



Ambiente de integração contínua

Ferramentas e soluções para
diminuir a complexidade do
desenvolvimento

Helder da Rocha (helder@argonavis.com.br)



Helder da Rocha

- **Java**, since 1995
- **Objective-C** & iPhone since 2008
- HTML, JavaScript, Web
- www.argonavis.com.br
- www.helderdarocha.com.br



Objetivos

- Como lidar com a complexidade crescente do **processo de desenvolvimento**
- Que **ferramentas e soluções** existem para isso?



Lei da preservação da complexidade

- A complexidade não desaparece!
- Ela se **muda para outro lugar** e se esconde
 - e continua a crescer por lá



Para onde foi a complexidade?

- Com orientação a objetos, ferramentas, frameworks, etc. **programar ficou mais simples**
- Mas a complexidade aumentou nos **processos e configuração** dos ambientes
 - Manter repositórios, versões, dependências
 - Configuração, metadados, mapeamentos, ligações
 - Integração, protocolos, adaptadores, filtros, proteções
 - Processo, build, documentação, testes



Como diminuir a complexidade?

- Nos processos: **mais feedback**
 - Metodologias de desenvolvimento ágeis
 - Ferramentas para **organizar e simplificar** o processo
 - Ferramentas para **coletar e organizar** informações automaticamente: **métricas, qualidade**, prazos, etc.
- Na configuração: **menos duplicação de esforços**
 - **Automatizar processos** como build, deploy, versionamento, etc.
 - **Integrar ambientes** e ferramentas
 - **Gerar código**



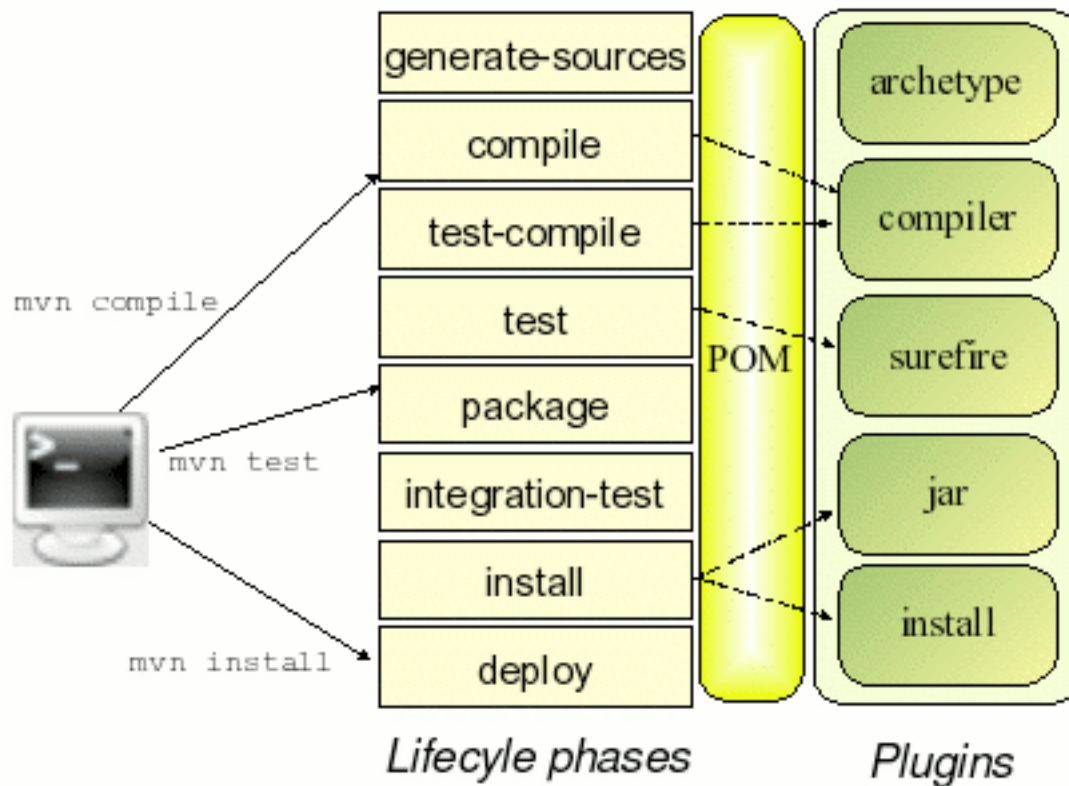
Ferramentas

- Build
 - Ant / Ivy / Maven
- Qualidade e métricas
 - PMD / Checkstyle / Findbugs
 - Sonar
 - JUnit / Cobertura
- Documentação
 - Javadoc via Ant e Maven, Graphviz
- Geração de código, configuração
 - XSLT / XPath



