5
Modelo de Medição das Plataformas

Como dito anteriormente, um dos objetivos desse projeto é avaliar sistematicamente soluções orientadas a agentes de software (OAS) e soluções orientadas a objetos (OO) considerando os esforços de desenvolvimento, operacional e de manutenção. A técnica escolhida para fazer a avaliação é a medição. Essa avaliação deve produzir uma comparação das técnicas de implementação OAS com as OO, contendo as vantagens, as desvantagens e as irrelevâncias entre as duas soluções.

Como a realização da medição por si só não produz nenhum resultado, a abordagem GQM foi selecionada para guiar na medição e análise do projeto. Dessa forma, é possível identificar os pontos chave para a comparação entre as duas técnicas de implementação e gerar uma interpretação a partir dos dados obtidos.

Inicialmente, o domínio do trabalho foi estudado e, a partir desse estudo, três objetivos foram estabelecidos para a realização da comparação. O primeiro é o desenvolvimento dos produtos. Na comparação entre as duas técnicas de implementação é importante confrontar a criação do produto a partir da técnica selecionada.

Tabela 5 - Objetivo desenvolvimento

Goal		
Objeto *O que será analisado?*	Análise das	Técnicas de desenvolvimento utilizando OO puro e Jade
Propósito *Por que o objeto será analisado?*	Com o propósito de	Avaliar
Foco de qualidade *Qual propriedade do objeto será analisada?*	De acordo com a	Facilidade de compreensão e facilidade de escrita
Ponto de vista *Quem irá usar os dados coletados?*	Do ponto de vista do	Pesquisador / desenvolvedor
Ambiente *Em qual meio a analise ocorre?*	No contexto de	Sistema GeoRisc
Questions		

Qual é a facilidade de compreensão do desenvolvedor na implementação do produto?	
Qual é a facilidade de escrita do desenvolvedor na implementação do produto?	
Metrics	
Número de linhas de código (LOC)	A quantidade de linhas do código pode afetar na facilidade de compreensão e na facilidade de escrita.
Número de operações (NOO)	A quantidade de operações pode afetar na facilidade de compreensão e na facilidade de escrita.
Número de classes (NOC)	A quantidade de classes pode afetar na facilidade de compreensão e na facilidade de escrita.
Peso dos métodos por classe (WMC)	A complexidade dos métodos pode ser um indicador da complexidade de escrita e compreensão do mesmo.

A facilidade de escrita é calculada com o esforço dividido pela densidade de defeitos. No entanto, por falta de tempo e recursos esses parâmetros não foram calculados. Para contornar esse problema foi selecionado um conjunto de métricas que podem influenciar na facilidade de escrita e na facilidade de compreensão. Algumas métricas desse conjunto têm limitações, que no decorrer do trabalho são comentadas, mas a avaliação em conjunto do resultado das métricas reduz a possibilidade de conclusões precipitadas.

O segundo objetivo é a operação dos produtos. Não se pode deixar de comentar a operacionalização dos produtos na comparação de técnicas. Caso uma técnica produza um produto que tenha uma usabilidade ou desempenho maior que o produto de outra técnica, esse fato deve ser levado em consideração.

Tabela 6 - Objetivo operação

Goal		
Objeto *O que será analisado?*	Análise das	Técnicas de desenvolvimento utilizando OO puro e Jade
Propósito *Por que o objeto será analisado?*	Com o propósito de	Avaliar
Foco de qualidade *Qual propriedade do objeto será analisada?*	De acordo com o	Desempenho e facilidade de utilização
Ponto de vista *Quem irá usar os dados coletados?*	Do ponto de vista do	Pesquisador / desenvolvedor
Ambiente *Em qual meio a analise ocorre?*	No contexto de	Sistema GeoRisc
Questions		

Quanto tempo demora-se para executar uma tarefa no sistema?	
Metrics	
Tempo da Tarefa (TT)	As tarefas podem ter tempos diferentes de acordo com a arquitetura utilizada.

