TESTES ESTÁTICOS

Prof. Me. Rober Marcone Rosi

SUMÁRIO

REVISÃO E O PROCESSO DE TESTE

- Ao contrário dos testes dinâmicos, **as técnicas de teste estático não pressupõem a execução do software que está sendo testado. Elas são:**
  - **manuais (revisão) ou**
  - **automatizadas (análise estática).**

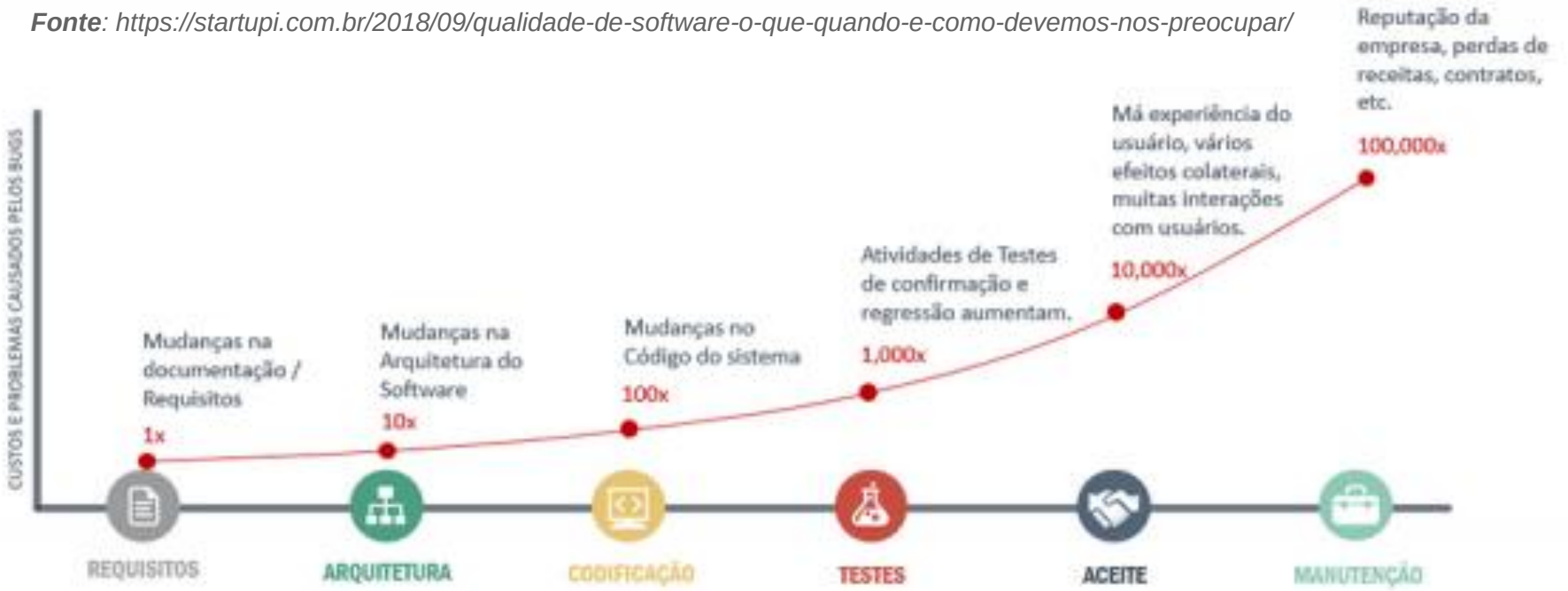
- Revisões, análises estáticas e testes dinâmicos
  - têm os mesmos objetivos – identificar defeitos.
  - são complementares: as diferentes técnicas podem encontrar diferentes tipos de defeitos eficazmente e eficientemente.
  - Em contraste com o teste dinâmico, revisões encontram defeitos ao invés de falhas.

