Estudo de Visualizações da Evolução de Códigos Fonte de Software

Marcello Henrique Dias de MOURA *
Hugo Alexandre Dantas do NASCIMENTO †
Instituto de Informática – Universidade Federal de Goiás
Goiânia, Goiás, Brasil
{marcellomoura,hadn}@inf.ufg.br

Palavras chave: Visualização de Informações, Evolução de Código Fonte, Evolução de Software, Histórico de Desenvolvimento de Software.

1 Introdução

É difícil mostrar o crescimento de um *software*[1] e o esforço desprendido pelos desenvolvedores para a construção do mesmo. Várias abordagens têm sido utilizadas para tentar medir esse esforço e demonstrar o tamanho do sistema como, por exemplo, o uso de métricas de *software* [4]. Certas abordagens utilizam técnicas de visualização de informações [7] para apresentar tais dados.

Visualização de informações [1] é um processo de transformação de dados em uma estrutura visual permitindo ampliar a cognição sobre as mesmas e suas relações.

O presente artigo faz uma revisão de alguns trabalhos sobre o uso de visualização de informações com animação de imagens para demonstrar o tamanho, a complexidade e o esforço de programação de sistemas computacionais.

A revisão é a base para o desenvolvimento de uma nova solução computacional para identificação do padrão de crescimento de código e de outras estruturas de informação, como parte de um projeto de mestrado de um dos autores.

Na Seção 2 apresentamos as visualizações e logo em seguida, a conclusão na seção 3.

*Aluno de Mestrado
†Professor Orientador
[1]Programa de computador

2 Visualizações de *Software*

2.1 O projeto Code_swarm

O Code_swarm foi desenvolvido por Ogawa et. al. [6]. Tem como objetivo explorar dados usando técnicas de visualização de informação orgânica [3]. Outro objetivo do projeto é demonstrar interações dos desenvolvedores com o código fonte possibilitando identificar momentos de alto fluxo de trabalho.

Outros projetos foram utilizados como entrada pelo Code_swarm para analizar sua eficácia: O projeto de contrução do servidor de página web Apache, o de contrução do ambiente de desenvolvimento integrado Eclipse e da linguagem de programação Python. Esse último, permitiu claramente verificar o momento em que outros desenvolvedores começaram a trabalhar ativamente no projeto, que antes era feito quase que exclusivamente pelo seu criador, conforme ilustrado na Figura 1.

2.2 O projeto Gource

O Gource [2] que também é um *Software Livre*[2] é uma ferramenta para criação de vídeos para demonstração da evolução do código baseados no histórico de desenvolvimento de *software* com animações.

A representação visual é feita a partir da árvore de diretórios do projeto. Os subdiretórios são representados como galhos e os arquivos indicados por folhas na forma de esferas coloridas de acordo com sua extensão. Na medida que os desenvolvedores registram suas alterações (termo conhecido como *commit*[3]) um raio de luz incide sobre as folhas (arquivos) alteradas gerando uma sensação de construção ou manipulação da árvore.

A cor do raio de luz indica a ação do usuário: verde=adicionar, laranja=modificar e vermelho=excluir. Rótulos aparecem sobre os arquivos, desenvolvedores ou diretórios na medida em que eles são modificados e vão se apagando na medida em que saem do contexto. O campo de visão pode ser alterado para destacar um arquivo, um desenvolvedor ou acompanhar o projeto inteiro.

A Figura 2 ilustra o Gource aplicado aos dados de histórico do desenvolvimento de uma ferramenta *web*.

[2]Característica de um programa poder ser executado, estudado, alterado e distribuído sendo necessário somente a citação de seu autor.

[3]Submissão de um conjunto de mudanças em um código fonte

2.3 O projeto CodeCity

O projeto CodeCity [8] fornece uma visualização de código fonte usando uma cidade como metáfora. A estrutura do código fonte é representada como edificações na topologia de uma metrópole.

A visualização faz uso de orientação a objetos[4], onde o código é organizado em pacotes contendo classes e as classes, por sua vez, são compostas de métodos e atributos.

