Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Coordenação de Pós-Graduação em Informática - COPIN

PROJETO DE SOFTWARE ORIENTADO A OBJETO

RESUMO - Conscientious Software
Abril de 2008

João Arthur Brunet Monteiro
Mestrado em Ciências da Computação - CEEI/UFCG
Campina Grande, Abril de 2008

1 Resumo

Este documento tem por objetivo sumarizar as idéias expostas no artigo *Conscientious Software* [1], além de apresentar minhas opiniões a respeito do assunto.

No artigo em questão, os autores advogam que inovações na construção de software devem ser implantadas de maneira a prover software que seja responsável pelo seu comportamento ao longo do tempo, usando técnicas como monitoração, recuperação automática de falhas, testes automáticos, configuração automática etc. Em minha opinião, o problema abordado pelos autores reside na dificuldade que se encontra hoje em dia na execução de tarefas como especificação, instalação, atualização e teste de software, que necessitam de mão de obra dos programadores, aumentando o custo da fabricação de software.

Para que essas inovações sejam implantadas é necessário, entre outras atividades, a criação de linguagens de alto nível que especifiquem como os componentes irão se comportar durante a evolução do software, descrevendo atividades como o momento certo para execução de testes, a maneira como os componentes vão se comunicar, quando irá se dar essa comunicação, além de várias outros aspectos dinâmicos do software. Exatamente por tratar-se de aspectos dinâmicos, é preciso também melhorar atividades como monitoração de software e conexão de componentes usando *very late binding*.

Os autores apresentam no artigo vários princípios para construção de *Conscentious Software*. Estes princípios e idéias são baseados nas mudanças que estão ocorrendo na construção de software. Por questões de espaço não colocarei todos neste resumo, mas achei por bem destacar os que, em minha opinião, são os principais pontos para suportar a construção de software consciente:

- Re-design contínuo durante o ciclo de vida do software.
- Arquiteturas que permitem atender à mudanças com menor esforço possível, além de serem adaptáveis facilmente.
- Empacotar todas as dependências usadas na construção do software para permitir futuras mudanças caso necessário. Esta atividade, em minha opinião, é um tanto difícil de ser adotada pelas empresas, uma vez que em muitas delas existe proteção do código fonte.
- Detecção de erros e recuperação a falhas em tempo de execução.
- Confecção e execução contínua de testes.

Este último ponto eu considero imprescindível para que software consciente seja concebido. Isto porque com a execução contínua de testes, percebe-se a falha mais rapidamente e, como já é de conhecimento, quanto mais cedo se percebe uma falha, menor é o custo de reparar a falta que a gerou.

Trata-se de um assunto bastante relevante na ciência da computação. De fato, outros autores já abordaram este caráter evolutivo dos software [2]. No entanto, para que seja realidade, este novo paradigma precisa ser implantado aos poucos, mesmo porque ele requer atividades que ainda são muito incipientes.

Uma conclusão interessante dos autores é que a computação ainda é uma ciência nova e em pleno crescimento, mas possui muitas limitações em seu estado atual. Por isso, acredito que pesquisas são necessárias em diversas áreas da computação para que o termo software consciente seja realidade. Ainda, acredito que avanços em direção a este novo paradigma já podem ser identificados.

Referências

[1] R. P. Gabriel and R. Goldman. Conscientious Software. *ACM SIGPLAN Notices*, 41(10):433–450, 2006.

[2] D. Parnas. Software aging. *Proceedings of the 16th international conference on Software engineering*, pages 279–287, 1994.