

Karina Guarizzo

0502050

# **MÉTRICAS DE SOFTWARE**

Jaguariúna

2008

Karina Guarizzo

0502050

## **MÉTRICAS DE SOFTWARE**

Monografia apresentada à disciplina Trabalho de Graduação III, do Curso de Ciência da Computação da Faculdade de Jaguariúna, sob a orientação do Prof. Ms. Peter Jandl Jr, como exigência parcial para conclusão do curso de graduação.

Jaguariúna

2008

GUARIZZO, Karina. **Métricas de Software**. Monografia defendida e aprovada na FAJ em 11 de Dezembro de 2008 pela banca examinadora constituída pelos professores:

---

Prof. Ms. Peter Jandl Jr – FAJ Orientador

---

Prof. José Arnaldo G. Nunes - FAJ

---

Prof<sup>a</sup>. Selma Cintra

Agradeço a Deus, pois sem ele não estaria aqui, aos meus pais e ao meu namorado, pelo incentivo, pela força e apoio nos momentos mais difíceis. O meu muito obrigado.

## **AGRADECIMENTOS**

A realização deste Trabalho teve a colaboração de muitas pessoas, com muito carinho, manifesto minha gratidão, em especial:

Ao Mestre/Professor Peter, pela atenção e dedicação, que ao longo deste trabalho esteve presente, auxiliando nas dificuldades e contribuindo para o sucesso desta jornada.

Ao meu namorado, que foi meu maior presente na faculdade, que está sempre ao meu lado, pelo amor, carinho, compreensão e a disposição que sempre teve em me ajudar e me levantar nos momentos mais difíceis durante esses quatro anos.

Aos Professores do Curso de Ciência da Computação, pelo ensino e sabedoria que me proporcionaram durante esses quatro anos.

À minha família, por ser a pessoa que sou hoje, sempre me mostrando o melhor caminho, me incentivando para sempre lutar pelos ideais, sem desistir, durante todo o tempo, foram minha luz para prosseguir esta caminhada.

Aos amigos(as) que tive a oportunidade de conhecer, em especial no meu primeiro ano de faculdade, tivemos momentos inesquecíveis na qual vou guardar boas lembranças para o resto de minha vida.

Ao meu chefe pela ajuda, os conhecimentos que me proporcionou, contribuindo para a realização deste trabalho.

GUARIZZO, Karina. **Métricas de Software**. 2008. Monografia (Bacharelado em Ciência da Computação) – Curso de Ciência da Computação da Faculdade de Jaguariúna, Jaguariúna.

## RESUMO

Com o crescimento constante da demanda pelos serviços de desenvolvimento de software, vem aumentando a procura por técnicas e ferramentas de melhoria da qualidade. Essa melhoria do processo de software é um objetivo fundamental para as organizações e deve estar baseada em medições. No entanto, definir, coletar e analisar um conjunto de métricas não é uma tarefa trivial.

Pressman (1995) afirma, que ao solicitar e avaliar as medidas da produtividade e da qualidade de software, a alta administração pode estabelecer metas significativas de melhoria do processo de Engenharia de Software.

Este Trabalho foi desenvolvido com o objetivo de mostrar a importância que as métricas (medições de software) têm em seu ciclo de vida. Com a aplicação das métricas de software, o resultado final é favorável para as organizações, para o cliente e principalmente para o usuário final. Antes de o software ser entregue, é possível detectar e prever erros, falhas e possíveis empecilhos decorrentes do desenvolvimento.

Palavras- chave: ENGENHARIA DE SOFTWARE, QUALIDADE, MÉTRICAS.

# SUMÁRIO

Lista de Figuras.....	ix
Lista de Tabelas.....	x
1. INTRODUÇÃO.....	1
2. AS MÉTRICAS DE SOFTWARE .....	3
2.1. Definições de métricas de software .....	3
2.2. Definições complementares.....	4
2.3. Categorização das métricas .....	4
2.3.1. Métricas diretas e indiretas.....	4
2.3.2. Métricas orientadas a tamanho e a função .....	5
2.3.3. Métricas de produto e produtividade.....	6
2.3.4. Métricas de qualidade e métricas técnicas .....	6
2.3.5. Métricas privadas e públicas.....	7
2.4. Utilização das métricas .....	7
2.5. Justificativa do uso das métricas .....	7
2.6. Coleta, computação e avaliação das métricas .....	8
2.7. Medição o software.....	8
2.7.1. Medidas técnicas.....	9
2.8. Processo de medição do software .....	9
2.8.1. Os quatro papéis da medição .....	9
2.9. O limite de medir .....	10
3. METODOLOGIAS.....	11
3.1. Goal Question Metrics .....	11
3.2. Metodologia cascata.....	11
4. MÉTRICAS PARA ORGANIZAÇÕES PEQUENAS .....	13
4.1. Estimativas .....	13
4.2. Etiquetas de métricas .....	13
4.3. Principais barreiras para utilizações das métricas.....	14
4.4. Vantagens das métricas de software .....	14
4.5. Desvantagem das métricas de software .....	15
5. TESTE DE SOFTWARE .....	16
5.1. Objetivos da atividade de teste.....	16
6. FERRAMENTAS.....	17
6.1. Critério de escolha da linguagem de programação .....	17

6.2.	O software escolhido .....	18
6.3.	Ferramenta AppPerfect .....	19
6.4.	Ferramenta Eclipse Metrics Plugin .....	24
6.5.	Ferramenta FindBugs .....	28
7.	COMPARAÇÃO DOS RESULTADOS .....	34
7.1.	Tabela de resultados .....	34
7.2.	Quadro comparativo .....	35
7.3.	Avaliação geral .....	35
7.4.	Conclusão das ferramentas e das métricas .....	36
8.	CONCLUSÕES.....	37
9.	REFERÊNCIAS BIBLIOGRÁFICAS.....	38



## LISTA DE FIGURAS

Figura 01: Processo de medição de produto.....	09
Figura 02: Modelo Cascata.....	12
Figura 03: Índice de linguagem de programação .....	18
Figura 04: Inclusão do software na plataforma Eclipse.....	19
Figura 05: Execução da ferramenta AppPerfect.....	20
Figura 06: Ferramenta apresenta o nome do autor do software.....	20
Figura 07: Ferramenta analisa o software.....	21
Figura 08: Algumas métricas obtidas.....	21
Figura 09: Relatório exportado pra PDF.....	22
Figura 10: Relatório exportado para EXCEL.....	22
Figura 11: Relatório exportado para HTML.....	23
Figura 12: Relatório exportado para XML.....	23
Figura 13: Resumo do projeto.....	24
Figura 14: Execução da ferramenta Metrics.....	24
Figura 15: Algumas métricas do Eclipse Metrics Plugin.....	25
Figura 16: Métricas da classe Frame.....	26
Figura 17: Alerta de número de parâmetros.....	26
Figura 18: Erros e advertências.....	27
Figura 19: Pacote padrão do software (default package).....	27
Figura 20: Relatório exportado para XML.....	28
Figura 21: Execução da ferramenta FindBugs.....	29
Figura 22: Teste de ponto flutuante.....	29
Figura 23: Método se manifesta.....	30
Figura 24: Alerta que método pode falhar.....	31
Figura 25: Variável com valor nulo.....	32
Figura 26: Chamar método desliga máquina virtual Java.....	33

## LISTA DE TABELAS

Tabela 01: Ferramenta para as métricas.....	17
Tabela 02: Resultados (números quantitativos) das métricas .....	34
Tabela 03: Resultados descritivos das métricas .....	35
Tabela 04: Avaliação das métricas .....	36

# 1 INTRODUÇÃO

Segundo Sommerville (Engenharia de Software - 2003), Engenharia de Software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software. Em geral os engenheiros de software adotam uma abordagem sistemática e organizada em seu trabalho, uma vez que essa é, com freqüência, a maneira mais eficaz de produzir software de alta qualidade. No entanto a engenharia tem a ver, em grande parte, com a questão de selecionar o método mais apropriado para um conjunto de circunstâncias, e uma abordagem mais criativa e informal para o desenvolvimento pode ser eficaz em algumas circunstâncias.

Através da Engenharia de Software pode-se buscar um dos fatores essenciais no processo de desenvolvimento que é a qualidade. Porém, para produzir um processo com qualidade, á princípio é necessário conhecer seu conceito, suas características e aplicar seus métodos e técnicas para obter os resultados desejados.

Os pontos principais de qualidade são, verificar se o produto foi desenvolvido corretamente, e validar se o produto ficou de acordo com a especificação de requisitos.

Algumas organizações de desenvolvimento de software sabem da importância de ter um produto com qualidade, não apenas seu conceito, mas a colocação de todo processo em prática. Mas, infelizmente não são todas que pensam desta forma. Mesmo que algumas tenham o objetivo de aplicar métodos de qualidade, elas não possuem o conhecimento de como medir essa qualidade.

Pressman (Engenharia de Software - 1995) afirma que a medição faz parte de uma série de “medicações” que podem ajudar a curar a aflição de software, ela oferece benefícios em nível estratégico, em nível de projeto e em nível técnico.

Este trabalho mostra a importância de serem aplicados testes nos softwares desenvolvidos, e para isso são necessárias métricas de software. Assim é necessário um estudo sobre as medidas do software, seus conceitos, ferramentas de métricas e sua utilização no sentido de explorar suas funcionalidades.

A organização deste trabalho é a seguinte: no capítulo 2 foram apresentadas as métricas de software, suas definições e tipos, além do tratamento de algumas medidas. No capítulo 3 foram apresentadas algumas metodologias que podem ser utilizadas durante o processo de software. Em seguida, no capítulo 4, foram citadas métricas para organizações de pequeno porte, uma empresa quando está em seu momento de crescimento, esta fase, é a hora certa de se aplicar a engenharia de software, assim, acompanhando todo o ciclo de

vida do projeto. No capítulo 5, foi apresentado um breve comentário sobre os testes de software, prosseguindo, no capítulo 6, foram mostradas as ferramentas utilizadas para extrair as métricas, inclusive o software escolhido para a realização dos testes. No capítulo 7, foi feita uma comparação com os resultados das métricas, analisando as diferenças entre as três ferramentas utilizadas. E para finalizar, no capítulo 8, foi feita a conclusão de todo este trabalho, ou seja, a comparação dos resultados das métricas.

## **2 AS MÉTRICAS DE SOFTWARE**

### **2.1 Definições de métricas de software**

Uma métrica de software é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada, diz Sommerville (Engenharia de Software - 2003). Coletadas essas medições, as questões em relação ao software poderão ser respondidas e confirmações poderão ser feitas, de que as melhorias do software alcançaram, ou não, a meta desejada.

A métrica de software tem como princípio especificar as funções de coleta de dados de avaliação e desempenho, atribuindo essas responsabilidades a toda a equipe envolvida no projeto e analisar os históricos dos projetos anteriores.

Quando se fala de métricas deve-se ter em mente que se trata de dados, números quantitativos que irão mostrar em forma de indicadores o estado atual de um determinado projeto. A medição, sendo tão importante assim, deveria ser aplicada em todas as fases do ciclo de vida do projeto, e não somente na fase de desenvolvimento, o que é mais comum. Isso fará com que a própria equipe do projeto fique mais confiante diante dos resultados obtidos e melhore cada vez mais o processo pelo qual o projeto é desenvolvido, evoluindo sempre para a qualidade do processo e do produto.

A medição tem seu papel muito importante dentro da engenharia de software, especialmente na gerência de projetos de software, seja qual for a metodologia a ser utilizada. Ela é analisada por gerentes de projetos de software e coletada pelos engenheiros de software. Então sem as métricas dispõe-se apenas de dados subjetivos os quais não serão de grande importância, pois como foi visto anteriormente, as métricas são expressas de forma quantitativa, ou seja, em números. As métricas são feitas então em três fases: coleta de dados, cálculo dos dados e análise dos dados. É importante saber escolher a metodologia que melhor se encaixa no projeto, trazendo resultados mais precisos, sejam eles bons ou ruins.

Tendo então realizada a medição, poderão ser feitas estimativas de custos e prazos de término do projeto ou entrega do produto final. O mais importante a ser ressaltado é que a aplicação das métricas deve ser muito bem planejada e que seus resultados devem ser apresentados de uma forma clara de modo que todos possam entender os resultados obtidos. Feito isso, o resultado que se tem é um conjunto de dados que apresenta a idéia do processo e um entendimento do projeto. Permite aos gerentes de projetos de software

aperfeiçoar e melhorar o processo de desenvolvimento do produto e avaliar a qualidade do produto que está sendo produzido.

## 2.2 Definições complementares

Esta seção inclui outras definições importantes para o entendimento das métricas de software, que são:

**Medida:** Fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo.

Segundo Vasconcelos (Métricas de Software - 2005), medida é uma função de mapeamento.

**Medição:** Ato de determinação de uma medida.

**Métrica:** Medida quantitativa do grau em que um sistema se encontra em relação a um determinado atributo.

**Indicadores:** Métrica ou combinação de métricas que fornece uma compreensão de um processo/projeto/produto.

## 2.3 Categorização das métricas

As métricas podem ser categorizadas de maneiras diferentes, tais como métricas diretas e indiretas, ou métricas orientadas a tamanho, ou funções, entre outras que serão citadas neste capítulo.

### 2.3.1 Métricas diretas e indiretas

As métricas podem ser categorizadas de maneiras diferentes, tais como métricas diretas e indiretas; ou métricas orientadas a tamanho ou funções.

As métricas diretas são aquelas onde os atributos são observados (por exemplo: custo, esforço, quantidade de linhas de código produzidas, total de defeitos registrados).

As métricas indiretas são aquelas obtidas a partir de outras métricas (por exemplo: eficiência, confiabilidade, qualidade, funcionalidade).

O custo e o esforço exigidos para se construir o software, o número de linhas de código produzido e outras medidas diretas são relativamente fáceis de serem reunidas,

desde que convenções específicas para medição sejam estabelecidas antecipadamente. Porém, a qualidade e a funcionalidade do software, ou seja, eficiência e capacidade de manutenção, são mais difíceis de serem avaliadas e somente podem ser medidas indiretamente.

### 2.3.2 Métricas orientadas a tamanho e função

As métricas orientadas a tamanho, consideram o tamanho do software produzido (linhas de código), referem-se a todas as atividades da engenharia (análise, projeto, código, teste), tais como:

Produtividade:	KLOC/pessoa-mês
Qualidade:	defeitos/KLOC
Custo:	\$/LOC
Documentação:	páginas de documentação/KLOC

As métricas orientadas a função, em vez de contar as linhas de código, a métrica orientada à função concentra-se na funcionalidade do software (o que é entregue). Essa métrica consiste em um método para medição de software do ponto de vista do usuário, que determina de forma consistente o tamanho e complexidade de um software.

