весьма нетривиален

- Что включает в себя build automation:
  - Управление зависимостями
  - Версионирование
  - Компиляция и сборка
  - Выполнение тестов и сбор метрик
  - Генерация кода и конфигурации
  - Деплоймент
  - Публикация

Управление зависимостями

- Под зависимостями здесь понимаются сторонние библиотеки, используемые кодом
- Основные проблемы:
  - Управление транзитивными зависимостями
  - Присутствие всех необходимых зависимостей на разных фазах жизненного цикла
  - SNAPSHOT-зависимости
  - JAR Hell



Apache Ant

- Инструмент автоматизации билд-процесса
- Позволяет релизовать практически любую схему сборки приложения
- Не содержит собственных моделей жизненного цикла
- Позволяет писать сборочные скрипты в своей xml-конфигурации
- Как следствие, `build.xml` часто распухает до огромных размеров
- Один из самых быстрых сборочных инструментов
- Поддерживается всеми популярными IDE
- Используется в NetBeans в качестве внутренней билд-системы

Apache Ant

- Пример билд-скрипта для ant:

```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="clobber" depends="clean" description="remove all artifact files">
        <delete file="hello.jar"/>
    </target>
    <target name="compile" description="compile the Java source code to class files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file for the application">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Apache Maven

- Наиболее популярная на сегодняшний день build-система, стандарт de-facto
- Использует декларативную конфигурацию
- Предоставляет стандартную модель жизненного цикла
- Великолепно управляет зависимостями
- Отлично интегрирован со всем, с чем можно. И с чем нельзя тоже.
  - IDE
  - Системы контроля версий
  - CI-tools, Sonar
- Часто критикуется за недостаточную гибкость
- https://community.jboss.org/wiki/MavenProblems

maven

T · · Systems

Maven: фазы жизненного цикла

- **Validate**

  Проверка POM-файлов на предмет валидности и непротиворечивости модели
- **Compile**

  Компиляция исходного кода
- **Test**

  Выполнение unit-тестов
- **Package**

  Сборка и упаковка в архивы, jar, war, ear, etc
- **Integration-test**

  Выполнение интеграционных тестов
- **Verify**

  Проверка собранных архивов
- **Install**

  Загрузка артефактов в локальный репозиторий
- **Deploy**

  Публикация артефактов в удаленный репозиторий и деплоймент

Maven: управление зависимостями

- Зависимости проекта необходимо декларировать явным образом
- Зависеть можно как от сторонних библиотек, так и от других модулей текущего проекта
- Для каждой зависимости указывается как минимум groupId, artifactId и version
- Транзитивные зависимости подключаются по умолчанию
- Чтобы запретить подключенние транзитивных зависимостей используется тэг <exclude>
- Scope показывает, на каких фазах жизненного цикла maven будет добавлять зависимость в classpath

```
<project>
    ...
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>3.0.5.RELEASE</version>
            <scope>compile</scope>
            <optional>true</optional>
        </dependency>
    </dependencies>
</project>
```

Maven: плагины

- Все, что делает Maven, выполняется тем или иным плагином
- Стандартная модель уже включает в себя несколько плагинов
- Их можно конфигурировать и добавлять свои

- Плагины выкачиваются из репозиториев, как и зависимости
- Существует maven-antrun-plugin, который позволяет выполнять Ant-таски из Maven-билда
- Можно подключать дополнительные репозитории для плагинов
- Или даже писать свои плагины, если не хватает существующих

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-eclipse-plugin</artifactId>
    <version>2.6</version>
    <configuration>
        <downloadSources>true</downloadSources>
    </configuration>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.3.2</version>
    <configuration>
        <source>1.6</source>
        <target>1.6</target>
        <encoding>UTF-8</encoding>
    </configuration>
</plugin>
<plugin>
    <groupId>org.codehaus.sonar</groupId>
    <artifactId>sonar-maven3-plugin</artifactId>
    <version>2.9</version>
</plugin>
```

Maven: архетипы

- Архетип – шаблон maven-проекта под определенные технологии
- Они хранятся в репозиториях maven, как и все остальное
- Позволяет сгенерировать проект на пустом месте командой

```
mvn archetype:generate
```

- Будет создана необходимая структура директорий, сгенерирован pom.xml
- Авторы фреймворков и библиотек часто публикуют архетипы к собственным технологиям в репозитории maven
- В центральном репозитории их более 500
- Создать проект из архетипа может и IDE

Gradle

- Релиз 1.0 вышел 12 июня 2012
- Позиционируется как замена Maven и Ant
  - Предоставляет стандартную модель жизненного цикла
  - Дает возможность её кастомизировать

- Пишется на Groovy DSL, что дает гораздо более компактную и читаемую конфигурацию по сравнению с XML
- Последние версии IDE уже поддерживают Gradle
- С недавнего времени есть плагины для интеграции с Sonar, Jenkins, etc.
- GitHub также поддерживает Gradle
- Spring и Hibernate собираются при помощи Gradle

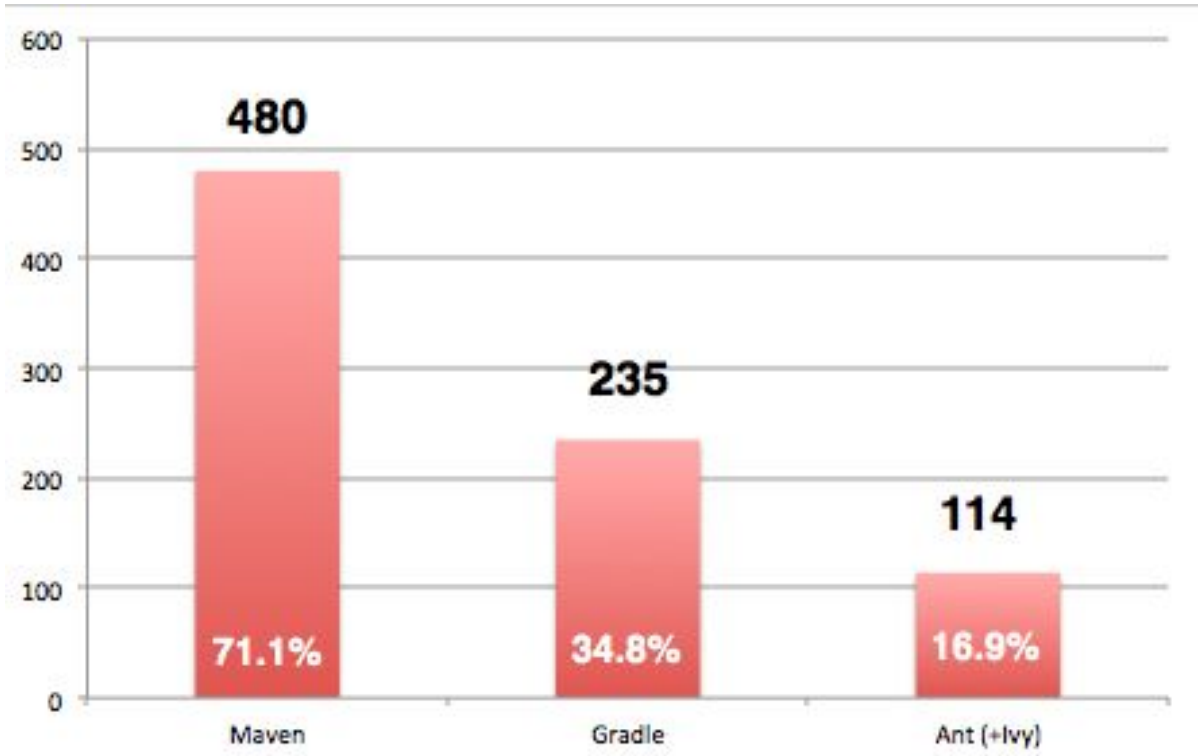
Gradle: Пример

```
apply plugin: 'java'
apply plugin: 'eclipse'

sourceCompatibility = 1.5
version = '1.0'
jar {
    manifest {
        attributes 'Implementation-Title': 'Gradle Quickstart',
                   'Implementation-Version': version
    }
}

repositories {
    mavenCentral()
}

dependencies {
    compile group: 'commons-collections', name: 'commons-collections', version: '3.2'
    testCompile group: 'junit', name: 'junit', version: '4.+'
}

test {
    systemProperties 'property': 'value'
}

uploadArchives {
    repositories {
       flatDir {
           dirs 'repos'
       }
    }
}
```

http://www.gradle.org/docs/current/userguide/userguide_single.html#tutorial_java_projects

