

FRAMEDOC: UM FRAMEWORK PARA A DOCUMENTAÇÃO DE COMPONENTES REUTILIZÁVEIS

Leonardo Gresta Paulino Murta

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Informática.

Apresentado por:

Leonardo Gresta Paulino Murta

Aprovado por:

Prof^a Cláudia Maria Lima Werner, D.Sc.
(Presidente)

Márcio de Oliveira Barros, M.Sc.
(Co-orientador)

Prof. Marcos Roberto da Silva Borges, Ph.D.

Prof. Marta Lima de Queirós Mattoso, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 1999

Aos meus pais, e à Bia.

Agradecimentos

A Deus, pois sem Ele nada seria possível.

Aos meus pais, que são o meu grande exemplo de vida, honestidade e dignidade.

À Professora Cláudia Werner, que sempre me apoiou com muita paciência e compreensão, tornando-se para mim muito mais que uma orientadora, uma grande amiga, digna de toda a minha gratidão e confiança.

Ao amigo e orientador Márcio Barros, que é meu guru não somente quanto ao projeto final, mas em qualquer aspecto relacionado com computação.

Ao Alexandre Dantas, pela ajuda na instanciação do FrameDoc no Ambiente Odyssey.

A todos os Professores do curso de Informática da UFRJ, que, além de grandes mestres, são grandes pessoas e serão lembrados por toda a vida. Em especial aos Professores Maria Luiza, Marta, Elaine, Marcos Borges e João Carlos.

À Bia, que teve toda a paciência do mundo para aceitar os fins de semana que não estava com ela, mas sim trabalhando no computador.

Aos meus amigos e colegas que direta ou indiretamente contribuíram para a realização dessa jornada. Em especial ao Vítor, Sascha e Humberto.

A todos os meus parentes que, desde pequenino, estavam torcendo por mim.

RESUMO

FRAMEDOC: UM FRAMEWORK PARA A DOCUMENTAÇÃO DE COMPONENTES REUTILIZÁVEIS

Leonardo Gresta Paulino Murta

Orientadores: Cláudia Maria Lima Werner e Márcio de Oliveira Barros

Este trabalho apresenta um *framework* para a documentação de componentes reutilizáveis, utilizando as tecnologias de multimídia e padrões.

A abordagem aqui proposta visa fornecer o ferramental necessário para o empacotamento de componentes, e sua posterior compreensão.

ABSTRACT

FRAMEDOC: A FRAMEWORK FOR DOCUMENTING REUSABLE COMPONENTS

Leonardo Gresta Paulino Murta

Supervisor: Cláudia Maria Lima Werner and Márcio de Oliveira Barros

This work presents a framework for the documentation of reusable components, using hypermedia and pattern technologies.

The proposed approach provide the necessary tools for component packaging and its subsequent understanding.

SUMÁRIO

1 – INTRODUÇÃO	1
1.1 – MOTIVAÇÃO	1
1.2 – OBJETIVOS.....	4
1.3 – ORGANIZAÇÃO.....	4
2 – FERRAMENTAS DE DOCUMENTAÇÃO EXISTENTES.....	6
2.1 – JAVADOC	6
2.2 – RATIONAL ROSE 98.....	8
2.3 – FAST CASE	9
2.4 – DOCUMENTAÇÃO DE COMPONENTES NO AMBIENTE MEMPHIS.....	11
2.5 – OUTRAS FERRAMENTAS	13
2.6 – COMPARAÇÃO ENTRE AS FERRAMENTAS	13
3 – ABORDAGEM PROPOSTA: FRAMEDOC	16
3.1 – MODELO CONCEITUAL.....	16
3.2 – ETAPAS PARA A DOCUMENTAÇÃO DE COMPONENTES.....	18
3.2.1 – ETAPA DE INSTANCIAÇÃO	19
3.2.2 – ETAPA DE DEFINIÇÃO DOS TEMPLATES	20
3.2.3 – ETAPA DE CRIAÇÃO E ATUALIZAÇÃO DA DOCUMENTAÇÃO.....	22
3.2.4 – ETAPA DE VISUALIZAÇÃO DA DOCUMENTAÇÃO	24
4 – PROJETO E IMPLEMENTAÇÃO.....	26
4.1 – MODELO DETALHADO.....	26
4.2 – ETAPA DE INSTANCIAÇÃO	27
4.3 – ETAPA DE DEFINIÇÃO DOS TEMPLATES	27
4.4 – ETAPA DE CRIAÇÃO E ATUALIZAÇÃO DA DOCUMENTAÇÃO	29
4.5 – ETAPA DE VISUALIZAÇÃO DA DOCUMENTAÇÃO	31
5 – INSTANCIAÇÃO DO FRAMEDOC NO AMBIENTE ODYSSEY.....	37
5.1 – AMBIENTE ODYSSEY.....	37
5.2 – ETAPAS PARA A INSTANCIAÇÃO.....	37
5.2.1 – PERSISTÊNCIA DO GERENTE DE DOCUMENTAÇÃO	38
5.2.2 - ACESSO ÀS JANELAS DE CONFIGURAÇÃO E GERAÇÃO	39
5.2.3 – CADASTRAMENTO DAS CATEGORIAS.....	40

5.2.4 – CADASTRAMENTO DOS COMPONENTES.....	41
5.2.5 – ACESSO AOS PAINÉIS DE UM COMPONENTE	42
6 – CONCLUSÕES	43
6.1 – CONTRIBUIÇÕES.....	43
6.2 – LIMITAÇÕES	44
6.3 – TRABALHOS FUTUROS	44
REFERÊNCIAS BIBLIOGRÁFICAS	46
APÊNDICE A – DIAGRAMAS DE SEQÜÊNCIA	48
A.1 – ETAPA DE INSTANCIACÃO	48
A.2 – ETAPA DE DEFINIÇÃO DOS TEMPLATES.....	52
A.3 – ETAPA DE CRIAÇÃO E ATUALIZAÇÃO DA DOCUMENTAÇÃO	59

1 – INTRODUÇÃO

1.1 – MOTIVAÇÃO

A reutilização de software pode ser abordada sob duas perspectivas complementares [Moore; 1991]. A primeira, conhecida como desenvolvimento *para* reutilização, se baseia em identificar os componentes¹ 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 modificar o componente selecionado, integrando-o ao projeto em desenvolvimento. A figura 1.1 exibe a interação entre esses pontos de vista.

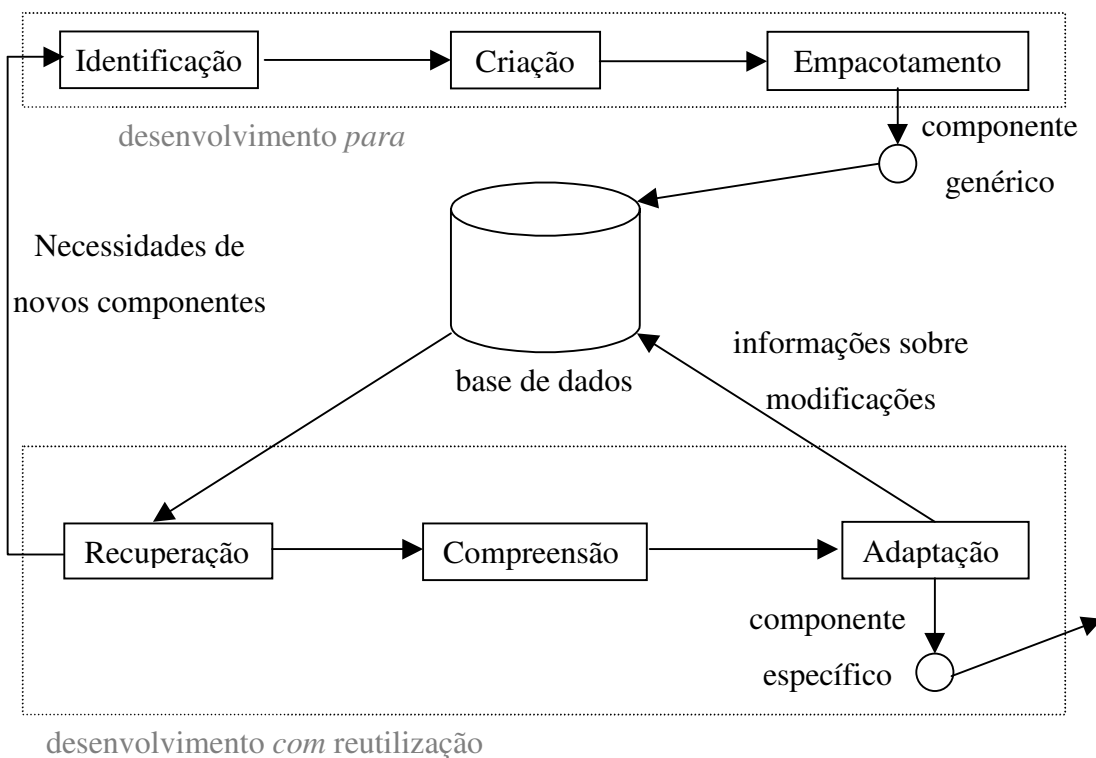


Figura 1.1: Processo de reutilização

¹ No contexto abordado, um componente pode ser visto como um produto de software em qualquer nível de abstração (ex.: classes, casos de uso, diagramas, especificações, etc.).

O desenvolvimento *para* reutilização equivale à *Engenharia de Domínio*, que tem como objetivo construir componentes reutilizáveis que solucionem problemas de domínios de conhecimento específicos [Werner; 1999a]. Esses componentes têm que ser genéricos o bastante para que possam ser utilizados em aplicações desenvolvidas no domínio em questão.

O desenvolvimento *com* reutilização equivale a *Engenharia da Aplicação*, que tem como objetivo construir aplicações em um determinado domínio, utilizando os componentes genéricos já existentes [Werner; 1999a].

Para que a reutilização se torne efetiva, se faz necessário a disponibilização de ferramentas que suportem as atividades exibidas na figura 1.1 [Tracz; 1994]. Dentre estas atividades, as de empacotamento e compreensão se interrelacionam, criando a necessidade de uma ferramenta de documentação de componentes, que controle a criação e disponibilização de documentos que contenham informações sobre os componentes armazenados na base de dados. Este suporte fornecido na atividade de empacotamento facilitará a realização da atividade de compreensão.

O grupo de Reutilização da COPPE/UFRJ está construindo uma infra-estrutura de reutilização baseada em modelos de domínio (Projeto Odyssey²) [Werner; 1999b]. Esta infra-estrutura tem como objetivo prover o apoio necessário ao processo descrito na figura 1.1, incluindo uma ferramenta de documentação de componentes, objeto de estudo e desenvolvimento deste trabalho.

A documentação dentro do desenvolvimento de software não faz parte de uma fase definida, mas ocorre durante toda a sua existência, em paralelo com todas as fases do ciclo de vida. Ela pode ser estruturada de forma linear ou associativa [Borges; 1998].

A abordagem à documentação feita por textos lineares é, como o próprio nome diz, seqüencial. Todo e qualquer documento é desenvolvido para ser lido em uma ordem

² O Projeto Odyssey está detalhado na seção 5.1.

pré-determinada e inalterável. O texto linear força ao leitor um nível de abstração pré-determinado, não dando a ele a opção de, em um determinado momento, saber mais sobre um determinado tema. Essa documentação terá como característica a disjunção entre os documentos gerados (ou a geração de um documento monolítico).

O suporte oferecido por uma abordagem associativa (baseada em hipertexto) será semelhante ao de textos lineares, só que sem a necessidade da linearidade. Poderemos considerar como um nó do hipertexto qualquer documentação de componente gerado por qualquer fase. Essa documentação fará referências (ligações) à documentação de outros componentes gerados na mesma ou de outra fase do ciclo de vida. Assim teremos, por exemplo, a documentação do componente "especificação do projeto ABC", que tem uma ligação para a documentação do componente "diagrama UML do framework colégio", que tem ligação para a documentação do componente "código em Java da classe Aluno".

O pensamento humano não trabalha de forma seqüencial. Todo mecanismo relacionado à manipulação de informações (que está diretamente associada à capacidade do pensamento humano) deve ser o mais próximo possível do modo que trabalha o pensamento. Esse modo é o associativo, que é a forma de trabalho dos hipertextos. Algumas vantagens dos hipertextos em relação aos textos lineares são [Borges; 1998]:

- Proporcionam conectividade entre as informações;
- Oferecem uma interface compatível com o modo de raciocínio humano;
- Permitem ao usuário definir o nível de abstração desejado (navegando entre as documentações dos componentes).

Devido às fases do processo de desenvolvimento de software serem altamente inter-relacionadas, os textos lineares apresentam uma deficiência: a não capacidade de expressar a fronteira deste inter-relacionamento. Por exemplo [Murta; 1998b]:

- Utilizando um único texto linear como a documentação de todo o processo, não teremos uma visão individualizada de cada etapa (dificuldade de

localização da informação desejada);

- Utilizando vários textos lineares com a documentação de cada etapa, não teremos uma visão global do processo (dificuldade de relacionar informações pseudo distintas).

1.2 – OBJETIVOS

Este trabalho tem como objetivo fornecer o suporte à documentação de componentes no Ambiente Odyssey³. Com a necessidade deste suporte de documentação de componentes, foi idealizado qual seria a melhor forma de implementá-lo. Para que a solução adotada não fosse dependente do Ambiente Odyssey, e que pudesse ser utilizada por outros ambientes de desenvolvimento de software seria necessária a construção de um framework⁴, que suportasse a criação e a disponibilização da documentação de componentes de software: o FrameDoc.

1.3 – ORGANIZAÇÃO

Este trabalho está organizado da seguinte forma:

O capítulo 1 definiu o problema e indicou o caminho que será seguido no decorrer deste trabalho para a resolução do problema definido.

O capítulo 2 exhibe, o estado atual das ferramentas de documentação existentes, fornecendo uma comparação entre elas.

O capítulo 3 descreve uma possível abordagem para o problema de documentação de componentes reutilizáveis. Essa descrição é apresentada em um nível abstrato, equivalente à fase de análise.

³ Ambiente de Desenvolvimento de Software construído pelo Projeto Odyssey.

⁴ 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; 1990].

O capítulo 4 apresenta as decisões de projeto necessárias para a construção do FrameDoc e alguns aspectos da implementação.

No capítulo 5, é apresentado um caso de uso⁵ do FrameDoc. Inicialmente é descrito o Ambiente Odyssey e, em seguida, como o FrameDoc foi instanciado neste contexto.

Finalmente, o capítulo 6 lista as contribuições deste trabalho, apontando possíveis limitações da abordagem adotada e indicando características que o FrameDoc não suporta, e que podem ser contempladas em trabalhos futuros.

O apêndice A apresenta os diagramas de seqüência construídos na fase de projeto do FrameDoc.

⁵ Apesar de o Projeto Odyssey ser o motivo da criação do FrameDoc, ele pode também ser visto como um caso de uso.

2 – FERRAMENTAS DE DOCUMENTAÇÃO EXISTENTES

Este capítulo apresenta um conjunto de ferramentas de documentação existentes, algumas acadêmicas, outras comerciais, enfatizando sua utilização tanto na criação da documentação (desenvolvimento *para* reutilização) quanto na visualização da documentação (desenvolvimento *com* reutilização). Dentre elas estão: JavaDoc, Rational Rose 98, Fast Case e a ferramenta de documentação de componentes do ambiente MEMPHIS.

Ao final do capítulo é feita uma comparação entre as ferramentas apresentadas.

2.1 – JAVADOC

A linguagem Java [Sun; 1999] descrita em [Murta; 1998a] permite um estilo especial de comentário de código, além dos habituais herdados da linguagem C++⁶, que são “//” para uma linha e “/* ... */” para um bloco. Este estilo especial é utilizado pelo padrão de documentação JavaDoc [Sun; 1999]. Ele é 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, juntamente com seus métodos e atributos. Essa descrição se dá através de *tags*, no interior do bloco especial de comentário, que começam por “@”. Uma classe comentada (documentada) através de JavaDoc pode ser submetida a um interpretador, que irá gerar um HTML como documentação. A figura 2.1 exibe um exemplo de comentário JavaDoc em um método, e a figura 2.2 mostra 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;

⁶ Para que o aprendizado da linguagem Java fosse mais eficiente, foi adotada uma sintaxe similar a de C++.

- *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.

```
/**
 * Altera o diagramador sobre o qual o agente atuará
 *
 * @param d O diagramador desejado
 */
public void SetDiagram(DCDiagramBox d)
{
    Diagram = d;
}
```

Figura 2.1 – Trecho de código documentado usando Javadoc
(Fonte: Projeto Odyssey – COPPE/UFRJ)

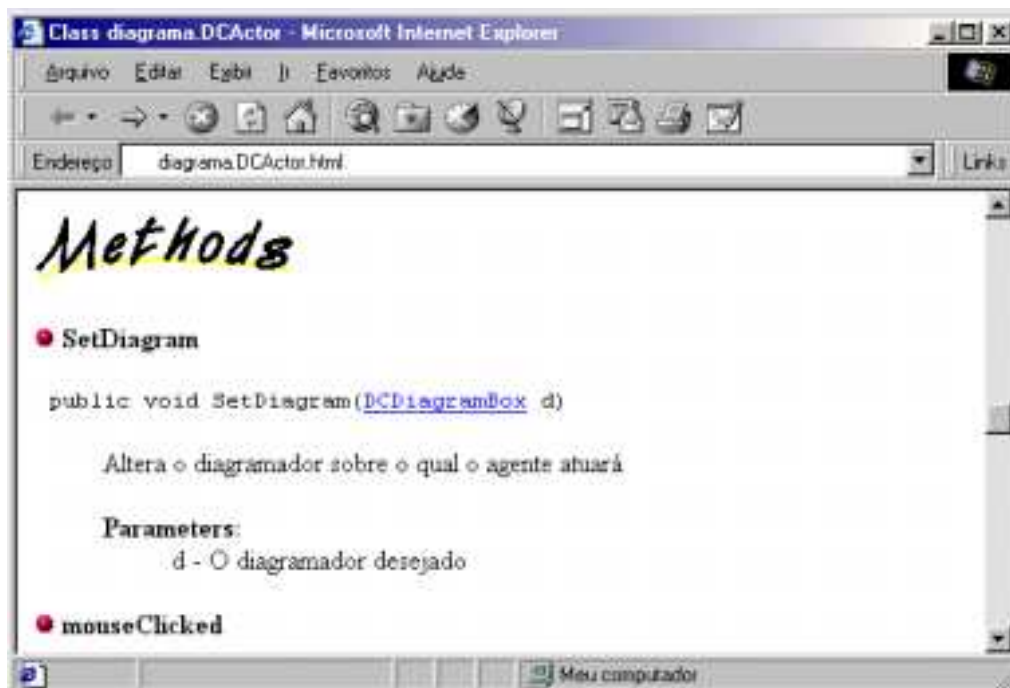


Figura 2.2 – Página HTML gerada pelo Javadoc

O Javadoc apresenta vantagens como a geração da documentação em um formato portátil e a utilização de hiperlinks, entretanto não permite a utilização de multimídia e a configuração de padrões de documentação.

2.2 – RATIONAL ROSE 98

O Rational Rose 98 [Rational, 1998] é hoje uma das ferramentas CASE orientadas a objetos mais utilizadas em projetos comerciais e acadêmicos.

A documentação no Rose é formada pelos dados contidos no componente (ex.: atributos e métodos de uma classe) e por um campo textual específico para documentação, que contém 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 Word, através da opção de geração via OLE (localizada em *Report|Documentation Report*).

Esta geração via OLE cria um documento monolítico no formato *doc*. 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 devido à lentidão do protocolo OLE, e a criação de um único módulo de documentação para todo o projeto, não fornecendo conectividade entre as informações geradas.

A figura 2.3 exibe uma classe construída no Rose. Esta classe contém um atributo e três métodos, sendo que um deles é o construtor da classe.

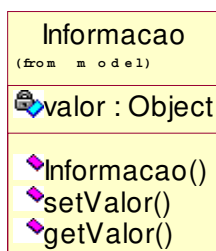


Figura 2.3 – Classe modelada no Rational Rose 98

Para esta classe, é gerada uma documentação composta por uma página de capa, contendo as informações do projeto e as opções selecionadas para a geração da documentação, uma página de índice e um conjunto de páginas com a documentação propriamente dita. A figura 2.4 exibe a página de documentação gerada para a classe mostrada na figura 2.3.