Uma abordagem foi sugerida por Allan Albrecht (IFPUG 1994), baseada nesta proposta chamada de pontos por função. A análise de pontos por função focaliza a perspectiva de como os usuários “enxergam” os resultados que um sistema produz. Ela se baseia parcialmente em dados subjetivos, implicando a organização estabelecer um plano de implantação da sistemática da medição, definindo padrões para contagem. Isto é fundamental para que os resultados das medições possam ser comparados entre os projetos, gerando uma linha de referência (*baseline*) das informações históricas coletadas e armazenadas.

FP (Function Points)

$FP = \text{contagem total} * [0.65 + 0.01 * \sum (Fi)]$

Fi = valores de ajuste de complexidade (i = 1,...14)

Produtividade:	FP/pessoa-mês
Qualidade:	defeitos/FP
Custo:	\$/FP
Documentação:	páginas de documentação/FP

### 2.3.3 Métricas de produto e produtividade

As métricas de produto se ocupam com as características do próprio software, elas se dividem em duas classes, diz (CLARO – Métricas de Software):

- **Métricas estáticas**, que são coletadas por medições feitas das representações do sistema, como projeto, programa ou documentação.
- **Métricas dinâmicas**, que são coletadas por medições feitas de um programa em execução.
- **Métricas de produtividade**, concentram-se na saída do processo de engenharia de software (por exemplo: número de casos de uso, iteração).

### 2.3.4 Métricas de qualidade e métricas técnicas

As métricas de qualidade, oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente (por exemplo: erros, fase).

Segundo Pressman (Engenharia de Software – 1995) existem muitas medidas de qualidade de software, elas incluem:

- **Corretitude**: um programa deve operar corretamente, caso contrário, oferecerá pouco valor aos seus usuários. Corretitude é o grau em que o software executa a função que é dele exigida.
- **Manutenibilidade**: a manutenção de software é responsável por mais esforço do que qualquer outra atividade de engenharia de software. Manutenibilidade é a facilidade com que um programa pode ser corrigido se um erro for encontrado, adaptado se o seu ambiente se modificar ou ampliado se o cliente desejar inclusões e alterações nos requisitos funcionais. Não existe nenhuma forma de se medir a manutenibilidade diretamente, deve-se usar medidas indiretas.
- **Integridade**: a integridade de software vem tornando-se cada vez mais importante na era dos hackers e dos vírus. Esse atributo mede a capacidade que um sistema tem de se suportar ataques à sua integridade, ataques podem ser feitos a todos os três componentes do software: programas, dados e documentos.
- **Usabilidade**: se um programa não for *user friendly* (amigável ao usuário) estará destinado ao fracasso, mesmo que as funções que eles executem sejam valiosas.



E as métricas técnicas, concentram-se nas características do software e não no processo por meio do qual o software foi desenvolvido, por exemplo complexidade lógica, manutenibilidade.

### **2.3.5 Métricas privadas e públicas**

As métricas privadas se referem ao escopo da equipe do projeto de software (por exemplo: defeitos para funções importantes do software, erros encontrados durante revisões técnicas formais).

As métricas públicas geralmente assimilam informações que anteriormente eram privadas de uma equipe (por exemplo: proporções de defeitos de projeto, esforço, tempo transcorrido e dados relacionados), são coletados e avaliados tentando descobrir indicadores.

## **2.4 Utilização das métricas**

Para um bom desempenho das métricas de software, é preciso antes de utilizá-las, planejar, organizar e identificar os pontos principais, diz Sommerville (Engenharia de Software – 2003)

Inicialmente, definir um padrão para as métricas:

- Escolha de medições a serem feitas
- Seleção de componentes a serem avaliados
- Medição de características dos componentes
- Adquirir as ferramentas necessárias
- Identificar medições anômalas

## **2.5 Justificativa do uso das métricas**

Com a utilização das métricas é possível conseguir melhorias e resultados mais satisfatórios do software, mais segurança para os gerentes de projeto. É uma maneira de eliminar os obstáculos, corrigir erros e falhas, antes mesmo do produto ser entregue ao cliente. Alguns itens a serem analisados:

- Entender e aperfeiçoar o processo de desenvolvimento
- Melhorar a gerência de projetos e o relacionamento com clientes
- Avaliar produtividade do processo
- Reduzir frustrações e pressões de cronograma
- Embasar solicitações de novas ferramentas e treinamentos
- Formar uma linha básica para estimativas

- No nível técnico, as medições são importantes para determinar parâmetros como quantidade de teste necessário e impacto de mudanças.

## **2.6 Coleta, computação e avaliação das métricas**

Idealmente, os dados necessários para se estabelecer uma linha básica foram compilados continuamente. Infelizmente, isso raramente acontece. Por conseguinte, a coleta de dados requer uma investigação histórica dos projetos passados para se reconstruir os dados exigidos. Logo que os dados foram coletados, a computação das métricas é possível. A avaliação dos dados concentra-se nas razões subjacentes para os resultados obtidos.

## **2.7 Medição do software**

Se não medir, não haverá nenhuma maneira real de determinar se está ou não melhorando. Pressman (1995) afirma, a medição faz parte de uma série de “medicações” que podem ajudar a curar a aflição de software, ela oferece benefícios em nível estratégico, em nível de projeto e em nível técnico.

As medições e as métricas ajudam a entender o processo técnico usado para desenvolver um produto. O processo é medido num esforço para melhorá-lo, assim como o produto é medido num esforço para aumentar sua qualidade. Também são necessárias para analisar a qualidade e a produtividade do processo de desenvolvimento; bem como a manutenção do produto de software construído.

Medir ajuda a obter o auto-conhecimento para saber se o que se tem é o suficiente, quem é a empresa envolvida, e em qual ponto se encontra o projeto. Também ajuda a entender a pressão imediata, saber o que dever ser feito, e qual o caminho a seguir. Também, entender porque a medição é importante para avaliação e garantia de qualidade de software, conhecer algumas métricas e suas aplicações, entender o que é um plano de métricas e como escrever um.

Gomes A (Metricas e Estimativas de Software - 2008) afirma, medir e estimar é a parte mais importante de um projeto de sistema bem-sucedido.

Assim, preparar-se para o futuro, sabendo da atual situação, qual o próximo passo e se caso precisar mudar a direção, ter a confiança que não haverá impactos, causando problemas futuros.

Pode-se utilizar a aplicação de medição em:

- Processo de software, com o objetivo de melhorá-lo de forma contínua, visão estratégica de organização.
- Projeto de software, para auxiliar na estimativa, no controle de qualidade, na avaliação de produtividade e no controle de projeto.

### 2.7.1 Medidas técnicas

Medidas técnicas são necessárias para qualificar a performance técnica dos produtos do ponto de vista do desenvolvedor, diz (CORDEIRO – Métricas de Software). Por outro lado, medidas funcionais são necessárias para qualificar a performance dos produtos pela perspectiva do usuário. Medidas funcionais devem ser independentes das decisões do desenvolvimento técnico e implementação. Tais medidas podem ser utilizadas para comparar a produtividade de diferentes técnicas e tecnologias.

## 2.8 Processo de medição do software

Sommerville (2003) afirma que, um processo de medição de software, pode ser parte de um processo de controle de qualidade, cada componente do sistema é analisado, e os diferentes valores da métrica devem ser comparados entre si. Medições anômalas devem ser utilizadas para enfocar o esforço de garantia de qualidade nos componentes que possam apresentar problemas de qualidade. Como pode ser observado na Figura 01.

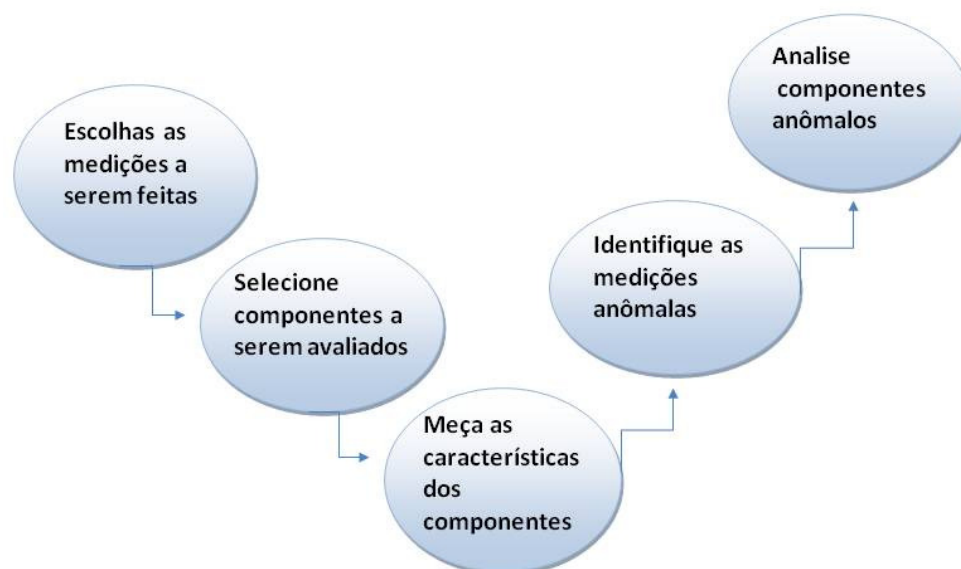


Figura 01 – Processo de medição de produto (Fonte: Sommerville, 2003)

### 2.8.1 Os quatro papéis da medição

Os papéis da medição podem ser considerados como quatro: caracterizar, avaliar, prever e aperfeiçoar.

#### 1. Caracterizar

- Caracterizar para ter entendimento do processo, produtos e recursos
- Estabelecer marcos básicos
- Prever ou aperfeiçoar

## **2. Avaliar**

- Avaliar para determinar o status com referência aos planos
- Sensores para avaliar quanto os projetos e processos estão fora de controle
- Verificar o modo para trazer os projetos de volta ao controle
- Verificar o cumprimento de metas de qualidade
- Verificar os impactos de melhoramentos de tecnologias

## **3. Prever**

- Para poder planejar
- Observação de todo o processo e do produto como forma de utilizar valores observados para prever outros
- Ajudam a extrapolar tendências, onde as estimativas de custos, prazos e qualidade podem ser atualizadas.

## **4. Aperfeiçoar**

- Coletar informações quantitativas para ajudar a identificar bloqueios, causas fundamentais, ineficiências.
- Melhorar a qualidade do produto e o desempenho do sucesso
- Avaliamos para determinar o status com referência aos planos
- Sensores para avaliar quanto os projetos e processos estão fora de controle
- Verificar o modo para trazer os projetos de volta ao controle
- Verificar o cumprimento de metas de qualidade
- Verificar os impactos de melhoramentos de tecnologias

## **2.9 O limite de medir**

Medir é importante, porém, esta medida é limitada. Gerentes de projetos, devem saber o momento exato de encerrar as medições, para que o software não seja prejudicado, e as outras fases do projeto também.

Medir até o momento de se alinhar os objetivos do projeto (necessidades), com os objetivos da empresa, estabelecer um programa de métricas adequado, fundamentado e gradual e não medir mais do que é necessário. Não é possível medir, se não conseguir:

- controlar;
- gerenciar;
- melhorar; e
- trabalhar.

### 3 METODOLOGIAS

Metodologia, na engenharia de software, pode ser considerada como um conjunto estruturado de práticas, que pode ser seguido e repetido durante todo o processo de produção de software.

Seguem duas metodologias que podem ser utilizadas: GQM e Cascata.

#### 3.1 GQM (Goal Question Metrics)

Vasconcelos (2005) explica que esta metodologia é usada para definir o conjunto de métrica a ser coletado, proposto por Basili e Rombach's (IEEE - 1988). E é baseada no fato de que deve existir uma necessidade clara e objetiva associada a cada métrica.

O significado de GQM (*Goal Question Metrics*) é:

- **GOAL:** Quais são as metas/objetivos?
- **QUESTION:** Quais questões se deseja responder?
- **METRICS:** Quais métricas poderão ajudar?

O objetivo da GQM é assegurar que todos os defeitos são corrigidos antes do software ser liberado para uso. Suas fases são:

- I. Planejamento
- II. Definição
- III. Coleta de dados
- IV. Interpretação

#### 3.2 Metodologia Cascata

Segundo Gomes B (Metodologias de Desenvolvimento de Software - 2008), no modelo em cascata o projeto segue uma série de passos ordenados. Ao final de cada fase, a equipe do projeto finaliza uma revisão, o desenvolvimento não continua até que o cliente esteja satisfeito com os resultados. Pode ser observado o modelo cascata na Figura 02.

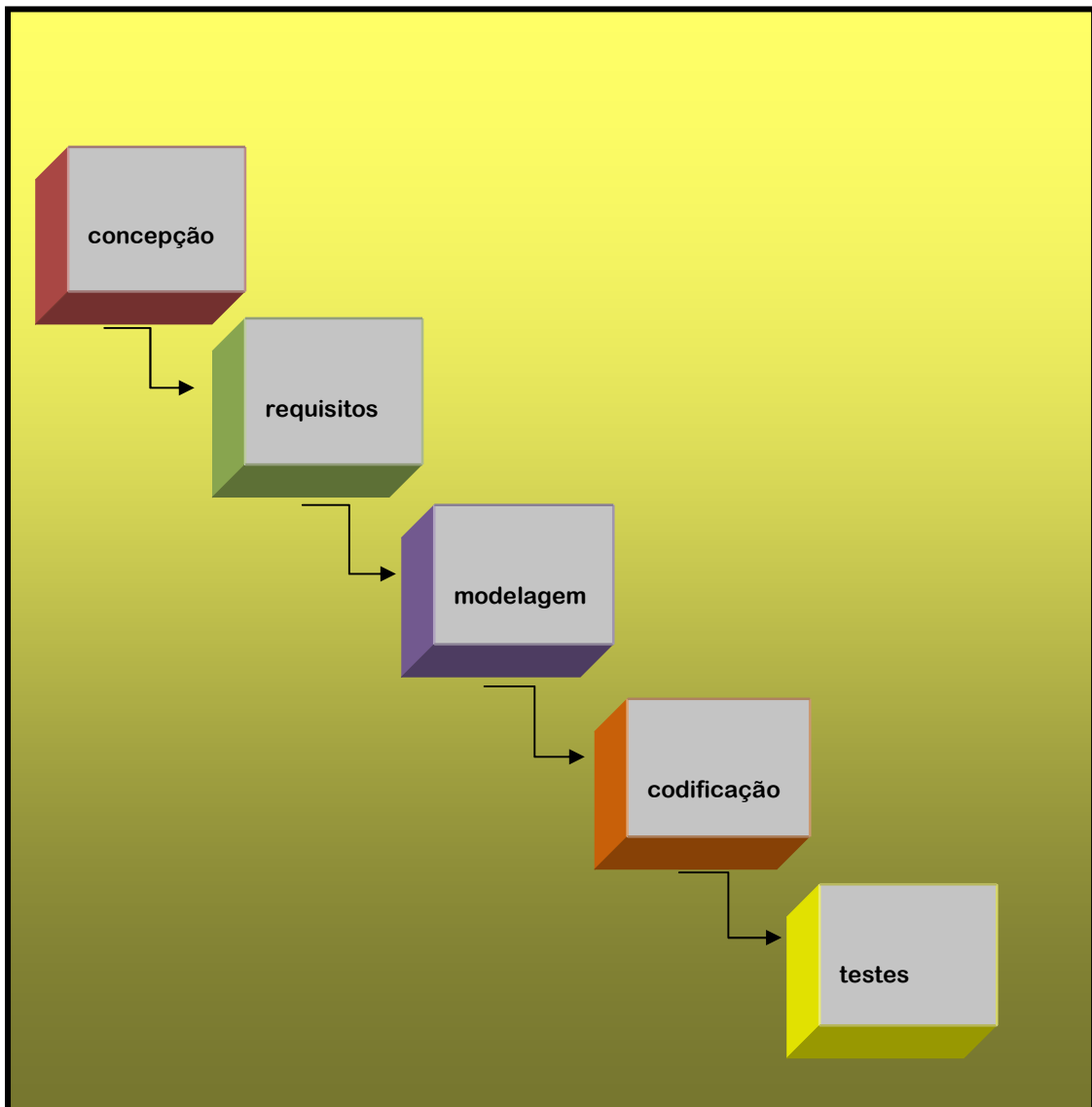


Figura 02 – Modelo Cascata (Fonte: Sommerville, 2003)

Se for necessário efetuar alguma modificação, voltar os passos de desenvolvimento do projeto é complicado. A metodologia em cascata é extremamente formal, pode-se afirmar que é baseada em documentos e com certeza possui uma enorme quantidade de “entregáveis” e saídas que nada mais são do que documentos. Outras características deste modelo é o alto valor dado ao planejamento. O forte planejamento inicial reduz a necessidade de planejamento contínuo conforme o andamento do projeto.

