Métricas de Código

Universidade de Brasília - Campus Gama

Gestão de Portfólios e Projetos

Caio Felipe Dias Nunes 14/0133305

João Paulo Busche da Cruz 14/0023348

Vinicius Pinheiro da Silva Corrêa 14/0066543

# 1. Processo de Medição de Software

Conforme Gava et al (2006) apud Vazquez et al (2003) o termo medida é por definição a quantificação de uma característica e o termo medição corresponde a um conjunto de operações tendo como objetivo a determinação de uma medida. Implementar medições de software é uma tarefa complexa, pois exige a tomada de decisão, sobre quais medidas podem ser utilizadas e quais são úteis dentro de um projeto de software. Gava et al (2006) acredita que essa tarefa deve ser implementada segundo orientação de uma visão prática.  
 As experiências com vários projetos de desenvolvimento e manutenção sugerem, segundo Gava et al (2006), duas características que podem auxiliar no processo de medição, sendo elas:

* A coleta, análise e a comunicação dos dados oriundos da medição devem ter um relacionamento direto com as necessidades de informação dos tomadores de decisão. Esta característica propõe que a solução de medição pode ser definida como uma solução guiada pela necessidade de informação, onde sua implementação é orientada a resolução destas necessidades. Junto a este entendimento, é necessário definir sempre quais informações são necessárias serem analisadas, além de como que realmente deverá ser feito as medições (Gava et al, 2006 apud BASILI, 1992 MCGARRY, 2002 PSM, 2005);
* O processo de medição deve contemplar através de um modelo as atividades necessárias para especificar adequadamente qual informação de medição é importante, como as medições e os resultados da análise serão aplicados e como determinar se os resultados das análises são válidos. O processo deve ser flexível e adaptável, pois há diferentes necessidades em cada projeto (Gava et al, 2006 apud MCGARRY, 2002 PSM, 2005).

# 2. A importância de fazer medições

Medições são extremamente importante no âmbito de qualidade. Quando se fala de código de qualidade, produtos que estejam em boas condições, não existe a possibilidade de se chegar a este resultado sem medições. As medições seguem uma linha clara de , baseado em dados coletados e reais, identificar problemas e sugerir melhorias em diversos contextos.

É importante que um software seja medido em diversos aspectos diferentes, para que se tenha uma análise mais precisa da qualidade do mesmo. Quando se trabalha com software no mundo real, em que ele realmente possui um propósito e é desenvolvido para um público ou usuário específico, ter noção de onde podem haver melhorias e problemas é extremamente relevante para a organização como um todo.

# 3. Métricas Fundamentais

## 3.1. Tamanho

Métrica presente em todas as abordagens, devido a sua utilização pra estimação de prazos, custos, esforço, etc. Além disso, serve para normalização, permitindo comparações entre projetos diretamente. Ainda, em conjunto com outras métricas, pode trazer métricas derivadas extremamente relevantes.

Ex: densidade de falhas = num falhas/ tamanho

Existem diversas métricas de tamanho, adequadas à tipos distintos de projetos, sendo algumas:

* Pontos de Função
* Linhas de Código
* Pontos de Caso de Uso
* Cosmic Function Points

A desvantagem é que, por ser tratar de uma medida básica, não é passível de análise se não for combinada com outras medidas básicas.

## 3.2. Esforço

Medida básica, que representa contagem do número de horas que uma ou mais pessoas utilizam para executar determinada tarefa/projeto. Define-se um nível de medida desejado ( esforço por tarefas, atividades, fases, projetos, etc), e então registra-se as horas associadas a este nível. Soma-se estas horas e então, tem-se o esforço definido. O mesmo pode ser calculado por pessoa-mês, pessoa-dia, etc.

As medidas de esforço são úteis para gerar históricos de esforço nos projetos que podem ser utilizados em futuras estimativas de tempo, custo, etc em que serão mantidas as mesmas pessoas de uma equipe. Também servem para determinar a produtividade da equipe, e realizar tomadas de decisões baseadas nisso.

A grande dificuldade de se trabalhar com métricas de esforço é a confiabilidade dos dados, pois envolve o lado humano da questão, as horas realmentes dedicadas a determinada atividade, etc e vai além do conhecimento da medição em si.

## 3.3. Duração

Medida básica que determina o tempo de duração das atividades do projeto. É elaborado o conceito de “pronto” pela equipe, e então é registrada uma data de início e de conclusão, sendo medidos em diferentes níveis ( dias, semanas, meses).