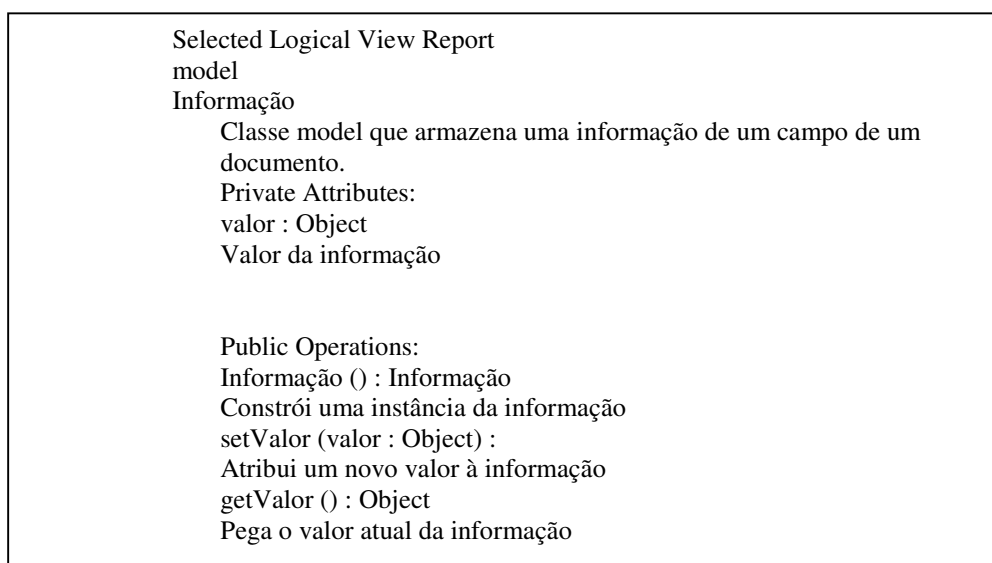


Figura 2.4 – Documentação gerada pelo Rational Rose 98

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. A figura 2.5 exibe a edição dos atributos de uma classe no Fast Case.

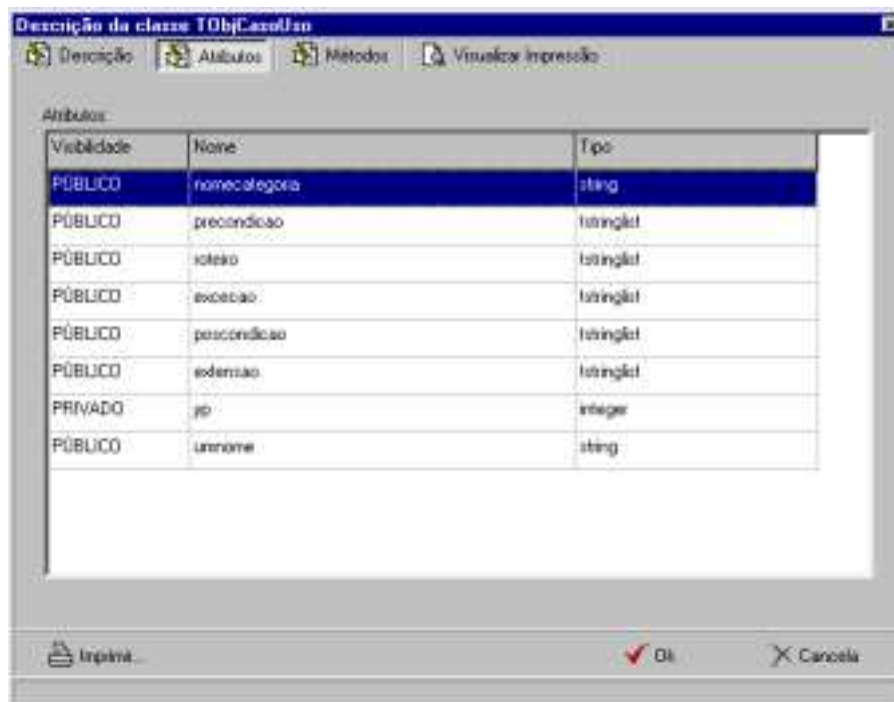


Figura 2.5 – Edição de atributos de uma classe no Fast Case.

Uma característica do Fast Case não existente no Rose é a possibilidade de visualizar a documentação antes de gerá-la. A figura 2.6 exibe a visualização da classe editada na figura 2.5.



Figura 2.6 – Visualização da documentação de uma classe no Fast Case.

As desvantagem do Fast Case são a falta de suporte para a definição de padrões de documentação e a falta de suporte para o uso de hiperlinks e multimídia.

2.4 – DOCUMENTAÇÃO DE COMPONENTES NO AMBIENTE MEMPHIS

A solução prevista no Ambiente MEMPHIS⁷ [Werner; 1996] para a documentação dos seus componentes [Silva; 1998] consiste na geração de padrões de documentação⁸ na forma de templates, para que estes sejam utilizados no momento do empacotamento do componente.

Um template é definido como um conjunto de itens, cada qual armazenando um tipo de informação.

A ferramenta conta com itens dos tipos: texto, diagrama, referência a um hiperdocumento, lista, som e vídeo. A figura 2.7 exibe o editor de formato de documento no momento da seleção do tipo de um item de documentação.

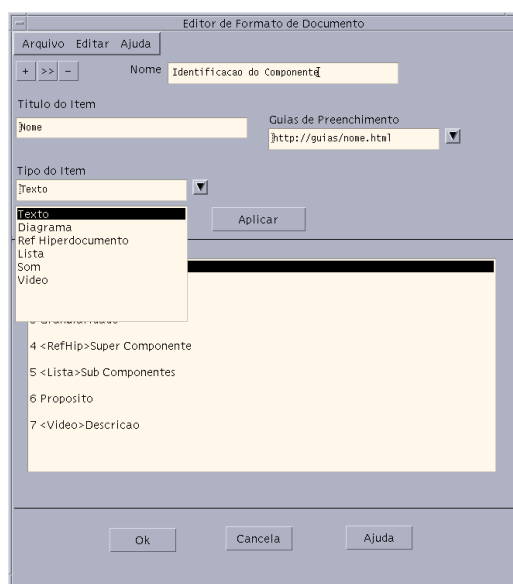


Figura 2.7: Criação de um template no Ambiente MEMPHIS (Fonte: [Silva; 1998])

⁷ O Ambiente MEMPHIS é um Ambiente de Desenvolvimento de Software Baseado em Reutilização.

⁸ Os padrões de documentação definem em um nível meta quais as informações que serão contidas nos documento que seguirem o padrão em questão.

Tanto a geração do template quanto o preenchimento e a visualização são realizados dentro do próprio Ambiente MEMPHIS. A visualização da documentação externa ao MEMPHIS foi planejada, porém não está implementada na versão atual.

No modo de leitura da documentação, a ferramenta suporta a exibição dos seus itens de documentação utilizando o componente visual apropriado para cada tipo de item. Por exemplo: para exibir um texto é utilizado o componente visual *memo box*; para a exibição de um vídeo é utilizado um *video player*, como exibido na figura 2.8.

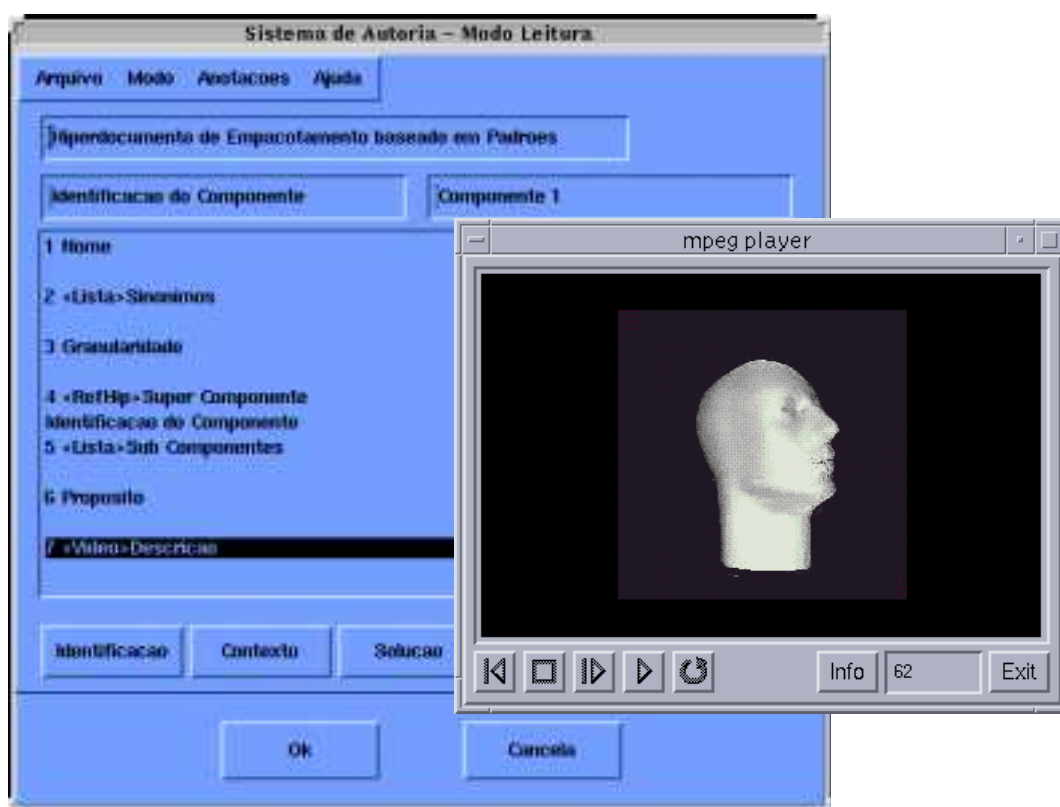


Figura 2.8: Exibição de um item de vídeo no Ambiente MEMPHIS (Fonte: [Silva; 1998])

Esta ferramenta tem como vantagem o uso de multimídia e de padrões na documentação e como desvantagem a não exportação da documentação.

2.5 – OUTRAS FERRAMENTAS

Na literatura existem outras ferramentas que se propõem a documentar componentes reutilizáveis, no contexto de abordagens de reutilização, como por exemplo [Silva; 1998]:

- RIG: Padrões de integração para várias bibliotecas de componentes reutilizáveis;
- ORGANON: Ambiente baseado em conhecimento apoiando a reutilização;
- REBOOT: Aplicação de reutilização holística, cobrindo não somente aspectos técnicos, mas organização como um todo;
- KAPTUR: Documentação de decisões de projeto utilizando hipertexto.

Entretanto, por falta de documentação detalhada e a indisponibilidade das ferramentas para testes, não foi possível identificar os mecanismos utilizados para a criação e visualização da documentação.

2.6 – COMPARAÇÃO ENTRE AS FERRAMENTAS


As ferramentas descritas neste capítulo foram comparadas segundo os seguintes critérios, que são características cobertas totalmente, parcialmente, ou não cobertas pelas ferramentas:


- **Exportação da documentação em formato portátil:** Indica se a ferramenta permite salvar os documentos gerados em um formato aberto (que possa ser visualizado por programas de licença gratuita, como por exemplo: html, ps, rtf, pdf, etc.);
- **Utilização de hiperlinks na documentação:** Indica se a ferramenta permite a criação de referências (links) de componentes ou partes de componentes da documentação para outros componentes ou sites na Internet;
- **Criação e configuração de padrões de documentação:** Indica se a ferramenta permite a criação de templates, ou formato pré-definido e

configurável, para que a documentação ocorra de forma padronizada, de modo que o usuário forneça informações pré-determinadas, para cada tipo de componente, podendo alterá-las assim que o desejar;

- **Geração de documentação modular:** Indica se a ferramenta permite a geração de documentos separados para componentes distintos, evitando o acúmulo de informações em documentos monolíticos;
- **Utilização de multimídia na documentação:** Indica se a ferramenta permite que sejam utilizados recursos de imagem, som e vídeo para documentar os componentes;
- **Visualização da documentação antes da geração:** Indica se a ferramenta permite que a documentação seja visualizada antes da sua geração, com uma formatação similar à que será gerada;
- **Geração da documentação implícita ao componente:** Indica se a ferramenta permite a exibição de características inerentes ao próprio componente na sua documentação, como por exemplo: atributos e métodos de uma classe ou pré e pós condições de um caso de uso.

Os critérios acima foram definidos de acordo com as deficiências das ferramentas avaliadas em relação à documentação de componentes reutilizáveis. Apesar de não formarem um conjunto completo, servem como uma guia de objetivos a serem atingidos pelo FrameDoc.

A tabela exibida na figura 2.9 classifica as ferramentas de acordo com os critérios apresentados. Os campos preenchidos com  indicam que os critérios em questão são cobertos totalmente pela ferramenta.

Os casos preenchidos com  indicam que em alguns aspectos o critério é satisfeito, mas em outros não. Isto ocorre, por exemplo, no Rational Rose 98, no critério “exportação da documentação em formato portátil”, pois o formato de geração da documentação (MS-Word), apesar de ser amplamente utilizado, não é aberto e nem tem licença gratuita. Também ocorre no MEMPHIS, no critério “Utilização de hiperlinks na documentação”, pois não é possível criar referências para sites na Internet.

Critério\Ferramenta	JavaDoc	Rose 98	Fast Case	MEMPHIS
Exportação da documentação em formato portátil	✓	±	✓	
Utilização de hiperlinks na documentação	✓			±
Criação e configuração de padrões de documentação				✓
Geração de documentação modular	✓		✓	✓
Utilização de multimídia na documentação				✓
Visualização da documentação antes da geração			✓	
Geração da documentação implícita ao componente	✓	✓	✓	

Figura 2.9: Comparativo entre as ferramentas exibidas no capítulo

A partir desta avaliação, concluímos que nenhuma das ferramentas exibidas neste capítulo atende completamente os critérios propostos, o que motiva a construção de uma ferramenta para a documentação de componentes reutilizáveis.

3 – ABORDAGEM PROPOSTA: FRAMEDOC

Neste capítulo são apresentadas, em nível de análise, as características desejadas em uma ferramenta de documentação de componentes reutilizáveis.

A abordagem proposta para a construção da ferramenta é baseada em *frameworks* [Wirfs-Brock; 1990], que são um conjunto de classes formando uma arquitetura de alto nível, para solucionar um determinado problema.

3.1 – MODELO CONCEITUAL

A figura 3.1 exibe o modelo de classes do FrameDoc, utilizando a notação UML [Fowler; 1997]. O modelo apresenta somente as classes (desconsiderando seus atributos e métodos). Um modelo mais detalhado será apresentado no próximo capítulo.

Tanto o modelo apresentado na figura 3.1 quanto os demais modelos de classes, casos de uso e seqüência foram construídos no Rational Rose 98 [Rational, 1998].

O FrameDoc foi modelado segundo o padrão MVC (*model – view – controller*) [Gamma; 1995], onde classes que armazenam as informações (modelos) são separadas das classes que representam as informações (visões) e das classes que manipulam as informações. As classes envolvidas no FrameDoc são:

- GerenteDocumentação: Serve como ponto de acesso a chamadas externas ao FrameDoc.
- 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 exibir uma.
- 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.

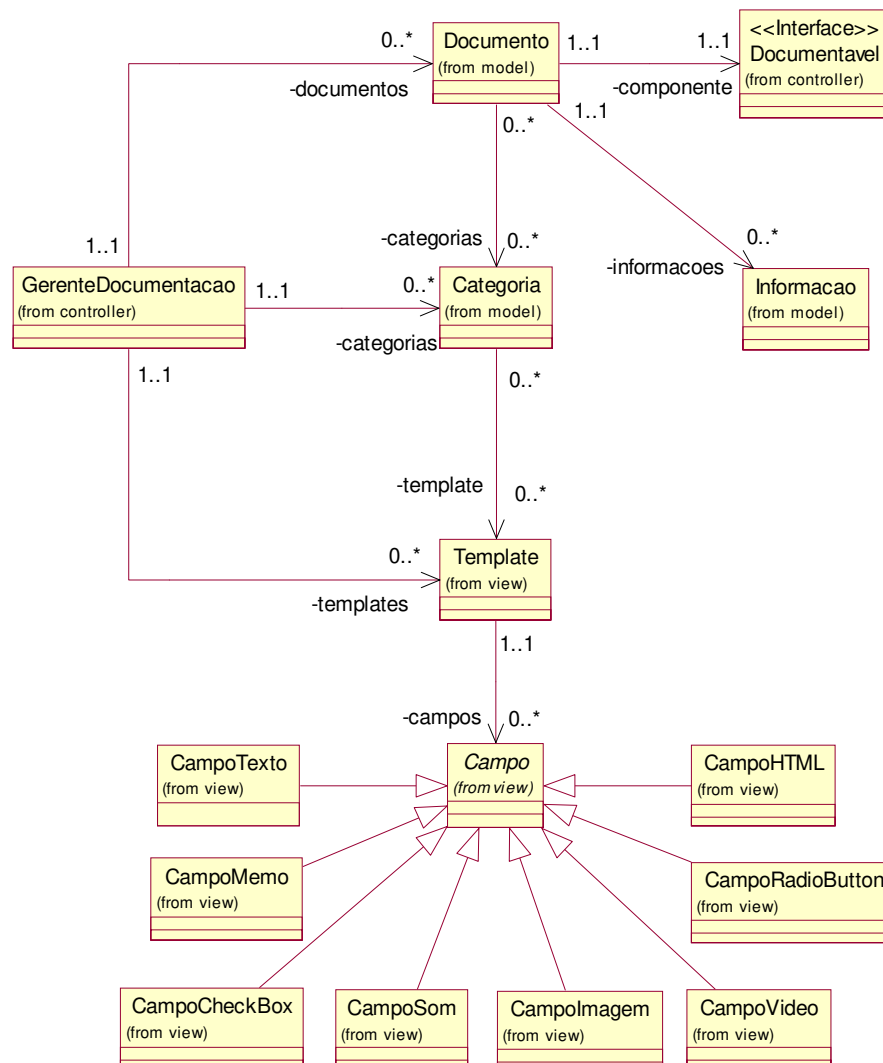


Figura 3.1: Modelo de classes do FrameDoc em nível abstrato

3.2 – ETAPAS PARA A DOCUMENTAÇÃO DE COMPONENTES

O suporte à documentação no contexto do FrameDoc pode ser dividido em quatro etapas, como descrito na figura 3.2.

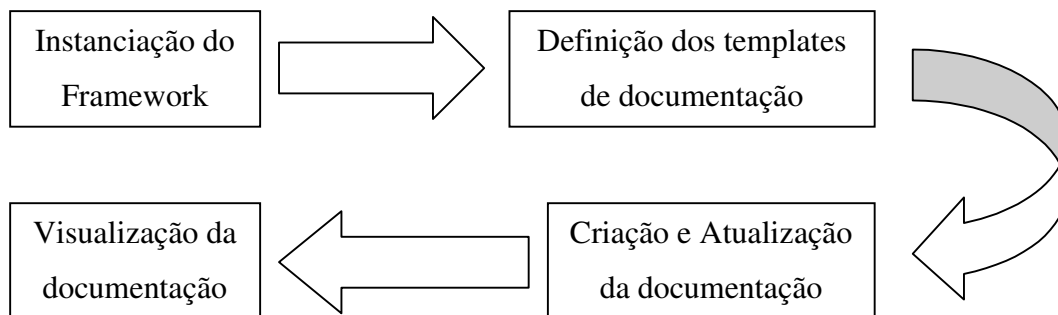


Figura 3.2: Etapas para a documentação de componentes

A primeira etapa, a instanciação do framework, está relacionada com as atividades necessárias para se registrar um componente para que este seja documentado. Esta etapa deve ser executada pelo ambiente de desenvolvimento de software que instancia o FrameDoc.

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

A terceira etapa, de criação e atualização da documentação, está relacionada com a atividade de empacotamento⁹, descrita na figura 1.1. Ela deve ser executada pelo Engenheiro de Software, que pode ser um Engenheiro de Domínio ao criar o componente, ou um Engenheiro da Aplicação ao adaptar o componente.

⁹ De certa forma também está relacionada com a atividade de adaptação, pois o suporte de documentação fornecido pelo FrameDoc permite documentar componentes tanto a nível de Engenharia de Domínio (componentes genéricos) quanto a nível de Engenharia da Aplicação (componentes específicos). A documentação de um componente específico é a adaptação da documentação do componente genérico à aplicação em questão.

A quarta etapa, de visualização da documentação, está relacionada com a atividade de compreensão do componente, também descrita na figura 1.1. Ela deve ser executada pelo Engenheiro da Aplicação ao selecionar (e tentar compreender) o componente desejado.

A seguir, descreveremos cada uma das etapas de suporte à documentação do FrameDoc.

