UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

GERADOR DE DOCUMENTAÇÃO PARA LINGUAGEM C, UTILIZANDO TEMPLATES

VILMAR ORSI

BLUMENAU
2006

VILMAR ORSI

GERADOR DE DOCUMENTAÇÃO PARA LINGUAGEM C, UTILIZANDO TEMPLATES

Proposta de Trabalho de Conclusão de Curso submetida à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso I do curso de Ciências da Computação — Bacharelado.

Profª. Joyce Martins - Orientadora

BLUMENAU
2006

1 INTRODUÇÃO

A evolução da indústria de software tem acompanhado de perto, senão de forma mais acelerada, a evolução da indústria tecnológica, auxiliando cada vez mais áreas no desenvolvimento de suas atividades. Como tudo que se produz, software também precisa de manutenção e adaptação.

Para reduzir o tempo gasto em manutenção e adaptação, nada melhor que um software bem construído e documentado de forma correta. Neste ponto surge um problema: considerável parte dos programas[1] escritos não é documentada corretamente em função de orçamento, prazos, entre outros fatores.

Áece (2003) afirma que a criação de documentação é uma fase muito importante no desenvolvimento de um software e que esta fase é ignorada por muitos programadores, o que vem a dificultar bastante quando o projeto é utilizado por outros. Sant'Anna (2001) acrescenta que a maioria dos programas possui pouca documentação que em geral não reflete a realidade das atualizações feitas. Além do mais, os ambientes de programação não facilitam a geração de documentação.

Tentando resolver parte deste problema, surge a idéia de desenvolver uma ferramenta que automatize a documentação de software, a partir da análise dos códigos fonte dos programas, assim como dos comentários encontrados nos mesmos. Comentários bem elaborados, com marcações indicando autoria, funcionalidades, entre outras características, podem ser convertidos em documentos que além de auxiliarem na reutilização de código, servem como ponto de apoio quando se fizer necessária manutenção ou adaptação de alguma das funcionalidades do software.

Algumas linguagens de programação, como Java e C# por exemplo, já contam com ferramentas com características semelhantes. Sendo assim, optou-se por desenvolver um gerador de documentação para a linguagem C, com o intuito de aplicação no meio acadêmico, objetivando desenvolver nos "potenciais programadores" o hábito de documentarem seus códigos fonte.

A escolha do C se dá pelo fato de a referida linguagem não possuir um gerador de documentação acessível ao meio acadêmico, e por ser esta a linguagem de programação utilizada na disciplina de Programação I do curso de Ciências da Computação da

[1] *Software* e programa são utilizados no decorrer do texto como sinônimos.

Universidade Regional de Blumenau (FURB), instituição onde se pretende viabilizar o uso do gerador de documentação pelos acadêmicos.

A documentação será gerada a partir dos códigos fonte de programas desenvolvidos na linguagem C. Tais códigos serão analisados léxica e sintaticamente, tendo informações extraídas tanto das linhas de código, quanto dos comentários dispostos nos mesmos. Para os códigos fonte não comentados serão gerados documentos simples e para os comentados, documentos complexos. As informações, após ordenadas e modeladas, de acordo com o *template*[2] desenvolvido e/ou selecionado pelo usuário, serão apresentadas através de um conjunto de arquivos no formato *HyperText Markup Language* (HTML), compondo a documentação do software.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta para a geração automática de documentação para programas escritos em C.

Os objetivos específicos do trabalho são:

a) disponibilizar analisadores léxico e sintático para a análise e extração de informações do código fonte;
b) definir um conjunto de marcadores especiais com significados particulares como autor e data, que serão utilizados quando da descrição dos comentários no código fonte dos programas;
c) gerar a documentação de programas no formato HTML;
d) utilizar *templates* para a formatação da documentação gerada.

1.2 RELEVÂNCIA DO TRABALHO

> Ferramentas provendo facilidades de documentação permitem agilizar o desenvolvimento de software, realizando algumas tarefas automaticamente, e

[2] Rocha (2005) define *templates* como "[...] arquivos que contém [sic] variáveis e blocos especiais que serão dinamicamente transformados a partir de dados enviados pelo Motor de Templates gerando as interfaces das aplicações [...]".

