JHI JOURNAL OF HEALTH INFORMATICS

Modelo openEHR: avaliação da qualidade de Projeto Orientado a Objeto

OpenEHR Model: quality evaluation for Object-Oriented Design

Modelo openEHR: evaluación de la Calidad del Diseño Orientado a Objetos

Reinaldo Araújo Alkimim[1], Fernando Silva Parreiras[2], Marcelo Rodrigues dos Santos[3], Zilma Silveira Nogueira Reis[3], Cristiana Fernandes De Muylder[4]

Descritores: Registros Eletrônicos de Saúde; Desenho de Programas de Computador; Desenvolvimento Experimental

RESUMO

Objetivo: Avaliar a qualidade de Projeto Orientado a Objeto (POO) do Modelo de Objetos da openEHR, utilizando métricas de orientação a objeto (OO). **Método:** Um estudo experimental foi conduzido com artefatos da implementação em Java do openEHR e com métricas OO do modelo de qualidade QMOOD. **Resultados:** Identificou-se que os atributos de qualidade Reusabilidade e Funcionalidade, satisfizeram as expectativas do modelo de qualidade. Já os atributos Extensabilidade e Flexibilidade, mostraram-se instáveis, enquanto Facilidade de Compreensão ficou em queda. Complementarmente, foram identificados seis problemas de POO em estratégias de detecção de problemas de POO. **Conclusão:** O Modelo de Objetos da openEHR tem ganho de novos recursos e tem a capacidade de reutilizar módulos já existentes para resolver um novo problema. Entretanto, novos requisitos em recursos já existentes podem ser mais trabalhosos, assim como a adaptação do projeto para novos recursos. Também apresenta dificuldade de ser aprendido e compreendido devido ao aumento da complexidade.

Keywords: Electronic Health Records; Software Design; Experimental Development

ABSTRACT

Objective: To evaluate the quality of Object Oriented Design (OOD) of the openEHR Object Model, using Object Oriented (OO) metrics. **Methods:** An experimental study was designed and conducted using the openEHR Java reference implementation artifacts and with OO metrics of QMOOD quality model. **Results:** The results identified that the quality attributes of Reusability and Functionality satisfied the expectations of the quality model. The quality attributes of Flexibility and Extensibility proved unstable, while the quality attribute Understandability decreased. In addition, were identified six problems of OOD in detection strategies of problems of OOD. **Conclusion:** The openEHR Object Model gained new features and can reuse existing modules to solve a new problem. However, new requirements on existing resources can be difficult, as well as the adaptation of the project to new features. It also present difficulties to be learned and understood because of increase in the complexity.

Descriptores: Registros Electrónicos de Salud; Diseño de Programas Informáticos; Desarrollo Experimental

RESUMEN

Objetivo: Evaluar la calidad de Proyecto Orientado a Objeto (POO) del Modelo de Objetos del openEHR, utilizando métricas de orientación del objetos (OO). **Método:** Un estúdio experimental fue se llevó a cabo con artefactos de la implementación en Java del openEHR y con métricas OO del modelo de calidad QMOOD. **Resultados:** Ha identificado que los atributos de calidad Reusabilidad e Funcionalidad, han satisfecho las expectativas del modelo de calidad. Sin embargo los atributos Extensabilidad y Flexibilidad, han resultado ser inestables, mientras Facilidad de Comprensión ha establecido en baja. Además, se ha identificado seis problemas de POO en estrategias de detección de problemas de POO. **Conclusión:** El Modelo de Objetos del openEHR ha adquirido nuevas características y tiene la capacidad de reutilizar los módulos existentes para resolver un nuevo problema. Sin embargo, los nuevos requisitos sobre los recursos existentes puede ser engorroso, así como la adaptación del diseño para nuevos recursos. También presentan dificultades para aprender y comprender debido a la mayor complejidad.

[1] Mestrando em Sistemas de Informação e Gestão do Conhecimento pela Fundação Mineira de Educação e Cultura - FUMEC, Belo Horizonte (MG), Brasil.
[2] Doutor em Ciência da Computação Summa Cum Laude pela Universität Koblenz-Landau, Alemanha.
[3] Doutor em Ciência da Informação pela Universidade Federal de Minas Gerais - UFMG, Belo Horizonte (MG), Brasil.
[4] Doutora em Administração pela Fundação Mineira de Educação e Cultura - FUMEC, Belo Horizonte (MG), Brasil.

Autor Correspondente: **Reinaldo Araújo Alkimim**
e-mail: **reinaldoalkimim@hotmail.com**

Artigo recebido: **30/03/2015**
Aprovado: **31/08/2016**

INTRODUÇÃO

Dos principais padrões que tratam a interoperabilidade em sistemas Registro Eletrônico de Saúde (RES), destacam-se a Fundação openEHR, *Comite Européen de Normalisation* (CEN) 13606, *Heath Level Seven* (HL7) e o *Integrating the Healthcare Enterprise* (IHE), *Digital Imaging and Communications in Medicine* (DICOM) e *Medical Markup Language* (MML)[1-3].