Maven (maven.apache.org)

- Gerencia builds e repositórios



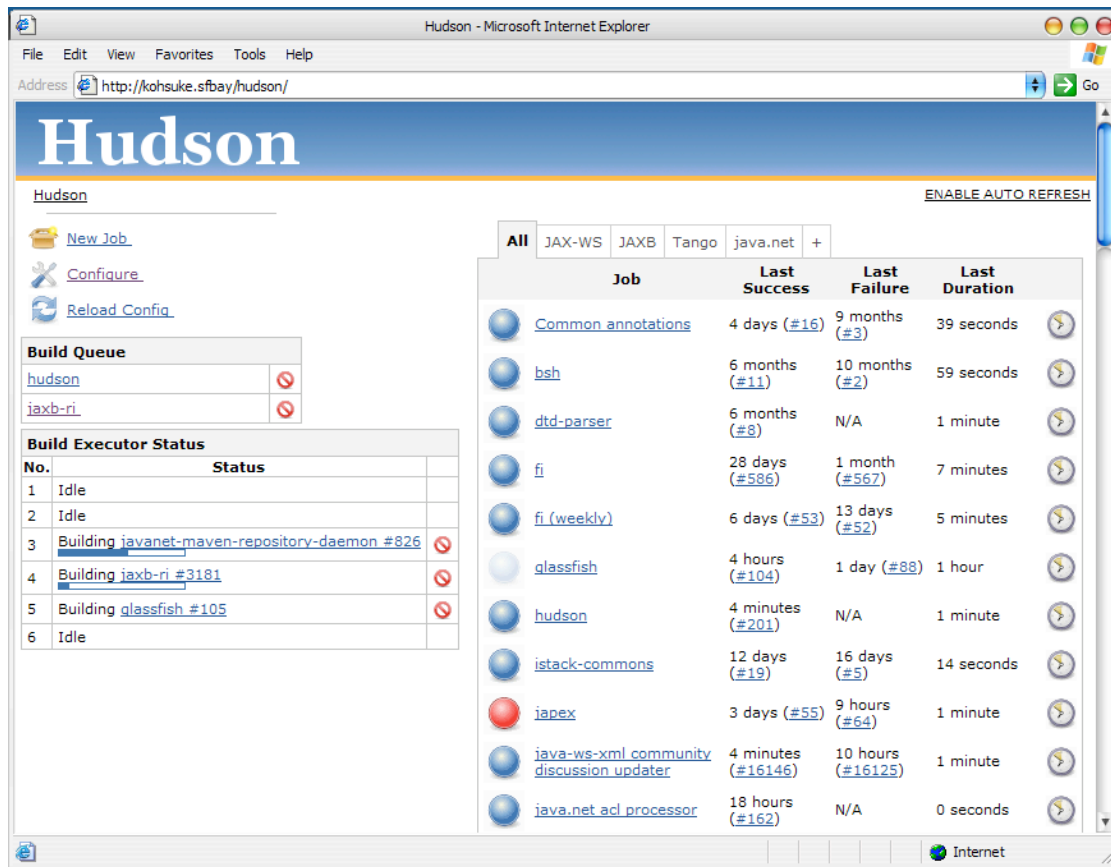
Ant + Ivy (ant.apache.org)

- Gerencia builds (Ant) e repositórios
 - Alternativa ao Maven
 - Podem ser integrados a um ambiente Maven



Hudson (https://hudson.dev.java.net/)

- Integração contínua + Feedback contínuo



The screenshot shows the Hudson web interface in a Microsoft Internet Explorer browser window. The address bar displays `http://kohlsuke.sfbay/hudson/`. The main heading is "Hudson" with a sub-heading "Hudson" and a link to "ENABLE AUTO REFRESH".

