Framedoc: Um *Framework* para a Documentação de Componentes Reutilizáveis

Leonardo Gresta Paulino Murta
Márcio de Oliveira Barros
Cláudia Maria Lima Werner
{murta, marcio, werner}@cos.ufrj.br
COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511 – CEP. 21945-970
Rio de Janeiro – Brasil

Abstract

This paper presents a framework that supports the creation and visualization of documents describing reusable components. Such documents are built as hypertexts, using hypermedia and pattern technologies. The proposed approach provides the necessary tools for component packaging and understanding. It's possible to generate a web site with a HTML representation of the component documentation. This site can be used for finding and selecting components to be reused. We present the elements composing the proposed framework and its instantiation in a software development infrastructure.

Palavras-chave: Documentação de Componentes, Reutilização de Software, Padrões de Documentação, Hipermídia.

1 Introdução

A reutilização de software pode ser abordada sob duas perspectivas complementares (MOORE et al., 1991). A primeira, conhecida como *desenvolvimento para reutilização*, se baseia em identificar os componentes que devem ser criados, criar os componentes identificados e empacotar esses componentes, disponibilizando-os em uma base de dados. A segunda perspectiva, conhecida como *desenvolvimento com reutilização*, se baseia em recuperar um conjunto de componentes da base de dados, selecionar, a partir deste conjunto, o componente mais adequado e modificá-lo, integrando-o ao projeto em desenvolvimento. No contexto abordado, um componente pode ser visto como um produto de software em qualquer nível de abstração, como, por exemplo, classes, casos de uso, diagramas, especificações, entre outros. A Figura 1 exibe a interação entre as duas perspectivas de reutilização de software.

Para que a reutilização se torne efetiva, se faz necessária a disponibilização de ferramentas que suportem as atividades exibidas na Figura 1 (TRACZ, 1994). Além disso, as atividades de empacotamento, recuperação e compreensão podem exigir que mais informações sejam incorporadas ao componente reutilizável. Isto cria a necessidade de uma ferramenta de documentação, que controle a criação e disponibilização de documentos que contenham essas informações adicionais sobre os componentes. Este suporte, fornecido na atividade de empacotamento, facilitará a realização das atividades de recuperação e compreensão.

Nos processos de desenvolvimento de software, a documentação não faz parte de uma fase definida, mas ocorre ao longo de sua existência, em paralelo com as demais fases do ciclo de vida. A documentação dos artefatos de software pode ser estruturada de forma linear ou associativa.

A documentação através de textos lineares é, como o próprio nome diz, seqüencial. Os documentos são desenvolvidos para serem lidos em uma ordem predeterminada e inalterável. O texto linear força ao leitor um nível de abstração específico, não dando a ele a opção de saber mais sobre um determinado tema a qualquer momento. Este tipo de documentação tem como característica a disjunção entre os vários documentos gerados, ou a geração de um único documento monolítico.

O suporte oferecido por uma abordagem de documentação associativa, ou seja, baseada em hipertextos, não se prende à linearidade dos documentos, permitindo a criação e navegação através de referências entre os documentos. Utilizando a abordagem associativa, qualquer documento descrevendo um componente de software pode ser considerado um nó de um hipertexto, contendo referências para documentos de outras fases do processo de desenvolvimento do componente ou para documentos de componentes relacionados. Por exemplo, a documentação do componente "*Especificação do projeto ABC*" pode conter uma ligação para a documentação do componente *"Diagrama de classes do framework colégio"*, que, por sua vez, pode conter uma ligação para a documentação do componente *"Código em Java da classe Aluno"*. O hipertexto forma uma rede, conectando os componentes relacionados dentro do projeto.

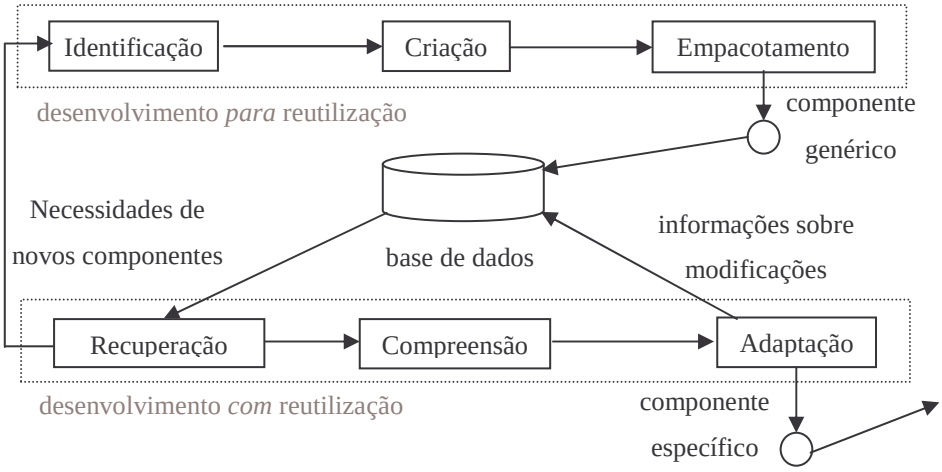


Figura 1: Processo de Reutilização.

O pensamento humano não trabalha de forma seqüencial. Os mecanismos relacionados à manipulação de informações estão diretamente associados à capacidade do pensamento humano. Assim, estes mecanismos devem operar de modo similar ao próprio pensamento. Esse modo é o associativo, que é a forma de trabalho dos hipertextos (W3C, 2001). Podemos ressaltar algumas vantagens dos hipertextos em relação aos textos lineares, entre elas:

- Proporcionar conectividade entre as informações;
- Oferecer uma interface compatível com o modo de raciocínio humano;
- Permitir ao usuário a determinação do nível de abstração desejado.