São utilizadas para gerar históricos da duração de projetos, que serão úteis em estimativas futuras, bem como utilizados para indicadores e análises de monitoramento de projetos. É primordial que nesse tipo de medição, o conceito do que é considerado “pronto” ou não esteja claro, para evitar problemas com a medição, pois somente quando for aferido que a tarefa está realmente pronta, de acordo com os critérios, é que será contabilizada à medição.

## 3.4. Custo

Medida derivada (indireta) , que representa o valor monetário necessário para executar um projeto, ou partes deste. Calcula-se convertendo o esforço em horas, multiplicando estas em horas pelo custo de cada uma e somando-se os outros custos (infraestrutura, treinamentos, custos administrativos, etc).

As maiores dificuldades encontradas nas medidas de custo são profissionais que apoiam diversos projetos simultaneamente, co-relação entre níveis organizacionais de um projeto e destes níveis com outros projetos simultaneamente, custo de horas que não estão associadas a um único projeto somente, horas utilizadas em reuniões e atividades similares, etc.

## 3.5. Progresso

Indicador que indica a situação do andamento do projeto em relação ao prazo do mesmo. Mostra se o projeto está atrasado, no prazo ou adiantado, e pode ter foco em atividades ou produtos.

São utilizadas para o cálculo de progresso as seguinte estimativas:

* Data prevista para conclusão de atividade (estimativa)
* Data prevista para conclusão de atividade(medida básica)
* Duração (em dias) prevista para atividade

Em projetos iterativos, ou com muitas atividades sendo realizadas paralelamente, é complexo acompanhar o progresso por produtos (entregas, etc), sendo priorizada monitoramento de atividades ou produtos em caminho crítico. É importante por isso, que os indicadores sejam capturados rapidamente, com atividades bem granularizadas, permitindo controle efetivo.

## 3.6. Qualidade

Métricas de qualidade podem ser analisadas de pontos de vistas diferentes, dependendo do usuário ou conhecimento sobre o sistema. Na perspectiva de cliente, a qualidade está atrelada a satisfação do cliente em relação ao produto oferecido. Representa um indicador que indica a percepção dos usuários em relação ao produto ou serviço. Podem ser analisadas por entrevistas, questionários, etc, e apresentadas de maneiras gráficas por tabelas, gráficos de satisfação, etc.

É possível analisar a qualidade também, do ponto de vista da quantidade de defeitos que o sistema tem. Isso implica em analisar o sistema/serviço oferecido em diferentes fases do projeto e verificar a evolução do mesmo ao longo do tempo, garantindo que a quantidade de defeitos seja a menor possível.

Isto pode ser calculado através da medição básica da quantidade de defeitos, e derivando essas medições, podem ser verificada densidade dos defeitos, falhas, etc. Esses indicadores de defeito são úteis para monitorar e controlar qualidade dos produtos, monitorar o projeto em si e monitorar processos de verificação, validação e etc.

E por último, também é possível analisar a qualidade de um sistema baseado na sua confiabilidade, com propósito de medir uma taxa de falhas do sistema em operação. Podem ser detectadas medidas básicas como tamanho em KLOC da aplicação pronta, número de falhas no tempo, etc, e medidas derivadas como Taxa de Falhas, Densidade de Falhas, etc. E ainda, indicadores como gráficos com as densidades e taxas esperadas, etc.

Essa análise é utilizada como indicador da capacidade do processo produzir produtos que apresentem poucas falhas, bem como análise da efetividade das técnicas de redução de defeitos.

# 4. Como definir as Métricas

## 4.1. Plano GQM (*Goal - Question - Metrics*)

O *GQM* é um modelo que foi originalmente definido para a avaliar defeitos, para um conjunto de projetos da *NASA Goddard Space Flight Center*. A aplicação envolve um conjunto de casos de experimento de estudo e depois foi expandida para incluir vários casos de experimentos de abordagem. Embora o modelo originalmente costumava definir objetivos e avaliar objetivos de um determinado projeto, num determinado ambiente, ele foi expandido para um contexto maior.

O *GQM* é um modelo baseado no pressuposto de que uma organização para medir de uma forma importante, deve primeiro especificar seus objetivos e seus projetos, então deve traçar os objetivos para os dados que vão definir esses objetivos operacionalmente, e finalmente fornecer um *framework* para interpretação destes dados, relacionados aos objetivos estabelecidos.

