Coverage Based Debugging Visualization

Danilo Mutti
Orientador Prof. Dr. Marcos Lordello Chaim

Motivação

- Defeitos são parte inerente do processo de desenvolvimento
- É uma atividade
  - notoriamente difícil
  - que consome grande quantidade de tempo
- Paradigma atual
  - Baseado quase que exclusivamente no uso de breakpoints e variantes

Objetivos

- Propor uma metáfora visual tri-dimensional para representar informações de depuração em programas grandes.
- Fornecer um roteiro que auxilie na localização de defeitos baseado na posição e cor dos elementos da metáfora.
- Disponibilizar ferramentas de busca e filtragem de elementos específicos na metáfora.

Estado da Arte

- Na indústria
  - *Breakpoints* e funcionalidades baseadas neste recurso
  - *Step over*
  - *Step into*
  - *Single stepping*
  - Avaliação de funções
- Na academia
  - Depuração delta
  - *Slicing* estático/dinâmico
  - Depuração baseada em cobertura

CodeForest

- Conjunto de casos de teste automatizados utilizando a biblioteca junit é executado.
- Informações de cobertura de código são coletadas:
  - comandos executados por cada caso de teste.
- Depuração baseada em cobertura:
  - atribui um valor de suspeição a nós (conjuntos de comandos executados sequencialmente), métodos e classes.

CodeForest

- Valor de suspeição é baseado na frequência de execução de um nó por casos de teste de sucesso e, principalmente, por casos de teste de falha.
- Atribui-se um *valor de suspeição* entre 0 e 1

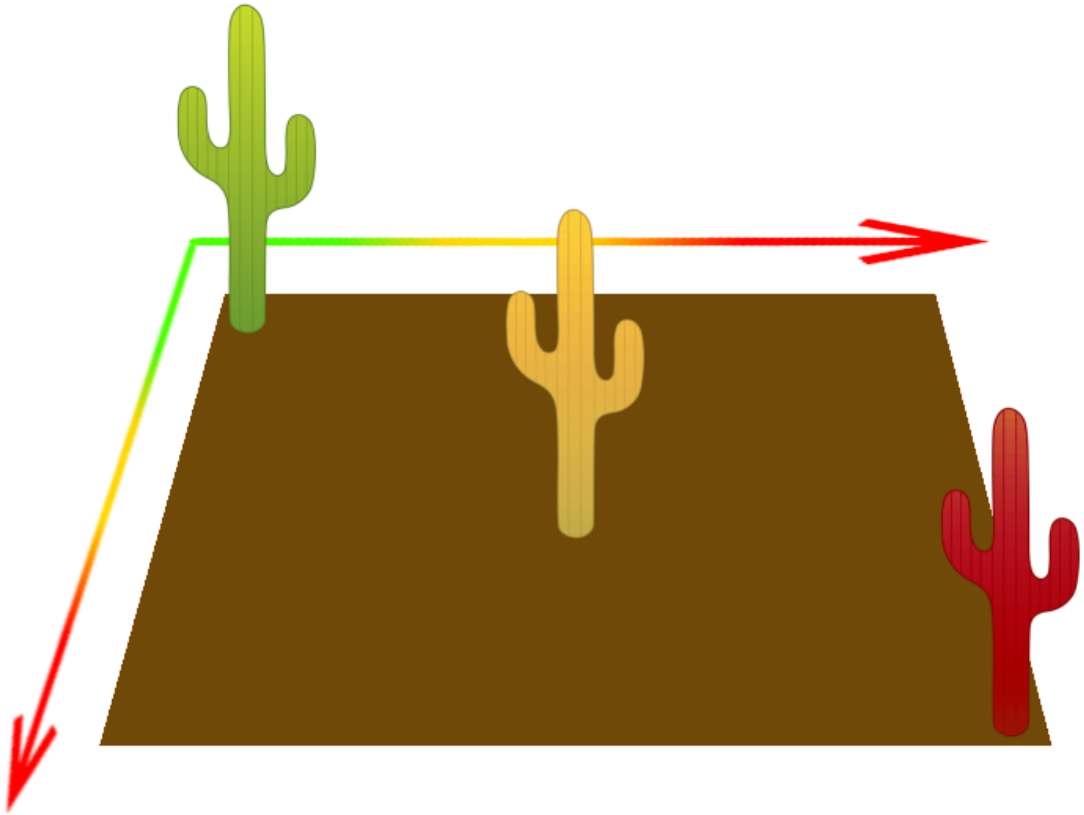
  para cada nó (conjunto de comandos executados sequencialmente), método e classe
- A suspeição de cada elemento (nó, método e classe) é o insumo para a visualização.