No desenvolvimento do objetivo de operação foram constatados alguns desafios. Primeiramente não foram encontradas métricas ou ferramentas de medição com relação ao desempenho de um sistema qualquer. Dessa forma, baseado na pesquisa, foi selecionada a métrica de tempo de tarefa (TT) que visa medir o tempo gasto para completar uma tarefa no sistema. Essa métrica pode identificar se uma técnica pode necessitar de um processamento adicional afetando no tempo para a conclusão de uma tarefa, essa métrica é descrita com mais detalhes no item 6.3.1.

Outro problema encontrado foi a medição da facilidade de utilização. Ao contrário do desempenho essa característica tem vários meios de ser medida e existem muitos trabalhos na literatura. No entanto, os sistemas foram desenvolvidos para serem idênticos e a finalidade desse objetivo é comparar a facilidade de utilização entre as técnicas de desenvolvimento e não do sistema propriamente dito. Além disso, comentários qualitativos sobre a utilização dos sistemas serão feitos a fim de complementar a análise desse objetivo.

O terceiro e não menos importante é a manutenção do produto. Visto que, 80% dos recursos disponíveis são utilizados nesta etapa do ciclo de vida de um produto (PIGOSKY, 1996), é indispensável medir os benefícios da manutenção de um produto gerado por uma técnica para outro produto.

Tabela 7 - Objetivo manutenção

Goal		
Objeto *O que será analisado?*	Análise das	Técnicas de desenvolvimento utilizando OO puro e Jade
Propósito *Por que o objeto será analisado?*	Com o propósito de	Avaliar
Foco de qualidade *Qual propriedade do objeto será analisada?*	De acordo com a	Facilidade de manutenção
Ponto de vista *Quem irá usar os dados coletados?*	Do ponto de vista do	Pesquisador / desenvolvedor
Ambiente *Em qual meio a analise ocorre?*	No contexto de	Sistema GeoRisc
Questions		
Qual é o esforço do desenvolvedor na compreensão do produto?		

Qual é o esforço do desenvolvedor na alteração do produto?	
Metrics	
Acoplamento entre objetos (CBO)	Um sistema altamente acoplado pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.
Falta de coesão nos métodos (LCOM)	Um sistema com baixa coesão pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.
Número de atributos (NOA)	Um sistema com muitos atributos pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.
Peso dos métodos por classe (WMC)	Um sistema métodos complexos pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.
Número de operações (NOO)	Um sistema com muitas operações pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.
Número de classes (NOC)	Um sistema com muitas classes pode dificultar a compreensão e a alteração, afetando negativamente a manutenção.

Assim como comentado no objetivo desenvolvimento, o objetivo manutenção não pode ser coletado com métricas próprias para medir a facilidade de alteração do produto. No entanto, um conjunto de métricas que influenciam na compreensão do produto e na dificuldade de alteração do produto foram selecionadas, de modo que, o *gap* entre a medição ideal e a realizada no trabalho seja minimizado.

5.1.Métricas utilizadas

Neste item são discutidos os benefícios de cada métrica selecionada, o funcionamento, o cálculo e uma análise sobre possíveis resultados.

5.1.1.Número de Linhas de Código (LOC)

Essa métrica pode ser encontrada em (FENTON, 1996) e é uma das métricas mais simples. Ela é responsável por contar o número de linhas de código que tem no sistema. Linhas em branco e linhas de comentários não são considerados como código. Esse tipo de métrica é influenciada pelo estilo de programação do desenvolvedor, por isso, para o experimento foi desenvolvido

um formatador de código para que os produtos tenham o mesmo estilo de programação.

O tamanho de um sistema em LOC varia dependendo da linguagem de programação, ferramentas e técnicas utilizadas. Uma funcionalidade em um determinado tipo de linguagem pode ser feita em uma quantidade de linhas menor do que em outra linguagem. Esse fato pode ser um indicador de esforço, visto que, o desenvolvedor teve que escrever uma quantidade de linhas maior em uma linguagem.