A representação visual é feita da seguinte maneira (ver Figura 3): As classes e interfaces são representadas como construções; os pacotes representam o piso; a saturação da cor do piso representa o nível de aninhamento dos pacotes e subpacotes; o número de métodos representa a altura das construções, enquanto que a largura representa o número de atributos; as extensões da linguagem ou bibliotecas são subúrbios da cidade; o sistema usa uma correlação entre a quantidade de classes, métodos e atributos e alguma inteligência para definir dimensão, posição, cor, saturação e transparência.

Casos de testes foram elaborados e aplicados. Um deles, em torno de 8.000 classes e mais de 130 mil linhas de código [9], demonstrou ser possível a utilização da visualização para grandes projetos. Toda essa estrutura visual integrada e interativa pode prejudicar a cognição se o código fonte a representar for extremamente extenso, apesar de ser possível uma visualização por demanda ou desabilitar algumas funcionalidades com o intuito de não sobrecarregar a apresentação.

2.4 O projeto StarGate

Em 2008, foi apresentado um projeto de visualização chamado StarGate [5]. Tal projeto é focado em desenvolvedores, chamado de atores, e tem como objetivo facilitar o entendimento das complexas interações entre esses atores e o repositório de *software*, além da evolução do repositório ao longo do tempo.

A estrutura de diretórios do projeto forma um desenho de anéis que representa um portal (*Gate*), os atores como estrelas (*Star*) ou esferas onde seu tamanho é proporcional ao número de modificações efetuadas e por fim sua posição é aproximada aos arquivos com quem mais interagiu conforme Figura 4.

Quando uma estrela é selecionada em torno dos anéis surge uma estrutura definida como *stardust* (poeira estelar) que são pontos que representam o histórico de evolução dos arquivos, sua cor se assemelha a do autor que efetuou modificações, quanto mais fora do centro, mais próximo do tempo atual, ou seja, segue a linha do tempo de dentro para fora.

[4]Orientação a Objetos é um paradigma de programação que imita (ou simula) o mundo real permitindo assim abstrair os conceitos que temos puramente intrínsecos para um mundo computacional.

Algums problemas foram encontrados como, por exemplo identificar a melhor ordem de apresentação dos diretórios visinhos que afetaria drásticamente a posição dos atores na visualização e deixaria um usuário familiarizado com a estrutura um tanto confuso. Existe também uma dificuldade em identificar e visualizar precisamente as alterações efetuadas pelos atores, a visualização parece boa para visão geral mas não para detalhes.

3 Conclusão

Analisamos visualizações recentes e representativas que permitem vislumbrar a evolução de código fonte e/ou analisar as estruturas de código de uma forma mais amigável e interativa.

As visualizações apoiam tanto os desenvolvedores como os gerentes de projetos de software na identificação de pontos de maior esforço, demandados pelas equipes e áreas do código de maior interesse. Elas possibilitam também a integração de um novo membro na equipe, o qual pode entender melhor a evolução da complexa estrutura já desenvolvida.

Nosso estudo, busca identificar padrões, técnicas e dificuldades, evolvidas na criação de visualizações da evolução de códigos fonte. É comum o crescimento e a complexidade de aplicações, portanto é evidente a utilidade de visualizações como ferramenta de apoio para equipe de desenvolvimento, permitindo também, disponibilizar de uma forma empírica um trabalho minucioso e especializado.

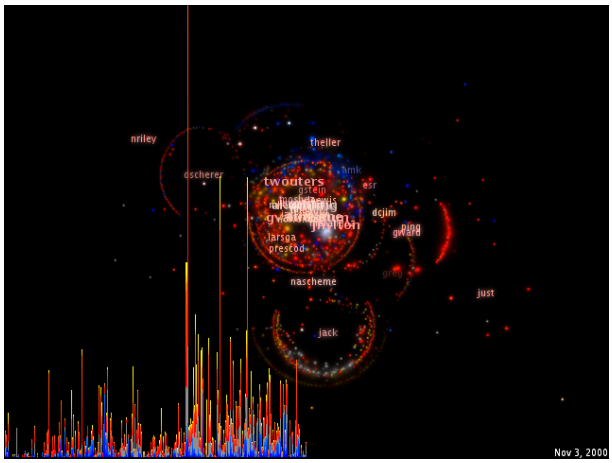

Figura 1: Momento de grande atividade no desenvolvimento da linguagem Python. http://www.vimeo.com/1093745

