UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DA PRODUÇÃO

DOCUMENTAÇÃO DA ARQUITETURA DE SISTEMAS E FRAMEWORKS PARA PROCESSAMENTO E ANÁLISE DE IMAGENS: UMA ABORDAGEM BASEADA EM VISÕES DA UML E PADRÕES

DISSERTAÇÃO DE MESTRADO

Patricia Blini Estivalete

Santa Maria, RS, Brasil

2007

DOCUMENTAÇÃO DA ARQUITETURA DE SISTEMAS E FRAMEWORKS PARA PROCESSAMENTO E ANÁLISE DE IMAGENS: UMA ABORDAGEM BASEADA EM VISÕES DA UML E PADRÕES

por

Patrícia Blini Estivalete

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-graduação em Engenharia da Produção, Área de Concentração em Tecnologia da Informação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Engenharia da Produção.**

Orientador: Prof. PhD Marcos Cordeiro d'Ornellas

Santa Maria, RS, Brasil.

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia da Produção

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

DOCUMENTAÇÃO DA ARQUITETURA DE SISTEMAS E FRAMEWORKS PARA PROCESSAMENTO E ANÁLISE DE IMAGENS: UMA ABORDAGEM BASEADA EM VISÕES DA UML E PADRÕES

elaborada por
Patrícia Blini Estivalete

como requisito parcial para obtenção do grau de
Mestre em Engenharia da Produção

COMISÃO EXAMINADORA:

Marcos Cordeiro d'Ornellas, PhD.
(Presidente/Orientador)

Cesar Tadeu Pozzer, Dr. (UFSM)

Lisandra Manzoni Fontoura, Dra. (URI)

Santa Maria, de 21 de Dezembro de 2007.

Ao meu marido, Emerson, pelo exemplo de retidão, de confiança, amor, apoio e dedicação em todos os momentos.

“Estar no poder é como ser
uma dama. Se tiver que
lembrar às pessoas que você é,
você não é.”
(Margaret Thatcher)

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me dado sabedoria, fé, garra para que eu não desistisse de meus objetivos, capacidade para que eu pudesse realizá-los e por permitir que eu sinta-O sempre iluminando meus passos.

Aos meus pais, Paulo e Helena, meus maiores educadores, obrigada pelo exemplo de vida, de família, de dedicação, de amor e pelos desejos concedidos a mim em detrimento dos seus.

Ao meu marido, Emerson, que dá força aos meus sonhos, apoio aos meus planos e alegria a todos os meus dias. Dedico essa mensagem a você: Os "grandes guerreiros" carregam sempre consigo a garra, a coragem e a força para driblar as dificuldades que irão enfrentar, mas que ao final dessa luta árdua sairão vitoriosos e orgulhosos por terem lutado por seus ideais. Que possamos escrever nossa história alicerçados nesses atributos.

Ao meu orientador, Marcos C. d'Ornellas, pelos ensinamentos, pela atenção e por ter oportunizado novos conhecimentos.

Ao Prof PhD Ademar Aguiar da Universidade do Porto em Portugal, o meu agradecimento pela ajuda por mim solicitada de forma casual quando esteve no Brasil. Embora distante, seus ensinamentos foram de fundamental importância para que este trabalho pudesse ser concluído com sucesso. Felizmente a tecnologia oportuniza contatos que transcendem o "mar salgado", outrora tão temido pelos portugueses e deveras cultuado pelo grande poeta Fernando Pessoa.

Ao Colega, Daniel Michelon que estendeu sua mão, dispondo seu tempo e seus conhecimentos no sentido de auxiliar na implementação do trabalho.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Engenharia da Produção
Universidade Federal de Santa Maria

DOCUMENTAÇÃO DA ARQUITETURA DE SISTEMAS E FRAMEWORKS PARA PROCESSAMENTO E ANÁLISE DE IMAGENS: UMA ABORDAGEM BASEADA EM VISÕES DA UML E PADRÕES

AUTORA: Patrícia Blini Estivalete
ORIENTADOR: Marcos Cordeiro d'Ornellas
Data e Local da Defesa: Santa Maria, 21 dezembro de 2007.

Na documentação da arquitetura de software deve ser considerado dois pontos relevantes: i) a arquitetura é composta por várias visões, e ii) cada visão identifica um conjunto de componentes e interações, permitindo melhor compreensão entre desenvolvedores e projetistas do sistema. A documentação dos componentes, quando se tratando de reusabilidade, deve apresentar um conjunto de informações referente a sua funcionalidade e aplicabilidade a fim de adaptá-los em aplicações de mesmo domínio. Quando pensando em interação entre componentes, as informações relacionadas às mensagens de comunicação devem ser bem identificadas, para que o processo de funcionamento seja claramente entendido. Este trabalho apresenta a criação de um catálogo de Padrões de Documentação da Arquitetura de sistemas e *frameworks* para processamento e análise de imagens, abordando o conceito de *design patterns* e visões arquiteturais da UML, as quais servirão de base para elaboração dos padrões. Através do uso desses, os desenvolvedores sentirão mais clareza do que é necessário documentar e mais seguros em reusar componentes previamente testados e documentados. Apresenta-se também neste trabalho, o desenvolvimento de um ambiente de *templates* dos Padrões na ferramenta MoinMoinWiki, com o objetivo de criar um cenário para o trabalho colaborativo, e também um repositório de informações sobre a documentação da arquitetura de sistemas e frameworks no domínio de imagens.

Palavras-chave: Padrões de Projeto, Documentação de Software, Arquitetura de Software, Wiki.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Production Engineering
Federal University of Santa Maria

DOCUMENTATION OF SYSTEMS ARCHITECTURE AND FRAMEWORKS FOR IMAGES PROCESSING AND ANALYSIS: AN APPROACH BASED ON VISIONS OF UML AND PATTERNS

AUTHOR: Patrícia Blini Estivalete
ADVISOR: Marcos Cordeiro d'Ornellas
Date and city: Santa Maria, 21th december 2007.

In the documentation of software architecture, it must be considered two relevant points: i) the architecture is composed by several visions, and ii) each vision identifies a set of components and interactions which allows a better understanding between developers and system programmers. The documentation of the components, when treating reusability, must present a set of information concerning its functionality and applicability in order to adapt them in applications of same domain. When we think of interaction among components, the information concerning the communication messages must be well identified so that the process of working may be clearly understood. This work shows the creation of a catalog of documentation patterns for the architecture of image processing and analysis systems approaching on the concept of design patterns and architecture visions of UML which will be used as the base for designing the Patterns. Through its use, the developers will be more sure of which is necessary to document and safer when reusing components previously tested and documented. It is also shown in this work the development of templates environment of the Patterns in the tool MoinMoinWiki aiming at creating a scenery for collaborative work and also a repository of information on documentation of systems architecture and frameworks within image domain.

Key-words: Design Pattern, Software Documentation, Software Architecture, Wiki

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, a arquitetura de software ganhou reconhecimento nas pesquisas da engenharia de software, possibilitando a descoberta de novos estilos arquiteturais para desenvolvimento de sistemas. Conforme Sommerville (2004), a arquitetura de software define, através de um alto nível, o sistema em termos de componentes, interação entre eles e atributos e funcionalidades de cada componente.

A arquitetura pode ser observada em termos das visões arquiteturais. As visões são usadas para representar aspectos diferentes do sistema, como por exemplo, uma visão arquitetural pode documentar a estrutura de um sistema como uma descrição de camadas na qual mostra o componente, suas responsabilidades e a interação com outros componentes, enquanto outra visão documenta a estrutura de um sistema em termos de caso de uso de suas funcionalidades.

Apesar das necessidades de documentação da arquitetura, observa-se a importância do desenvolvimento de regras claras e objetivas específicas a domínio para prover o reuso dos componentes de software.

Ao longo deste trabalho será retratado o cenário de problemas associados à documentação de sistemas para processamento e análise de imagens, enfatizando a necessidade do uso dessas no desenvolvimento de sistemas. Foi elaborada uma abordagem sobre algumas ferramentas baseadas em WikiWikiWeb existentes, para que sirvam de embasamento aos resultados deste trabalho.

Este trabalho tem como resultados a análise e o desenvolvimento de vários aspectos relacionados à documentação da arquitetura de sistemas para processamento e análise de imagens, o que inclui: a) desenvolvimento de padrões de documentação baseados em *design patterns* e nas visões arquiteturais da UML, objetivando-se a criação de regras claras, b) aplicação desses padrões na construção de *templates* integrados a ferramenta MoinMoinWiki e, c) realização de um Estudo de Caso para validação dos padrões de documentação a serem utilizados na catalogação da arquitetura do sistema “Arthemis”.

1.1 Objetivo

Este trabalho tem como objetivo principal a criação de um catálogo de Padrões de Documentação da Arquitetura no domínio de processamento e análise de imagem, abordando o conceito de *design patterns* e visões arquiteturais da UML, as quais definem os princípios básicos de documentação de arquitetura de *software*.

Outro objetivo é o desenvolvimento de um ambiente de *templates* desses Padrões na ferramenta MoinMoinWiki, intitulado como "DocArqWiki", para a documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, retornando o resultado dessa catalogação de forma clara e objetiva.

De forma secundária, o ambiente wiki constitui-se como um repositório de informações sobre a documentação da arquitetura de sistemas e frameworks na área de imagens, já que para a documentação desses é necessário seguir uma padronização pré-estabelecida para que então, as informações sejam repassadas aos desenvolvedores.

Pretende-se também que as soluções sejam aplicadas com objetivo de tornar mais fácil ao desenvolvedor o encontro de ambientes de documentação que sejam compatíveis com seu domínio de aplicação, tornando-os cientes do que os documentos devem apresentar, proporcionando maior confiabilidade na utilização desses recursos.

Outro objetivo deste trabalho é apresentar novas contribuições à área de engenharia de software e a desenvolvedores de sistemas no domínio de processamento e análise de imagens, buscando assim, adicionar aos "Padrões de Documentação da Arquitetura" valores para alcançar o estado da arte na área. Com isso, este trabalho visa contribuir para a área de documentação de sistemas para processamento e análise de imagens, vista a demanda pela padronização de documentos, caracterizando cada domínio.

Dessa forma, acredita-se em um aumento na criação de documentação da arquitetura de sistemas na área de imagens. O ambiente de documentação permite ainda que haja uma maior integração entre os autores dos documentos e seus colaboradores.

1.2 Motivação

A motivação encontrada para propor este trabalho deve-se ao fato de que a qualidade do desenvolvimento de software pode ser colocada em risco pela falta de uma padronização para a documentação da arquitetura específica a cada domínio de aplicação.

Pesquisadores em engenharia de software vêm aperfeiçoando métodos, técnicas e ferramentas em busca de melhores resultados na área de documentação de software. Nesse contexto, estudos em *design patterns* têm auxiliado no desenvolvimento de novas técnicas para elaboração de documentos padronizados visando catalogar a arquitetura de software.

Assim, aos poucos, a documentação vai se tornando essencial no ciclo de desenvolvimento de software pelo fato de existir normas claras e específicas a cada área. Isso proporciona, cada vez mais, aos desenvolvedores a confiabilidade de que seus projetos arquitetônicos serão bem registrados.

Devido a não catalogação correta das informações relevantes da arquitetura decorrente do uso de ferramentas não apropriadas e devido a processos muito genéricos oferecidos aos desenvolvedores, fica evidente a necessidade da criação de Padrões de Documentação específicos para a área de processamento e análise de imagens que possa informá-lo sobre as regras de registro da arquitetura.

Através dos padrões de documentação da arquitetura oferecidos, da maior transparência quanto ao que é preciso ser documentado e com a disponibilização da documentação do projeto da arquitetura, os desenvolvedores se sentirão mais seguros em reutilizar componentes previamente testados e documentados.

Os padrões foram desenvolvidos com vistas a atender os pontos importantes identificados na arquitetura de sistemas para processamento e análise de imagens durante a pesquisa de mestrado. Com isso, novas informações relevantes a esse domínio podem ser adaptadas aos Padrões.

Outra motivação encontrada foi a necessidade de disponibilizar os padrões em um ambiente colaborativo, chamado “DocArqWiki”, onde, baseado nesses, serão definidos *templates* a serem seguidos pelos desenvolvedores, de modo a atender os requisitos dos documentos da arquitetura de sistemas para processamento e análise de imagens.

A metodologia usada neste trabalho basea-se em uma abordagem focada na arquitetura de software, visando buscar novas técnicas para documentá-la de maneira padronizada, a qual envolveu um conjunto de ações: a) definição de como documentar arquitetura de software; b) definição e adaptação de visões arquiteturais para a padronização de documentos; c) criação

de padrões de documentação da arquitetura de sistemas para processamento e análise de imagens e d) integração dos padrões a um ambiente colaborativo para realização da documentação.

1.3 Estrutura da Dissertação

Este trabalho está organizado da seguinte maneira:

No capítulo 2 são abordados os conceitos de arquitetura de software evidenciando a importância da arquitetura para domínio e baseada em componentes, bem como, estilos e padrões arquiteturais. Descreve também características da reutilização de arquiteturas específicas a domínios.

No capítulo 3 é apresentado um estudo sobre os elementos relevantes para a criação da documentação da arquitetura de software.

No capítulo 4 são apresentadas ferramentas *wikis* disponíveis na *web* que possam apoiar a documentação da arquitetura de *software* de forma colaborativa.

No capítulo 5 é detalhado todo o desenvolvimento dos padrões de documentação da arquitetura de sistemas e *frameworks* para processamento e análise de imagens baseados em visões da UML e *design patterns.*

No capítulo 6 é apresentada uma visão geral do ambiente colaborativo "DocArqwiki", disponibilizando os *templates* referentes a cada Padrão criado.

No capítulo 7 é apresentado um estudo de caso referente ao uso dos padrões criados na documentação da arquitetura do sistema "Arthemis".

No capítulo 8 são apresentadas as conclusões e contribuições desta dissertação, bem como trabalhos futuros.

2 ARQUITETURA DE SOFTWARE

A necessidade de construir software cada vez mais complexo pode afetar a confiança dos usuários em relação à usabilidade do sistema, caso a arquitetura não esteja bem estruturada. Quando bem definida, o projeto de uma arquitetura pode fornecer um nível de abstração que permite a análise do comportamento do sistema como um todo, sem a necessidade de se conhecer detalhes de implementação (CARVALHO, 2001).

De um lado, uma arquitetura bem estruturada pode ajudar assegurar que um sistema satisfará as exigências fundamentais em tais áreas como desempenho, confiabilidade, portabilidade, estabilidade e interoperabilidade e, de outro, uma arquitetura ruim pode ser desastroso. Problemas com interoperabilidade podem dificultar o desenvolvimento de componentes em uma arquitetura (DAVIS, GAMBLE e PAYTON, 2002).