3.2.1 – ETAPA DE INSTANCIAÇÃO

Para que o FrameDoc possa ser instanciado em um ambiente de desenvolvimento de software, é necessário que cada componente do ambiente seja registrado no FrameDoc e associado a uma determinada categoria. Por exemplo: a classe “aluno”, ao ser criada, deverá ser registrada no FrameDoc e, posteriormente, associada à categoria “Classe”, pois a classe aluno é um componente do tipo classe.

Para que uma categoria exista, o ambiente de desenvolvimento de software deve cadastrá-la no FrameDoc. Este cadastramento terá que acontecer no momento da inicialização do ambiente, para que só ocorra uma vez, e deve ser efetuado para todas as categorias que o ambiente suporta (ex.: Classe, Caso de Uso, Ator, Estado, Diagrama de Classes, etc.).

Quando um componente for removido do ambiente de desenvolvimento de software, também deve ser removido do FrameDoc. A figura 3.3 exhibe todos os casos de uso relacionados com a instanciação do FrameDoc.

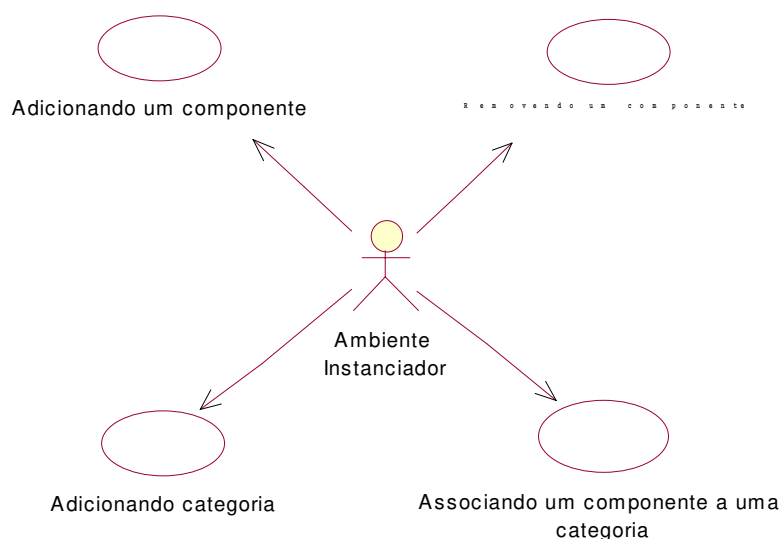


Figura 3.3: Diagrama de casos de uso relacionados com a instânciação do FrameDoc

3.2.2 – ETAPA DE DEFINIÇÃO DOS TEMPLATES

Após a instanciação do FrameDoc será necessário configurá-lo, ou seja, criar os templates de documentação necessários e associá-los a categorias. A etapa de instanciação serviu como uma configuração inicial, definindo qual componente está associado com qual categoria, mas essa configuração pode ser alterada. Um motivo para alterar a configuração inicial pode ser, por exemplo, um componente necessitar de uma documentação diferenciada dos outros componentes da mesma categoria.

A diferença em relação à etapa de instanciação é que nesta etapa não será possível adicionar ou remover componentes do FrameDoc, pois, do ponto de vista do usuário, isto será automático (graças à etapa de instanciação).

A etapa de instanciação já definiu um conjunto inicial de categorias. Nesta etapa será 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 será 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, deve ser possível adicioná-los, removê-los e configurá-los. Ao solicitar a configuração de um template, o FrameDoc deve exibir um painel, que pode ser visto como um meta-editor de documentos, com a lista de todos os campos do template e opções para editar e remover um campo existente ou adicionar um novo campo.

Todos os casos de uso descritos nesta fase estão modelados na figura 3.4.

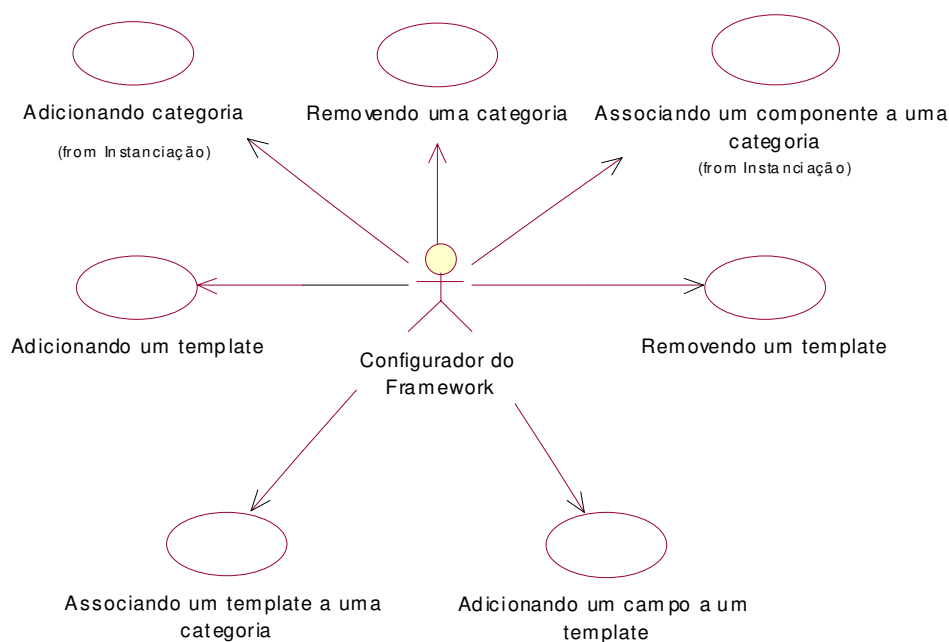


Figura 3.4: Diagrama de casos de uso relacionados com a configuração do FrameDoc

Inicialmente o FrameDoc deverá fornecer três templates genéricos (adaptações dos templates propostos em [Silva, 1998]), que podem ser utilizados para a documentação de qualquer tipo de componente. Esses templates são:

1. Identificação do Componente: Contém dados que ajudam o Engenheiro da Aplicação a decidir se realmente é este o componente desejado;
2. Contexto: Descreve a aplicação do componente;
3. Componentes Relacionados: Informa componentes que também podem ser úteis para solucionar o problema em questão.

O template de Identificação do Componente é composto pelos seguintes campos:

- Nome: Termo que identifique o componente;
- Sinônimos: Outros termos que também servem para identificar o componente;
- Autor: Nome do criador do componente;
- Data: Data de criação do componente;
- Histórico de Modificações: Nome do modificador e data de modificação do componente;
- Propósito: Descrição dos propósitos do componente;
- Descrição do Problema: Descrição da razão de existência do componente.

O template de Contexto é composto pelos seguintes campos:

- Domínio de Aplicação: Domínio de aplicação para o qual o componente foi projetado;
- Aplicabilidade: Situações típicas em que o componente pode ser usado;
- Casos de Uso: Situações reais de uso do componente.

O template de Componentes Relacionados é composto pelos seguintes campos:

- Similares: Componentes que se propõem a resolver problemas similares;
- Usados em Conjunto: Componentes que são necessários para a utilização deste (pertencentes ao mesmo framework).

3.2.3 – 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 infra-estrutura para a posterior visualização desta documentação.

Para que a documentação de um componente seja criada, o ambiente de

desenvolvimento de software que instanciou o FrameDoc deve solicitar os painéis de documentação de um determinado componente (que foi previamente cadastrado, na etapa de instanciação). Este pedido deve ser repassado para cada categoria associada ao componente e para cada template associado à categoria. Os templates devem disponibilizar 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 a serem disponibilizados são (descritos na figura 2.9):

- 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 a posterior visualização da documentação, o FrameDoc deve permitir a geração de páginas HTML. O pedido do ambiente de desenvolvimento de software instanciador do FrameDoc deve ser processado de forma similar ao pedido de painéis de documentação, sendo que neste caso o campo deve disponibilizar o código HTML que exibe a informação desejada.

Os trechos de código HTML retornados pelos campos formarão a página de documentação de um determinado componente. É também necessário fornecer, na página de documentação, 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 pode ser extraída das propriedades do componente (ex.: uma classe pode ter como documentação implícita os seus atributos, seus métodos e seu código fonte; um diagrama de classes pode ter como

documentação implícita o desenho do seu diagrama, etc.).

A figura 3.5 exibe o diagrama de casos de uso descritos nesta seção.

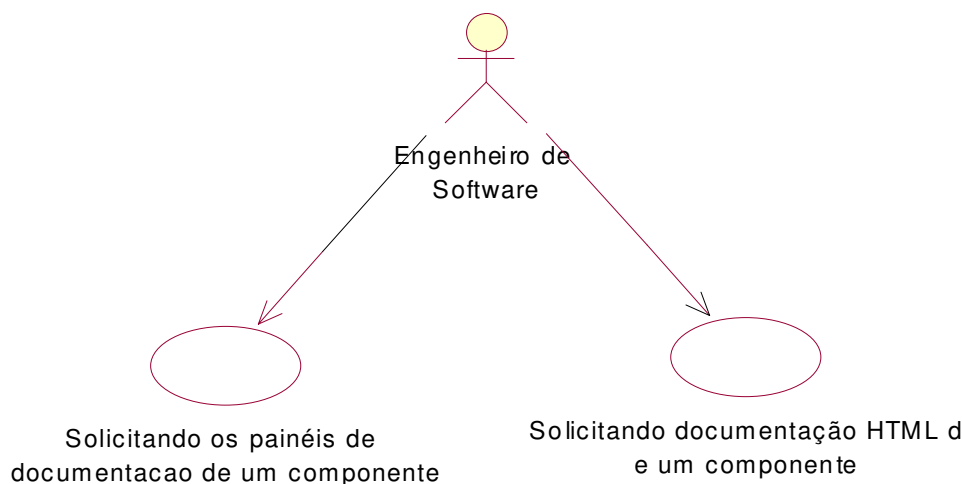


Figura 3.5: Diagrama de casos de uso relacionados com a utilização do FrameDoc

3.2.4 – ETAPA DE VISUALIZAÇÃO DA DOCUMENTAÇÃO

Para a visualização da documentação dos componentes, deve ser utilizado um padrão aberto (suportado por visualizadores gratuitos), como por exemplo: HTML, RTF, PDF, PS, TXT, etc. O padrão HTML foi escolhido por ser amplamente utilizado na Internet e fornecer o suporte necessário para o uso de hiperlinks e multimídia.

O ambiente deve poder, a qualquer momento, gerar uma versão da documentação dos seus componentes. A geração da documentação deve acontecer a nível de componente.

Através de uma lista com todos os componentes, deve ser possível determinar quais irão participar da nova versão de documentação, deixando de exibir externamente documentações incompletas de componentes em construção. Também deve ser possível gerar a documentação de todos os componentes de uma determinada categoria.

Para que o acesso à documentação gerada seja amigável, se torna 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, contém links para a documentação de seus componentes.

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.

O FrameDoc tem como porta de acesso a classe GerenteDocumentacao, que é um singleton¹⁰. Para se obter a instância do gerente, é necessário utilizar o método *getInstancia()*, que tem o seu diagrama de seqüência exibido na figura 4.2. Os demais diagramas de seqüência estão descritos no apêndice A.

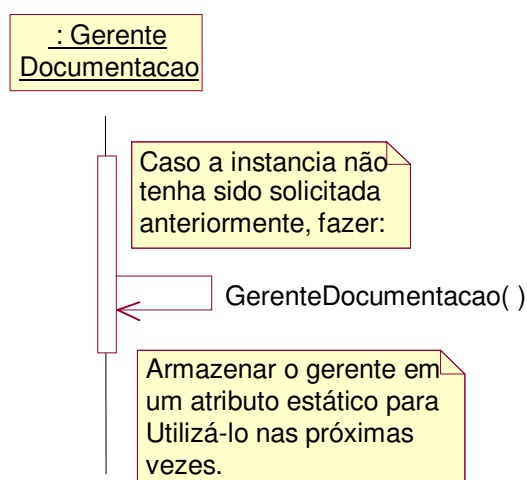


Figura 4.2: Diagrama de seqüência do método *getInstancia()*

4.2 – ETAPA DE INSTANCIAÇÃO

A etapa de instanciação do FrameDoc é responsável por preparar o ambiente de desenvolvimento de software para suportar a documentação de seus componentes. Para que este suporte seja criado, são necessários acessos à geração de documentação e à configuração da documentação do FrameDoc. O capítulo 5 mostra em detalhe a instanciação do FrameDoc no ambiente Odyssey.

4.3 – ETAPA DE DEFINIÇÃO DOS TEMPLATES

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

¹⁰ Padrão de projeto que garante que uma classe terá somente uma instância, e fornece um ponto de acesso global para ela [Gamma; 1995]. Geralmente utilizado quando se necessita de uma única instância de uma classe.

para a documentação.

A figura 4.3 exibe a janela de configuração dos documentos, onde é possível associar documentos com categorias. No exemplo selecionado, o documento “Advogado” está associado a uma categoria “Ator”.

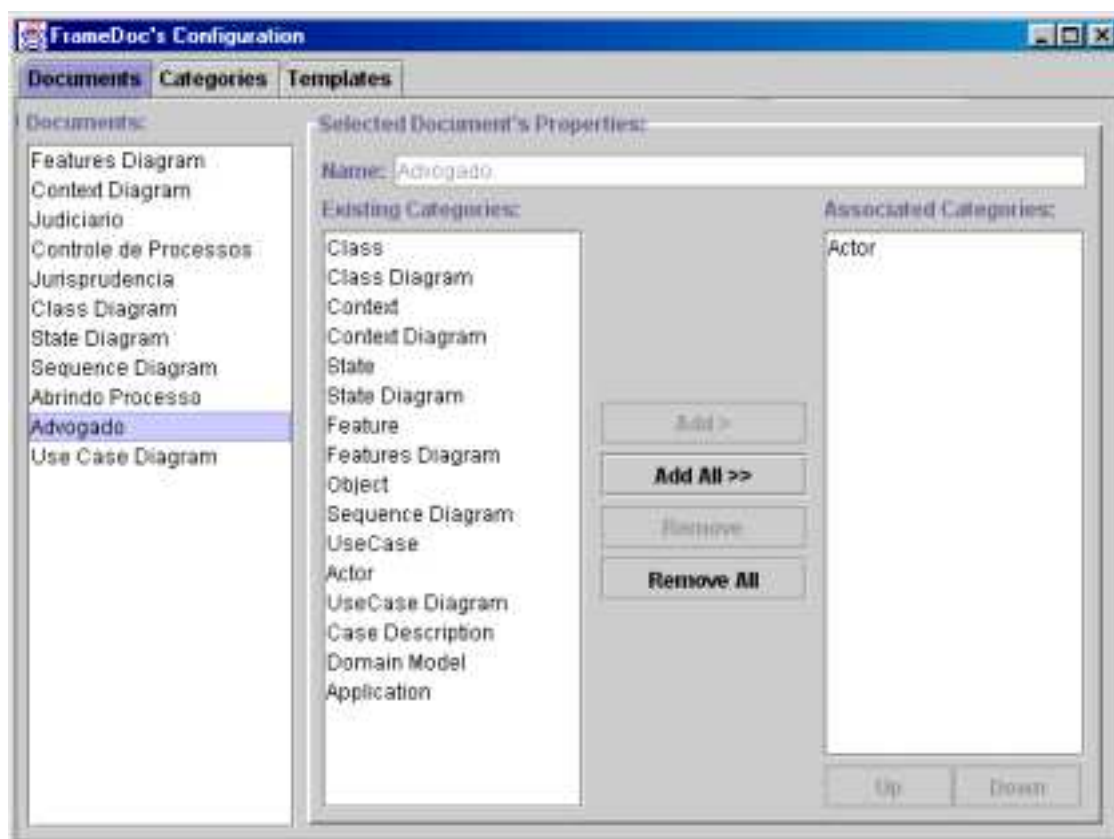


Figura 4.3: Janela de configuração dos documentos

A figura 4.4 exibe a janela de edição de categorias, onde é possível criar, remover e renomear categorias e associar categorias com templates de documentação. No exemplo selecionado, a categoria “Ator” está associada aos templates de “Identificação”, “Contexto” e “Componentes Relacionados”.

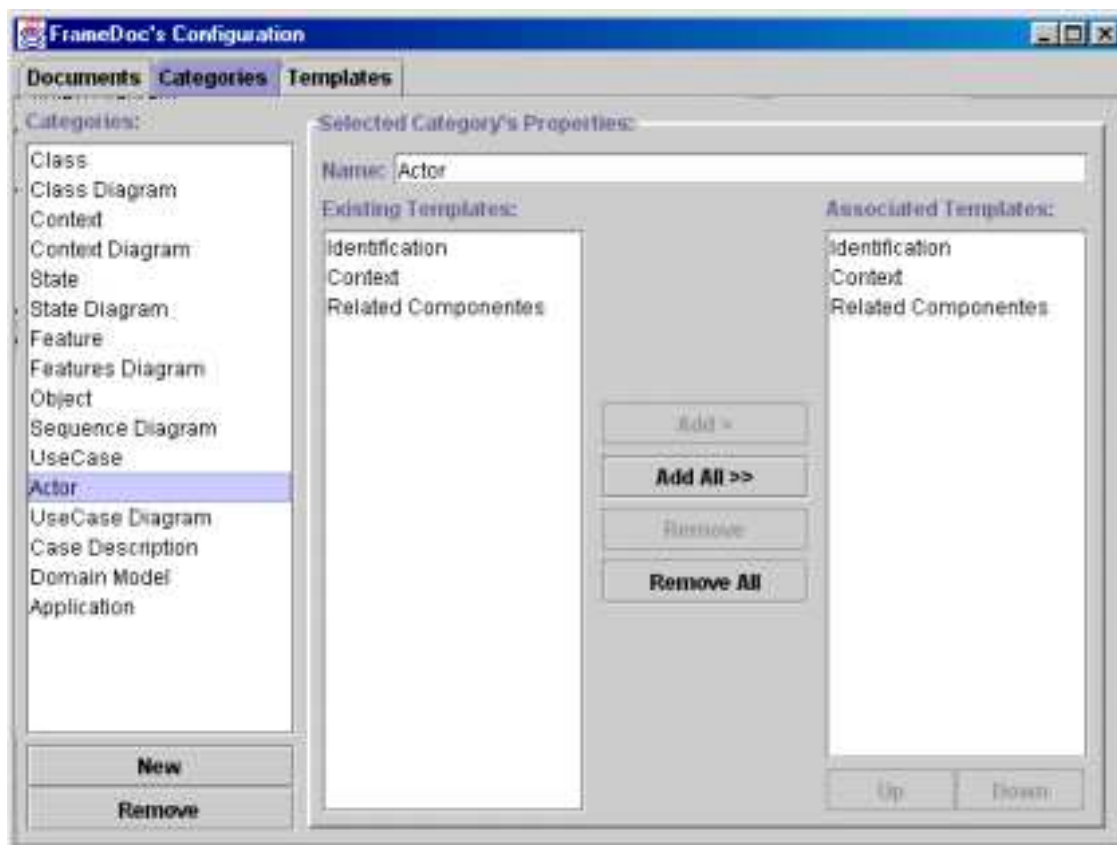


Figura 4.4: Janela de edição de categorias

A figura 4.5 exibe a janela de edição de templates, onde é possível criar, remover e renomear templates. Também é possível adicionar campos aos templates e configurar os campos com um nome e um valor default. 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 mini-editor de HTML para a configuração do seu valor default.

4.4 – ETAPA DE CRIAÇÃO E ATUALIZAÇÃO DA DOCUMENTAÇÃO

A etapa de criação e atualização da documentação fornece ao Engenheiro de Software o suporte para preencher os templates definidos na etapa anterior. A figura 4.6 exibe o preenchimento do template “Contexto” do componente “Advogado”¹¹.

¹¹ Apesar das figuras 4.6 e 4.7 serem referentes ao Ambiente Odyssey, o fornecimento dos painéis de documentação é independente do ambiente em que o FrameDoc está instanciado.