## 4 MÉTRICAS PARA ORGANIZAÇÕES PEQUENAS

Pequenas organizações são empresas que possuem uma quantidade pequena de funcionários, consideradas de pequeno porte. Assim, a estratégia é iniciar um processo de melhoria na fase de crescimento da empresa, com poucos desenvolvedores, com projetos pequenos, o gerenciamento ainda é controlável e o tempo ainda não é um problema.

As métricas citadas abaixo podem ser consideradas para pequenas organizações, iniciantes em projetos de software, adquirindo desde o começo, a disciplina da engenharia de software.

- Tempo (horas ou dias) transcorridos entre o momento em que o pedido foi feito até que a avaliação seja completada
- Esforço (pessoas/horas) para realizar a avaliação
- Esforço necessário para fazer a modificação (pessoas/hora)
- Tempo necessário
- Erros descobertos durante o trabalho
- Defeitos descobertos

### 4.1 Estimativas

Uma das atividades fundamentais do processo de gerenciamento de projetos de software é o planejamento do esforço humano exigido, duração cronológica do projeto e custo deve ser derivadas. Uma série de técnicas de estimativas foram disponibilizadas para o desenvolvimento de software, segundo Pressman (Engenharia de Software – 1995):

- O escopo do projeto deve ser estabelecido antecipadamente
- Métricas de software são utilizadas e o histórico de aferições passadas é usado como uma base a partir da qual estimativas são feitas
- O projeto é dividido em pequenas partes que são estimadas individualmente

### 4.2 Ética das métricas

As métricas devem sempre fornecer benefícios para a organização com o intuito de aperfeiçoar o seu nível de maturidade, seguem algumas etiquetas:

Bom senso e sensibilidade empresarial quando interpretar dados de métricas

- Fornecer realimentação aos indivíduos que coletam medidas e métricas
- Não usar métricas para avaliar indivíduos

- Trabalhar com profissionais e indivíduos para estabelecerem metas claras e métricas que devem ser usadas para alcançá-las
- Nunca usar métricas para ameaçar indivíduos
- Dados de métricas que indicam uma área problemática não devem ser considerados negativos

### **4.3 Principais barreiras para utilizações das métricas**

As organizações de software, muitas delas, tem o planejamento das métricas, porém, devido alguns fatores, não é possível desenvolver o processo de medição. Abaixo seguem alguns desses fatores:

- Falta de comprometimento da alta gerência
- Medir custo caro
- Os maiores benefícios vêm a longo prazo
- Má utilização das métricas
- Grande mudança cultural necessária
- Dificuldade de estabelecer medições apropriadas e úteis
- Interpretações de dados realizadas de forma incorreta
- Obter o comprometimento de todos os envolvidos e impactados
- Estabelecer um programa de medições é fácil, o difícil é manter

### **4.4 Vantagens das métricas de software**

O uso das métricas de software permite obter uma série de vantagens:

Diminuir:

- Defeitos
- Prazo de entrega
- Desperdício
- Custo

Aumentar:

- Satisfação do cliente
- Produtividade dos recursos
- Visibilidade das ações
- Qualidade de gerenciamento



## **4.5 Desvantagem das métricas de software**

Uma desvantagem é que a métrica de software não oferece cem por cento de confiança em seus resultados. A métrica serve de base para o conhecimento no campo da medição na gestão de projetos, com ajuda de projetos que já foram concluídos no passado.

## **5 TESTE DE SOFTWARE**

Segundo Pressman (Engenharia de Software – 1995), a atividade de teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão de especificação, projeto e codificação.

Realizados de forma cuidadosa e criteriosa, o teste assume uma importância cada vez maior dado o impacto sobre o funcionamento e o custo, está sendo portanto um fator de muita importância nas empresas de software.

### **5.1 Objetivos da atividade de teste**

Os objetivos das atividades de teste são:

- A atividade de teste é o processo de executar um programa com a intenção de descobrir um erro
- Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto
- Um teste bem sucedido é aquele que revela um erro ainda não descoberto

Alguns fatos das principais causas para o insucesso e o alto custo dos sistemas de informação:

- A falta de maturidade
- O desinteresse das empresas de desenvolvimento de sistemas
- A baixa popularidade deste assunto entre os profissionais da área de informática

## 6 FERRAMENTAS

A maioria das aplicações hoje é concebida e desenvolvida para uso simultâneo por um grande número de usuários. Às vezes, existem problemas subjacentes no código, que podem causar erros, assim, torna-se particularmente crítico para o dono da obra garantir que a aplicação está disponível e confiável.

Através das métricas, surge a coleta de dados, observando se existem métricas adequadas, se as comparações foram produtivas e se alguns dos resultados contribuíram para a melhoria do software. Durante os estudos realizados, foi pesquisado e encontrada muitas ferramentas de métricas, de diversos tipos, de varias linguagens.

Na Tabela 01 são relacionadas algumas ferramentas pesquisadas.

Tabela 01 – Ferramentas para métricas

LINGUAGEM	FERRAMENTAS			
<b>C</b>	CQual	Splint	Frama-C	CCured
<b>C/C++</b>	QA-C	HP Code Advisor	CMT++	PREfast
<b>JAVA</b>	AppPerfect	Metrics	FindBugs	SCL
<b>C#</b>	SLOCCount	Stonehenge	MyscoolPlan	Black

As ferramentas de teste escolhidas foram AppPerfect (Open source software), Eclipse Metrics Plugin (Open source software) e FindBugs (Open source software).

### 6.1 Critério de escolha da linguagem de programação

Antes de se ser definida a escolha do software e as seleções das ferramentas, foi feita uma pesquisa sobre a utilização (popularidade) das linguagens de programação. Como mostra a Figura 03, a linguagem Java, está na primeira posição.

Position Nov 2008	Position Nov 2007	Delta in Position	Programming Language	Ratings Nov 2008	Delta Nov 2007	Status
1	1	=	Java	20.299%	-0.24%	A
2	2	=	C	15.276%	+1.31%	A
3	4	↑	C++	10.357%	+1.61%	A
4	3	↓	(Visual) Basic	9.270%	-0.96%	A
5	5	=	PHP	8.940%	+0.25%	A
6	7	↑	Python	5.140%	+0.91%	A
7	8	↑	C#	4.026%	+0.11%	A
8	11	↑↑↑	Delphi	4.006%	+1.55%	A
9	6	↓↓↓	Perl	3.876%	-0.86%	A
10	10	=	JavaScript	2.925%	0.00%	A
11	9	↓↓	Ruby	2.870%	-0.21%	A
12	12	=	D	1.442%	-0.26%	A
13	13	=	PL/SQL	0.939%	-0.24%	A
14	14	=	SAS	0.729%	-0.40%	A--
15	18	↑↑↑	ABAP	0.570%	-0.08%	B
16	19	↑↑↑	Pascal	0.511%	-0.13%	B
17	17	=	COBOL	0.510%	-0.20%	B

Figura 03 - Índice de linguagem de programação (Fonte: <http://www.tiobe.com>)

## 6.2 O software escolhido

O software utilizado para a realização das métricas foi o AS - Autômata Simulador, um programa desenvolvido em Java (Jandl, 1999), que simula um robô em um ambiente fechado de quatro paredes, cuja função é percorrer o espaço sem encostar nas paredes.

A figura 04, mostra a inclusão deste software na plataforma Eclipse, para os devidos testes.

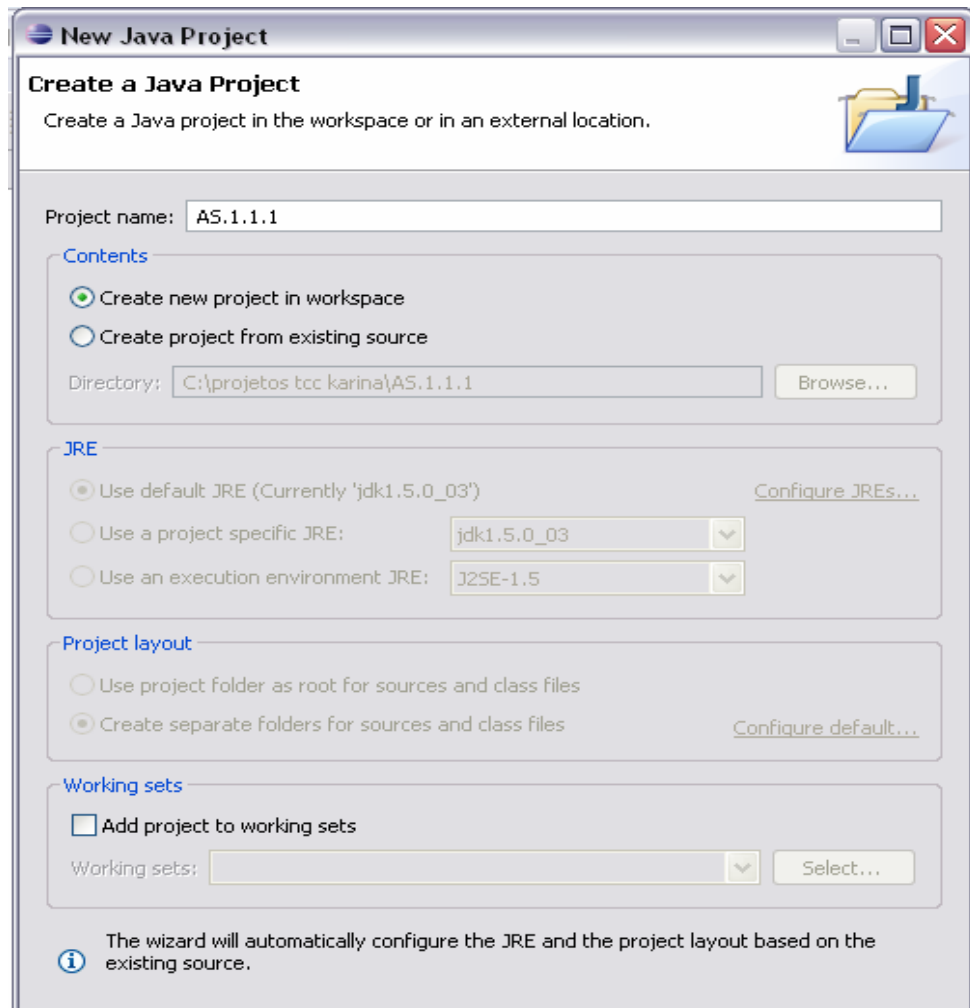


Figura 04 - Inclusão do software na plataforma Eclipse

## 6.3 Ferramenta AppPerfect

O *AppPerfect Java Profiler* é uma ferramenta *open source*, para programas em Java. Projetada para ajudar a encontrar erros, bem como recursos vitais do sistema em que você está executando o aplicativo. Ele elimina a necessidade de passar longas horas a atravessar o código, alertando os problemas de sua aplicação.

Objetivos:

- Encontrar e traçar erros.

Características:

- Aponta os problemas associados à sua aplicação;
- Fornece informações estatísticas completas e precisas.

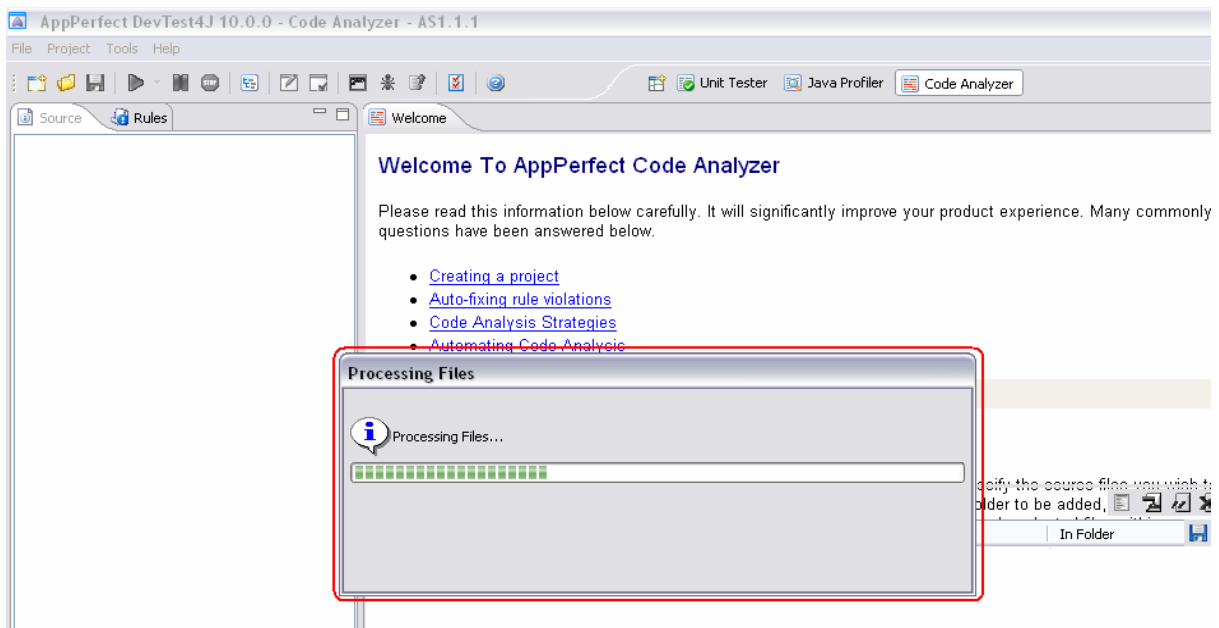


Figura 05 - Execução da ferramenta AppPerfect

A Figura 06 mostra que a ferramenta apresenta o nome do autor do software e, após verificação, mostra que é um software sem irregularidades.