Devido às etapas do processo de desenvolvimento de software serem altamente relacionadas entre si, os textos lineares apresentam uma outra deficiência quando aplicados na documentação destas etapas: a incapacidade de expressar a fronteira deste inter-relacionamento. Por exemplo:

- Utilizando um único texto linear como a documentação de todo o processo, não teremos uma visão individualizada de cada etapa, gerando dificuldade de localização da informação desejada;
- Utilizando vários textos lineares com a documentação de cada etapa, não teremos uma visão global do processo, gerando dificuldade de relacionar informações pseudodistintas.

O objetivo deste trabalho é fornecer um *framework* para o suporte à documentação de componentes reutilizáveis, utilizando as tecnologias de hipermídia e padrões, apoiando a execução das etapas de empacotamento, recuperação e compreensão do processo de reutilização descrito na Figura 1. Um *framework* pode ser considerado como um projeto ou arquitetura de alto nível, consistindo de classes que são especialmente projetadas para serem refinadas e usadas em grupo (WIRFS-BROCK et al., 1990). O **FrameDoc**, nome adotado para o *framework* proposto, foi utilizado no contexto da infraestrutura Odyssey (WERNER et al., 2000), um conjunto de ferramentas que fornece suporte à reutilização de software orientado a técnicas de engenharia de domínios.

No contexto do Odyssey, o *desenvolvimento para reutilização*, descrito na Figura 1, equivale à Engenharia de Domínio, que tem como objetivo construir componentes reutilizáveis que solucionem problemas de domínios de conhecimento específicos (BRAGA et al., 1999). Esses componentes têm que ser genéricos o bastante para que possam ser utilizados em um conjunto de aplicações desenvolvidas para o domínio em questão. Também nesse contexto, o *desenvolvimento com reutilização* equivale à Engenharia da Aplicação, que tem como objetivo construir aplicações em um determinado domínio, utilizando os componentes genéricos já existentes (MILER, 2000) .O engenheiro de software responsável por executar as atividades de Engenharia de Domínio é conhecido como engenheiro de domínio, enquanto o engenheiro de software responsável pelas atividades de Engenharia de Aplicação é conhecido como engenheiro de aplicação.

Este trabalho está organizado em 4 seções. Esta seção exibe a motivação, o objetivo e a organização deste trabalho. A seção 2 exibe alguns trabalhos relacionados. A seção 3 descreve a abordagem proposta. Finalmente, a seção 4 descreve as contribuições, limitações e trabalhos futuros.

2 Trabalhos Relacionados

Existe uma grande variedade de abordagens que apóiam a documentação de componentes de software em diversos níveis de abstração. Para componentes de código, podemos citar o **JavaDoc**, e para componentes de análise e projeto, o suporte à documentação oferecido pelas ferramentas **Rational Rose** e **Fast Case**, além do ambiente **Memphis**. Foram identificados os seguintes critérios, que devem ser providos por abordagens de documentação de componentes de software:

- **Exportação da documentação em formato portável:** a ferramenta de suporte à documentação de componentes reutilizáveis deve salvar os documentos gerados em um formato aberto, que possa ser visualizado por programas de licença gratuita, como, por exemplo, HTML, Postscript, RTF, PDF, entre outros;
- **Utilização de *hyperlinks* na documentação:** a ferramenta de suporte à documentação deve permitir a criação de referências (*hyperlinks*) entre componentes, entre partes de componentes e para *sites* na Internet;
- **Criação e configuração de padrões de documentação:** a ferramenta deve permitir a criação de *templates*, ou formatos predefinidos e configuráveis de documentação, para que esta ocorra de forma padronizada. A documentação padronizada exige que o desenvolvedor forneça informações previamente definidas para cada tipo de componente;
- **Geração de documentação modular:** a ferramenta deve permitir a geração de documentos separados para componentes distintos, evitando o acúmulo de informações em documentos monolíticos;
- **Utilização de multimídia na documentação:** a ferramenta deve permitir que sejam utilizados recursos de imagem, som e vídeo para documentar os componentes;
- **Visualização da documentação antes da geração:** a ferramenta deve permitir que a documentação seja visualizada antes da sua geração, com uma formatação similar à que será gerada;
- **Geração da documentação implícita ao componente:** a ferramenta deve permitir a exibição de características inerentes ao próprio componente na sua documentação, como, por exemplo, os atributos e métodos de uma classe ou pré e pós condições de um caso de uso.

Esses critérios não formam um conjunto completo, mas servem como base para avaliação das abordagens de documentação apresentadas na literatura. Utilizando esses critérios, foi construída uma tabela de comparação, apresentada na seção 4, exibindo como as abordagens descritas a seguir e a abordagem proposta cumprem os critérios.

