Teste de Software Orientado a Objeto

Ricardo Argenton Ramos

O que diferencia teste de software OO de testes Convencionais?

- Técnicas e abordagens são normalmente semelhantes, porém novos desafios são apresentados (Ex. herança e polimorfismo);
- Processos OO iterativos e incrementais nos dá a oportunidade de melhorar nossos processos de teste convencionais:
  - Mudar a Visão de Teste como um “Mal necessário” – Teste pode contribuir para se desenvolver o produto certo deste o início
  - Como iniciar a atividade de teste o quanto antes e contextualizá-la dentro de um processo de desenvolvimento
  - Como efetivamente escolher o que precisa ser testado e implementar de uma forma eficiente.
- Mudanças na forma como desenvolvemos software provoca mudanças na forma como testamos (metas, formatos, etc)

Impacto de OO na Testabilidade

- Impacto do Encapsulamento no Teste
  - Limita a controlabilidade e a observabilidade.
  - Teste requer...
    - Um relatório completo do estado concreto e abstrato de um objeto.
    - A possibilidade de alterar esse estado facilmente.
  - Encapsulamento
    - Falta de visibilidade dos estados.
      - Além do comportamento, objetos também encapsulam estados.
    - Dificuldade na inicialização dos itens de dados.
    - Dificuldade na chamada dos métodos.

Impacto de OO na Testabilidade

- Impacto da Herança no Teste
  - Quando retestar???
  - A herança pode levar à falsa conclusão de que subclasses que herdam características de superclasses não precisam ser testadas, reduzindo assim o esforço com os testes.
  - Na verdade, a herança define um novo contexto para os métodos.
    - O comportamento dos métodos herdados pode alterar-se em virtude dos outros métodos chamados dentro da classe herdada.

Impacto de OO na Testabilidade

- Impacto da Herança no Teste
  - Um método testado em uma superclasse precisa ser retestado ao ser reutilizado em uma subclasse.
  - Mesmo que um método seja herdado integralmente de uma superclasse sem nenhuma modicação (herança estrita), este deverá ser retestado no contexto da subclasse.
  - Herança Múltipla
    - Dificulta ainda mais a realização dos testes

Impacto de OO na Testabilidade

- Impacto do Polimorfismo e do Acoplamento Dinâmico no Teste
  - Cada possibilidade de acoplamento de uma mensagem polimórfica é uma computação única, e requer testes separados.
    - O fato de diversos acoplamentos polimórficos trabalharem corretamente não garante que todos irão trabalhar.
  - Indecidibilidade ao Teste
    - Acoplamento Dinâmico: antecipar acoplamentos.

Teste de Software

- Processo para descobrir a existência de defeitos em um software.
- Um defeito pode ser introduzido em qualquer fase de desenvolvimento ou manutenção como resultado de um ou mais “bugs” – imprecisão, desentendimentos, omissões e direcionamento a soluções particulares, inconsistências, incompletude.
- *Debugging* – Processo de encontrar bugs associados a um defeito.
- Objetivos:
  - Mostrar que a aplicação faz o esperado
  - Mostrar que a aplicação não faz mais do que o esperado
- **Teste é um processo referencial:** É necessário existir uma definição precisa do que se quer verificar e quais os resultados esperados.

Teste de Software – Cont.

- Todas as representações de um software podem e devem ser testados.
- Teste não implica em garantia de qualidade: é necessário ter um conjunto de métodos para a prevenção e remoção de defeitos.
- Nenhuma quantidade absurda de testes pode melhorar a qualidade de um software: teste ajuda a identificar problemas que poderíamos ter evitado.
- Garantia de qualidade requer processos além da execução de testes. Entretanto, processos de planejamento e especificação de testes executados mais cedo podem contribuir nesta direção.

Fases de Teste

- As fases de teste são independentes do paradigma de desenvolvimento de software utilizado, seja ele procedimental ou orientado a objetos.
  - **Teste de Unidade**
  - **Teste de Integração**
  - **Teste de Sistema**
  - **Teste de Aceitação**

Como Fica os Testes OO

- Teste de Classe (substitui teste de unidade clássico)
- Teste de Interação (substitui teste de integração clássico)
- Teste de Sistema (e subsistema)

Teste de Unidade

- Teste de Unidade
  - Um método...
  - Uma classe...
  - Um grupo de classes…
  - Método: menor unidade a ser testada.
  - Classe à qual o método pertence...
    - Driver do método
      - Sem a classe não é possível executar um método.
  - Teste Intra-Método
    - Testa um método específico de uma classe.

Teste de Integração

- Teste de Integração
  - Classe: engloba um conjunto de atributos e métodos que manipulam tais atributos.
    - Métodos da mesma classe podem interagir entre si para desempenhar funções específicas.
  - Teste Inter-Método
    - Testa a integração entre métodos.

Teste de Integração

- Teste Intra-Classe
  - Testa as interações entre métodos públicos fazendo chamada a esses métodos em diferentes seqüências.
  - Identifica possíveis seqüências de ativação de métodos inválidas que levem o objeto a um estado inconsistente.

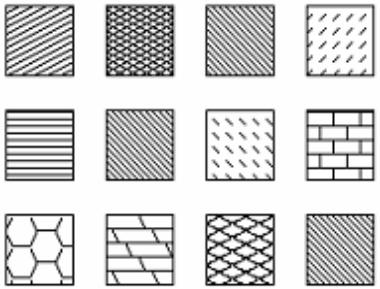
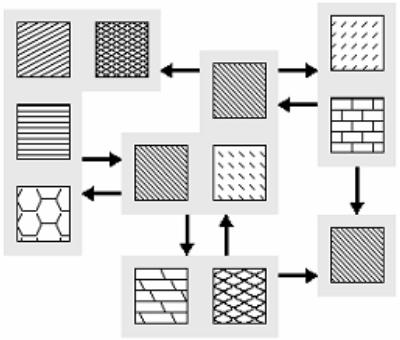
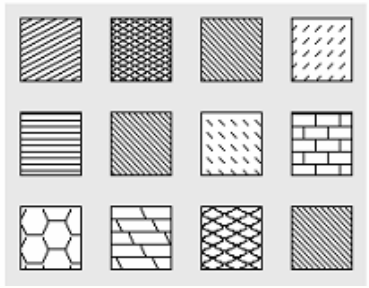
Teste de Integração

- Teste Inter-Classe
  - Testa as interações entre métodos públicos.
  - Não apenas os métodos em uma única classe, mas também aqueles em classes distintas.

Teste de Sistema

- Considera o software OO como um todo.
- Em geral, utilizam-se critérios funcionais.
  - Não apresenta diferenças fundamentais entre programas procedimentais e OO.

Fases de Teste

Teste Procedimental		Teste Orientado a Objetos
	Teste de Unidade	
Sub-rotina ou função		Método
	Teste de Integração	
Duas ou mais unidades		Classe
	Teste de Sistema	
Toda aplicação		Toda aplicação

Critérios Funcionais

- Podem ser aplicados tanto no teste de programas procedimentais como no teste de programas OO.

- Particionamento de Equivalência
- Análise do Valor Limite

- Método de Partição-Categoria
  - Categorias representam as principais características do domínio de entrada da função em teste.
    - Definir categorias, e particioná-las em classes de equivalência de entradas (choices).

Particionamento de Equivalência

- Para aplicar este critério de teste na aplicação a ser testada, deve-se seguir o seguinte roteiro:
  1. Estabelecer as classes válidas (valores permitidos para o atributo) e as inválidas (valores proibidos), para cada atributo do *software*.
  2. Verificar se o sistema permite a gravação no banco de dados de variáveis classificadas em ambas as classes.

Exemplo

Variável de Entrada	Classes de Equivalência válidas	Classes de Equivalência inválidas	Elementos Requeridos	Saída Esperada	Saída Obtida
Peso da Criança (P)	P>0	P<=0	1-2,4kg 2- 0kg 3- -2,5kg	1-Cad 2- NCad 3-NCad	1- OK 2-ERRO 3-ERRO
Data de Nascimento (DN)	Data_nasc <= sysdate	Data_nasc > sysdate	4- 05/04/2009 5- 06/04/2009 6- 16/04/2009	4- Cad 5- Cad 6- NCad	4- OK 5- OK 6-ERRO

Exemplo de Elementos Requeridos do critério Particionamento em Classes de Equivalência para o Cadastro de Crianças

Análise do Valor Limite

- Em vez de se concentrar somente nas condições de entrada, a análise de valor limite deriva os casos de teste também do domínio de saída. Para aplicar este critério, deve-se seguir o seguinte roteiro:
  1. Estabelecer as classes válidas e as inválidas para cada atributo do *software* (caso o critério descrito anteriormente tenha sido aplicado, essa etapa não precisa ser repetida).
  2. Definir os limites dos atributos, ou seja, selecionar dados com um valor inferior, igual e outro superior ao limite.
  3. Verificar se o sistema permite salvar no banco essas variáveis.