A abordagem da Fundação openEHR para RES é considerada um padrão muito completo e tem ganhado atenção na comunidade de informática em saúde[3-4]. A própria Fundação openEHR credita o sucesso devido à aceitação formal do CEN 13606 como um padrão da Europa e da *International Standardization Organization* (ISO)[5]. Outra vantagem é a existência de um ferramental pronto para uso como *Archetype Editor*, *Archetype Workbench*, *Template Designer*, *Clinical Knowledge*[6]. Portanto, justifica-se a escolha do padrão openEHR como objeto de estudo deste trabalho.

A Fundação openEHR é uma empresa sem fins lucrativos fundada pela *University College London* do Reino Unido e a *Ocean Informatic* Pty Ltd da Austrália. Ela tem como objetivo principal facilitar a criação e o compartilhamento de registro de saúde de pacientes para a comunidade médica[5,7].

Entretanto, implementar a especificação do padrão openEHR pode ser uma tarefa trabalhosa. Segundo Velte[3], trata-se de um padrão complexo e difícil de entender para alguém novo na área. Em função disso, as instituições de saúde podem enfrentar muitos desafios durante a implementação de sistemas de RES[4].

A utilização de métricas da Engenharia de *Software* podem minimizar estes desafios. Na área de Engenharia de *Software*, em função de interesse de pesquisadores, existe um estudo de métricas na busca de soluções para os diversos desafios e problemas relacionados ao desenvolvimento de *software*. Um dos benefícios das métricas de *software* é avaliar a qualidade dos produtos e processos permitindo uma melhor compreensão do *software* por parte dos desenvolvedores[8]. Especificamente, as métricas de *software* de Orientação a Objetos (OO) são as mais direcionadas ao padrão openEHR, visto que ele possui um Modelo de Objetos que utiliza os conceitos de OO. A abordagem OO pode melhorar o desenvolvimento de *software*, incluindo fatores como maior reusabilidade e extensibilidade, o que caracteriza a qualidade de um Projeto Orientado a Objeto (POO). Com o intuito de ajudar os gerentes e desenvolvedores a atingir esses objetivos, uma variedade de métricas de OO tem sido propostas para ajudar a controlar POO[9].

Até a realização desse trabalho não se observaram estudos dessa natureza voltados para o Modelo de Objetos da fundação openEHR. Neste trabalho, é evidenciado a importância do sistema de RES, a aceitação do Modelo de Objetos da openEHR e os benefícios relacionados com a aplicação de métricas de *software*. Nesse cenário, uma avaliação do Modelo de Objetos da openEHR, em relação à qualidade de POO, torna-se relevante para qualquer organização que pretenda utilizá-lo na implementação de um sistema de RES com base na abordagem da Fundação openEHR.

O presente trabalho tem como objetivo geral avaliar a qualidade de POO do Modelo de Objetos da Fundação openEHR com base em métricas de Orientação a Objetos.

MÉTODOS

Esta pesquisa caracteriza-se como qualitativa de estudo experimental em Engenharia de *Software*. Para este trabalho foi adotado o estudo experimental em Engenharia de *Software* proposto por Wohlin *et. al.*[10] Esta pesquisa também tem o caráter empírico, visto que os artefatos e os métodos são aplicados em sistemas reais[11]. Estudos empíricos veem sendo utilizados nas últimas décadas para avaliar os pontos fortes e fracos dos artefatos de engenharia de *software*[12].

A avaliação da qualidade foi feita com base no *Quality Model for Object-Oriented Design* (QMOOD) proposto por Bansiya e Davis[13]. O modelo QMOOD tem como objetivo avaliar atributos de qualidade de POO, que podem ser quantificados por intermédio de métricas OO[14-15]. Dos seis atributos de qualidade disponíveis no modelo QMOOD foram selecionados Extensibilidade, Facilidade de Compreensão, Flexibilidade, Funcionalidade e Reusabilidade. Bansiya e Davis[13] ressaltam que esse conjunto de atributos de qualidade não é exclusivo, ou seja, eles podem ser alterados de acordo com os objetivos desejados. Este foi o caso de não ter sido utilizado o atributo de qualidade Eficácia, visto que o projeto da openEHR foi especificado com uma capacidade de funcionalidades e comportamentos que não afeta o objetivo deste trabalho.

As métricas de *software* são consideradas um mecanismo importante para avaliar a qualidade de POO. Entretanto, apresentam uma limitação em obter interpretações adequadas, pois o valor da métrica não deixa clara a causa da não conformidade e, como consequência, afeta a relevância dos resultados[16-17]. Como o modelo QMOOD possui essa limitação, este estudo utiliza mecanismos de estratégias de detecção. Esses mecanismos capturam desvios dos princípios e heurísticas da OO por meio de valores limites com base em estudos empíricos[15]. As estratégias de detecção de problemas de POO, propostas por Lanza e Marinescu[16], foram adotadas nesse estudo para auxiliar na interpretação dos resultados.

O contexto do experimento corresponde ao Modelo de Objetos da openEHR desde suas versões iniciais até as últimas versões, incluindo o código fonte do *framework* Java *Reference Implementation* (JRI) do openEHR. O projeto JRI é uma implementação de referência na linguagem de programação Java, reconhecida e publicada no sítio da Fundação openEHR[18]. A utilização de todas as versões do projeto JRI no experimento é necessário porque o modelo QMOOD avalia os atributos de qualidade de POO da forma como eles são evoluídos de uma versão para outra.