2.1 JavaDoc

A linguagem Java (SUN MICROSYSTEMS, 2001a) permite um estilo especial de comentário de código, além dos habituais herdados da linguagem C++. Este estilo é utilizado pelo padrão de documentação JavaDoc (SUN MICROSYSTEMS, 2001b), sendo identificado por "/**" na primeira linha, um "*" no início de cada linha seguinte e por "*/" ao final. O padrão JavaDoc tem como objetivo descrever cada classe de um programa Java, juntamente com seus métodos e atributos. Essa descrição se realiza através de tags, apresentadas no interior do bloco especial de comentário. Uma classe documentada através de JavaDoc pode ser submetida a um interpretador, que irá gerar uma versão HTML da documentação. A Figura 2 exibe um exemplo de comentário JavaDoc em um método e a página HTML gerada a partir desta documentação. Dentre as tags definidas no padrão JavaDoc, podemos destacar:

- **author:** autor (criador) de uma classe, atributo ou método;
- **version:** versão atual de uma classe, atributo ou método;
- **param:** semântica dos argumentos de um método;
- **return:** semântica do retorno de um método;
- **deprecated:** indica a partir de qual versão um método está obsoleto e qual método deve ser usado em seu lugar;
- **exception** (ou **throws**, sinônimo adicionado no Javadoc 1.2): descreve as exceções que um método pode disparar.

O JavaDoc tem como pontos positivos a exportação de documentação em formato portável, a utilização de referências entre os documentos, a geração modular de documentos e a geração de documentação a partir das informações implícitas à classe Java. Entretanto, o JavaDoc não permite a criação de padrões personalizados para a documentação, a utilização e de multimídia e a visualização da documentação em formato similar a geração final.

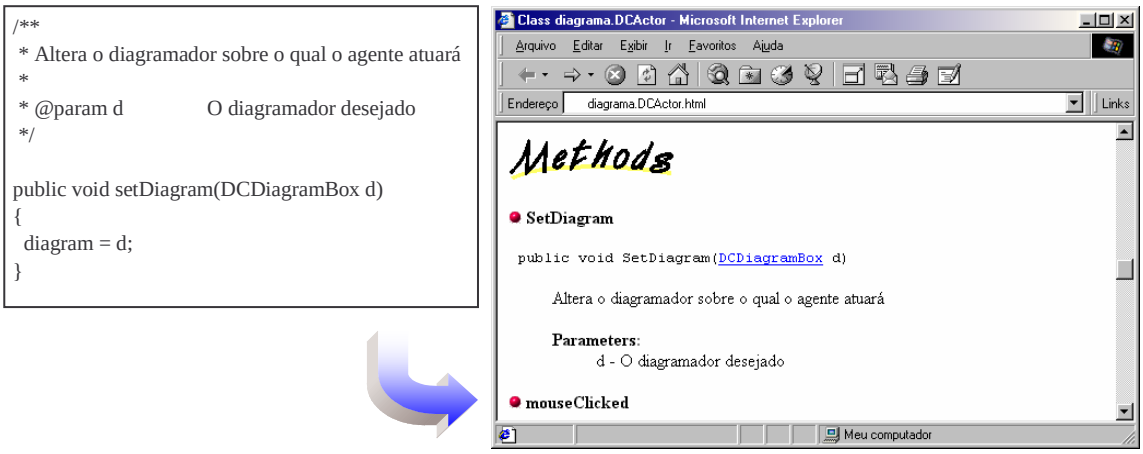


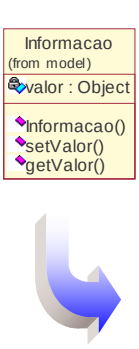
Figura 2: Documentação utilizando o JavaDoc.

2.2 Rational Rose

O Rational Rose (RATIONAL, 2001) é uma das ferramentas CASE que suportam a orientação a objetos mais utilizadas atualmente em projetos comerciais e acadêmicos. A documentação na ferramenta (não estamos considerando o uso de extensões para o Rose) é formada pelos dados contidos no componente, como os atributos e métodos de uma classe, e por um campo textual que deve ser preenchido com a descrição do componente. Por exemplo, uma classe tem como documentação o seu nome, um texto de descrição, seus atributos e métodos, com os respectivos textos de descrição.

A visualização da documentação pode ser feita no próprio Rose, pela janela de propriedades do componente, ou no Microsoft Word, através da opção de exportação de documentação, que utiliza o padrão OLE de interoperabilidade entre as ferramentas. Esta geração cria um documento monolítico no formato DOC (proprietário do Word).

Uma vantagem desta abordagem é a portabilidade, pois apesar de não ser um padrão aberto, o DOC é utilizado amplamente. Entretanto, existem desvantagens, como a demora na geração da documentação, devido à baixa performance do padrão OLE, e a criação de um único módulo de documentação para todo o projeto, não fornecendo conectividade entre as informações geradas. A Figura 3 exibe uma classe modelada no Rose e a documentação gerada para essa classe.



Selected Logical View Report
model
Informação
Classe model que armazena uma informação de um campo de um documento.
Private Attributes:
valor : Object
Valor da informação

Public Operations:
Informação () : Informação
Constrói uma instância da informação
setValor (valor : Object) :
Atribui um novo valor à informação
getValor () : Object
Pega o valor atual da informação

Figura 3: Documentação utilizando o Rational Rose.

2.3 Fast Case

O Fast Case (SILVEIRA, 1999) é uma ferramenta CASE orientada a objetos que utiliza um subconjunto da notação UML e tem como objetivo dar suporte para criação rápida de sistemas de pequeno e médio porte.

O suporte de documentação de componentes no Fast Case é semelhante ao do Rose, diferenciando no padrão adotado para a geração da documentação, que é HTML. Com essa abordagem, foram sanadas as deficiências observadas no Rose, pois a geração é feita diretamente em arquivo, sem a utilização de OLE, e a documentação de cada componente é armazenada em um arquivo individual.