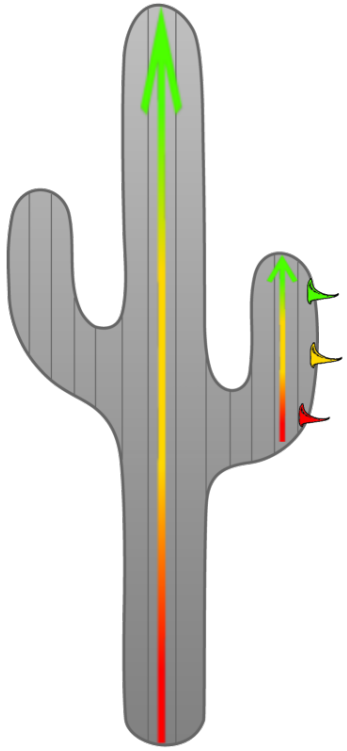
CodeForest

- Metáfora na qual
  - Árvores representam classes;
  - Galhos representam métodos;
  - Folhas representam linhas de código.

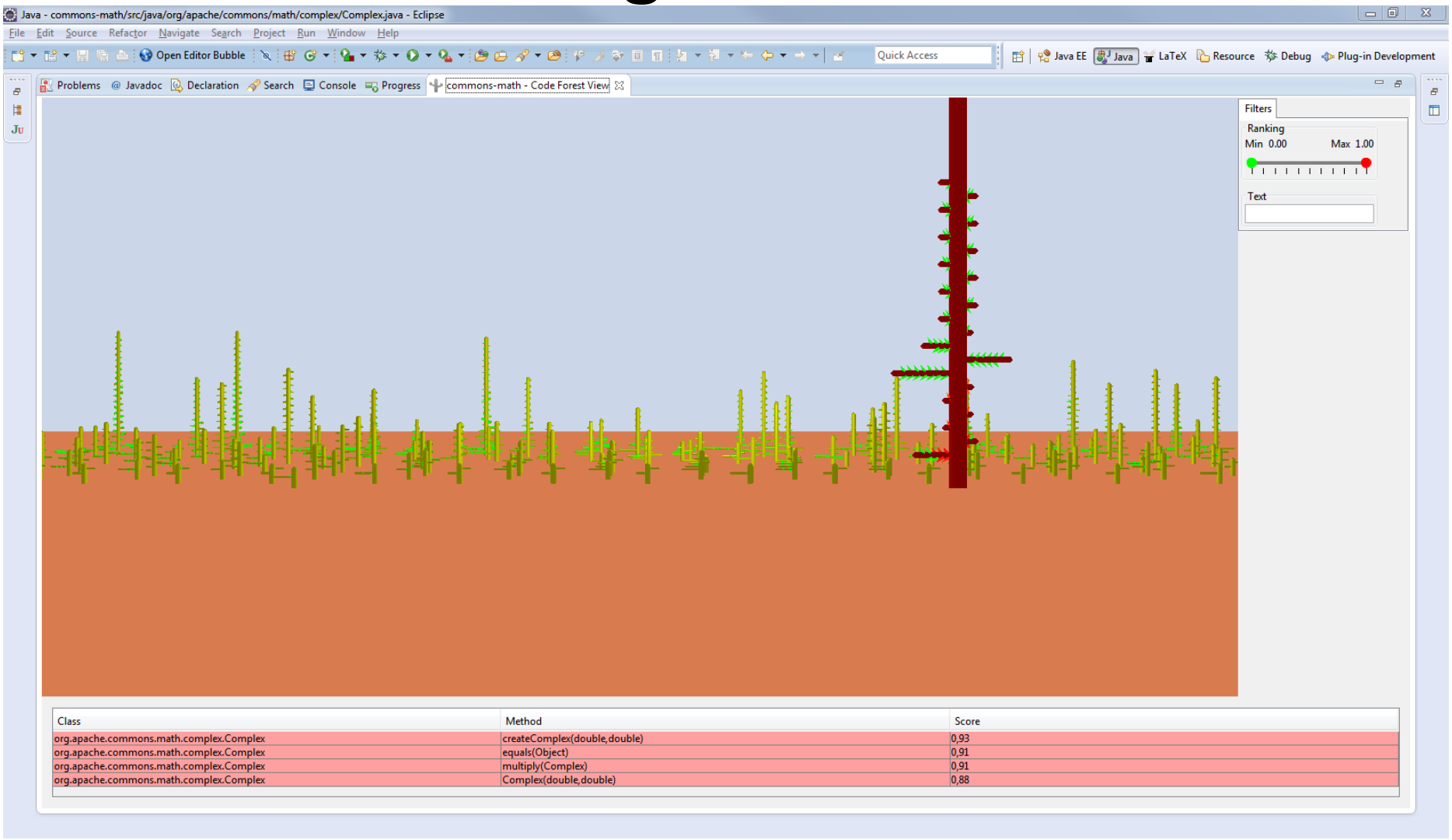
- Desafio
  - Exibir uma quantidade muito grande de dados em um espaço limitado.

