Pontifícia Universidade Católica de Minas Gerais

Engenharia de Software

**Resenha do Artigo**

# Big Ball of Mud

**Autores:** Brian Foote e Joseph Yoder

**Aluno:** Luiz Nery
**Disciplina:** Projeto de Software
**Professor:** João Paulo Aramuni

Belo Horizonte, 9 de setembro de 2024

# Sumário

# 1 Resumo

O artigo *Big Ball of Mud*, de Brian Foote e Joseph Yoder, apresenta uma análise sobre arquiteturas de software caóticas, denominadas *Big Ball of Mud*, que surgem pela falta de planejamento estrutural e pela pressão do mercado. Embora essas arquiteturas sejam vistas de forma negativa por teorias tradicionais, os autores mostram que elas são comuns em sistemas reais devido a prazos apertados, mudanças constantes nos requisitos e necessidade de produtividade. Além de descreverem os padrões arquitetônicos típicos dessas soluções, o artigo reflete sobre como desenvolvedores podem evoluir esses sistemas, equilibrando agilidade e qualidade, e propõe abordagens para minimizar os efeitos negativos. Em conclusão, Foote e Yoder argumentam que o entendimento dos fatores que levam à criação de uma *Big Ball of Mud* pode guiar decisões estratégicas de design e manutenção de software.

# 2 Introdução

O conceito de *Big Ball of Mud* (BBoM) é uma metáfora para sistemas de software desorganizados, cuja arquitetura surge de maneira descontrolada, muitas vezes sem um plano claro. No artigo de Foote e Yoder (1997), os autores discutem a prevalência desse tipo de arquitetura em diversas organizações de software, em contraste com a teoria, que preconiza um planejamento mais estruturado. Eles detalham como fatores como prazos apertados e constantes alterações nos requisitos levam ao surgimento de arquiteturas confusas, sem modularidade ou coesão.

A contribuição principal do estudo está na análise das razões que levam à formação dessas arquiteturas desorganizadas e nos impactos negativos sobre a manutenção e evolução do software. O artigo é organizado para, primeiramente, definir o conceito de *Big Ball of Mud*; depois, investigar suas causas e consequências; e, por fim, explorar como os desenvolvedores podem lidar com esses sistemas, oferecendo estratégias para reverter ou amenizar os danos causados por esse tipo de arquitetura.

# 3 Revisão da Literatura

A literatura sobre *Big Ball of Mud* revela que essa arquitetura surge de um desenvolvimento ad-hoc, onde a falta de planejamento leva a sistemas complexos e de difícil manutenção. Segundo Foote e Yoder (1997), sistemas com pouca modularidade e alto acoplamento são consequências diretas da

ausência de um design coerente desde o início. Essas características tornam os sistemas propensos a se tornarem desordenados ao longo do tempo.

Estudos anteriores, como o de Brooks (1995), apontam que a ausência de um planejamento arquitetônico inicial rigoroso aumenta exponencialmente a complexidade do sistema, dificultando a evolução. McConnell (2004) complementa ao destacar que essa prática pode ser uma resposta às pressões do mercado, que demandam agilidade em detrimento da qualidade estrutural. Foote e Yoder ampliam esse debate, argumentando que muitos sistemas acabam funcionando sem uma arquitetura clara, mas se tornam problemáticos a longo prazo.

Dessa forma, a literatura confirma o trade-off entre a necessidade de entrega rápida e a sustentabilidade do software. A prática de priorizar prazos pode parecer eficaz em um primeiro momento, mas resulta em sistemas cada vez mais difíceis de modificar e expandir.

# 4 Análise Crítica

O artigo de Foote e Yoder apresenta um olhar crítico sobre o fenômeno *Big Ball of Mud*, sugerindo que, embora indesejável, essa arquitetura é um reflexo prático das necessidades comerciais. O texto explora não apenas as causas que levam à criação de sistemas desordenados, mas também as soluções possíveis para diminuir esses efeitos. A análise sugere que a falta de uma arquitetura planejada desde o início resulta em um acúmulo de complexidade, o que dificulta tanto a manutenção quanto a evolução do software.

Além disso, os autores discutem a dívida técnica que surge como consequência de adotar uma arquitetura desorganizada. Eles defendem que, mesmo que o *Big Ball of Mud* seja uma realidade em muitos projetos, é possível lidar com esses sistemas de maneira estratégica, implementando melhorias incrementais que, ao longo do tempo, resultam em um software mais estável e gerenciável.

# 5 Conclusão

O artigo "Big Ball of Mud"de Foote e Yoder conclui que sistemas de software desordenados são uma realidade frequente devido à falta de planejamento arquitetônico. A principal contribuição dos autores é a reflexão sobre como esses sistemas podem ser gerenciados e melhorados, mesmo quando já estão profundamente imersos no caos. O estudo sugere que, embora práticas rápidas possam ser vantajosas a curto prazo, elas frequentemente geram pro-

blemas complexos a longo prazo, criando dificuldades significativas de manutenção e evolução.