On the left side, there are navigation links: "New Job", "Configure", and "Reload Config". Below these are two tables:

Build Queue

Job	Status
hudson	⊘
jaxb-ri	⊘

Build Executor Status

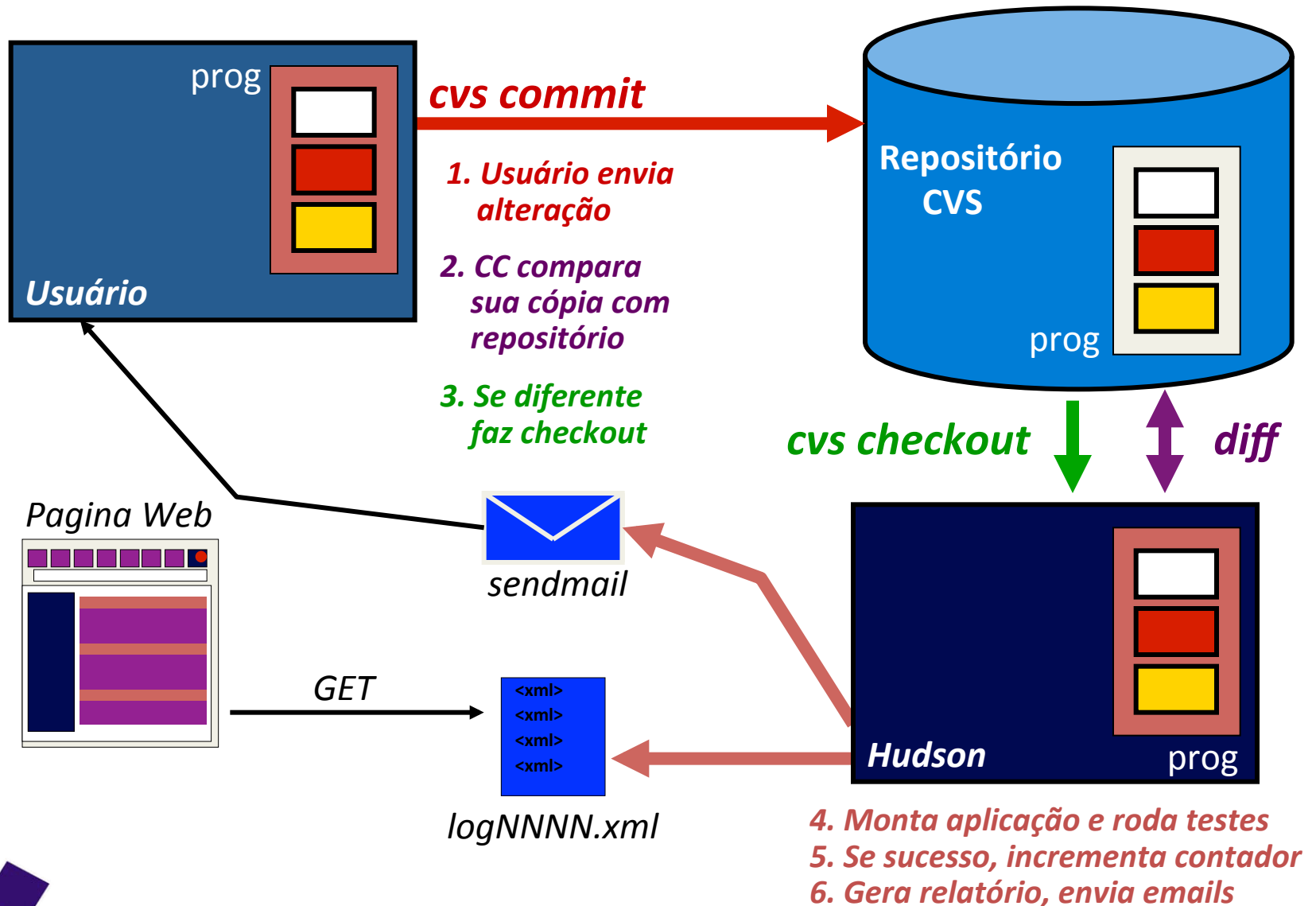
No.	Status
1	Idle
2	Idle
3	Building javanet-maven-repository-daemon #826 ⊘
4	Building jaxb-ri #3181 ⊘
5	Building glassfish #105 ⊘
6	Idle

The main content area displays a table of jobs with columns: Job, Last Success, Last Failure, and Last Duration. The jobs are filtered by "All" and "java.net".