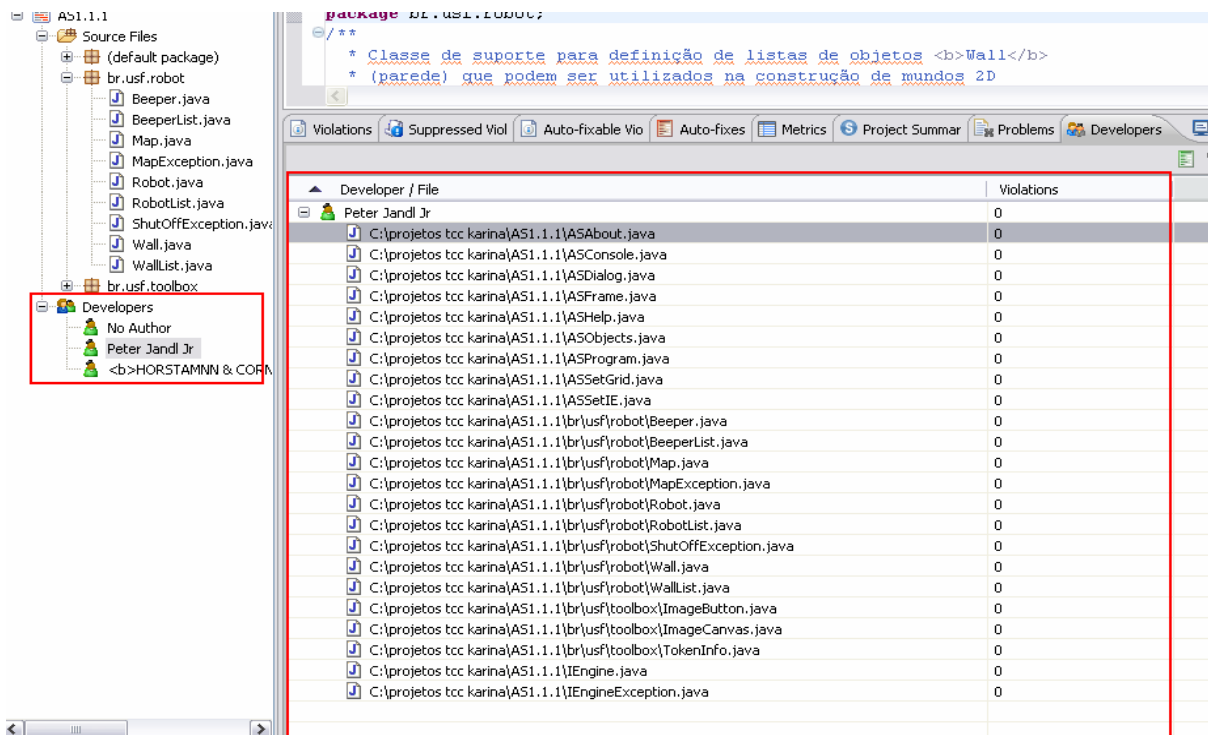


Figura 06 - Ferramenta apresenta o nome do autor do software

A Figura 07 mostra o momento em que a ferramenta analisa todo o projeto. Podem ser acompanhados o tempo restante e a quantidade já verificada.

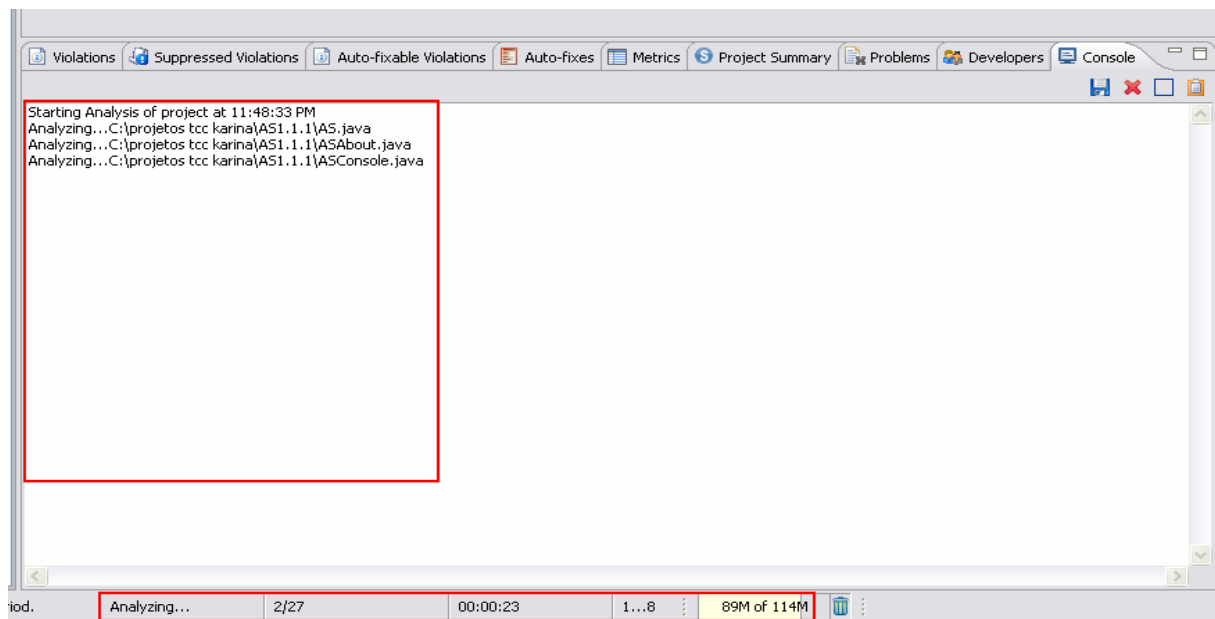


Figura 07 - Ferramenta analisa o software

Os resultados são observados na Figura 08.

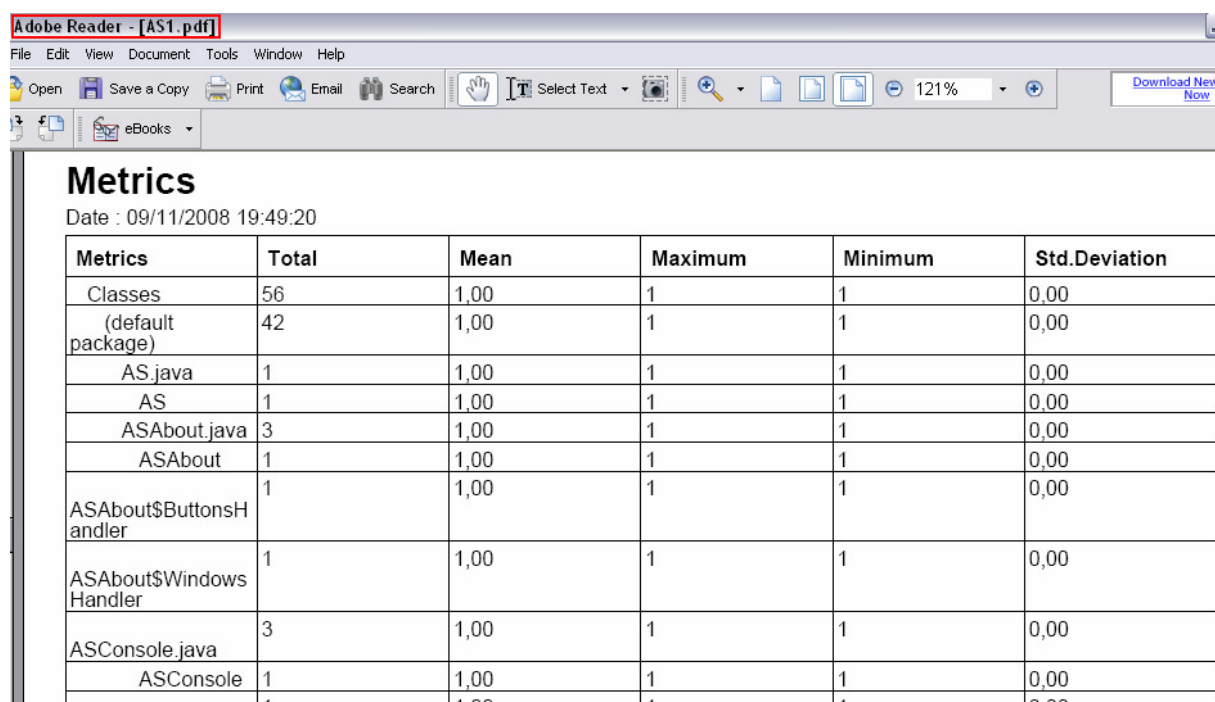
Metrics	Total	Mean	Maximum	Minimum	Std.Deviat...
Classwise Lines	3,355	59,91	803	5	122,48
+ (default package)	2,225	52,98	803	5	127,67
+ br.usf.robot	781	86,78	399	12	123,08
+ br.usf.toolbox	349	69,80	125	5	45,00
Method Complexity	661	2,71	20	1	3,03
+ (default package)	394	3,13	20	1	3,32
+ br.usf.robot	197	2,19	17	1	2,57
+ br.usf.toolbox	70	2,50	10	1	2,78
Method Parameters	275	1,13	8	0	1,45
+ (default package)	141	1,12	8	0	1,46
+ br.usf.robot	94	1,04	7	0	1,47
+ br.usf.toolbox	40	1,43	5	0	1,35
Methods	244	4,36	41	1	8,11
+ (default package)	126	3,00	39	1	5,99
+ br.usf.robot	90	10,00	41	2	13,69
+ br.usf.toolbox	28	5,60	13	1	4,96
Nested Block Depth	376	1,54	6	1	0,89
+ (default package)	211	1,67	6	1	1,01
+ br.usf.robot	121	1,34	4	1	0,69
+ br.usf.toolbox	44	1,57	4	1	0,82
Non-static Attributes	120	2,14	15	0	3,61
+ (default package)	79	1,88	15	0	3,74
+ br.usf.robot	27	3,00	10	1	3,27
+ br.usf.toolbox	14	2,80	7	0	2,64
Static Attributes	17	0,30	5	0	0,80
+ (default package)	11	0,26	5	0	0,85
+ br.usf.robot	3	0,33	1	0	0,47
+ br.usf.toolbox	3	0,60	2	0	0,80
Total Lines	6.507	80,33	809	0	127,11
+ Total Blank Lines	1.537	56,93	362	0	88,80
+ Total Code Lines	3.433	127,15	809	7	171,49
+ Total Comment Lines	1.537	56,93	362	0	88,80

Figura 08 – Algumas métricas obtidas

Esta ferramenta apresenta número total de linhas de código, por completo:

- Linhas do código;
- Linhas em branco; e
- Linhas comentadas.

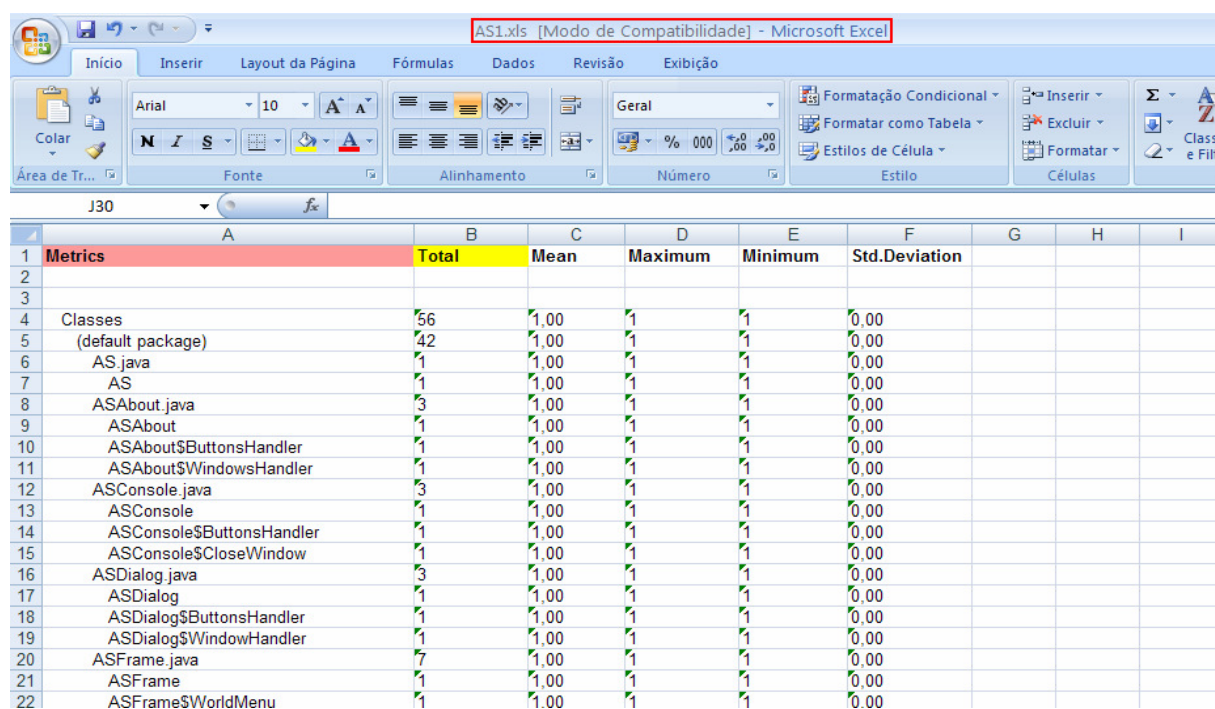
Essa ferramenta tem a função de gerar um relatório dos resultados das métricas e permite exportar seus resultados nos formatos: PDF (Figura09), Excel (Figura 10), HTML (Figura 11) e XML (Figura 12).