Exemplo

Variável de Entrada	Entrada	Saída Esperada	Saída Obtida
Data de Nascimento (DN)	1- 00 2- 01 3- 02 4- 30 5- 31 6- 32 7-11 8-12 9-13 10- ano atual-1 11- ano atual 12-ano atual+1	1- NCAD 2- CAD 3- CAD 4- CAD 5- CAD 6- NCAD 7- CAD 8- CAD 9- NCAD 10- CAD 11- CAD 12- NCAD	1- ERRO 2- OK 3- OK 4- OK 5- OK 6- ERRO 7- OK 8- OK 9- ERRO 10- OK 11- OK 12- ERRO
Peso da Criança (P)	13- P=-1 14- P=0 15- P=1	13- NCAD 14- NCAD 15-CAD	13- ERRO 14- ERRO 15- OK

Exemplo de elemento requerido do critério Análise dos Valores Limites para o Cadastro de Crianças

Critérios Estruturais

- Teste de Fluxo de Dados em Classes
  (Harrold & Rothermel)
  - Abordagem para testar métodos individuais e as interações entre os métodos dentro de mesma classe.
  - Nova representação para testar os métodos que são acessíveis fora da classe e podem ser utilizados por outras classes.
    - Grafo de Fluxo de Controle de Classe (GFCC)
  - Diferentes níveis de teste são considerados.
    - Teste Intra-Método
    - Teste Inter-Método
    - Teste Intra-Classe
    - Teste Inter-Classe

Critérios Estruturais

- Estratégia de Teste Incremental Hierárquica *(Perry & Kaiser)*
  - Identificar quais métodos herdados necessitam de novos casos de teste para serem testados.
  - Identificar quais métodos podem ser retestados aproveitando os casos de teste elaborados para o teste da superclasse.
    - Reduzir os custos no teste das subclasses.
  - Teste da superclasse
    - Teste Intra-Método
      - Construção do GFC de cada método.
  - Teste Intra-Classe e Inter-Classe
      - Construção do GFCC de cada classe.
  - Teste da subclasse

Ferramentas Para Testes OO

- JUnit (Teste unitário para programas Java) - Testes sobre o código fonte. (http://www.junit.org/)

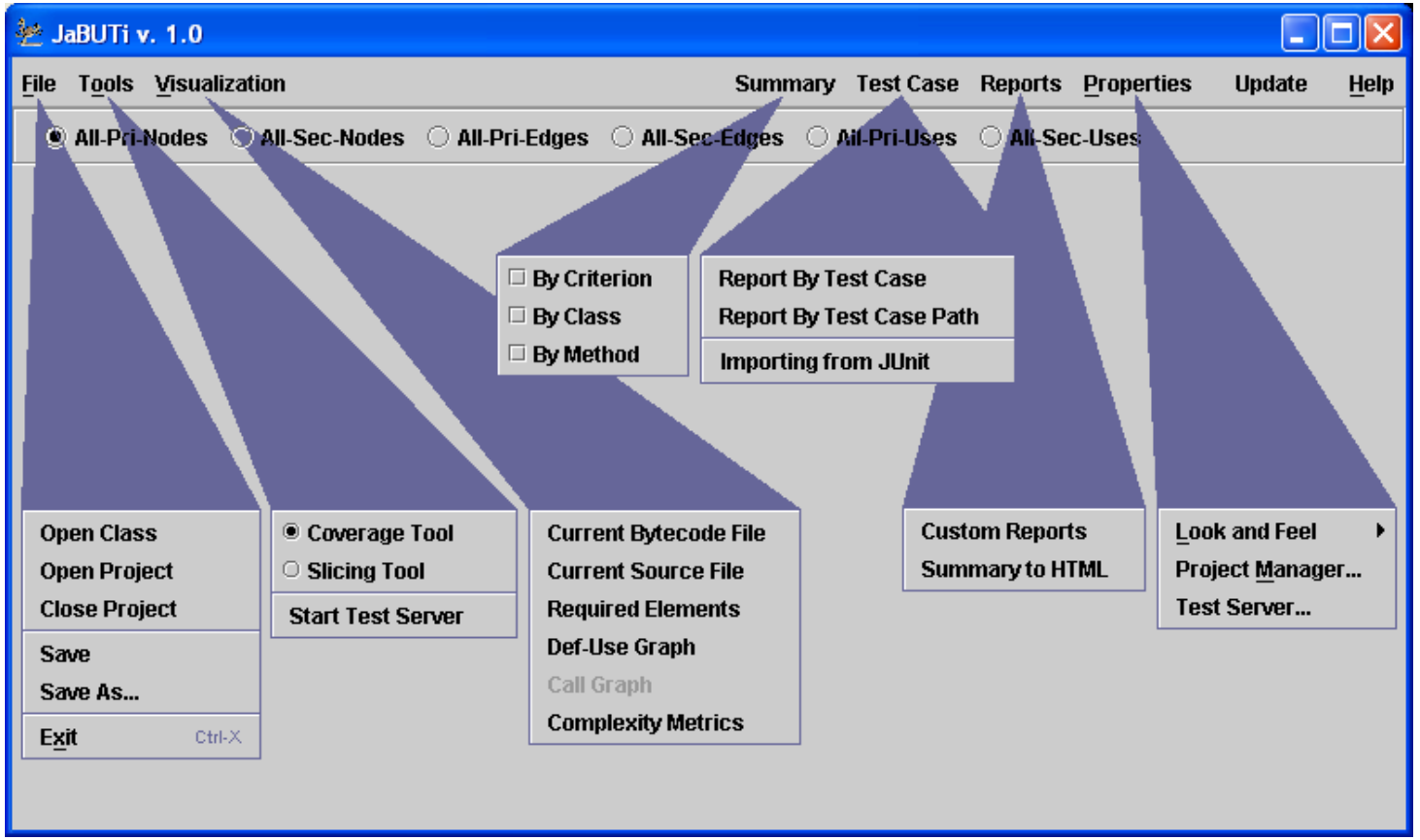
- JABUTI (*Java Bytecode Understanding and Testing*) - Os testes são feitos sobre o programa objeto, ou seja, sobre o *bytecode* Java e não sobre o programa fonte. (http://ccsl.ime.usp.br/pt-br/project/jabuti)

Ferramenta JABUTI

- Ferramenta de suporte ao teste estrutural para programas Java.
- Estão implementados critérios baseados em fluxo de controle e critérios baseados em fluxo de dados.
- Java Bytecode
- Características
  - Projeto de teste
  - Importação de casos de teste
  - Inserção e remoção de casos de teste dinamicamente
  - Casos de teste podem ser habilitados ou desabilitados
  - Geração de relatórios (resumo, por critério, por classe, por método, por caso de teste)