Job	Last Success	Last Failure	Last Duration
Common annotations	4 days (#16)	9 months (#3)	39 seconds
bsh	6 months (#11)	10 months (#2)	59 seconds
dtd-parser	6 months (#8)	N/A	1 minute
fi	28 days (#586)	1 month (#567)	7 minutes
fi (weekly)	6 days (#53)	13 days (#52)	5 minutes
glassfish	4 hours (#104)	1 day (#88)	1 hour
hudson	4 minutes (#201)	N/A	1 minute
istack-commons	12 days (#19)	16 days (#5)	14 seconds
iapex	3 days (#55)	9 hours (#64)	1 minute
java-ws-xml community discussion updater	4 minutes (#16146)	10 hours (#16125)	1 minute
java.net acl processor	18 hours (#162)	N/A	0 seconds



Integração contínua



Hudson: dashboard

Usuarios CVS
que fazem
commit ou
montam o
projeto

The screenshot shows the Hudson dashboard interface. At the top, there's a navigation bar with the Hudson logo and a search box. Below that, a main content area displays a table of build history. The table has columns for status (S, W, L), job name, last success, last failure, and last duration. Annotations with blue arrows point to specific elements: 'Último build falhou' points to the 'W' status icon; 'Estado dos builds (quanto a violações, testes, sucesso, etc.)' points to the 'S' status icon; 'Tempo desde última falha' points to the 'Last Failure' column; 'Tempo desde último sucesso' points to the 'Last Success' column; and 'Último build obteve sucesso' points to the 'L' status icon. On the left, a sidebar contains navigation links like 'People', 'Build History', and 'Project Relationship'. At the bottom, there's a footer with page generation information and the Hudson version number.

Último build falhou

Estado dos builds (quanto a violações, testes, sucesso, etc.)

Tempo desde última falha

Tempo desde último sucesso

Último build obteve sucesso

S	W	Job ↓	Last Success	Last Failure	Last Duration
●	☁	Workspace_CRM	20 hr (#13)	55 min (#27)	4 min 0 sec
●	☁	[REDACTED]	35 min (#51)	23 hr (#37)	7 min 7 sec
●	☁	[REDACTED]	15 min (#24)	2 hr 16 min (#22)	8 min 50 sec

Build Queue
No builds in the queue.