**Metrics**  
Date : 09/11/2008 19:49:20

Metrics	Total	Mean	Maximum	Minimum	Std.Deviation
Classes	56	1,00	1	1	0,00
(default package)	42	1,00	1	1	0,00
AS.java	1	1,00	1	1	0,00
AS	1	1,00	1	1	0,00
ASAbout.java	3	1,00	1	1	0,00
ASAbout	1	1,00	1	1	0,00
ASAbout\$ButtonsHandler	1	1,00	1	1	0,00
ASAbout\$WindowsHandler	1	1,00	1	1	0,00
ASConsole.java	3	1,00	1	1	0,00
ASConsole	1	1,00	1	1	0,00

Figura 09 - Relatório exportado para PDF



**AS1.xls [Modo de Compatibilidade] - Microsoft Excel**

Metrics	Total	Mean	Maximum	Minimum	Std.Deviation
Classes	56	1,00	1	1	0,00
(default package)	42	1,00	1	1	0,00
AS.java	1	1,00	1	1	0,00
AS	1	1,00	1	1	0,00
ASAbout.java	3	1,00	1	1	0,00
ASAbout	1	1,00	1	1	0,00
ASAbout\$ButtonsHandler	1	1,00	1	1	0,00
ASAbout\$WindowsHandler	1	1,00	1	1	0,00
ASConsole.java	3	1,00	1	1	0,00
ASConsole	1	1,00	1	1	0,00
ASConsole\$ButtonsHandler	1	1,00	1	1	0,00
ASConsole\$CloseWindow	1	1,00	1	1	0,00
ASDialog.java	3	1,00	1	1	0,00
ASDialog	1	1,00	1	1	0,00
ASDialog\$ButtonsHandler	1	1,00	1	1	0,00
ASDialog\$WindowHandler	1	1,00	1	1	0,00
ASFrame.java	7	1,00	1	1	0,00
ASFrame	1	1,00	1	1	0,00
ASFrame\$WorldMenu	1	1,00	1	1	0,00

Figura 10 - Relatório exportado para EXCEL



AS1.1.1 - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço C:\Documents and Settings\gu\Desktop\APPPERFECT\RESULTADOS\METRICS\AS1.html

**AS1.1.1 : Metrics**

Date : 09/11/2008 19:49:58

Metrics	Total	Mean	Maximum	Minimum	Std.Dev
Classes	56	1,00	1	1	0,00
(default package)	42	1,00	1	1	0,00
AS.java	1	1,00	1	1	0,00
AS	1	1,00	1	1	0,00
ASAbout.java	3	1,00	1	1	0,00
ASAbout	1	1,00	1	1	0,00
ASAbout\$ButtonsHandler	1	1,00	1	1	0,00
ASAbout\$WindowsHandler	1	1,00	1	1	0,00
ASConsole.java	3	1,00	1	1	0,00
ASConsole	1	1,00	1	1	0,00
ASConsole\$ButtonsHandler	1	1,00	1	1	0,00
ASConsole\$CloseWindow	1	1,00	1	1	0,00
ASDialog.java	3	1,00	1	1	0,00

