Resenha: Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies[1]

MAT08 - Evolução de Software (2012.2)

Joenio Marques da Costa

20 de dezembro de 2012

Resumo

Um bom design num sistema de software envolve, entre outras coisas, em ter uma boa modularização, isto permite principalmente que os desenvolvedores alterem o sistema de maneira segura sem interferir em outras partes do mesmo. Não é raro no entando encontrarmos sistemas com designs inadequados e de difícil manutenção e evolução, usualmente estas dificuldades são enfrentadas através de ferramentas IDE, analisadores léxicos ou histórico de sistemas de controle de versão. Estas abordagens ajudam a encontrar conceitos espalhados pelo código-fonte, mas levam sempre ao mesmo resultado: o desenvolvedor é apresentado a linhas de código-fonte que serão analisadas manualmente, este processo é trabalhoso e dificulta a localização de conceitos. Com isto em mente é proposto o *Grafo de Conceitos*, uma representação visual dos conceitos encontrados no código-fonte, efetiva, de fácil criação, manipulação e análise. Esta proposta foi validada através da criação de uma ferramenta chamada *FEAT*, ela cria, analiza e visualiza os *Grafos de Conceitos* de forma fácil e interativa. Em um cenário comum o desenvolvedor percorre os arquivos de código-fonte em busca dos conceitos relacionados a manutenção. Através da ferramenta *FEAT* o desenvolvedor apenas monta um modelo que será então convertido num *Grafo de Conceito*. Para avaliar a efetividade da ferramenta *FEAT* e os *Grafos de Conceitos* foram feitos vários estudos de caso que comprovaram a eficácia ao dar ao desenvolvedor a visão necessária ao fazer manutenção. Não vejo pontos fracos no artigo, mas não concordo com a real necessidade de ferramentas de apoio neste sentido, conhecimento técnico e experiência de desenvolvimento são os ingredientes comuns para manter e evoluir sistemas de software.

Referências

[1] M. P. R. G. C. Murphy, "Concern graphs finding and describing concerns using structural program dependencies," p. 11, 2002.