6
Conclusão

O termo transparência é muitas vezes utilizado no contexto da ciência da computação no sentido de ocultar informações e processos. Por exemplo, neste contexto, tornar uma funcionalidade transparente para o usuário significa deixá-la disponível sem que o usuário precise saber como ela funciona, apenas que faz o que o usuário espera. Nesse sentindo transparência é utilizada para enfatizar a facilidade de uso sem que haja preocupação com detalhes que poderiam demandar investimento de tempo pelos usuários. Neste trabalho utilizamos o termo transparência no sentido real da palavra, que é trazer luz ao que está oculto.

A transparência de software é um temo muito abrangente [Oliveira07], por isso neste trabalho procuramos focar especificamente na transparência do código fonte. Para atendermos ao requisito de transparência de código não basta disponibilizar o código fonte do software, também devemos fornecer meios que facilitem sua compreensão.

Ao procurarmos trabalhos relacionados na literatura descobrimos que a documentação de código fonte é baseada na adoção de estruturas pré-definidas, que armazenam descrições em linguagem natural, inseridas no código fonte. Estas estruturas são padronizadas para que permitam a atuação de ferramentas automatizadas na extração, tanto do código, quanto da documentação.

Nossa principal contribuição com este trabalho é um método de desenvolvimento baseado no refinamento de cenários. O objetivo deste método é produzir um documento único, o código fonte, onde teremos a nossa disposição a documentação, em forma de cenários integrados com o código, e o código propriamente dito. Este método também utiliza o *framework* MVC para uma melhor organização do código e o LAL para uma documentação complementar..

Outra contribuição importante foi o novo C&L, resultado do processo de re-engenharia realizado. Um software livre desenvolvido inteiramente no ambiente Lua-Kepler, com sua arquitetura dividida em camadas de acordo com o *framework* MVC e com cenários documentando seu código fonte, o que o tornou mais transparente a nível de código e, consequentemente, mais fácil de se

PUC-Rio - Certificação Digital Nº 0711275/CA

evoluir. Esperamos que o software mais transparente a nível de código atraia mais desenvolvedores interessados em participar de alguma forma do projeto.

E finalmente, acreditamos que esta experiência bem sucedida servirá de estímulo para futuros projetos de desenvolvimento de sistemas Web com o uso de Lua.

6.1. Comparação com trabalhos relacionados

Em [Knuth84] é proposta a linguagem WEB, que mistura uma linguagem de implementação com comentários. Com o uso de dois programas específicos é possível extrair de um arquivo WEB o código que será compilado e a documentação. Mas, segundo o próprio autor, esta linguagem foi propositalmente projetada para ser complexa, de modo que não pudesse ser usada por qualquer um, apenas por cientistas da computação.

Nosso objetivo ao desenvolver este trabalho foi totalmente o oposto. Adotamos o uso de cenários em linguagem natural como forma de anotação e de tecnologias como o *framework* MVC e Lua, para torná-lo simples e fácil de ser usado por qualquer um. Mas, diferente de [Knuth84], não possuímos uma ferramenta que extraia de forma automática a documentação do código fonte.

Seguindo a mesma linha, o trabalho exposto em [Cassino96] propõe um *framework* para construção de ferramentas de apoio a programação literária, e exemplifica a instanciação deste *framework* através do desenvolvimento de uma ferramenta chamada “nome”. A documentação final é apresentada em LaTeX [Lamport86].

Acreditamos que esta forma de documentação é menos dinâmica do que a que propomos, que permite a navegação através de elos. Além disto, como os cenários são inclusos no código em forma de comentários, eles podem ser compilados e executados juntamente com o código. Assim, só precisamos de um programa para extrair os cenários para a documentação, não há a necessidade de extração do código fonte.

O trabalho [Christoph04] propõe o uso de cenários [Leite00] como forma de anotar o código. Estes cenários podem ser construídos antes, depois ou concomitantemente com o código. A forma de anotação abordada é genérica, isto é, pode ser aplicada a softwares com qualquer arquitetura, cabe ao desenvolvedor adaptá-la as suas necessidades.