Metáfora CodeForest

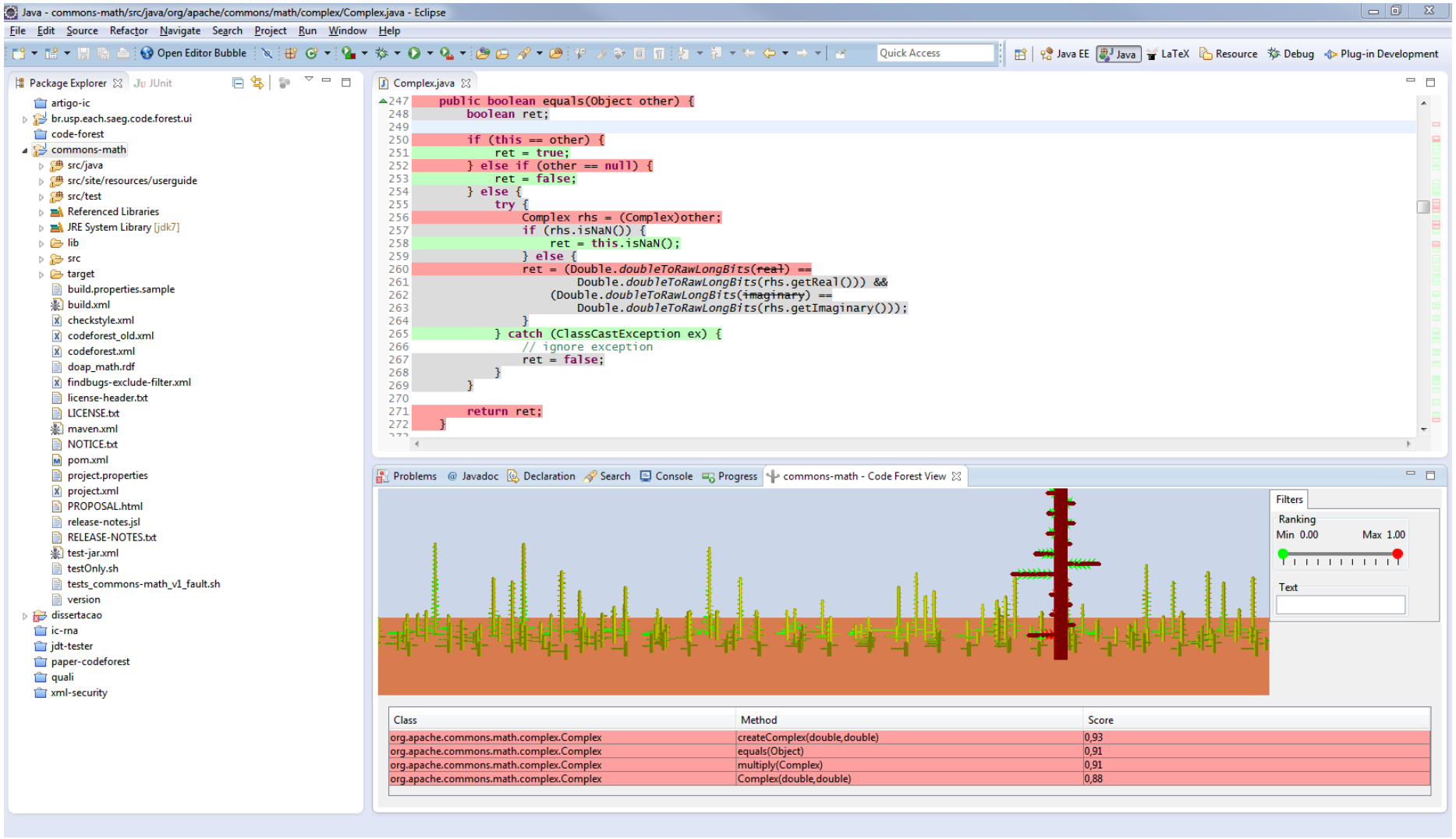




CodeForest Plug-in



CodeForest Plug-in



Manual de Instruções

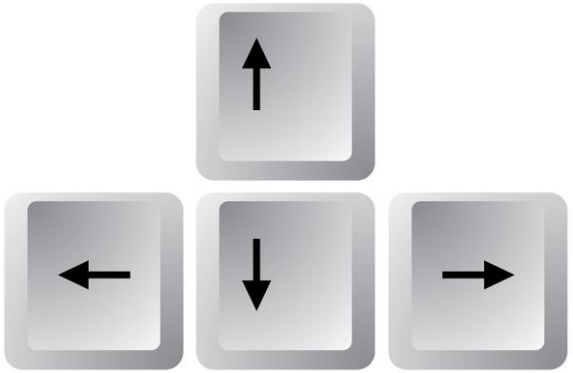
- Fazer a análise do projeto
  - Botão Direito > Code Forest > Perform analysis
- Linhas de código pintadas de acordo com o grau de suspeição
  - Alto
  - Médio Alto
  - Médio Baixo
  - Baixo
  - N/A
- Visualização da Floresta
  - Botão Direito > Code Forest > View as Code Forest

Manual de Instruções

- Trimmer
  - Ajusta o filtro de suspeição para os valores mínimos e máximos escolhidos.
- Filtro de texto
  - Deixa visíveis somente os elementos que possuem o texto digitado;
  - Independente de caixa alta ou baixa.
- Utilizados em conjunto "AND"
- Roteiro de busca baseado nos métodos mais suspeitos.

Manual de Instruções

- Movimentação para frente, para trás e para os lados com as setas.
- Controles especiais
  - Command (Mac) / Control (Win) - Escada
  - Shift - Pescoço
- Reset da view
  - tecla =



Manual de Instruções - Ferramentas

- Clique (esquerdo) em qualquer elemento da Floresta abre o elemento escolhido
  - Tronco - 1a linha da classe;
  - Galho - 1a linha do método;
  - Folha - linha de código.

- Clique (esquerdo) em qualquer item do script
  - Filtro de texto pelo nome do método;
  - Trimmer entre o grau de suspeição do método (máx) e um limiar mínimo.

Commons Math