A seguir são apresentados os conceitos de arquitetura de software, assim como, arquiteturas de domínio e baseada em componentes. São abordados também os assuntos referentes a estilos, padrões e visões arquiteturais.

2.1 Conceitos de arquitetura de software

A arquitetura de software pode ser caracterizada como a estrutura e organização em vários componentes que possuem comunicação entre si, e pode ser vista como a busca da solução técnica para um problema estrutural. Hopkins (2000) diz que um componente de software é visto como um pacote físico executável com uma interface bem definida e pública.

Para Jazayeri, Ran e Linden (2000), o projeto da arquitetura de software é visto como uma ferramenta para analisar a complexidade do sistema, o qual deve conter os requisitos funcionais e não funcionais e um conjunto de componentes e seus relacionamentos. Já Krafzig, Banke e Slama (2004) relatam que a arquitetura de software é um conjunto de declarações, que descreve os componentes de software e atribui funcionalidades de sistema para cada um deles. Ela descreve a estrutura técnica, limitações e características dos componentes, bem como as interfaces entre eles.

Com o avanço da complexidade no desenvolvimento de software, a questão da modelagem da arquitetura de software tornou-se uma das principais preocupações em termos de qualidade e confiabilidade (CLEMENTS e NORTHROP, 1996).

À medida que se tem uma arquitetura bem estruturada para definir a construção do sistema, a comunicação entre projetistas e desenvolvedores torna-se mais fácil, possibilitando maior qualidade no desenvolvimento do sistema e garantindo o reuso de soluções.

Assim, com a interação entre usuários, arquitetos, analistas e desenvolvedores visando o foco no projeto arquitetural e mantendo o direcionamento das decisões arquitetônicas determinadas anteriormente, o reuso, é encaminhado em vários níveis como: reutilização não só de componentes de software, mas também de idéias, modelos, estilos e padrões relacionados à arquitetura. Com isso, os projetos arquiteturais oferecem um guia concreto e complexo para o desenvolvimento de software (SHAW, 2001).

Pesquisas na área abordam a utilização de *design patterns* na construção de estilos arquiteturais como o Padrão *Layer* que proporciona aos projetistas uma maior flexibilidade no desenvolvimento por subdividir o sistema em camadas diferentes e que se bem projetadas podem ser facilmente reutilizadas.

Assim, um padrão arquitetônico nomeia e identifica os elementos chaves de uma arquitetura e seus relacionamentos com o intuito de torná-la útil para a criação de sistemas reutilizáveis (GAMMA et al., 2000). Já Buschmann (1996) diz que padrões arquitetônicos expressam um esquema de organização estrutural essencial para sistemas de software e que fornecem um conjunto de subsistemas predefinidos, especificando suas responsabilidades e incluindo regras e diretrizes para organizar os relacionamentos entre eles.

Dessa forma, a utilização de padrões na modelagem de arquiteturas de software permite a documentação de idéias já comprovadas relacionadas à estrutura do sistema, facilitando assim, a reutilização, manutenção e extensão do sistema.

2.2 Arquitetura baseada em componentes

A arquitetura de componentes deve ser estruturada em camadas distintas, de modo que os componentes possuem diferentes níveis de abstração e as camadas inferiores disponibilizam serviços para as camadas superiores (XIA, MICHAEL e WONG 2000). Para Blois e Becker (2002), uma arquitetura componentizada possibilita plugar e desplugar componentes de forma a construir uma aplicação adequada aos propósitos específicos de cada domínio.

De acordo com Brown (2000), a organização da arquitetura de componentes é baseada em três camadas de serviços, como pode ser visualizado na Figura 1.

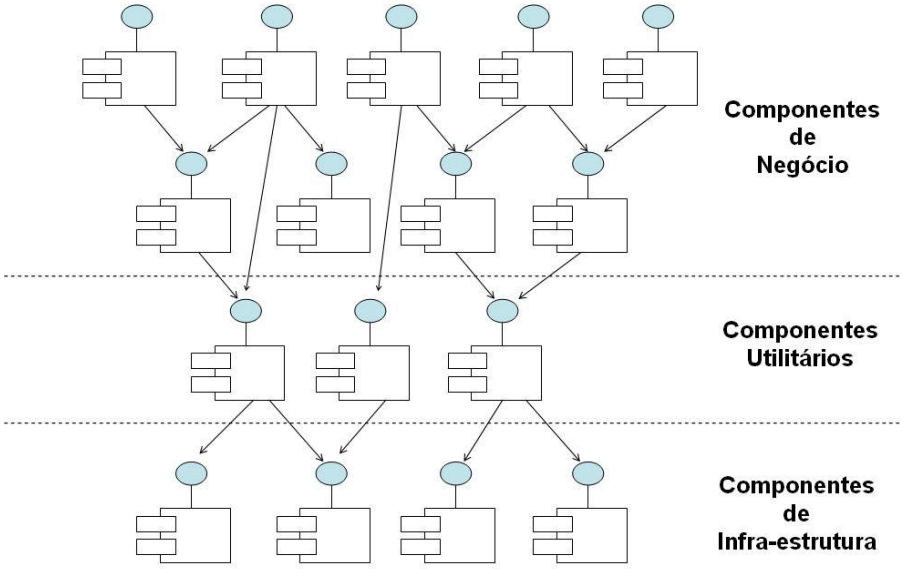

Figura 1: Componentes de uma Arquitetura em Camadas

A seguir é detalhado do que trata cada uma dessas camadas, baseado nas definições propostas por Brown (2000), e é mostrada uma breve explicação da importância dos componentes.

- **Camada 1 - Componentes de Negócio:** esses componentes são requisitos necessários para representar, implementar e implantar conceitos do domínio e são criados pelos tipos de negócio, possuindo relação com às características conceituais e de entidade do domínio.

Como apresentado por Herzum e Sims (2000) e Cheesman e Daniels (2000) esta camada identifica duas categorias de componentes de negócio: componentes de entidade e componentes de processo. Componentes de entidade representam os conceitos básicos de um domínio de aplicação, onde os processos de negócio operam. Já os componentes de processo representam o comportamento do domínio da aplicação, estando relacionados aos casos de uso que modelam as características funcionais do negócio.

- **Camada 2 – Componentes Utilitários:** os componentes utilitários prestam serviços genéricos necessários ao desenvolvimento das aplicações (BROWN, 2000).

Os componentes desta camada não pertencem a nenhum domínio específico, mas podem ser usados por diversas aplicações de negócio, como, por exemplo, formulários genéricos.

- **Camada 3 – Componentes de Infra-estrutura:** os componentes de infra-estrutura são responsáveis por permitir a integração com as plataformas de execução do

software. Para Brown (2000), eles estão mais relacionados às tecnologias de desenvolvimento do que com os componentes das camadas superiores.

Na arquitetura de domínio, os componentes utilitários e de infra-estrutura exercem os serviços de interação por meio das mensagens de comunicação entre os componentes, os quais muitas vezes são disponibilizados na arquitetura através de um componente, com uma interface bem definida, permitindo a integração com os outros componentes que requerem seus serviços.

Os componentes de negócio, utilitários e de infra-estrutura podem ser desenvolvidos sem dependência de outro componente, pois são originários de elementos diferentes. Porém, os componentes de processo precisam dos componentes de negócio, mais especificamente de suas interfaces fornecidas para a criação do componente e do seu diagrama de interação.

Para cada componente de processo é modelado um diagrama de interação, o qual documenta as solicitações de serviço realizadas pelo usuário ao requisitar um serviço fornecido pela interface do componente de processo.

2.3 Arquiteturas de software para domínios e baseadas em componentes

A arquitetura de software, no contexto de reutilização, deve atender aos requisitos de um domínio de aplicação, e ainda, precisa apresentar uma estrutura de referência para o domínio modelado, a qual pode ser utilizada em diversos sistemas da área.

Assim, se a arquitetura for modelada para uma única aplicação não é uma tarefa trivial, o nível de complexidade aumenta quando a arquitetura construída representa um domínio específico. Dessa forma, a complexidade influência no custo, tempo e esforço de construção desta arquitetura, e na formação esperada da equipe de desenvolvimento para lidar com tal nível de complexidade (MENDES, 2002).

Para obter os resultados esperados de um projeto arquitetural bem estruturado, é necessário que o arquiteto de software possua habilidades e técnicas para construí-la. De forma que, também é preciso ter conhecimento sobre os requisitos das aplicações, das tecnologias utilizadas na construção da arquitetura e dos processos de desenvolvimento adequados para a criação das aplicações. As habilidades devem estar alinhadas aos objetivos organizacionais, os quais influenciam nas decisões técnicas (VITHARANA *et al.*, 2003).

Nas seções seguintes, apresentam-se os princípios básicos para estruturar uma arquitetura específica para domínio e a importância do uso de estilos e padrões arquiteturais.

2.3.1 Arquiteturas de software específicas para domínio

Uma arquitetura de software específica para domínio deve propor uma solução estruturada para os requisitos que compõem uma aplicação para certo domínio, a qual é conhecida como Arquiteturas de Software Específicas para Domínio, ou *Domain Specific Software Architeacture* (DSSA). Uma DSSA fornece uma base estrutural para a interoperabilidade de componentes dentro de um domínio (MENDES, 2002).

Assim, uma DSSA deve apresentar mais do que uma estrutura arquitetural a um domínio de aplicação. Ela representa um conjunto de componentes de software específico a certo domínio, generalizados para reuso em aplicações de mesma área e dispostos em uma estrutura padronizada (HAYES, 1994).

De acordo com Tracz (1995), Xavier (2001) e Mendes (2002), uma DSSA é composta por quatro elementos principais, conforme mostra a Figura 2:

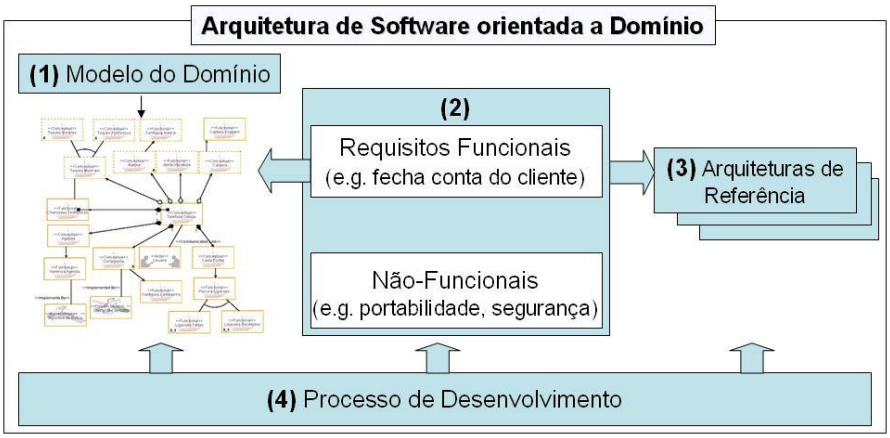

Figura 2: Elementos de uma DSSA

Nessa figura pode-se observar que: 1) modelo do domínio ou modelo de características do domínio da aplicação. 2) requisitos de referência, isto é, requisitos funcionais e não-funcionais do domínio, sendo que estes últimos estabelecem restrições sobre a arquitetura e a implementação. 3) arquitetura de referência satisfaz os requisitos funcionais e não-funcionais do domínio. 4) processo de desenvolvimento apóia a Engenharia de Domínio como um todo.

O modelo de domínio possibilita que os projetistas compreendam os diversos aspectos de um domínio de aplicação sem duplicidade de informação. Um domínio pode ser definido por um conjunto de problemas comuns que definem os requisitos de referência para modelagem da aplicação. Com o seu uso, é possível descrever o contexto que as aplicações são criadas, caracterizando seus atributos e interações.

Os requisitos de referência são divididos em requisitos funcionais, que são características de definição do contexto do problema, e requisitos não-funcionais, de projeto e de implementação, representam as características restritivas do contexto da solução.

A arquitetura de referência pode não atender todos os requisitos funcionais e não-funcionais de um domínio de aplicação. Dessa forma, é possível criar uma ou mais arquiteturas de referência, possibilitando ao projetista maior flexibilidade na construção da arquitetura da aplicação.

Miler (2000) ressalta a diferença entre a arquitetura de referência e a arquitetura de aplicação. A arquitetura de referência é construída na fase do projeto de domínio e compreende a arquitetura de um conjunto de aplicações. Já a arquitetura de aplicação representa uma instância da arquitetura de referência.

O uso de arquitetura de referência em certo domínio de aplicação possibilita ao projetista a integração e a reutilização de idéias, métodos e componentes dentro de um contexto de desenvolvimento de mesma área. Com isso, busca-se atingir níveis mais altos de qualidade, reduzindo o tempo de desenvolvimento e os riscos que possam acontecer durante o processo de criação do software.

2.3.2 Estilos e padrões arquiteturais

Um estilo arquitetural define uma coleção de regras de modelagem de projeto que identificam componentes e seus conectores, os quais podem ser usados para desenvolver aplicações em certos domínios. Conectores representam a interação entre componentes em um estilo arquitetural. Cada conector possui a especificação de um protocolo que define suas propriedades, como: os tipos de interfaces que pode intermediar e a ordem em que os eventos acontecem (SHAW e GARLAN, 1996).

Os estilos arquiteturais compõem um vocabulário que expressam uma nomenclatura comum, como: tipos de componentes, tipos de conectores, estruturas de controle e interação entre componentes. Existem vários estilos arquiteturais para modelagem de projetos arquiteturais chamados de *Pipes and Filters*, Camadas, Baseado em Eventos e Organização Orientada a Objetos.

De acordo com Gacek (1998), alguns estilos estão sendo descritos como idiomas arquiteturais. A Tabela 1 mostra os estilos arquiteturais mais utilizados.

Pipes & Filters Transformadores incrementais de cadeias Ex.: Pipes do Unix, processamento de sinais, compiladores tradicionais.
Cliente- servidor Serviços compartilhados, fornecidos através da solicitação dos clientes distribuídos. Ex.: Servidores de arquivos e banco de dados distribuídos.
Camadas (*Layer*) Sistemas particionados em camadas, que atuam como máquinas virtuais. Ex.: Núcleos de S.O., ISO OSI.
Processos de comunicação Sistema é composto de processos independentes e concorrentes. Ex.: Muitos sistemas distribuídos.
Interpretadores Ligações entre programas abstratos e as máquinas sobre as quais devem rodar. Ex.: Sistemas baseados em regras.
Baseados em eventos e Invocação implícita A interação entre os componentes é realizada através do anúncio de um ou mais eventos. Ex.: Mecanismos de invocação.
Repositório Sistemas que possuem um repositório de informações compartilhadas. Ex.: Ambientes de programação com repositório de programas.
Organização em programa principal e subrotinas Estabelecimento de hierarquias e procedimentos. Ex.: Programas estruturados.
Orientados a um domínio específico Customização de uma estrutura para uma família de aplicações Ex.: Arquiteturas desenvolvidas para domínios de aviação.