REVISÃO E O PROCESSO DE TESTE

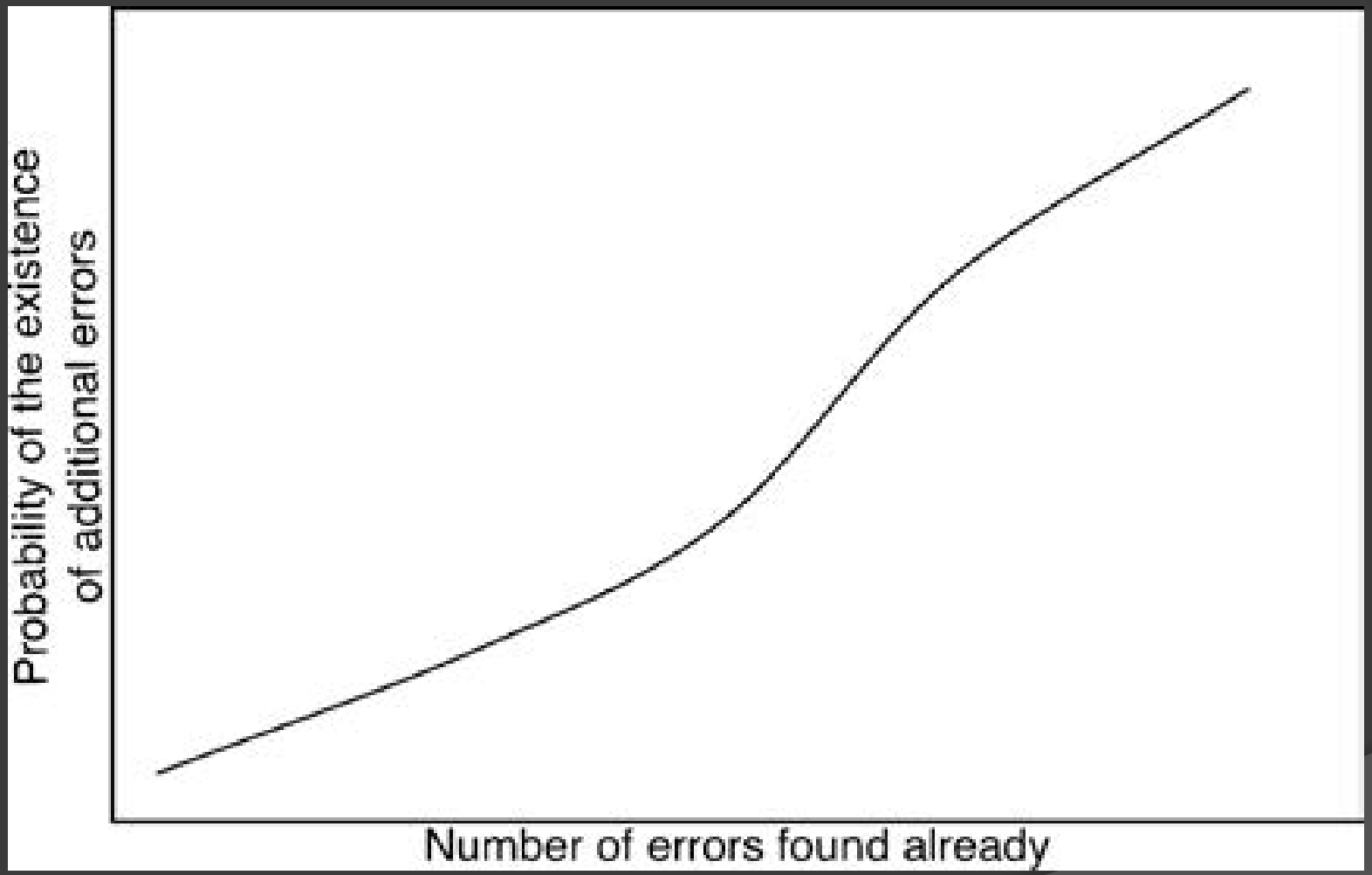
Revisão - Introdução

- É uma maneira de testar o produto de software (incluindo o código) e pode ser realizada bem antes da execução do teste dinâmico.
- Defeitos detectados durante as revisões o mais cedo possível no ciclo de vida do software são muitas vezes mais baratos do que aqueles detectados e removidos durante os testes (ex.: defeitos encontrados nos requisitos).