Популярность build-систем

2013



http://java.dzone.com/articles/java-build-tools-survey-0

T · · Systems

Practice #2 – создание проекта

- Скачать и установить Maven*
- Создать Maven web проект из архетипа используя командную строку*
- Собрать проект из командной строки

```
mvn clean install
```

- Импортировать проект в Eclipse

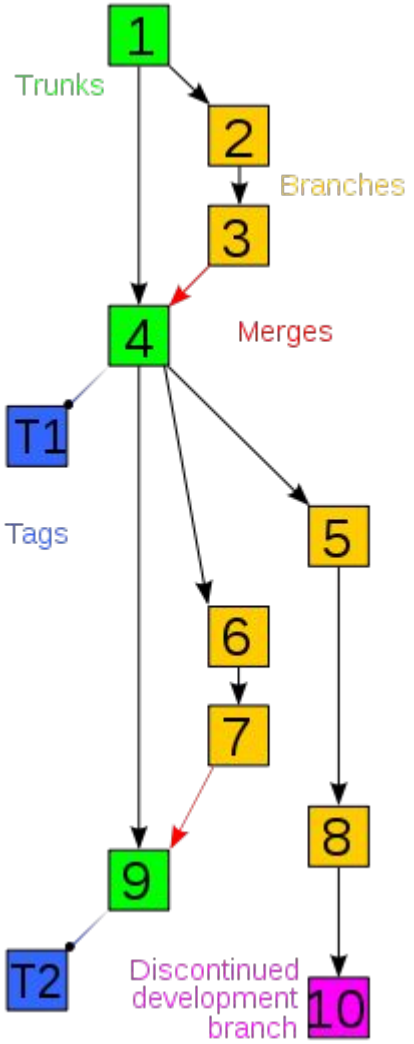
* Хороший мануал - http://habrahabr.ru/post/77382/

Agenda

- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

Системы контроля версий (VCS/SCM)

- Предназначены для командной работы над одним набором файлов исходного кода
- Нумеруют изменения кода, выстраивая последовательную цепочку состояний (ревизий)
- Хранят историю изменений по файлам и папкам
- Позволяют разработчикам эффективно обмениваться изменениями исходного кода



Системы контроля версий (VCS/SCM)

- Могут делать слияние конкурирующих изменений
- Большинство алгоритмов слияния плохо обрабатывает бинарные файлы
- Могут выполнять откат изменений до указанной ревизии
- Как правило позволяют работать с несколькими ветвями разработки и переключаться между ними

Системы контроля версий: глоссарий (1/2)

- Branch. Направление разработки, независимое от других. Ветвь представляет собой копию части хранилища, в которую можно вносить свои изменения, не влияющие на другие ветви. Документы в разных ветвях имеют одинаковую историю до ветвления и разные — после.
- Check-in, commit. Создание новой версии, фиксация изменений. Распространение изменений, сделанных в рабочей копии, на хранилище документов.
- Check-out, clone. Извлечение документа из хранилища и создание рабочей копии.
- Conflict. Конфликт — ситуация, когда несколько пользователей сделали изменения одного и того же участка документа.

Системы контроля версий: глоссарий (2/2)

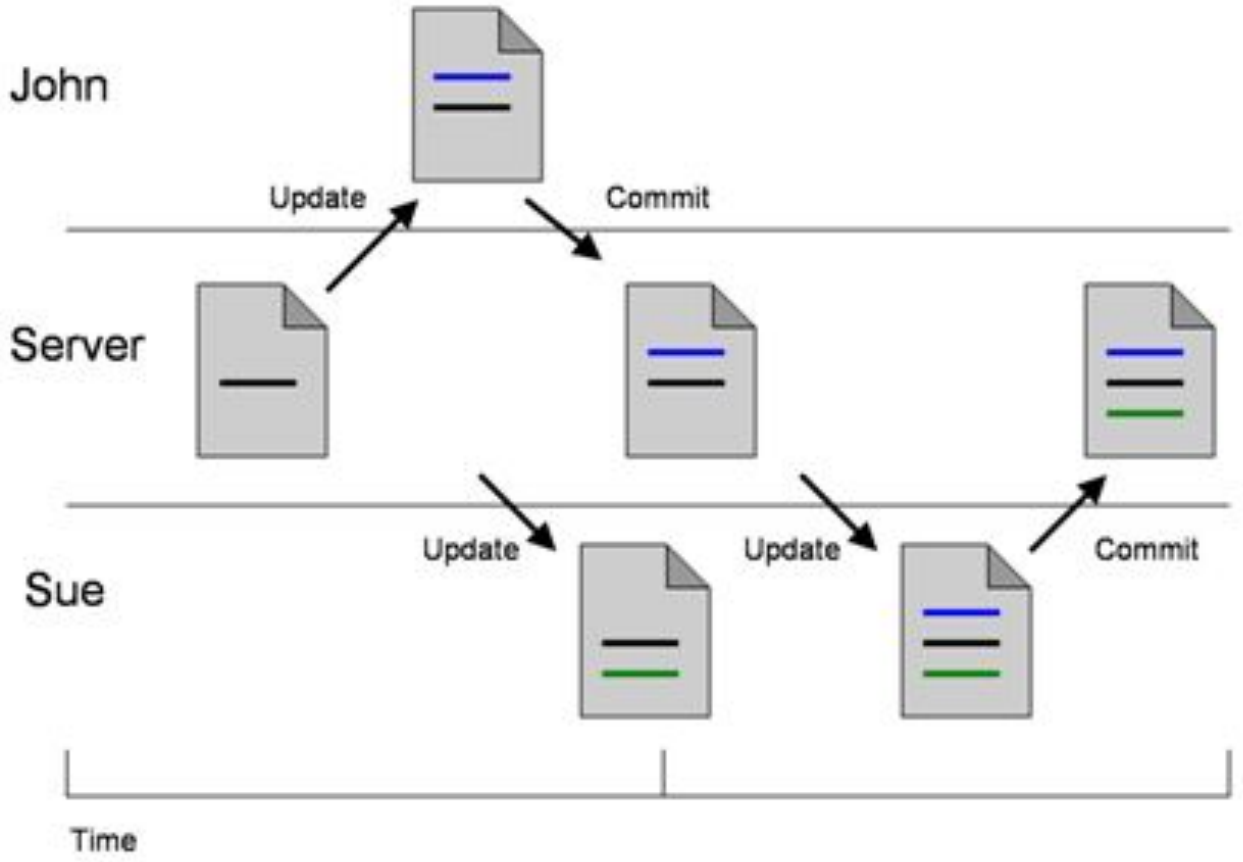
- Head. Основная версия — самая свежая версия для ветви/ствола, находящаяся в хранилище. Сколько ветвей, столько основных версий.
- Merge, integration. Слияние — объединение независимых изменений в единую версию документа. Осуществляется, когда два человека изменили один и тот же файл или при переносе изменений из одной ветки в другую.
- Repository. Хранилище документов — место, где система управления версиями хранит все документы вместе с историей их изменения и другой служебной информацией.
- Revision. Версия документа. Системы управления версиями различают версии по номерам или хэшам, которые назначаются автоматически.