#	Status
1	Idle
2	Idle

Page generated: Nov 26, 2009 5:58:30 PM Hudson ver. 1.335

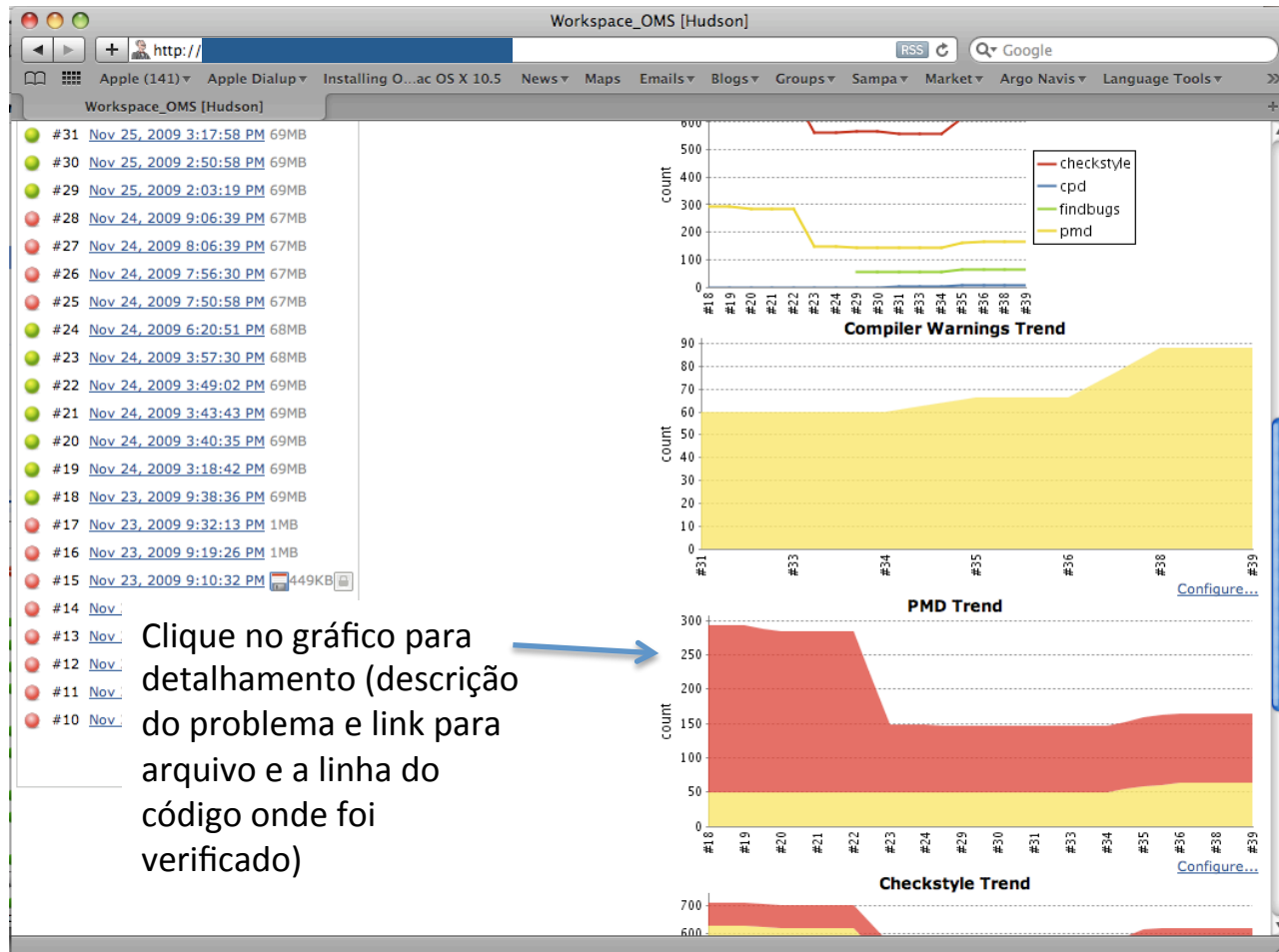


PMD + Checkstyle + Findbugs

- Ferramentas de análise estática
- Verificam **qualidade** do código
- Geralmente são incluídas no ambiente de desenvolvimento (Eclipse, Netbeans)
 - Mas também podem ser executadas via Ant, Maven e incluídas na integração contínua
- **Sonar** e **Hudson** são ótimos para apresentar resultados de PMD, Checkstyle e Findbugs
- Onde encontrar
 - <http://pmd.sourceforge.net/>
 - <http://checkstyle.sourceforge.net>
 - <http://findbugs.sourceforge.net/>



Gráficos de tendências: Hudson



Relatórios de violações e falhas de testes

The screenshot shows a web browser window displaying a Hudson report. The report lists various code violations, each with a line number and a description. One violation is highlighted with a red box, and a blue arrow points to it from the text 'Detalhes de uma violação de estilo'.

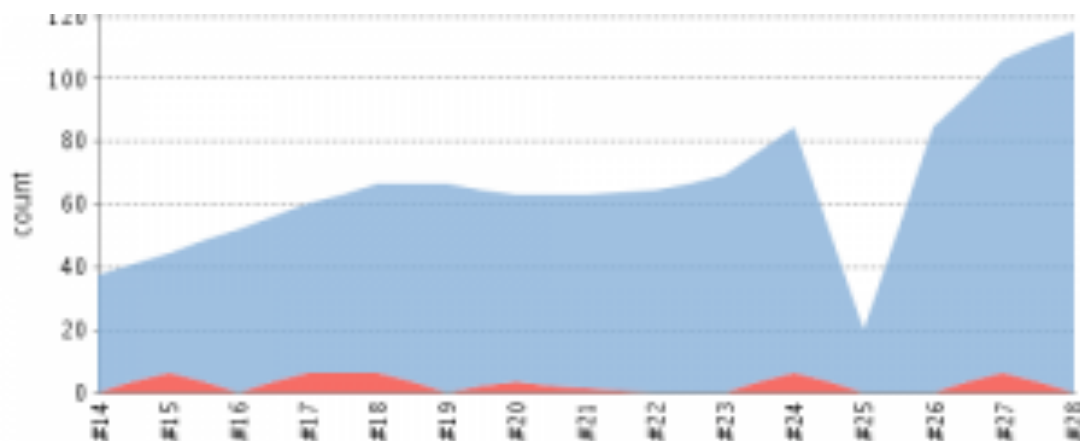
Line	Severity	Class	Detail
279	High	AvoidDuplicateLiterals	The String literal "" appears 5 times in this file; the first occurrence is on line 279