Tabela 1: Estilos arquiteturais mais conhecidos.

Estudos apresentam várias propostas de estilos arquiteturais. Alguns considerados genéricos e que podem, a princípio, ser utilizados por qualquer domínio de aplicação, como, camadas e *pipe & filter* (SHAW e GARLAN, 1996). Já outros estilos são orientados a domínio Emmerich et al. (2001) ou orientados ao desenvolvimento baseado em componentes (HERZUM e SIMS, 2000). Mesmo que exista certa dificuldade prática em distinguir padrões de estilos arquiteturais, observa-se que os estilos são mais abstratos que os padrões. Por exemplo, Camadas, pode ser considerado como um estilo arquitetural ou como um padrão, porém representam níveis diferentes de abstração.

Métodos de desenvolvimento baseado em componentes sugeridos por D'Souza e Wills (1999), Brown (2000) e Herzum e Sims (2000) e a proposta de modelagem de arquiteturas de referência baseadas em componentes indicadas por Collins-Cope e Matthews (2000) recomendam o uso do estilo arquitetural em camadas. Dessa forma, os componentes, que representam as abstrações do domínio, como: negócio, processo, infra-estrutura e utilitário, são organizados nas camadas da arquitetura conforme o serviço que disponibilizam.

Estilo arquitetural baseado em tipos encapsula em apenas um componente todas as instâncias de um tipo de negócio, permitindo o acesso às suas informações através de suas interfaces. Assim, estes componentes podem ser chamados de gerentes de instância. Estes estilos podem ser encontrados nos métodos de desenvolvimento baseado em componentes, Catalysis, definido por Cheesman e Daniels (2000) e UML *Components* (D'SOUZA e WILLS, 1999).

Logo, o estilo baseado em instâncias proposto por Herzum e Sims (2000) possibilita que o acesso seja feito diretamente as instâncias dos tipos de negócio. Porém, para este acesso ser viável é preciso ter dois tipos de componentes: o componente de entidade, que são as instâncias e o componente, coleção, que serve como repositório para a criação e gerência de componentes entidade, fornecendo serviços de busca de componentes pertencentes a mesma coleção.

O uso do estilo ou padrão arquitetural em camadas integrado com os estilos baseados em tipos ou instâncias pode representar uma boa alternativa para a modelagem de uma arquitetura, pois aumenta a flexibilidade do sistema pela definição de diversos níveis de abstração. Por outro lado, usualmente o estilo em camadas diminui o desempenho da arquitetura criada (SHAW e GARLAN, 1996).

De acordo com Buschmann et al. (1996), padrões arquiteturais expressam um esquema de organização estrutural de sistemas. A partir desta organização, são definidos os subsistemas e suas interações. Padrões são descritos através de uma estrutura que define alguns elementos que devem ser seguidos, tais como: nome do padrão, contexto, problema, solução, estrutura, usos conhecidos e padrões relacionados. Em Buschmann et al. (1996) são apresentados os padrões arquiteturais mais conhecidos, como, por exemplo, os padrões *Model-View- Controller* (MVC) e Camadas.

No entanto, a vantagem de usar padrões arquiteturais é que documentam decisões importantes sobre os elementos da arquitetura e de suas interações. E ainda, possibilitam a coordenação entre os projetistas da arquitetura da aplicação.

3 DOCUMENTANDO ARQUITETURA DE SOFTWARE

Uma arquitetura de software para um sistema é a estrutura ou estruturas do sistema que inclui elementos, suas propriedades externamente visíveis e a relação entre eles (BASS, CLEMENTS e KAZMAN, 2003).

Baseado nesse conceito, pesquisadores da área substituíram o termo estrutura para visões quando se referiram à documentação da arquitetura de software. Visões contêm elementos e relações entre os elementos. Cada visão mostra elementos e relações de determinado tipos, omitindo os demais (STAFFORD e WOLF, 2001). A definição dos tipos de elementos e relações que aparecem em cada visão serve de base para criar uma padronização para a documentação da arquitetura.

Segundo Merson (2000), documentar a arquitetura de software é uma questão de documentar as visões que são relevantes e então adicionar as informações referentes a cada visão. Se considerar a documentação da arquitetura em camadas é preciso registrar as propriedades dos componentes e suas interações (BACHMANN et al., 2000).

A seguir são abordados os conceitos de como documentar arquitetura baseada nas visões da UML, mais especificamente na visão lógica e de caso de uso e as vantagens sobre o uso de ferramentas de trabalho cooperativo, como *wikis*, na criação de documentação da arquitetura de software.

3.1 Documentando arquitetura baseada nas visões da UML

A linguagem de modelagem *Unified Modeling Language* (UML) proporciona uma forma padrão para a criação de projetos arquiteturais de sistemas. Essa contempla aspectos conceituais, tais como processos de negócios e funções do sistema, e ainda, itens concretos como as classes escritas em linguagens de programação, esquemas de banco de dados e componentes de software reutilizáveis (BOOCH, RUMBAUGH e JACOBSON, 2005).

A UML é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas de softwares. É utilizada também para modelagem de negócio e outros sistemas que não são softwares, representando uma coleção das melhores práticas de engenharia na modelagem de sistemas (OMG, 2006). Ainda, ela pode ser vista como uma notação extensível que inclui definições de diagramas de modelagem para as

diversas atividades do desenvolvimento, desde as primeiras até as mais refinadas (BOOCH, RUMBAUGH e JACOBSON, 2005).

Como mencionado por Hilliard (1999), a UML é composta por visões que podem capturar aspectos estruturais e comportamentais do desenvolvimento de software. Visões estruturais utilizam componentes, pacotes e casos de uso. Por sua vez, as visões comportamentais são representadas por cenários, estados e atividades.

As visões arquiteturais são usadas para representar aspectos distintos de um sistema, cada visão pode prover um modelo de algum aspecto do sistema (IEEE, 2000) e (HILLIARD, 2000). Uma visão arquitetural pode documentar a estrutura de um sistema como uma descrição de camadas na qual o componente representa agrupamentos lógicos de código, enquanto outra visão poderia documentar a estrutura de um sistema em termos de sua configuração de execução, na qual componentes representam processos em comunicação.

A UML é composta por cinco visões arquiteturais, como: visão lógica, visão de processo, visão de implantação, visão de implementação e de caso de uso. Neste trabalho será abordada a visão lógica e de caso de uso para elaboração dos Padrões de Documentação da Arquitetura de sistemas e frameworks para processamento e análise de imagens. A visão lógica documenta as características de reuso dos componentes relacionados a um domínio específico, a interface de acesso entre os componentes, a interação entre eles e a estrutura de pacotes. Já a visão de caso de uso mostra o comportamento do sistema em termos de seqüências de ações observadas por um ator.

Assim, na documentação de componentes de software deve constar um conjunto de informações que auxilie no entendimento de sua funcionalidade e sua aplicabilidade com a finalidade de reuso em aplicações de mesmo domínio. Segundo Heineman e Council (2001), componentes de software devem possuir uma documentação adequada para auxiliar na busca, recuperação, adaptação e integração no sistema e que ateste sua confiabilidade.

É necessário criar uma documentação padrão para a interface dos componentes, pois, conforme Weinreich & Sametinger (2001), ela define os serviços que os desenvolvedores podem requisitar de um componente e as restrições que devem ser observadas ao implementá-las.

3.2 Documentando arquitetura com apoio de wikis

A necessidade da interação na aprendizagem e criação de documentação para a arquitetura de software pode ser observada sob a ótica da aprendizagem cooperativa. Nos últimos 20 anos, a pesquisa em Aprendizagem Cooperativa Apoiada por Computador (CSCL - *Computer-Supported Cooperative Learning*) deu origem a diversos ambientes que apóiam os processos de aprendizagem promovidos através de esforços colaborativos. A aprendizagem cooperativa é uma técnica com a qual os integrantes de um grupo se ajudam nos processos de aprendizagem, atuando como parceiros entre si e com o mediador, e visando adquirir conhecimento sobre um dado objeto (SMYSER, 1993).

As ferramentas *wikis* permitem criar um espaço interativo para que ocorra a troca de conhecimento entre os integrantes de um projeto de software. O seu funcionamento pode servir como uma forma de acesso colaborativo na criação e manutenção da documentação da arquitetura de software, criando soluções simples para problemas complexos através da construção de *templates* para catalogação dos componentes e de suas interações de maneira colaborativa.

As *wikis* estabeleceram-se de forma extremamente rápida (MÖLLER, 2005). A maioria das ferramentas *wikis* surgiu com a intenção de gerenciar documentos e ajudar na colaboração de trabalhos em grupo, permitindo que os integrantes contribuíssem com novas informações de forma rápida, fácil e prática.

Esses sistemas permitem que usuários geograficamente distribuídos ou co-presentes, conectados via uma rede de computadores, colaborem em tempo real através de um espaço de trabalho compartilhado (GUTWIN, STARK e GREENBERG, 1995).

A seguir são apresentados conceitos de wiki e trabalho colaborativo baseados em *wikis*. São abordados também os assuntos referentes à comunicação, coordenação, cooperação e percepção em ambientes de trabalho cooperativo, como *wikis*.

3.2.1 Conceitos de wikis

Leuf e Cunningham (2001) definem um *wiki* como "uma coleção livremente expansível de páginas *web* interligadas num sistema de hipertexto para armazenar e modificar informação - um banco de dados, onde cada página é facilmente editada por qualquer usuário com um browser". O princípio básico de uma ferramenta *wiki* é que o texto pode ser editado e

alterado pelos usuários a qualquer tempo sem a autorização do editor da versão anterior. Em *wikis* abertos, qualquer usuário tem a liberdade de editar uma página. Já em *wikis* com acesso restrito para edição, apenas usuários com permissão podem cooperar com novas informações.

O acesso às informações relevantes, como histórico de páginas, torna-se imprescindível para que os colaboradores mantenham sob controle novas contribuições, identifiquem as diferenças entre o novo conteúdo e o anterior, podendo fazer correções ou restaurar a página para a versão anterior.

Uma característica relevante do ambiente *wiki* é a possibilidade de criação e alteração de páginas, sem que o usuário tenha conhecimentos na área de desenvolvimento de páginas web. De acordo com Haetinger *et al* (2005), normalmente, não existe qualquer revisão antes da aceitação de cada modificação, ou seja, um artigo pode ser publicado, antes de sua aceitação pelo responsável do projeto.

A utilização de ferramentas *wikis* remete a idéia de conectivismo, que de acordo com Siemens (2004) significa: "o conectivismo é guiado pela noção de que as decisões são baseadas em fundamentos que mudam rapidamente". Assim, a informação obtida anteriormente pode ser modificada criando um ambiente de aprendizagem colaborativa através da efetiva participação dos usuários.

De acordo com Haetinger et al (2005), as ferramentas *wikis* proporcionam um novo formato de trabalho de co-autoria cooperativa, de mídia hipertextual e com navegação estruturada de forma não linear, onde cada página pode ter uma ou várias ligações a outras páginas, utilizando uma sintaxe específica através do "padrão link", garantindo desse modo que todos os integrantes de um projeto possam cooperar.

O hipertexto nas ferramentas *wikis* pode ser definido como hipertexto cooperativo, pois os usuários cadastrados no ambiente têm a permissão para interagir com qualquer texto, possibilitando a construção do conhecimento entre os usuários do projeto (PRIMO e RECUERO, 2003).

Segundo Davenport e Prusak (2004), as ferramentas de armazenamento de conhecimento de forma colaborativa, como as *wikis*, ferramentas de hipertexto e *groupware* que facilitam a colaboração através do compartilhamento de informações, permitem trabalho em grupo através de recursos integrados de comunicação, cooperação e coordenação.

3.2.2 Wikis e o trabalho cooperativo

O ambiente computacional que implementa os processos de apoio à cooperação, e assim possibilita o trabalho conjunto e a troca de informações, denomina-se trabalho cooperativo apoiado por computador (CSCW – *Computer Supported Cooperative Work*) ou *groupware* (BORGES, CAVALCANTI e CAMPOS, 1995). De acordo com Fluckiger (1995), CSCW concentra o desenvolvimento de sistemas baseados em computador para dar suporte e melhorar o trabalho de grupos de usuários com interesses, objetivos ou atividades comuns.

Ellis, Gibbs e Rein (1991) definem *groupware* como sistemas baseados em computador que apóiam a cooperação entre um grupo de pessoas, e, assim, possibilitam o trabalho em conjunto e a troca de informações, provendo uma interface para um ambiente compartilhado. Através desses sistemas um grupo de pessoas pode colaborar, comunicar suas idéias, coordenar suas atividades, resolver problemas, compartilhar informações e negociar significados (BRINCK, 1998).

Trabalhos realizados na área de CSCW podem ser caracterizados por três grandes desafios a serem vencidos: comunicação, coordenação e colaboração (CHANG, ZHANG e JIANG, 2001). Os ambientes implementados baseados em CSCW devem considerar esses desafios para prover as funcionalidades necessárias ao trabalho cooperativo.

As *wikis*, por terem características voltadas para a colaboração e cooperação de conteúdos, ganham espaço como ferramentas que apóiam o trabalho cooperativo, oportunizando as pessoas a compartilharem idéias e experiências promovendo a interatividade e o conhecimento. A ferramenta deixou de ser apenas um modelo de indexação e formatação de textos para ser um local com espaço para a criação de diálogos e debates (PRIMO e RECUERO, 2003).

O trabalho cooperativo permite a complementação de capacidades, de conhecimentos e de empenho, e a interação entre pessoas com características complementares. Um grupo tem mais capacidade de propor idéias, levantar as vantagens e desvantagens, selecionar as viáveis e tomar decisões (FUKS, RAPOSO e GEROSA, 2003).

No desenvolvimento de documentação para arquitetura de software esses fatores são importantes, por permitir que os desenvolvedores e projetistas produzam informações mais detalhadas do que se atuassem individualmente. O uso de ferramentas colaborativas proporciona ao grupo de desenvolvedores e projetistas a percepção da real situação da documentação do sistema, principalmente, com o acesso a ferramenta wiki que permite criar

documentos baseados em padrões para documentação da arquitetura, conforme descrito no capítulo 5.