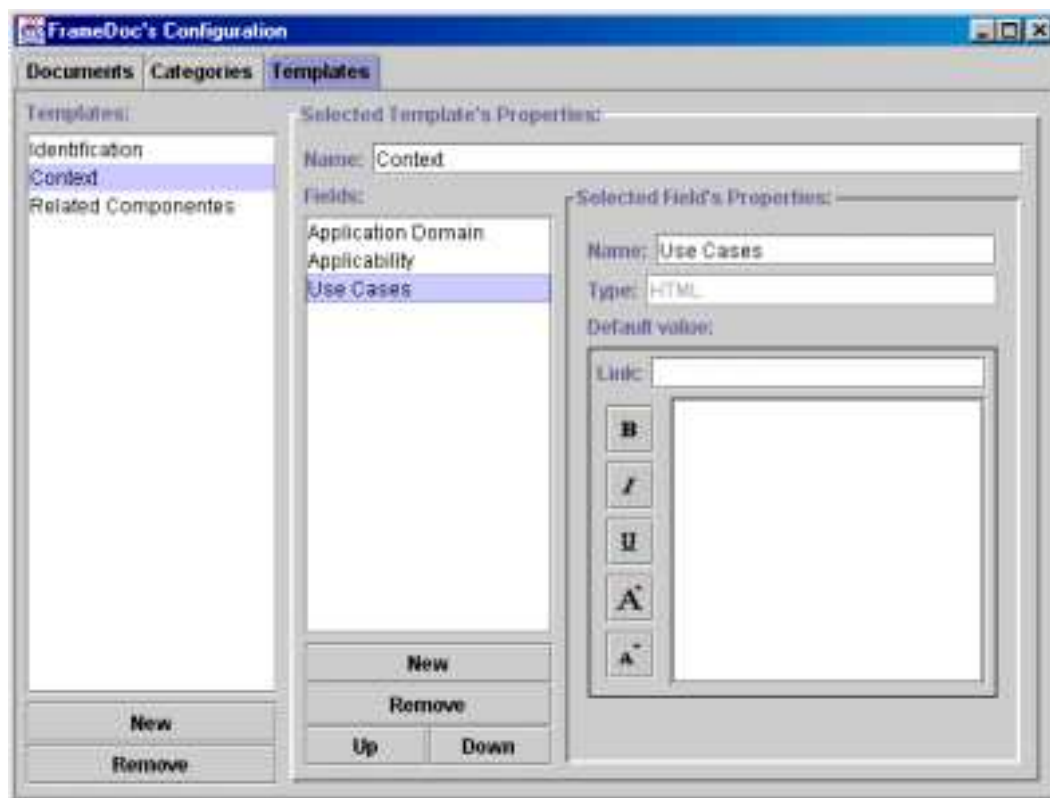


Figura 4.5: Janela de edição dos templates

Para que esta etapa se realize, é necessário que o FrameDoc forneça os painéis de documentação de um componente. No caso do Odyssey, esses painéis compõem as propriedades do componente.

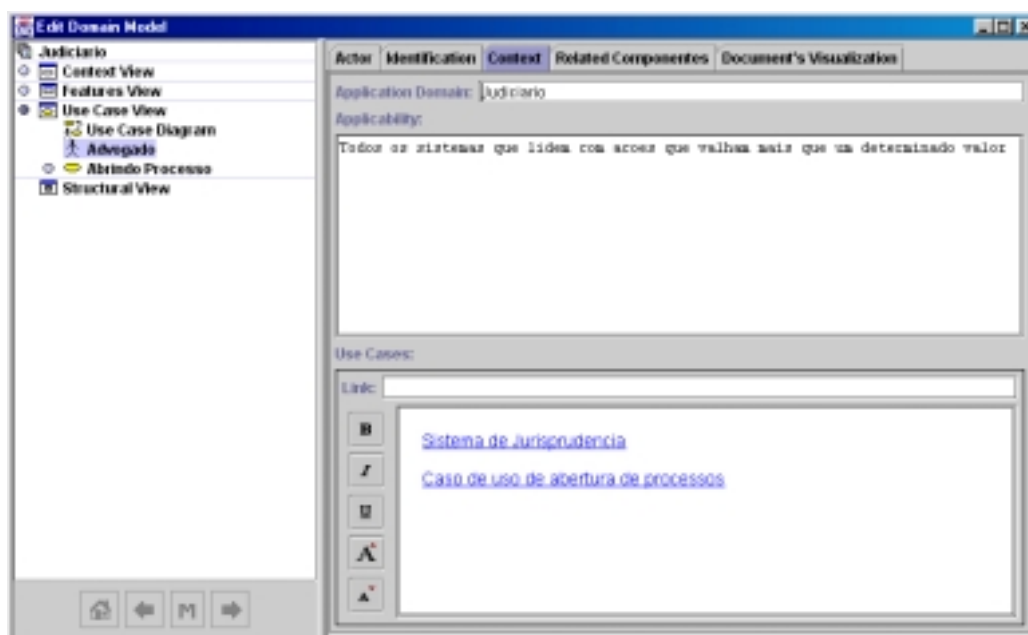


Figura 4.6: Janela de edição da documentação

4.5 – ETAPA DE VISUALIZAÇÃO DA DOCUMENTAÇÃO

A visualização da documentação é feita através da geração da documentação HTML. O formato HTML foi escolhido por ser aberto, ter suporte a multimídia e ser o padrão utilizado para páginas na Internet.

A documentação pode ser visualizada internamente ou externamente ao ambiente de desenvolvimento de software.

Para suportar a visualização da documentação interna ao ambiente, foi criado um visualizador de HTML, que é fornecido com os demais painéis de documentação, como exibido na figura 4.7.

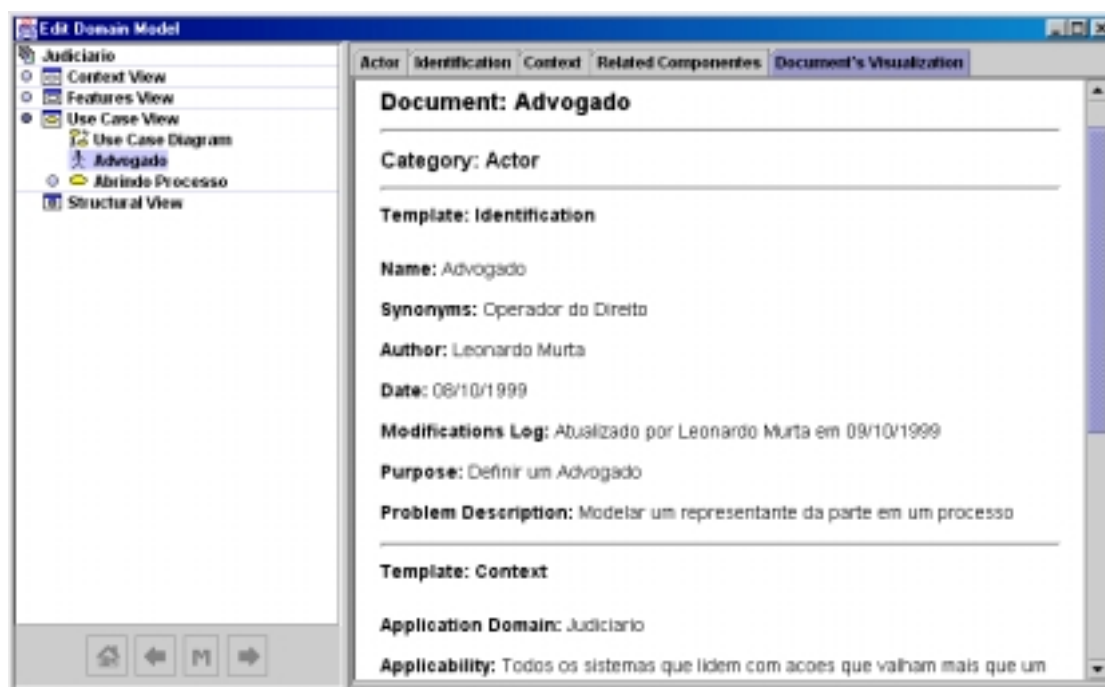


Figura 4.7: Janela de visualização da documentação

Para fornecer suporte à visualização da documentação externa ao ambiente, foi criado um gerador de páginas de HTML, que cria as páginas dos documentos e páginas de índices para ajudar na busca dos componentes. Este gerador de HTML está exposto na figura 4.8.

Após a geração da documentação, esta poderá ser visualizada por qualquer browser e exibida em sites na Internet.

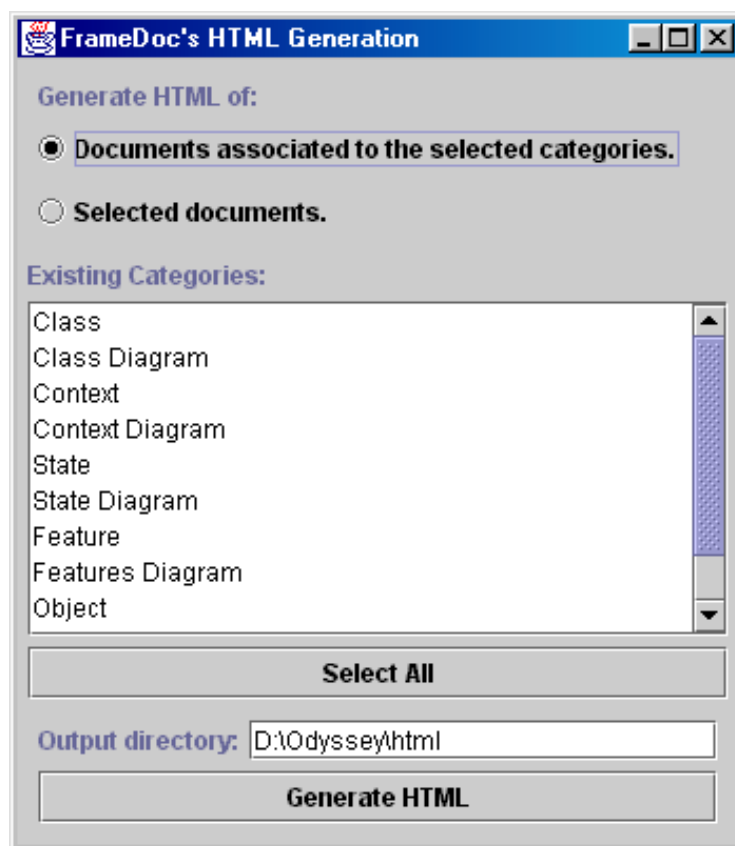


Figura 4.8: Janela de geração da documentação

As informações geradas nas páginas HTML são obtidas dos campos de documentação do FrameDoc. Um campo do FrameDoc pode exibir informações em dois formatos, de acordo com seus métodos abstratos *getPainelEdicao()* e *getDocumentacaoHTML()*. Uma chamada ao método *getPainelEdição()* fornece o painel de edição da informação que o campo representa, de modo que esta possa ser apresentada em uma janela do ambiente. Uma chamada ao método *getDocumentacaoHTML()* fornece uma *String* com o código HTML da informação representada pelo campo.

A figura 4.9.a exibe o painel gerado pelo campo *texto*. Este painel é composto por um título e um campo de edição de texto de somente uma linha. A figura 4.9.b exibe o HTML gerado.

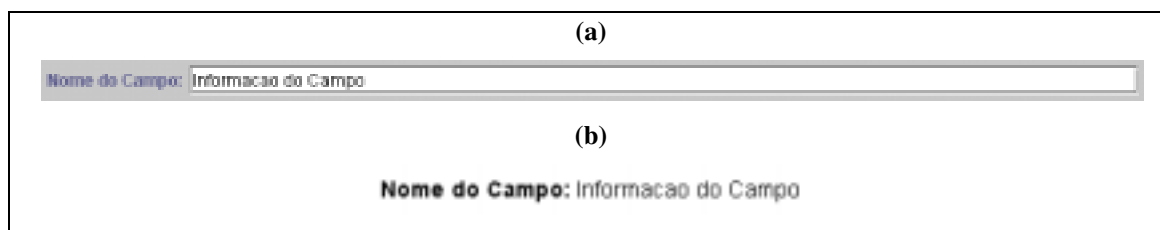


Figura 4.9: Exibição de informações por um campo texto

A figura 4.10.a exibe o painel gerado pelo campo *memo*. Este painel é composto por um título e um campo de edição de texto de múltiplas linhas. A figura 4.10.b exibe o HTML gerado.

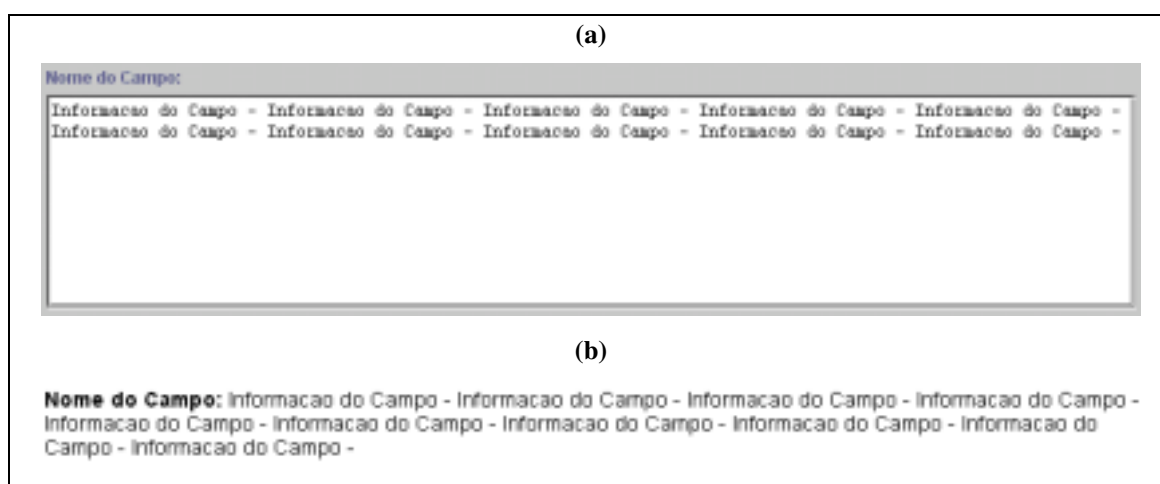


Figura 4.10: Exibição de informações por um campo memo

A figura 4.11.a exibe o painel gerado pelo campo *Check Box*. Este painel é composto por um título e uma lista de itens, que podem ser selecionados. Este campo permite a seleção de múltiplos itens. A figura 4.11.b exibe o HTML gerado.

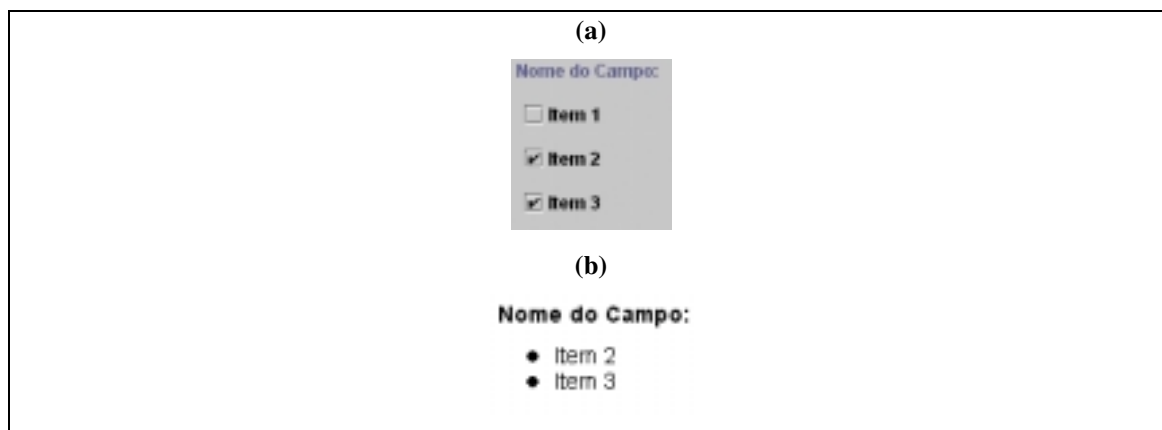


Figura 4.11: Exibição de informações por um campo check box

A figura 4.12.a exibe o painel gerado pelo campo *Radio Button*. Este painel é composto por um título e uma lista de itens, onde somente um item pode estar selecionado em um instante de tempo. A figura 4.12.b exibe o HTML gerado.

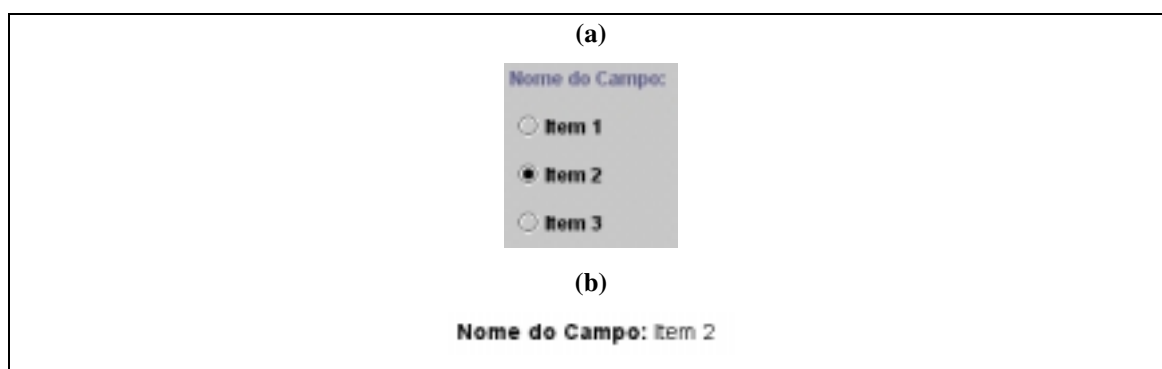


Figura 4.12: Exibição de informações por um campo radio button

A figura 4.13.a exibe o painel gerado pelo campo *Som*. Este painel é composto por um título e um campo de texto, onde deve ser fornecido o endereço do arquivo de som. A figura 4.13.b exibe o HTML gerado, que contém um link para tocar o som.

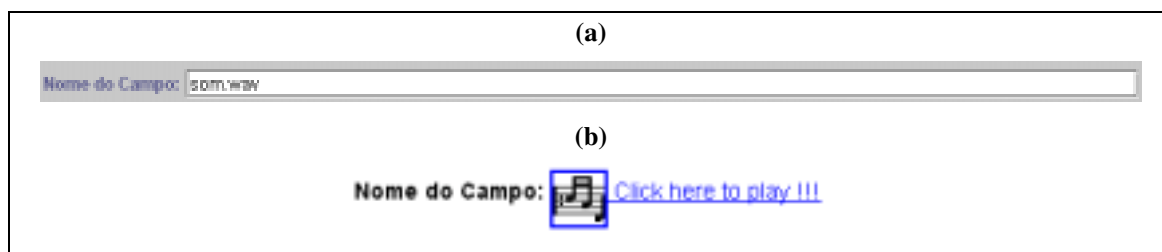


Figura 4.13: Exibição de informações por um campo som

A figura 4.14.a exibe o painel gerado pelo campo *Imagem*. Este painel é

composto por um título e um campo de texto, onde deve ser fornecido o endereço do arquivo de imagem. A figura 4.14.b exibe o HTML gerado.

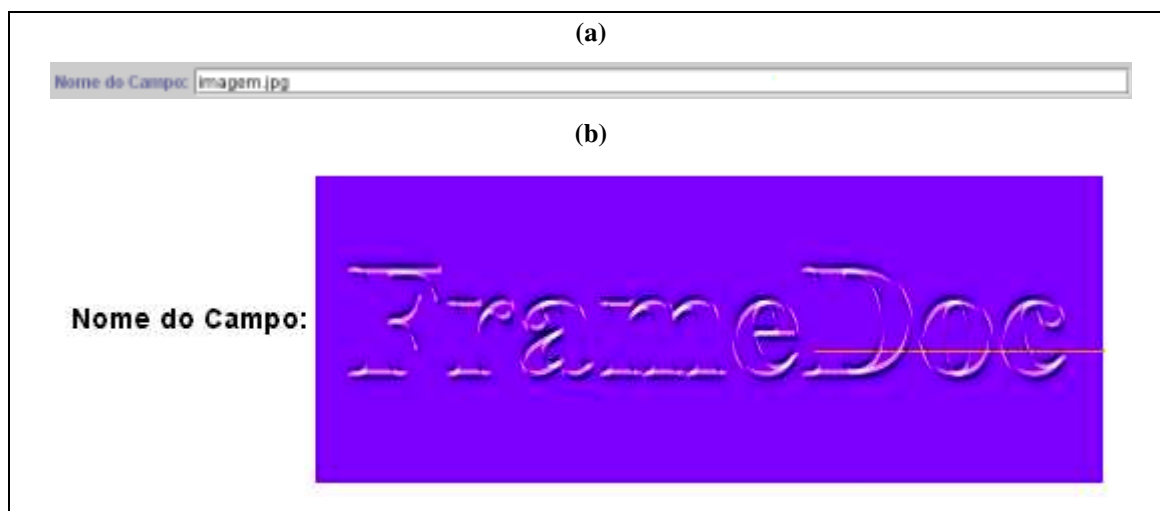


Figura 4.14: Exibição de informações por um campo imagem

A figura 4.15.a exibe o painel gerado pelo campo *Vídeo*. Este painel é composto por um título e um campo de texto, onde deve ser fornecido o endereço do arquivo de vídeo. A figura 4.15.b exibe o HTML gerado, que contém um link para exibir o vídeo.