> permitem que modificações sejam mais facilmente realizadas, simplificando a tarefa de manutenção. [...] A maioria das ferramentas não oferece um bom suporte à documentação, de forma que o desenvolvedor se vê, muitas vezes, usando um processador de texto para documentar atividades do processo de software. (NUNES; SOARES; FALBO, 2004, p. 1).

Nunes, Soares e Falbo (2004, p. 1) afirmam que, a documentação, embora seja um aspecto essencial para a qualidade do software, é negligenciada em decorrência de fatores como falta de uma política organizacional e de ferramentas para apoiar a documentação. Visando tentar resolver parte deste problema, o presente trabalho objetiva oferecer uma ferramenta *freeware* para a geração automática de documentação de programas desenvolvidos na linguagem C.

A possibilidade de gerar documentação automaticamente, através do código fonte dos programas implementados, possibilita uma economia de custos e tempo, sendo que basta realizar comentários concisos durante o desenvolvimento do programa para poder gerar uma documentação também concisa. Outra questão importante está associada a manutenção de software, uma vez que se os comentários forem alterados, refletindo a realidade atual do código fonte, a nova documentação pode ser gerada garantindo a coerência entre programa e documentação.

Por fim, o uso de *templates* dispõe ao usuário a possibilidade de formatar a documentação da maneira que melhor lhe convir, podendo criar seus próprios padrões de documentos e tornando o processo de geração menos rígido.

1.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar levantamento da bibliografia sobre documentação de software, geração automática de documentação de software, analisadores de linguagens (léxico e sintático), *templates* e motores de *templates*;

b) elicitação de requisitos: detalhar e reavaliar os requisitos, observando as necessidades levantadas durante a revisão bibliográfica e com o professor Maurício Capobianco Lopes, professor da disciplina de Programação I do curso de Ciências da Computação;

c) especificação de marcadores: especificar com definição regular o conjunto de marcadores especiais, no estilo dos marcadores do JavaDoc, a serem utilizados nos comentários para delimitar características como autoria, data, versão, etc., para posterior geração da documentação;

d) especificação da análise dos códigos fonte C: buscar, estudar e entender a estrutura da *Backus-Naur Form* (BNF) do C, adaptando-a para incluir a especificação dos marcadores especiais definidos na etapa anterior, para que possa ser utilizada na implementação dos analisadores léxico e sintático;

e) definição do motor *templates*: buscar um motor de *templates* que possa ser utilizado com a linguagem Object Pascal;

f) especificação da ferramenta: especificar a ferramenta com análise orientada a objeto utilizando a *Unified Modeling Language* (UML). Será usada a ferramenta Enterprise Architect para o desenvolvimento dos diagramas de caso de uso, de classes e de seqüência;

g) implementação: implementar a ferramenta proposta, utilizando o ambiente de programação Delphi 7.0, o Gerador de Analisadores Léxicos e Sintáticos (GALS) e o motor de *templates* definido na etapa (e);

h) testes: apresentar versões parciais da ferramenta aos acadêmicos da disciplina de Programação I do curso de Ciências da Computação, para serem utilizadas na geração da documentação dos programas desenvolvidos nas atividades da disciplina. Serão aplicados questionários junto aos acadêmicos e ao professor da referida disciplina para fins de avaliação da ferramenta proposta. Os resultados da aplicação da ferramenta e dos questionários poderão implicar em possíveis alterações na especificação da mesma.

As etapas serão realizadas nos períodos relacionados no Quadro 1.

	2006													
	maio		jun.		jul.		ago.		set.		out.		nov.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico	■	■	■											
elicitação dos requisitos		■	■											
especificação de marcadores			■	■										
especificação da análise dos códigos fonte C				■	■									
definição do motor de *templates*					■	■								
especificação da ferramenta					■	■	■							
implementação						■	■	■	■	■	■			
testes							■	■	■	■	■	■		

Quadro 1 - Cronograma

2 REVISÃO BIBLIOGRÁFICA

No presente capítulo são apresentados alguns aspectos teóricos relacionados ao trabalho. Na seção 2.1 são evidenciadas a importância da documentação de software e da geração automática de documentação de software. Na seção 2.2 são apresentadas informações sobre entrada e saída de dados do gerador de documentação, analisadores de linguagens (léxico/sintático) e motores de *templates*. Na seção 2.3 são descritos três trabalhos correlatos.