Os autores destacam a importância de uma abordagem mais planejada no desenvolvimento de software, visando reduzir os efeitos negativos de uma arquitetura desordenada. Como futuras direções, o artigo sugere a necessidade de explorar estratégias para lidar com a dívida técnica gerada por esses sistemas, oferecendo insights valiosos sobre como manter a agilidade sem sacrificar a sustentabilidade a longo prazo. Dessa forma, Foote e Yoder enfatizam que o equilíbrio entre pragmatismo e qualidade arquitetônica é essencial para a construção de sistemas de software duráveis.

# 6 Referências


1. ALEXANDER, C. – *The Timeless Way of Building*. Oxford University Press, Oxford, UK, 1979. http://www.oup-usa.org/

2. ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M. – *A Pattern Language*. Oxford University Press, Oxford, UK, 1977. http://www.oup-usa.org/

3. ALEXANDER, C. – *The Oregon Experiment*. Oxford University Press, Oxford, UK, 1988. http://www.oup-usa.org/

4. BECK, K. – *Smalltalk Best Practice Patterns*. Prentice Hall, Upper Saddle River, NJ, 1997.

5. BECK, K.; CUNNINGHAM, W. – *A Laboratory for Teaching Object-Oriented Thinking*. OOPSLA '89 Proceedings, New Orleans, LA, October 1-6 1989, pp. 1-6.

6. BOOCH, G. – *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, Redwood City, CA, 1994.

7. BRAND, S. – *How Buildings Learn: What Happens After They're Built*. Viking Press, 1994.

8. BROOKS, F. P., Jr. – *The Mythical Man-Month (Anniversary Edition)*. Addison-Wesley, Boston, MA, 1995.

9. COPLIEN, J. O. – *A Generative Development-Process Pattern Language*. First Conference on Pattern Languages of Programs (PLoP '94), Monticello, Illinois, August 1994. Pattern Languages of Program Design, edited by James O. Coplien and Douglas C. Schmidt. Addison-Wesley, 1995. http://heg-school.awl.com/cseng/swpatterns/

10. CUSUMANO, M. A.; SHELBY, R. W. – *Microsoft Secrets*. The Free Press, New York, NY, 1995.

11. FOOTE, B. – *Designing to Facilitate Change with Object-Oriented Frameworks*. Masters Thesis, University of Illinois at Urbana-Champaign, 1988. http://www-sal.cs.uiuc.edu/ foote

12. FOOTE, B.; OPDYKE, W. F. – *Life-cycle and Refactoring Patterns that Support Evolution and Reuse*. First Conference on Pattern Languages of Programs (PLoP '94), Monticello, Illinois, August 1994. Pattern Languages of Program Design, edited by James O. Coplien and Douglas C. Schmidt. Addison-Wesley, 1995. http://heg-school.awl.com/cseng/swpatterns/

13. FOOTE, B.; YODER, J. – *Architecture, Evolution, and Metamorphosis*. Second Conference on Pattern Languages of Programs (PLoP '95), Monticello, Illinois, September 1995. Pattern Languages of Program Design 2, edited by John Vlissides, James O. Coplein, and Norman L. Kerth. Addison-Wesley, 1996. http://heg-school.awl.com/cseng/swpatterns/

14. FOOTE, B.; YODER, J. – *The Selfish Class*. Third Conference on Pattern Languages of Programs (PLoP '96), Monticello, Illinois, September 1996. Pattern Languages of Program Design 3, edited by Robert Martin, Dirk Riehle, and Frank Buschmann. Addison-Wesley, 1997. http://heg-school.awl.com/cseng/swpatterns/

15. GABRIEL, R. P. – *Lisp: Good News Bad News and How to Win Big*. http://www.cs.washington.edu/homes/ctkwok/worseisbetter/worseisbetter.html

16. GABRIEL, R. P. – *Patterns of Software: Tales from the Software Community*. Oxford University Press, Oxford, UK, 1996. http://www.oup-usa.org/

17. GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. – *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995. http://www.awl.com/cp/Gamma.html

18. INGALLS, D. H. H. – *The Evolution of the Smalltalk Virtual Machine*. Smalltalk-80: Bits of History, Words of Advice, edited by Glenn Krasner. Addison-Wesley, 1983.

19. JOHNSON, R. E.; FOOTE, B. – *Designing Reusable Classes*. Journal of Object-Oriented Programming, vol. 1, no. 2, June/July 1988, pp. 22-35. http://laputa.isdn.uiuc.edu/drc.html

20. ROBERTS, D.; JOHNSON, R. E. – *Evolve Frameworks into Domain-Specific Languages*. Third Conference on Pattern Languages of Programs (PLoP '96), Monticello, Illinois, September 1996. Pattern Languages of Program Design 3, edited by Robert Martin, Dirk Riehle, and Frank Buschmann. Addison-Wesley, 1997. http://heg-school.awl.com/cseng/swpatterns/