Uma característica do Fast Case não existente no Rose é a possibilidade de visualizar a documentação em seu formato final antes de gerá-la. As desvantagens do Fast Case são a falta de suporte para a definição de padrões de documentação e a falta de suporte para o uso de *hyperlinks* e multimídia. A Figura 4 exibe o suporte de documentação provido pelo Fast Case.

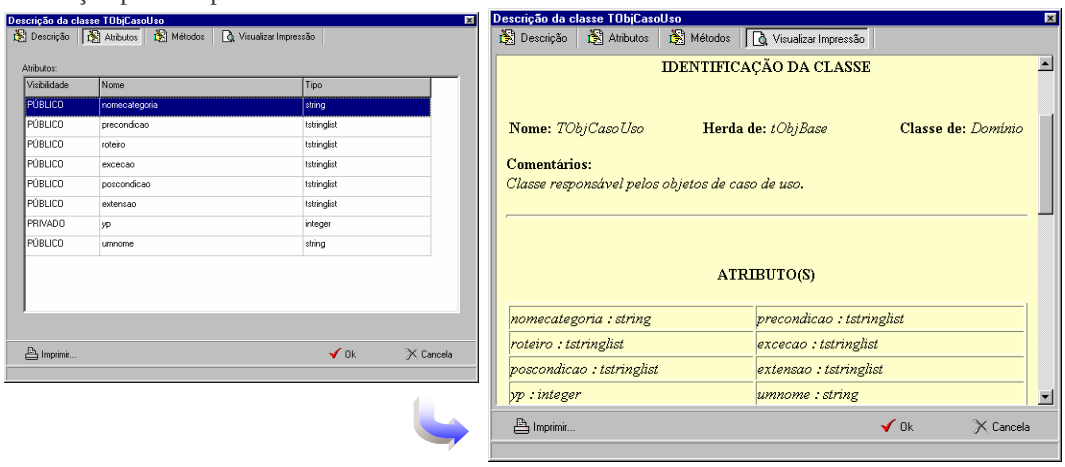


Figura 4: Documentação utilizando o Fast Case.

2.4 Memphis

O ambiente Memphis (WERNER et al., 1996) é um ambiente de desenvolvimento de software baseado em reutilização. O suporte à documentação oferecido pelo ambiente consiste na geração de padrões de documentação na forma de *templates*, que são utilizados no empacotamento dos componentes reutilizáveis (SILVA, 1998). Os padrões de documentação definem quais as informações relevantes para a documentação de um tipo de componente reutilizável, exigindo que estas informações estejam presentes na documentação destes componentes. Um *template* é definido como um conjunto de itens, cada qual armazenando um tipo de informação. A ferramenta de suporte à documentação do ambiente oferece diversos tipos de itens, como texto, diagramas, referências a um hiperdocumento, listas, sons e vídeos. A Figura 5 exibe a documentação de um componente através do Memphis.

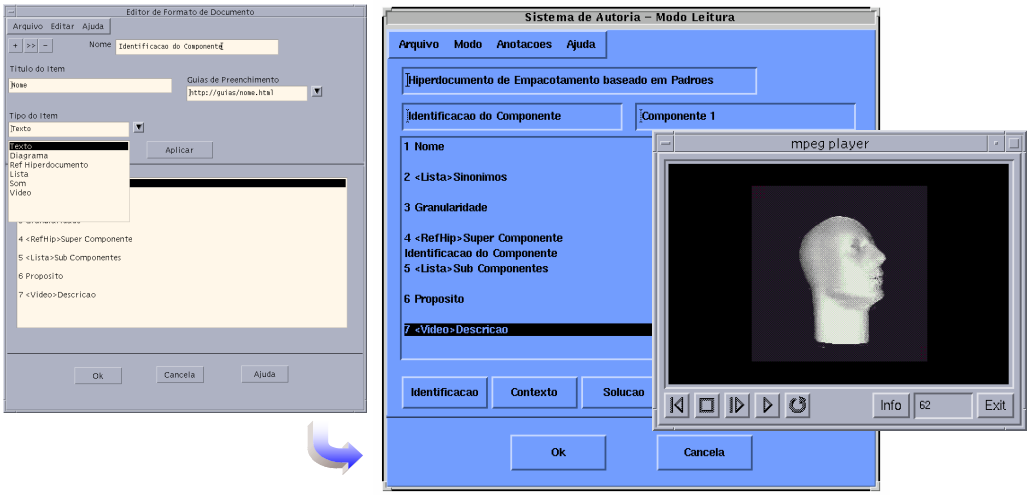


Figura 5: Documentação utilizando o Memphis.

Tanto a geração do *template* quanto o preenchimento e a visualização dos documentos são realizados dentro do próprio ambiente Memphis. A exportação da documentação em um formato que possa ser visualizado fora do ambiente Memphis foi planejada, porém não está implementada na versão atual. No modo de leitura da documentação, a ferramenta suporta a exibição dos seus itens utilizando um componente visual apropriado para cada tipo de item. Esta abordagem tem como vantagem o uso de multimídia e de padrões na documentação. Sua principal desvantagem está relacionada com a incapacidade de exportar os documentos produzidos.

3 Abordagem Proposta

O FrameDoc foi modelado segundo o padrão MVC (model – view – controller) (GAMMA et al., 1995), onde as classes que armazenam as informações (modelos) são separadas das classes que representam as informações (visões) e das classes que manipulam as informações (controles). A Figura 6 exibe um diagrama de classes focalizando os principais elementos componentes do FrameDoc.