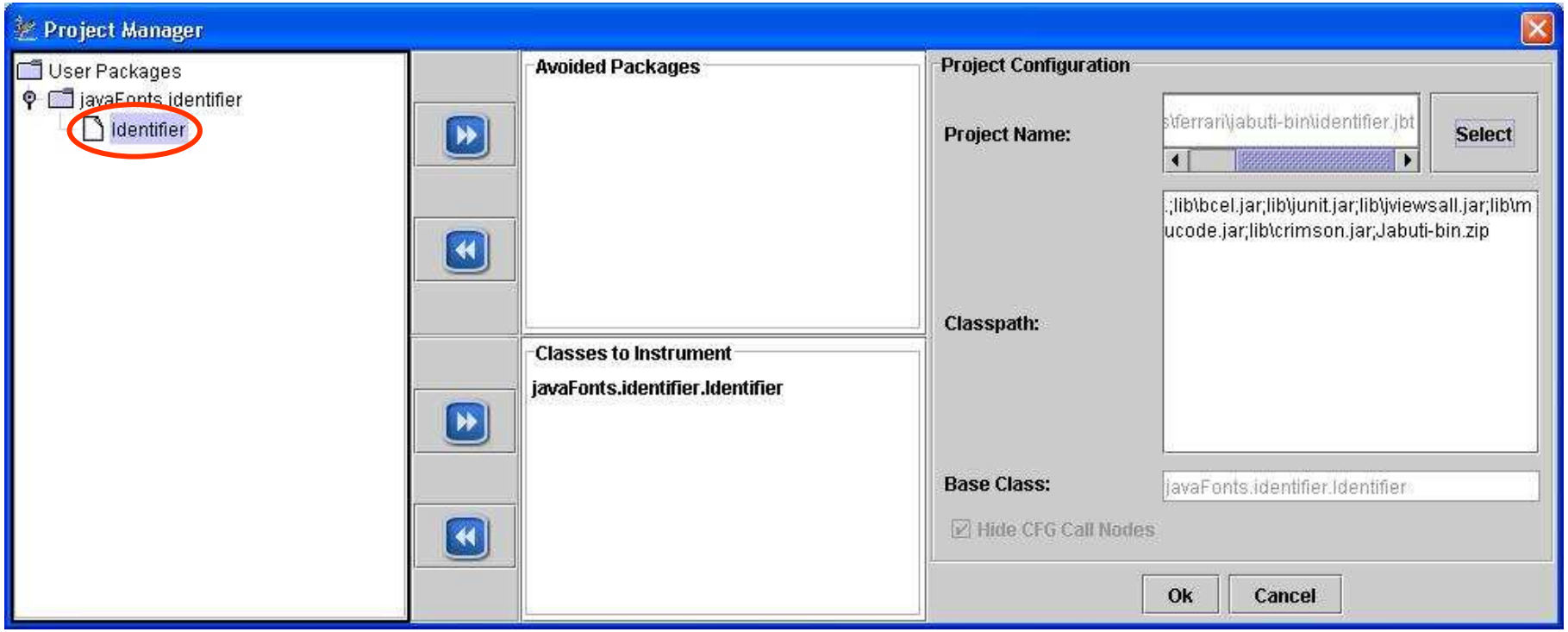
Ferramenta JaBUTi

- **Tela Principal**



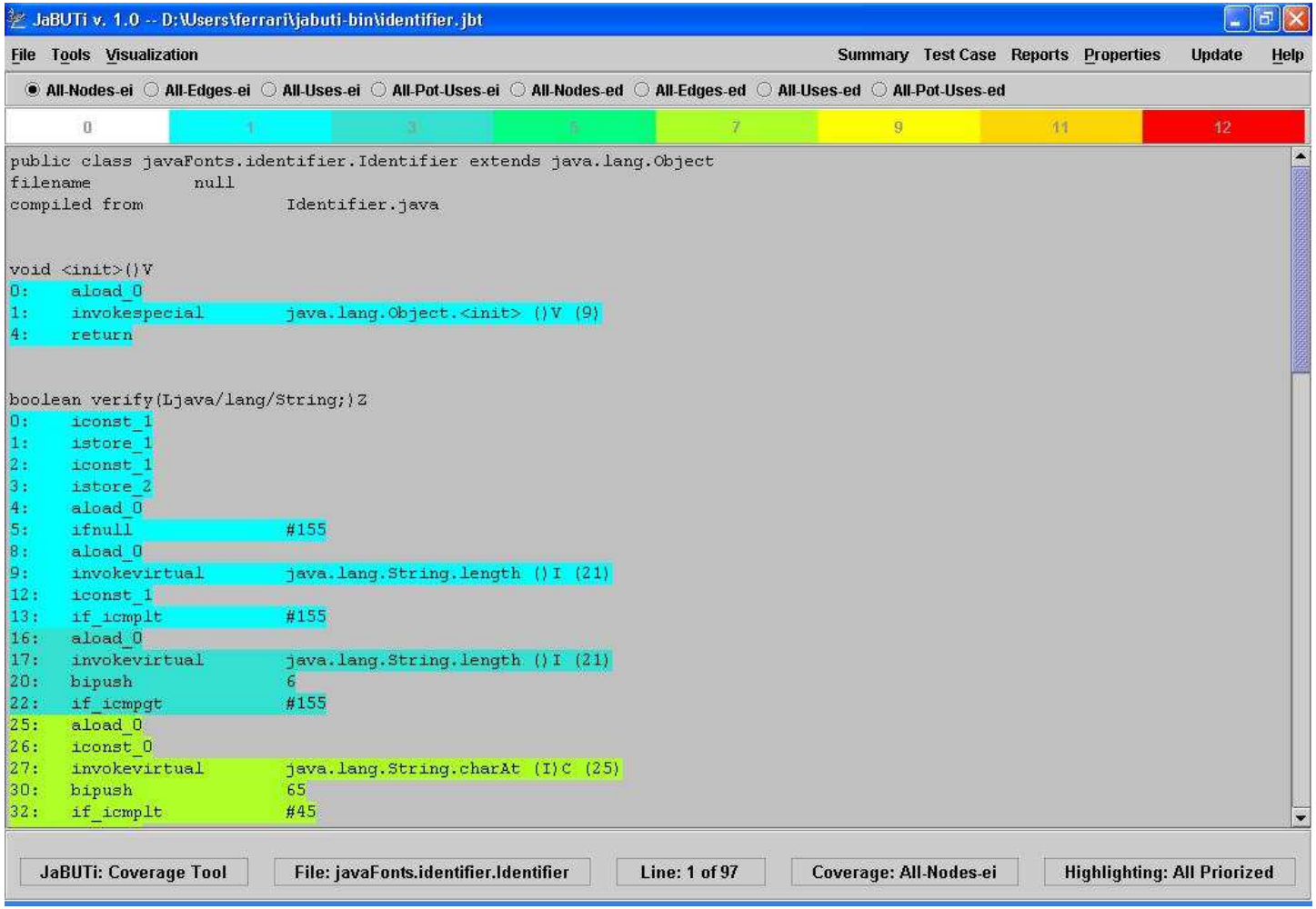
Ferramenta JaBUTi

- **Projeto de Teste**
  - Seleção da classe base



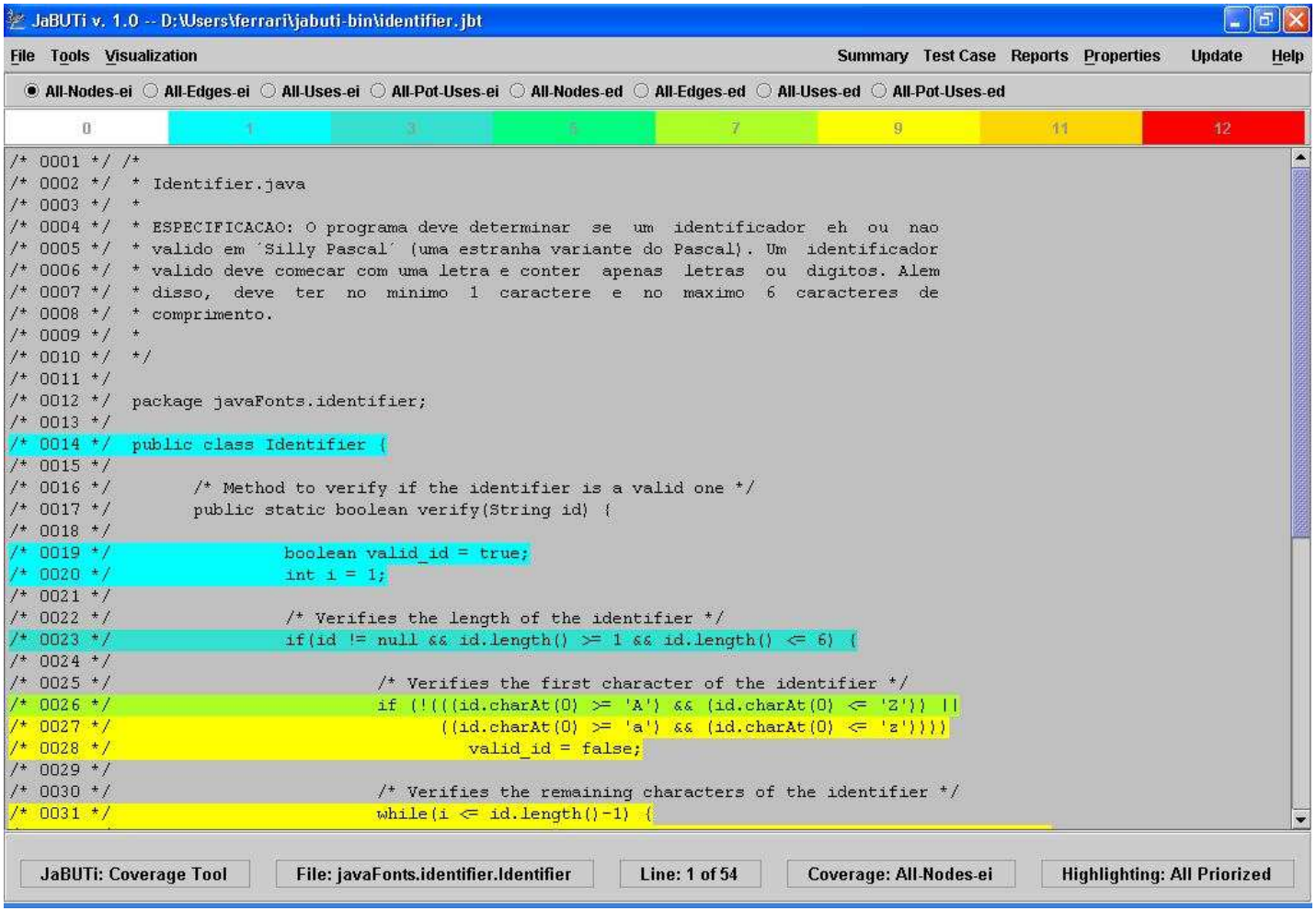
Ferramenta JaBUTi

- **Visualização do Bytecode**