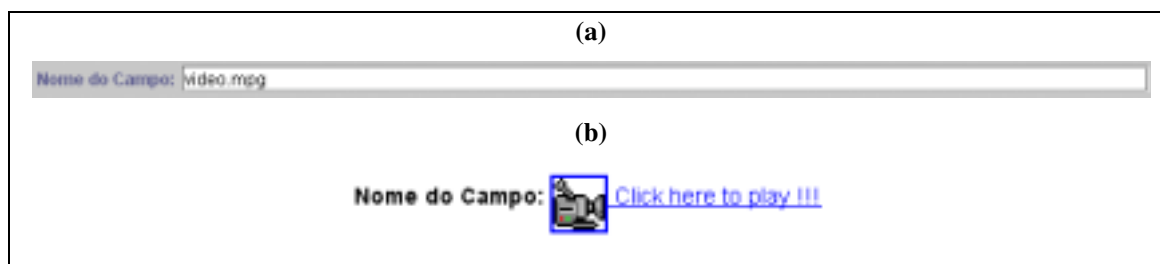


Figura 4.15: Exibição de informações por um campo vídeo

A figura 4.16.a exibe o painel gerado pelo campo *HTML*. Este painel é composto por um título e um campo de edição de HTML, com opções de formatação de texto para negrito, itálico, sublinhado, fonte grande, fonte pequena e criação de links para outros documentos ou para a Internet. A figura 4.16.b exibe o HTML gerado.

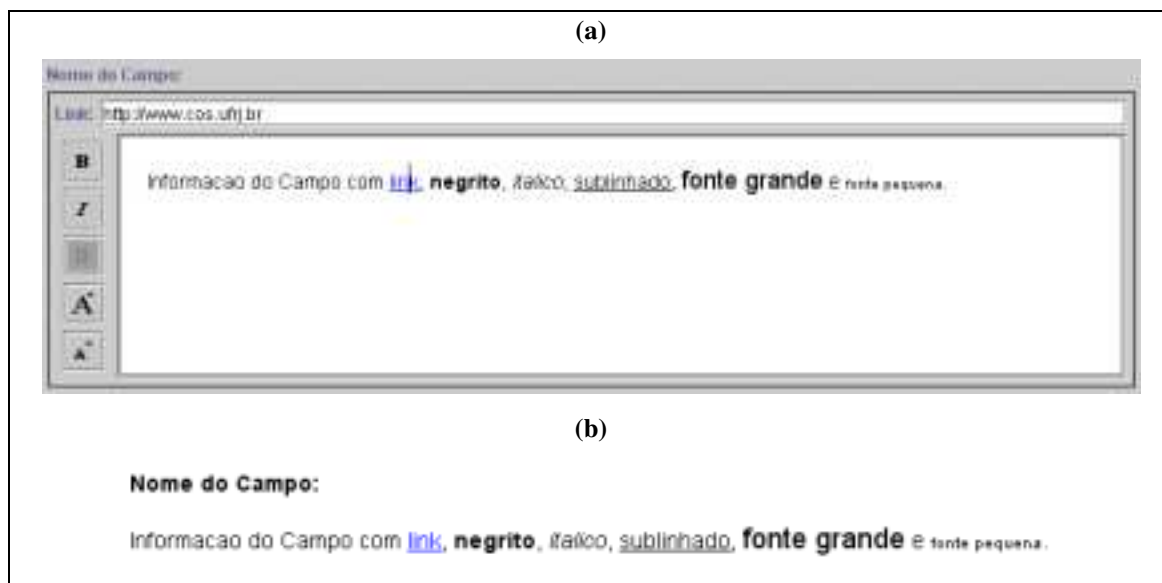


Figura 4.16: Exibição de informações por um campo HTML

No apêndice A estão descritos os diagramas de seqüência necessários para a definição de todos os casos de uso descritos neste capítulo.

5 – INSTANCIAÇÃO DO FRAMEDOC NO AMBIENTE ODYSSEY

Para validar o FrameDoc foi utilizado o Ambiente Odyssey. Este capítulo descreve, no nível de abstração de projeto e implementação, como o FrameDoc foi instanciado no Ambiente Odyssey, e como é a sua utilização dentro do ambiente.

5.1 – AMBIENTE ODYSSEY

O Ambiente Odyssey é uma infra-estrutura de reutilização baseada em modelos de domínio, que se propõe a apoiar vários domínios de aplicação. A fase de povoamento de informações (genéricas) do domínio é vista como desenvolvimento para reutilização (Engenharia de Domínio), e cria o suporte necessário para o desenvolvimento com reutilização (Engenharia de Aplicação), que é a adaptação dessas informações para uma determinada aplicação dentro do domínio.

Essas informações (tanto do domínio quanto da aplicação) são armazenadas no Odyssey através de diferentes diagramas (seguindo o padrão UML). No contexto do ambiente, um componente é qualquer modelo ou elemento de modelo em qualquer nível de abstração. Por exemplo: diagramas, elementos de um diagrama (classes, casos de uso, atores,...), uma aplicação, um domínio de aplicação, etc.

Além do trabalho aqui apresentado, diversos outros trabalhos estão sendo desenvolvidos no contexto do Projeto Odyssey. Estes trabalhos estão relacionados com aspectos como navegação inteligente sobre os modelos do domínio, de forma a capturar as necessidades dos reutilizadores e sugerir melhores caminhos de busca [Braga; 1999a], ferramenta para apoio à gerência de riscos [Barros; 1999], ferramenta de suporte à criação e adaptação de arquiteturas específicas de domínio [Xavier; 1999] e mediador inteligente para acesso a bases distribuídas [Braga; 1999b].

5.2 – ETAPAS PARA A INSTANCIAÇÃO

Para que a instanciação do FrameDoc no Odyssey aconteça, é necessário

implementar as seguintes etapas:

- Persistência do gerente de documentação;
- Acesso às janelas de configuração e geração de documentos;
- Cadastramento das categorias;
- Cadastramento dos componentes;
- Acesso aos painéis de um componente.

5.2.1 – PERSISTÊNCIA DO GERENTE DE DOCUMENTAÇÃO

A versão atual do Odyssey implementa a persistência do gerente de documentação via serialização. Versões futuras também utilizarão o Gerente de Objetos Armazenados GOA++ [Mauro; 1997] (fornecendo também uma opção para a busca, via OQL).

No armazenamento via serialização, todos os objetos têm que ser guardados no mesmo arquivo para que não percam as suas referências. Para que o FrameDoc fosse armazenado junto com os diagramas, foi criada uma tabela *hash* (que funciona como a raiz de persistência do ambiente), onde são inseridos os domínios do Odyssey e o gerente de documentação do FrameDoc. A figura 5.1 exibe o trecho de código da serialização e a figura 5.2 exibe o trecho de código da desserialização. A variável *dados* é uma tabela hash, *NOME_ARQUIVO* é o nome do arquivo que contém a tabela hash, *GERENTE_DOC* é chave para o gerente de documentação na tabela hash e *LISTA_DOMINIOS* é chave para a lista de domínios na tabela hash.

```
dados.put (GERENTE_DOC, GerenteDocumentacao.getInstancia ());  
dados.put (LISTA_DOMINIOS, this);  
FileOutputStream fileOut = new FileOutputStream (NOME_ARQUIVO);  
ObjectOutputStream out = new ObjectOutputStream (fileOut);  
out.writeObject (dados);  
out.close ();
```

Figura 5.1 : Trecho de código de serialização do gerente de documentação

```

FileInputStream fileIn = new FileInputStream(NOME_ARQUIVO);
ObjectInputStream in = new ObjectInputStream(fileIn);
dados = (Hashtable) in.readObject ();
in.close ();
GerenteDocumentacao.setInstancia(
    (GerenteDocumentacao)dados.get(GERENTE_DOC));
return ((ListaDominios)dados.get(LISTA_DOMINIOS));

```

Figura 5.2 : Trecho de código de desserialização do gerente de documentação

5.2.2 - ACESSO ÀS JANELAS DE CONFIGURAÇÃO E GERAÇÃO

Para que o Odyssey exiba as janelas de configuração da documentação e geração da documentação é necessário criar um item de menu para elas e cadastrar o tratamento do clique no item. A figura 5.3 exibe o trecho de código que executa essa tarefa.

```

private JMenuItem itemFrameDocGeneration =
    new JMenuItem("Documents Generation");
itemFrameDocGeneration.addActionListener(new
    MainWindow_FrameDocGeneration_actionAdapter(this));

private JMenuItem itemFrameDocConfig =
    new JMenuItem("Documents Configuration");
itemFrameDocConfig.addActionListener(new
    MainWindow_FrameDocConfig_actionAdapter(this));

```

Figura 5.3 : Trecho de código de criação do item de menu para as janelas de configuração e geração da documentação

Quando for clicado o item de menu, deve ser efetuada a abertura da respectiva janela. A figura 5.4 exibe o trecho de código responsável pela abertura da janela de configuração, e a figura 5.5, da janela de geração.

```

GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();
JFrame janela = gerente.getJanelaConfiguracao ();
gerente.centralizaJanela(janela);
janela.show ();

```

Figura 5.4 : Trecho de código responsável pela abertura da janela de configuração

```

GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();
JFrame janela = gerente.getJanelaGeracao ();
gerente.centralizaJanela(janela);
janela.show ();

```

Figura 5.5 : Trecho de código responsável pela abertura da janela de geração

As figuras 5.6 e 5.7 exibem os acessos à geração de documentação do FrameDoc e à configuração da documentação do FrameDoc no Odyssey.

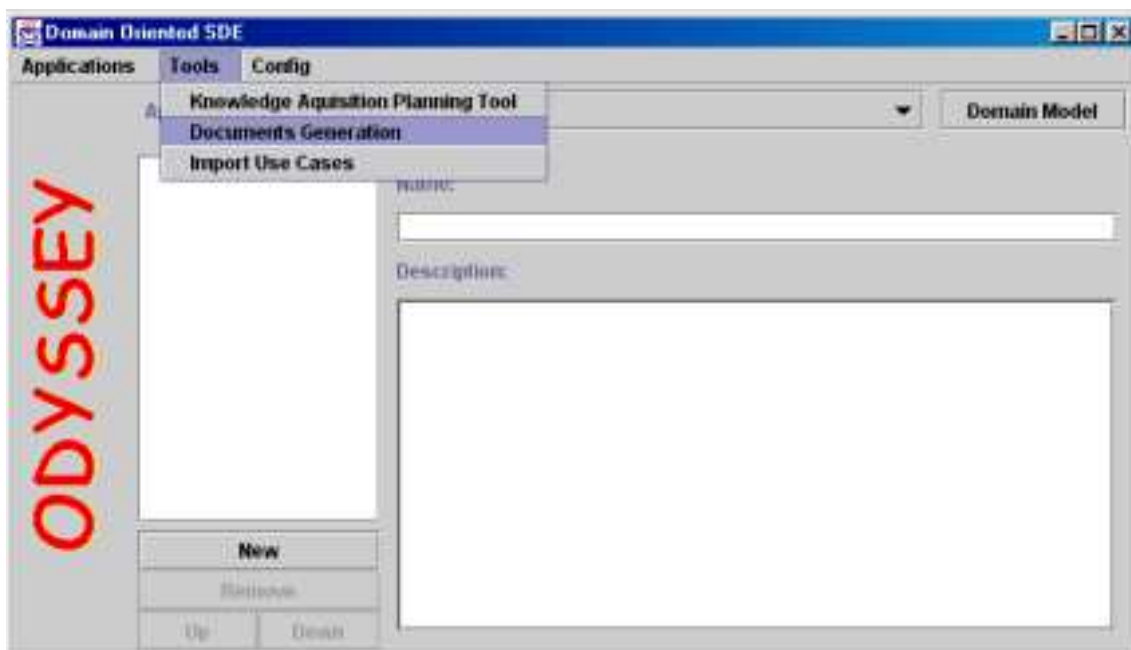


Figura 5.6: Acesso à geração da documentação do FrameDoc no Odyssey

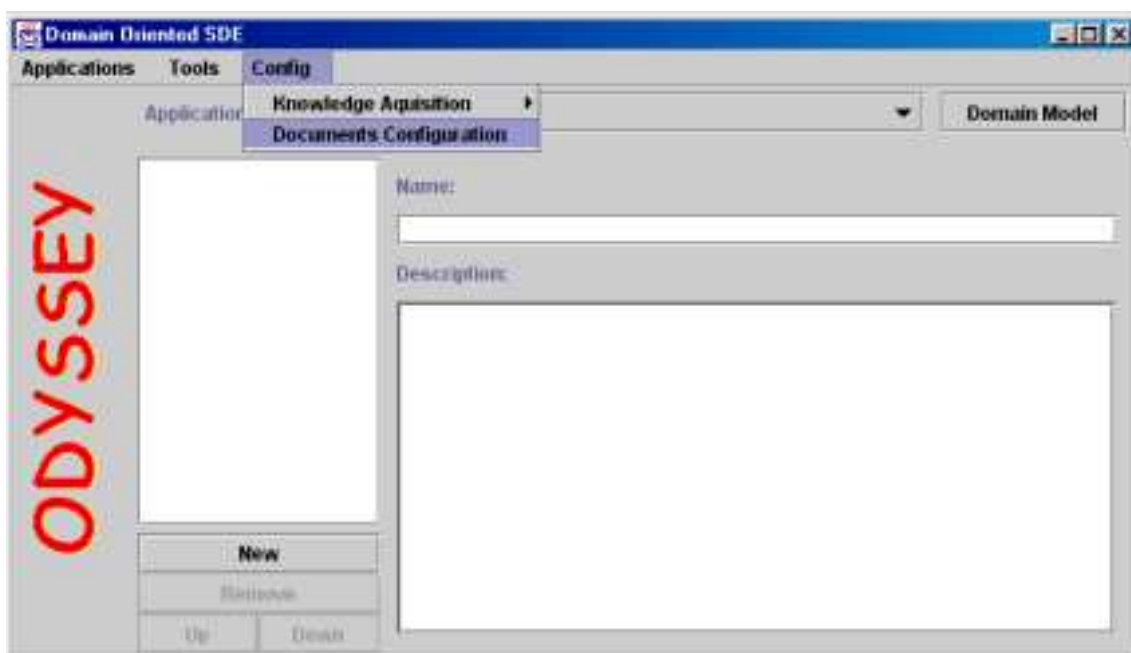


Figura 5.7: Acesso à configuração da documentação do FrameDoc no Odyssey

5.2.3 – CADASTRAMENTO DAS CATEGORIAS

Quando o gerente de documentação é construído pela primeira vez, é necessário povoá-lo com as categorias suportadas pelo Odyssey. A figura 5.8 exibe o trecho de

código responsável por esse povoamento.

```
GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();
gerente.addCategoria("Class");
gerente.addCategoria("Class Diagram");
gerente.addCategoria("Context");
gerente.addCategoria("Context Diagram");
gerente.addCategoria("State");
gerente.addCategoria("State Diagram");
gerente.addCategoria("Feature");
gerente.addCategoria("Features Diagram");
gerente.addCategoria("Object");
gerente.addCategoria("Sequence Diagram");
gerente.addCategoria("UseCase");
gerente.addCategoria("Actor");
gerente.addCategoria("UseCase Diagram");
gerente.addCategoria("Case Description");
gerente.addCategoria("Domain Model");
gerente.addCategoria("Application");
```

Figura 5.8 : Trecho de código responsável pelo cadastramento das categorias

Conforme visto anteriormente, as figuras 4.3 e 4.4 exibem, respectivamente, as janelas de configuração dos documentos e das categorias.

5.2.4 – CADASTRAMENTO DOS COMPONENTES

Para que um componente do Odyssey possa ser documentado pelo FrameDoc é necessário que o componente implemente a interface “Documentavel”, que se cadastre no FrameDoc quando for construído e se remova do FrameDoc quando for destruído. A figura 5.9 exhibe o trecho de código inserido no construtor de um componente do tipo *classe*, e a figura 5.10 exhibe o trecho de código para a remoção de um componente do FrameDoc.

```
GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();
gerente.addComponente(this);
gerente.associaComponenteCategoria(this, "Class");
```

Figura 5.9 : Trecho de código responsável pelo cadastramento de um componente do tipo *classe*

```
GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();
gerente.removeComponente(componente);
```

Figura 5.10 : Trecho de código responsável pela remoção de um componente

5.2.5 – ACESSO AOS PAINÉIS DE UM COMPONENTE

Sempre quando for necessário editar a documentação de um componente, deve ser efetuada uma solicitação dos painéis de documentação ao FrameDoc. A figura 5.11 exibe o trecho de código responsável por preparar os painéis de documentação de um documento para exibição dentro de um TabPanel.

```
GerenteDocumentacao gerente = GerenteDocumentacao.getInstancia();  
JPanel[] paineis = gerente.getPaineisComponente(item);  
for (int i = 0; i < paineis.length; i++)  
    tbPanel.addTab(paineis[i].toString(), new JScrollPane(paineis[i]));
```

Figura 5.11 : Trecho de código responsável pela solicitação dos painéis de documentação de um componente

Conforme visto no capítulo anterior, a figura 4.6 exibe a edição dos painéis de documentação no Odyssey.

6 – CONCLUSÕES

6.1 – CONTRIBUIÇÕES

Na seção 2.2 foi efetuada uma comparação entre as ferramentas de documentação estudadas, o que identificava algumas deficiências nas ferramentas. A figura 6.1 fornece o comparativo das ferramentas incluindo o FrameDoc.

Critério\Ferramenta	JavaDoc	Rose 98	Fast Case	MEMPHIS	FrameDoc
Exportação da documentação em formato portátil	✓	±	✓		✓
Utilização de hiperlinks na documentação	✓			±	✓
Criação e configuração de padrões de documentação				✓	✓
Geração de documentação modular	✓		✓	✓	✓
Utilização de multimídia na documentação				✓	±
Visualização da documentação antes da geração			✓		✓
Geração da documentação implícita ao componente	✓	✓	✓		✓

Figura 6.1: Comparativo entre ferramentas de documentação

O FrameDoc cumpre os quesitos abordados na figura 6.1 de acordo com as seguintes características:

- Exportação a sua documentação utilizando o padrão HTML;
- Utilização de hiperlinks 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”.

6.2 – LIMITAÇÕES

A versão atual do FrameDoc fornece três campos multimídia, que são som, vídeo e imagem, mas eles permitem somente a manutenção da referência ao arquivo da mídia relacionada, não fornecendo suporte interno nem para vídeo nem para som. Para que esse suporte fosse implementado, seria necessária a utilização do Java Mídia Framework, fornecido pela SUN, o que tornaria o FrameDoc (e consequentemente o Odyssey) maior e mais lento.

6.3 – TRABALHOS FUTUROS

O FrameDoc 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 não existência desses campos, é possível emulá-los na atual versão com os campos de memo (no lugar de lista de texto) e html (no lugar de lista de referências e referência).

Atualmente o FrameDoc, ao gerar a documentação HTML de um componente, gera também uma página de índice para essa documentação. Apesar de ajudar, a página de índice para a documentação não soluciona o problema da localização do componente desejado para a reutilização. Uma possível solução seria a construção de um agente de busca. A figura 6.2 representa essa possível solução, onde o agente deveria, a partir de

um conjunto de regras, encontrar fatos (mapeados nos campos dos templates e armazenados em uma base de conhecimento) que juntos com uma consulta fornecida pelo usuário, satisfaçam essas regras ao serem submetidas a uma máquina de inferência.

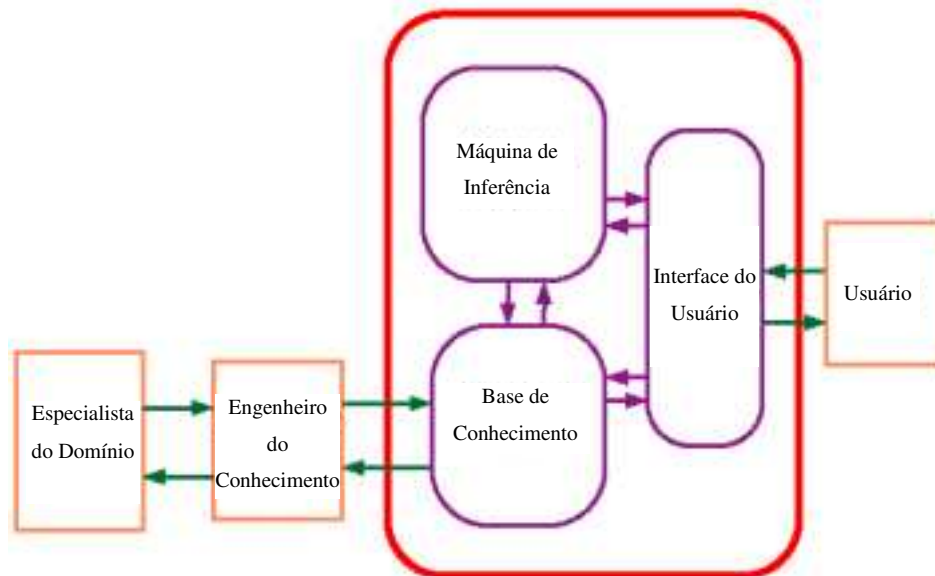


Figura 6.2: Arquitetura de um sistema baseado em conhecimento [Silva; 1999].