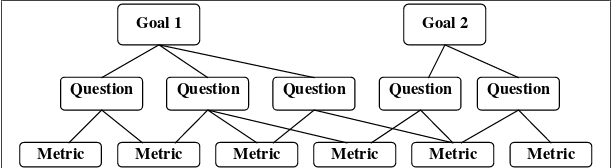


Figura X - Estrutura da hierarquia do modelo GQM

### 4.1.1. Definir Objetivos (*Goal*)

Objetivos são definidos por razões variadas, a um objeto, respeitando vários modelos de qualidade, com diferentes pontos de vista, relacionados a um ambiente particular. Os objetos de medição são:

* **Produtos:** Artefatos, entregavéis e documentos que são produzidos durante o ciclo de vida do sistema. Ex.: Especificações, casos de testes, programas, etc.
* **Processos:** Atividades relacionadas ao *software*, geralmente associadas com o tempo. Ex.: Testes, entrevistas, etc.
* **Recursos:** Itens utilizados no processo para produzir as saídas desejadas. Ex.: Humanos, *hardwares, softwares,* etc.

### 4.1.2. Definir Questões (*Question*)

Um conjunto de questões que são utilizadas para caracterizar o caminho que será realizado para cumprir um objetivo específico, baseado na caracterização padrão. Questões tentam caracterizar um objeto de medição, para selecionar métricas de qualidade e determinar sua qualidade, a partir de um determinado ponto de vista.

### 4.1.3. Definir Métricas (*Metrics)*

Um conjunto de dados associados a cada questão, com o objetivo de respondê-las de forma quantitativa. Os dados podem ser caracterizados em:

* **Objetivos:** Dependem apenas do objeto a ser medido, e não do ponto de vista, do qual são determinados. Ex.: números de versões de um documento, tamanho de um programa, etc.
* **Subjetivos:** Dependem tanto do objeto a ser medido, tanto do ponto de visto, do qual são determinados. Ex.: nível de satisfação de um cliente, etc.

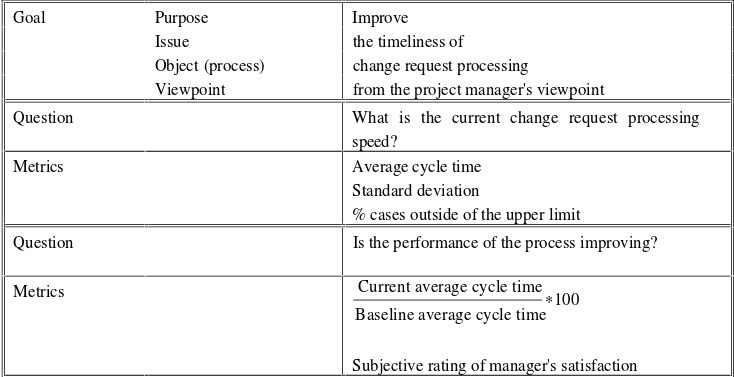


Figura X - Resultado padrão do modelo GQM

## 4.2. Plano *PSM* (*Practical Software and Systems Measurement*)

O *PSM* é um modelo que surgiu em 1994 a partir de uma iniciativa do Departamento de Defesa estadunidense (McGarry et al., 1997) e que foi publicado apenas em 1997 na forma de um manual, que vem sendo atualizado e hoje está em sua quarta edição(Aguiar, 2002). O *PSM* foi utilizado para servir de base para a formulação da área “Measurement and Analysis” do *CMMI*, que viria a ser sucessor do *CMM*, e está formalizado sobre a norma ISO/IEC 15.939 - *Software Engineering - Software Measurement Process Framework*(Aguiar, 2002)*.*

O *PSM* é um modelo focado na estruturação da mensuração em um projeto de software e tem seu foco na resolução de dois problemas, sendo eles: como especificar formalmente as medidas a serem utilizadas; como conduzir o processo de medição. Para alcançar estes objetivos, o *PSM* utiliza dois modelos: modelo de informação e modelo de processo

### 4.2.1. Modelo de Informação

O modelo de informação foi adotado para a resolução do problema existente de como especificar formalmente as medidas a serem utilizadas(Aguiar, 2002).

Segundo Gava et al (2006), o modelo de informação fornece um mecanismo formal para unir as necessidades de coleta de informação definidas no processo de engenharia de software aos produtos que podem ser mensurados. Pode ser visto na figura abaixo como que a necessidade deve evoluir para um plano de medição.