Para que ocorra o trabalho colaborativo em um grupo, é necessário que seus integrantes se comuniquem através da troca de informações em um ambiente de cooperação, e essas sejam gerenciadas pela coordenação. A comunicação estabelecida através das trocas de informações gera acordos que são administrados pela coordenação, que por sua vez organiza e dispõe as tarefas que são executadas na cooperação.

Ao cooperar os desenvolvedores têm necessidade de se comunicar para negociar e tomar decisões sobre situações não previstas. Através da percepção, o desenvolvedor se informa sobre o que está ocorrendo e sobre o que os outros integrantes do grupo estão fazendo, coletando informações pertinentes para suas atividades (FUKS, RAPOSO e GEROSA, 2003). A Figura 3 mostra a relação dos princípios de comunicação, coordenação, cooperação e percepção, a qual norteia o processo de colaboração em um grupo de trabalho.

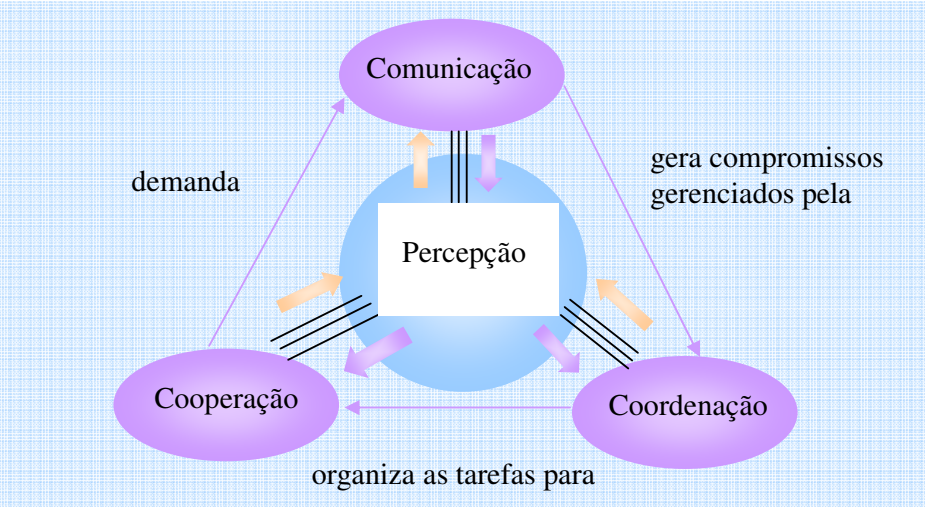

Figura 3: Diagrama do modelo 3C de Colaboração

A comunicação possibilita a troca de conhecimento e a negociação de tarefas. Através da coordenação, os membros e as atividades recebem apoio em caso de conflito, perda de comunicação e de cooperação. A cooperação é o trabalho conjunto dos membros do grupo imbuídos na realização de uma tarefa em um ambiente compartilhado. Através de recursos de percepção o indivíduo obtém retorno de suas ações e de seus colegas.

3.2.2.1 Comunicação

Comunicação é a ação de trocar informações para que ocorra um acordo comum entre as idéias discutidas. De acordo com Fuks, Raposo e Gerosa (2003), na colaboração, a comunicação é direcionada para a ação. Os integrantes de um grupo de trabalho trocam idéias, apresentam seus argumentos com a finalidade de gerar novos conhecimentos em um determinado contexto.

Segundo Fuks (2000), no seu princípio, comunicação era intrinsecamente ligada à cooperação, mas com sua evolução a comunicação oportunizou novas técnicas como auto-expressão. A interação de um grupo de trabalho depende da informação e da eficiência da comunicação entre seus membros (MELODY, 1994).

Através da comunicação, o grupo debate pontos de vista para alinhar e refinar as idéias, o que é fundamental para que o grupo consiga realizar tarefas interdependentes, não completamente descritas ou que necessitem de negociação (FUSSEL et al., 1998).

As ferramentas *wikis* possibilitam a comunicação assíncrona entre os integrantes de um grupo, por isso, devem permitir elementos de percepção de forma que as tarefas cooperativas sejam coordenadas. Os elementos perceptíveis são as funcionalidades previstas para o ambiente ou as interações que nele ocorrem, e devem se tornar conhecidos para que o andamento do trabalho do grupo aconteça de maneira correta (ASSIS, 2000).

A comunicação assíncrona gerada pelas ferramentas *wikis* permite a edição colaborativa de documentos entre os integrantes de um grupo de trabalho sem restrição de tempo, ou seja, eles podem cooperar sempre que tiverem novas informações que oportunize novos conhecimentos ao grupo.

3.2.2.2 Coordenação

A comunicação baseada em ação gera acordos de idéias. Para que se estabeleçam estes acordos e se realize o trabalho colaborativo, é de fundamental importância a coordenação das atividades. Segundo Raposo et al. (2001), a coordenação organiza o grupo para evitar que esforços de comunicação e de cooperação sejam perdidos e que as tarefas sejam realizadas de maneira correta e no tempo previsto.

A coordenação de atividades desempenhada pelo grupo é necessária para garantir o cumprimento dos acordos estabelecidos na comunicação e a realização do trabalho colaborativo através da soma das tarefas de cada integrante.

Um coordenador é definido com a intenção de administrar o grupo nas atividades de cooperação. O coordenador gerencia os integrantes e tarefas e analisa o desenvolvimento dos processos de trabalho. Alguns grupos cumprem com suas obrigações, organizando as atividades de cooperação por área de interesse, diminuindo as informações repassadas ao coordenador do ambiente (DRON, BOYNE e MITCHELL, 2001). Esta visão é apropriada para grupos pequenos, coesos e com participantes competentes e comprometidos (TELES, 2004).

Nas ferramentas *wikis* a coordenação é mediada pelo gerente do ambiente, o qual define as atividades para cada integrante do grupo e monitora cada interação, analisando os conteúdos publicados.

Um ponto relevante que deve ser considerado nas ferramentas *wikis* é a quantidade de conteúdos que o coordenador analisa, a fim de que as informações publicadas por seus autores e co-autores estejam devidamente corretas. Com isso, os elementos de percepção devem ser projetada de forma que auxilie a tomada de decisão frente algum problema na publicação de conteúdos.

3.2.2.2 Cooperação

Cooperação é a ato de trabalhar conjuntamente em um ambiente compartilhado buscando a troca de informações. Em um ambiente de cooperação, os integrantes de um grupo criam e alteram documentos com o objetivo de compartilhar novos conhecimentos em um determinado contexto (ASSIS, 2000).

O trabalho cooperativo além de viabilizar a complementação de capacidades, do auxílio mútuo e da motivação advindos da colaboração, os novos profissionais são preparados a se relacionar, a negociar, a se expor, a liderar, a ter responsabilidade, e a se comunicar, coordenar e cooperar (FUKS, 2000).

De acordo com Motta e Borges (2000), as ferramentas de colaboração devem possibilitar a avaliação dos objetos no repositório compartilhado, de modo a dar subsídio a um sistema de recomendação.

As ferramentas *wikis* proporcionam o controle de versão, possibilitando recuperar edições anteriores, comparar conteúdos entre versões e alterar informações. Este recurso deve ser considerado em atividades de cooperação, pois em caso de problemas é preciso recuperar a última versão, e ainda, identificar o autor, a data e as alterações realizadas nas páginas, a fim de que sejam tomadas medidas de segurança ao ambiente de trabalho.

As interações do grupo são registradas no histórico de páginas, que representam as atividades de cooperação realizadas no ambiente. Quando um colaborador está editando uma página wiki, ela fica bloqueada para acesso simultâneo, para que não haja conflito nos dados que serão publicados.

3.2.2.2 Percepção

Perceber é adquirir conhecimento, por meio dos sentidos, do que está acontecendo e do que as outras pessoas estão fazendo, mesmo sem se comunicar diretamente com elas (BRINCK e MCDANIEL, 1997). Para permitir uma colaboração eficiente, a percepção torna-se um fator fundamental na comunicação, coordenação e cooperação de um grupo de trabalho. Segundo Fuks e Assis (2001), perceber as atividades dos outros indivíduos é essencial para o fluxo e naturalidade do trabalho e para diminuir as sensações de impessoalidade e distância, comuns nos ambientes virtuais.

As informações de percepção fornecidas pelo acompanhamento da colaboração são especialmente úteis para o coordenador do grupo, que precisa saber, quais integrantes estão cooperando com as atividades do ambiente e como as informações são disponibilizados, a fim de que o grupo possa adquirir novos conhecimentos.

Os integrantes de ambientes colaborativos buscam nos elementos de percepção as informações necessárias para montar seu contexto de trabalho e identificar as intenções dos companheiros do grupo, de maneira a fornecer ajuda quando necessário (BAKER et al., 2001).

Os Recursos de percepção proporcionam o entendimento compartilhado. O entendimento fornecerá elementos para a coordenação de atividades dentro do ambiente e possibilitará a cooperação. Os elementos de percepção devem ser associados aos mecanismos de comunicação, coordenação e cooperação, de modo que o ambiente proporcione o trabalho colaborativo.

4 FERRAMENTAS WIKI

Neste capítulo são abordadas ferramentas *wikis*, utilizadas para apoiar o trabalho colaborativo na web.

As ferramentas *wikis* possuem como característica a possibilidade de que documentos sejam editados de forma cooperativa, utilizando uma linguagem simples, através de um navegador web, e também a geração de novos conhecimentos de forma compartilhada em qualquer tempo e lugar.

Entretanto, o uso incorreto dessas ferramentas pode gerar a perda de informações, quando os integrantes do grupo não estão envolvidos no processo de cooperação, comunicação e coordenação, que são fundamentais para o trabalho colaborativo.

A seguir apresentam-se algumas ferramentas *wikis* que apóiam o trabalho colaborativo, permitindo que integrantes do grupo possam cooperar na edição de documentos.

4.1 TWiki

O Twiki é uma ferramenta de escrita colaborativa na WEB, que permite a interação de várias pessoas, compartilhando informações em um processo de cooperação em grupo. Através do Twiki, é possível desenvolver documentação em formato de hipertexto através da web, de uma forma dinâmica e rápida.

Para que a colaboração ocorra, é necessário que a ferramenta possua características que facilitem a interação entre os usuários presentes nesse ambiente e forneçam serviços para comunicação, cooperação e coordenação.

Segundo Schmitt (2006), mesmo que o ambiente seja extremamente útil para o trabalho colaborativo e possua uma interface amigável para que os participantes possam interagir na construção de conteúdos, existe uma deficiência no que diz respeito às atividades de coordenação e comunicação. Isto se explica porque a idéia original da ferramenta Twiki é, justamente, permitir uma construção absolutamente igualitária e dinâmica de hipertextos. Já na última versão, a ferramenta contempla os serviços de comunicação, como por exemplo, mensagens através de email e comentários adicionados no perfil de cada usuário.

Na ferramenta são permitidos vários acessos concorrentes a uma página, o que gera um armazenamento das alterações feitas simultaneamente. No entanto, os participantes não podem decidir qual é o conteúdo mais adequado, ou se a página deve ser utilizada por mais de

um usuário ao mesmo tempo. Assim, uma página editada por mais de um usuário deve ser analisada, afim de que se obtenha uma solução do conteúdo publicado.

O controle de acesso à página no Twiki permite que se definam, para cada página, os usuários que podem modificá-la ou visualizá-la. Essa restrição é feita por comentários textuais dentro da própria edição da página e depende da criação de usuários e da definição de grupos, feita pelo administrador, como pode ser visualizado na Figura 4.

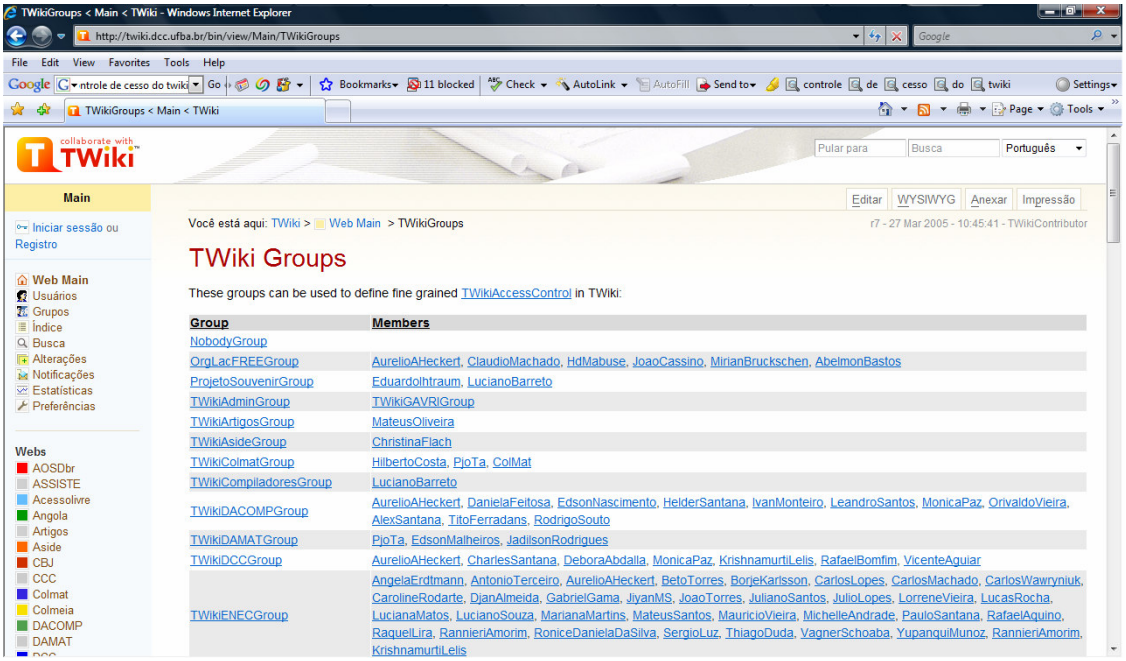

Figura 4: Tela de grupos da Twiki da Universidade Federal da Bahia (UFBA)

Cada membro do ambiente Twiki possui sua própria lista sobre quais são os membros desse grupo, a cada admissão de um novo integrante, ou exclusão de um membro, essa lista é atualizada e as informações são disponibilizadas para os integrantes do grupo através do serviço de *WebNotify* que se encarregará de repassar o email aos usuários com as devidas modificações, sem a necessidade de conferir se algo mudou.

O serviço de *WebNotify*, pode ser visto como o coordenador das atividades realizadas pelo grupo no ambiente wiki, tanto na atualização dos membros dos grupos quanto na edição ou exclusão de conteúdos, garantindo que as tarefas de cooperação sejam enviadas à todos integrantes. A comunicação é mediada por troca assíncrona de mensagens textuais de correio eletrônico e comentários. Já a cooperação acontece na edição colaborativa de uma página wiki, buscando a troca de informações de forma compartilhada.

