Identificação de Bad Smells em Softwares a partir de Modelos UML

Aluno: Henrique G. Nunes[1], Orientadora: Mariza A. S. Bigonha[1], Co-orientadora: Kecia Aline Marques Ferreira[2]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) - Belo Horizonte – MG – Brasil

[2] Departamento de Computação – Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) - Belo Horizonte – MG – Brasil.

Aluno: henrique.mg.bh@gmail.com, Orientadora: mariza@dcc.ufmg.br, Co-orientadora: kecia@decom.cefetmg.br

Ingresso: 1º semestre de 2012, Previsão de Conclusão: 1º semestre de 2014

Data de aprovação da Proposta de Dissertação: 20/12/2012

Resumo. *Métricas de software podem auxiliar na identificação de desvios de projeto, conhecidos na literatura como bad smells. Alguns trabalhos têm utilizado essa abordagem para a avaliação de código-fonte. Todavia, é importante a existência de métodos que permitam a identificação de problemas estruturais nas fases iniciais do ciclo de vida do software. A dissertação apresentada neste artigo visa contribuir nesse aspecto, propondo um método e uma ferramenta para a identificação de bad smells, via métricas de software, em sistemas orientados por objetos a partir de modelos UML.*

Evento(s) do CBSoft relacionado(s): SBES, WTDSoft

Identificação de Bad Smells em Softwares a partir de Modelos UML

Henrique G. Nunes[1], Mariza A. S. Bigonha[1], Kecia Aline Marques Ferreira[2]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) - Belo Horizonte – MG – Brasil

[2] Departamento de Computação – Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) - Belo Horizonte – MG – Brasil.

henrique.mg.bh@gmail.com, mariza@dcc.ufmg.br, kecia@decom.cefetmg.br

Resumo. *Métricas de software podem auxiliar na identificação de desvios de projeto, conhecidos na literatura como bad smells. Alguns trabalhos têm utilizado essa abordagem para a avaliação de código-fonte. Todavia, é importante a existência de métodos que permitam a identificação de problemas estruturais nas fases iniciais do ciclo de vida do software. A dissertação apresentada neste artigo visa contribuir nesse aspecto, propondo um método e uma ferramenta para a identificação de bad smells, via métricas de software, em sistemas orientados por objetos a partir de modelos UML.*

1. Caracterização do Problema

Assim como nos códigos-fonte, as métricas obtidas a partir de modelos *UML* podem permitir analisar confiabilidade, manutenibilidade e complexidade de um sistema. Dentre os diagramas da UML, destaca-se o diagrama de classes que representa classes de objetos e suas relações (Yi et al., 2004). A análise desse diagrama pode fornecer métricas relacionadas à manutenibilidade e complexidade do software no início do projeto, como o número total de classes, métodos e relacionamentos, por exemplo. Porém, em um projeto real é inviável calcular tais métricas manualmente (Girgis et al., 2009).

O objetivo desta dissertação é definir um método de identificação de *bad smells* em softwares a partir de modelos *UML*. Para a definição desse método, inicialmente serão identificados os principais *bad smells* relacionados à manutenibilidade de software. A partir daí, serão identificadas as métricas de software que podem auxiliar a detectá-los em modelos *UML*, bem como os valores referência propostos na literatura para tais métricas. Uma ferramenta para aplicação do método proposto está em fase de desenvolvimento.

2. Fundamentação Teórica

Fowler (1999) define *bad smell* como um indicador de possível problema estrutural em código-fonte, que pode ser melhorado via refatoração. Dentre os *bad smells* descritos na literatura, foram selecionados inicialmente para a realização deste trabalho:

- ***God Class***: classes que tendem a centralizar a inteligência do sistema (Marinescu, 2002).

- ***Shotgun Surgery***: a mudança em uma classe que gera pequenas mudanças em muitas outras classes (Marinescu, 2002).
- ***Indecent Exposure***: quando uma classe revela dados internos (Hamza, 2008).

Métrica de software é um valor numérico relacionado a algum atributo de software, permitindo, dentre outros aspectos, a comparação entre atributos da mesma natureza (Sommerville, 2011) e a identificação de componentes com *bad smells*.

Marinescu (2004) define estratégia de detecção como "*uma expressão quantificável de uma regra, que permite avaliar se fragmentos de código estão de acordo com essa regra*". Para definir formalmente alguns *bad smells*, Marinescu (2004) e Lanza & Marinescu (2006) usam técnicas das estratégias de detecção como mecanismos de filtragem e composição, que aplicam métricas e valores referência. Bertrán (2009) afirma que por meio da adaptação de estratégias de detecção definidas por Marinescu (2004), projetistas podem localizar diretamente classes e métodos em modelos *UML* afetados por um *bad smell* particular, em vez de deduzir o problema a partir de um extenso conjunto de valores anormais de métricas.