Na primeira fase de operação do experimento foram obtidos o código fonte do projeto JRI, especificações e diagramas UML do Modelo de Objetos. Todos estes artefatos estão disponíveis no sítio da Fundação openEHR.

Tabela 1 - Relacionamento de versões do Modelo de Objetos da openEHR X JRI.

Ordem	Modelo de Objetos openEHR		Java Reference Implementation	
	Versão	Data	Versão	Data
1	0.9	04/05/2004	Não teve equivalente	
2	0.95	15/03/2005	0.95	22/03/2006
3	1.0	07/02/2006	1.0	10/08/2006
4	1.0.1	15/04/2007	1.0.1	13/12/2007
5	1.0.2	31/12/2008	Não teve equivalente	
6	Current Baseline		1.0.5	

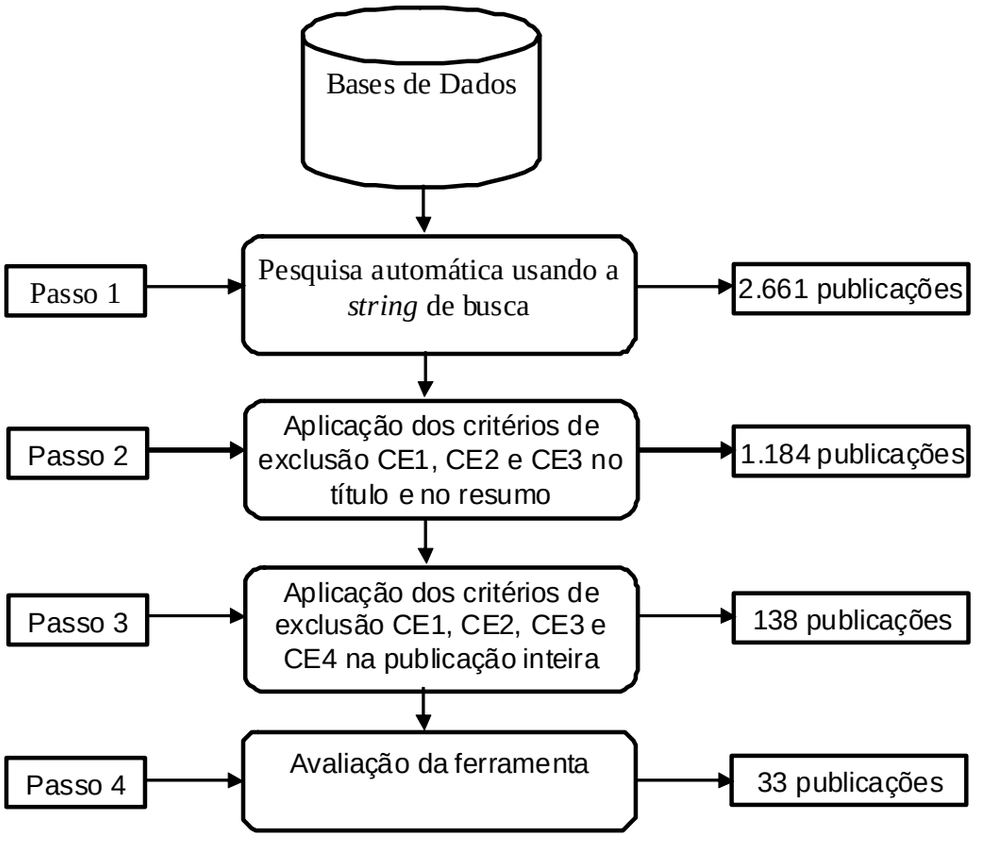

Figura 1 - Estratégia de pesquisa para a questão de pesquisa Q4

Um mapeamento das versões do Modelo de Objetos em relação às versões do JRI foi efetuado para identificar a correlação entre eles. O resultado é exibido na Tabela 1, onde é evidenciado que as versões 0.9 e 1.0.2 do Modelo de Objetos openEHR não tiveram implementações no JRI. Além disso, a versão atual no projeto JRI é identificada como 1.0.5, enquanto no Modelo de Objetos é nomeada como *Current Baseline*.

As ferramentas utilizadas no experimento para extração das métricas foram definidas por meio do método de Revisão Sistemática da Literatura com a seguinte questão de pesquisa: quais as ferramentas disponíveis para coleta de métricas de OO que podem ser utilizadas no Modelo de Objetos da openEHR?

A pesquisa foi realizada nas seguintes bases de dados internacionais: ACM *Digital Library*, *Academic OneFile*, Biblioteca Digital de Teses e Dissertações (BDTD), Biblioteca Virtual em Saúde (MEDLINE e LILACS), BioMed Central, Cambridge *Journals*, *Engineering Village*, *Crossref, Directory of Open AccesS Journals*, EBSCO, Emerald, IEEE Xplore, Inspec, ISI *Web of Science*, SAGE *Journals*, *ScienceDirect*, Scopus, *Springer*, U.S. *National Library of Medicine* e Wiley.