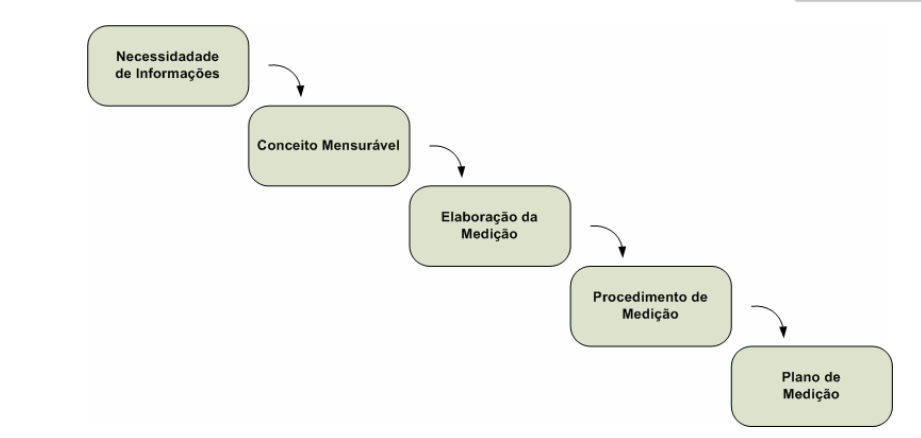


Figura X - Evolução a partir da necessidade de informações até o plano de medição. Adaptado do PSM (Gava et al, 2006)

Segundo Gava et al (2006) para se iniciar o planejamento de medição, inicialmente é necessário reconhecer as necessidades de informações que existem, pois as informações dão um suporte à tomada de decisões que atingem tanto no âmbito operacional quanto nos estratégicos e organizacionais. Essas informações úteis devem poder ser adquiridas através da medição de diferentes elementos.

O conceito mensurável é uma ideia sobre as entidades que deveriam ser medidas com a finalidade de satisfazer as necessidades de informações que foram levantadas (Gava et al, 2006).

A elaboração da medição consiste em formalizar o conceito mensurável, especificando o que será medido e como os dados gerados e combinados podem satisfazer as necessidades de informações, tornando-o o construtor de medição (Gava et al, 2006).

O procedimento de medição tem a finalidade de definir os mecanismos que farão a coleta e organização dos dados para o construtor de medição (Gava et al, 2006).

O Plano de Medição então é gerado através da união das necessidades de informações aplicáveis, construtores de medição e procedimentos de medição. A execução desse plano deve definir em seu corpo os artefatos resultantes (Produto de Informação) com o resultado desta execução. Nesses artefatos resultantes é importante que reúnam os indicadores, interpretações e recomendações resultantes da saída (Gava et al, 2006).

Segundo Aguiar (2002) os seguintes conceitos são definidos no *PSM:*

* Atributo: propriedade característica de uma entidade de software: podem ser processos, produtos (artefatos), projetos ou recursos;
* Medida base: medida de um único atributo definida por um método de medição específico, cuja execução fornece o valor da medida;
* Medida derivada: medida ou quantidade definida como uma função de duas ou mais medidas base, ou medidas derivadas;
* Indicador: é a base quantitativa para a análise das medições e para as tomadas de decisão;
* Produto de informação: dados consolidados das medições em um artefato, portanto, é o que deve ser apresentado aos usuários.

O modelo de informação do PSM pode ser representado de maneira simplificada pela figura abaixo

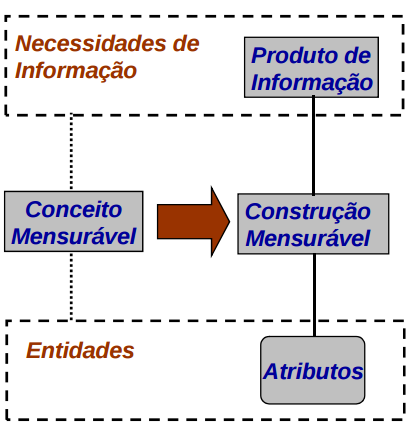


Figura X - Modelo de Informação PSM. (Aguiar, M., sd)

A construção mensurável pode ser apresentada graficamente, apresentando desde o atributo que irá ser medido, até o indicador que apresenta informações que são úteis para resolução da necessidade de informação. A figura auto-explicativa abaixo apresenta o modelo da representação gráfica para esta construção.