Outra característica do Twiki é o recurso de modelos que é utilizado para desenvolver uma versão multi-linguagem da ferramenta a fim de atender a diversidade de usuários com

idiomas diferentes. Assim, o ambiente permite a personalização da interface de acordo com o idioma escolhido, proporcionando aos usuários mais segurança no compartilhamento de informações.

4.2 MediaWiki

A Mediawiki tem um conjunto de características que englobam a possibilidade de geração automática de *links*, a edição do conteúdo por qualquer usuário, a existência de regras de formatação de texto e de *hiperlinks* para navegação (PONTES, SOUZA e BARBOSA, 2005). Também possui alguns mecanismos de defesa, onde o administrador do sistema pode restringir a leitura ou edição de conteúdo a grupos específicos de usuários, bem como o bloqueio de acesso por determinados endereços IP.

De acordo com Ebersbach et al (2005), a Mediawiki é um software livre com licença GNU GPL[1] que tem como uma de suas principais características, a facilidade de uso, até mesmo por usuários sem conhecimento em desenvolvimento de páginas web.

A ferramenta apresenta características relevantes de segurança, como histórico de página que contém informação sobre a data, a hora, o nome do usuário e o IP de acesso, relativas a todas as alterações realizadas, detalhadas por página e por autor; visualização da diferença entre quaisquer versões de páginas e controle de permissões para os diferentes tipos de usuários ou grupos de usuários.

Quando o usuário se registra em um ambiente wiki é criada uma conta de acesso, que ao cooperar, tanto na forma de criação de novas páginas ou na alteração das já existentes, é armazenado seu perfil pelo controle de versões que gera um histórico de todas as contribuições realizadas por usuário, e ainda, possibilita a recuperação da última versão caso as informações sejam deletadas.

A partir da autenticação o controle de versão é quem se encarregará de armazenar todas as interações dos usuários com o ambiente, disponibilizando informações ao grupo pelo histórico de páginas.

A Mediawiki oferece a possibilidade dos usuários cadastrados criarem e editarem suas páginas e alterarem outras páginas, contribuindo com novas informações. Ao criarem uma

[1] GNU General Public License (Licença Pública Geral), GNU GPL ou simplesmente GPL, é a designação da licença para software livre idealizada por Richard Stallman no final da década de 1980, no âmbito do projeto GNU da Free Software Foundation. A GPL é a licença com maior utilização por parte de projetos de software livre.

nova conta, cada usuário cadastrado no sistema tem permissão para utilizar funcionalidades específicas, como possuir um perfil próprio, criar filtros específicos e ter um maior controle sobre seus conteúdos de interesse. Para cadastrar uma nova conta de usuário deve selecionar a opção *Login/create account*, conforme mostrado na Figura 5.

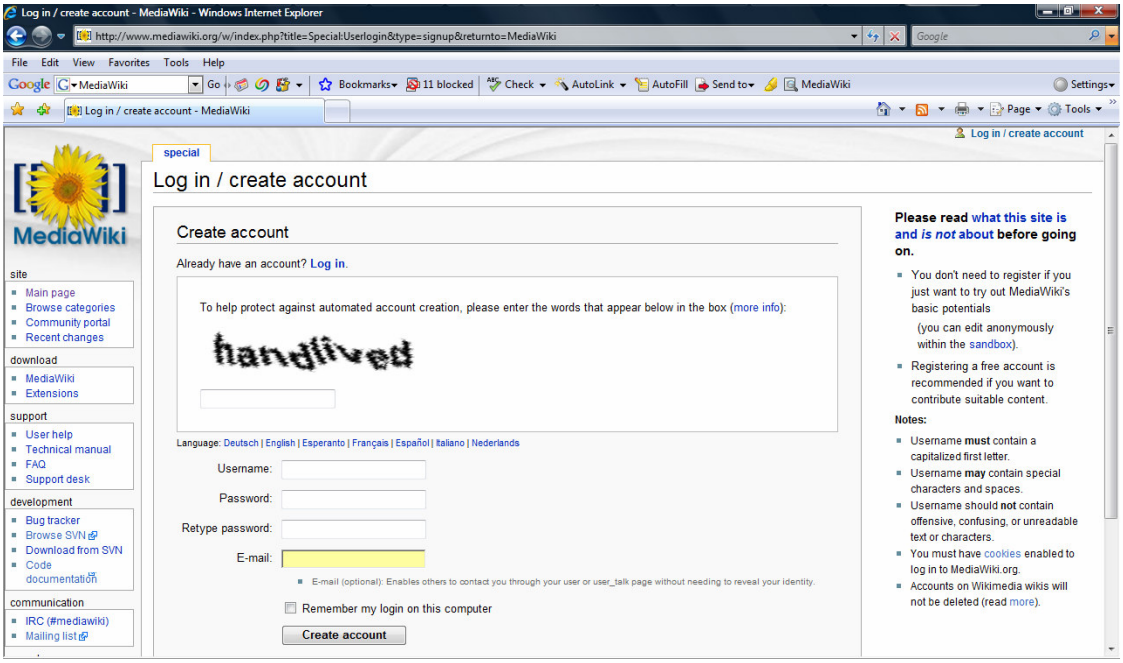

Figura 5: Tela de criação de usuário

Após a criação da conta e da definição de suas preferências de usabilidade, o usuário está habilitado para cooperar com o ambiente, produzindo novos conhecimentos ao grupo de trabalho.

Para maior segurança das informações publicadas, a Mediawiki, permite a restrição de acesso à criação de contas através de linhas de código implementadas na configuração da ferramenta, a fim de que impossibilite a entrada de intrusos. O administrador se responsabilizará em cadastrar todos os usuários no banco de dados do ambiente de maneira a proporcioná-los mais segurança.

Embora que a Mediawiki forneça recursos de segurança na criação de contas, trazendo mais tranqüilidade aos usuários do grupo, não traz certeza da possibilidade de invasão por usuários maliciosos, sendo necessário o uso de ferramentas específicas.

4.3 SnipSnap

A SnipSnap é um software livre desenvolvido em Java e disponibilizado sob licença GPL. A ferramenta é expansível por meio de macros e filtros. Os macros são extensões especiais para formatação de tabelas e códigos fontes, como por exemplo, definição dos parâmetros de exibição de um arquivo de imagens. Já os filtros são recursos utilizados para formatação de textos, como criação de títulos, subtítulos, *links*, entre outros.

A ferramenta wiki, SnipSnap, é um projeto realizado pelo instituto *Fraunhofer Gesellschaft,* que tem como objetivo o desenvolvimento de um ambiente que permita a edição colaborativa de documentos em grupo, contemplando as características de comunicação, cooperação e coordenação abordadas no CSCW.

Os serviços de comunicação, como *post comment,* são mostrados em todas as páginas do ambiente, permitindo que os usuários cadastrados possam incluir seus comentários, com a intenção de sugerir novas idéias. Os comentários oferecem informações de percepção para os outros membros do grupo ou para o próprio criador da anotação. O uso desse recurso permite que os usuários saibam quem interagiu, quando ocorreu essa interação e quem é o responsável por uma determinada ação.

Outro elemento de percepção disponível na ferramenta é o histórico de página, que armazena informações referentes às tarefas de cooperação realizadas pelos usuários em uma determinada página do ambiente, como pode ser visualizado na Figura 6.

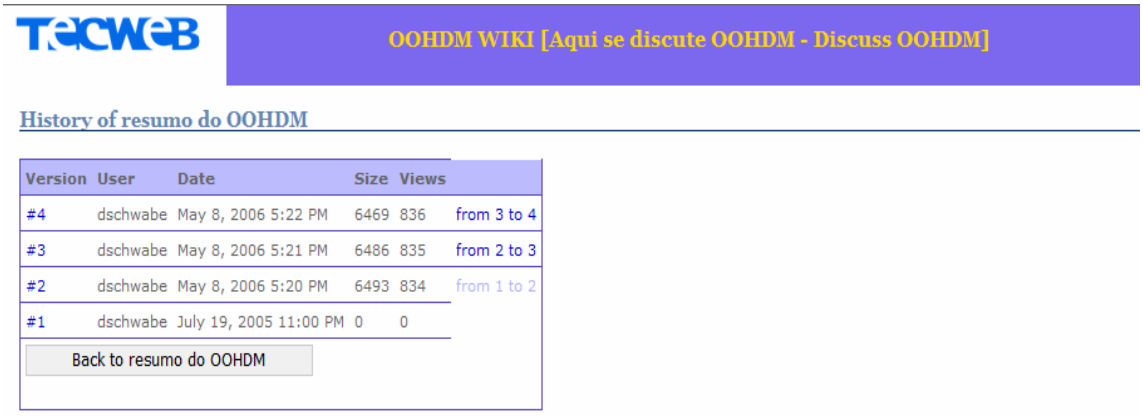

Figura 6: Tela do histórico de página do ambiente TECWEB.

O ambiente TECWEB, gerado pela ferramenta SnipSnap, possibilita a aquisição de conhecimento por meio da percepção das atividades de cooperação dos usuários, buscando assim, coordenar as interações e tornar o trabalho em grupo mais participativo. O coordenador

garante que os compromissos feitos durante o processo de comunicação sejam cumpridos, de forma que cada contribuição proporcione novos conhecimentos ao grupo.

A SnipSnap oferece funcionalidades para editar e exibir a estrutura e o conteúdo de documentos no ambiente, e para notificar o usuário através de informações de percepção, como por exemplo, restrição de acesso simultâneo na edição de páginas, permitindo que não haja conflito na publicação dos conteúdos.

4.3 XWiki

A Xwiki é um software livre para edição colaborativa de documentos na web, construída a partir de componentes que definem os recursos de comunicação, cooperação, coordenação e percepção, permitindo o trabalho colaborativo.

A ferramenta Xwiki oferece aos usuários à capacidade de desenvolver um ambiente colaborativo através de suas características básicas, como edição de páginas, controle de versão, administração de direitos, organização de documentos por domínio, uso de interface de programação de aplicativos (API), histórico de documentos, anexos de arquivos, pesquisa, adição de modelos para a interface do ambiente, e estendê-la através de *plugins*, conforme mostrado na Figura 7.

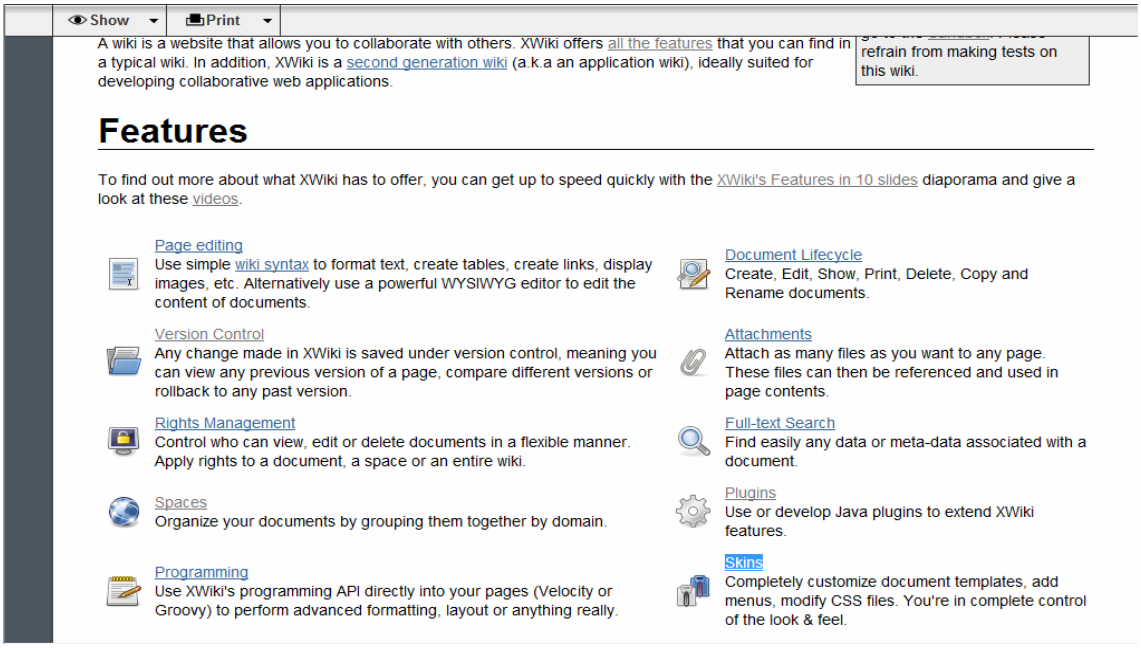

Figura 7: Tela principal da ferramenta Xwiki.

Os usuários do ambiente colaborativo, Xwiki, buscam nos elementos de percepção, como controle de versão e administração de direitos, informações claras sobre as atividades de cooperação e sobre as restrições de acesso de usuários e de grupos.

O histórico gerado pelo controle de versão possibilita o rastreamento do usuário através da observação de suas ações no ambiente, sendo possível saber suas informações pessoais e as ações executadas pelo mesmo durante a interação com a página wiki.

Quando o usuário faz uma requisição de colaboração a uma página que tem restrição para edição, deverá encaminhar ao coordenador do grupo uma solicitação de acesso para que possa cooperar com novos conteúdos. O coordenador utilizando o recurso, administração de direitos, cadastra as preferências do usuário selecionando o tipo de usuário e o grupo que fará parte, e ainda, selecionará o nível de permissão concedido a eles, como mostrado na Figura 8.

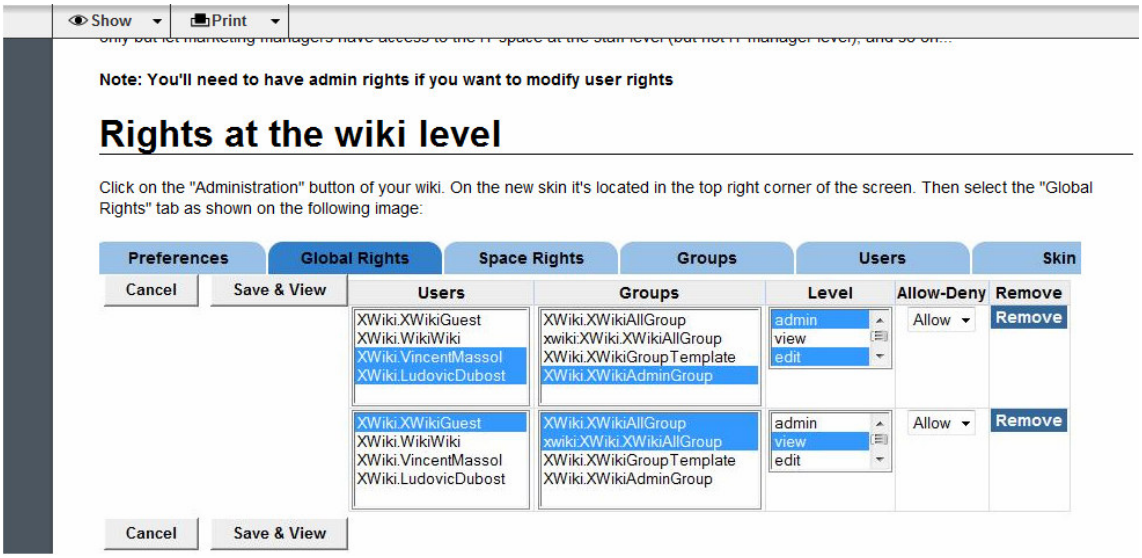

Figura 8: Tela de administração de direitos da wiki.