***Fonte**: https://startupi.com.br/2018/09/qualidade-de-software-o-que-quando-e-como-devemos-nos-preocupar/*



REVISÃO E O PROCESSO DE TESTE



Fonte: *Glenford J. Myers – The Art of Software Testing, 2012*

REVISÃO E O PROCESSO DE TESTE

Revisão - Introdução

- **Uma revisão pode ser feita inteiramente como uma atividade manual, mas há também ferramentas de suporte.**
- A principal atividade manual é examinar o produto de trabalho e fazer os comentários sobre ele.
- **Qualquer software pode ser revisado, incluindo a especificação de requisitos, diagramas, código, plano de teste, especificação de teste, casos de teste, script de teste, manual do usuário ou páginas web.**

REVISÃO E O PROCESSO DE TESTE

Revisão - Introdução

- Os benefícios das revisões incluem
  - a detecção e correção antecipada de defeitos,
  - ganho no desenvolvimento em termos de produtividade,
  - redução do tempo no desenvolvimento,
  - redução do custo e tempo de teste,
  - menos defeitos e
  - melhoria na comunicação.

- A revisão pode encontrar omissões, por exemplo, nos requisitos, que não são normalmente encontrados no teste dinâmico.

- Os defeitos mais facilmente encontrados durante revisões do que em testes dinâmicos são:
  - desvios de padrões
  - defeitos de requisitos
  - defeitos de modelagem
  - manutenibilidade insuficiente e
  - especificação incorreta de interfaces.

PROCESSO DE REVISÃO

- As revisões variam de muito informais para muito formais (ex.: bem estruturadas e reguladas).

- A formalidade do processo de revisão é relacionada a fatores como
  - a maturidade do processo de desenvolvimento
  - requisitos legais e reguladores ou
  - a necessidade de acompanhamento de auditoria.

- O modo como uma revisão é conduzida depende do seu objetivo, como por exemplo:
  - encontrar defeitos
  - obter compreensão, discussão ou decisões por um consenso.

TIPOS DE REVISÃO

- Um único documento pode ser objeto para mais de uma revisão.

- Se mais de um tipo de revisão for usado, a ordem pode variar. Por exemplo, uma revisão informal pode ser conduzida antes de uma revisão técnica, ou uma inspeção pode ser executada em uma especificação de requisitos antes de um acompanhamento com clientes.

- **Os tipos de revisão são:**
  - **Revisão formal**
  - **Revisão informal**
  - **Acompanhamento**
  - **Revisões técnicas**
  - **Inspeção**

FASES DE UMA REVISÃO FORMAL

- Uma revisão formal normalmente possui as seguintes fases principais:
  - Planejamento
  - *Kick-off*
  - Preparação individual
  - Reunião de revisão
  - Retrabalho
  - Acompanhamento

FASES DE UMA REVISÃO FORMAL

- **Planejamento:**
  - Definir os critérios de revisão.
  - Selecionar a equipe.
  - Alocar as funções.
  - Definir os critérios de entrada e de saída para os diversos tipos de revisão formal (ex.: inspeção).
  - Selecionar quais as partes dos documentos serão vistos.
  - Checar os critérios de entrada (para diversos tipos de revisão formal).

- ***Kick-off:***
  - Distribuir os documentos.
  - Explicar os objetivos, processos e documentos para os participantes.

- **Preparação individual:**
  - Análise da documentação para a reunião de revisão.
  - Anotar os defeitos em potenciais, questões e comentários.

FASES DE UMA REVISÃO FORMAL

- **Reunião de revisão:**
  - Discussão ou registro, com resultados documentados ou anotações (para os tipos de revisões mais formais).
  - Anotar os defeitos, fazer recomendações para o tratamento de defeitos ou tomar decisões sobre os defeitos.
  - Examinar, avaliar e registrar questões durante as reuniões de acompanhamento.

- **Retrabalho:**
  - Resolver defeitos encontrados, tipicamente feitos pelo autor.
  - Registrar os status atuais dos defeitos (para revisões formais).

- **Acompanhamento:**
  - Checar se os defeitos foram encaminhados.
  - Obter métricas.
  - Checar os critérios de saída (para tipos de revisões formais).