Системы контроля версий: рабочий цикл

- Создание рабочей копии. Операция выполняется однократно.
- Обновление рабочей копии. Операцию обновления (update) рабочей копии необходимо выполнять регулярно.
- Модификация проекта. Работа производится локально и не требует обращений к серверу VCS.
- Фиксация изменений. По завершению очередного этапа работ, разработчик фиксирует (commit) свои изменения, передавая их на сервер.

Системы контроля версий: рабочий цикл

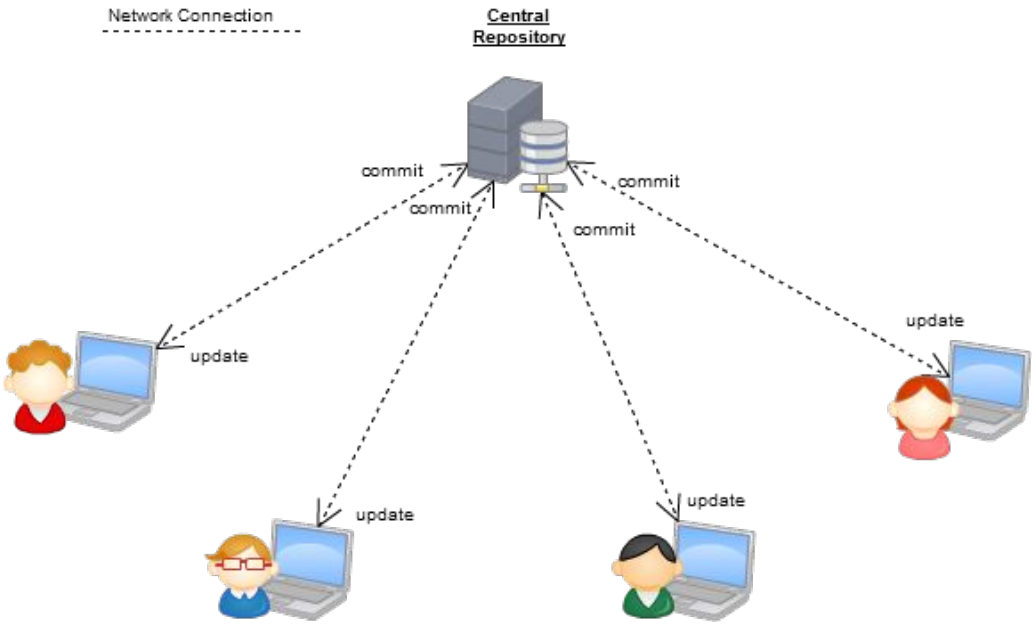
- Процесс работы в команде двух разработчиков:



Централизованные системы контроля версий

- Весь обмен изменениями происходит через центральный репозиторий (сервер)
- Позволяют вести сквозную последовательную нумерацию ревизий
- Хорошо работают для проектов с жесткой вертикалью управления

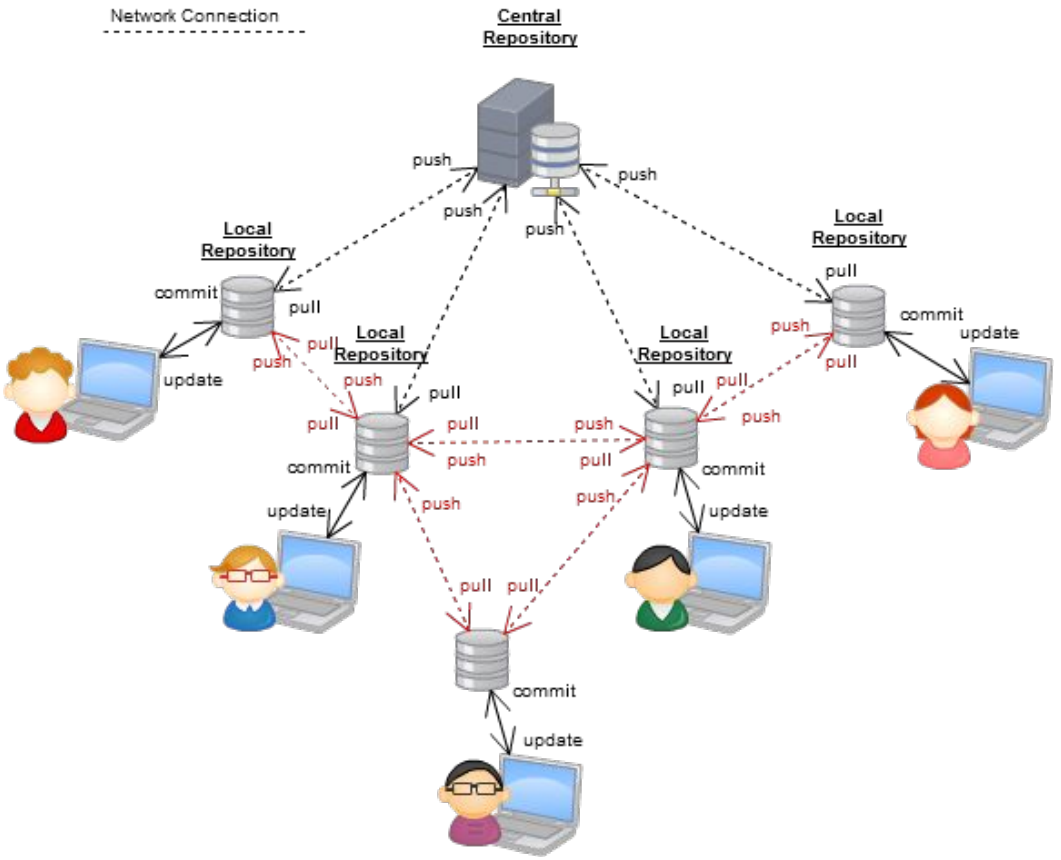
- Примеры
  - CVS
  - SVN
  - ClearCase
  - Perforce



Распределенные системы контроля версий

- Позволяют делать частичную интеграцию изменений непосредственно от автора или коллег без участия центрального репозитория (которого вообще может не быть)

- Хорошо работают для децентрализованных по управлению либо сильно разветвленных (fork) проектов
- Примеры
  - Git
  - Mercurial
  - Bazaar



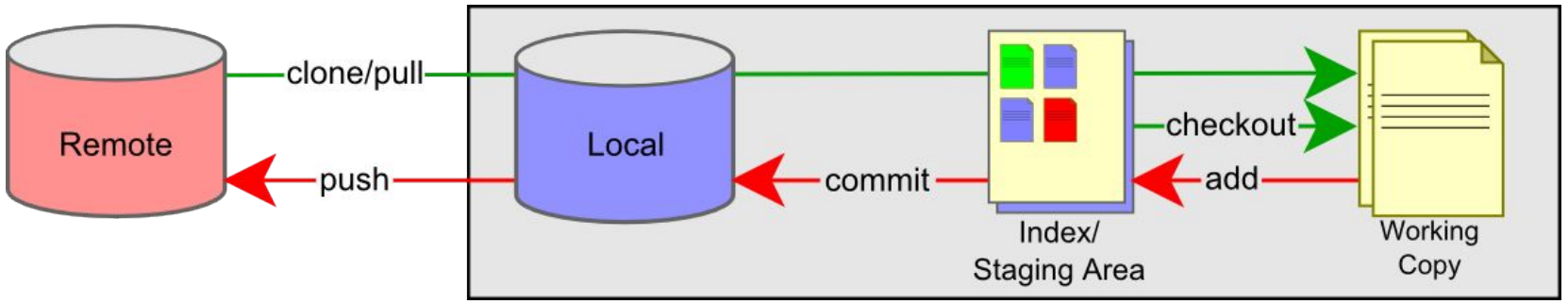
T · · Systems

Subversion (SVN)