Figura 11 - Relatório exportado para HTML

C:\Documents and Settings\gu\Desktop\APPPERFECT\RESULTADOS\METRICS\AS1.xml - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço C:\Documents and Settings\gu\Desktop\APPPERFECT\RESULTADOS\METRICS\AS1.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
- <Report Name="ResultMetrics">
- <TableData NumRows="1696" NumColumns="6">
- <TableRow RowCount="0">
- <TableColumn Name="Metrics">
<![CDATA[ Classes ]]>
</TableColumn>
- <TableColumn Name="Total">
<![CDATA[ 56 ]]>
</TableColumn>
- <TableColumn Name="Mean">
<![CDATA[ 1,00 ]]>
</TableColumn>
- <TableColumn Name="Maximum">
<![CDATA[ 1 ]]>
</TableColumn>
- <TableColumn Name="Minimum">
<![CDATA[ 1 ]]>
</TableColumn>
- <TableColumn Name="Std.Deviation">
<![CDATA[ 0,00 ]]>
</TableColumn>
</TableRow>

```

Figura 12 – Relatório exportado para XML

A ferramenta também fornece um gráfico com um resumo do projeto avaliado (Figura 13), trazendo os resultados:

- em azul: rigor-médio
- em vermelho: rigor-baixo
- em verde: rigor-alto
- em amarelo: rigor-crítico

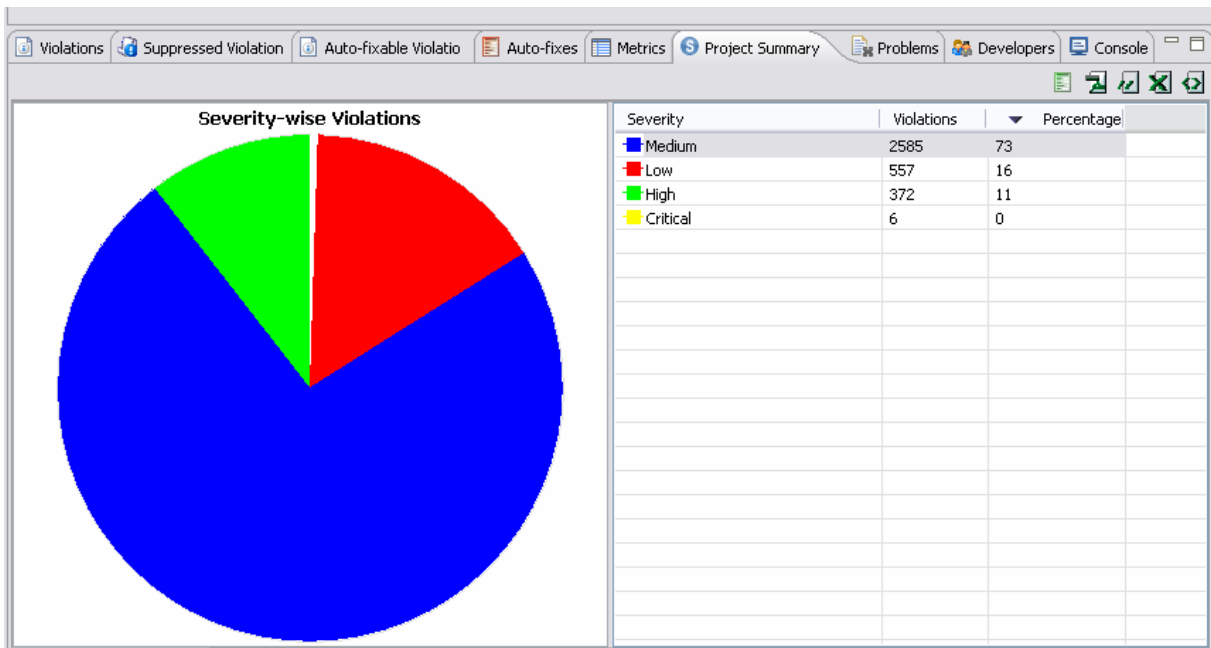


Figura 13 - Resumo do projeto

## 6.4 Ferramenta Eclipse Metrics Plugin

O Eclipse Metrics Plugin é uma ferramenta *open source* que apresenta métricas para programas em Java e funciona como um plugin para a plataforma Eclipse (IDE Java), como ilustrado na Figura 14. Seu objetivo é fornecer métricas e cálculo. Sua principal característica é medir métricas com média e desvio padrão.

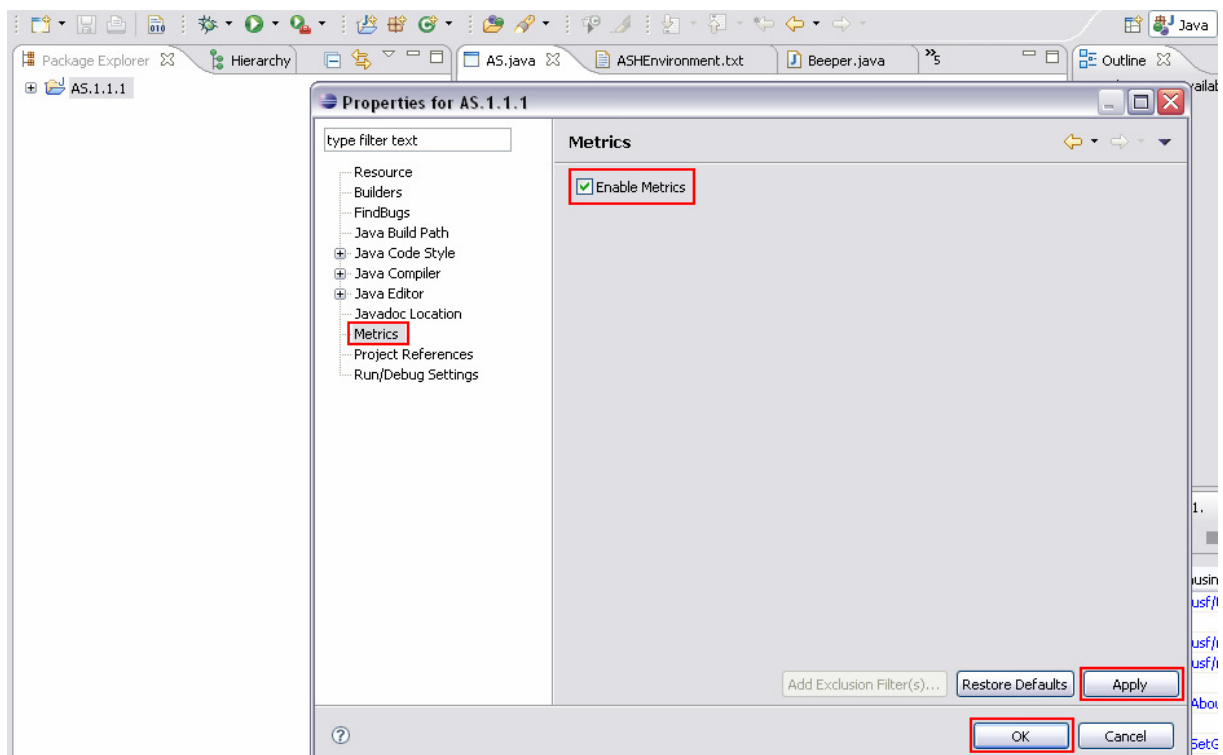
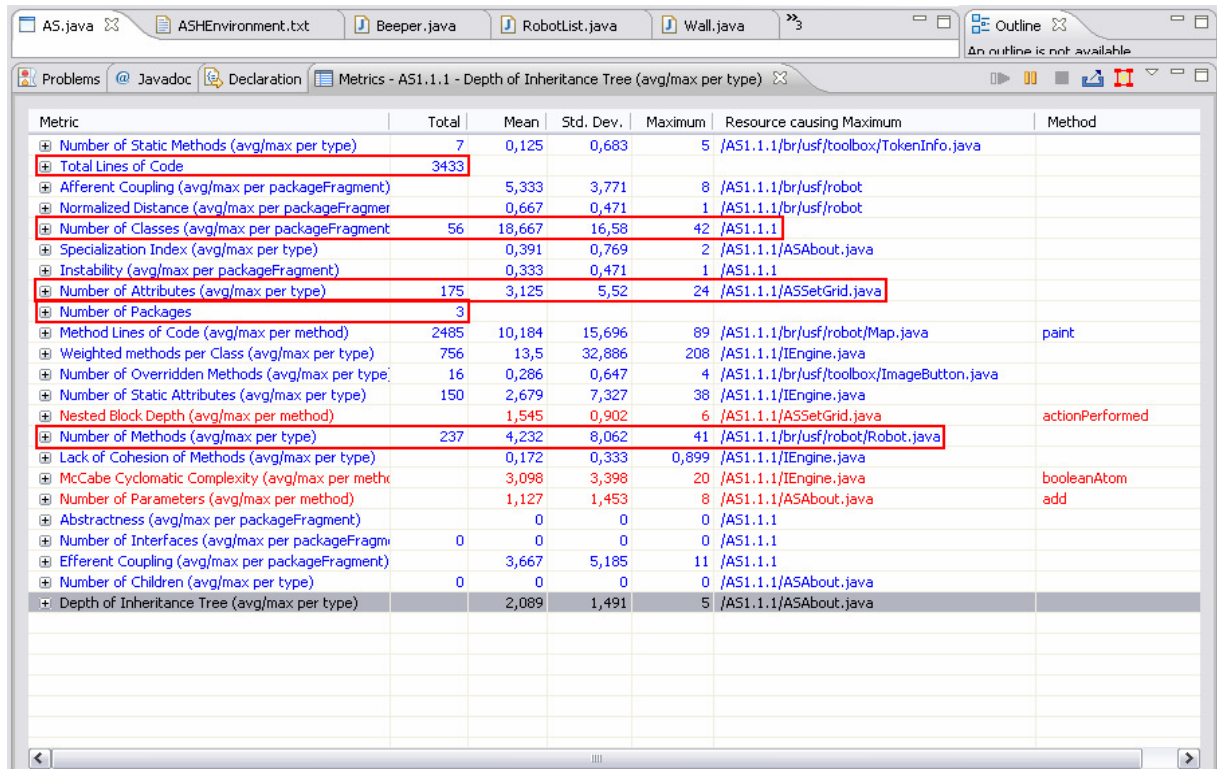


Figura 14 - Execução da ferramenta Metrics

Alguns resultados obtidos com esta ferramenta, ilustrados na Figura 15, são:

- Total de linhas de código;
- Número de classes;
- Número de atributos;
- Número de pacotes; e
- Número de métodos.



Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Static Methods (avg/max per type)	7	0,125	0,683	5	/AS1.1.1/br/ufsf/toolbox/TokenInfo.java	
<b>Total Lines of Code</b>	<b>3433</b>					
Afferent Coupling (avg/max per packageFragment)	5,333	3,771	8	/AS1.1.1/br/ufsf/robot		
Normalized Distance (avg/max per packageFragment)	0,667	0,471	1	/AS1.1.1/br/ufsf/robot		
Number of Classes (avg/max per packageFragment)	56	18,667	16,58	42	/AS1.1.1	
Specialization Index (avg/max per type)	0,391	0,769	2	/AS1.1.1/ASAbout.java		
Instability (avg/max per packageFragment)	0,333	0,471	1	/AS1.1.1		
Number of Attributes (avg/max per type)	175	3,125	5,52	24	/AS1.1.1/ASSetGrid.java	
Number of Packages	3					
Method Lines of Code (avg/max per method)	2485	10,184	15,696	89	/AS1.1.1/br/ufsf/robot/Map.java	paint
Weighted methods per Class (avg/max per type)	756	13,5	32,886	208	/AS1.1.1/Engine.java	
Number of Overridden Methods (avg/max per type)	16	0,286	0,647	4	/AS1.1.1/br/ufsf/toolbox/ImageButton.java	
Number of Static Attributes (avg/max per type)	150	2,679	7,327	38	/AS1.1.1/Engine.java	
Nested Block Depth (avg/max per method)	1,545	0,902	6	/AS1.1.1/ASSetGrid.java		actionPerformed
Number of Methods (avg/max per type)	237	4,232	8,062	41	/AS1.1.1/br/ufsf/robot/Robot.java	
Lack of Cohesion of Methods (avg/max per type)	0,172	0,333	0,899	/AS1.1.1/Engine.java		
McCabe Cyclomatic Complexity (avg/max per method)	3,098	3,398	20	/AS1.1.1/Engine.java		booleanAtom
Number of Parameters (avg/max per method)	1,127	1,453	8	/AS1.1.1/ASAbout.java		add
Abstractness (avg/max per packageFragment)	0	0	0	0	/AS1.1.1	
Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/AS1.1.1	
Efferent Coupling (avg/max per packageFragment)	3,667	5,185	11	/AS1.1.1		
Number of Children (avg/max per type)	0	0	0	0	/AS1.1.1/ASAbout.java	
Depth of Inheritance Tree (avg/max per type)		2,089	1,491	5	/AS1.1.1/ASAbout.java	

Figura 15 – Algumas métricas do Eclipse Metrics Plugin

Em geral, a ferramenta apresenta métricas do projeto como um todo, porém, caso o usuário necessite somente das métricas das classes, tem-se essa opção (Figura 16).

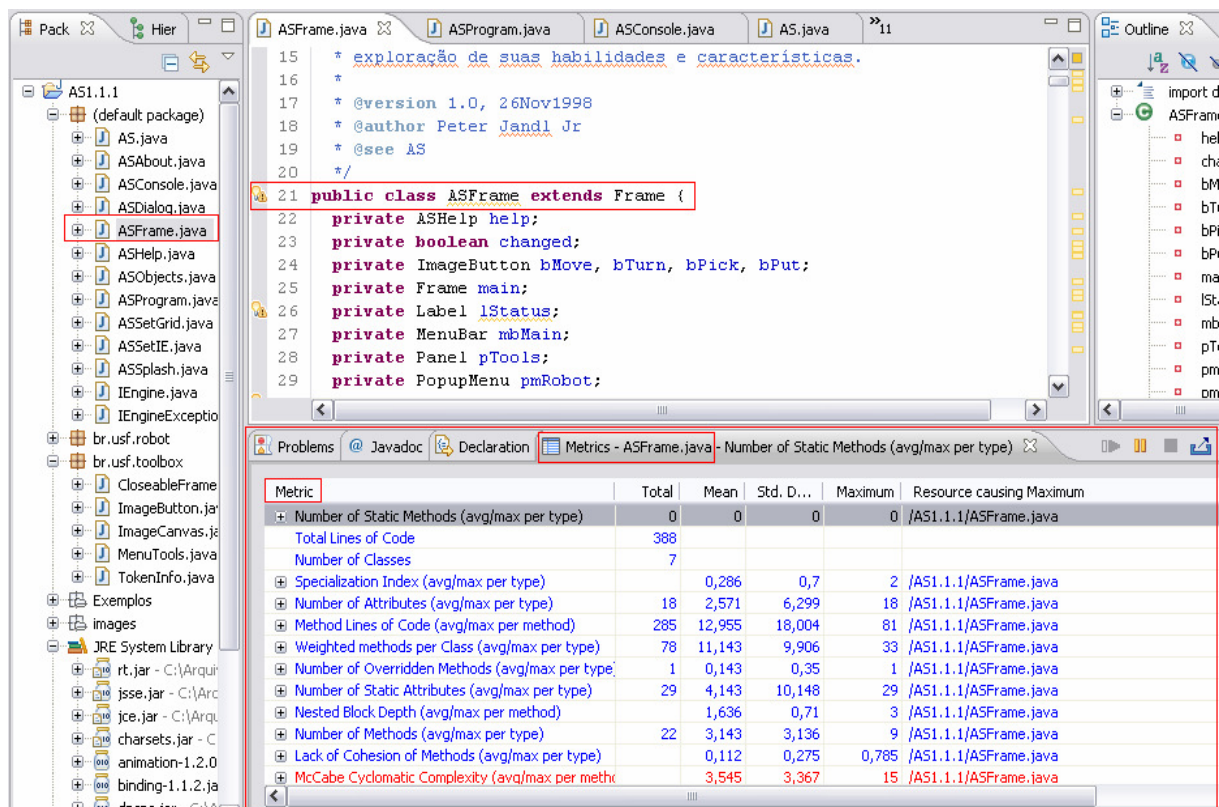


Figura 16 - Métricas da classe Frame

A Figura 17 apresenta um alerta da quantidade de parâmetros do trecho de código, pode ser observado que a ferramenta traz a mensagem em vermelho, para se destacar entre as outras, já que está apontando um alerta.

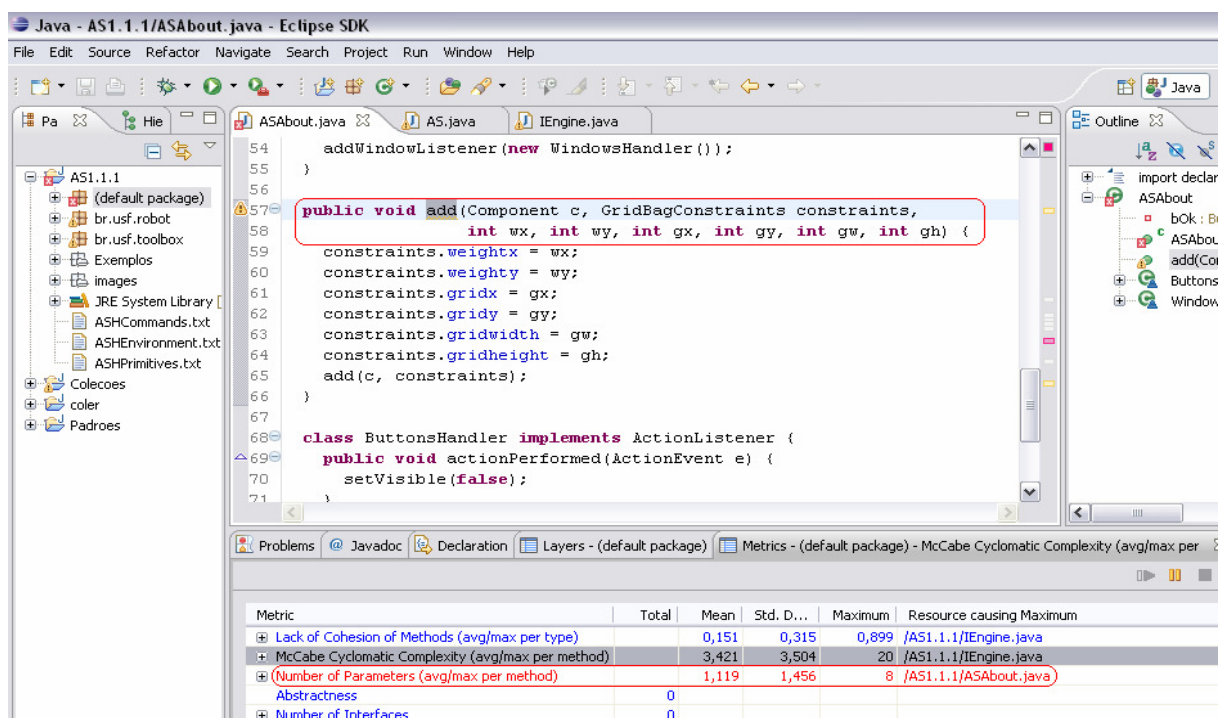


Figura 17 - Alerta de números de parâmetros

A Figura 18 ilustra como a ferramenta traz os erros e advertências do software.

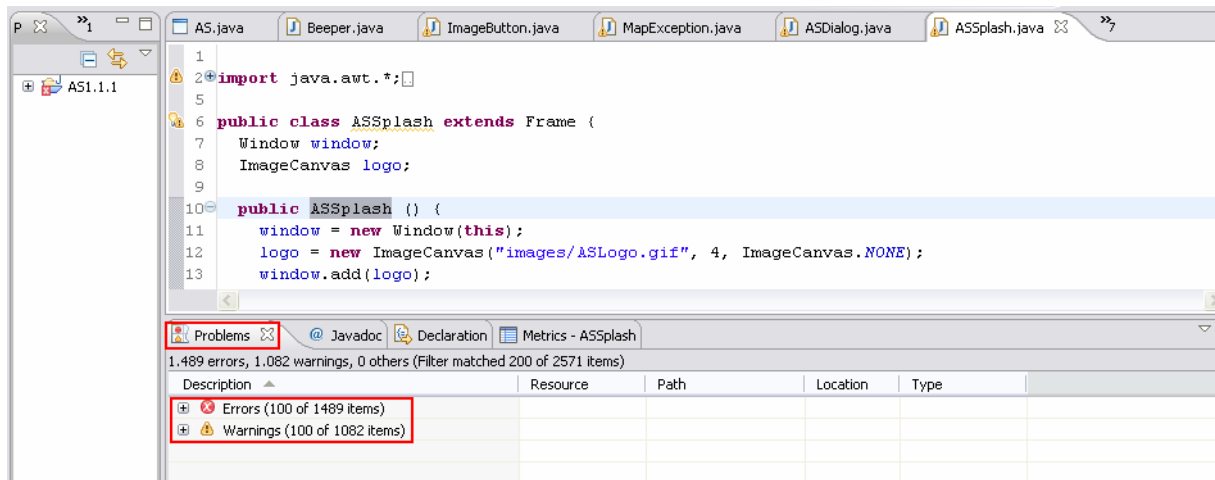


Figura 18 – Erros e advertências

A Figura 19 mostra o *default package* do software, do que ele é composto.

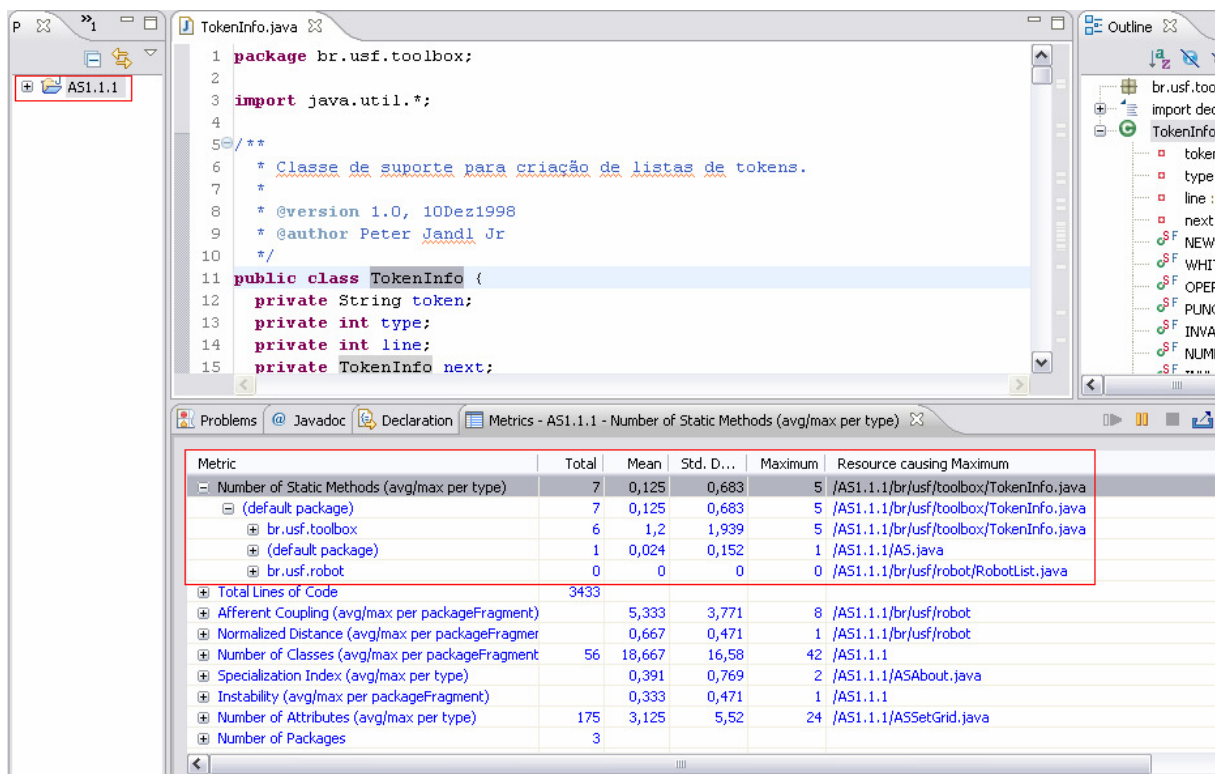


Figura 19 - Pacote padrão do software (default package)

A Figura 20 ilustra um relatório exportado em formato XML.

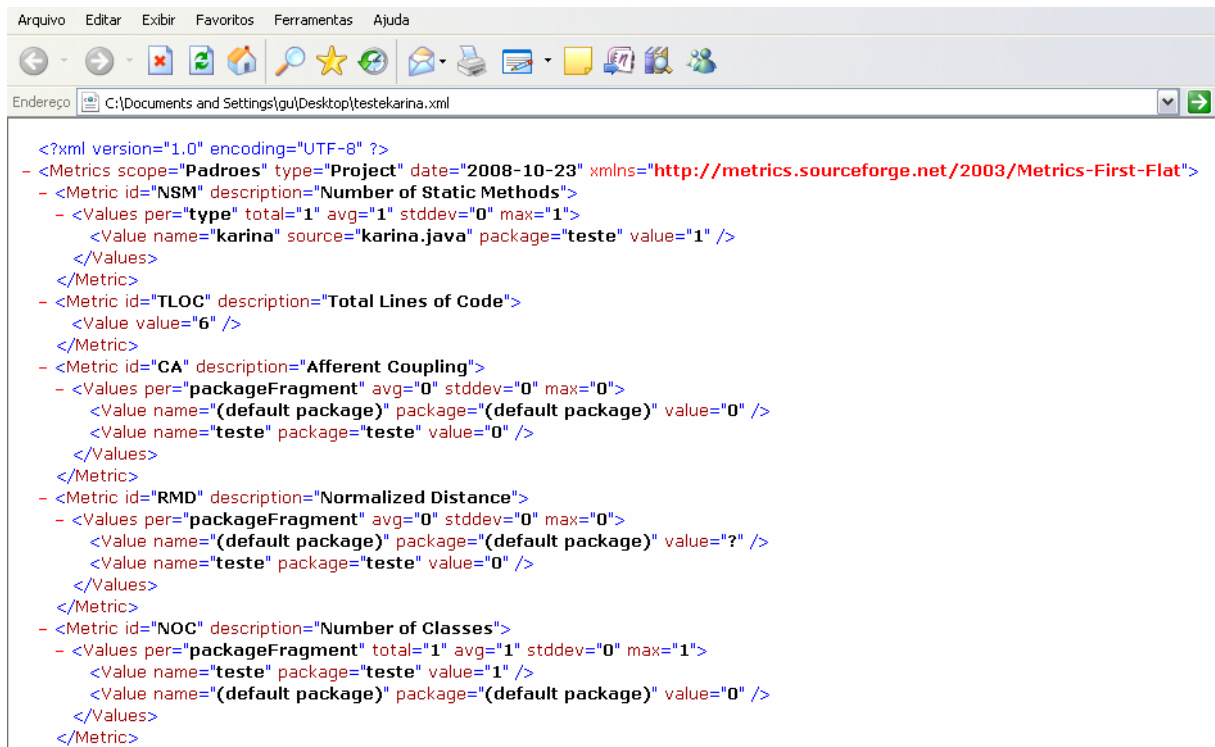


Figura 20 – Relatório exportado para XML

## 6.5 Ferramenta FindBugs

FindBugs também é uma ferramenta open source, utilizada juntamente com a plataforma Eclipse.

Objetivos:

- Encontrar erros em programa Java.

Características:

- Baseia-se em bugs padrões (um padrão é uma maneira de dizer que o código é muitas vezes um erro).
- Utiliza análise estática para analisar bytecodes (arquivo binário Java).