Foi definida a estratégia de pesquisa que utilizou uma *string* de busca a partir das seguintes palavras-chave em inglês: Ferramenta e Métrica e Orientação a Objeto. Assim, a *string* de busca ficou da seguinte forma: (*Metric AND Tool AND Object-Oriented*) *OR* (*Metric AND Tool AND OO*). Como seleção, foram considerados os critérios de inclusão: CI1) idiomas português, inglês e espanhol; CI2) artigos completos publicados em periódicos e conferências; e CI3) teses e dissertações defendidas. Como critérios de exclusão: CE1) ensaios; CE2) artigos de revisão; CE3) artigos em duplicidade; e CE4) ferramentas que não estão disponíveis para *download*. Como resultado, foram encontradas 33 publicações, entre as quais 22 ferramentas estavam disponíveis para avaliação (Figura 1).

Das ferramentas encontradas, foi feita uma primeira avaliação para identificar quais eram aplicadas em código fonte em Java, visto que os Modelos de Objetos do

openEHR teriam que ser medidos pela implementação JRI, Foram descartadas as ferramentas que avaliam apenas modelos em notação UML, já que métricas de OO que precisam de implementações em seus métodos ficam impossibilitadas de serem aplicadas. Também foram descartadas as ferramentas eram aplicadas em código-fonte diferente da linguagem Java que é utilizada na implementação JRI.

Por fim, foram escolhidas as ferramentas InFusion e *Chidamber and Kemerer Java Metrics* (CKJM) por possuírem um número maior de métricas implementadas[19-20].

Na segunda fase da operação do experimento, as métricas foram aplicadas nos artefatos do Modelo de Objetos da openEHR, por meio da ferramenta CKJM, para obter os atributos de qualidade definidos pelo QMOOD. No caso das estratégias de detecção de problemas de POO, primeiramente, foram parametrizados os valores limites na ferramenta InFusion. Depois, as estratégias de detecção foram executadas nos artefatos do Modelo de Objetos da openEHR para que os problemas de POO fossem identificados. Os resultados obtidos em ambas as ferramentas foram exportados para serem analisados e interpretados na atividade análise dos resultados do experimento.

RESULTADOS E DISCUSSÃO

Primeiramente, ressalta-se que os resultados aqui apresentados descrevem a qualidade de POO que é percebida por pessoas que estão relacionadas ao projeto e desenvolvimento de sistemas. Portanto, eles não são relevantes para usuários de sistemas de RES desenvolvidos com base no Modelo de Objetos da openEHR.

Existiram dois grupos de resultados: os resultados das estratégias de detecção de problemas de POO e os resultados dos atributos de qualidade com base no modelo QMOOD. De forma colaborativa, o primeiro grupo de resultado é relacionado com o segundo. Em alguns casos de problemas de POO encontrados, são apresentadas algumas técnicas de refatoração para a melhoria da qualidade.

A Tabela 2 exibe os resultados das estratégias de detecção. Em todas as versões do Modelo de Objetos analisadas não houve problemas de projeto nas estratégias de detecção Acoplamento Intensivo, Cirurgia de Espingarda, Herança do Pai Recusada e Quebrador de Tradição. De forma contrária, as estratégias de detecção Classe de Dados e Inveja de Funcionalidade tiverem problemas de projeto em todas as versões analisadas do Modelo de Objetos. Já as estratégias Classe Deus e Classe Cérebro tiveram problemas apenas nas duas últimas versões e na última versão, respectivamente.

A estratégia Classe Deus identificou algumas classes com muitas responsabilidades, na qual destaca-se a classe *Archetype*. Com esse tipo de classe existirá dificuldades de evolução[16]. Para correção desse problema de projeto (refatoração), Shatnawi e Li[21] propõem os seguintes passos: decidir como dividir a classe, criar a nova classe, fazer uma ligação entre as duas classes, mover os campos para a nova classe, e por fim, mover métodos para a nova classe.

Os métodos identificados na estratégia Inveja de

Tabela 2 - Resultado das estratégias de detecção de problemas de POO.

Estratégias de Detecção	Versão do Modelo de Objetos			
	0.95	**1.0**	**1.0.1**	**1.0.5**
Acoplamento Intensivo	0	0	0	0
Cirurgia de Espingarda	0	0	0	0
Classe Cérebro	0	0	0	1
Classe de Dados	20	30	26	20
Classe Deus	0	0	1	7
Herança do Pai Recusada	0	0	0	0
Inveja de Funcionalidade	2	2	5	7
Método Cérebro	2	2	3	10
Quebrador de Tradição	0	0	0	0

Tabela 3 - Resultado das propriedades de projeto baseado no QMOOD.

Propriedades de Projeto	Versão do Modelo de Objetos			
	0.95	**1.0**	**1.0.1**	**1.0.5**
Abstração	0,31	0,29	0,39	0,41
Acoplamento	5,40	5,86	5,69	5,89
Coesão	0,41	0,42	0,44	0,45
Complexidade	1.548	1.821	2.862	3.635
Composição	1,25	1,33	0,95	0,88
Encapsulamento	0,70	0,71	0,69	0,65
Herança	0,05	0,04	0,05	0,05
Hierarquias	52,00	57,00	91,00	109,00
Polimorfismo	0,25	0,23	0,33	0,26
Tamanho do Projeto	169,00	194,00	231,00	269,00
Troca de mensagens	1.126,00	1.284,00	1.681,00	2.148,00

Funcionalidade fazem parte apenas das classes *XMLSerializer* e *ADLSerializer*. De acordo com Lanza e Marinescu[16], tal problema de projeto é resolvido movendo o método para a classe onde ele está mais acoplado.