As principais classes participantes do FrameDoc são:

- **GerenteDocumentação:** serve como ponto de acesso a chamadas externas ao FrameDoc. A utilização de uma ponto único de acesso ao FrameDoc, através do padrão *singleton* (GAMMA et al., 1995), facilita a instanciação e aumenta a manutenibilidade do *framework*;
- **Campo:** define a forma de exibição de uma informação. Existem vários tipos de campos, cada um apropriado para um tipo de informação. Um campo não guarda informações, mas sabe como exibi-las;
- **Informação:** item de informação de um tipo específico. O tipo será definido pelo campo gerador desta informação. Para que a informação seja exibida, o campo apropriado deve ser selecionado;

- **Documento:** armazena todas as informações relacionadas a um componente dentro do FrameDoc. Apesar de poder guardar as informações, não sabe como representá-las;
- ***Template*:** armazena um conjunto de campos. Pode ser visto como um meta-documento;
- **Categoria:** agrupa documentos e *templates*. É a forma idealizada para relacionar vários documentos com um *template* e vários *templates* com um documento;
- **Documentável:** representa um componente dentro do FrameDoc. Todo documento pertencerá a um e somente um componente e um componente só terá um documento no FrameDoc.

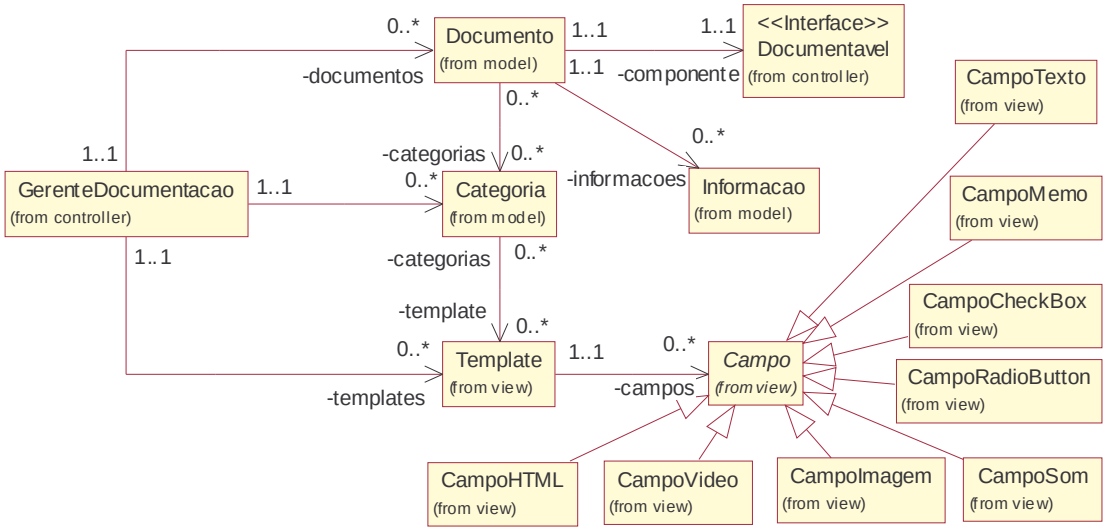


Figura 6: Diagrama de classes do FrameDoc.

A abordagem proposta nesse trabalho pode ser dividida em quatro etapas, conforme descrito na arquitetura exibida na Figura 7.

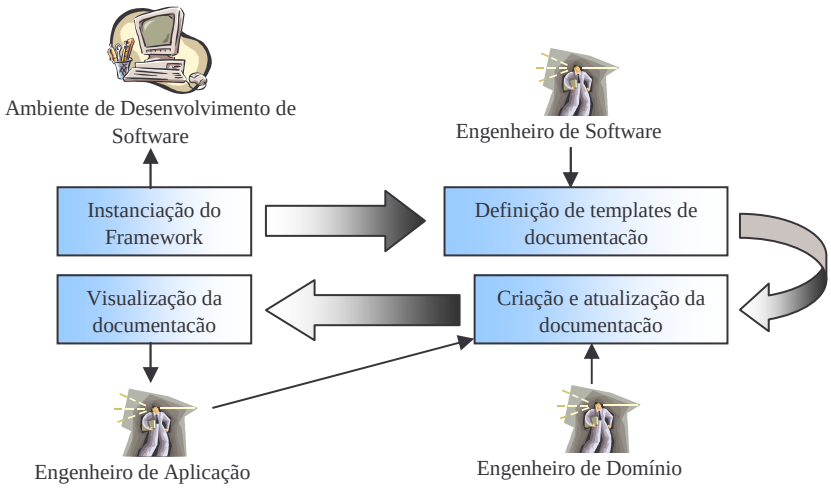


Figura 7: Arquitetura do FrameDoc.

A primeira etapa, a **instanciação do *framework***, está relacionada com as atividades necessárias para se registrar um componente, de forma que este seja documentado dentro de um ambiente. Esta etapa deve ser executada no ambiente de desenvolvimento de software que instancia o FrameDoc. A instanciação de um *framework* ocorre através da implementação de suas classes abstratas e interfaces, ou pela configuração de seus parâmetros.

A segunda etapa, a **definição de *templates* de documentação**, deve ser executada pelo configurador do ambiente de desenvolvimento de software, representado por um Engenheiro de Software que tenha conhecimento sobre quais informações são necessárias para documentar um determinado tipo de componente.