Utilizando este recurso será possível que os usuários conheçam a descrição das regras impostas pelo ambiente wiki, informando quais são os direitos dos usuários e dos grupos, proporcionando assim, mais segurança às informações.

Para possibilitar a coordenação do grupo são necessárias informações sobre as cooperações dos usuários com o ambiente, armazenadas pelo controle de versão. O acompanhamento da participação é necessário para o coordenador monitorar o andamento das atividades de modo a intervir quando necessário e para os usuários compararem e ajustarem suas contribuições.

A cooperação através da edição de página viabiliza que os usuários compartilhem idéias e se comuniquem na obtenção de um objetivo comum, produzindo resultados

satisfatórios no desempenho das tarefas propostas pelo grupo. As tarefas originam-se dos acordos estabelecidos durante a comunicação, devendo ser cumpridas pela cooperação entre os usuários e retornadas por elementos de percepção.

Através dos elementos de percepção oferecidos pelo ambiente Xwiki, eventos importantes, como atividades de cooperação e regras de acesso podem ser percebidas e interpretadas pelos usuários, sem que haja descontinuidade no processo de colaboração. O acesso a essas informações, torna-se imprescindível para que o coordenador e os usuários possam conhecer os integrantes e as tarefas que desempenham.

As informações armazenadas em uma página wiki devem ser mantidas e gerenciadas apenas por usuários que tenham permissão e que sejam de confiança do grupo. Isso se deve de modo a não infringir os conteúdos publicados.

O Xwiki oferece uma interface que disponibiliza recursos, como *wysiwyg*, *objects*, *class*, *access rights* e *history* na edição de uma página wiki, conforme ilustrado na Figura 9.

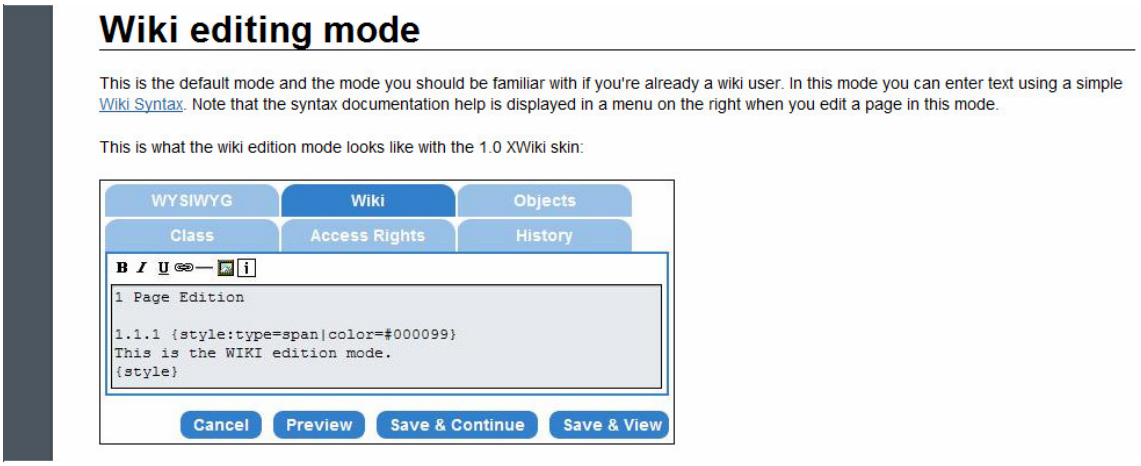

Figura 9: Tela de edição de página

Nessa figura pode-se observar que: 1) uma página wiki no modo de edição disponibiliza elementos de percepção importantes para a comunicação, coordenação e cooperação entre o grupo de trabalho. 2) wysiwyg dispõe de uma interface com alguns recursos de formatação de um editor comum. 3) *objects e class* oferecem um conjunto de propriedades que adicionam informação a uma página wiki, como por exemplo, direitos de segurança. 4) *access rights* mostra a configuração dos direitos de acesso para usuários e grupos. 5) *history* apresenta os registros de interação dos usuários com a página wiki.

O serviço de pesquisa, *full-text search*, permite que benefícios sejam oferecidos aos usuários, como a possibilidade de mostrá-los o nome da página referenciado pela busca, a

organização por domínio de conteúdo, configurado através do recurso *spaces,* a data de atualização e o nome do último usuário que contribuiu.

O recurso, *Space Rights*, permite que os usuários organizem seus documentos por domínio de conteúdo, facilitando a administração das informações e dos direitos de usuários e grupos. Através desse recurso é possível cadastrar quais usuários e grupos terão acesso a um determinado domínio do ambiente, como mostrado na Figura 10.

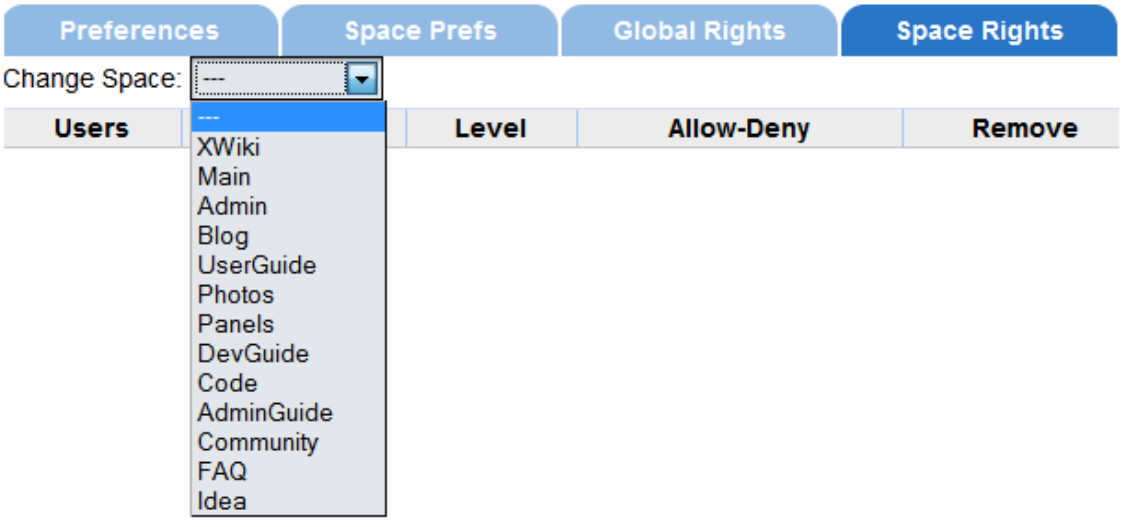

Figura 10: Tela de edição de direitos de acesso, *Space Rights.*

Várias vantagens são apresentadas através do recurso, *Space Rights*: i) o usuário fica ciente sobre quais usuários e grupos fazem parte de cada domínio e quais são suas permissões; ii) inspira mais confiabilidade aos usuários que, ao tomarem conhecimento de que o ambiente possui regras de proteção, se sentem mais seguros para publicarem suas idéias, e iii) impede que usuários não autorizados possam editar ou alterar conteúdos.

Todas as ferramentas *wikis* apresentadas neste capítulo, se utilizadas corretamente podem permitir o compartilhamento de informações em um ambiente de trabalho colaborativo mediadas pelos elementos de comunicação, cooperação, coordenação e percepção.

Nos próximos capítulos serão apresentados os trabalhos desenvolvidos e os resultados obtidos, tendo como base a pesquisa bibliográfica, de modo a desenvolver trabalhos relevantes à documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens.

5 CONSTRUINDO PADRÕES DE DOCUMENTAÇÃO DA ARQUITETURA DE SOFTWARE

A partir de conclusões obtidas através de pesquisas realizadas em documentação de software, foi possível notar que a maioria das ferramentas para esse fim, não define Padrões com regras claras a um domínio específico de aplicação, tornando-as muito genéricas na visão dos projetistas de software.

Merson (2000) relata que os engenheiros de software possuem experiência no desenvolvimento de software, conhecimento em tecnologias e *design patterns*, mas pecam pelo fato de não criarem e publicarem a documentação da arquitetura de software de forma adequada. Ainda, sua pesquisa demonstra a importância da criação de *templates* de documentação da arquitetura que atendam às necessidades de cada domínio de aplicação e do princípio básico, o qual define que a documentação da arquitetura deve ser baseada em visões.

Um trabalho feito pela equipe de pesquisadores do *Software Engineering Institute* mostra que para documentar uma arquitetura em camadas é necessário catalogar as propriedades dos componentes e suas relações (BACHMANN et al., 2000).

Com base nessas pesquisas, ressalta-se que tais documentos informem aos projetistas regras claras e específicas ao domínio do software. Também, devem fornecer as características relevantes de cada componente e suas interações no contexto de desenvolvimento da arquitetura do sistema.

Devido a isso, é proposta uma padronização para a documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, de maneira que possibilite os projetistas documentarem e publicarem suas arquiteturas de forma clara e objetiva com o propósito de reuso. A criação desses Padrões foi baseada na visão lógica e de caso de uso da UML, capítulo 3.

Para padronizar a documentação foram utilizados Padrões, que de acordo com Borches (2000), podem ser compreendidos como uma forma de armazenar e disponibilizar conhecimento por meio de textos e esboços em um formato estruturado, cuja solução é de sucesso já que os mesmos podem ser aplicados em problemas comuns em um determinado contexto. Gamma et al. (1995) relata que os padrões de projeto capturam soluções que foram criadas e implementadas ao longo do tempo. Para Alexander, Ishikawa e Silverstein (1977) padrão é uma solução de sucesso para um problema recorrente em um determinado contexto. Coplien e Harrison (2004) apresentam o padrão como uma configuração estrutural recorrente que resolve um problema em um determinado contexto.

Os padrões de documentação da arquitetura, desenvolvidos neste trabalho, abordam assuntos referentes ao modelo de arquitetura em camadas, isto é, mostram as características de reuso relevantes dos componentes e suas interações na construção da modelagem da arquitetura.

5.1 Padrões e suas características

Os padrões são utilizados em várias abordagens e definidos por diferentes autores em suas respectivas áreas de atuação, no entanto todas as definições mostram um principal objetivo para os padrões: o seu reuso (LOBATO e ZORZO, 2007b).

Para que determinadas regras venham a ser um padrão, essas devem apresentar usos conhecidos que comprovem sua eficácia na resolução do problema (OLIVEIRA, BALBY e GIRARDI, 2004). Dessa forma, o objetivo dos padrões é apresentar um problema e propor a solução, o contexto no qual o problema ocorre, restrições que levam à diferentes considerações durante a elaboração da solução e as conseqüências de uso do padrão, documentando a essência da solução, para que essa possa ser utilizada inúmeras vezes em situações similares.

Fazendo uso de padrões, neste caso, de documentação da arquitetura, ocorre um ganho no desenvolvimento de padronização de documentos para a arquitetura de sistemas e *frameworks* para processamento e análise de imagens, já que as soluções encontram-se desenvolvidas e comprovadas suas relevâncias. Esses poderão ser seguidos, permitindo uma otimização de tempo na criação de documentação gerado pelos desenvolvedores, uma vez que as regras são claras e de fácil entendimento.

Com a reutilização dos padrões, aumenta-se a qualidade da documentação da arquitetura de sistemas, já que as soluções foram elaboradas frente a um problema recorrente, comprovadas e testadas, diminuindo o risco de construir uma documentação ineficiente aos desenvolvedores.

5.2 Metodologia para formatação dos padrões

Na pesquisa realizada sobre documentação da arquitetura de sistemas e *frameworks* para processamento e análise de imagens, foi observado que algumas visões arquiteturais não tinham sido encontradas na padronização da documentação. Baseado nessa observação, para a

definição dos padrões foram utilizados algumas dessas visões a fim de definir uma documentação que apresente regras claras aos desenvolvedores de software.

O formato e estilo dos padrões foram baseados na "Linguagem de Padrões para escrita de Padrões" de Meszaros e Doble (1996), onde é evidenciado que os padrões são mais fáceis de entender e aplicar quando alguns elementos estão presentes no seguinte formato, como:

- **Nome**: descreve um nome que referencia o objetivo principal do padrão;
- **Contexto**: descreve em que circunstâncias o problema ocorre;
- **Problema**: descreve o problema que o padrão soluciona;
- **Forças**: descreve as considerações positivas e negativas que influenciam o uso do padrão;
- **Solução**: descreve claramente o que é necessário para resolver o problema;
- **Conseqüências**: descreve as conclusões da aplicabilidade do padrão;
- **Usos Conhecidos**: descreve exemplos conhecidos da aplicação do padrão.

Com a utilização desses padrões, os desenvolvedores sentem-se mais seguros ao criarem a documentação de suas arquiteturas, uma vez que possuem informações bem claras e objetivas do que catalogar.

5.3 Catálogo de padrões definidos.

Os padrões definidos foram organizados de acordo com as notações da visão lógica e de caso de uso da UML, formando o catálogo de padronização para a documentação da arquitetura. A organização desses Padrões pode ser visualizada na Figura 11.