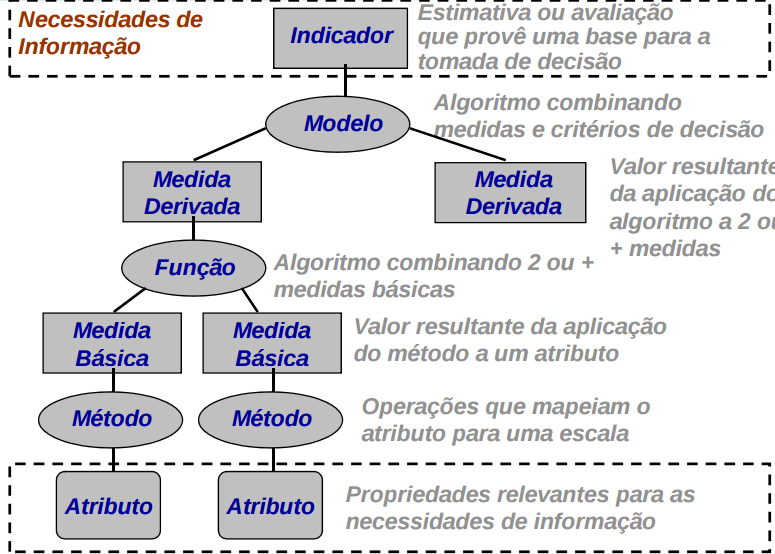


Figura X - Modelo para construção mensurável. (Aguiar, M., sd)

Um exemplo de aplicação é para o análise sob a produtividade que uma equipe pode ter em um projeto qualquer, a representação da construção mensurável para a produtividade neste exemplo é:

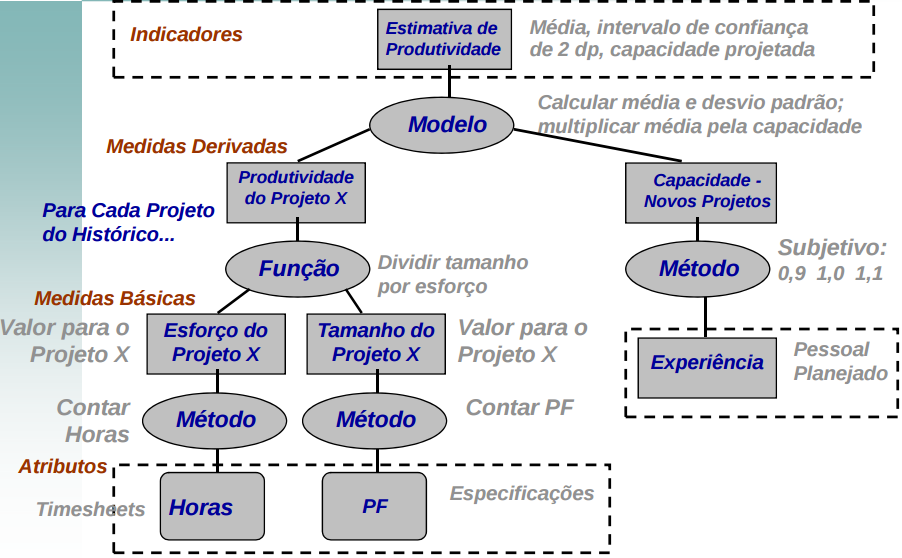


Figura X - Exemplo de construção mensurável de Proatividade. (Aguiar, M., sd)

### 4.2.2. Modelo de Processo

O modelo de processo foi adotado para a resolução do problema existente de como como conduzir o processo de medição(Aguiar, 2002).

Segundo Gava et al (2006) o processo de medição descreve as atividades de medição que geralmente podem ser aplicados em todas as circunstâncias referente às necessidades específicas. O processo consiste em quatro atividades iterativas: estabelecer, planejar, executar e avaliar, sendo similar ao ciclo PDCA (Plan-Do-Check-Act) proposto por Deming (1986). O modelo de processo proposto pelo *PSM* pode ser visto na figura abaixo.