A terceira etapa, a **criação e atualização da documentação**, está relacionada com as atividades de empacotamento e adaptação, descritas no processo apresentado na Figura 1, e deve ser executada pelo Engenheiro de Domínio e pelo Engenheiro da Aplicação. O Engenheiro de Domínio é responsável pela documentação inicial dos componentes, gerada durante o processo de engenharia de domínio. O Engenheiro de Aplicação é responsável pela atualização desta documentação, acrescentando ou alterando as informações de acordo com a experiência resultante do uso do componente em uma ou mais aplicações desenvolvidas para o domínio.

A quarta etapa, a **visualização da documentação**, está relacionada com a atividade de compreensão do componente, também descrita na Figura 1. Ela deve ser executada pelo Engenheiro da Aplicação ao selecionar, e tentar compreender, o componente desejado.

A seguir discutimos brevemente cada uma dessas etapas.

3.1 Etapa de Instanciação

A instanciação do FrameDoc consiste na preparação do ambiente de desenvolvimento de software para permitir o acesso às janelas disponibilizadas pelo *framework* e na implementação das definições de documentação. Essas definições são providas pela interface “Documentavel”, que deve ser implementada pelos componentes que devem ser documentados.

Para que essa instanciação do FrameDoc na infraestrutura Odyssey fosse possível, foi necessário que cada componente manipulado pelas ferramentas do Odyssey fosse registrado no FrameDoc e associado a uma determinada categoria. Para que uma categoria exista, ela deve ser previamente cadastrada no FrameDoc. O cadastramento acontece no momento da inicialização do Odyssey e é efetuado para todas as categorias que a infraestrutura suporta, como, por exemplo, classes, casos de uso, atores, estados, diagrama de classes, entre outros. Sempre que um componente é removido do Odyssey, sua documentação é automaticamente removida do FrameDoc.

3.2 Etapa de Definição de Templates

A etapa de definição dos *templates* fornece o suporte necessário para toda a configuração da documentação do FrameDoc. O acesso à área de configuração deve ser restrito ao Engenheiro de Software responsável por definir os padrões a serem adotados para a documentação.

A etapa de instanciação serviu como uma configuração inicial, definindo qual componente está associado a qual categoria, mas essa configuração pode ser alterada. Um motivo para alterar a configuração inicial pode ser, por exemplo, o fato de um componente necessitar de uma documentação diferenciada dos demais componentes da mesma categoria.

A etapa de definição de *templates* se destaca da etapa de instanciação porque nesta etapa não é possível adicionar ou remover documentos do FrameDoc. Do ponto de vista do usuário, estas inclusões e remoções são automáticas, ocorrendo sempre que um novo componente for adicionado ou removido do Odyssey.

A etapa de instanciação já definiu um conjunto inicial de categorias. Na etapa de definição de *templates* é possível adicionar novas categorias e associá-las a *templates* já existentes, o que indiretamente é uma associação de um conjunto de componentes (os que estão associados à categoria) com os *templates*. Também é possível remover categorias (exceto as introduzidas na configuração inicial), o que indiretamente é uma desassociação dos componentes relacionados à categoria com os *templates* associados.

Quanto aos *templates*, é possível adicioná-los, removê-los e configurá-los. Ao solicitar a configuração de um *template*, o FrameDoc exibe um painel, que pode ser visto como um meta-editor de documentos, ou seja, um editor do estrutura dos documentos. O editor apresenta uma lista com os campos do *template*, oferecendo opções para editar, remover e adicionar campos. A Figura 8 exibe a janela de edição de *templates*. No exemplo selecionado, o *template* “Contexto” contém os campos “Domínio de Aplicação”, “Aplicabilidade” e “Casos de Uso”. Para cada campo é estabelecido um tipo e um valor *default*. O campo selecionado no exemplo é o de “Casos de Uso”, que fornece um editor de HTML para a configuração do seu valor *default*.

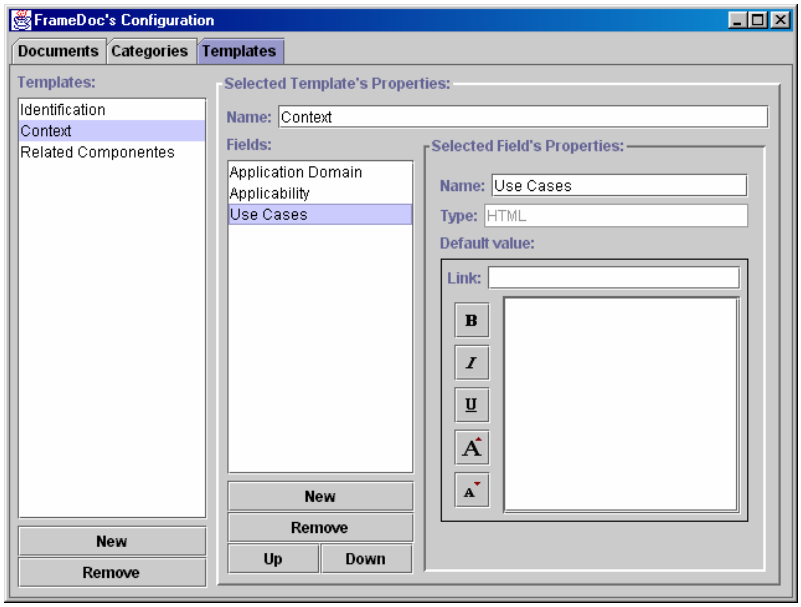


Figura 8: Janela de cadastro de *templates*.