A estratégia de detecção Classe de Dados foi a que teve o maior número de ocorrências em todas as versões do Modelo de Objetos. As classes detectadas possuem a característica de ter muitos dados, que são manipulados por outras classes, ao invés de implementar seus próprios métodos. Exemplos encontrados são as classes *Archetyped*, EHR, *EHRExtract* e *Entry*.

A estratégia de detecção Classe Cérebro encontrou um problema de projeto na classe *Flattener* somente na última versão do Modelo de Objetos. Além de enquadrar-se nas características de complexidade e na existência de muitos Métodos Cérebro. É a maior classe do JRI com 1.732 linhas de código.

A estratégia Método Cérebro detectou problemas de projeto em todas as versões do Modelo de Objetos, sendo que houve um aumento significativo na última versão por causa da inclusão do controle de modelos de arquétipos. Esse modelo de arquétipos envolveu a classe *Flattener*, citada anteriormente, a qual possui muitos Métodos Cérebro.

Na segunda parte, os resultados das propriedades de projeto do modelo QMOOD foram obtidos utilizando as métricas relacionadas as propriedades de projeto, conforme exibido na Tabela 3.

Como os valores das métricas possuem intervalos diferentes para cada propriedade de POO, então é necessário fazer uma normalização de todas as versões, dividindo o resultado de cada versão pela primeira versão. Assim, os resultados da Tabela 3 foram normalizados em relação aos valores da versão 0.95 e apresentadas na Tabela 4. Para uma melhor visualização, os mesmos dados da Tabela 4 são apresentados na Figura 2.

As propriedades Complexidade, Hierarquias, Troca de Mensagens e Tamanho do Projeto tiveram valores maiores em relação à versão anterior. Como as novas versões incorporam novas funcionalidades, então, era esperado que a cada nova versão do Modelo de Objetos os valores dessas propriedades fossem maiores.

De uma maneira discreta, os valores da propriedade Coesão também aumentaram em relação à versão anterior. Isso é um bom indicador, visto que uma alta coesão é desejada em POO, pois um módulo altamente coeso tende a ser mais compreensível, modificável e de fácil manutenção[22-23]. O que corrobora com a composição dessa propriedade nas fórmulas dos atributos de qualidade Reusabilidade e Facilidade de Compreensão que serão discutidos posteriormente.

Tabela 4 - Resultado normalizado das propriedades de projeto baseado no QMOOD.

Propriedades de Projeto	Versão do Modelo de Objetos			
	0.95	1.0	1.0.1	1.0.5
Abstração	1,00	0,94	1,26	1,32
Acoplamento	1,00	1,09	1,05	1,09
Coesão	1,00	1,02	1,07	1,10
Complexidade	1,00	1,18	1,85	2,35
Composição	1,00	1,06	0,76	0,70
Encapsulamento	1,00	1,01	0,99	0,93
Herança	1,00	0,80	1,00	1,00
Hierarquias	1,00	1,10	1,75	2,10
Polimorfismo	1,00	0,92	1,32	1,04
Tamanho do Projeto	1,00	1,15	1,37	1,59
Troca de mensagens	1,00	1,14	1,49	1,91

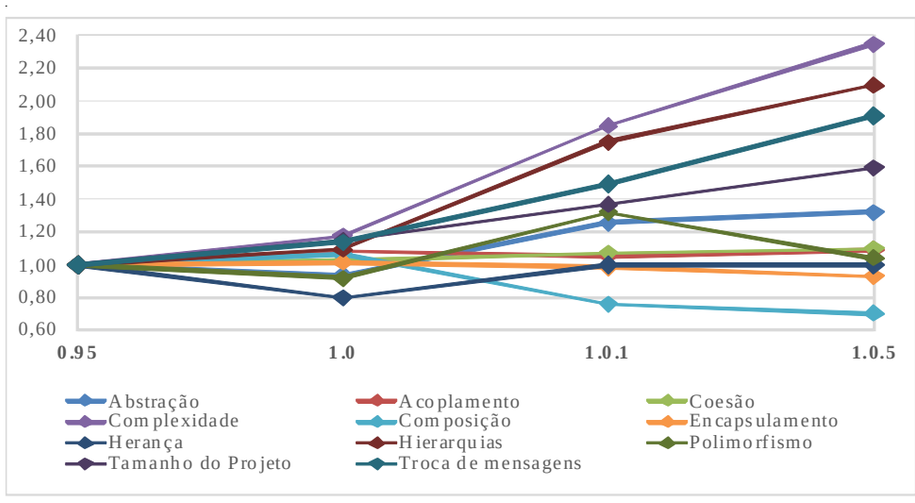

Figura 2 - Resultado das propriedades de projeto baseado no QMOOD

Em relação ao Acoplamento, o valor diminuiu da versão 1.0 para 1.01, mas manteve o valor original na versão 1.0.5. O ideal é que não houvesse aumento nessa propriedade, pois é desejado um baixo acoplamento em POO[23]. Isso significa que a dependência entre os módulos aumentou na versão atual. Assim, uma modificação em um módulo pode resultar em mudanças em outros módulos, o que pode gerar um efeito dominó e dificultar o entendimento do projeto.