Detalhes de uma violação de estilo



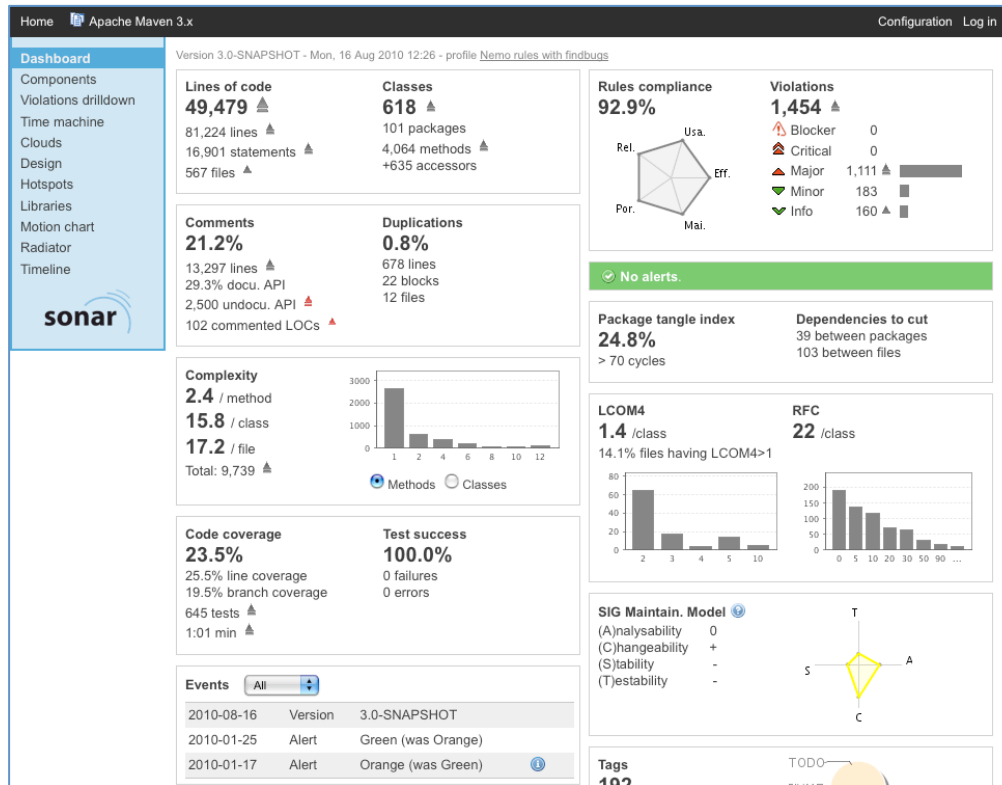
JUnit e Cobertura

- Seu código deve ter testes unitários
- Cobertura identifica o nível de cobertura desses testes (métricas via Sonar)
- Hudson gera gráfico de tendências



Sonar (www.sonarsource.org)

- Instale no servidor Web; Gere os dados via plug-in do Maven (mesmo em projetos Ant)
 - `mvn sonar:sonar`



Geração de javadoc

- UML Graph + Javadoc (a cada build)
 - <http://www.umlgraph.org>
 - <http://www.graphviz.org>

The screenshot displays a web browser window with the following content:

- Package Tree:** A list of packages including `com.beer.business.data`, `com.beer.business.domain`, `com.beer.business.service`, `com.beer.common`, and `com.beer.web`.
- Class Diagram:** Shows the `com.beer.business.data.LoginDaoImpl` class implementing the `com.beer.business.data.LoginDao` interface and the `com.beer.common.BaseDao` interface. The `LoginDao` interface has a `login(String, String, String)` method. The `BaseDao` interface has methods `getConnection()`, `getConnection(String)`, `getConnection(String, String, String, String, String, String)`, and `closeConnection(ResultSet)`. The `LoginDaoImpl` class has a `login(String, String, String)` method.
- Code Snippets:**

```
public class LoginDaoImpl
    extends BaseDao
    implements LoginDao

    + login(String, String, String)
```

```
public interface LoginDao
    + login(String, String, String)
```