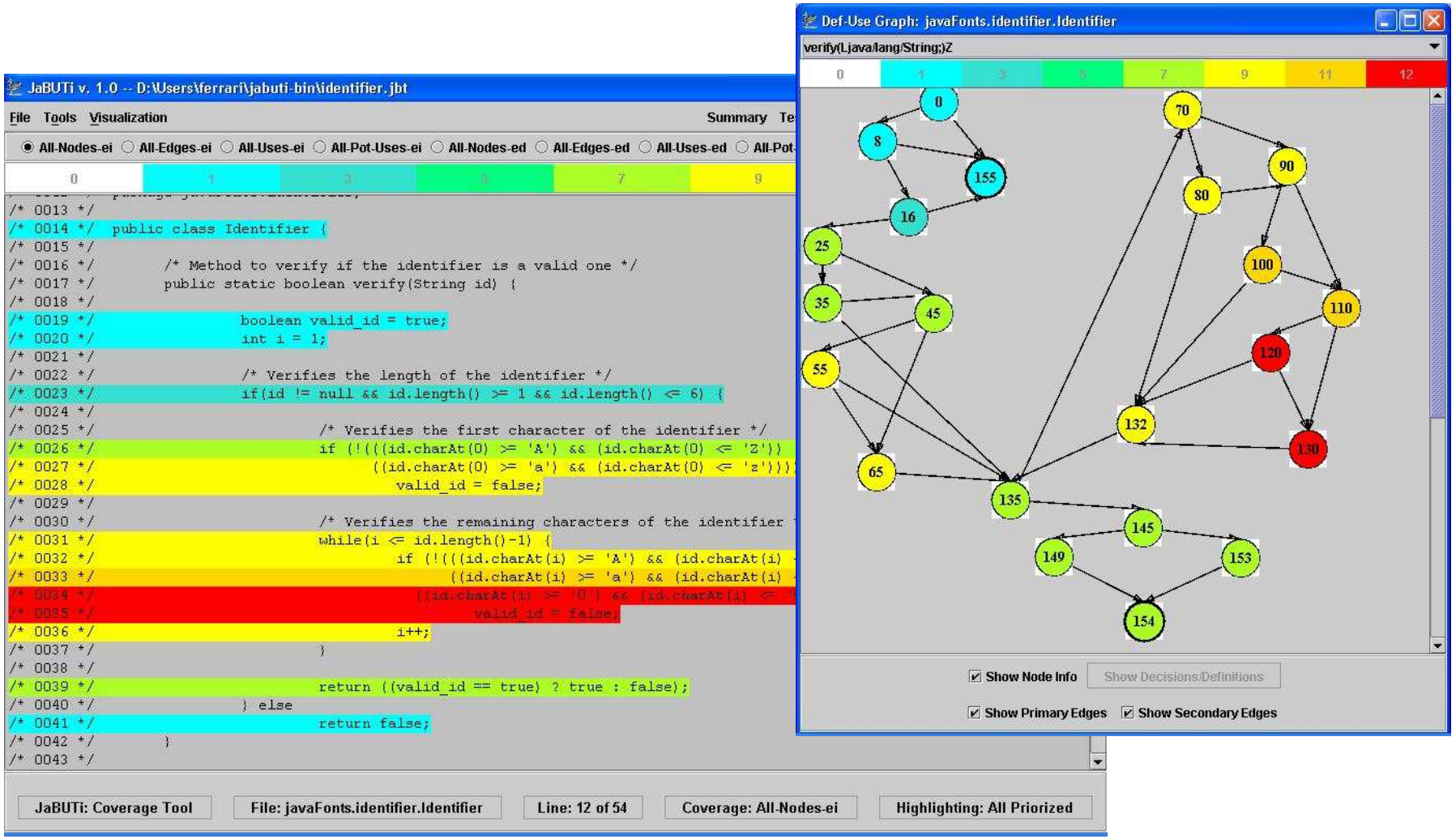
Ferramenta JaBUTi

- **Visualização do Código-Fonte**



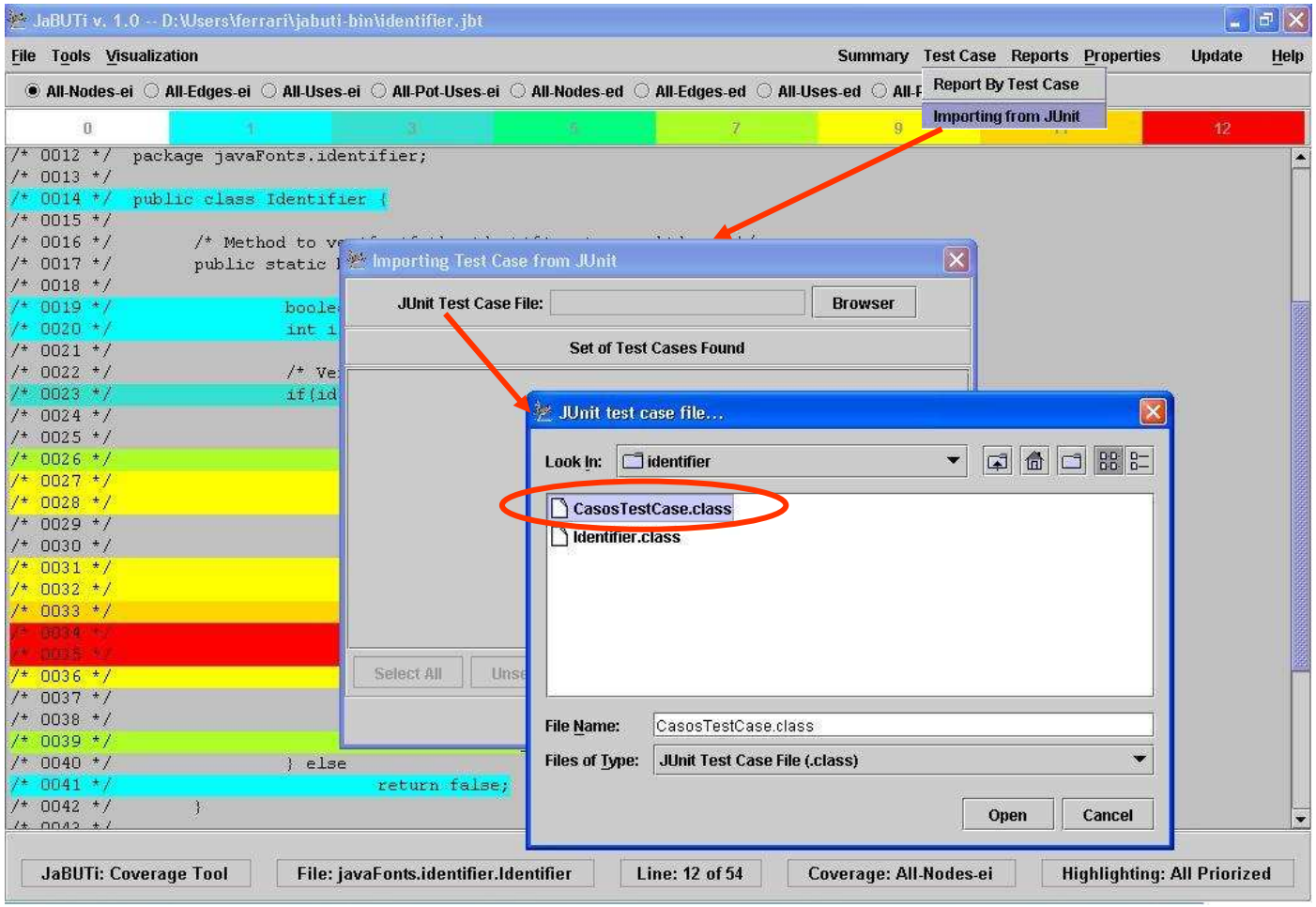
Ferramenta JaBUTi

- **Visualização do Código-Fonte e do GFC**



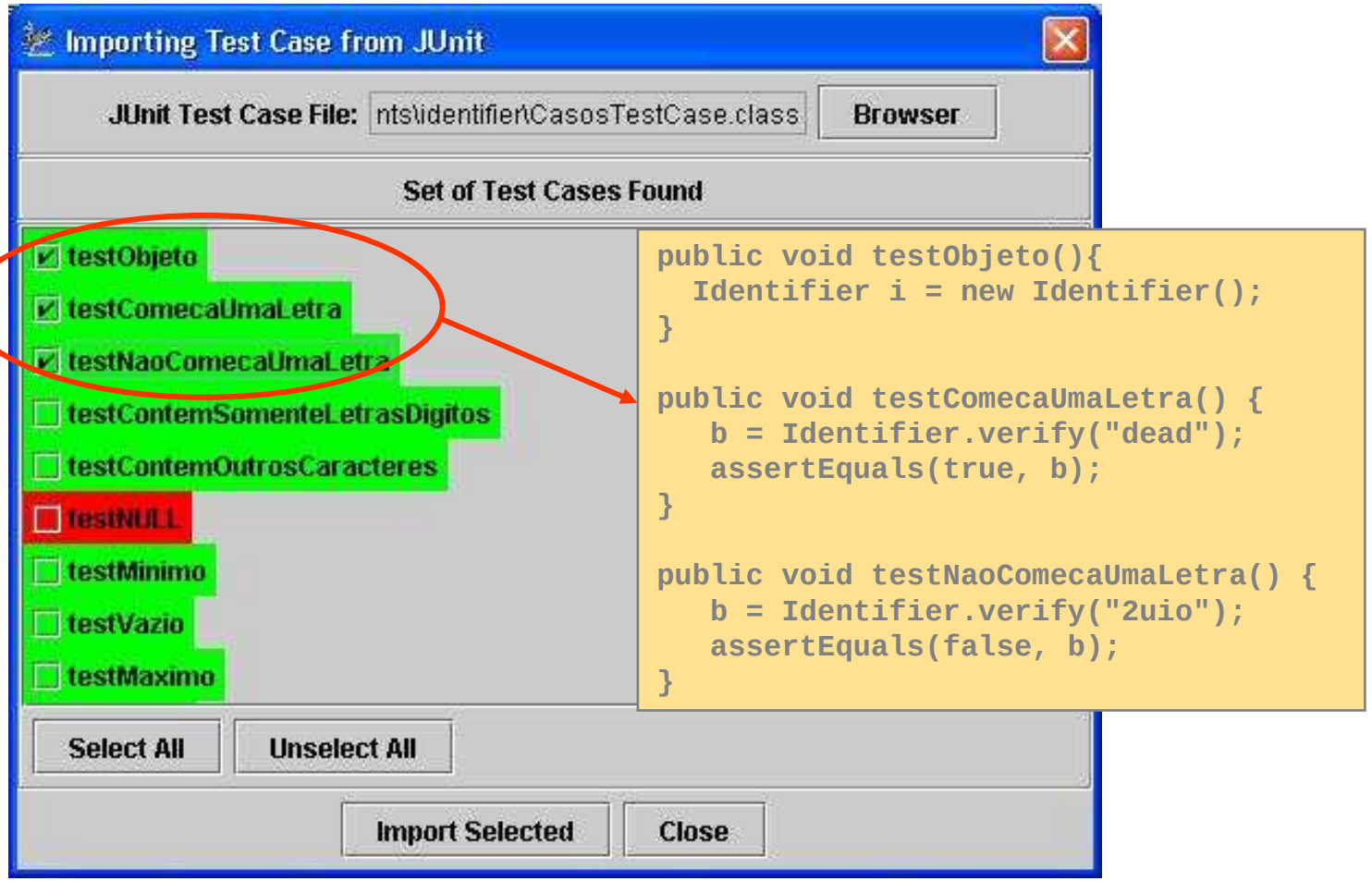
Ferramenta JaBUTi

- **Seleção da Classe de Casos de Teste**