REFERÊNCIAS BIBLIOGRÁFICAS

- [Barros; 1999] **Barros, M.O.; Werner, C.M.L.; Travassos, G.H.**; Risk Analysis: a Key Success Factor for Complex System Development; 12th International Conference on Software & System Engineering and their Applications; Paris, 1999 (aceito para publicação)
- [Borges; 1998] **Borges, M.**; Notas de aula do curso de Tópicos Especiais em Sistemas de Informação, DCC/UFRJ, 1998
- [Braga; 1999a] **Braga, R.M.M.; Werner, C.M.L.; Mattoso, M.L.Q.**; An Agent System for Domain Information Discovery and Filtering; submetido ao 6th International Conference on Software Reuse, 1999
- [Braga; 1999b] **Braga, R.M.M.; Mattoso, M.L.Q.; Werner, C.M.L.**; The Use of Mediators for Component Retrieval in a Reuse Environment; Component-Based Software Engineering Processes Workshop, TOOLS 30; Santa Barbara, EUA, 1999
- [Fowler; 1997] **Fowler, M.; Scott, K.**; UML Distilled – Applying the Standard Object Modeling Language; Addison-Wesley, 1997
- [Gamma; 1995] **Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.**; Design Patterns – Elements of Reusable Object-Oriented Software; Addison-Wesley, 1995
- [Mauro; 1997] **Mauro, R.C.; Mattoso, M.L.Q.**; GOA++: Tecnologia, implementação e extensões aos serviços de gerência de objetos; XII Simpósio Brasileiro de Banco de Dados, SBC, Fortaleza, 1997
- [Moore; 1991] **Moore, J.M.; Bailin, S.C.**; Domain Analysis: Framework for reuse; Domain Analysis and Software System Modeling, 1^a ed., Califórnia, IEEE Computer Society Press Tutorial, pp. 179-202, 1991
- [Murta; 1998b] **Murta, L.G.P.; Borges, M.**; Trabalho do curso de Tópicos Especiais em Sistemas de Informação, DCC/UFRJ, 1998
- [Murta; 1998a] **Murta, L.G.P.; Barros, M.O.; Werner, C.M.L.**; Café da Manhã com Java; Relatório Técnico Odyssey-03/98, COPPE/UFRJ, 1998
- [Rational, 1998] **Rational**; página na Internet em: <http://www.rational.com>
- [Silva; 1998] **Silva, M.F.**; Documentação de Componentes Reutilizáveis: uma abordagem baseada nas tecnologias de Hipermídia e Padrões; Tese de Mestrado,

COPPE/UFRJ, 1998

[Silva; 1999] **Silva J.C.P.**; Notas de aula do curso de Tópicos Especiais em Inteligência Artificial, DCC/UFRJ, 1999

[Silveira; 1999] **Silveira, D.**; FAST CASE - Uma Ferramenta CASE para o Desenvolvimento Visual de Sistemas Orientados a Objetos; Tese de Mestrado, IM/NCE/UFRJ, 1999

[Sun; 1999] **Sun Microsystems**; página na Internet em:

<http://java.sun.com/products/jdk/javadoc/index.html>

[Tracz; 1994] **Tracz, W.**; Software Reuse Myths Revisited; 16th International Conference on Software Engineering, Sorrento, Itália, 1994

[Werner; 1996] **Werner, C.M.L.; Travassos, G.H.; Rocha, A.R.; Werneck, V.**; Memphis: um Ambiente para Desenvolvimento de Software baseado em Reutilização: Definição da Arquitetura; Relatório Técnico Memphis 3/96, COPPE/UFRJ, 1996

[Werner; 1999a] **Werner, C.M.L.**; Notas de aula da disciplina Tópicos Especiais em Engenharia de Software IV; COPPE/UFRJ, 2o. período, 1999

[Werner; 1999b] **Werner, C.M.L.; Mattoso, M.L.Q.; Braga, R.M.M.; Barros, M.O.; Murta, L.G.P.; Dantas, A.R.**; Odyssey: Infra-estrutura de Reutilização baseada em Modelos de Domínio; XIII Simpósio Brasileiro de Engenharia de Software, Caderno de Ferramentas, Florianópolis, 1999

[Wirfs-Brock; 1990] **Wirfs-Brock, R.J.; Johnson, R.E.**; Surveying Current Research in Object-Oriented Design; Communications of the ACM, 33 (9), 1990

[Xavier; 1999] **Xavier, J.R.**; Uma Ferramenta de Apoio à Instanciação de Arquiteturas de Domínio; Workshop de Teses em Engenharia de Software, SBES99; Florianópolis, 1999

APÊNDICE A – DIAGRAMAS DE SEQUÊNCIA

A.1 – ETAPA DE INSTANCIACÃO

A etapa de instanciação do FrameDoc é composta pelos casos de uso *Adicionando um componente*, *Adicionando uma Categoria*, *Associando um componente a uma categoria* e *Removendo um componente*.

O caso de uso *Adicionando um componente* é mapeado para o método *addComponente(Documentavel componente)*, que tem seu diagrama de sequência exibido na figura A.1.

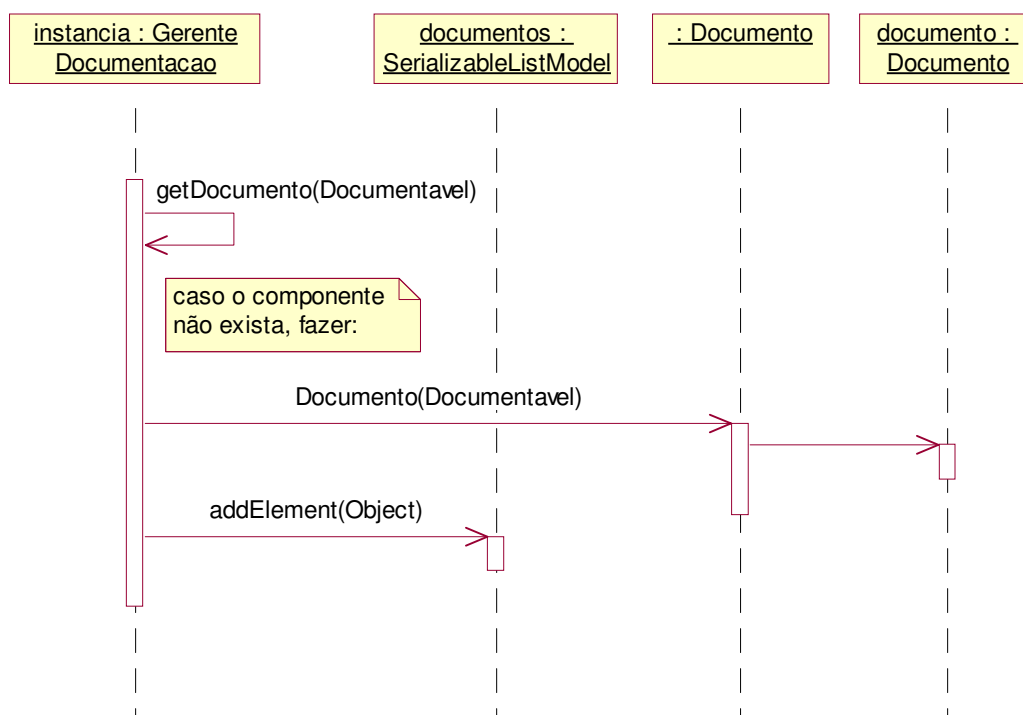


Figura A.1: Diagrama de sequência do método *addComponente(Documentavel componente)*

O método *addComponente(Documentavel componente)* faz uso do método *getDocumento(Documentavel componente)*, que tem o seu diagrama de sequência exibido na figura A.2.

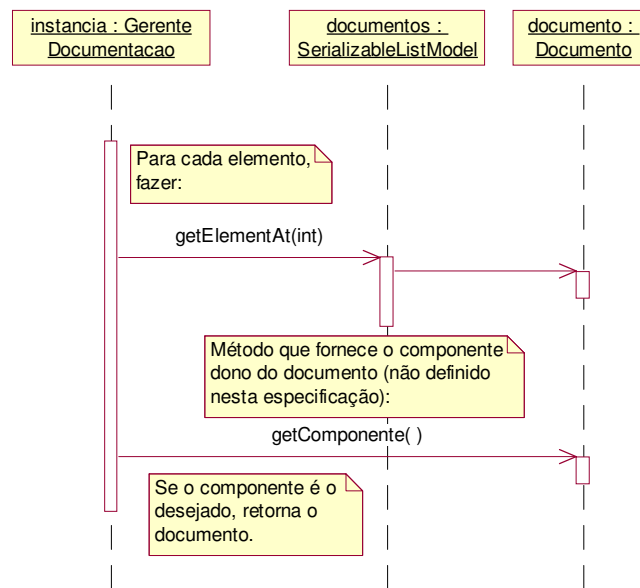


Figura A.2: Diagrama de seqüência do método *getDocumento(Documentavel componente)*

O caso de uso *Adicionando uma categoria* é mapeado para o método *addCategoria(String nome)*, que tem seu diagrama de seqüência exibido na figura A.3.

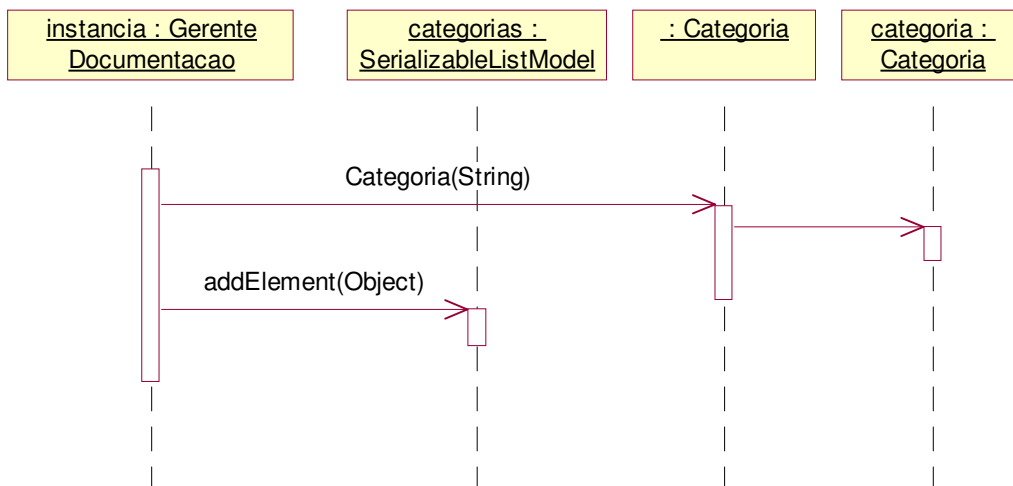


Figura A.3: Diagrama de seqüência do método *addCategoria(String nome)*

O caso de uso *Associando um componente a uma categoria* é mapeado para o método *associaComponenteCategoria(Documentavel componente, String nomeCategoria)*, que tem seu diagrama de seqüência exibido na figura A.4.

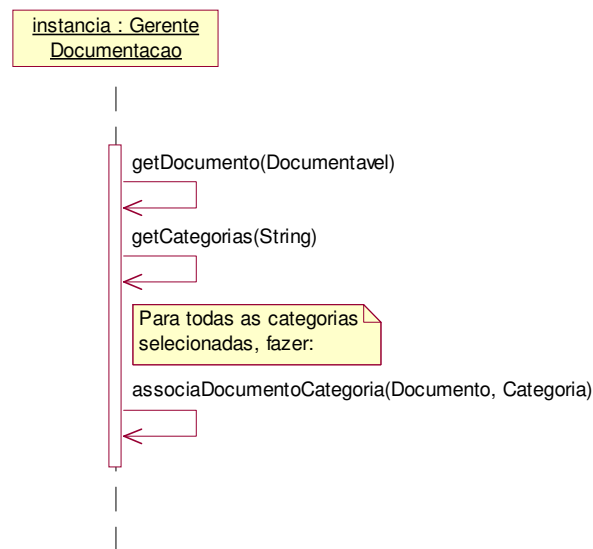


Figura A.4: Diagrama de seqüência do método *associaComponenteCategoria(Documentavel componente, String nomeCategoria)*

O método *associaComponenteCategoria(Documentavel componente, String nomeCategoria)* faz uso dos métodos *getDocumento(Documentavel componente)*, que tem o seu diagrama de seqüência exibido na figura A.2, *getCategorias(String nome)*, que tem o seu diagrama de seqüência exibido na figura A.5 e *associaDocumentoCategoria(Documento documento, Categoria categoria)*, que tem o seu diagrama de seqüência exibido na figura A.6 .

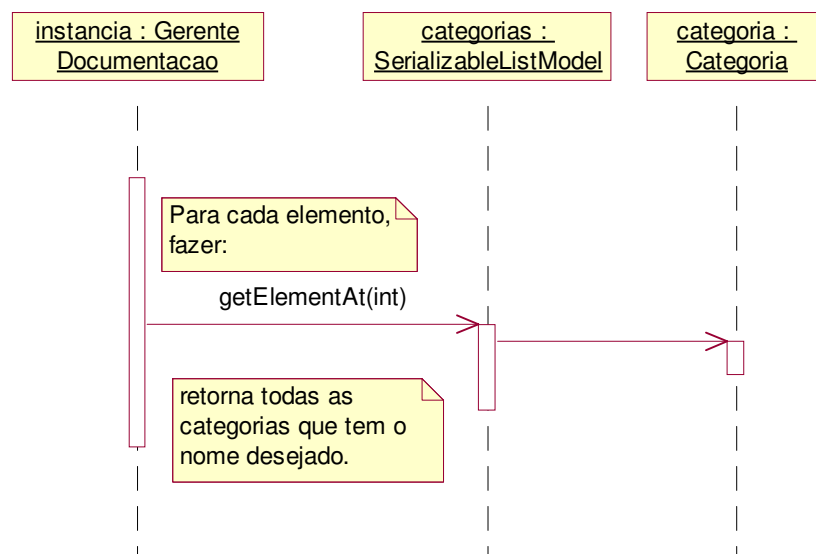


Figura A.5: Diagrama de seqüência do método *getCategoria(String nome)*

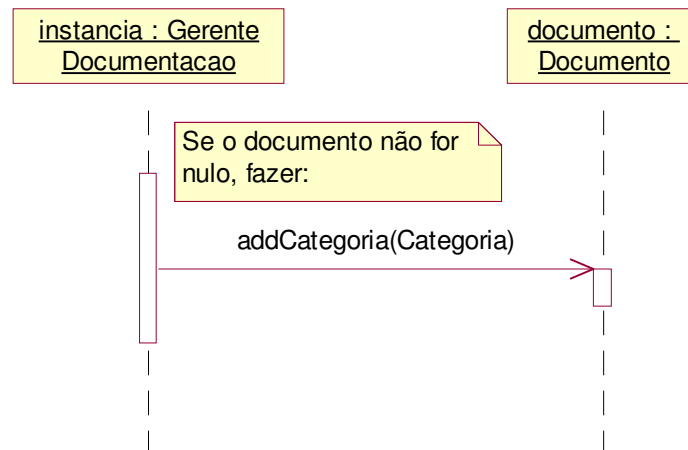


Figura A.6: Diagrama de seqüência do método *associaDocumentoCategoria(Documento documento, Categoria categoria)*

O método *associaDocumentoCategoria(Documento documento, Categoria categoria)* faz uso do método *addCategoria(Categoria categoria)*, que tem o seu diagrama de seqüência exibido na figura A.7.

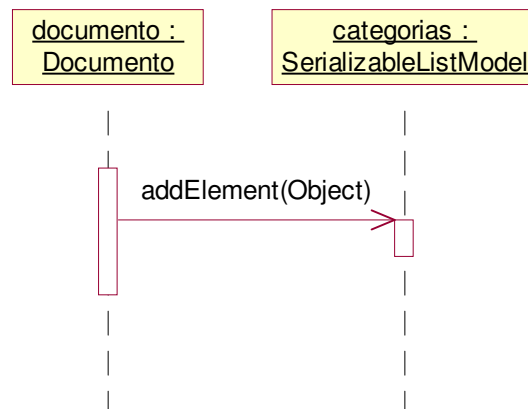


Figura A.7: Diagrama de seqüência do método *addCategoria(Categoria categoria)*

O caso de uso *Removendo um componente* é mapeado para o método *removeComponente(Documentavel componente)*, que tem seu diagrama de seqüência exibido na figura A.8.

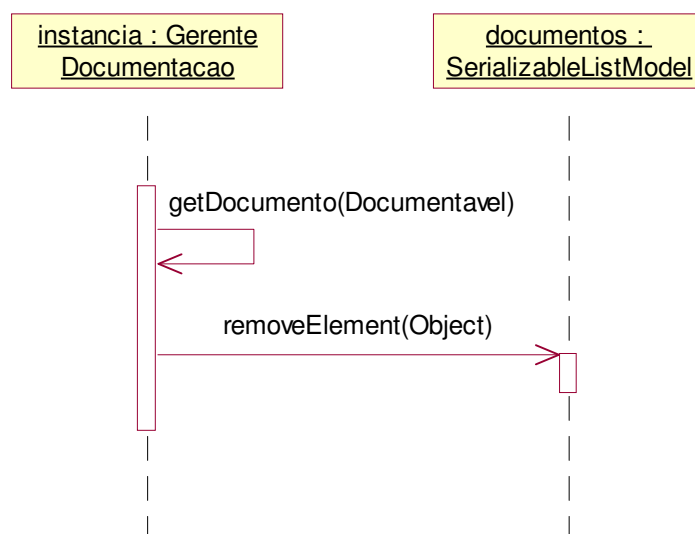


Figura A.8: Diagrama de seqüência do método *removeComponente(Documentavel componente)*

O método *removeComponente(Documentavel componente)* faz uso do método *getDocumento(Documentavel componente)*, que tem o seu diagrama de seqüência exibido na figura A.2.

A.2 – ETAPA DE DEFINIÇÃO DOS TEMPLATES

A etapa de definição dos templates é composta pelos casos de uso *Adicionando uma Categoria*, *Adicionando um template*, *Associando um componente a uma categoria*, *Associando um template a uma categoria*, *Adicionando um campo a um template*, *Removendo uma categoria* e *Removendo um template*.

Os casos de uso *Adicionando uma Categoria* e *Associando um componente a uma categoria* já foram definidos na etapa de instanciação do FrameDoc (Seção A.1).

O caso de uso *Adicionando um template* é mapeado para o método *addTemplate(String nomeTemplate)*, que tem seu diagrama de seqüência exibido na figura A.9.

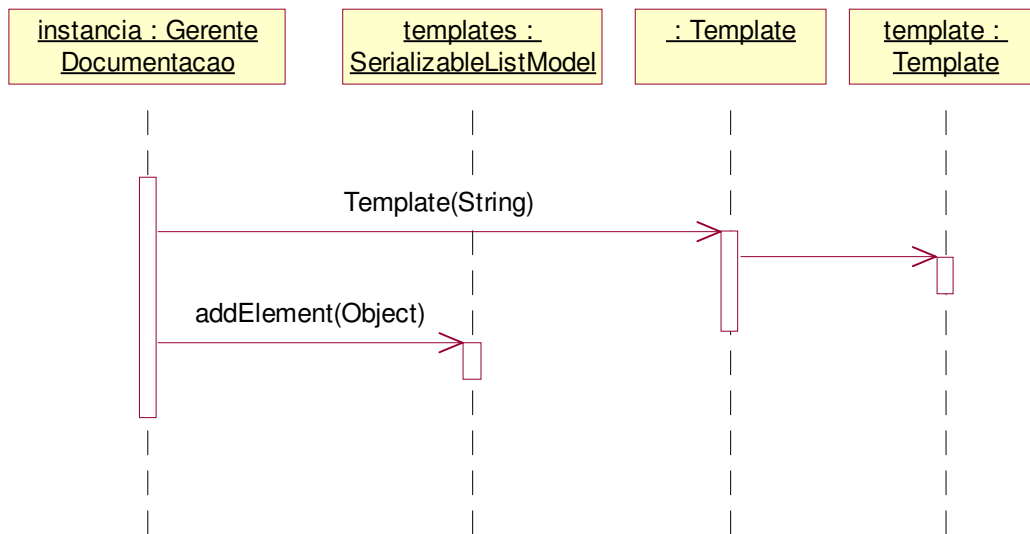


Figura A.9: Diagrama de seqüência do método *addTemplate(String nomeTemplate)*

O caso de uso *Associando um template a uma categoria* é mapeado para o método *associaTemplateCategoria(String nomeTemplate, String nomeCategoria)*, que tem seu diagrama de seqüência exibido na figura A.10.