- Централизованная система контроля версий, свободная даже для коммерческого использования
- Создавалась для преодоления недостатков CVS
- Много лет была(и есть) de-facto стандартом для разработчиков
- Преимущества:
  - Легко и быстро осваивается
  - Хорошая интеграция с самым разным софтом
- Недостатки:
  - Слияние веток может быть очень трудоемким
  - Однажды добавленную в репозиторий информацию удалить оттуда уже нельзя
  - Нет возможности ассоциировать одну рабочую копию с несколькими репозиториями

Git

- Распределенная система контроля версий с открытым исходным кодом
- Первоначально создавалась для ядра Linux
- Преимущества:
  - Легкость работаты с ветками
  - Поддержка любого количества удаленных репозиториев
  - Впечатляющая производительность
  - Github
- Недостатки:
  - Высокий порог вхождения
  - Слабая поддержка многомодульных проектов



Practice #3 – создание своего репозитория

- Зарегистрироваться на https://github.com/
- Создать новый репозиторий.
- Загрузить проект в репозиторий на Github используя Eclipse.
- Отправить своему куратору письмо со ссылкой на репозиторий.

Agenda

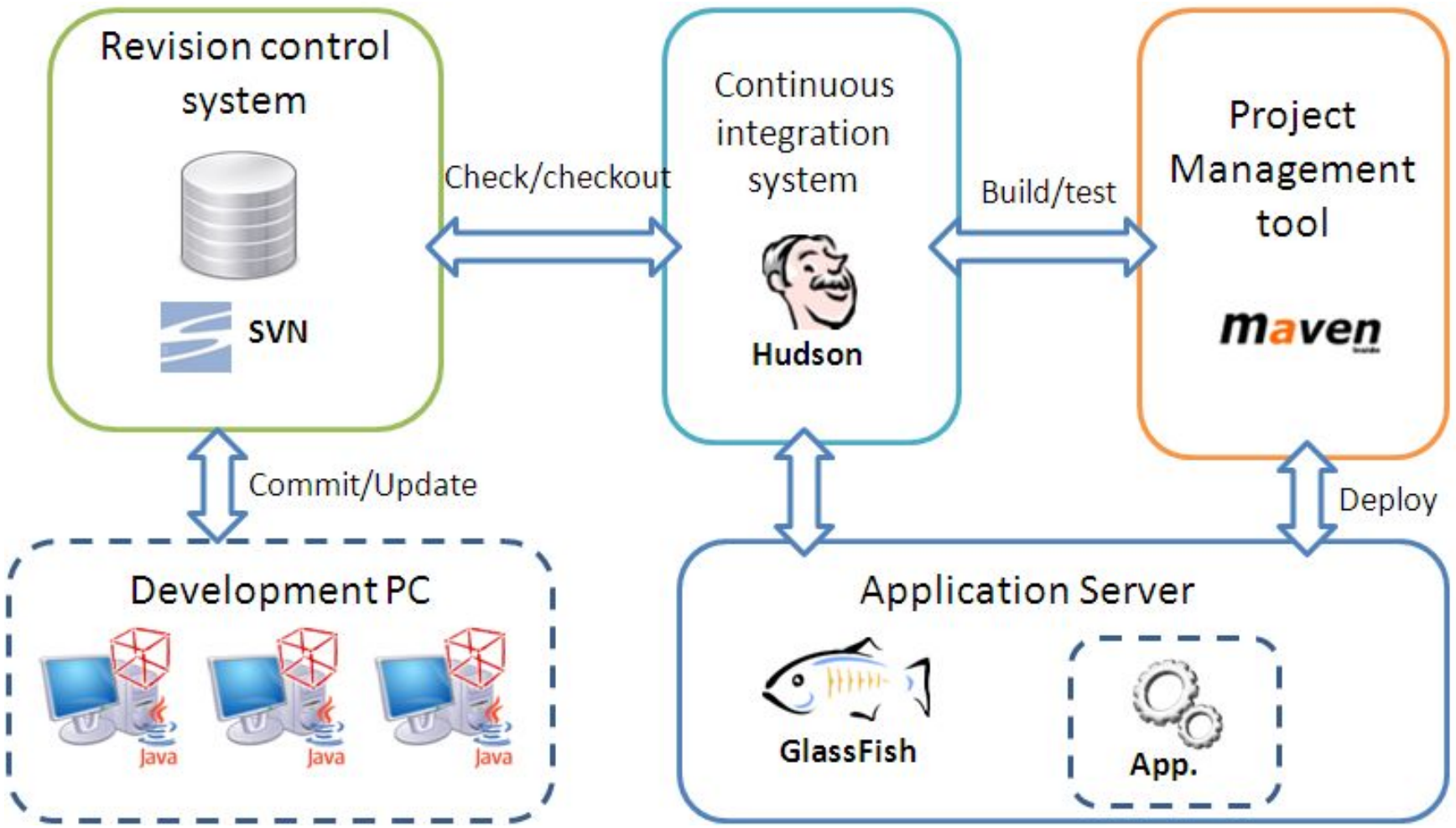
- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

Continuous Integration (CI)

- Для автоматизации интеграционных процессов применяются системы непрерывной интеграции (Continious Integration System, CIS)
- Принцип действия таких систем состоит в следующем:
  - CIS производит мониторинг системы контроля версий
  - При изменении исходных кодов в репозитории производится обновление локального хранилища
  - Выполняются необходимые проверки и модульные тесты
  - Исходные коды компилируются в готовые выполняемые модули
  - Выполняются тесты интеграционного уровня
  - Генерируется отчет о тестировании
  - В случае ошибок на предыдущих фазах билд считается неуспешным и всем заинтересованным лицам рассылаются уведомления на почту
- Популярные CIS: Jenkins (Hudson), Atlassian Bamboo, Cruise Control

Continuous Integration (CI)

CI: полный цикл разработки



Agenda

- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

Why so serious?

Пишите свои программы так, как будто человек, который их будет поддерживать, является серийным маньяком-убийцей и знает ваш домашний адрес.

(Стив Макконнелл, «Совершенный код»)

```
public void addDependence(int toId, int type) {
    if (!getAllPossibleValues(toId, 0, type).isEmpty()) {

        boolean b1 = false;
        boolean b2 = false;
        switch (type) {
            case 1:
                b1 = true;
                break;
            case 2:
                b2 = true;
                break;
        }

        Set<TariffOption> newSet = new HashSet();
        for (TariffOption to1 : this.tariff.getTariffOptions()) {
            if (to1.getTariffOptionId() == toId) {
                Dependence d = new Dependence();
                d.setTariffOption(to1);
                d.setDependenceIsRequire(b1);
                d.setDependenceIsIncompatible(b2);
                d.setDependenceAnotherOperationId(0);
                to1.getDependences().add(d);
            }
            newSet.add(to1);
        }
        this.tariff.setTariffOptions(newSet);
    }
}
```