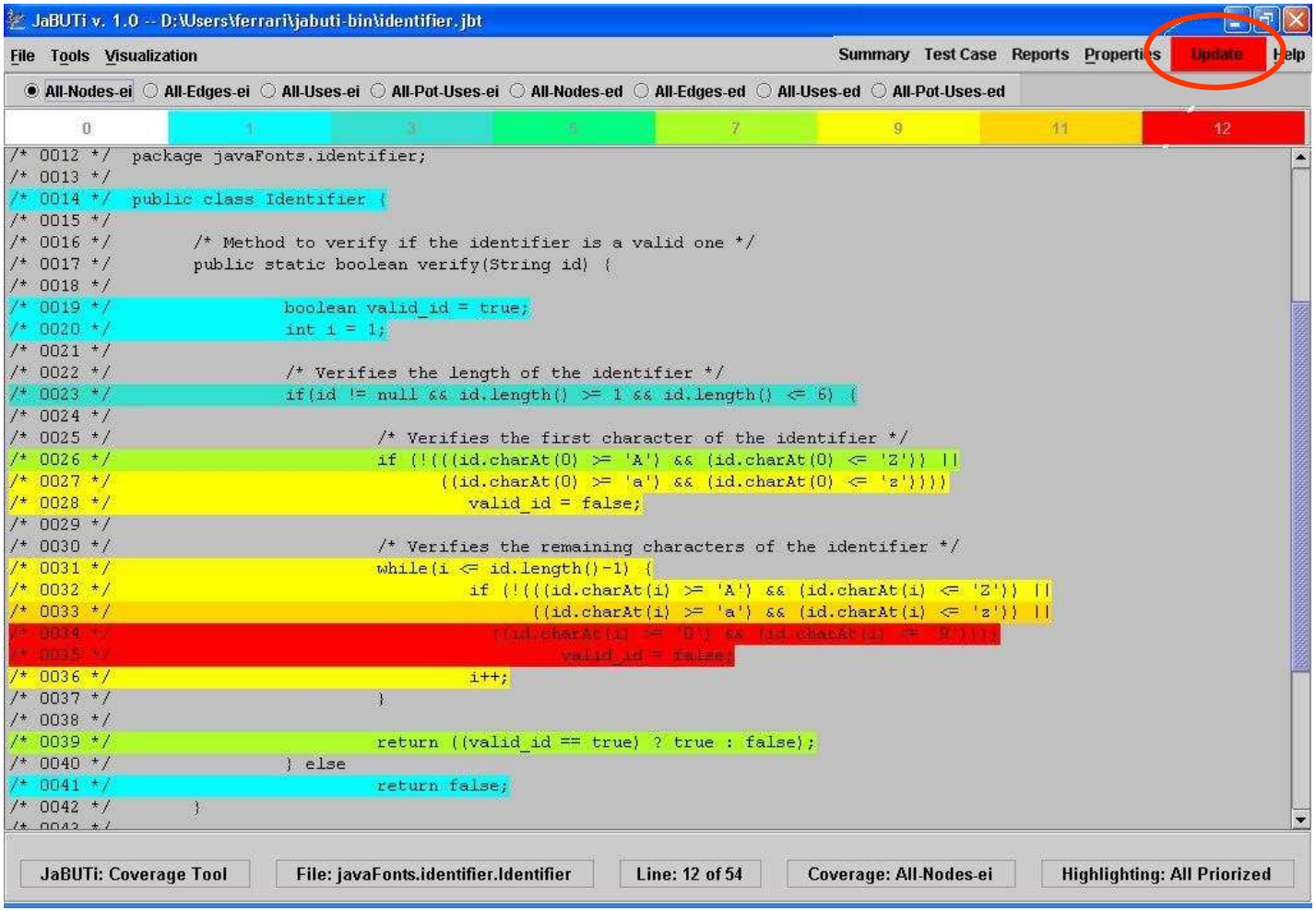
Ferramenta JaBUTi

- **Seleção e Importação de Casos de Teste**



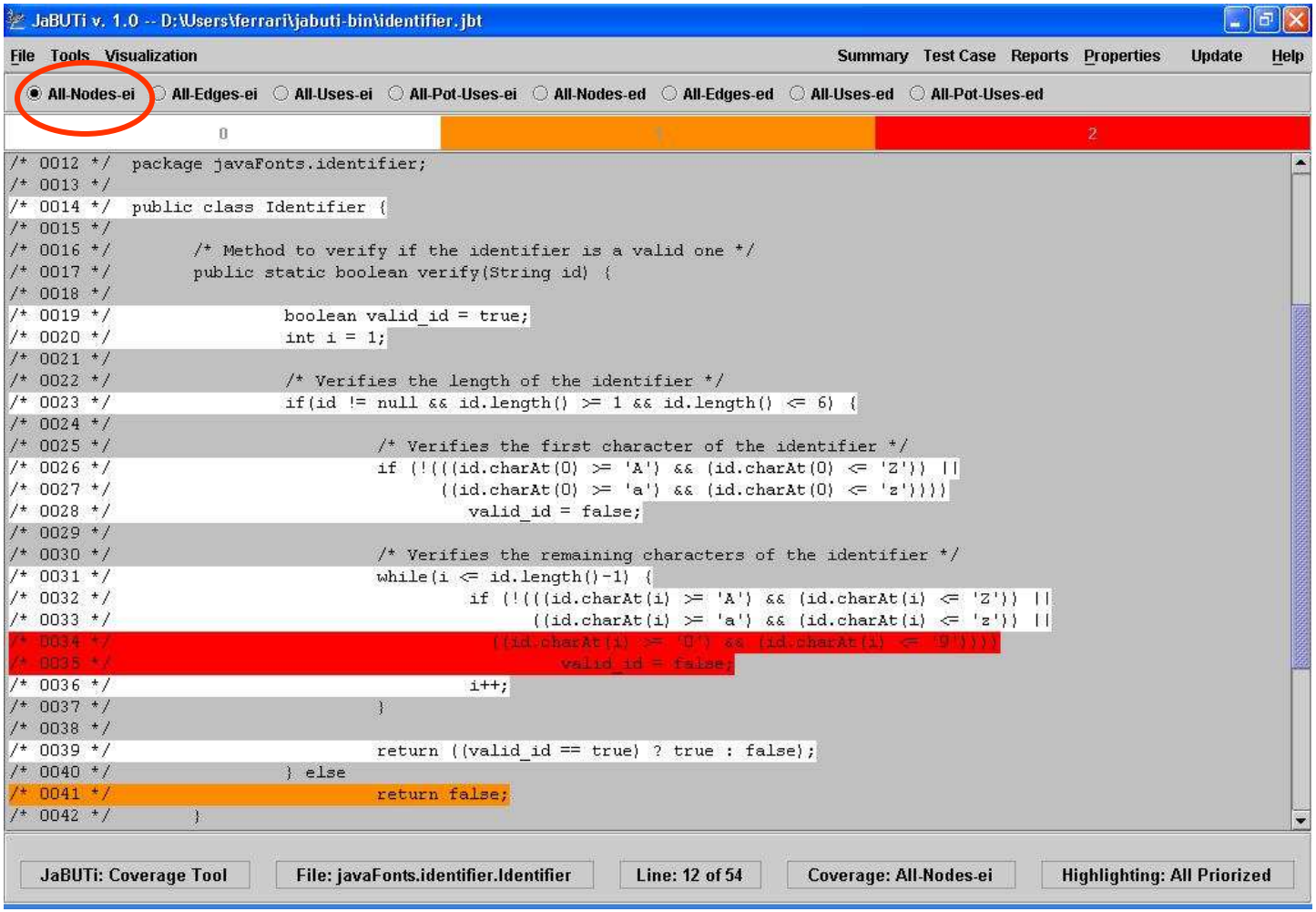
Ferramenta JaBUTi

- **Atualização da Cobertura**



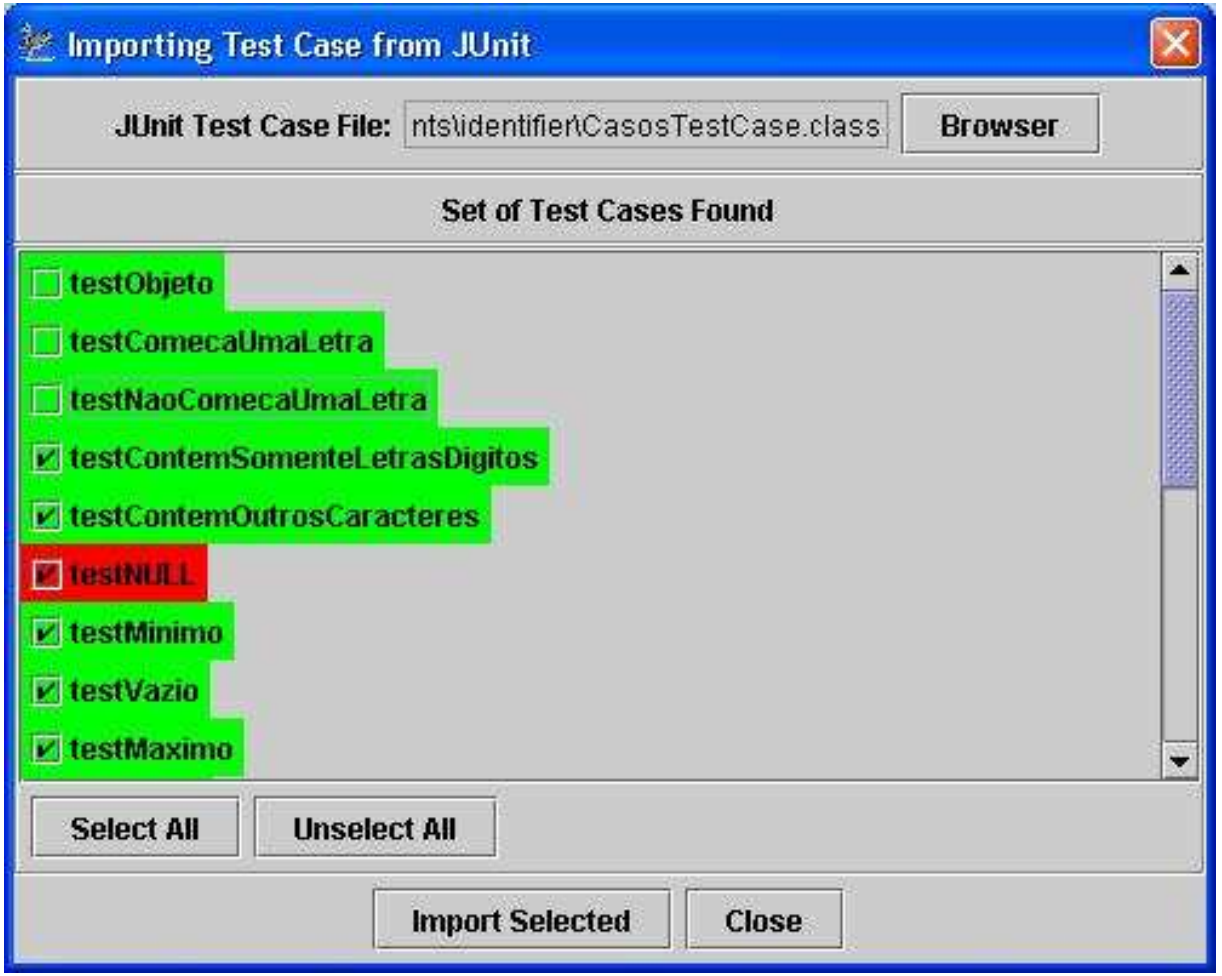
Ferramenta JaBUTi

- **Cobertura Parcial (Todos-Nós$_{ei}$)**