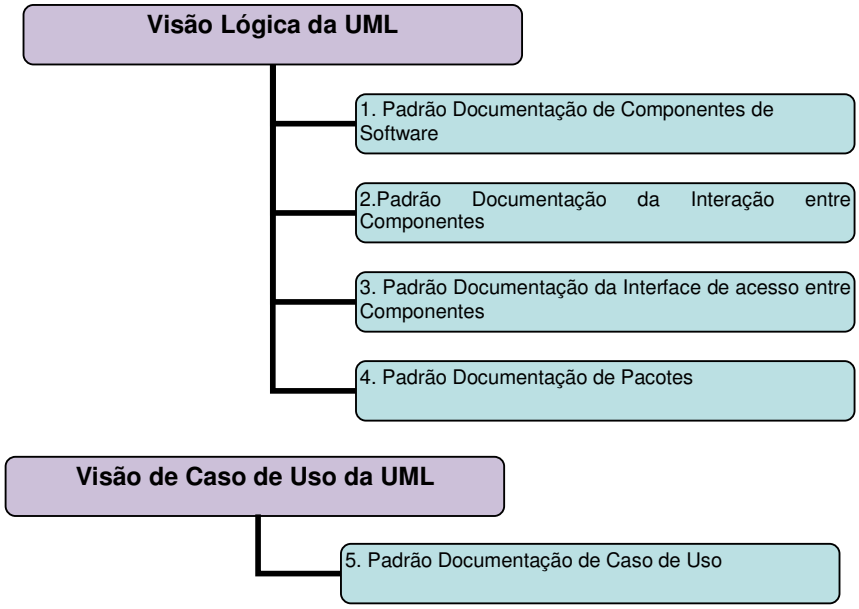

Figura 11: Definição dos Padrões para Documentação da Arquitetura

1 - Nome: Padrão Documentação de Componentes de Software

Contexto:

A documentação das técnicas de processamento e análise de imagens deve abordar as características de reuso dos componentes, de forma a mostrar quais são os problemas encontrados neste domínio e como as soluções são aplicadas.

Problema:

Como documentar componentes de sistemas e frameworks para processamento e análise de imagens especificando as características de reuso?

Forças:

1. A área de processamento e análise de imagens apresenta vários problemas, como suavização de imagens ruidosas, detecção de bordas e correção de iluminação, entre outros, que precisa utilizar técnicas de processamento e análise de imagens com o objetivo de resolvê-las. Dessa forma, é de vital importância a documentação detalhada de seus componentes, à medida que os projetistas necessitam reutilizá-los;
2. Os componentes de software para processamento e análise de imagens de imagens devem ser bem documentados, com o intuito de auxiliar o projetista a compreendê-los e utilizá-

los em outras aplicações que precisam implementar técnicas de processamento e análise de imagens. Na documentação dos componentes é importante constar características de reuso, ou seja, um conjunto de informações que descreve o problema, a funcionalidade do componente, atributos da imagem fonte, operações sobre imagens, parâmetros de entrada do componente, interface de acesso, métodos matemático, dentre outros;

3. As técnicas de processamento e análise de imagens estão baseadas em métodos matemáticos, os quais precisam ser considerados na documentação, uma vez que, definem as operações de aplicabilidade da técnica;

4. Quanto mais complexas são as técnicas de processamento e análise de imagens, mais difícil se torna a sua documentação quando não se têm uma padronização específica para o domínio da aplicação.

Solução:

Defina o problema encontrado em imagens digitais bem como a técnica de processamento e análise de imagem utilizada para resolvê-lo. É importante ainda documentar as características de reuso do componente, tais como: nome, funcionalidade, interface de acesso dos componentes, operadores do método/técnica, parâmetros de entrada da técnica, atributos da imagem fonte, métodos matemáticos, atributo da imagem resultado, resultado da aplicação da técnica, de modo que o projetista possa compreender melhor sua funcionalidade.

A descrição abordada na Tabela 1, mostra como é aplicada a solução para o problema de documentação de componentes de software para área de processamento e análise de imagens.

Documentação de Componentes
Definição do problema em imagens: Realce de bordas
Técnica de processamento de imagens: Segmentação de imagens por detecção de bordas
Nome do componente: MethodSegEdgeDetection

Funcionalidade do componente:

O componente executa a segmentação de imagens digitais baseado na detecção de contornos. Essa técnica utiliza máscaras de convoluções (kernels) para detectar as bordas da imagem. Para que essa detecção ocorra é necessário executar duas convoluções na imagem, sendo que a primeira detecta as bordas horizontais e a segunda detecta as bordas verticais. Após a aplicação dessas máscaras, um processo de limiarização global é executado com o objetivo de definir claramente o que é borda e o que é plano de fundo da imagem.

Interface de acesso:

MethodInterface

Operadores do método/técnica:

O método utiliza operadores de gradiente, como Robert, Sobel, Prewitt, FreiAndChen.

Parâmetros de entrada da técnica:

(0): Integer - Flag que indica o tipo de máscara que será utilizada

- (0) RobertsSegmentation (3x3)
- (1) SobelSegmentation (3x3)
- (2) PrewittSegmentation (3x3)
- (3) FreiAndChenSegmentation (3x3)

(1): Integer - Threshold (Limiar)

Atributo da imagem fonte:

(0): PlanarImage - imagem a ser segmentada

Métodos matemáticos

$ImagemDestino(x,y) = (Imagem1(x,y))^2 + (Imagem2(x,y))^2$

Atributo da imagem resultado

Tabela 2: Exemplo da Aplicação do Padrão 1

Conseqüências:

Com a documentação das características de reuso dos componentes o projetista torna-se mais confiante em relação às soluções aplicadas a certos problemas e a usabilidade em outros sistemas de mesmo domínio.

Usos conhecidos:

O framework Athena de Freitas (2006) e o sistema Arthemis[2] disponibilizam aos projetistas a documentação dos componentes, a qual referencia soluções para problemas recorrentes na área de imagens.

[2] Software de Processamento e Análise de Imagens em desenvolvimento pelo GPIM – Grupo de Processamento de Informações Multimídia – Universidade Federal de Santa Maria.

2 – Nome: Padrão Documentação da Interação entre Componentes

Contexto:

Na modelagem da arquitetura em camadas de sistemas e frameworks para processamento e análise de imagens é de fundamental relevância documentar as mensagens de comunicação entre os componentes, pois, é através delas que o desenvolvedor irá entender o funcionamento do sistema.

Problema:

Como documentar as técnicas de processamento e análise de imagens através da especificação das mensagens de interação entre componentes, representadas no diagrama de comunicação da UML?

Forças:

1. Na documentação da interação entre as camadas de uma arquitetura de sistemas e frameworks para processamento e análise de imagens, é possível registrar as mensagens de comunicação entre os componentes pertencentes às camadas distintas da arquitetura;
2. A documentação do fluxo de controle das técnicas de processamento e análise de imagens permite ao desenvolvedor entender seu processo de funcionamento, através da especificação dos componentes participantes e das mensagens de execução de operações entre os componentes;
3. As mensagens ilustradas no diagrama de comunicação devem ser bem documentadas, uma vez que, são elas que conduzem as operações que os componentes devem executar para a realização das técnicas de processamento e análise de imagens;
4. A utilização do Padrão de documentação da arquitetura possibilita aos desenvolvedores, regras claras do que é necessário para catalogar o funcionamento das técnicas de processamento e análise de imagens, facilitando assim, a criação de documentos relevantes do sistema.

Solução:

As atividades descritas abaixo são baseadas no diagrama de comunicação da UML, que ilustra a interação dos componentes, através das mensagens de comunicação do fluxo de controle, conectando os componentes participantes através das operações solicitadas.

1. Definir o nome da técnica de processamento e análise de imagens;
2. Citar o caso de uso da funcionalidade e nomeá-lo;

3. Especificar o ator do cenário de caso de uso;
4. Elencar os componentes e suas mensagens de interação com outros componentes na realização de uma funcionalidade especificada no caso de uso.

Aplicação do Padrão:

Documentação da Interação entre os Componentes

Técnica de processamento e análise de imagens:
Segmentação de imagem por detecção de bordas

Definição do caso de uso:
Segmentar por contorno de bordas

Ator da aplicação da técnica:

Usuários de sistemas de processamento e análise de imagens

Fluxo de controle:

O componente **Nome** envia a mensagem:
Número da mensagem: mensagem (parâmetro) para o componente **Nome**

Fluxo 1:

O componente `principal` envia a mensagem:
`1: instancia()` para o componente `guiprincipal`;

Fluxo 2:

O componente `guiprincipal` envia uma mensagem:
2: `adicionaMenu("AbrirArquivo")` para ele mesmo;

Fluxo 2.1 e 2.2:

O componente `guiprincipal` envia a mensagem:
2.1: `escolheComponente(1)` para o componente `strategyComponente`;
2.2: `principalAbrir()` para o componente `principalAbrirArquivo`;

Fluxo 3 e 3.1:

O componente `principalAbrirArquivo` envia a mensagem:
3: `principalExibirImagem(imagem,diretorio)` para o componente `principalExibirImagem`;
3.1: `instancia()` para o componente `guiExibirImagem`;

Fluxo 3.2:

O componente `guiExibirImagem` envia a mensagem:
3.2: `atualizaImagem(imagem)` para o componente `acessoCamadaInferior`;

Fluxo 4, 4.1 e 4.2:

O componente `guiprincipal` envia a mensagem:
4: `adicionaMenu("Segmentação por contornos")` para ele mesmo;
4.1: `escolheComponente(3)` para o componente `strategyComponente`;
4.2: `principalGerarMascara()` para o componente `principalGeraMascara`;

Fluxo 5:

O componente `principalGeraMascara` envia a mensagem:
5: `instancia()` para o componente `guiGerarMascara`;

Fluxo 6:

O componente `guiGerarMascara` envia a mensagem:
6: `atualizaDadosEntrada(tamanho, tipo, limiar)` para o componente `acessoCamadaInferior`;

Fluxo 7:

O componente `componenteSegmentacaoContorno` envia a mensagem:
7: `getParametros()` para o componente `acessoCamadaInferior`;
7.1: `getImagem()` para o componente `acessoCamadaInferior`;
7.2: `executaSegmentacao()` para ele mesmo
7.3: `setImagem(resultado)` para o componente `acessoCamadaInferior`;

Fluxo 8:

O componente `acessoCamadaInferior` envia a mensagem:
8: `avisaObserver(resultado)` para o componente `observer`.

Tabela 3: Exemplo da Aplicação do Padrão 2

Conseqüências:

Com a utilização da documentação da interação entre os componentes, os projetistas ficam mais seguros quanto ao funcionamento das técnicas de tratamento de imagem, uma vez que as mensagens dos fluxos de controle definem as operações entre os componentes.

Usos conhecidos:

O framework Athena de Freitas (2006) e os sistemas Arthemis e MeSegHi de Lourega (2006) utilizam a documentação da interação entre os componentes para registrarem a interação entre os componentes que compõem as camadas da arquitetura.

3- Nome: Padrão Documentação da Interface de acesso entre Componentes

Contexto:

Atualmente, alguns desenvolvedores de sistemas e framewok para processamento e análise de imagem estão utilizando técnicas de desenvolvimento baseado em componentes nos seus projetos arquiteturais. Com isso, a documentação da interface entre os componentes se faz necessária, pois é por meio dela que ocorre o acesso de novos componentes ao sistema.

Problema:

Como documentar a interface entre os componentes de sistemas para processamento e análise de imagens especificando as características dos dados de acesso?

Forças:

1. O Uso da interface de acesso entre componentes na arquitetura em camadas de sistemas para processamento e análise de imagens possibilita que os projetistas conheçam apenas o componente e sua interface para implementar os algoritmos, ao invés, de entender toda a lógica do sistema. Na documentação da interface de acesso entre os componentes é importante constar características dos dados de acesso, ou seja, um conjunto de informações que descreve a funcionalidade da interface, domínio da aplicação na área de imagens, atributos da interface no âmbito de imagens, parâmetros das operações, dentre outros;
2. A documentação da interface de acesso entre os componentes, tem por objetivo, auxiliar o projetista a compreender como integrar os algoritmos das técnicas de processamento e análise de imagens ao sistema, de forma, clara e padronizada;
3. Os parâmetros de entrada da interface de acesso entre os componentes devem ser bem documentados, uma vez que, é através deles que os projetistas irão instanciar os dados para implementar os algoritmos de processamento e análise de imagens.

Solução:

A solução é definida pelos seguintes passos:

1. Definir o nome da interface de acesso;
2. Na descrição da funcionalidade da interface, deve constar seu objetivo e sua aplicabilidade;
3. Especificar o domínio de aplicação na área de imagens;

4. Descrever o nome, parâmetro e funcionalidade das operações.

Aplicação do Padrão:

Documentação da Interface de acesso entre os Componentes

Nome da interface de acesso: `acessoCamadaInferior`

Funcionalidade da interface de acesso: A classe `acessoCamadaInferior` tem o objetivo de prover uma interface de acesso aos componentes do Athena, um *framework* para a construção de interfaces humano-computador no domínio da segmentação de imagens. Com essa interface evita-se que o desenvolvedor, ao utilizar o Athena, tenha que entender toda a hierarquia de classes do framework. Além disso, por meio dessa interface o desenvolvedor acessa os dados de entrada necessários aos algoritmos de segmentação e armazena a imagem resultado, a qual será atualizada pelo componente `exibirImagem`.

Domínio da aplicação na área de imagens: Interfaces gráficas de aplicações que envolvem a segmentação de imagens em suas soluções.

Atributos da interface de acesso e suas funcionalidades:

`imagem (PlanarImage)`: armazena a imagem acessada pelo usuário;

`parametrosEntrada (Vector)`: contem os parâmetros de entrada para os algoritmos de segmentação.

Nome, parâmetros e funcionalidades das Operações:

`avisaObserver(PlanarImage):void` → método que observa se o parâmetro `imagem` da classe `acessoCamadaInferior` foi alterado. Caso afirmativo, esse método solicita ao componente **exibirImagem** que atualize a imagem corrente;

`getImagem():void` → método utilizado para acessar o atributo `imagem` da classe `acessoCamadaInferior;`

`getParametros():Object` → método que acessa os parâmetros de entrada necessários aos algoritmos de segmentação;

`setImagem(PlanarImage)` → método que armazena a imagem resultante do processo de segmentação na classe `acessoCamadaInferior;`

`setParametros()`→ método que armazena os parâmetros de entrada dos algoritmos de segmentação no vetor `parametrosEntrada` da classe `acessoCamadaInferior.`

Tabela 4: Exemplo da Aplicação do Padrão 3

Conseqüências:

A documentação da interface de acesso entre componentes, no domínio de imagens, possibilita ao projetista conhecer a área de aplicabilidade, sua funcionalidade, suas operações, como também, a forma de interligar o componente ao sistema por meio de serviços da interface.

Usos conhecidos:

O framework Athena de Freitas (2006) e os sistemas Arthemis e MeSegHi de Lourega (2006) utilizam a documentação da interface de acesso entre os componentes para registrarem as operações de contrato de serviço entre os componentes.

4- Nome: Padrão Documentação de Pacotes

Contexto:

O diagrama de pacotes da UML expressa a hierarquia dos componentes, organizadas semanticamente em grupos. Dessa forma, os componentes de sistemas para processamento e análise de imagens, pertencentes a uma mesma camada da arquitetura podem fazer parte de pacotes distintos em função de sua semântica.