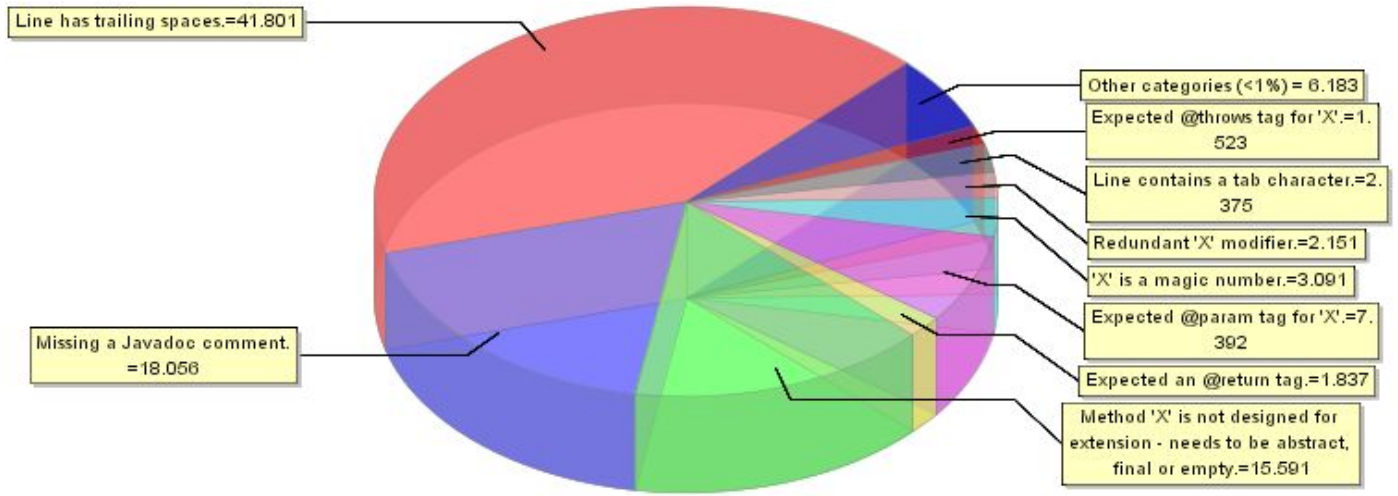
TariffController.java

PMD

- PMD – статический анализатор кода для языка Java
- Позволяет автоматически контролировать единообразие стиля кодирования
- Автоматически распознает
  - Дублирование кода
  - Неэффективные низкоуровневые реализации
  - Неиспользуемый код
  - Низкоуровневые антипатерны
- Конфигурируется перечнем правил в XML
- Есть плагин для maven'а и таск для ant'а
- Интегрирован с многими IDE

Checkstyle

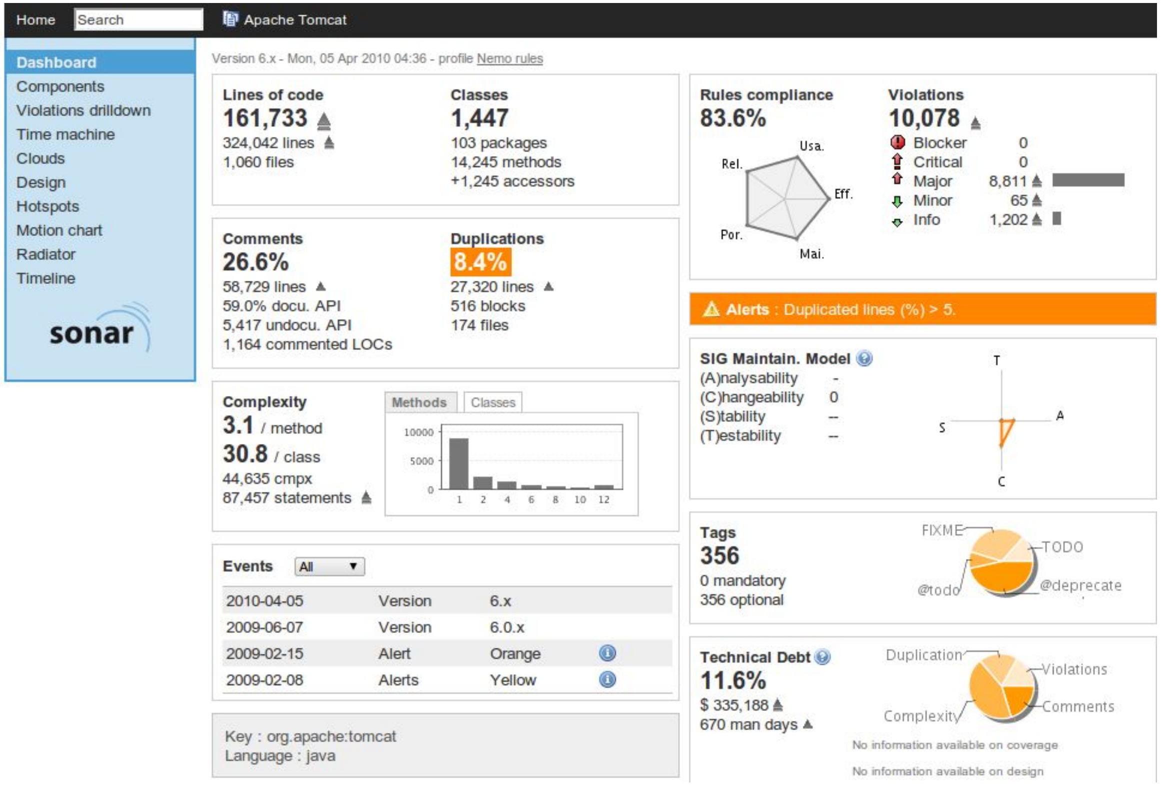
- Еще один статический анализатор исходного кода
- Делает упор на соблюдение стандартов кодирования, например Java Code Conventions
- Позволяет конфигурировать набор применяемых правил
- Отлично интегрирован с популярными IDE и build-системами
- При работе в IDE может подсвечивать ошибки прямо в процессе написания кода