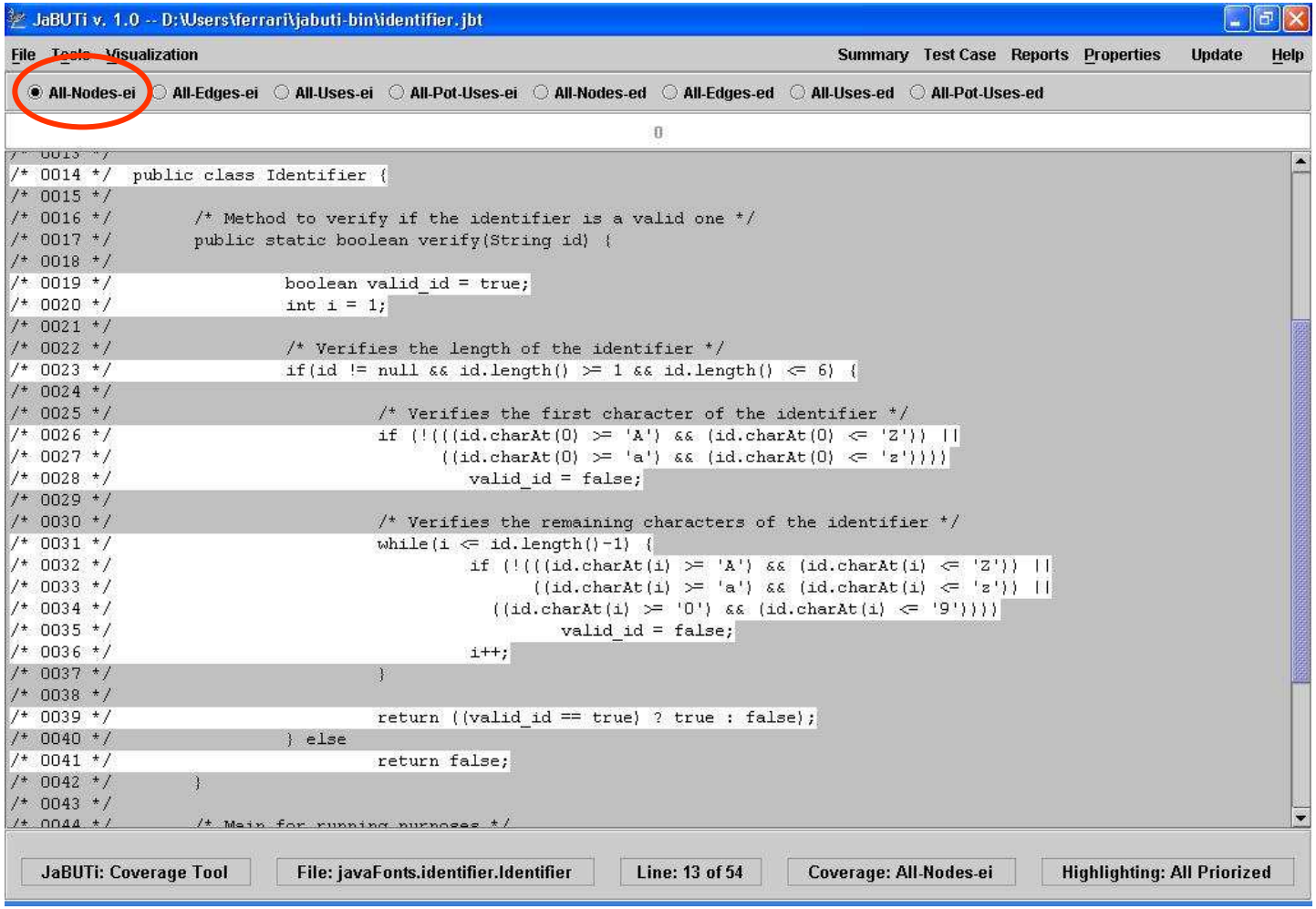
Ferramenta JaBUTi

- **Seleção e Importação de Casos de Teste**



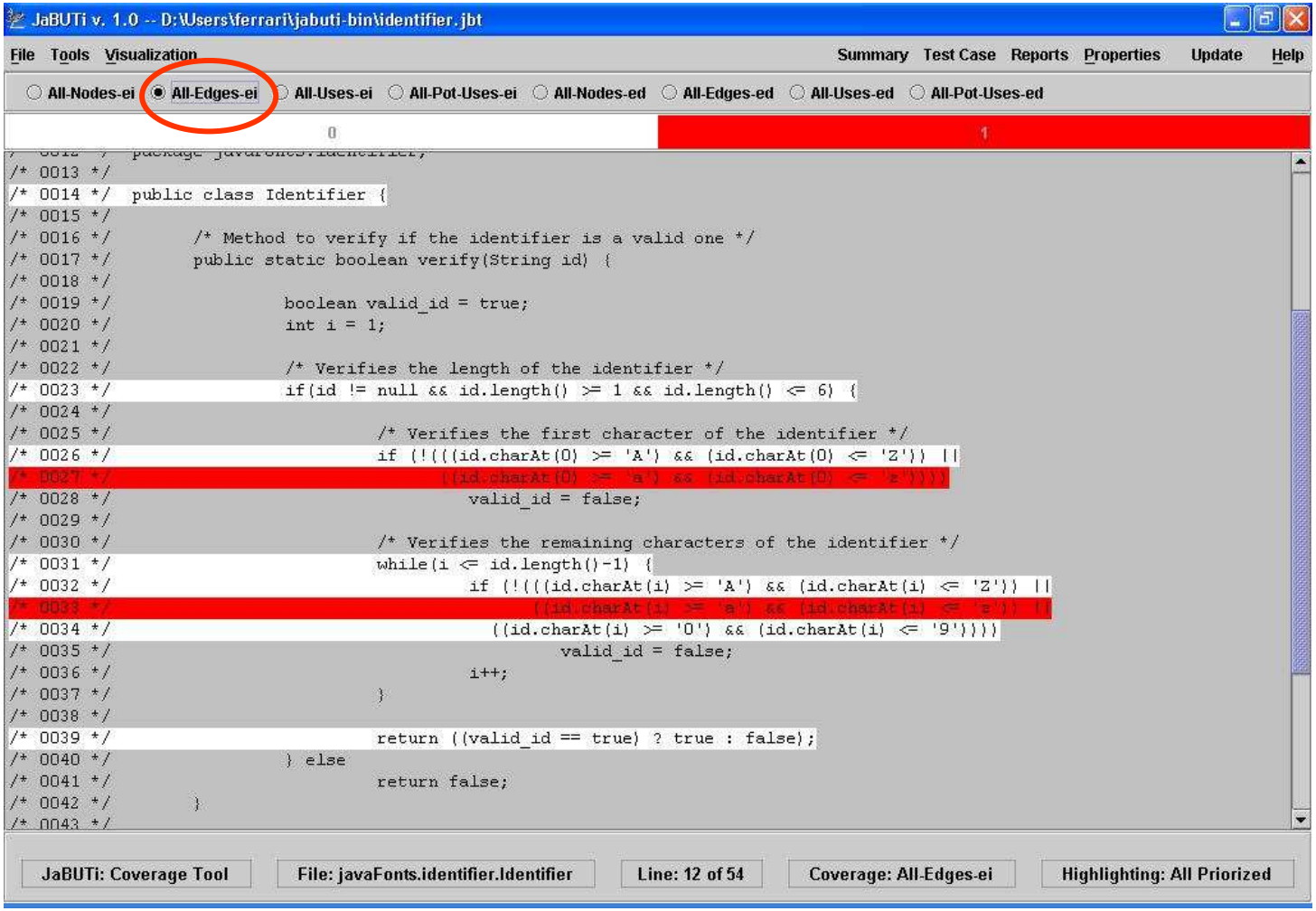
Ferramenta JaBUTi

- **Cobertura Total (Todos-Nós$_{ei}$)**



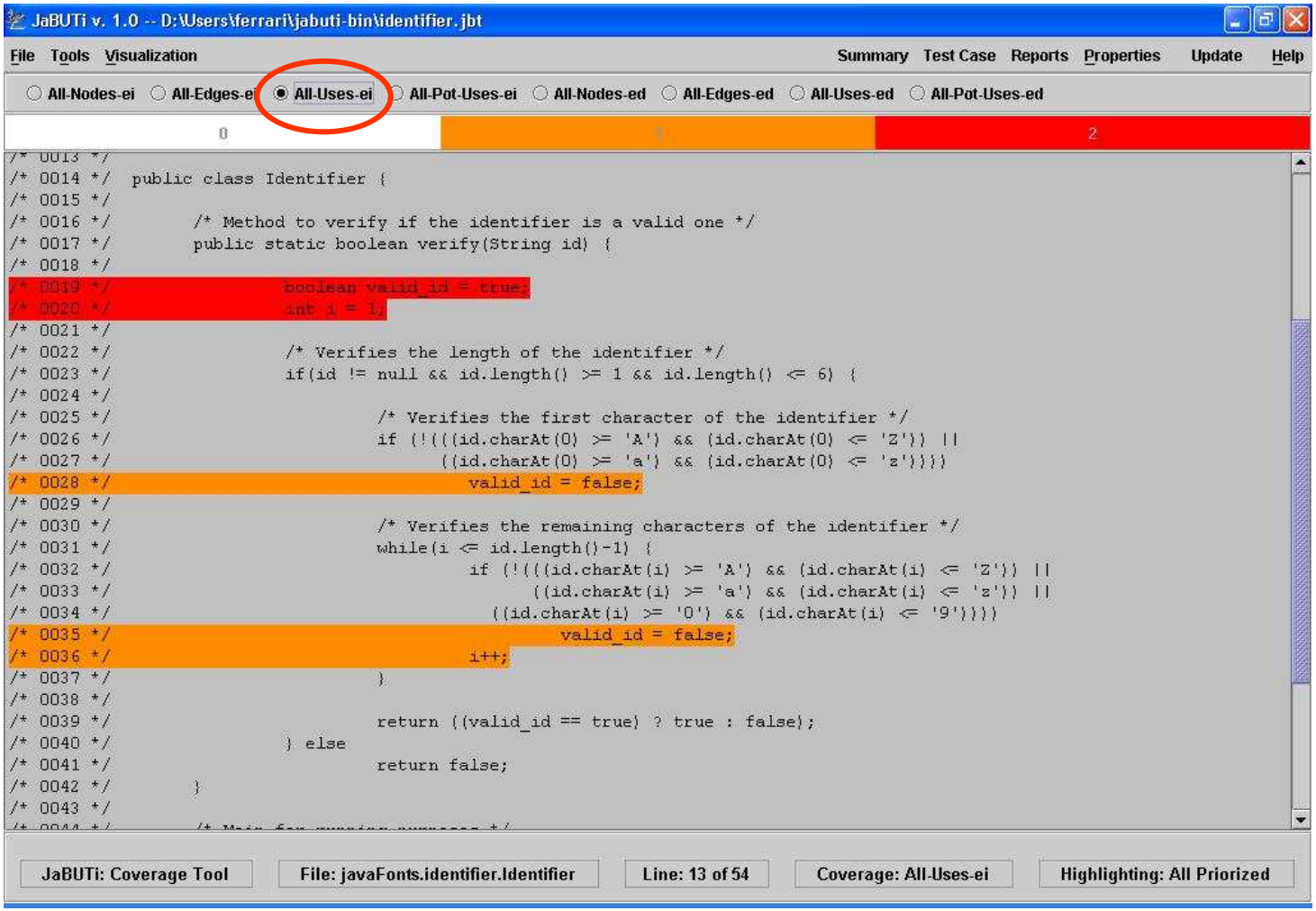
Ferramenta JaBUTi

- **Cobertura Parcial (Todas-Arestas$_{ei}$)**