O valor da Composição diminuiu nas últimas duas versões e o número de Hierarquias cresceu expressivamente, o que indica que, em alguns casos, a relação de composição entre algumas classes foi substituída pela Herança.

A propriedade de Herança teve o mesmo valor em todas as versões, com exceção da versão 1.0 na qual teve uma leve queda. Ou seja, manteve-se estável. Dessa maneira, essa propriedade irá afetar positivamente o atributo de qualidade Reusabilidade do Modelo de Objetos, pois tem impacto importante no POO[24].

A propriedade polimorfismo foi a única que teve diferentes variações nas versões. O valor diminuiu na versão 1.0, aumentou na versão 1.01 e voltou a diminuir na versão 1.0.5. O valor da propriedade Encapsulamento esteve estável da versão 0.95 para a versão 1.0 e teve uma leve queda da versão 1.01 para a versão 1.05. Isso não é um bom indicador, pois mostra que os atributos e métodos estão mais disponíveis para outras classes, o que aumenta a complexidade e dificulta o processo de depuração[25]. De forma complementar, Shatnawi e Alzu'bi[24] citam que a falta de encapsulamento viola a segurança de classes e, consequentemente, a possibilidade de defeitos de *software*.

O resultado da Tabela 4 foi utilizado nas fórmulas dos atributos de qualidade descritos no modelo QMOOD. Assim, foram obtidos os resultados dos atributos de qualidade em cada versão do Modelo de Objetos, os quais são exibidos na Tabela 5.

Os atributos de qualidade Flexibilidade, Funcionalidade, Extensibilidade e Reusabilidade estão agrupados na Figura 3 para uma análise temporal. O atributo de qualidade Facilidade de Compreensão foi separado no Figura 4 para ter uma visualização mais adequada em virtude de conter valores negativos.

Os atributos de qualidade Reusabilidade e Funcionalidade aumentaram da versão anterior para a

Tabela 5 - Resultado dos atributos de qualidade com base no QMOOD.

Atributo de Qualidade	Versão do Modelo de Objetos			
	0.95	**1.0**	**1.0.1**	**1.0.5**
Facilidade de Compreensão Compreensão	-0,99	-1,06	-1,58	-1,77
Flexibilidade	1,00	0,97	1,02	0,83
Funcionalidade	1,00	1,07	1,43	1,59
Extensibilidade	1,00	0,79	1,26	1,14
Reusabilidade	1,00	1,13	1,43	1,75

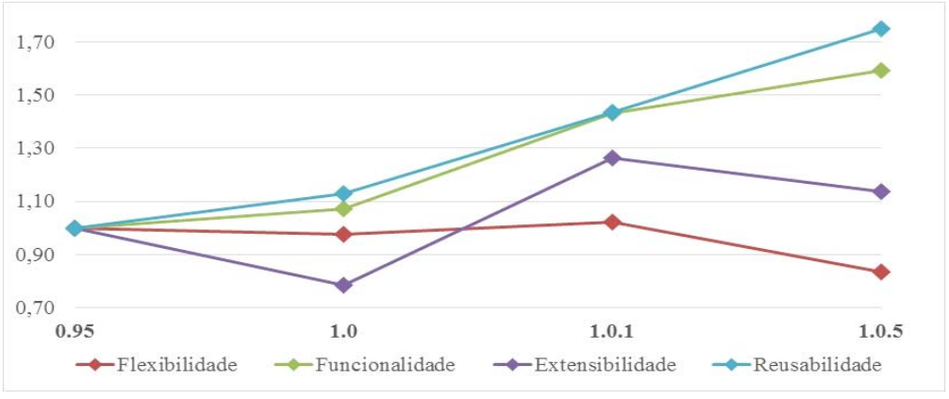

Figura 3 - Resultado dos atributos de qualidade Flexibilidade, Funcionalidade, Extensibilidade e Reusabilidade.

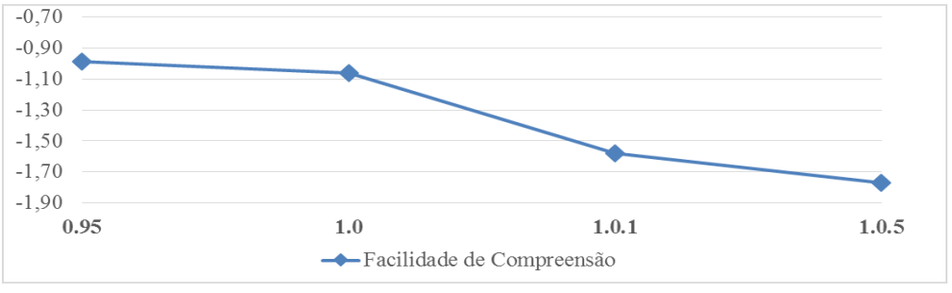

Figura 4 - Resultado do atributo de qualidade Facilidade de Compreensão.

próxima versão. Como as novas versões do Modelo de Objetos da openEHR incorporaram novos recursos, então, esperava-se que fossem incrementados os valores desses atributos. Além disso, problemas de projeto da versão anterior são corrigidos na versão seguinte. De acordo com Bansiya e Davis[13] existem duas razões principais para novas versões: implementação de novos recursos e correção de *bugs* que foram descobertos em versões anteriores.