PUC-Rio - Certificação Digital Nº 0711275/CA

Em nosso trabalho utilizamos o mesmo modelo de cenários [Leite00], mas diferente de [Christoph04] nosso método é restrito a sistemas que pode ser descritos segundo o *framework* MVC. Além disto, em nossa proposta, os cenários são construídos antes do código fonte e então refinados durante o processo de escrita do código.

Os padrões [JavaDoc] e [PHPDoc] utilizam cabeçalhos com *tags* pré-definidas para anotar cada função. Nós utilizamos cenários para anotar não só funções, mas também arquivos e código HTML. Além disto, a utilização de cenários é muito mais abrangente, pois não se limita apenas a um cabeçalho. Os cenários, através de seus episódios, detalham passo a passo o funcionamento de todo o código.

Estes padrões permitem a transformação automática destes comentários em páginas HTML navegáveis. No nosso trabalho isto só é possível se houver um esforço extra para que os cenários inclusos no código sejam manualmente cadastrados na ferramenta C&L.

Em [Staa00] são definidos padrões para documentação de código através de comentários, para facilitar sua compreensão. Nosso trabalho também utiliza padrões como forma de documentação, estes padrões são os cenários. Esta padronização visa uniformizar o estilo de programação, facilitando o entendimento, manuseio e evolução de código escrito por terceiros.

6.2. Dificuldades encontradas

Encontramos dificuldades no início da codificação do novo C&L devido a falta de documentação e exemplos do uso da linguagem Lua para desenvolvimento de sistemas Web. Diferente de outras linguagens, como Java e PHP, que podemos obter respostas para a grande maioria de nossas dúvidas, de várias fontes diferentes, através de simples consultas na Web, as únicas fontes de documentação disponíveis são a página do próprio projeto Kepler e listas de discussão. Pouquíssima documentação foi encontrada de outras fontes e os exemplos são bem escassos e limitados.

Quando o nosso trabalho de re-engenharia do C&L começou, o projeto Kepler ainda estava em uma versão beta. Durante a codificação na versão beta encontramos falhas em alguns dos módulos do ambiente. Para resolvê-los, interagimos com a comunidade de desenvolvimento e outros usuários através da lista de discussão, reportando os problemas encontrados. Acompanhamos as

PUC-Rio - Certificação Digital Nº 0711275/CA

soluções adotadas e o lançamento da nova versão do ambiente já com as correções. Destacamos que a comunidade de desenvolvimento interage bastante com os usuários, e trabalha arduamente no sentido de resolver os problemas que aparecem o mais rapidamente possível, mas mesmo assim ainda leva um tempo considerável.

Durante a construção da nova arquitetura tivemos dificuldade em manter a integridade entre os cenários cadastrados no C&L e os cenários inclusos do código fonte. Toda vez que evoluíamos o cenário do código fonte, tínhamos que evoluir o mesmo cenário cadastrado no C&L e vice-versa. Desta forma, toda alteração teve que ser feita duas vezes, o que gerou um considerável trabalho extra.

6.3. Trabalhos Futuros

Como descrito na subseção anterior, uma das dificuldades encontradas no decorrer deste projeto foi manter a integridade dos cenários inclusos no código fonte e os cenários cadastrados no software C&L. Para resolver este problema pretendemos estender o C&L, permitindo que os cenários sejam importados automaticamente do código fonte para dentro do software.

A navegabilidade entre os cenários que compõem o software só é possível quando se utiliza o C&L para visualizá-los. Um de nossos projetos futuros é a criação de um ambiente de programação que desse suporte a utilização de cenários, permitindo a navegação através dos cenários no próprio código fonte.

Pretendemos também fazer a integração entre o LAL do domínio da aplicação e o LAL do espaço de nomes. Esta integração permitirá um rastreamento que possibilitará a identificação dos conceitos do domínio da aplicação que deram origem aos nomes que compõem o espaço de nomes. Um dos benefícios desta integração seria a possibilidade de verificar se os nomes foram escolhidos adequadamente.

Por fim, dois trabalhos importantes que pretendemos realizar futuramente são a utilização do método por outras pessoas e a realização de experimentos comparativos. Com isto esperamos determinar o nível de complexidade do método proposto e se há algum ganho com a sua utilização, comparado com o desenvolvimento sem o uso de nenhum método e com o auxílio de outros métodos.

PUC-Rio - Certificação Digital Nº 0711275/CA