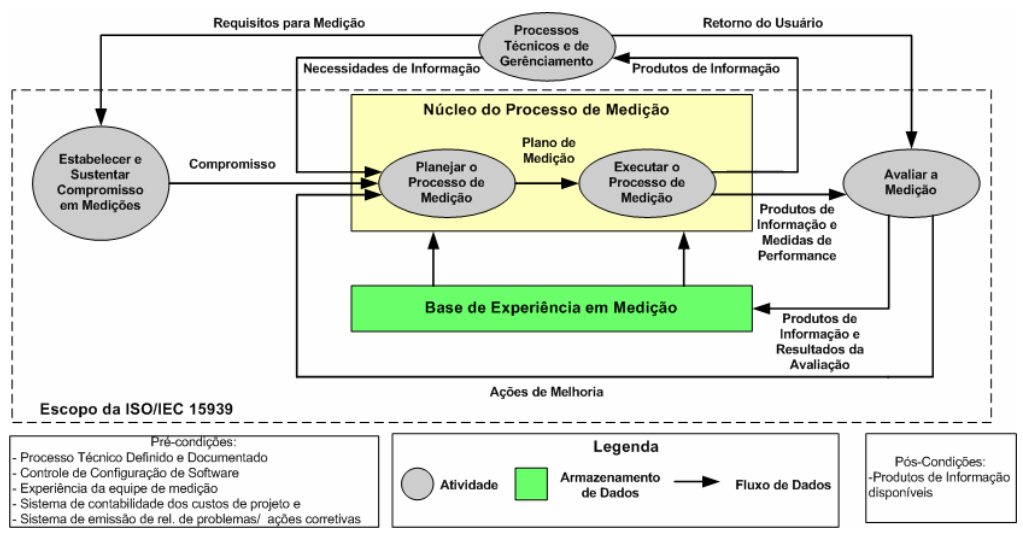


Figura X - Processo de medição de software. Adaptado da ISO/IEC 15939 (Gava et al, 2006)

Segundo Gava et al (2006) a atividade referente ao Planejamento do Processo de Medição permite que sejam integrados as medidas com os processos técnico e de gerenciamento. Durante o momento em que o plano de medição é implementado, os dados da medição devem ser coletados e analisados e o produto da informação é repassado para os responsáveis pela tomada de decisão.

A atividade Avaliar a Medição utiliza técnicas de análise e de medição, avaliando o processo e as medidas obtidas, auxiliando para que haja melhorias, de tal forma que o processo de medição deva ser atualizado constantemente com a finalidade de atender as necessidades de informações e promover um aumento de maturidade do processo em si (Gava et al, 2006).

A atividade Estabelecer e Sustentar Compromisso em Medições tem como finalidade garantir a suportabilidade das medições no nível de projeto e no nível de organização, provendo recursos para implementação de um programa de medição contínuo (Gava et al, 2006).

A atividade Processos Técnicos e de Gerenciamento não faz parte do processo de medição, mas mantém uma interface. Nesta atividade que os tomadores de decisões trabalham, tendo como insumo os processos definidos, as necessidades de informações e os produtos de informação. Com estes insumos os tomadores de decisão efetuam suas tomadas de decisão e enviam *feedback* sobre o processo para a atividade de Avaliar a Medição (Gava et al, 2006).

Segundo Gava et al (2006) “[...]O modelo de processo de medição é iterativo em sua concepção, de modo a capturar a experiência e as lições aprendidas nas versões anteriores da aplicação deste processo[...]”.

# 5. Alguns exemplos de métricas de código

## 5.1. Complexidade Ciclomática

Métrica que mede a quantidade de caminhos/voltas que um software fará a partir de determinado ponto.

Essa métrica serve para mensurar a complexidade de um determinado módulo do código (classe, método, etc), a partir da contagem do número de caminhos independentes que ele pode executar até o fim. Isso retorna um indicador numérico, que - baseado em um plano de qualidade previamente definido - identifica o quão complexo esse módulo está e a necessidade de uma refatoração.

## 5.2. Duplicação de código

Mede a quantidade de código duplicado ou similar que um código fonte apresenta. Baseado em similaridades identificadas, essa métrica indica possíveis pontos onde talvez a criação de um método ou classe atuasse de uma maneira mais eficiente, impedindo duplicação de lógicas semelhantes e aumentando a produtividade dos desenvolvedores.

## 5.3. Churn

Indicador que mostra quantidade de vezes que um determinado pedaço do código fonte sofreu alterações. A análise do churn é bastante peculiar, visto que ela não indica exatamente um problema, mas um possível foco de refatoração. A análise vai indicar em quais pontos do código estão tendo mais pessoas atuando, maiores modificações, etc, e isso pode levar a revelar um problema oculto exatamente nesse trecho.

## 5.4. Smells

“Smells” funcionam como métricas visuais que indicam possíveis pontos de refatoração do código, indicando diversos problemas diferentes. As smells funcionam como marcadores no código que mostram que determinados problemas existem ali e que precisam ser solucionados. As smells existentes variam bastante, podendo ser desde checagens de folha de estilo até trechos de código muito complexos.