PAPÉIS E RESPONSABILIDADES EM UMA TÍPICA REVISÃO FORMAL

- **Gerente:** toma decisão durante a realização da revisão, aloca tempo nos cronogramas de projeto e determina se o objetivo da revisão foi atingido.

- **Moderador:** a pessoa que lidera a revisão do documento ou conjunto de documentos, incluindo o planejamento da revisão, e o acompanhamento após a reunião. Se necessário, o moderador mediará entre os vários pontos de vista e é muitas vezes quem responderá pelo sucesso da revisão.

- **Autor:** é a pessoa que escreveu ou que possui a responsabilidade pelos documentos que serão revisados.

- **Revisores:** indivíduos com conhecimento técnico ou de negócio (também chamados inspetores), que, após a preparação necessária, identificam e descrevem os defeitos encontrados no produto em revisão. Revisores podem ser escolhidos para representar diferentes funções e perspectivas no processo de revisão, e é parte integrante de qualquer reunião de revisão.

- **Redator:** documenta todo o conteúdo da reunião, problemas e itens em aberto que foram identificados durante a reunião.

PAPÉIS E RESPONSABILIDADES EM UMA TÍPICA REVISÃO FORMAL

- Olhando os documentos de diferentes perspectivas e usando "*checklists",* tornamos a revisão mais eficaz e eficiente.

- Por exemplo, um *"checklist"* baseado em perspectivas tais como a do usuário, desenvolvedor, testador, operador, ou um *"check-list"* típico de problemas de requisitos pode ajudar a descobrir problemas não detectados anteriormente.

TIPOS DE REVISÃO

- **Revisão informal:**
  - Não existe processo formal.
  - Pode haver programação em pares ou um líder técnico revisando a modelagem e o código.
  - A documentação é opcional.
  - A importância pode variar dependendo do revisor.
  - Principal propósito: uma forma de obter algum benefício a um baixo custo.

TIPOS DE REVISÃO

- **Acompanhamento:**
  - Reunião conduzida pelo autor.
  - Cenários, grupos de discussão, exercícios práticos.
  - Sessões sem restrição de tempo.
  - Opcionalmente há uma reunião preparatória dos revisores.
  - Opcionalmente, relatórios de revisão e lista de defeitos encontrados são preparados.
  - Opcionalmente há um redator.
  - Na prática pode variar de informal para muito formal.
  - Principal propósito: aprendizagem, obtenção de entendimento e encontrar defeitos.

TIPOS DE REVISÃO

- **Revisões técnicas:**
  - Documentado, processo de detecção de defeito definido que inclui colegas especialistas ou técnicos com a participação opcional da gerência.
  - Pode ser feito por um colega sem a participação da gerência.
  - Idealmente são conduzidas por um moderador treinado (que não seja o autor).
  - Reunião preparatória dos revisores.
  - Opcionalmente usa checklists.
  - Elaboração de um relatório de revisão, que inclui a lista de defeitos encontrados, se o produto de software corresponde às suas exigências e, quando apropriado, recomendações relacionadas com as descobertas.
  - Na prática, pode variar de informal para muito formal.
  - **Principais propósitos:** discussão, tomada de decisões, avaliar alternativas, encontrar defeitos, resolver problemas técnicos e checar a conformidade da padronização das especificações.

TIPOS DE REVISÃO

- **Inspeção :**
  - Conduzida pelo moderador (que não seja o autor).
  - Geralmente é uma análise por pares.
  - Papéis definidos.
  - Utilização de métricas.
  - Processo formal baseado em regras e utilização de checklist.
  - Entrada especificada e critérios de saída para a aceitação do produto de software.
  - Reunião de preparação.
  - Relatório de inspeção, lista de defeitos encontrados;
  - Processo de acompanhamento formal.
  - Opcionalmente, ter aperfeiçoamento do processo e um leitor.
  - Principal propósito: encontrar defeitos.

- Acompanhamento, revisões técnicas e inspeções podem ser executados dentro de um grupo de pessoas no mesmo nível organizacional.
- Este tipo de revisão é chamado de **“revisão por pares”.**

FATORES DE SUCESSO PARA AS REVISÕES

- Cada revisão tem um objetivo claramente definido.
- A pessoa adequada para os objetivos da revisão deve ser envolvida.
- Testadores são valorizados como revisores que contribuem para a revisão e aprendizado sobre o produto o que lhes permite preparar os testes facilmente.
- Defeitos encontrados são encorajados e expressados objetivamente.
- Deve-se lidar com os problemas pessoais e aspectos psicológicos (ex.: fazer com que a reunião seja uma experiência positiva para o autor).