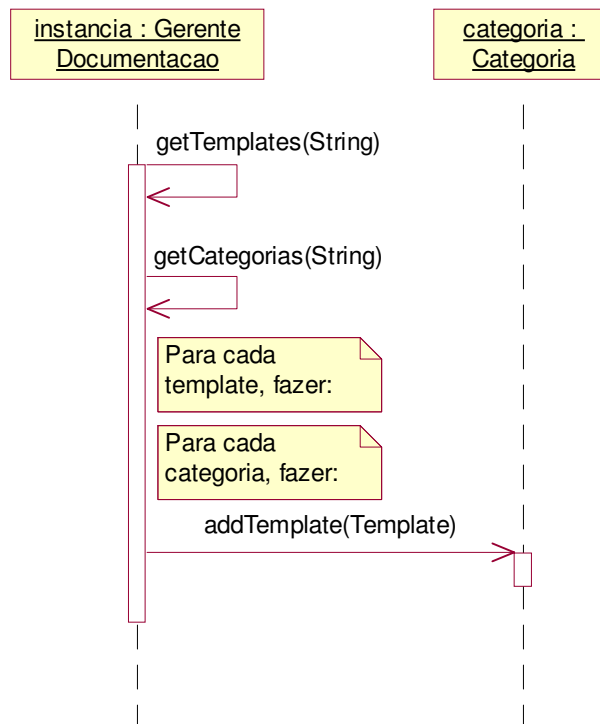


Figura A.10: Diagrama de seqüência do método *associaTemplateCategoria(String nomeTemplate, String nomeCategoria)*

O método *associaTemplateCategoria(String nomeTemplate, String nomeCategoria)* faz uso dos métodos *getTemplates(String nome)*, que tem o seu diagrama de seqüência exibido na figura A.11, *getCategorias(String nome)*, que tem o seu diagrama de seqüência exibido na figura A.5 e *addTemplate(Template template)*, que tem o seu diagrama de seqüência exibido na figura A.12.

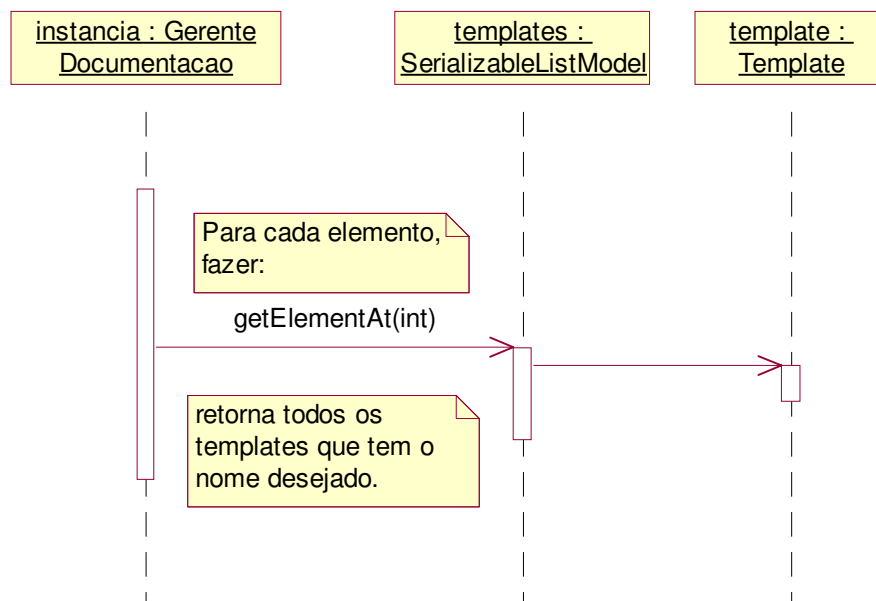


Figura A.11: Diagrama de seqüência do método *getTemplates(String nome)*

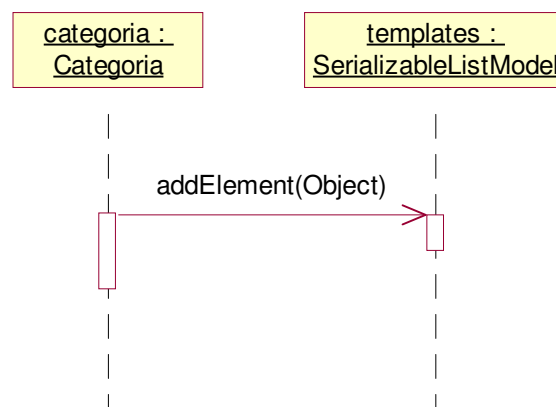


Figura A.12: Diagrama de seqüência do método *addTemplate(Template template)*

O caso de uso *Adicionando um campo a um template* é mapeado para o método *addCampo(int tipoCampo, String nomeCampo, String nomeTemplate)*, que tem seu

diagrama de seqüência exibido na figura A.13.

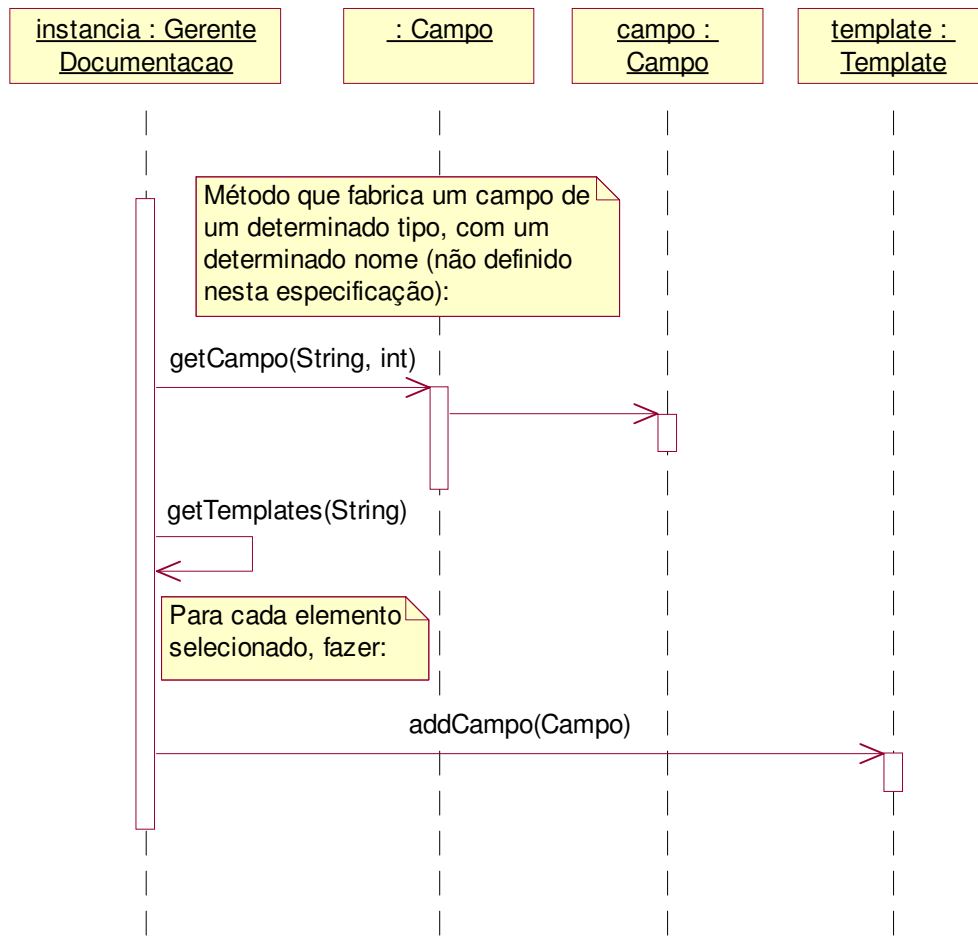


Figura A.13: Diagrama de seqüência do método *addCampo(int tipoCampo, String nomeCampo, String nomeTemplate)*

O método *addCampo(int tipoCampo, String nomeCampo, String nomeTemplate)* faz uso dos métodos *getTemplates(String nome)*, que tem o seu diagrama de seqüência exibido na figura A.11 e *addCampo(Campo campo)*, que tem o seu diagrama de seqüência exibido na figura A.14.

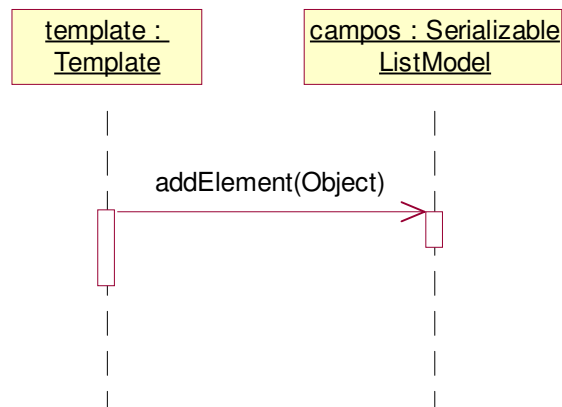


Figura A.14: Diagrama de seqüência do método *addCampo(Campo campo)*

O caso de uso *Removendo uma categoria* é mapeado para o método *removeCategoria(Categoria categoria)*, que tem seu diagrama de seqüência exibido na figura A.15.

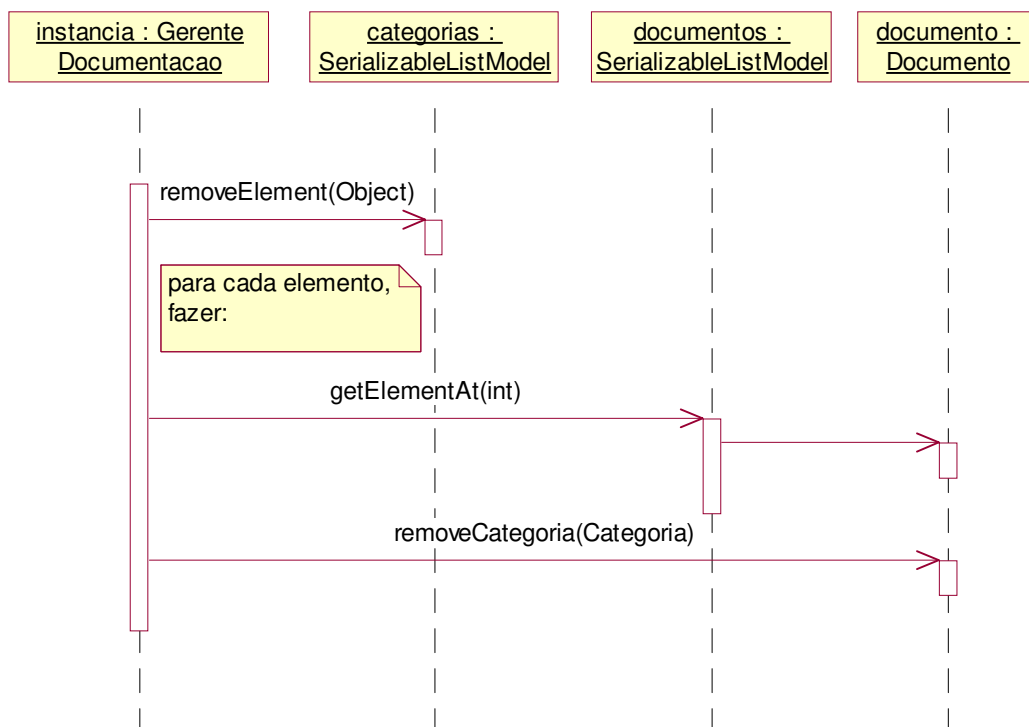


Figura A.15: Diagrama de seqüência do método *removeCategoria(Categoria categoria)*

O método *removeCategoria(Categoria categoria)* faz uso do método *removeCategoria(Categoria categoria)*, que tem o seu diagrama de seqüência exibido

na figura A.16.

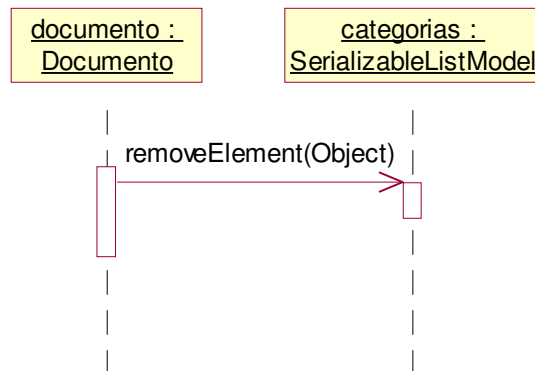


Figura A.16: Diagrama de seqüência do método *removeCategoria(Categoria categoria)*

O caso de uso *Removendo um template* é mapeado para o método *removeTemplate(Template template)*, que tem seu diagrama de seqüência exibido na figura A.17.

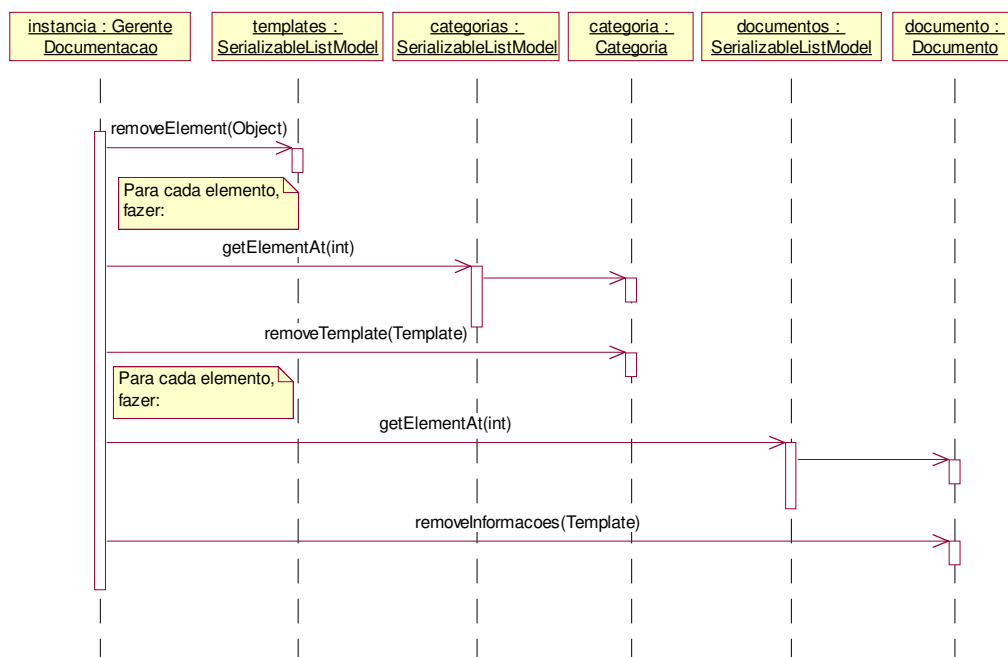


Figura A.17: Diagrama de seqüência do método *removeTemplate(Template template)*

O método *removeTemplate(Template template)* faz uso dos métodos *removeTemplate(Template template)*, que tem o seu diagrama de seqüência exibido na

figura A.18 e *removeInformacoes(Template template)*, que tem o seu diagrama de seqüência exibido na figura A.19.

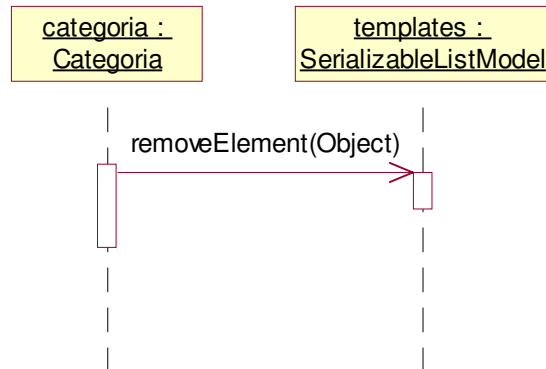


Figura A.18: Diagrama de seqüência do método *removeTemplate(Template template)*

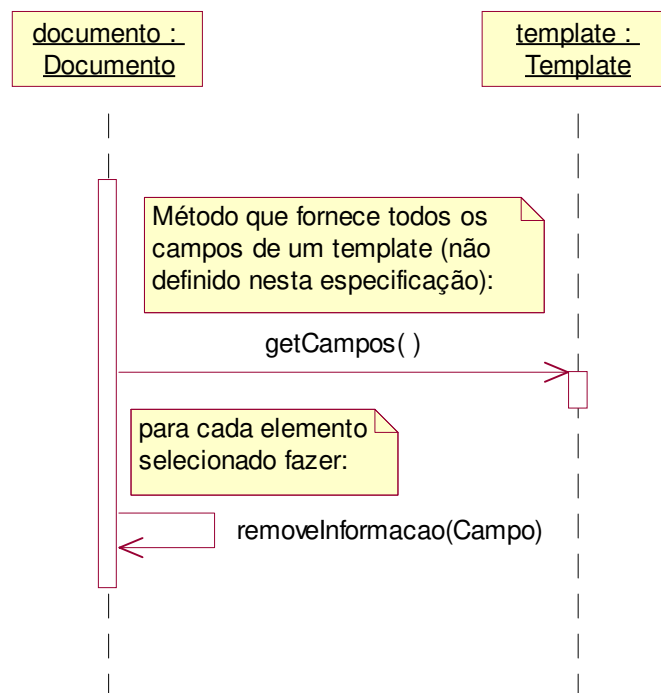


Figura A.19: Diagrama de seqüência do método *removeInformacoes(Template template)*

O método *removeInformacoes(Template template)* faz uso do método *removeInformacao(Campo)*, que tem o seu diagrama de seqüência exibido na figura A.20.

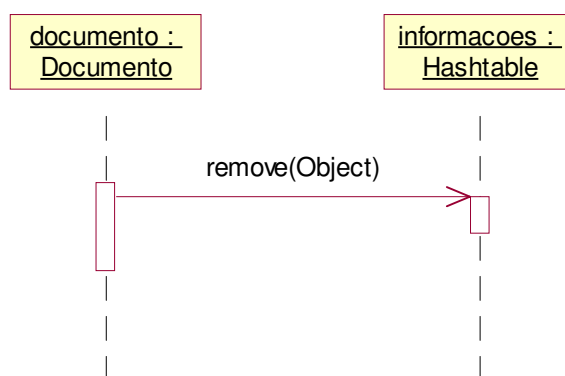


Figura A.20: Diagrama de seqüência do método *removeInformacao(Campo campo)*

A.3 – ETAPA DE CRIAÇÃO E ATUALIZAÇÃO DA DOCUMENTAÇÃO

A etapa de criação e atualização da documentação é composta pelos casos de uso *Solicitando os painéis de documentação de um componente* e *Solicitando documentação HTML de um componente*.

O caso de uso *Solicitando os painéis de documentação de um componente* é mapeado para o método *getPaineisComponente(Documentavel componente)*, que tem seu diagrama de seqüência exibido na figura A.21.

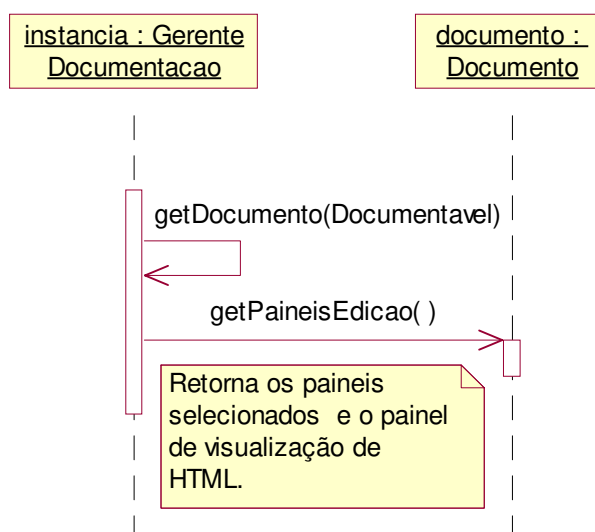


Figura A.21: Diagrama de seqüência do método *getPaineisComponente(Documentavel componente)*

O método *getPaineisComponente(Documentavel componente)* faz uso dos métodos *getDocumento(Documentavel documentavel)*, que tem o seu diagrama de seqüência exibido na figura A.2 e *getPaineisEdicao()*, que tem o seu diagrama de seqüência exibido na figura A.22.