3.3 Etapa de Criação e Atualização da Documentação

Esta etapa pode ser executada de duas formas distintas. A primeira está relacionada com a criação da documentação de um componente, e a segunda com a criação da infraestrutura para a posterior visualização da documentação.

Para que a documentação de um componente seja criada, o Odyssey solicita os painéis de documentação de um determinado componente (previamente cadastrado, na etapa de instanciação). Este pedido é repassado para cada categoria associada ao componente e para cada *template* associado à categoria. Os *templates* disponibilizam os seus campos com as informações já existentes para que possam ser atualizadas (ou criadas, no caso das informações ainda não existirem). Os campos de informação disponibilizados são:

- **Texto:** suporte para a entrada de uma linha de texto;
- **Memo:** suporte para a entrada de várias linhas de texto;
- **CheckBox:** suporte para a seleção de vários itens de um conjunto de itens;
- **RadioButton:** suporte para a seleção de um item de um conjunto de itens;
- **Som:** permite a seleção de um arquivo de som;
- **Imagem:** permite a seleção de um arquivo de imagem;
- **Vídeo:** permite a seleção de um arquivo de vídeo;
- **HTML:** permite a entrada de um texto com muitas linhas, e a associação de partes desse texto com páginas HTML (localizadas na Internet) ou com outros campos de documentação de componentes.

Para criar o suporte para posterior visualização da documentação, o FrameDoc permite a geração de páginas HTML. O pedido do Odyssey é processado de forma similar ao pedido de painéis de documentação, sendo que neste caso o campo disponibiliza o código HTML que exibe a informação desejada. Os trechos de código HTML retornados pelos campos formam a página de documentação de um determinado componente. É também fornecida a documentação implícita ao componente, ou seja, uma documentação existente no componente sem a necessidade do preenchimento de *templates* com campos pré-estabelecidos. Essa documentação é extraída das propriedades do componente. Por exemplo, uma classe que tem como documentação implícita os seus atributos, seus métodos e seu código fonte. Um diagrama de classes tem como

documentação implícita o desenho do seu diagrama. Esta documentação é gerada pelo próprio componente, a partir de um método pertencente à interface “Documentavel”.

3.4 Etapa de Visualização da Documentação

Para a visualização da documentação dos componentes reutilizáveis de software, selecionamos o padrão HTML, que é um padrão aberto e suportado por visualizadores gratuitos. O padrão HTML foi escolhido por ser amplamente utilizado na Internet e por fornecer o suporte necessário para o uso de *hyperlinks* e multimídia. A documentação pode ser visualizada internamente ou externamente ao ambiente de desenvolvimento de software. Para suportar a visualização interna da documentação, foi criado um visualizador de HTML, que é fornecido com os demais painéis de documentação. A Figura 9 exibe a visualização de um documento que representa um elemento do domínio judiciário.

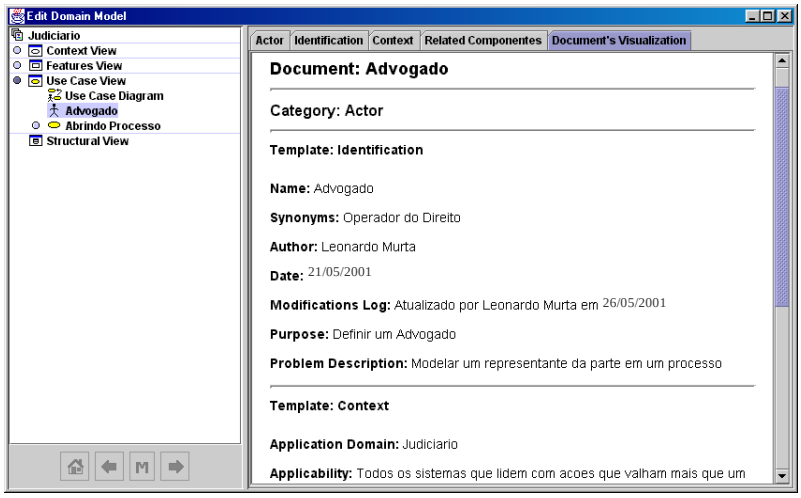


Figura 9: Janela de visualização da documentação.

O Odyssey pode, a qualquer momento, gerar uma versão da documentação dos seus componentes. A geração da documentação acontece no nível de componente. Através de uma lista com todos os componentes, é possível determinar quais irão participar da nova versão de documentação, deixando de exibir externamente documentações incompletas de componentes em construção. Também é possível gerar a documentação de todos os componentes de uma determinada categoria.

A documentação gerada poderá conter sons, imagens, vídeos e ligações para outras documentações ou páginas na Internet. O que determina os recursos de multimídia utilizados na documentação são os tipos dos campos contidos nos *templates* associados ao componente.

Para que o acesso à documentação gerada seja amigável, foi necessária a construção de páginas de índices para as documentações dos componentes. A página de índice principal (index.html) contém links para as páginas de índices de cada categoria. Essas, por sua vez, contêm links para a documentação de seus componentes. Após a geração da documentação, esta poderá ser visualizada por qualquer *browser* e exibida em sites na Internet. Todos os campos descritos na Seção 3.3 podem ser exibidos em modo de edição, quando estiverem dentro de uma janela do Odyssey, ou em modo de leitura, quando forem gerados em uma página HTML.

4 Conclusão

A Tabela 1 classifica as abordagens apresentadas na Seção 2 e a abordagem proposta de acordo com os critérios apresentados na Seção 2. Os casos preenchidos com ✓ indicam que o critério em questão foi atendido totalmente pela ferramenta. Os casos preenchidos com “PARCIAL” indicam que em alguns aspectos o critério é satisfeito, mas não completamente.