FATORES DE SUCESSO PARA AS REVISÕES

- A análise é conduzida em uma atmosfera de confiança, o resultado não será utilizado para a avaliação dos participantes.
- Técnicas de revisão são aplicadas de forma a combinar com o tipo e nível do software e revisores.
- Caso necessário, checklists ou papéis são utilizados para aumentar a eficiência na identificação de defeitos.
- Treinamento é um item importante para as técnicas de revisão, especialmente para as técnicas formais, assim como as inspeções.
- Gerenciamento é importante para um bom processo de revisão (ex.: incorporando o tempo adequado para as atividades de revisão nos cronogramas de projetos).
- Há uma ênfase em aprender e aprimorar o processo.

ANÁLISE ESTÁTICA POR FERRAMENTAS - INTRODUÇÃO

- O objetivo da análise estática é encontrar defeitos no código fonte do software e na modelagem.
- Análise estática é feita sem a execução do software examinado pela ferramenta; já o teste dinâmico executa o software.
- Análise estática pode localizar defeitos que são dificilmente encontrados em testes.
- Como as revisões, a análise estática encontra defeitos ao invés de falhas.
- Ferramentas de análise estática analisam o código do programa (ex.: fluxo de controle e fluxo de dados), gerando, como saída, arquivos do tipo HTML e XML, por exemplo.

ANÁLISE ESTÁTICA POR FERRAMENTAS - BENEFÍCIOS

- Os benefícios da análise estática são:
  - Detecção de defeitos antes da execução do teste **dinâmico.**
  - Conhecimento antecipado sobre aspectos suspeitos no código ou programa através de métricas, por exemplo, na obtenção de uma medida da alta complexidade.
  - Identificação de defeitos dificilmente encontrados por testes dinâmicos.
  - Detecção de dependências e inconsistências em modelos de software, como links perdidos.
  - Aprimoramento da manutenibilidade do código e construção.
  - Prevenção de defeitos, se as lições forem aprendidas pelo desenvolvimento.

ANÁLISE ESTÁTICA POR FERRAMENTAS – DEFEITOS MAIS COMUNS ENCONTRADOS

- Defeitos mais comuns descobertos por ferramentas de análise estática incluem:
  - Referência a uma variável com valor indefinido.
  - Inconsistências entre as interfaces dos módulos e componentes.
  - Variáveis que nunca são usadas ou impropriamente declaradas.
  - Código morto.
  - Falta de lógica ou lógica errada (loops infinitos).
  - Construções excessivamente complicadas.
  - Violação de padrões de programação.
  - Vulnerabilidade na segurança.
  - Violação de sintaxe e de modelos.

ANÁLISE ESTÁTICA POR FERRAMENTAS

- Ferramentas de análises estáticas são tipicamente usadas por desenvolvedores (checando regras pré-definidas ou padrões de programação) antes e durante o teste de componente e de integração e por projetistas durante a modelagem do software.

- Ferramenta de análise estática pode produzir um grande número de mensagens de advertências que precisam ser gerenciadas para permitir o uso mais efetivo da ferramenta.

- Compiladores podem oferecer algum suporte para a análise estática, incluindo o cálculo de métricas.

Categorização dos Defeitos

- 1. Defeitos que ocasionam uma pane da aplicação. Esta categoria engloba os mais severos defeitos que param toda a aplicação pela reação de alguma entrada do usuário.
- 2. Defeitos que causam uma falha lógica. Esta categoria engloba todos os defeitos que causem uma falha lógica da aplicação, porém não ocasionam uma pane, por exemplo, um valor resultante errado.
- 3. Defeitos com tratamento de erro insuficiente. Esta categoria engloba os pequenos defeitos que não ocasionam uma pane da aplicação nem resultam em falhas lógicas, mas que não possuem um tratamento apropriado.