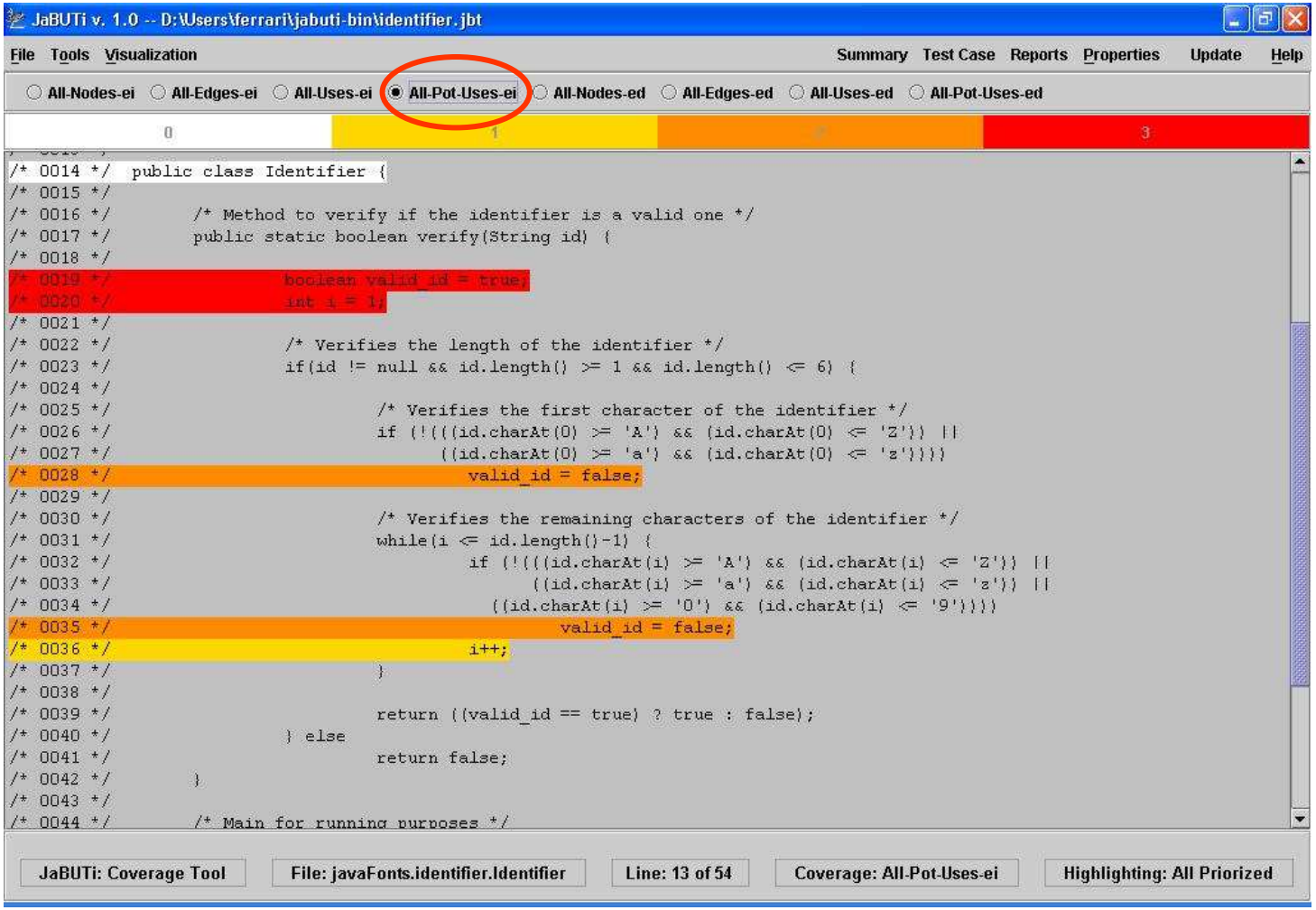
Ferramenta JaBUTi

- **Cobertura Parcial (Todos-Usos$_{ei}$)**



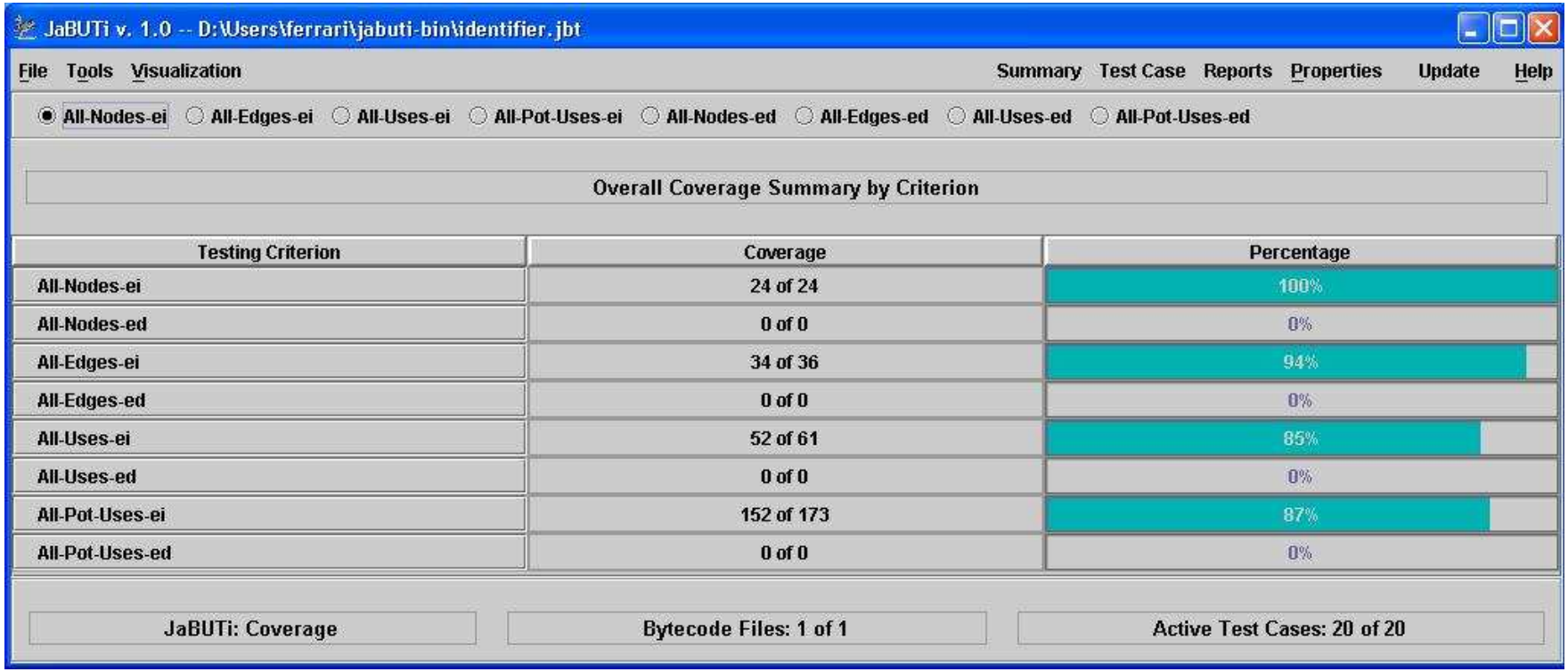
Ferramenta JaBUTi

- **Cobertura Parcial (Todos-Pot-Usos$_{ei}$)**



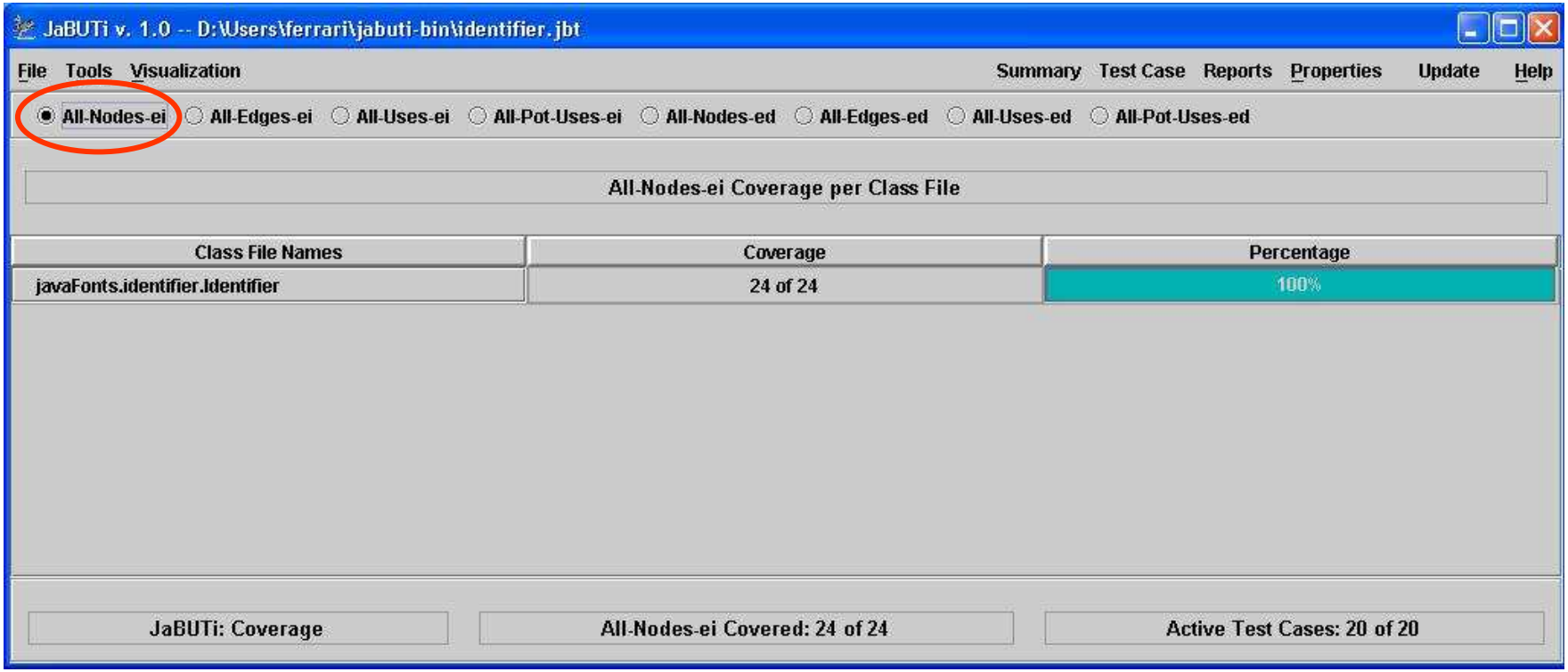
Ferramenta JaBUTi

- **Relatório da Cobertura por Critério de Teste**



Ferramenta JaBUTi

- **Relatório da Cobertura por Classe**
  - Todos-Nós$_{ei}$



Ferramenta JaBUTi

- **Relatório da Cobertura por Método**
  - Todos-Nós$_{ei}$

Ferramenta JaBUTi

- **Relatório da Cobertura por Caso de Teste**
  - Todos-Nós$_{ei}$