2.1 DOCUMENTAÇÃO DE SOFTWARE

Pfleeger (2004, p. 370) associa a qualidade da documentação ao sucesso do software. A documentação de software, além de se tornar um hábito no dia-a-dia dos programadores, deve ser facilitada através do desenvolvimento de ferramentas que automatizem este processo. Melhor ainda se a documentação do programa puder ser gerada de forma *in-line*[3].

De acordo com Herrington (2003, p. 7), códigos gerados de forma automatizada são muito mais consistentes do que os escritos a "mão", e têm todas as funcionalidades que se possa querer. A afirmação do autor é também aplicável para a geração de documentação, uma vez que a mesma, quando automatizada passa a seguir padrões.

2.2 ENTRADA E SAÍDA DE DADOS DO GERADOR DE DOCUMENTAÇÃO

No que tange a automatização da geração de documentação, a entrada dos dados para o gerador pode ser feita através da análise léxica e sintática do código fonte dos programas. Aho, Sethi e Ullman (1995, p. 38) apresentam como tarefa principal do analisador léxico, a leitura dos caracteres de entrada e produção de uma seqüência de símbolos básicos (*tokens*) que são utilizados pelo *parser* na análise sintática, além de tarefas secundárias como remover espaços em branco e comentários. A respeito de analisador sintático, Aho, Sethi e Ullman

[3] O programador escreve a documentação, enquanto escreve o código fonte do programa.

(1995, p. 72) afirmam que o mesmo tem a função de receber a cadeia de *tokens* enviada pelo analisador léxico, e conferir se tal cadeia pode ser gerada pela gramática da linguagem-fonte.

A saída do gerador, ou a documentação propriamente dita, pode ser através de arquivos texto, sendo que para os geradores de documentação analisados e descritos na próxima seção, o formato adotado é o HTML, que facilita a publicação. A saída pode ser gerada diretamente ou construída a partir de um conjunto de *templates*, escritos em uma linguagem específica.

> Esta linguagem irá definir os tipos de blocos especiais e como referenciar variáveis nos templates. Estas linguagens podem variar bastante em nível de complexidade, a depender do motor de templates - podem ser definidas apenas com estruturas básicas para substituição de valores de variáveis ou até ter estruturas de controle mais complexas como loops e comandos condicionais. (ROCHA, 2005).

O uso de *templates* possibilita que o programador personalize a documentação de seus programas.

2.3 TRABALHOS CORRELATOS

Algumas ferramentas desempenham papel semelhante ao proposto no presente trabalho, cada qual com as suas peculiaridades e para determinadas linguagens. Dentre elas foram selecionadas: JavaDoc, o gerador de documentação do C# e o CDoc.

2.3.1 JavaDoc

Através do uso do JavaDoc pode-se extrair os comentários do código fonte Java e criar uma documentação no formato HTML. Segundo Flanagan (2000, p. 22), a linguagem Java suporta três tipos de comentários, sendo eles: de linha, de várias linhas (ou de bloco) e especiais para a geração de documentação. Os comentários para a geração de documentação iniciam com os caracteres `/**` e finalizam com `*/` e são chamados de *doc comment*.

É importante destacar que os comentários para geração de documentação podem incluir, além dos próprios comentários, *tags* HTML simples, tais como `<I>`, `<CODE>`, entre outros, e *tags* de *doc comment* definidos pelo JavaDoc, como por exemplo `@author`, `@param`, etc., que delimitam informações adicionais. Os *doc comments* devem aparecer imediatamente

antes de uma definição de classe, interface, método ou campo (FLANAGAN, 2000, p. 214).

Segundo Paulo e Graça (2003), com o JavaDoc também é possível gerar documentação para pacotes de classes, sendo que os comentários para os mesmos devem ser escritos em um arquivo chamado `package.html`, que deve ser salvo no mesmo diretório onde forem salvos os arquivos com extensão Java, contendo o código fonte das classes do pacote que se deseja documentar. Quando for gerada a documentação, as informações contidas no arquivo `package.html` serão extraídas e adicionadas à primeira página da documentação HTML produzida para o pacote.

2.3.2 Gerador de documentação do C#

Segundo Haddad (2001, p. 375), a Microsoft desenvolveu uma ferramenta para permitir documentação automática de programas C#, seguindo o padrão *eXtensible Markup Language* (XML). Basta que sejam seguidos os padrões dos marcadores XML nos comentários do código fonte, e a documentação do mesmo poderá ser gerada. Haddad (2001, p. 375) alerta que simplesmente será gerada a documentação do código fonte, de acordo com os comentários apresentados, e não uma documentação do fluxo das informações no aplicativo.

