1
Introdução

A transparência é um termo chave que está presente em diversos contextos como o econômico e político, e atualmente um dos novos contextos em que se apresenta é a transparência de software. Sabemos que a comunicação através do software é realizada de maneira opaca, porém torna-se essencial que o software em si seja transparente, para que todos possam ter o conhecimento sobre o que exatamente o software faz [Fsf02]. Neste cenário a Transparência de Software torna-se um tema cada vez mais requisitado, seja pelos indivíduos, pela sociedade ou pelas organizações, já que o direito de ser informado e ter acesso a informação se constitui em um princípio democrático.

No futuro, a transparência vai ser um requisito exigido para toda aplicação, pois este é um conceito que está em processo de institucionalização, cabe portanto que pesquisas sejam realizadas no sentido de atender esta demanda para aqueles que usam o software ou são de alguma maneira afetados por ele [Leite06].

Software livre é um bom exemplo de transparência de software, onde a grande vantagem é a acessibilidade do código, que nos permite escolher, dentre as características do software, as que desejamos [Blank05]. Porém esta possibilidade está direcionada somente aqueles que possuem o conhecimento da programação.

A engenharia de software deve então propor mecanismos para tratar este novo requisito não funcional, a partir da utilização de métodos, técnicas e ferramentas apropriadas. A pesquisa sobre os meios para tornar o software mais transparente deve ser realizada com enfoque no ponto de vista do usuário, onde deve estar claro o que o software está fazendo ou pode fazer.

1.1.
Definições do Problema

O C&L é um software livre que propicia ao seu usuário um ambiente colaborativo que auxilia à edição de cenários e léxicos, descritos em linguagem natural semi-estruturada. Seu desenvolvimento foi realizado ao longo de anos,

de maneira incremental e com a participação de alunos da graduação e da pós-graduação [Silva04]. Este modo de desenvolvimento agregou bastante conhecimento ao grupo de engenharia de requisitos da PUC-Rio [GrupoER09], e a todos que se envolveram direta ou indiretamente no processo. Porém, a falta de uma arquitetura definida, de padronização e pouca documentação fez do C&L um software opaco a nível de código.

O problema de falta de documentação não é restrito ao C&L. Este é um problema comum na comunidade de desenvolvimento de software livre. Isso ocorre devido a informalidade destes projetos e a importância que os usuários finais dão apenas ao fato da instalação do software ser gratuita, sem se preocupar em cobrar dos desenvolvedores algum tipo de documentação. O resultado é que as únicas fontes de informação disponíveis sobre o software são, normalmente, o código fonte, fóruns e listas de e-mail contendo mensagens trocadas entre os integrantes do projeto. Esta falta de documentação acaba sendo um obstáculo a de entrada de novos participantes no desenvolvimento, manutenção e evolução do software. Como a entrada de novos membros é algo fundamental para projetos deste tipo, deve-se fazer um esforço para eliminar ou, se isto não for possível, pelo menos reduzir ao máximo este obstáculo.

1.2. Abordagem Proposta

Neste trabalho exploramos um método de desenvolvimento para software livre baseado no uso de cenários [Leite00]. O resultado da aplicação deste método será um documento único, o código fonte, onde teremos os cenários integrados com o código, facilitando sua leitura e entendimento, trazendo assim mais transparência para o software. Este método foi refinado durante sua aplicação na re-engenharia [Leite96] do software C&L, migrando da plataforma PHP [PHP09] para a plataforma Lua-Kepler [Kepler09] e adotando o *framework* MVC. Para produzir uma documentação complementar aos cenários inclusos no código fonte, utilizamos a técnica LAL [Leite 93] para mapear o espaço de nomes do software C&L.

O conceito de espaço de nomes é muito utilizado em diferentes áreas da ciência da computação. No nosso caso em particular, a nível de código fonte, o espaço de nomes nos fornece um contexto onde nomes são atribuídos a identificares únicos que comporão o código fonte do software (variáveis, funções, objetos) [Leite08].