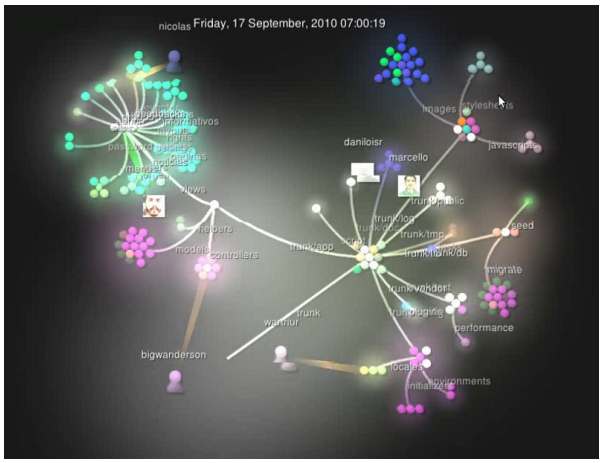

Figura 2: Visualização do desenvolvimento de um gerenciador de conteúdo usado pela Universidade Federal de Goiás.

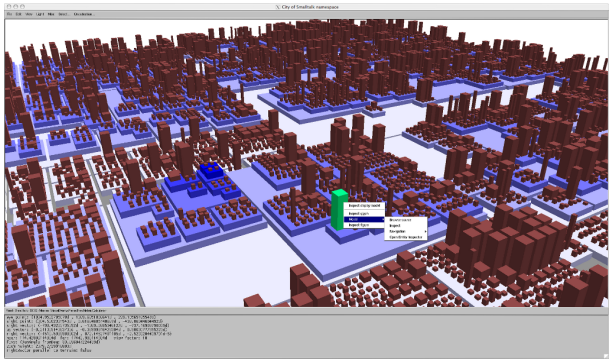

Figura 3: Uma interação com a interface do CodeCity.

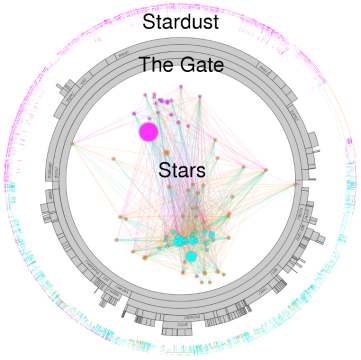

Figura 4: Uma exemplo de visualização contruída com StarGate.

Referências

[1] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Academic Press, London, 1999.

[2] W. R. Cook. Gource: visualizing software version control history. In *SPLASH - Systems, Programming and Applications*, pages 73–74, oct 2010.

[3] B. J. Fry. Master Thesis' Organic information design. Master's thesis, School of Architecture and Planning, Massachusetts Institute of Technology, 2000.

[4] C. Jones. *Software Assessments, Benchmarks, and Best Practices.* Addison-Wesley, 2000.

[5] M. Ogawa and K.-L. Ma. StarGate: A Unified, Interactive Visualization of Software Projects. In *In Proceedings of the 2008 IEEE Pacific Visualization Symposium*, pages 191–198, March 2008.

[6] M. Ogawa and K.-L. Ma. code_swarm: A Design Study in Organic Software Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1097–1104, 2009.

[7] R. Spence. *Information Visualization: Design for Interaction*. Prentice-Hall, Inc., 2007.

[8] R. Wettel and M. Lanza. Visualizing Software Systems as Cities. In *In Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis)*, pages 92–99, june 2007.

[9] R. Wettel and M. Lanza. Visual Exploration of Large-Scale System Evolution. In *In Proceedings of WCRE 2008 (15th Working Conference on Reverse Engineering)*, pages 219–228, oct 2008.