O FrameDoc cumpre os quesitos abordados na Tabela 1 de acordo com as seguintes características:

- Exportação da sua documentação utilizando o padrão HTML;
- Utilização de *hyperlinks* no campo HTML para outros documentos ou páginas na Internet;
- Criação de padrões de documentação, representados pelos *templates* de documentação;
- Geração de documentação modular, onde um módulo equivale a um componente;
- Utilização de multimídia na documentação, entretanto não permite a exibição dentro do ambiente, somente da documentação HTML exportada (via browser);
- Visualização da documentação por um painel especial fornecido junto com os demais painéis de edição da documentação;
- Geração da documentação implícita ao componente via método abstrato definido na interface "Documentavel".

Critério \ Abordagem	**JavaDoc**	**Rose 98**	**Fast Case**	**Memphis**	**FrameDoc**
Exportação em formato portável	✓	PARCIAL	✓		✓
Utilização de *hyperlinks* na documentação	✓			PARCIAL	✓
Criação de padrões de documentação				✓	✓
Geração de documentação modular	✓		✓	✓	✓
Utilização de multimídia na documentação				✓	PARCIAL
Visualização da documentação antes da geração			✓		✓
Geração de documentação implícita	✓	✓	✓		✓

Tabela 1: Comparação entre as ferramentas.

O FrameDoc possui algumas limitações. A versão atual fornece três campos multimídia, que são som, vídeo e imagem. Entretanto, o *framework* suporta apenas a manutenção da referência ao arquivo da mídia relacionada, não fornecendo suporte interno para vídeo ou som. Com isso, não é possível visualizar as informações multimídia dentro do ambiente que utiliza o *framework*, sendo esta visualização possível, somente com a documentação gerada, através do uso de um browser. Não é possível também desenvolver o conteúdo multimídia dentro da ferramenta, necessitando-se para tanto o uso de ferramentas externas.

Como trabalhos futuros, podemos citar a extensão do número de campos fornecidos e um mecanismo de busca de informações nos documentos produzidos. O FrameDoc atualmente tem um conjunto limitado de campos para a representação de informações. O campo a ser escolhido varia de acordo com o tipo de conhecimento que deve ser documentado. Além dos campos já implementados, seria útil a existência de um campo de lista de texto, um campo de lista de referências e um campo de referência. Apesar da inexistência desses campos, é possível representá-los através de campos memo (no lugar de lista de texto) ou HTML (no lugar de lista de referências e referência).

Na versão atual, o FrameDoc, ao gerar a documentação HTML de um componente, gera também uma página de índice para essa documentação. A página de índice auxilia, mas não soluciona o problema da localização dos componentes reutilizáveis desejados para uma aplicação. É necessária a construção de algum mecanismo de busca suportando a seleção de componentes. Outra possível melhoria seria utilizar XML na construção da documentação, separando a informação da sua representação.

Referências Bibliográficas

BRAGA, R. M. M., WERNER, C. M. L., 1999, "Odyssey-DE: Um Processo para Desenvolvimento de Componentes Reutilizáveis", *X CITS*, Curitiba.

GAMMA, E., HELM, R., JOHNSON, R., et al., 1995, "Design Patterns: Elements of Reusable Object-Oriented Software", primeira edição, *Addison Wesley*.

MILER, N., 2000, "A Engenharia de Aplicações no Contexto da Reutilização baseada em Modelos de Domínio", tese de mestrado, *COPPE/UFRJ*.

MOORE, J. M., BAILIN, S. C., 1991, "Domain Analysis: Framework for reuse", *IEEE Computer Society Press Tutorial*, pp. 179-202.

RATIONAL, 2001, "Rational Rose", disponível na Internet em http://www.rational.com/products/rose, acessado em 22/05/2001.

SILVA, M. F., 1998, "Documentação de Componetes Reutilizáveis: uma abordagem baseada nas tecnologias de Hipermídia e Padrões", tese de mestrado, *COPPE/UFRJ*.

SILVEIRA, D., 1999, "Fast Case: Uma Ferramenta Case para o Desenvolvimento Visual de Sistemas Orientados a Objetos", tese de mestrado, *IM/NCE/UFRJ*.

SUN MICROSYSTEMS, 2001a, "Java", disponível na Internet em http://www.java.sun.com, acessado em 11/05/2001a.

SUN MICROSYSTEMS, 2001b, "JavaDoc", disponível na Internet em http://java.sun.com/j2se/javadoc, acessado em 11/05/2001b.

TRACZ, W., 1994, "Software Reuse Myths Revisited", *16th International Conference on Software Engineering*, Sorento, Itália.

W3C, 2001, "HTML", disponível na Internet em http://www.w3.org/MarkUp/, acessado em 26/05/2001.

WERNER, C. M. L., BRAGA, R. M. M., MATTOSO, M., et al., 2000, "Infra-estrutura Odyssey: estágio atual", *XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, João Pessoa, outubro.

WERNER, C. M. L., TRAVASSOS, G. H., ROCHA, A. R., et al., 1996, "Memphis: Um Ambiente para Desenvolvimento de Software Baseado em Reutilização: Definição da Arquitetura", relatório técnico 3/96, *COPPE/UFRJ*.

WIRFS-BROCK, R. J., JOHNSON, R. E., 1990, "Surveying Current Research in Object-Oriented design", *Communications of the ACM*, volume 33 (9).