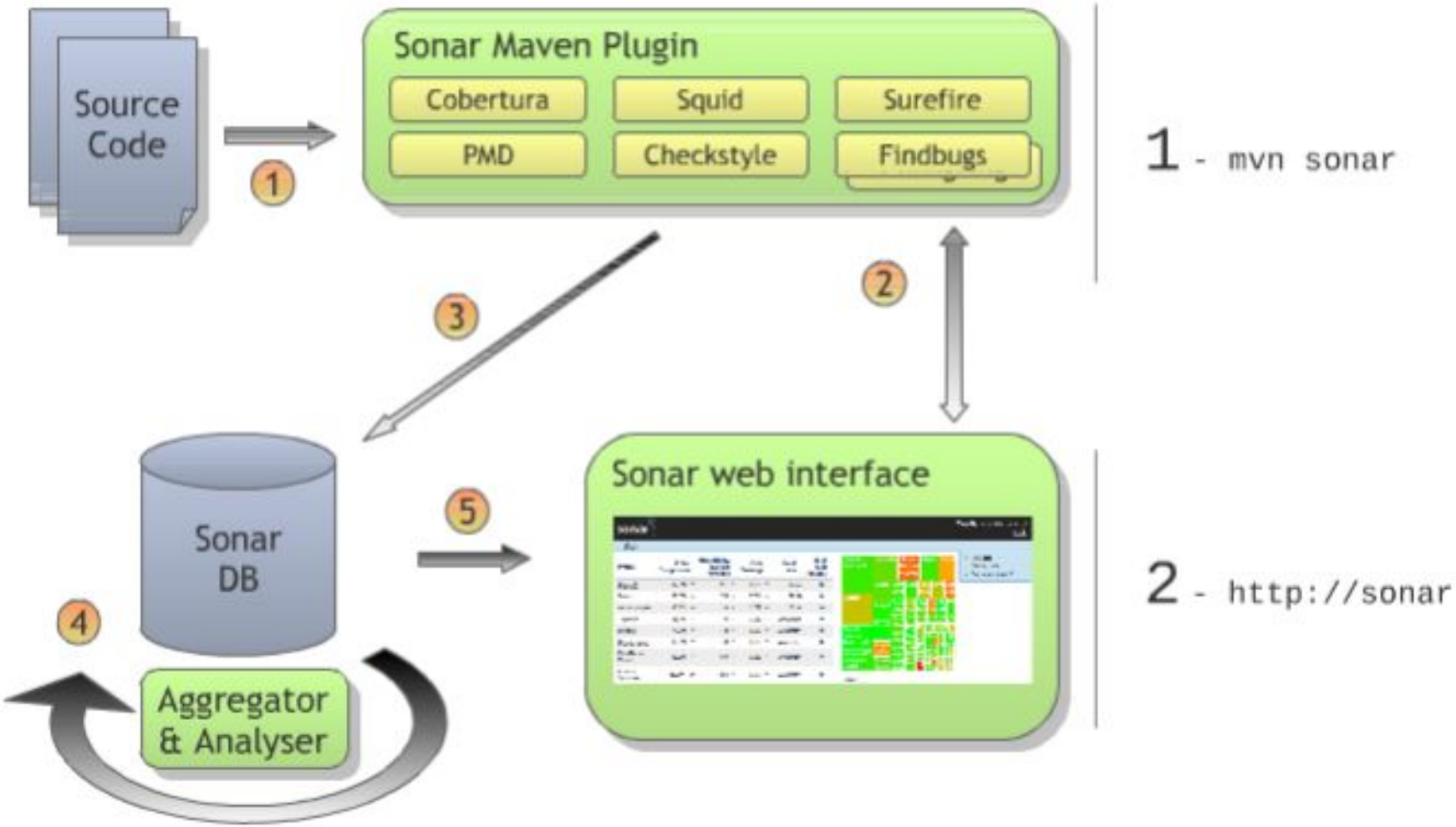
T · · Systems

Sonar

- Модульная open-source платформа для контроля качества исходного кода
- Умеет снимать огромное количество метрик
- Визуализирует их, генерирует отчеты и представляет динамику во времени
- Использует PMD, Checkstyle и собственные метрики качества
- Анализирует покрытие модульными тестами несколькими методами
- Ищет дубликаты
- Считает совокупный «технический долг»



Sonar: Рабочий цикл



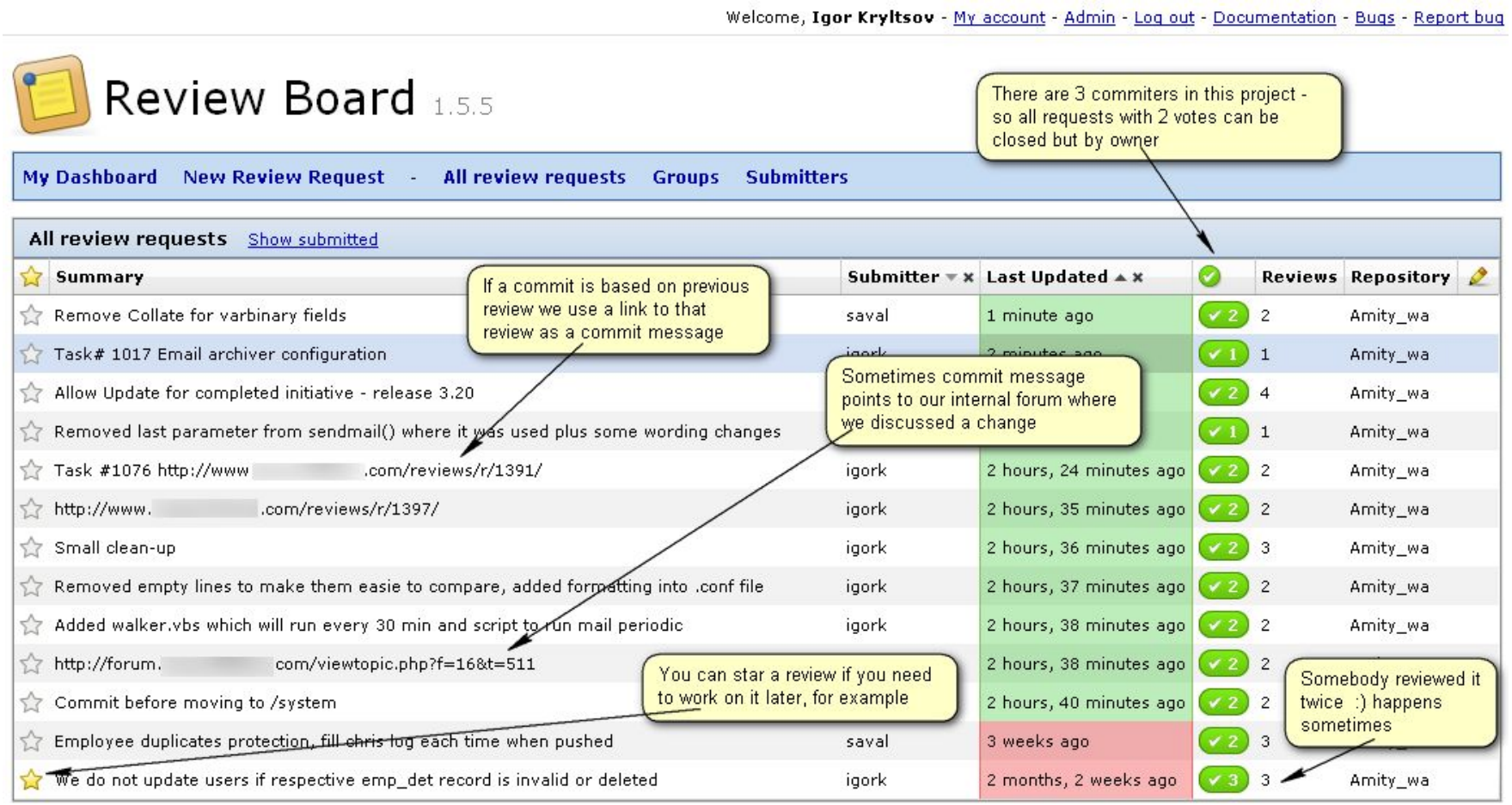
T · · Systems

Code Review

- Wiki: систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки

- Tools:
  - Review Board
  - Barkeep
  - Code Striker, etc.