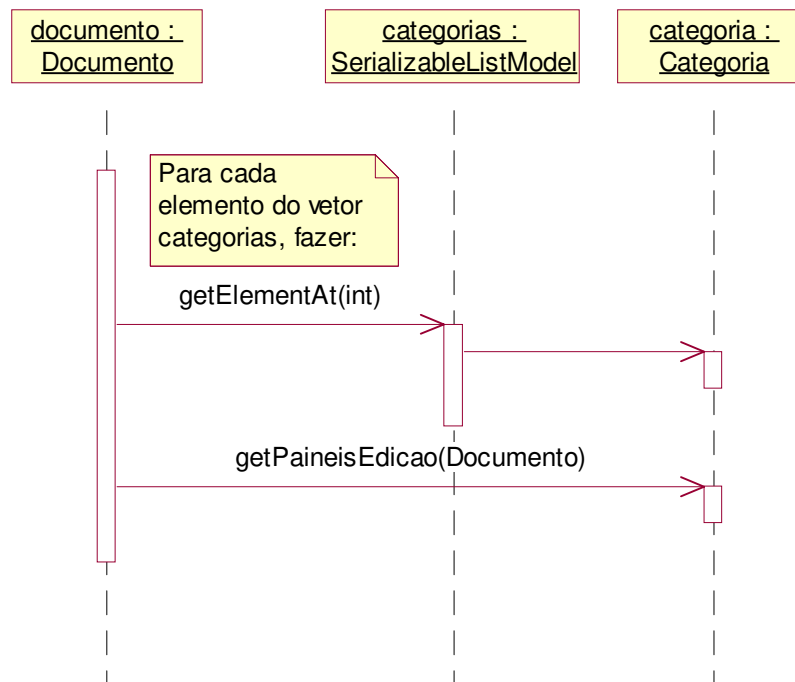


Figura A.22: Diagrama de seqüência do método *getPaineisEdicao()*

O método *getPaineisEdicao()* faz uso do método *getPaineisEdicao(Documento documento)*, que tem o seu diagrama de seqüência exibido na figura A.23.

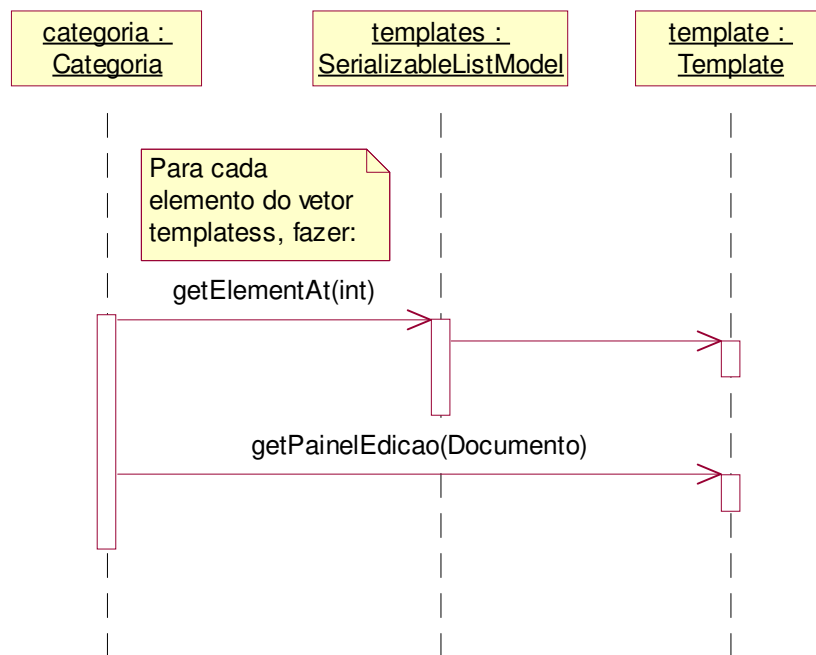


Figura A.23: Diagrama de seqüência do método *getPaineisEdicao(Documento documento)*

O método *getPaineisEdicao()* faz uso do método *getPainelEdicao(Documento documento)*, que tem o seu diagrama de seqüência exibido na figura A.24.

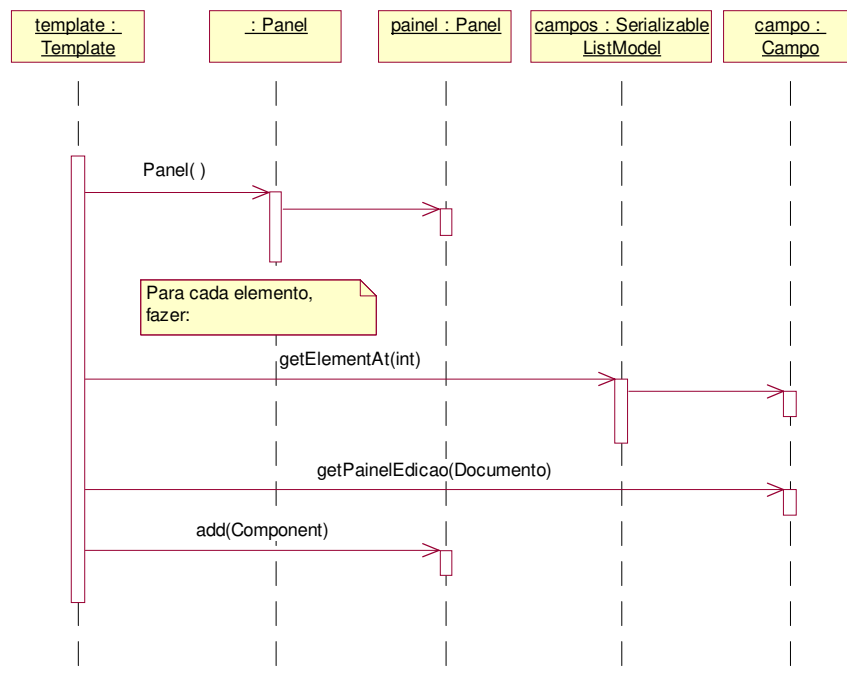


Figura A.24: Diagrama de seqüência do método *getPainelEdicao(Documento documento)*

O método *getPainelEdicao(Documento documento)* faz uso do método *getPainelEdicao(Documento documento)*, que tem o seu diagrama de seqüência exibido na figura A.25.

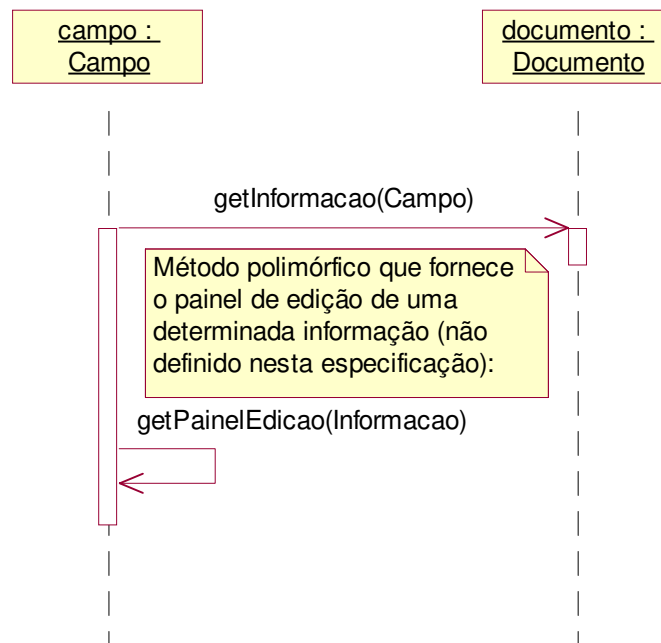


Figura A.25: Diagrama de seqüência do método *getPainelEdicao(Documento documento)*

O método *getPainelEdicao(Documento documento)* faz uso do método *getInformacao(Campo campo)*, que tem o seu diagrama de seqüência exibido na figura A.26.

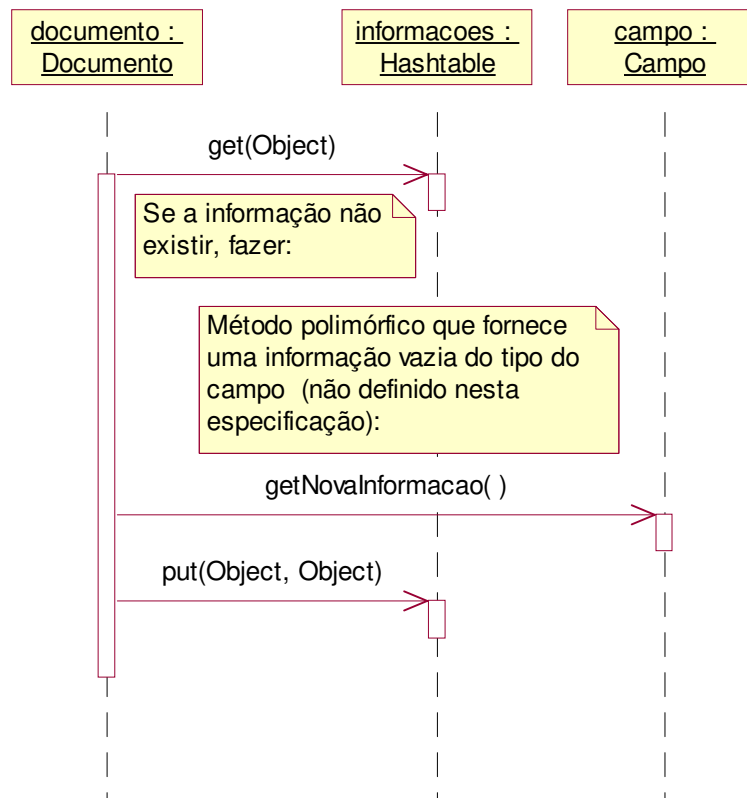


Figura A.26: Diagrama de seqüência do método *getInformacao(Campo campo)*

O caso de uso *Solicitando documentação HTML de um componente* é mapeado para o método *getDocumentacaoHTML(Documentavel componente)*, que tem seu diagrama de seqüência exibido na figura A.27.

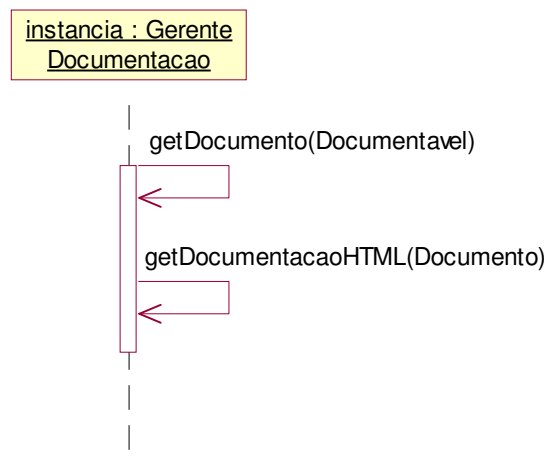


Figura A.27: Diagrama de seqüência do método *getDocumentacaoHTML(Documentavel componente)*

O método *getDocumentacaoHTML(Documentavel componente)* faz uso do método *getDocumentacaoHTML(Documento documento)*, que tem o seu diagrama de sequência exibido na figura A.28.

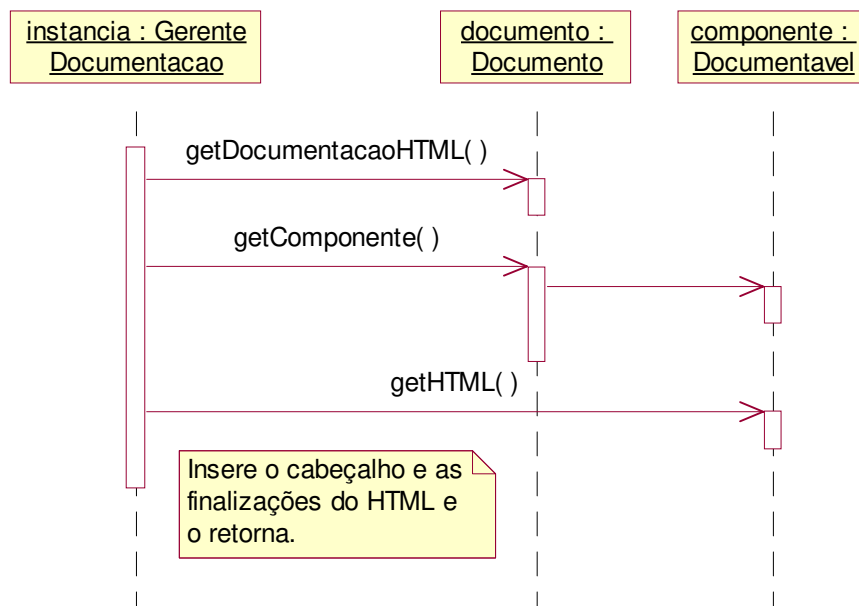


Figura A.28: Diagrama de sequência do método *getDocumentacaoHTML(Documento documento)*

O método *getDocumentacaoHTML(Documento documento)* faz uso dos métodos *getDocumentacaoHTML()*, que tem o seu diagrama de sequência exibido na figura A.29, *getComponente()*, que retorna o componente dono do documento e *getHTML()*, que retorna a documentação implícita do componente.

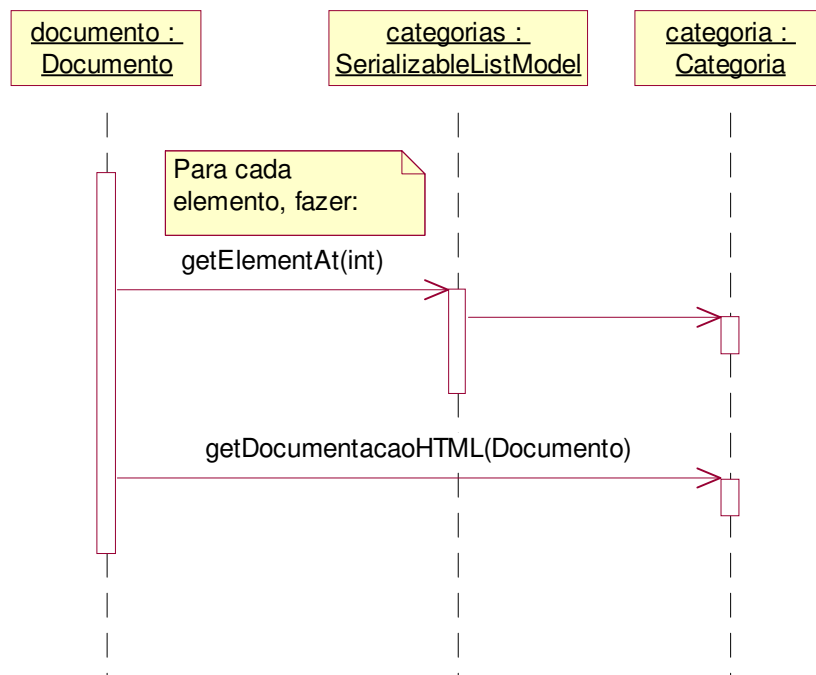


Figura A.29: Diagrama de seqüência do método `getDocumentacaoHTML()`

O método `getDocumentacaoHTML()` faz uso do método `getDocumentacaoHTML(Documento documento)`, que tem o seu diagrama de seqüência exibido na figura A.30.

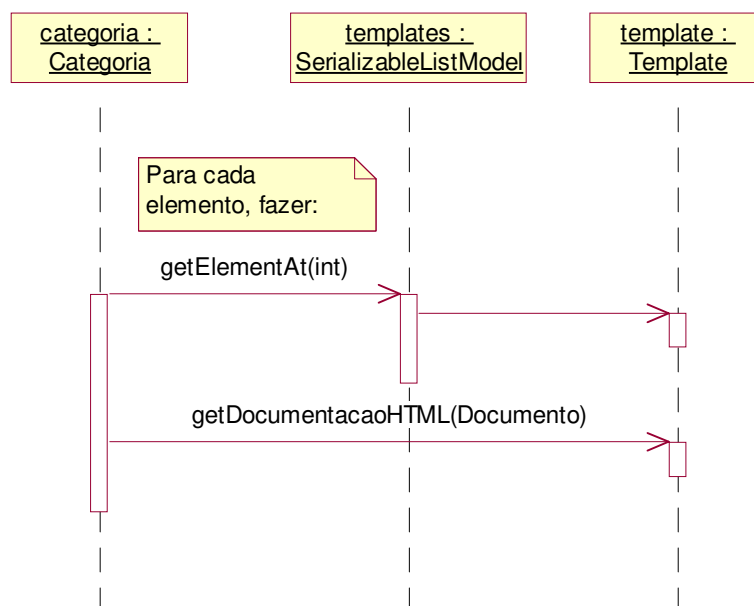


Figura A.30: Diagrama de seqüência do método `getDocumentacaoHTML(Documento documento)`

O método *getDocumentacaoHTML(Documento documento)* faz uso do método *getDocumentacaoHTML(Documento documento)*, que tem o seu diagrama de sequência exibido na figura A.31.

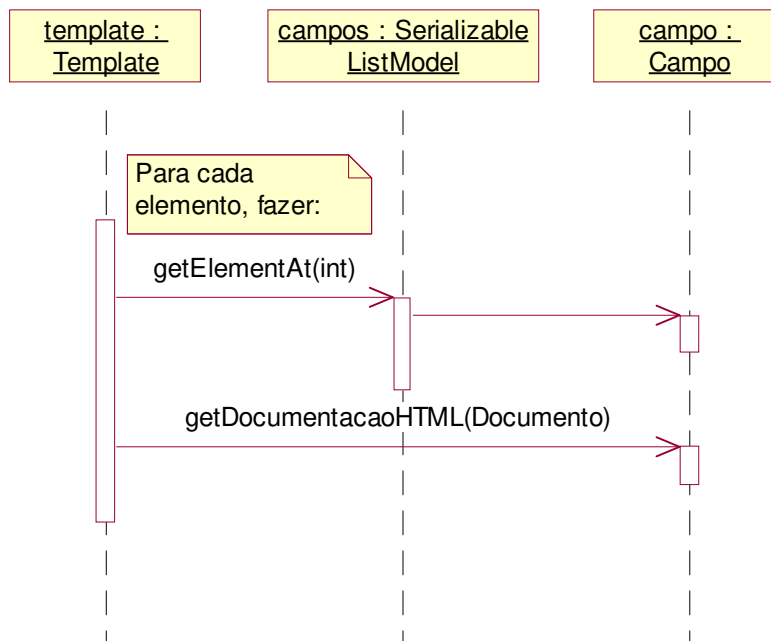


Figura A.31: Diagrama de sequência do método *getDocumentacaoHTML(Documento documento)*

O método *getDocumentacaoHTML(Documento documento)* faz uso do método *getInformacao(Campo campo)*, que tem o seu diagrama de sequência exibido na figura A.32.

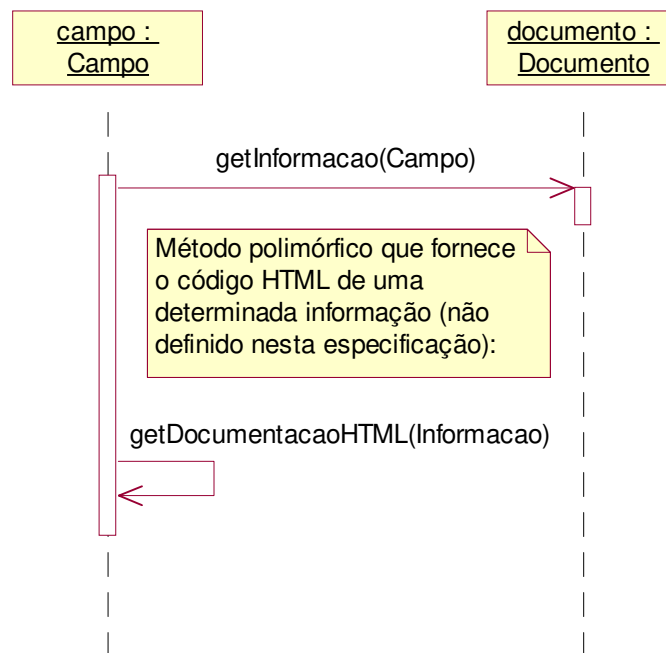


Figura A.32: Diagrama de seqüência do método *getInformacao(Campo campo)*