No entanto, essa métrica deve ser utilizada com cuidado, pois o desenvolvedor poderia necessitar de um esforço maior para implementar uma funcionalidade com poucas linhas em uma linguagem, do que, implementar a mesma funcionalidade em outra linguagem mesmo que usando mais linhas para tal. Assim, é importante utilizar essa métrica em combinação com outras métricas, a fim de que, a LOC não conduza a conclusões erradas.

5.1.2.Número de operações (NOO)

No trabalho (CHIDAMBER, 1994) é apresentada a métrica NOO. Essa métrica conta o número de operações, métodos, presentes em um objeto. Quanto maior o número de operações de um objeto, mais difícil é a compreensão do mesmo e conseqüentemente do sistema. Se uma classe tem um alto número de operações, isso pode ser um indício de que a classe não é coesa e poderia ser dividida em outras classes mais coesas.

No entanto, em alguns casos, sistemas com poucas operações e com muitos parâmetros e variáveis membro do objeto podem ser mais complexos do que sistemas com mais operações e menos parâmetros e objetos contendo menos atributos.

5.1.3.Número de classes (NOC)

No trabalho (CHIDAMBER, 1994) é apresentada a métrica NOC. Essa métrica conta o número de classes do sistema. Acredita-se que quanto maior o número de classes que o sistema possua, mais difícil será a compreensão e a manutenção do mesmo.

Entretanto, como nas outras métricas a NOC pode levar a conclusões errôneas. Um número maior de classes pode indicar que o sistema está mais

coeso. Ou seja, um número grande de classes altamente coesas é mais desejável do que um pequeno número de classes pouco coesas.

5.1.4.Tempo da tarefa (TT)

Essa métrica é extraída a partir de testes executados com uma ferramenta automatizada. A ferramenta simula a execução do programa de acordo com os passos desejados. Dessa forma, o cálculo será realizado a partir intervalo de tempo em que a ferramenta requer para realizar um caso de uso. Com essa métrica é possível verificar se uma tecnologia demora mais para terminar o mesmo caso de uso realizado em outra tecnologia.

Além da tecnologia, essa métrica pode ser afetada pela forma de implementação. Para minimizar esse problema foram utilizados padrões de desenvolvimento recomendados pelas duas tecnologias e o mesmo desenvolvedor implementou os dois sistemas de forma paralela. Assim, caso alguma tecnologia tenha perda de tempo motivada pela adoção de um padrão recomendado pela própria técnica, esse tempo será contabilizado como perda de tempo da tecnologia. E a forma de implementação dos sistemas não mudou em virtude do mesmo desenvolvedor ter implementado os dois sistemas.

5.1.5.Acoplamento entre objetos (CBO)

No trabalho (CHIDAMBER, 1994) é apresentada a métrica CBO. Essa métrica mede os relacionamentos existentes entre classes de objetos. Segundo os autores de (CHIDAMBER, 1994), o acoplamento acontece quando um objeto age em outro, por exemplo, quando métodos de um objeto usam métodos ou variáveis de outro. Essa métrica conta o número de tipos que são usados em declarações de atributos, parâmetros formais, tipos de retorno, declarações *throw* e variáveis locais.

O acoplamento entre os objetos pode afetar o sistema de diferentes formas. Segundo Chidamber e Kemerer, quanto maior o número de ligações, maior é a suscetibilidade a mudanças em outras partes do design, e conseqüentemente a manutenção é mais difícil. Além disso, a compreensão de um objeto depende do entendimento dos objetos nos quais ele está acoplado. Dessa forma, quanto maior o acoplamento do objeto, mais difícil é entendê-lo.

Dessa forma, o ideal é manter o CBO o mais baixo possível, pois quanto menor for o acoplamento, mais fácil será o reuso e a criação facilitada dos testes

dos componentes do sistema. Além disso, o CBO baixo facilita a compreensão dos objetos, conseqüentemente facilitando o entendimento do sistema. Dessa forma, a manutenção do sistema é facilitada pelos benefícios do baixo acoplamento.

5.1.6.Falta de coesão nos métodos (LCOM)

Essa métrica mede a falta de coesão de uma classe. A LCOM seleciona todos os pares de métodos de uma classe e verifica se esses compartilham algum atributo. Após a verificação é subtraído o número de pares que tem algum tipo de compartilhamento dos que não tem.