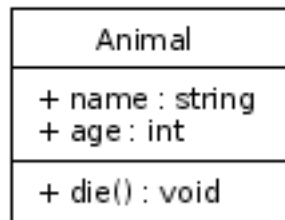
```
public interface BaseDao
    # getConnection() java.sql.Connection
    # getConnection(String) java.sql.Connection
    # getConnection(String, String, String, String, String, String) java.sql.Connection
    # closeConnection(ResultSet) java.sql.ResultSet, stmt java.sql.PreparedStatement, conn java.sql.Connection
```
- Field Summary:** Lists fields inherited from the `com.beer.business.data.LoginDao` interface, including `VERIFY_LOGIN`.
- Constructor Summary:** (Partially visible at the bottom).



Graphviz (www.graphviz.org)

- Linguagem declarativa para gerar gráficos
- Exemplo com UML

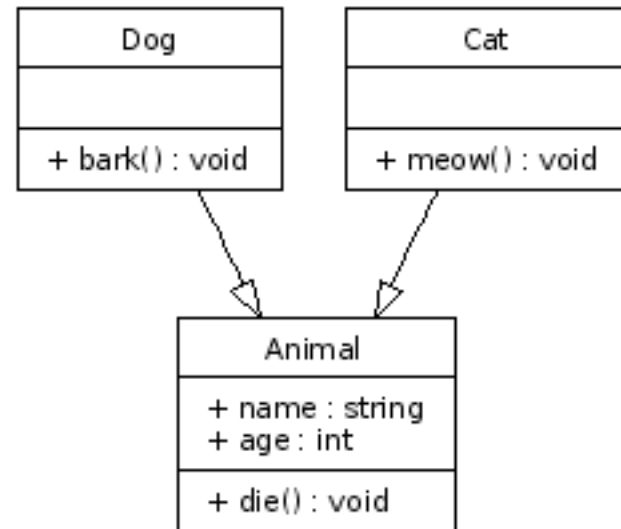
Animal [label = "{Animal|+ name : string|+ age : int|+ die() : void|}"]



edge [arrowhead = "empty"]