## 5.5. Coesão e Acoplamento

Funciona mais para linguagens orientadas a objetos, medindo coesão e acoplamento entre classes e métodos da aplicação. A coesão está relacionada à ideia de “responsabilidade única”, onde uma classe ou método deve executar somente uma operação, o que facilita sua reutilização e manutenção. Já o acoplamento se dá ao relacionamento de um método ou classe com outros, que deve ser pouco, mantendo a independência destes em relação a outros.

## 5.6. Quantidade de linhas por método

Mede exatamente um valor mínimo aceitável para quantidade de linhas por método, que varia de acordo com a linguagem. Essa métrica de tamanho em geral, funciona como um indicador coesão. Um método com uma quantidade elevada de linhas de código pode representar que a função não está atomizada, e que existe um excesso de responsabilidades delegadas a ela, o que pode levar a não reaproveitamento do código e dificuldades de manutenção, futuramente.

# 6. Ferramentas

Para efetuar a medição da qualidade estática de software existem ferramentas que auxiliam neste controle. Tendo ferramentas focadas para diversas situações*,* sendo elas focadas na folha de estilo ou em análise estática de código.

As ferramentas de folha de estilo tem como finalidade verificar se o código desenvolvido está seguindo os padrões de desenvolvimento pré estabelecidos, facilitando assim a manutenção do sistema por não haver estilos de programação diferentes. Existem várias ferramentas que fazem essa verificação, contudo cabe ao responsável pela qualidade escolher a ferramenta que melhor se adeque ao contexto. Como exemplo de ferramentas temos para o JAVA o *checkstyle* que é uma ferramenta simples que pode ser executada via linha de comando e que traz no próprio terminal seu resultado. O *checkstyle* também é implementado como plugin do eclipse, o que traz em realtime os avisos de quebra de estilo. Para o Ruby o *rubocop* é uma ferramenta bem completa em relação ao análise de estilo do código, esta assim como o *checkstyle* tem sua execução direto no terminal e pode também ser adicionado a editores de texto como o sublime3 e o atom.

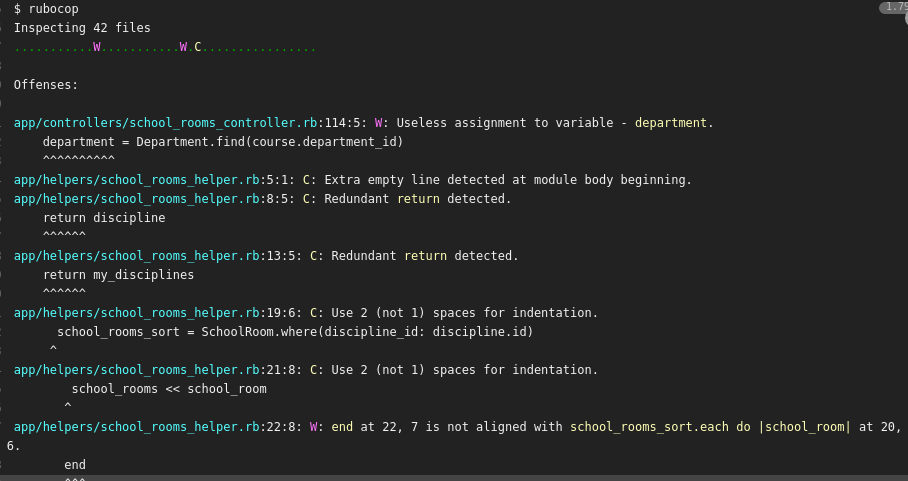


Figura X -Execução do rubocop

A análise estática de código tem como finalidade garantir a qualidade de código, verificando complexidade de código, acoplamento, duplicidade, etc. Essas métricas refletem em desempenho e manutenibilidade de software. Existem também várias ferramentas que auxiliam nessa análise, tendo algumas focadas em apenas uma linguagem de programação, como o *rubycritic,* para Ruby, ou que dão suporte a várias linguagens de programação, como o *CodeClimate,* o *Analizo*. Estas chegam a dar uma avaliação para cada arquivo dizendo a qualidade deste. O *rubycritic* é uma ferramenta instalada que roda via terminal e tem como output uma estrutura de pastas com arquivos html informando a situação do projeto. O *CodeClimate* é uma ferramenta online que sincroniza com repositórios e faz o análise após o commit. O *Analizo* roda em terminal e tem sua saída toda em um arquivo de texto informando todos os arquivos do projeto e suas respectivas métricas.