De acordo com (CHIDAMBER, 1994), baixa coesão aumenta a complexidade, levando a um provável aumento do número de defeitos injetados no software. Segundo os autores quanto maior a coesão, mais fácil é o entendimento do objeto, no entanto essa afirmação ainda não foi provada empiricamente pelos autores.

Porém essa métrica é bastante criticada na literatura. Em particular uma das críticas se refere aos erros de medição no caso de *getters* e *setters* que é comentado nesta dissertação.

5.1.7.Número de atributos (NOA)

Essa métrica conta o número de atributos de cada objeto. Os atributos herdados não são incluídos no cálculo. Quanto maior o número de atributos no objeto, mais difícil é entender o sistema. Dessa forma, quanto maior o número de atributos, mais difícil se torna a manutenção do sistema. Se uma classe tem um valor elevado de NOA, isso pode ser um indicador de que a classe necessita ser refatorada.

5.1.8.Peso dos métodos por classe (WMC)

Essa métrica mede a complexidade da classe através da soma simples das complexidades dos métodos da classe. Para fazer esse cálculo, é necessário ter a complexidade de todos os métodos da classe a ser medida. Os autores de (CHIDAMBER, 1994) não definiram uma forma obrigatória de calcular as complexidades dos métodos, com a finalidade de deixar a aplicação dessa métrica o mais geral possível. Neste trabalho será adotada a métrica *Cyclomatic Complexity* (MCCABE, 1976) para definir a complexidade de cada método.

O número de métodos e a complexidade desses é um indicador do esforço de desenvolvimento e de manter a classe. Além disso, quanto maior foi o número de métodos do objeto e mais elevada a complexidade desses métodos, mais difícil se torna o entendimento do objeto e conseqüentemente o entendimento do sistema.

5.1.9.Complexidade ciclomática (CC)

Essa métrica é utilizada no cálculo da métrica WMC. A CC foi proposta por (MCCABE, 1976) e representa a quantidade de caminhos de execução independentes a partir de um código fonte. Nesse processo, são contados o número de possíveis caminhos através de um algoritmo que conta o número de regiões distintas em grafo. Dessa forma, essa métrica se torna ideal para definir a complexidade de um método, necessária para a métrica WMC.

No entanto, como em outras métricas existem criticas na literatura a essa métrica. Alguns trabalhos falam que essa métrica se baseia em uma fraca fundação teórica não baseada em evidencia empírica. Esses trabalhos comentam que essa métrica é superada em alguns casos pela LOC.

5.2.Ponto de vista do desenvolvedor

Como dito anteriormente, esse trabalho não tem por objetivo listar exaustivamente diferentes tipos de métricas para fazer medições. Uma das metas desse trabalho é selecionar um subconjunto que possa auxiliar na comparação das técnicas de desenvolvimento de software.

Foram selecionados três conjuntos cada um contendo métricas que poderiam contribuir de alguma forma em atingir um objetivo.

No objetivo de desenvolvimento foi escolhida a métrica LOC pois essa métrica pode afetar o esforço gasto para escrever o software. A métrica NOC foi escolhida com o objetivo de medir a dificuldade que o desenvolvedor enfrenta em conhecer o sistema/domínio. As métricas WML e NOO foram selecionadas com o objetivo de medir a complexidade de escrever o software.

No objetivo operação a métrica TT foi escolhida pois efetivamente executa o software, fornecendo dados importantes sobre a utilização dos sistemas.

No objetivo manutenção o programador deve ter um bom entendimento do sistema em si e dos relacionamentos entre os componentes do sistema. Dependendo dos relacionamentos entre os componentes a manutenção pode se tornar custosa. Assim, foram selecionadas as métricas CBO, LCOM e NOA para auxiliar na medição do objetivo de manutenção. Foram utilizadas as métricas WMC, NOO e NOC do objetivo desenvolvimento pois também são importantes para manutenção no sentido de entender a complexidade do sistema.