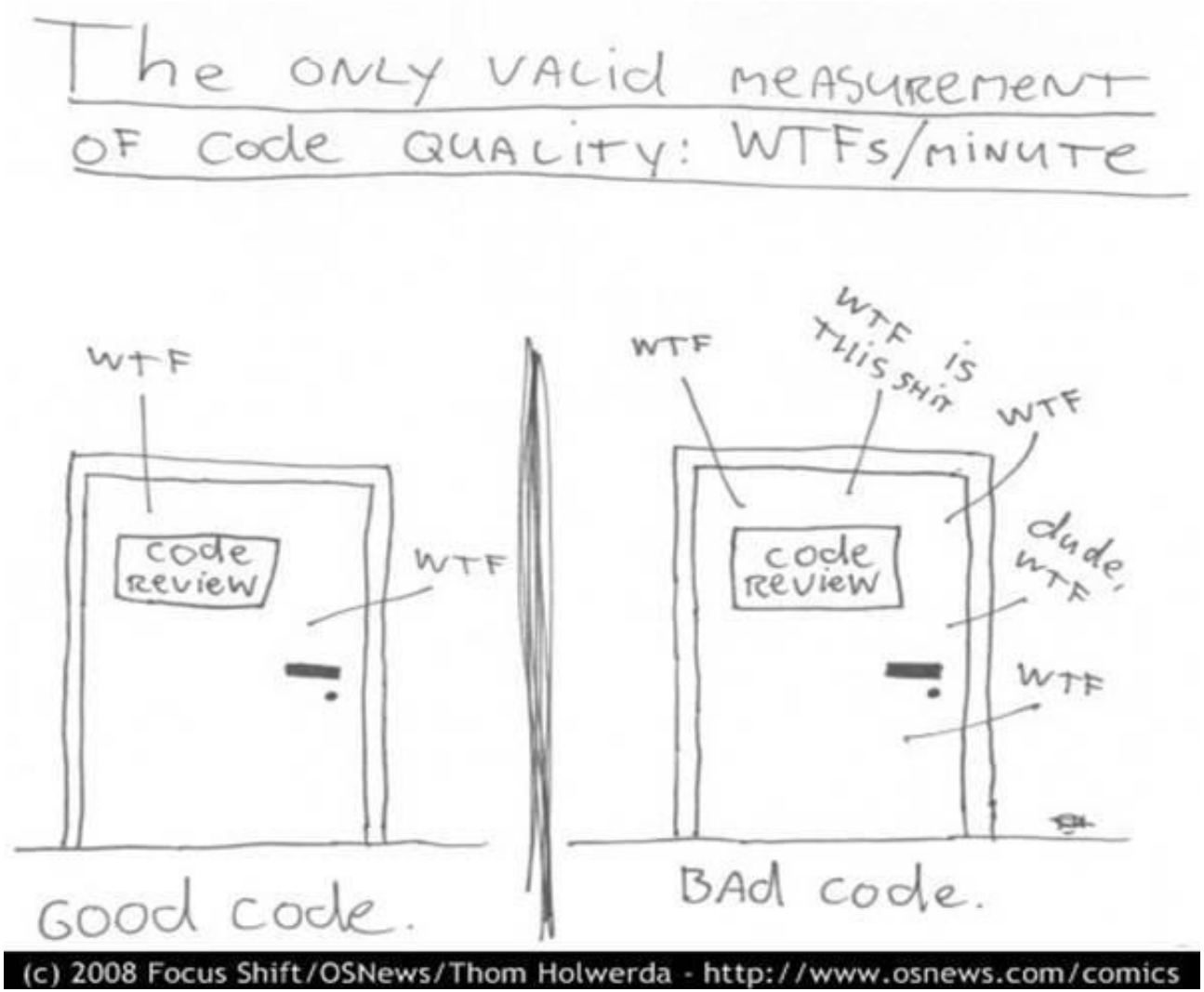
Review Board



Summary	Submitter	Last Updated		Reviews	Repository
Remove Collate for varbinary fields	saval	1 minute ago	✓ 2	2	Amity_wa
Task# 1017 Email archiver configuration	igork	2 minutes ago	✓ 1	1	Amity_wa
Allow Update for completed initiative - release 3.20			✓ 2	4	Amity_wa
Removed last parameter from sendmail() where it was used plus some wording changes			✓ 1	1	Amity_wa
Task #1076 http://www .com/reviews/r/1391/	igork	2 hours, 24 minutes ago	✓ 2	2	Amity_wa
http://www. .com/reviews/r/1397/	igork	2 hours, 35 minutes ago	✓ 2	2	Amity_wa
Small clean-up	igork	2 hours, 36 minutes ago	✓ 2	3	Amity_wa
Removed empty lines to make them easie to compare, added formatting into .conf file	igork	2 hours, 37 minutes ago	✓ 2	2	Amity_wa
Added walker.vbs which will run every 30 min and script to run mail periodic	igork	2 hours, 38 minutes ago	✓ 2	2	Amity_wa
http://forum. com/viewtopic.php?f=16&t=511		2 hours, 38 minutes ago	✓ 2	2	
Commit before moving to /system		2 hours, 40 minutes ago	✓ 2	2	
Employee duplicates protection, fill chris log each time when pushed	saval	3 weeks ago	✓ 2	3	
We do not update users if respective emp_det record is invalid or deleted	igork	2 months, 2 weeks ago	✓ 3	3	Amity_wa

T · · Systems

Code Review


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
Bad code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

Javadoc

- **Javadoc** — стандарт для документирования классов Java. Большинство сред разработки программного обеспечения автоматически генерируют HTML-документацию, используя Javadoc.

- Джавадокированию в вашей работе подлежат:
  - Все методы (кроме геттеров/сеттеров POJO конвенции)
  - Все классы
  - Все сложные алгоритмы

T · · Systems

Javadoc

Когда я возвращаюсь к разработке своего кода, который я не комментировал.

*http://developerslife.ru/12

Practice #4 – checkstyle

- Требуется: скачать и установить плагин Checkstyle
- Help->Eclipse Marketplace->Checkstyle Plug-in
- Внимательно читаем лицензионное соглашение и соглашаемся с ним
- Устанавливаем плагин

Agenda

- IDE
- Автоматизация build-процесса
- Системы контроля версий
- Continuous Integration
- Контроль качества исходного кода
- Дебаг, мониторинг и профилировка

Debug

- **Отладка** — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Дебагер есть во всех Java IDE. Наиболее удобный – в IDEA.

- Дебагер помогает:
  - Узнать значения переменной в моменте
  - Построчно отлаживать программу
  - Переходить вверх и вниз по стеку

Debug – чтобы не было потом вот так

```
public void addTariff(Tariff tariff) {
    EntityTransaction transaction = entityManager.getTransaction();
    try {
        transaction.begin();

        System.err.println("1");
        entityManager.persist(tariff);

        System.out.println("2");
        for (TariffOption to : tariff.getTariffOptions()) {
            to.setTariff(tariff);
            to.setTariffOptionId(null);
            System.out.println("2-1");
            entityManager.persist(to);

            System.out.println("2-2");
            for (Dependence d : to.getDependences()) {
                d.setTariffOption(to);
                d.setDependenceId(null);

                System.out.println("3-1");
                entityManager.persist(d);
                System.out.println("3-2");
            }
        }

        System.out.println("4");
        transaction.commit();
        System.out.println("5");
```

T · · Systems

Profiling

- Под профилировкой понимают сбор характеристик работающего приложения
- В них входит использование памяти, динамика процессоров, трассировка вызовов методов
- Профайлер помогает обнаружить
  - Горячие места в коде, которые стоит оптимизировать
  - Чем занята память и течет ли она
  - Bottleneck'и производительности
  - Deadlock'и, состояние starvation
  - Что и когда делает GC в приложении
- Профайлеры подразделяются на
  - Инструментирующие
  - Сэмплирующие