O atributo de qualidade Flexibilidade e Extensibilidade tiveram uma tendência de queda, com exceção da versão 1.0.1. Os principais fatores foram as propriedades de projeto Acoplamento e Polimorfismo. O Acoplamento aumentou indicando uma maior interdependência entre os objetos. Já o polimorfismo diminuiu, indicando que existem métodos que não utilizam recursos já implementados pela classe pai. Além disso, a propriedade de projeto Encapsulamento impactou, exclusivamente, no atributo de qualidade Flexibilidade, visto que os dados e métodos ficaram mais expostos e são utilizados por outras classes. Nesse sentido, afeta a capacidade do projeto ser adaptado para novos recursos em uma reestruturação.

Já com relação ao atributo Facilidade de Compreensão, os valores diminuíram da versão anterior para a versão seguinte, conforme exibido no Figura 4. Este comportamento é explicado pela adição de novos recursos no Modelo de Objetos, pois foram incluídas muitas classes e métodos nas novas versões, o que fez com que elas sejam mais difíceis de entender e aprender. Entretanto, conforme citado por Bansiya e Davis[13], é esperado que uma versão madura reverta a tendência de queda desse atributo após várias versões iniciais.

CONCLUSÃO

Este trabalho avaliou a qualidade de Projeto Orientado a Objeto (POO) do Modelo de Objetos da openEHR, utilizando o *framework* Java *Reference Implementation* (JRI) disponibilizado pela Fundação openEHR. A qualidade de POO é percebida por pessoas relacionados ao projeto e desenvolvimento de sistemas, ou seja, não envolve usuários de sistema. A avaliação da qualidade de POO foi dividida em duas partes: primeiro foram aplicadas as estratégias de detecção de problemas de POO em quatro versões do JRI. Posteriormente, foi feita uma avaliação dos atributos de qualidade de POO com base no modelo de qualidade QMOOD.

Na parte relativa à aplicação das estratégias de detecção de problemas de POO, foram identificados seis problemas de projetos do total de dez propostos nesta avaliação. Os problemas de POO encontrados foram Classe Deus, Inveja de Funcionalidade, Classe de Dados, Classe Cérebro e Método Cérebro. Na análise dos resultados foram citadas algumas técnicas de correção (*refactoring*) dos problemas de POO com base na literatura. Não foram encontrados problemas de POO Acoplamento Intensivo, Cirurgia de Espingarda, Herança do Pai Recusada e Quebrador de Tradição. O que permite concluir que, de uma maneira geral, os problemas de POO aumentaram de acordo com o aumento da complexidade do Modelo de Objetos da openEHR.

Na segunda parte, foram avaliados os atributos de qualidade Facilidade de Compreensão, Flexibilidade, Funcionalidade, Extensabilidade e Reusabilidade com base no modelo de qualidade QMOOD. No modelo de qualidade QMOOD é esperado que os atributos de qualidade aumentem ao longo das versões do modelo avaliado. Nesse sentido, os atributos de qualidade Reusabilidade e Funcionalidade do Modelo de Objetos satisfizeram as expectativas. Já os atributos de qualidade Extensabilidade e Flexibilidade do Modelo de Objetos mostraram-se instáveis. Por fim, o atributo de qualidade Facilidade de Compreensão do Modelo de Objetos teve queda em todas as versões. Entretanto, autores ressaltam que essa situação é normal até que uma versão madura reverta a tendência de queda desse atributo. Portanto, conclui-se que o Modelo de Objetos da openEHR tem ganho de novos recursos e tem a capacidade de reutilizar módulos já existentes para resolver um novo problema com pouco esforço. Entretanto, novos requisitos em recursos já existentes podem ser mais trabalhosos, assim como a adaptação do projeto para novos recursos. Além disso, o projeto apresenta uma dificuldade de ser aprendido e compreendido devido ao aumento constante de sua complexidade.

Este trabalho tem como benefício auxiliar qualquer organização que pretenda implementar um sistema de RES com base no Modelo de Objetos da openEHR, já que os resultados apresentados estão relacionados diretamente com o projeto e desenvolvimento de sistemas. Como são evidenciados os atributos de qualidade e os problemas de POO, é possível ter uma dimensão das facilidades e dos desafios que serão encontrados no seu projeto e na sua implementação. Um desafio, por exemplo, é lidar com o atributo de qualidade Facilidade de Compreensão, que indica que uma equipe inexperiente não deva ser envolvida no projeto. Por outro lado, o atributo de qualidade Funcionalidade mostra que a organização poderá contar com a facilidade de novos recursos sendo incorporados.

Uma sugestão de trabalho futuro seria realizar a mesma avaliação de qualidade de POO do modelo de referência da norma ISO 13606. A norma ISO 13606 foi baseada na implementação do pré-padrão europeu CEN 13606 e foi especificada pelo comitê técnico ISO/TC 215 de informática em saúde. Como o modelo especificado pela norma ISO 13606 é uma simplificação do modelo proposto pela fundação openEHR, seria interessante verificar se a simplificação da norma ISO 13606 permitiu obter uma avaliação diferente da qualidade de POO.