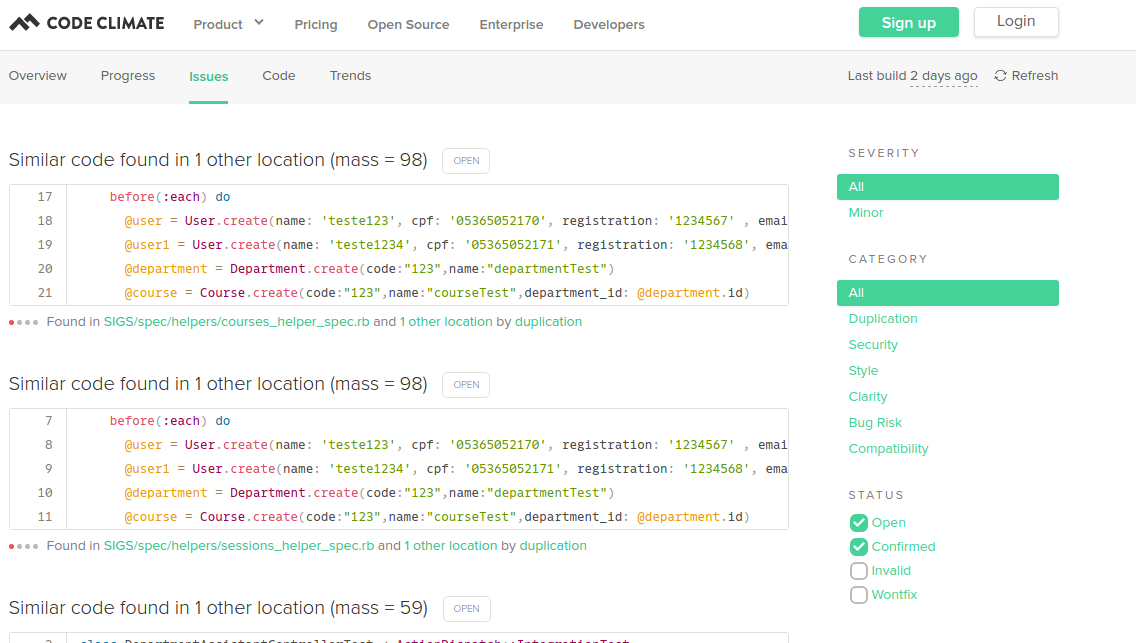


Figura X - Tela do codeclimate

# 7. Referências

Aguiar, M. (2002), ***PSM - O CMM da Mensuração de Software?***. Disponível em <http://www.metricas.com.br/downloads/PSM\_CMM\_Mensuracao\_Software.pdf> acessado em 01 de maio de 2016

Aguiar, M. (s.d.), ***Gerenciamento Objetivo de Projetos com PSM***. Disponível em <http://www.bfpug.com.br/Exame%20CSMS/PSM/Apresentacao\_PSM\_BFPUG\_2009-04-27.pdf> acessado em 01 de maio de 2016 apud McGarry, Card, et al., **“Practical Software Measurement - Objective Information for Decision Makers”** - Addison Wesley, 2001

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016 apud DEMING, W.E. (1986) - **Out of the Crisis. Cambridge**, MA: MIT Center for Advanced Engineering.

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016 apud VAZQUEZ, C. E.. & SIMÕES, G. S. & ALBERT, R. M. (2003) - Análise de Pontos de Função: Medidas Estimativas e Gerenciamento de Projetos de Software. Editora Érica. 1ª Edição. São Paulo.

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016 apud BASILI, V. R. (1992) - Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Universidade de Maryland: Relatório Técnico CS-TR-2956, Setembro.

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016 apud MCGARRY, J. et al. (2002) - Practical Software Measurement: Objective Information for Decision Makers. Addison-Wesley Professional

Gava, V. L., Batista, S., Gonçalves, R. F., Spinola, M. M.(2006), **Modelo do processo de medição de software: pré-condições para sua aplicação**. Disponível em <http://fattocs.com/files/pt/livro-apf/citacao/VagnerGava-SamuelBatista-RodrigoG-MauroSpinola-2006.pdf> acessado em 01 de maio de 2016 apud PSM - PRACTICAL SOFTWARE AND SYSTEMS MANAGEMENT. Apresenta modelos para medição de processos de desenvolvimento de software. Disponível em . Acesso em 16 de nov. de 2005.