3. Trabalhos Relacionados

Há na literatura algumas ferramentas que têm como objetivo fornecer dados sobre um projeto de software por meio do uso de métricas de modelos *UML*. Dentre elas destacam-se as propostas por Girgis et al. (2009), Soliman et al. (2010) e Nuthakki et al. (2011), que realizam coleta de métricas em diagramas de classes. Porém, essas ferramentas limitam-se à coleta de métricas, sem reportarem recomendações de melhorias nos projetos.

Marinescu (2002) realizou um dos principais trabalhos sobre estratégias de detecção de *bad smells* em *software* orientado por objetos a partir de métricas de software. As estratégias definidas por ele se baseiam em definir valores limites para as métricas utilizadas na detecção de *bad smells*. Porém, ele não define precisamente os valores limites a serem considerados.

Bertrán (2009) utiliza métricas de software para definir regras que identifiquem *bad smells* em diagramas de classes. Todavia, da mesma forma que Marinescu (2002), não são definidos, de forma sistemática, valores referência das métricas para a identificação dos *bad smells*. O trabalho de Bertrán (2009) explora a identificação de problemas estruturais em *software* a partir de modelos *UML*, todavia é restrito na quantidade de métricas em que se baseia.

O trabalho proposto nessa dissertação visa propor melhorias nos métodos de identificação de problemas arquiteturais em software orientado por objetos a partir de modelos *UML*. Para isso, será definido um método baseado em métricas e seus valores referência. A principal diferença entre a abordagem adotada no presente trabalho e a de trabalhos anteriores é que serão avaliadas novas possíveis métricas para representar os *bad smells* definidos por Fowler (1999), não se limitando ao que é definido em Marinescu (2002) e Bertrán (2009). Outra diferença será a escolha de métricas com valores referência definidos na literatura, em vez de utilizar valores arbitrários.

4. Estado Atual do Trabalho

Para a definição do modelo, foi realizada uma revisão da literatura para a identificação de *bad smells* a serem considerados, bem como as métricas a serem utilizadas. Os valores referência utilizados são aqueles definidos por Ferreira et al. (2012). A ferramenta para aplicação do modelo está em desenvolvimento e já permite a realização de análise dos *bad smells* que foram identificados na Seção 2. As métricas utilizadas e os *bad smells* que elas identificam estão na Tabela 1.

Tabela 1. Bad Smells e métricas utilizados nos experimentos preliminares

Bad Smell	**Referências**	**Métricas**	**Valores Referência**
God Class (Large Class)	Marinescu (2002) e Fowler(1999)	Nº de métodos públicos (NMP)	médio: 11 a 40 ruim: > 40
		Nº de conexões aferentes (NCA)	médio: 2 a 20 ruim: >20
Shotgun Surgery	Marinescu (2002)	Nº de conexões aferentes (NCA)	médio: 2 a 20 ruim: >20
Indecent Exposure	Hamza (2008)	Nº de atributos públicos (NAP)	médio: 1 a 10 ruim: >10

As questões de pesquisa avaliadas no experimento preliminar foram:

RQ1: Os resultados obtidos automaticamente pela ferramenta estão de acordo com o que foi identificado pelos especialistas?

RQ2: A utilização dessas métricas e seus valores referência foram úteis para identificação de *bad smells*?

A ferramenta desenvolvida tem como objetivo identificar *bad smells* em modelos *UML* a partir de diagramas de classes. Como não havia um repositório de diagramas de classes a disposição, utilizou-se a engenharia reversa do *software Enterprise Architecture* para que fosse possível a realização do experimento e para verificar se a metodologia adotada identifica *bad smells* nos diagramas. Exportou-se o diagrama de classes para o formato de arquivo *XMI*, por ser o modelo mais utilizado nos estudos avaliados do referencial teórico.

Os estudos preliminares foram realizados no diagrama de classes obtido a partir da engenharia reversa do *software* Mobile Media (2013), Versão 9. O *software* foi desenvolvido em *Java*, possui 56 classes e 4 interfaces, e trata-se de um aplicativo móvel para gerenciar imagens, músicas e vídeos.

Dois especialistas avaliaram o diagrama de classes para apontar classes com os *bad smells* avaliados. O primeiro especialista é um aluno de graduação que já cursou disciplinas relacionadas a programação modular e trabalha em projetos de iniciação científica na área de linguagens de programação. O segundo especialista é um Analista de Sistemas, que possui experiência de 8 anos em diagramas *UML* e programação Orientada por Objetos. Ambos apontaram as classes problemáticas em seu ponto de vista. Em seguida, o diagrama de classes foi submetido à análise da ferramenta. Os resultados reportados pela ferramenta foram comparados com as conclusões obtidas pela análise dos especialistas.

5. Avaliação dos Resultados

Para avaliação do *Indecent Exposure*, utilizou-se a métrica NAP como regra. Sete classes atingiram valores regulares para a métrica, variando entre 2 a 8. Embora elas não apresentem valores ruins para a métrica, elas não foram implementadas dentro dos valores ideias para a métrica. Uma classe atingiu um valor acima do valor referência considerado crítico, pois declara 17 atributos como públicos. As classes identificadas pelos especialistas estavam todas dentro dos valores ruins ou regulares identificados pela ferramenta.