Câmara (2003) explica que “enquanto os comentários em C# são iniciados com ‘//’, os comentários iniciados com ‘///’ indicam documentação que pode ser extraída pelo compilador e manipulada”. Sendo assim, sempre que o compilador encontrar os caracteres `///` considera tratar-se de um comentário a ser utilizado na geração da documentação e realiza a manipulação das informações contidas no comentário, de acordo com o elemento XML apresentado no mesmo.

De acordo com Galuppo, Matheus e Santos (2004, p. 467), os *tags*, ou elementos XML, utilizados quando da realização dos comentários para a geração de documentação, são padronizados e formam um conjunto de dezoito pares. Observa-se a necessidade de delimitar as informações com *tags* de início e fim, como por exemplo `<author>` Galuppo `</author>`.

2.3.3 CDoc

Segundo a Software BlackSmiths (2001), o CDoc pode gerar documentação para códigos fonte de programas desenvolvidos nas linguagens C, C++ e Java. Além da documentação, o CDoc pode também gerar diagramas com a árvore de chamadas de funções, árvore hierárquica de classes, cálculo ciclomático de complexidade, entre outros.

Assim como o JavaDoc e o gerador de documentação do C#, descritos nas seções anteriores, o CDoc necessita de comentários especiais para a geração de documentação complexa a partir do código fonte de programas, diferindo-se por não fazer uso de *tags* e/ou marcadores especiais. Em se tratando de programas desenvolvidos na linguagem C, Milvang (2004) apresenta que os comentários no código fonte devem ser delimitados pelos caracteres `/*` indicando o início do comentário e `*/` indicando o fim do mesmo.

3 REQUISITOS DO SISTEMA A SER DESENVOLVIDO

A ferramenta deverá:

a) gerar documentação, no formato HTML, de programas desenvolvidos na linguagem C, a partir do código fonte dos mesmos (requisito funcional – RF);
b) realizar análises léxica e sintática do código fonte de programas escritos na linguagem C, localizando informações para o documento a ser gerado (RF);
c) permitir o uso de *tags* HTML para formatação de texto nos comentários (RF);
d) usar *templates* para a modelagem do documento a ser gerado (RF);
e) disponibilizar uma interface para permitir a elaboração dos *templates* (RF);
f) disponibilizar um conjunto de marcadores especiais para serem utilizados nos comentários do código fonte (RF);
g) utilizar marcadores especiais no estilo dos marcadores definidos no JavaDoc (RNF);
h) ser implementado utilizando o ambiente Delphi 7.0 (requisito não-funcional – RNF);
i) ser compatível com o sistema operacional Windows 98, 2000 e XP (RNF).

4 CONSIDERAÇÕES FINAIS

Após a análise da problemática envolvendo a questão documentação de software, sugere-se o desenvolvimento de uma ferramenta para a geração automática de documentação a partir do código fonte de programas escritos em C. O desenvolvimento da referida ferramenta será voltado à geração de documentos de qualidade, que possam auxiliar no processo de implementação, manutenção e adaptação de software.

O gerador de documentação proposto fará uso de marcações especiais, dentro dos comentários de blocos e irá gerar a documentação no formato HTML, com as informações dispostas, conforme estabelecido pelo usuário quando da elaboração do *template* para os referidos documentos.

Para códigos fonte não comentados será possível gerar documentação com características básicas dos mesmos, tais como declaração de variáveis, cabeçalho de procedimentos e de funções. Quando da utilização de comentários, haverá a possibilidade da geração de documentos mais complexos a partir dos referidos marcadores especiais, que serão usados para delimitar informações como autor, data ou versão do código fonte.

As informações necessárias para gerar documentação serão extraídas dos códigos fonte escritos em C através de analisadores léxico e sintático. Através da análise léxica, serão coletadas informações dos comentários acrescidos ao código fonte, as quais serão armazenadas em uma estrutura de dados. O analisador sintático irá avaliar as cadeias de *tokens* enviadas pelo analisador léxico, buscando e armazenando informações sobre aquelas partes do código relevantes para a documentação que será gerada.