REFERÊNCIAS

1. Santos MR. Sistema de registro eletrônico de saúde baseado na norma ISO 13606: aplicacões na Secretaria de Estado de Saúde de Minas Gerais [Internet] [tese]. Belo Horizonte: Escola de Ciência da Informação, Universidade Federal de Minas Gerais; 2011 [citado 2013 mar 30]. Disponível em: http://dspace.lcc.ufmg.br/dspace/bitstream/1843/ECIC-

8L8HFJ/1/tese_eci_ufmg____marcelo_rodrigues_dos_santos___2011.pdf
2. Kalra D. Electronic health record standards. Yearb Med Inform. 2006;136-44.
3. Velte LM. Electronic health record repository based on the openEHR standard [Internet] [Dissertação]. Aveiro: Universidade de Aveiro; 2011 [cited 2013 abr 2]. Available from: http://ria.ua.pt/handle/10773/7479
4. Thurston LM. Flexible and extensible display of archetyped data: The openEHR presentation challenge. In: Proceedings of the Health Informatics Conference -HIC [Internet]. Health Informatics Society of Australia; 2006 Jul 25; Adelaide. [cited 2013 mar 20]. p. 28-36. Available from: http://my.openehr.org/wiki/download/attachments/2949261/006_Thurston.pdf
5. openEHR. openEHR - Foundation [Internet]. 2013a [citado 2013 abr 4]. Disponível em: http://www.openehr.org/about/foundation
6. Atalag K, Yang HY, Warren J. Assessment of software maintainability of openEHR based health information systems - A case study in endoscopy. e-J Health Inform. 2011;7(1):e3.
7. Kalra D, Beale T, Heard S. The openEHR Foundation. Stud Health Technol Inform. 2005;115:153-73.
8. 8. Mishra D, Mishra A. Object-Oriented inheritance metrics in the context of cognitive complexity. Fundam Inform. 2011;111(1):91-117.
9. Chidamber SR, Darcy DP, Kemerer CF. Managerial use of metrics for object-oriented software: an exploratory analysis. Softw Eng IEEE Trans On. 1998;24(8):629-39.
10. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. Experimentation in software engineering. Berlim: Springer; 2012.
11. Gousios G, Spinellis D. Conducting quantitative software engineering studies with Alitheia Core. Empir Softw Eng. 2013;19(4):885-925.
12. Ko AJ, LaToza TD, Burnett MM. A practical guide to controlled experiments of software engineering tools with human participants. Empir Softw Eng. 2013;20(1):110-41.
13. Bansiya J, Davis CG. A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng. 2002;28(1):4-17.
14. Alshammari B, Fidge C, Corney D. Security metrics for object-oriented class designs. In: 9th International Conference on Quality Software [Internet]. 2009 Ago 24-25; Jeju, Korea. [cited 2013 jun 11]. p. 11-20. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5381538
15. Bertrán IM. Avaliação da qualidade de software com base em modelos UML [Internet] [dissertação]. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro; 2009 [cited 2013 mar 26]. Available from: http://www2.dbd.puc-rio.br/pergamum/biblioteca/php/mostrateses.php?open=1&arqtese=0711290_09_Indice.html
16. Lanza M, Marinescu R. Object-Oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems [Internet]. Springer; 2006 [cited 2013 jun 11]. Available from: http://books.google.com.br/books?hl=pt-BR&lr=&id=gdLbgnaMaa0C&oi=fnd&pg=PA1&dq=Object-Oriented+Metrics+in+Practice+autor:LANZA&ots=s_vMtJElsQ&sig=wqqIV07_68oqUDtQOGHo8n1c6uU
17. Marinescu R. Detection strategies: metrics-based rules for detecting design flaws. Proceedings of the 0th IEEE International Conference on Software Maintenance [Internet]. 2004 Sep 11-14; Chicago, Illinois, USA. [cited 2013 jun 11]. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1357820
18. Chen R. openEHR Reference Model Java ITS [Internet]. 2006 [cited 2013 mar 20]. Available from: http://research.behdasht.gov.ir/uploads/256_1197_openEHR-JavaITS.pdf
19. InFusion [Internet]. 2013 [cited 2013 ago 14]. Available from: http://www.intooitus.com/products/infusion
20. CKJM extended [Internet]. 2011 [cited 2013 sep 21]. Available from: http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/
21. Shatnawi R, Li W. An empirical assessment of refactoring impact on software quality using a hierarchical quality model. Int J Softw Eng Appl. 2011;5(4):127.
22. Dallal JA. Qualitative analysis for the impact of accounting for special methods in object- oriented class cohesion measurement. Int J Comput Trends Technol. 2013;8(2):327-36.
23. Gulia P, Chhillar R. Design based object-oriented metrics to measure coupling and cohesion. Int J Eng Sci Technol. 2011;3(11):8178-85.
24. Shatnawi R, Alzu'bi A. A verification of the correspondence between design and implementation quality attributes using a hierarchal quality model. IAENG Int J Comput Sci. 2011;38(3):225-33.
25. Mwangi W, Wafula J, Waweru NS. An effort prediction framework for software code quality measurement based on quantifiable constructs for object oriented design. Int J Comput Trends Technol. 2014;10(1):36-52.