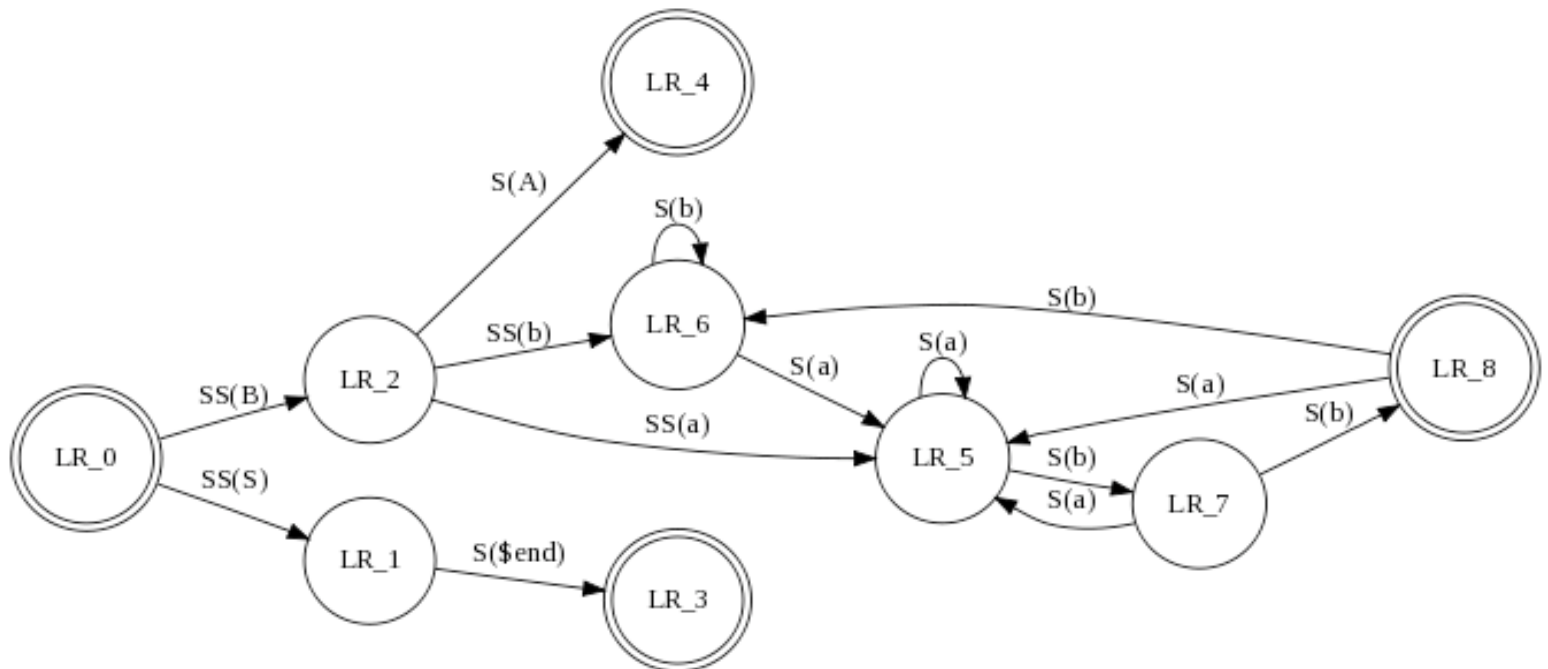
Dog -> Animal

Cat -> Animal



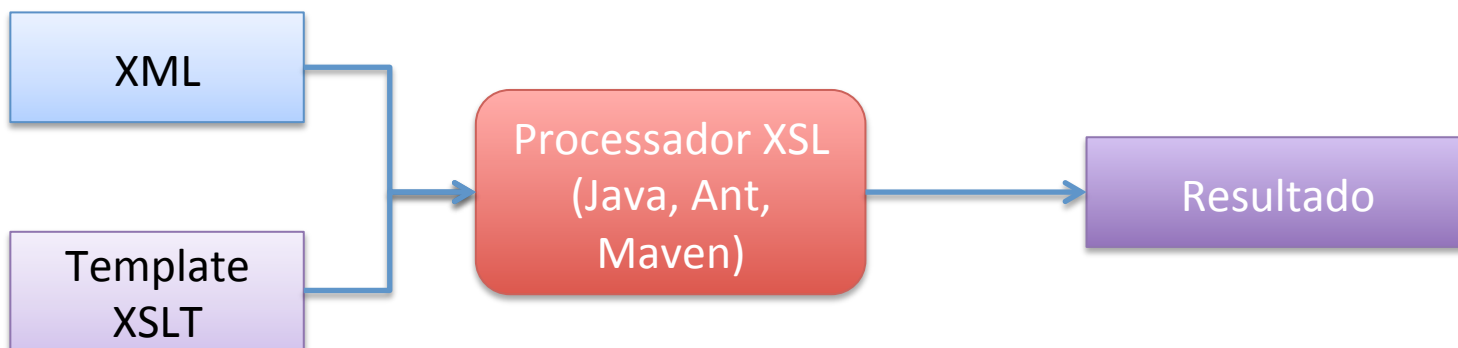
Graphviz

- Graphviz faz bem mais que UML
 - Use para gerar documentação gráfica complexa
 - Escreva “dot language” (ou gere a partir de XML com XSLT/XPath)



XSLT e XPath (www.w3.org)

- Pode ser usado via Java, tarefa do Ant, plug-in do Maven ou linha de comando
- Use XSLT/XPath para extrair dados de arquivos de configuração XML para
 - Gerar outros arquivos (de qualquer tipo)
 - Gerar PDF
 - Gerar documentação com tabelas HTML
 - Gerar documentação gráfica com Graphviz



Conclusão

- Vale a pena investir em tornar seu ambiente de desenvolvimento mais automatizado
 - Menos tempo perdido com **tarefas burocráticas**
 - **Informações atualizadas** e precisas sobre o estado atual do desenvolvimento e **qualidade** do código
 - **Subsídios** para tomadas de decisões em ambientes que adotam metodologias ágeis
- Há várias ferramentas open-source que podem ajudar nessa tarefa, como
 - Sonar, Hudson, Maven, Ant+Ivy
 - XSLT/Framemarker, Graphviz