- Gera uma listagem, trazendo um resumo dos números de bugs encontrados.
- Exibe as advertências e as fontes relevantes



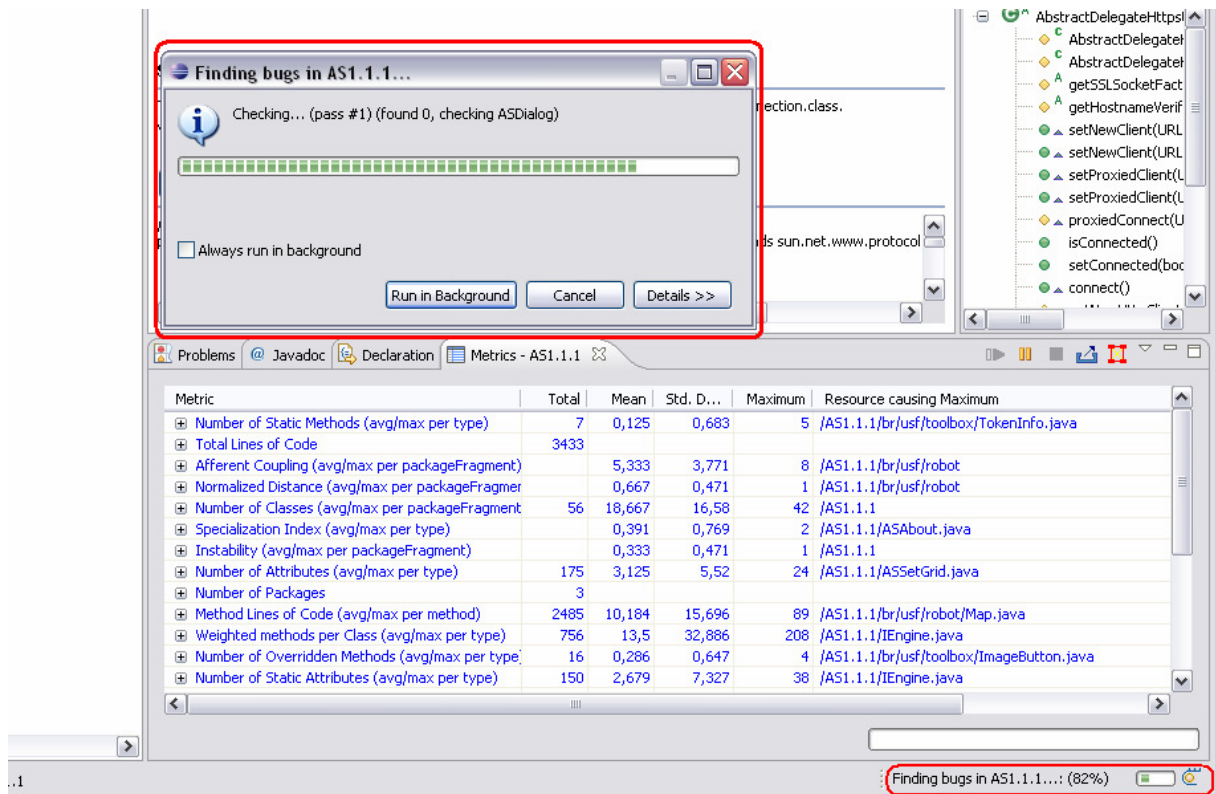


Figura 21 - Execução da Ferramenta FindBugs

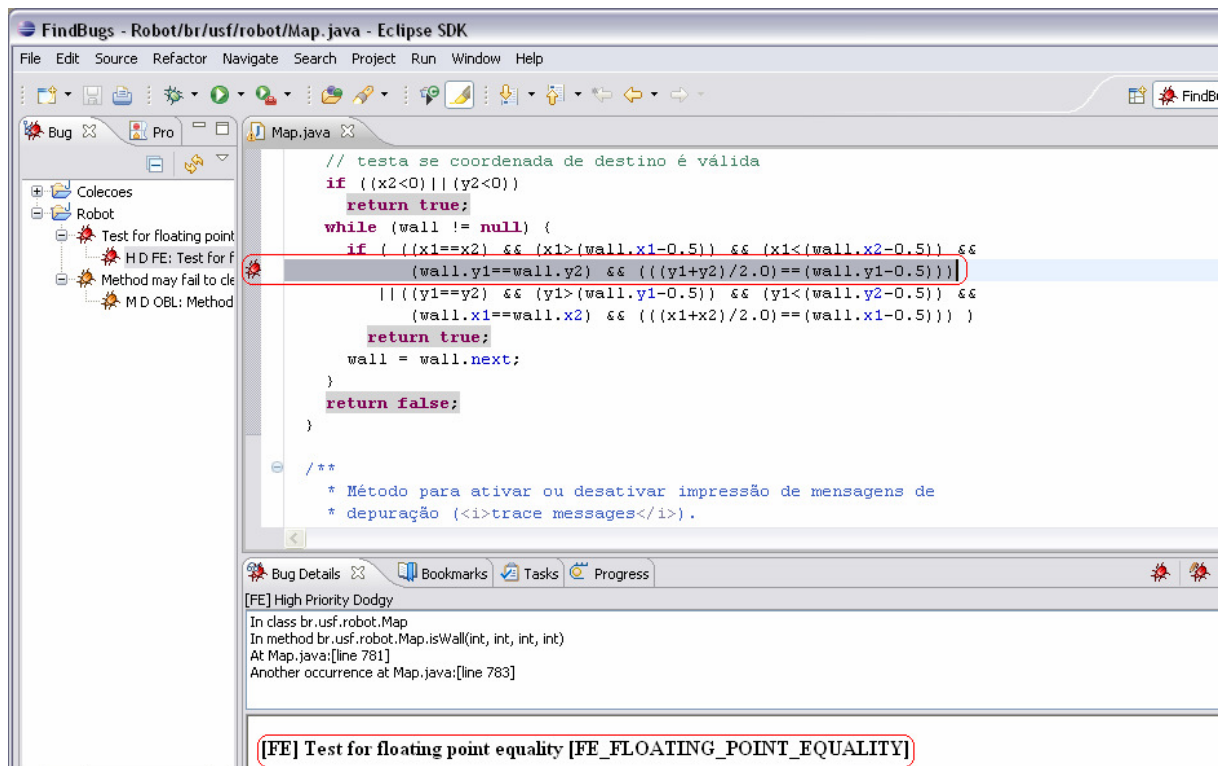


Figura 22 – Teste de ponto flutuante

Teste de ponto flutuante, em igualdade (float)

Esta operação compara dois valores de ponto flutuante para a igualdade, melhor dizendo cálculos de ponto flutuante, pois pode envolver o arredondamento, calculado float e duplos valores podem não ser exatos. Para os valores que devem ser precisos, tais como valores monetários, considere o uso de um determinado tipo de precisão tais como bigdecimal.

Detalhes Bug

Na classe robot.Map

No método robot.Mapa, é parede (int, int, int, int)

Em Map.java [linha 781]

Uma outra ocorrência no Map.java

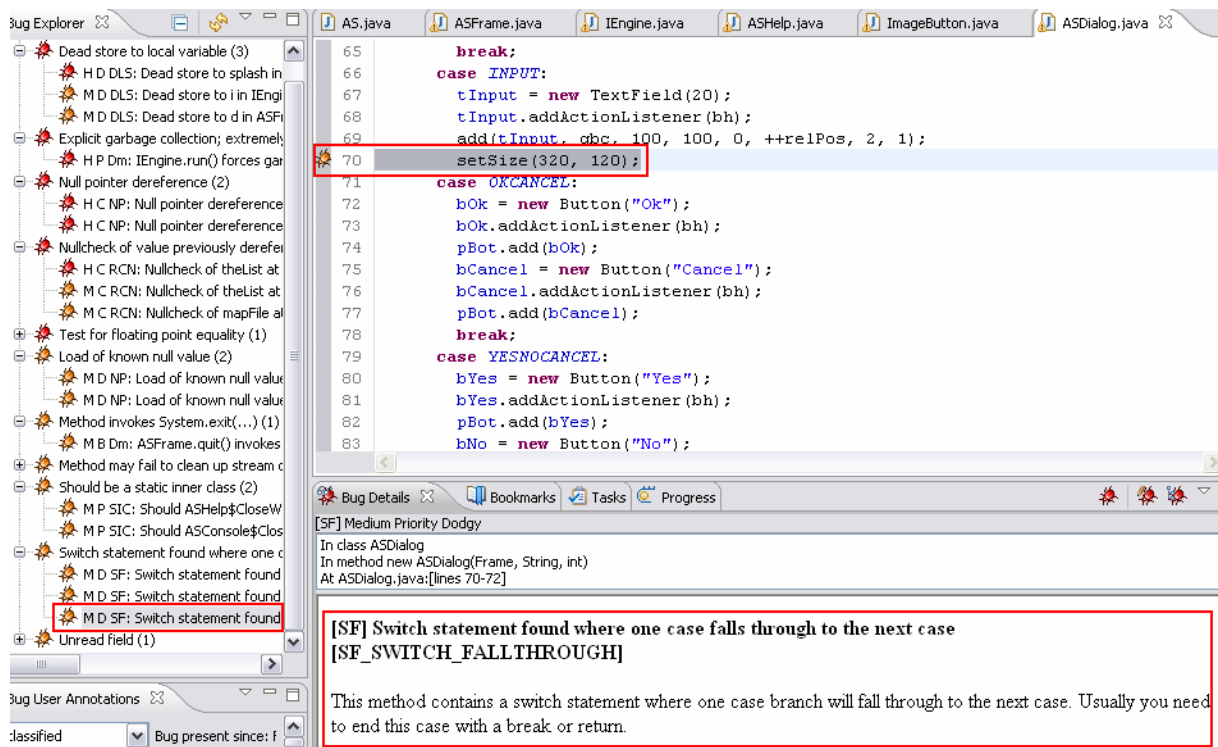


Figura 23 – Método se manifesta

Interruptor - declaração encontrada em um caso em que se registra até o próximo caso

Este método contém uma declaração onde um interruptor se manifesta, caso ramo caia até o próximo caso. Normalmente para eliminar este caso, ele executa um break ou um return.



Detalhes Bug

Na classe ASDialog

No novo método ASDialog (Frame, string, int)

Em ASDialog.java [linhas 70-72]

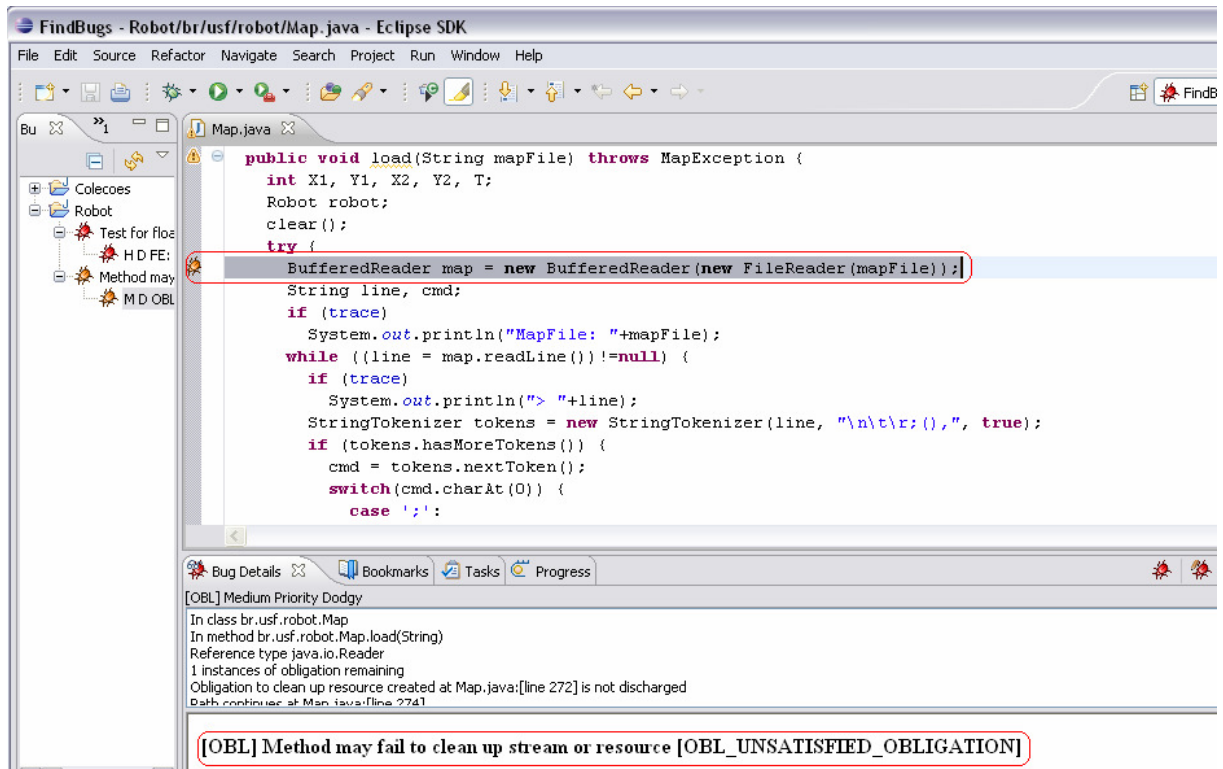


Figura 24 – Alerta que o método pode falhar

Método pode falhar

Este método pode falhar para limpar o banco de dados , ou outros recursos que exigem uma limpeza explícita. Em geral se o método abre um outro recurso, ele deverá usar um “tente um último bloco”, para garantir que o fluxo ou recurso seja limpo antes que o método retorne.

Detalhes Bug

Na classe robot.Map

No método robot.Map.load (String)

Referência tipo `java.io.Reader`

1 caso de obrigação remanescente

Obrigação de limpar recurso criado pelo mapa [linha 272] não é descarregada

Caminho continua no mapa [linha 274]

Caminho continua no mapa [linha 276]

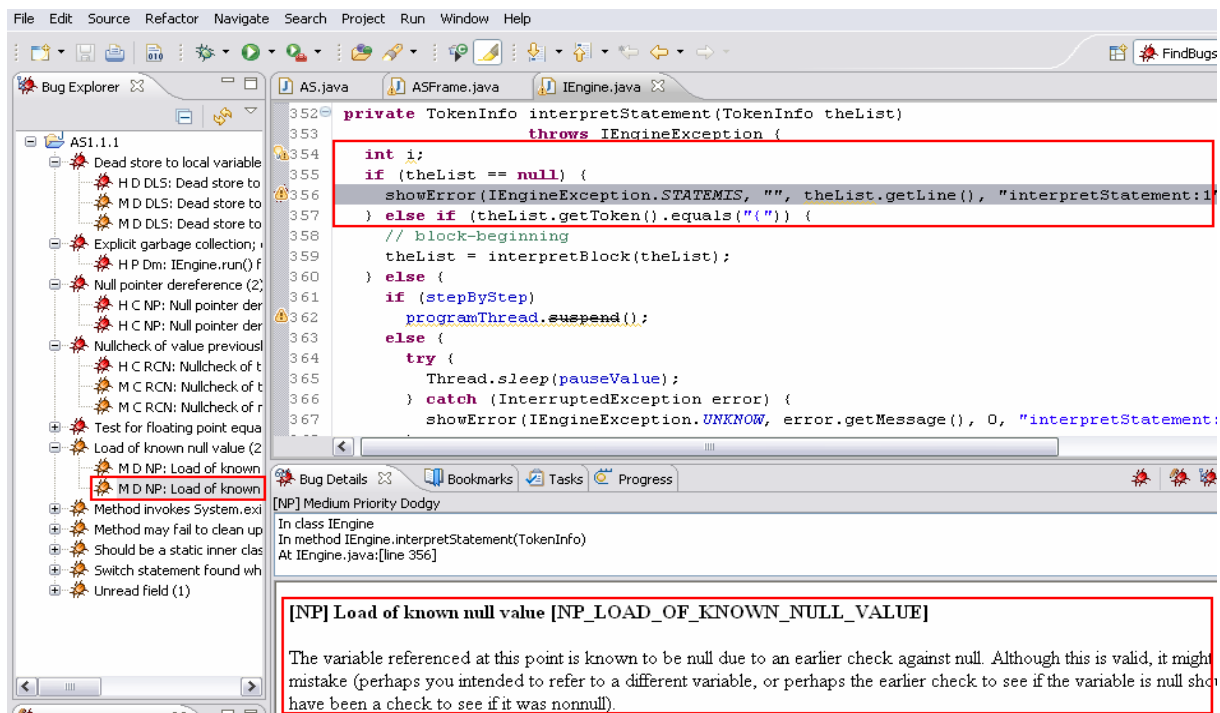


Figura 25 – Variável com valor nulo

Carga de valor nulo em conhecer variável IEngine

A variável referenciada nesta altura, é conhecida como um valor nulo, devido a uma verificação antecipada de encontrar valores nulos. Embora isso seja válido, pode ser um erro.

Detalhes Bug

Na classe IEngine

No método IEngine.interpretStatement

Em IEngine.java [linha 356]

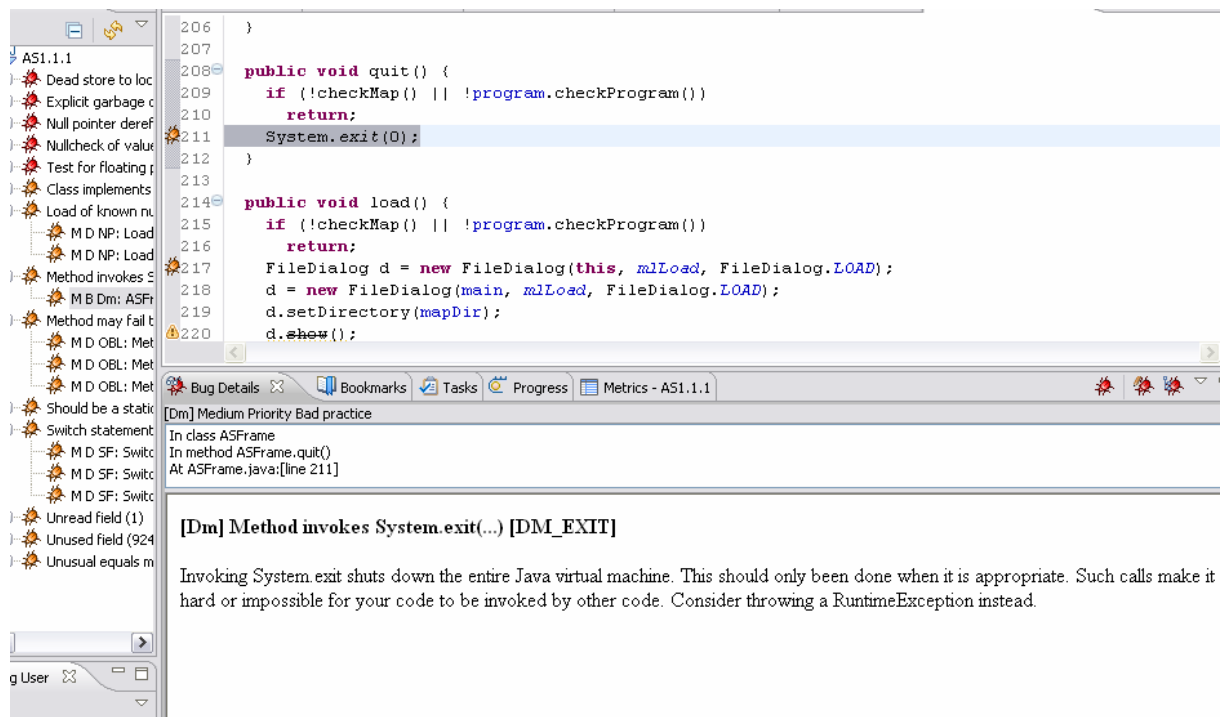


Figura 26 – Chamar método desliga máquina virtual Java

Isso somente deve ser feito se for apropriado, tais chamadas torna difícil ou impossível para este código de ser chamado por outro. O aplicativo recomenda lançar a exceção `RuntimeException` ao invés do uso de `System.exit()`.

Detalhes Bug

Na classe `ASFrame`

No método `ASFrame.quit()`

Em `AS Frame.java` [linha 211]

## 7 COMPARAÇÃO DOS RESULTADOS

### 7.1 Tabela de resultados

A Tabela 02 sumariza os resultados obtidos pelas ferramentas AppPerfect, Metrics e FindBugs.

Tabela 02 – Resultados quantitativos das métricas

Quesito / Métrica	AppPerfect	Metrics	FindBugs
1. Número de linhas de código	6.507	3.433	N/D
2. Número de atributos	137	175	N/D
3. Número de métodos	244	237	N/D
4. Numero de classes	56	56	N/D
5. Número de pacotes	N/D	3	N/D
6. Número de interfaces	N/D	0	N/D
7. Teste ponto flutuante	N/D	N/D	Não apresenta valores, apenas aponta as advertências e lista os bugs
8. Alerta: método pode falhar	N/D	N/D	Não apresenta valores, apenas aponta as advertências e lista os bugs.
9. Alerta: variável com valor nulo	N/D	N/D	Não apresenta valores, apenas aponta as advertências e lista os bugs.

Como pode ser observado na Tabela 02:

#### Resultados diferentes:

##### *Quesito 1*

Possui o mesmo tipo de métrica, mas com resultados diferentes

##### *Quesito 2*

Possui o mesmo tipo de métrica, mas com resultados diferentes

##### *Quesito 3*

Possui o mesmo tipo de métricas, mas com resultados diferentes

##### *Quesito 5*

Ferramenta AppPerfect e FindBugs não possui a métrica da ferramenta Metrics

##### *Quesito 6*

Ferramenta AppPerfect e FindBugs não possui a métrica da ferramenta da ferramenta Metrics

*Quesito 7*

Ferramenta AppPerfect e Metrics não possui a métrica da ferramenta da ferramenta FindBugs

*Quesito 8*

Ferramenta AppPerfect e Metrics não possui a métrica da ferramenta da ferramenta FindBugs

*Quesito 9*

Ferramenta AppPerfect e Metrics não possui a métrica da ferramenta da ferramenta FindBugs

**Resultados iguais:**

*Quesito 4*

Métricas iguais com valores iguais

## 7.2 Quadro comparativo

A Tabela 03 faz uma comparação dos resultados das métricas

Tabela 03 – Resultados descritivos das métricas

FERRAMENTA	RESULTADO
AppPerfect	Traz algumas métricas que a ferramenta Metrics também possui.
Metrics	Traz algumas métricas que a ferramenta AppPerfect não possui.
FindBugs	Traz métricas diferentes das ferramentas AppPerfect e Metrics.

## 7.3 Avaliação geral

Na Tabela 04 podem ser observadas uma breve avaliação de cada uma das ferramentas

Tabela 04 – Avaliação das métricas

<b>AppPerfect</b>	
INSTALAÇÃO	OK
OPEN-SOURCE	SIM
USO	FÁCIL UTILIZAÇÃO
COERÊNCIA DOS RESULTADOS	OK
UTILIDADE DOS RESULTADOS	POSITIVO PARA O SOFTWARE
<b>Metrics</b>	
INSTALAÇÃO	OK
OPEN-SOURCE	SIM
USO	FÁCIL UTILIZAÇÃO