Para calcular o *Shotgun Surgery*, utilizou-se a métrica de cálculos de conexões aferentes. De acordo com a ferramenta proposta, duas classes estão com valores ruins para a métrica NCA, se comparadas com os valores referência definidos por Ferreira et al. (2012), sendo 24 e 25 os valores encontrado para as duas. Pela ferramenta, ainda foram identificadas outras classes com valores entre 16 a 9 como valores regulares. Todas essas classes foram citadas na avaliação dos especialistas. Porém, outras classes com valores de 2 a 6 foram apontadas como regulares pela ferramenta, mas não foram citadas pelos especialistas.

A métrica NMP associada à NCA possibilita a identificação da ocorrência do *bad smell God Class*. Nos experimentos, uma classe apresentou valores razoáveis em ambas as métricas. Duas outras classes possuem um nível regular na métrica NMP e atingiram valores ruins na métrica NCA. O resultado é confirmado pela análise dos especialistas, que concluíram que a estrutura de ambas pode ser dividida em outras classes.

A resposta para a primeira questão de pesquisa é positiva, pois a maioria das classes avaliadas como problemáticas pelos especialistas atingiram níveis médio e ruim pela avaliação da ferramenta. A resposta da segunda questão de pesquisa também é positiva, pois foi possível identificar classes problemáticas utilizando os valores referência definidos na literatura. Esse estudo preliminar indica que o método proposto permite identificar *bad smells* a partir de modelos *UML* satisfatoriamente. Todavia, outros experimentos necessitam ser realizados. Além disso, outros *bad smells* deverão ser analisados no estudo.

6. Contribuições

O trabalho realizado nessa dissertação visa contribuir com a definição de um método para identificação de problemas estruturais em software nas fases iniciais do seu ciclo de vida. Os resultados desse trabalho gerarão as seguintes contribuições principais:

1. Definição um método de identificação de *bad smell* a partir de diagramas de classe da UML, utilizando métricas de software e seus valores referência.
2. Disponibilização de uma ferramenta que dê suporte para engenheiros de *software* na identificação de *bad smells* no início do projeto.

7. Conclusões

Essa dissertação de mestrado tem por objetivo a definição de um método para identificação de problemas estruturais em software nas fases iniciais do projeto. O método proposto consiste na identificação de *bad smells* em modelos UML, aplicando-se métricas de software e seus respectivos valores referência.

Este artigo relata os resultados preliminares do trabalho. Os resultados da análise dos especialistas e da ferramenta foram parecidos. Na continuidade do trabalho, será implementada a coleta de outras métricas e serão estudados outros *bad smells* que possam ser identificados com auxílio delas e de seus valores referência. Serão realizados experimentos com outros *softwares* abertos a fim de se avaliar o método proposto.

Referências

Bertrán, I. M. Avaliação da qualidade de software com base em modelos uml. Dissertação de Metrado da Puc Rio, Rio de Janeiro, 2009.

Ferreira, K. A. M. ; Bigonha, M. A. S. ; Bigonha, R. S. ; Mendes, L. F. O. ; Almeida, H. C. . Identifying thresholds for object-oriented software metrics. The Journal of Systems and Software, v. 85, p. 244-257, 2012.

Fowler, Martin. Refactoring: improving the design of existing code. Addison-Wesley Professional, 1999.

Girgis, Moheb R., Tarek M. Mahmoud, and Rehab R. Nour. "UML class diagram metrics tool." Computer Engineering & Systems, 2009. ICCES 2009. International Conference on. IEEE, 2009.

Hamza, H., et al. "Code smell eradication and associated refactoring." Proceedings of the European Computing Conference (ECC). 2008.

Lanza, M.; Marinescu, R. & Ducasse, S. Object-Oriented Metrics in Practice. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

Marinescu, Radu. "Measurement and quality in object-oriented design." Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on. IEEE, 2005.

Marinescu, R. Detection strategies: metrics-based rules for detecting design flaws. In Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on, pp. 350 – 359, 2004.

Mobile Media (2013). http://sourceforge.net/projects/mobilemedia/. Acessado em 31/07/2013.

Nuthakki, M. K.; Mete, M.; Varol, C. & Suh, S. C. Uxsom: Uml generated xml to software metrics. SIGSOFT Softw. Eng. Notes, 36(3):1—6, 2011.

Soliman, T.; El-Swesy, A. & Ahmed, S. Utilizing ck metrics suite to uml models: A case study of microarray midas software. In Informatics and Systems (INFOS), 2010 The 7th International Conference on, pp. 1 –6, 2010.

Sommerville, Ian. Engenharia de software. Vol. 8. São Paulo: Addison Wesley, 2007.

Yi, T.; Wu, F. & Gan, C. A comparison of metrics for uml class diagrams. SIGSOFT Softw. Eng. Notes, 29(5):1—6, 2004.