- commons.apache.org/proper/commons-math
  - Biblioteca matemática leve, contida com componentes estatísticos.
  - Contém componentes pequenos e facilmente integráveis.
- org.apache.commons.math3.stat.StatUtils
  - mean(double[] values)
  - variance(double[] values)
  - sumSq(double[] values)

XStream

- xstream.codehaus.org
  - Uma biblioteca simples para serializar objetos para XML e de XML para objetos.
- XStream xstream = new XStream();
  - Person joe = new Person("Joe");
  - String xml = xstream.toXML(joe);
  - Person newJoe = (Person) xstream.fromXML(xml);

Demo da CodeForest

- CodeForest é baseada em informação de cobertura coletada a partir de testes automatizados com o JUnit.
- Está informação já foi coletada por meio das ferramentas InSS (*Instrumentation Strategies Simulator*) e DCI (Depuração baseada em Cobertura de Integração).
- O programador pode, se quiser, utilizar o Plugin da CodeForest para localizar o defeito.
- Ele é livre para usar a CodeForest com os recursos para depuração do Eclipse ou somente esses recursos, se assim o desejar.

Conclusões

- O que está sendo avaliado é o processo de depuração e não o participante do experimento.
- Você é livre para abandonar o experimento no momento que quiser, sem qualquer prejuízo.
- As informações coletadas no experimento poderão ser utilizadas para publicações científicas, mas seu anonimato é garantido.