Categorização dos Defeitos

- Defeitos que violam os princípios da programação estruturada. Esta categoria engloba os defeitos que normalmente não impactam o software, mas podem ocasionar problemas de performance, segurança, entre outros.
- 5. Defeitos que diminuem a manutenibilidade do código. Esta categoria engloba todos os defeitos que somente afetam a legibilidade ou a mutabilidade do software.
- OBS: Claramente, pode-se constatar que os defeitos na categoria 1 possuem maior severidade e os da categoria 5 possuem menor severidade.

Categorização dos Defeitos

- Os verificadores estáticos de código, que são ferramentas automáticas para a verificação de código, podem verificar estilos de programação, erros ou ambos.
- Um verificador de erro utiliza a análise estática para encontrar código que viola uma propriedade de correção específica e que pode causar um comportamento anormal do programa em tempo de execução.
- Um verificador de estilo examina código para determinar se ele contém violações de regras de estilo de código particulares.

Categorização dos Defeitos

- Algumas regras de estilo, tais como "sempre colocar constantes no lado esquerdo de comparações" e "não utilizar instruções de atribuição na condição de instruções if ", podem ajudar a prevenir certos tipos de erros.
- Entretanto, violações destas orientações de estilo não são particularmente susceptíveis de serem erros.

Alguns exemplos de regras de estilo

Nome	*Categoria*	*Descrição*
Bloco *catch* vazio	3	Verifica a presença de blocos *catch* sem nenhuma instrução.
Atribuição em subexpressões	5	Verifica a presença de atribuições em subexpressões.
Comparação com constante	5	Verifica a presença de comparações com constante na qual esta não se encontra no lado esquerdo da comparação.
Instrução vazia	5	Verifica a presença de instruções vazias, isto é, apenas o ponto-e-vírgula (;).
Ordem de declaração	5	Verifica a presença de desordem na declaração das partes de classes ou interfaces de acordo com as convenções de código para a linguagem Java.
Padrão de nomenclatura	5	Verifica a presença de nomes de atributos, métodos e classes que estejam fora das convenções de código para a linguagem Java.
Posição da cláusula *default* em instruções *switch*	5	Verifica a presença de instruções *switch* na qual a cláusula *default* não vem após todas as claúsulas *case*.
Uso da referência *this*	5	Verifica a presença de acesso a algum atributo ou invocação de algum método da própria classe sem a utilização da referência *this*.
Uso de chaves	5	Verifica a presença de algum bloco condicional ou de repetição sem chaves de abertura e fechamento.

Alguns exemplos de erros

Nome	*Categoria*	*Descrição*
Conexão com banco de dados não encerrada	1	Verifica a presença de abertura de conexão ao banco de dados sem posterior fechamento.
Valor de retorno ignorado	2	Verifica a presença de algum retorno de função que foi ignorado.
Perda de referência nula (*null dereference*, em inglês)	3	Verifica a presença de acesso a um atributo ou chamada a um método a partir de uma referência que ainda está nula, o que ocasiona uma exceção que não será tratada.
Concatenação de *strings* em laços de repetição	4	Verifica a presença de concatenação de *strings* utilizando o operador '+' em laços de repetição.
Fluxo (*stream, em inglês*) não encerrado	4	Verifica a presença de abertura de fluxos sem posterior fechamento.
Não utilização de operador booleano de curto-circuito	4	Verifica a presença de possibilidade de lançamento de exceção na utilização de operadores booleanos que não são de curto-circuito, mas que deveriam ser.
Objetos iguais têm o mesmo código de dispersão (*hashcode*, em inglês)	4	Verifica a presença de objetos que redefinem o método '*equals*', mas não redefinem o método '*hashcode*' ou o inverso.
Utilização de '==' ao invés de '*equals*'	4	Verifica a presença de comparações de objetos utilizando o operador '==' ao invés do método '*equals*'.
Comparação com nulo redundante	5	Verifica a presença de comparação com nulo que está redundante.

Referências

- Baseado no material da Profa. Denise Franzotti Togneri e Prof. Ralf Luis de Moura
- VILLELA R. T. N. B. **Ferramentas para análise estática de códigos JAVA**. Monografia apresentada Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais. 2008.