COERÊNCIA DOS RESULTADOS	OK
UTILIDADE DOS RESULTADOS	POSITIVO PARA O SOFTWARE
<b>FindBugs</b>	
INSTALAÇÃO	OK
OPEN-SOURCE	SIM
USO	FÁCIL UTILIZAÇÃO
COERÊNCIA DOS RESULTADOS	NÃO TRAZ MÉTRICAS EM VALORES
UTILIDADE DOS RESULTADOS	POSITIVO PARA O SOFTWARE

## 7.4 Conclusão das ferramentas e das métricas

Com as ferramentas utilizadas, todas de open-source e fácil utilização, concluiu-se que, elas são distintas, com métricas diferentes uma das outras, trazendo vários resultados da medição do software.

## 8 CONCLUSÕES

Colocando todo o estudo em prática, pode-se perceber que existem muitas ferramentas e diversos tipos de métricas, facilitando as organizações, de iniciarem um processo de medição do software.

Após o uso de cada uma delas, pode ser feita a comparação entre ambas, a diferença que existe uma da outra, as qualidades, os tipos de métricas, facilidade de uso e, por conseguinte, analisar os resultados. Com os resultados, foi possível também uma comparação onde pode ser observado que as métricas trouxeram valores diferentes, diferenciando-se uma das outras.

Foi possível mostrar a importância de se medir um software, encontrar bugs através das ferramentas, apontar falhas e exibir as advertências. Com todas essas funções, pode-se dizer que um software depois de ter sido verificado por elas está pronto para as correções, e em seguida, quando estiver realmente corrigido e testado novamente, está apto para a entrega.

Os resultados esperados foram positivos, pois foi realmente mostrado o que são as métricas de software, porque utilizá-las, mostrou-se como medir um software, foi mostrado como as métricas são importantes, e como se inicia o processo de medição.

As ferramentas também contribuíram para este trabalho, mostrando as qualidades e funcionalidades existentes, cada métrica contida na ferramenta, trouxe um ponto positivo para o software, com elas foram possíveis mostrar que um software pode ser verificado linha por linha do código, analisando cada trecho. Mas, por outro lado, também trouxe alguns pontos negativos, as ferramentas trouxeram algumas falhas, erros e alertas, que na verdade podem ser considerados bons, pois são com esses avisos de falhas e erros de software que se tem um produto de qualidade, como foi citado nos primeiros capítulos, as métricas não apresentam 100% de garantia nos resultados.

Sem a análise das ferramentas, seria quase impossível descobrir esses erros, esse é um dos fatores principais das ferramentas de métricas, encontrar *bugs*, um outro fator importante é a busca das métricas diretas, indiretas e métricas orientadas á tamanho, que mostraram o software em relação ao seu conteúdo, fazendo a validação e garantindo a qualidade.

## 9 REFERÊNCIAS BIBLIOGRÁFICAS

- APPPERFECT. **Open Source Software**. Disponível em: [http://sourceforge.net/search/?type\\_of\\_search=soft&words=appperfect](http://sourceforge.net/search/?type_of_search=soft&words=appperfect)>. Acesso em: 01 de out. de 2008.
- CLARO, Daniela B. **Métricas de Software**. Disponível em: <http://www.inf.ufsc.br/~danclaro/download/disciplinas/M%E9tricas%20de%20Software.doc>>. Acesso em: 20 de fev. de 2008.
- CORDEIRO, Marco Aurélio. **Métricas de Software**. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/2000/bb101/metricas.htm>>. Acesso em 20 de fev. de 2008.
- FINDBUGS. **Open Source Software**. Disponível em: [http://sourceforge.net/search/?type\\_of\\_search=soft&words=findbugs](http://sourceforge.net/search/?type_of_search=soft&words=findbugs)>. Acesso em: 01 de out. de 2008.
- GOMES, Alvaro Eduardo. **Métricas e Estimativas de Software – O início de um rally de regularidade**. Disponível em: <http://www.apinfo.com/artigo44.htm>>. Acesso em: 20 de fev. de 2008.
- GOMES, Andrey. **Metodologias de Desenvolvimento de Software**, 2008 Disponível em: [http://www.andreygomes.com/index.php?option=com\\_content&view=article&id=1:metodologias-de-desenvolvimento-de-software&catid=1:metodologias&Itemid=2](http://www.andreygomes.com/index.php?option=com_content&view=article&id=1:metodologias-de-desenvolvimento-de-software&catid=1:metodologias&Itemid=2)>. Acesso em: 06 de nov. de 2008.
- IEEE. Transactions on Software Engineering, 1998. Acesso em: 15 de jul. de 2008.
- IFPUG. **Function point counting practices manual: V4.0**. Atlanta, 1994. Acesso em: 22 de ago. de 2008.
- JANDL, Peter. **Introdução ao Java**, 1999.
- METRICS PLUGIN, Eclipse. **Open Source Software**. Disponível em: [http://sourceforge.net/search/?type\\_of\\_search=soft&words=eclipse+metrics+plugin](http://sourceforge.net/search/?type_of_search=soft&words=eclipse+metrics+plugin)>. Acesso em: 01 de out. de 2008.
- PRESSMAN, Roger S. **Engenharia de Software**, São Paulo: Makron Books, 1995.
- SOMMERVILLE, I. **Engenharia de Software**. 6ª Ed. São Paulo: Addison Wesley, 2003.
- TIOBE. **Programação comunitária índice de novembro de 2008**. Linguagens populares. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>>. Acesso em: 11 de set. de 2008.
- VASCONCELOS, Alexandre. **Métricas de Software**, 2005. Disponível em: <http://www.cin.ufpe.br/~if720/slides/introducao-a-metricas-de-software.ppt>>. Acesso em: 20 de fev. de 2008.