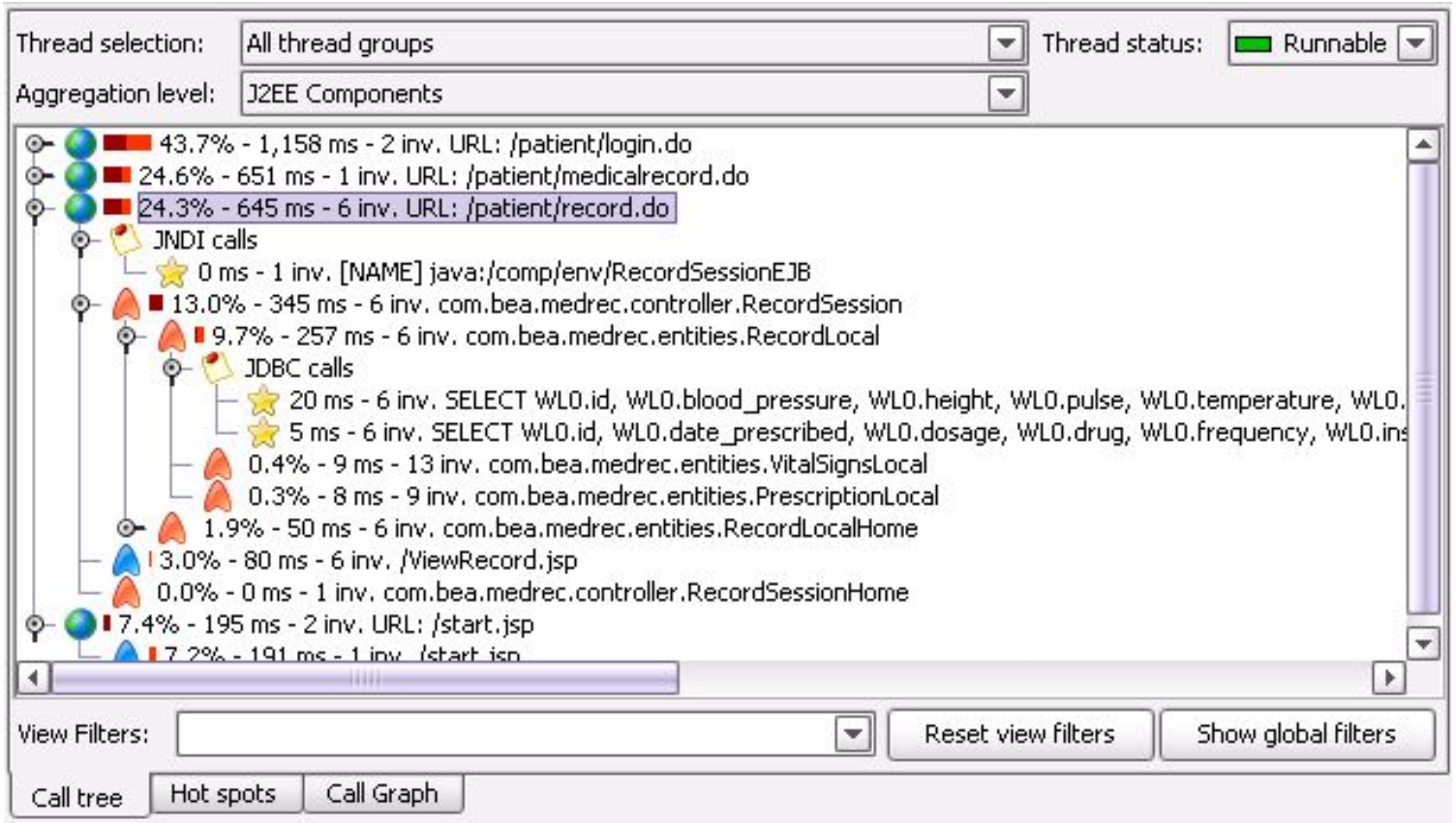
T · · Systems

VisualVM

- Инструментирующий профайлер из состава JDK, начиная с JDK 6u7
- Умеет инструментировать приложение на лету, не требуя перезапуска
- На самом деле представляет собой кусок NetBeans'а
- Является инструментирующим профайлером, то есть влияет на работу самого профилируемого приложения
- Очень простой в освоении
- Полностью покрывает потребности среднего разработчика в мониторинге и профилировке приложения

JProfiler

- Более серьезный инструмент, лучше показывает тонкие места
- Поддерживает удаленную профилировку
- Очень платный, но есть evaluation на 10 дней



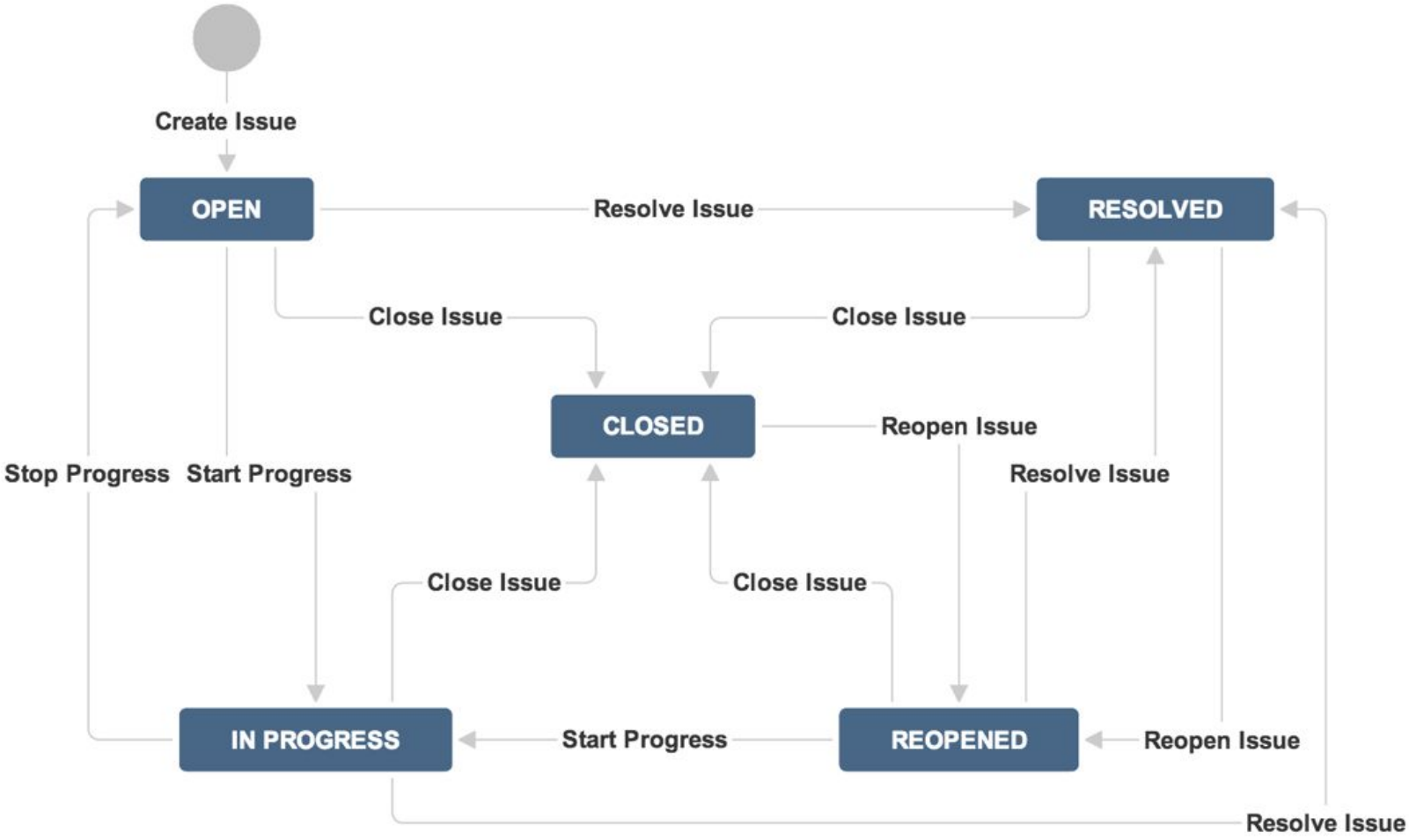
T · · Systems

What else?

- Task tracking system (+ Wiki)

- Примеры:
  - Atlassian Jira (+ Confluence)
  - Redmine
  - Bugzilla
  - Mantis
  - YouTrack

Default Workflow



T · · Systems


KEEP
CALM
AND
ASK ME
QUESTIONS

Homework

- Сделать maven проект из архетипа (например maven-archetype-webapp).
- Импортировать его в Eclipse.
- Запустить проект на Tomcat.
- Результаты отписать в группу.