A documentação gerada poderá ser padronizada pelos usuários, através dos *templates*, com definições de como será modelado o documento. Os *templates* serão utilizados para formatar as informações agrupadas durante as análises léxica e sintática. A utilização dos *templates* está vinculada à liberdade concedida ao usuário de selecionar quais informações e em que ordem devem ser apresentadas na documentação.

Em relação aos trabalhos correlatos apresentados, pode-se afirmar que a ferramenta proposta irá possuir várias características encontradas nas ferramentas citadas. No que tange a utilização dos comentários de bloco quando da geração da documentação, a sintaxe será semelhante a dos demais trabalhos apresentados, uma vez que será acrescentado um caracter, ou um conjunto deles, aos caracteres que definem tal comentário de bloco, transformando-o em um comentário de documentação.

Já os marcadores especiais serão semelhantes aos utilizados no JavaDoc. Será permitido também, como no JavaDoc, o uso de *tags* HTML em comentários de documentação.

Observa-se que a ferramenta proposta irá diferir do gerador de documentação do C# no que se refere à linguagem para a qual será aplicado, e do CDoc no que tange à licença de utilização, uma vez que será um gerador de documentação livre para programas desenvolvidos em C.

REFERÊNCIAS BIBLIOGRÁFICAS

ÁECE, I. **Criação de comentários e documentação em VB.NET**. [S.l.], 2003. Disponível em: <http://www.csharpbr.com.br/mostra_artigo.asp?id=0025>. Acesso em: 26 fev. 2006.

AHO, A. V.; SETHI, R.; ULLMAN, J. D. **Compiladores:** princípios, técnicas e ferramentas. Tradução Daniel de Ariosto Pinto. Rio de Janeiro: LTC, 1995.

CÂMARA, F. **O processo de compilação no C#**. [São Paulo], 2003. Disponível em: <http://www.linhadecodigo.com.br/artigos.asp?id_ac=135&pag=4>. Acesso em: 26 fev. 2006.

FLANAGAN, D. **Java:** o guia essencial. 3. ed. Tradução Kátia Roque. Rio de Janeiro: Campus, 2000.

GALUPPO, F.; MATHEUS, V.; SANTOS, W. **Desenvolvendo com C#**. Porto Alegre: Bookman, 2004.

HADDAD, R. I. **C#:** aplicações e soluções. São Paulo: Érica, 2001.

HERRINGTON, J. **Code generation in action**. Greenwich: Manning, 2003.

MILVANG, O. **CDoc**. Oslo, 2004. Disponível em: <http://www.ifi.uio.no/forskning/grupper/dsb/Software/Xite/ReferenceManual/cdoc_1.html>. Acesso em: 13 mar. 2006.

NUNES, V. B.; SOARES, A. O.; FALBO, R. A. Apoio à documentação em um ambiente de desenvolvimento de software. In: WORKSHOP IBEROAMERICANO DE INGENIERA DE REQUISITOS Y DESARROLLO DE AMBIENTES DE SOFTWARE, 7., 2004, Arequipa. **Anais...** Arequipa: IDEAS, 2004. Não paginado. Disponível em: <http://www.inf.ufes.br/~falbo/download/pub/2004-IDEAS-1.pdf>. Acesso em: 05 mar. 2006.

PAULO, J. L. e GRAÇA, J. **Uma introdução ao JavaDoc**. Lisboa, 2003. Disponível em: <http://po.tagus.ist.utl.pt/2005/apoio/Introjavadoc.html>. Acesso em: 12 mar. 2006.

PFLEEGER, S. L. **Engenharia de software**: teoria e prática. Tradução Dino Franklin. São Paulo: Prentice Hall, 2004.

ROCHA, L. **APE:** plataforma para o desenvolvimento de aplicações web com PHP. [Salvador], 2005. Disponível em: <http://twiki.im.ufba.br/bin/view/Aside/ProjetoConclusaoDeCursoAPEMonografia#4_2_Motores_de_templates>. Acesso em: 15 mar. 2006.

SANT´ANNA, M. **Documentação em programas C#**. São Paulo, 2001. Disponível em: <http://www.mas.com.br/Artigos/Doc_CSharp.htm>. Acesso em: 26 fev. 2006.

SOFTWARE BLACKSMITHS. **Welcome to Software Blacksmiths**: CDoc. Mississauga, [2001?]. Disponível em: <http://www.swbs.com/>. Acesso em: 13 mar. 2006.