As duas técnicas utilizadas neste trabalho, cenários e LAL, foram escolhidas porque são objeto de estudo do grupo de engenharia de requisitos da PUC-Rio [GrupoER09] há alguns anos, e trabalhos como [Christoph04], [Silva03] e [Leite08] comprovaram que seu uso neste contexto é promissor.

1.3. Trabalhos relacionados

Em um trabalho precursor sobre o uso de comentários como forma de documentação, [Knuth84] introduz o termo “programação literária”. Este termo identifica um método de desenvolvimento de programas bem documentados, utilizando a idéia de que os programas devem ser considerados uma obra literária. Neste trabalho, Knuth diz que devemos mudar nossa atitude tradicional na construção de programas, segundo ele “em vez de imaginarmos que nossa principal tarefa é instruir o computador sobre o que fazer, devemos nos concentrar em explicar a seres humanos o que queremos que o computador faça”.

Para dar suporte a sua idéia, Knuth desenvolveu um ambiente de programação literária chamado WEB. Neste ambiente os programadores devem programar em WEB, uma linguagem que mistura documentação com implementação, produzindo um único arquivo. Este arquivo será processado posteriormente por dois programas (*weave* e *tangle*), dando origem a dois novos arquivos, um contendo código, que poderá ser executado, e outro descrito em linguagem natural, que servirá de documentação.

Em um trabalho relacionado [Cassino96], Cassino propõe um *framework* para facilitar o desenvolvimento de ferramentas de suporte à programação literária. A motivação para criação deste *framework* veio da popularização da idéia de “programação literária”, que fez surgir um grande número de ferramentas de apóio a métodos semelhantes. O razão principal para o surgimento de tantas ferramentas foi o uso de diferentes linguagens. Cada ferramenta estava atrelada ao uso de uma linguagem específica.

Para demonstrar seu uso, foi desenvolvida uma ferramenta sobre o *framework* proposto. Esta ferramenta é dividida em dois programas. Um que extrai do arquivo fonte o código escrito pelo usuário e gera um arquivo que será posteriormente compilado para gerar o software desejado, e outro que processa o arquivo, adicionando informações e formatações para transformá-lo em um arquivo que possa ser processado por um formatador de textos e assim obter a

documentação final. Todo o processamento realizado pela ferramenta é baseado no uso de marcas pré-estabelecidas no arquivo fonte.

A idéia de uso de marcas no código fonte está por trás da criação do padrão JavaDoc [JavaDoc09]. Este padrão utiliza uma serie de marcadores para realizar a padronização de comentários no código Java e, posteriormente, produzir automaticamente um conjunto de páginas HTML navegáveis contendo as informações comentadas. Esta padronização também pode ser encontrada para as linguagens Lua [LuaDoc] e PHP [PHPDoc].

Em [Staa00] são estabelecidas regras para padronizar a utilização de comentários. Esta padronização visa uniformizar o estilo de programação, facilitando o entendimento, manuseio e evolução de código escrito por terceiros.

No trabalho [Christoph04] é sugerida a adoção da estrutura de cenários [Leite00] para padronizar o uso de comentários no código fonte. Segundo o autor, o uso de cenários ajudaria na organização e uniformidade dos comentários, facilitando a enumeração das funcionalidades e não-funcionalidades do software.

1.4. Organização do Documento

No Capítulo 2 detalharemos as duas técnicas de representação de conhecimento utilizadas neste trabalho, o LAL e cenários.

No Capítulo 3 apresentaremos a versão anterior do software C&L, a forma como foi desenvolvida, sua arquitetura e uma descrição detalhada de seu funcionamento.

No Capítulo 4 explicaremos o método baseado no uso de cenários e nossa proposta de mapeamento do espaço de nomes através do LAL.

No Capítulo 5 detalharemos passo a passo a aplicação do método proposto na re-arquitetura do software C&L.

No Capítulo 6 faremos uma comparação do trabalho realizado com os trabalhos relacionados, apresentaremos algumas conclusões e listaremos alguns trabalhos que precisam ser estudados.