Problema:

Como documentar as técnicas de processamento e análise de imagens, abordando a estrutura de informações contidas em um diagrama de pacotes?

Forças:

1. A documentação através de diagramas de pacotes é utilizada para modelar a arquitetura de sistemas para processamento e análise de imagens, isto é, organizar hierarquicamente os componentes semanticamente relacionados no mesmo grupo;
2. Na documentação de pacotes de sistemas para processamento e análise de imagens, é possível registrar como é estruturada a organização dos componentes abordada em uma arquitetura em camadas;
3. As características dos pacotes, tais como: domínio da aplicação do pacote, tipo do elemento, nome do elemento, descrição do domínio do elemento devem ser bem documentadas, com o intuito de disponibilizar aos projetistas o registro da aplicabilidade do pacote e o domínio de seus elementos pertencentes à arquitetura de sistemas para processamento e análise de imagens.

Solução:

A solução constitui-se nas seguintes etapas:

1. Especificar o nome da camada da arquitetura;
2. Definir o nome do pacote principal;

3. Descrever a área de domínio do pacote;
4. Criar a estrutura de organização do pacote nomeado na etapa 2;
5. Adicionar os tipos de elementos evidenciando o nome e o domínio de aplicabilidade.

Aplicação do Padrão:

Documentação de Pacotes
Nome da camada da arquitetura: Aplicação – segunda camada
Nome do pacote principal: `algebra`
Domínio da aplicação do pacote: Métodos algébricos utilizados em operações entre imagens.
O pacote **Nome do pacote:**
Tipo do elemento / Nome do elemento: descrição do domínio de aplicação
O pacote algebra:
pacote set: Neste pacote contém métodos de operações de conjunto aplicadas entre imagens, tais como diferença, media, negação, produto e histograma;
pacote geometry: Neste pacote contém métodos de operações geométricas aplicadas na imagem, como operação de escala;
pacote point: Neste pacote contém métodos de operações ponto a ponto aplicadas na imagem, como operação de brilho e da *Look-up table*.

Tabela 5: Exemplo da Aplicação do Padrão 4

Conseqüências:

Na documentação de pacotes lógicos no domínio de processamento e análise de imagens, é permitido ao desenvolvedor catalogar a organização dos pacotes e dos elementos que os constituí, criando assim, uma estrutura hierárquica dos componentes utilizados na modelagem do sistema.

Usos conhecidos:

O framework Athena de Freitas (2006) e o sistema Arthemis utilizam o "Padrão Documentação de Pacotes" para registrar a organização hierárquica dos pacotes que fazem parte de seus projetos arquiteturais.

5- Nome: Padrão Documentação de Casos de Uso

Contexto:

A documentação dos casos de uso dos requisitos funcionais de sistemas para processamento e análise de imagens descreve a aplicabilidade das funcionalidades observada por algum ator, e também, expressa os fluxos de eventos que representam uma seqüência de ações que apresentam as etapas de execução dos casos de uso da aplicação.

Problema:

Como documentar os requisitos funcionais das técnicas de processamento e análise de imagens, através da especificação dos casos de uso, o qual descreve o comportamento do sistema em termos de seqüências de ações.

Forças:

1. Na documentação dos casos de uso de sistemas para processamento e análise de imagens, é possível registrar os fluxos de eventos referentes à produção de um resultado coerente conforme observação do ator. Assim, é construído um conjunto de situações de sucesso e de erros nas interações entre os atores e as funcionalidades da aplicação que são respectivamente, denominados de fluxo principal e fluxo secundário;
2. Os cenários ilustrados pelos fluxos de eventos dos casos de uso devem ser bem documentados, uma vez que, são eles que conduzem a seqüência de ações de interação entre o ator e a técnica de processamento e análise de imagens.

Solução:

A solução aborda as seguintes etapas:

1. Criar um caso de uso que defina a funcionalidade e nomeá-lo;
2. Citar as pós-condições definidas na conclusão do caso de uso;
3. Adicionar as atividades selecionadas no fluxo principal;
4. Definir as atividades elencadas no fluxo secundário.

Aplicação do Padrão:

Documentação de Casos de Uso
Caso de uso: Gerar Mascara
Pós-condições: Imagem segmentada
Fluxo principal de atividades: 1 - A imagem selecionada pelo usuário deve ser em nível de cinza; 2 - Usuário escolhe o tamanho e o tipo do operador; 3 - O usuário escolhe um valor de limiar "mexendo" em uma barra que mostra alguns valores predefinidos do histograma; 4 - O usuário pode digitar um valor de limiar na área de texto; 5 - Opção *ok* gera as modificações na imagem exibida
Fluxo secundário de atividades: 1 - Se a imagem selecionada for colorida o método converterCor será chamado; 2 - Se o usuário não define o tamanho e o tipo de operador (máscara) o sistema seleciona o operador Frein-chen 3x3; 3 - Se o usuário não escolher um valor de limiar uma mensagem é exibida.

Tabela 6: Exemplo da Aplicação do Padrão 5

Conseqüências:

A documentação dos casos de uso de sistemas para processamento e análise de imagens contém informações sobre um conjunto de cenários de sucesso e de situações previstas que ilustram os atores interagindo com as funcionalidades do sistema.

Usos conhecidos:

O framework Athena de Freitas (2006) e o sistema Arthemis catalogam os cenários de interação entre o ator e a técnica de processamento e análise de imagens através do "Padrão de Caso de Uso".

6 O AMBIENTE "DOCARQWIKI"

Para solucionar as questões relacionadas à documentação da arquitetura de sistemas e *frameworks* para processamento e análise de imagens, foram criados padrões de documentação da arquitetura baseados nas visões arquiteturais da UML, conforme mostrado no capítulo 5.

Neste trabalho foi desenvolvido um ambiente colaborativo na ferramenta MoinMoinWiki, chamado "DocArqWiki", disponibilizando *templates* para os padrões propostos:

i) Padrão Documentação de Componentes de Software;

ii) Padrão Documentação da Interação entre Componentes;

iii) Padrão Documentação da Interface de acesso entre Componentes;

iv) Padrão Documentação de Pacotes;

v) Padrão Documentação de Casos de Uso.

Para criação dos padrões apresentados pelo "DocArqWiki" foi utilizado a visão lógica e de caso de uso da UML, a fim de definir regras claras para documentação de sistemas e frameworks para processamento e análise de imagens.

A seguir são apresentadas a descrição e as configurações do "DocArqWiki", assim como, os requisitos funcionais do ambiente *wiki*. É abordada também as funcionalidades dos *templates* dos padrões.

6.1 Descrição do ambiente

Quando técnicas e ferramentas colaborativas de documentação da arquitetura de software são utilizadas em conjunto, torna-se possível disponibilizar aos usuários regras claras e serviços de comunicação, cooperação e coordenação na criação da documentação.

Dessa forma, o "DocArqWiki" disponibiliza aos desenvolvedores um ambiente colaborativo para a documentação da arquitetura de sistemas e *frameworks* para processamento e análise de imagens, oportunizando as pessoas a compartilharem idéias e experiências promovendo a interatividade e o conhecimento.

Uma representação do modo de funcionamento do "DocArqWiki" é apresentada na Figura 12, onde deve ser informado pelo usuário o que deseja realizar e então a opção escolhida é executada pelo "DocArqWiki" e a resposta é retornada de forma clara e precisa.

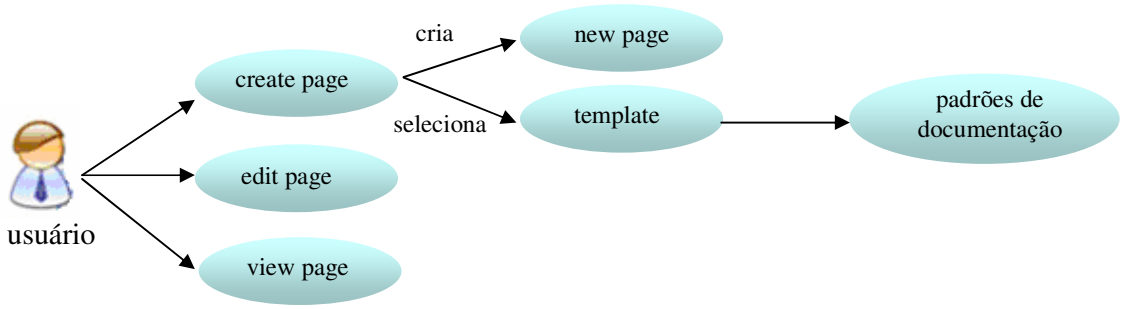

Figura 12: Diagrama de caso de uso do funcionamento do ambiente "DocArqWiki"

Observa-se nessa figura que, em relação ao processo de documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, referente ao que os ambientes *wikis* possibilitam, foram utilizadas as seguintes metodologias para criação de documentos padronizados. O "DocArqWiki" precisa inicialmente de uma entrada com a opção desejada do Padrão de documentação a ser seguido, como:

- *Template* do Padrão Documentação de Componentes de Software;
- *Template* do Padrão Documentação da Interação entre Componentes;
- *Template* do Padrão Documentação da Interface de acesso entre Componentes;
- *Template* do Padrão Documentação de Pacotes;
- *Template* do Padrão Documentação de Casos de Uso.

Para documentação da arquitetura é necessário que o usuário selecione qual *template* deseja utilizar, dentre as opções já cadastradas no "DocArqwiki". No entanto, os *templates* dos padrões podem ser atualizados, inseridos e apagados, tornando dessa forma o "DocArqWiki" adaptável a outros padrões de documentação.

O "DocArqWiki" utilizou o elemento, Solução, dos padrões de documentação da arquitetura, apresentados no capítulo 5, para criar os *templates* disponibilizados aos usuários, resultando uma catalogação clara e objetiva.

Conhecido o elemento, Solução, de cada padrão de documentação da arquitetura, foram então incluídas nos *templates* do "DocArqWiki" as características que definem seu formato e, já tendo as informações para cada *template,* torna-se possível criar a documentação de forma padronizada, retornando aos usuários soluções para problemas recorrentes.

Tais informações são referentes à solução proposta por cada padrão contemplando as características a serem documentadas pelos usuários. Isso é feito de modo que, com o conhecimento dessas informações, seja possível criar um repositório de documentação da arquitetura de sistemas e *frameworks* para processamento e análise de imagens.

Para elaboração das características do elemento, Solução, dos padrões de documentação da arquitetura foi utilizada as definições da visão lógica e de caso de uso, apresentados no capítulo 3. Foi necessária a documentação através dessas características, pois essas definem os componentes e suas interações.

A principal funcionalidade atribuída ao "DocArqWiki" é a possibilidade de oferecer Padrões de documentação da arquitetura para sistemas e frameworks para processamento e análise de imagens em um ambiente de trabalho colaborativo, de maneira que os desenvolvedores cooperem na construção de novos conhecimentos.

Na Figura 13 é mostrada a tela inicial do "DocArqWiki", onde algumas informações são disponibilizadas aos usuários.

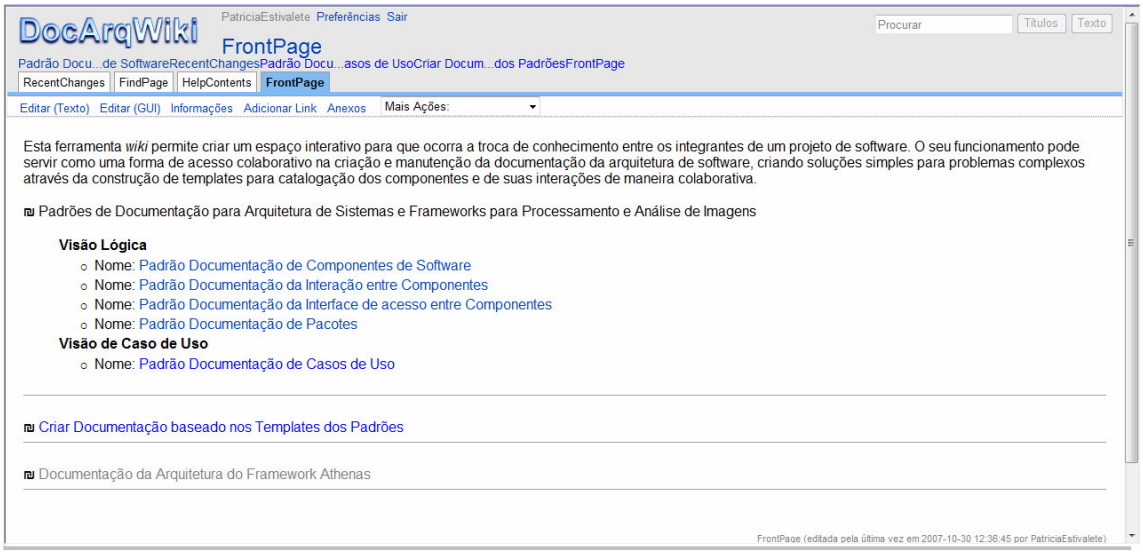

Figura 13: Tela principal do "DocArqWiki"

Nesta figura, além das principais funcionalidades disponibilizadas pelo "DocArqWiki" são também apresentadas outras informações, como *recentchanges e findpage*, às quais têm o intuito de mostrar aos usuários as recentes mudanças ocorridas no ambiente, e de disponibilizar um sistema de busca por conteúdo.

Através do menu de funcionalidades são disponibilizadas informações aos usuários de como utilizar o "DocArqWiki"; características e os passos seguidos para sua utilização; padrões de documentação da arquitetura disponíveis; *templates* criados através do elemento, Solução, dos padrões; opção de edição de páginas; histórico de contribuições; busca; ajuda, bem como a ferramenta que apoiou a construção do ambiente colaborativo.

As informações geradas pelo “DocArqWiki” são armazenadas em arquivos para a segurança dos dados, sejam elas informações de configuração do ambiente ou das páginas referente a documentação da arquitetura.

Esses arquivos estão no formato texto, podendo ser recuperados a qualquer momento, caso ocorra perda de dados na versão recente. As informações são exibidas aos usuários através do histórico de revisões, que identifica os dados do autor a cada interação.

Em relação às *recentchanges*, essas são informações referentes a todas as alterações efetuadas no ambiente, sendo registrado as páginas que foram deletadas pelos usuários. A partir dessas informações, é possível que o coordenador verifique o andamento das atividades de cooperação.

Na Figura 14 é mostrada as *recentchanges* da tela inicial do “DocArqWiki”, o qual aborda todas as interações realizadas na página pelos usuários, armazenando seus dados de modo que fique visível aos integrantes que estão cooperando e as tarefas que estão executando.

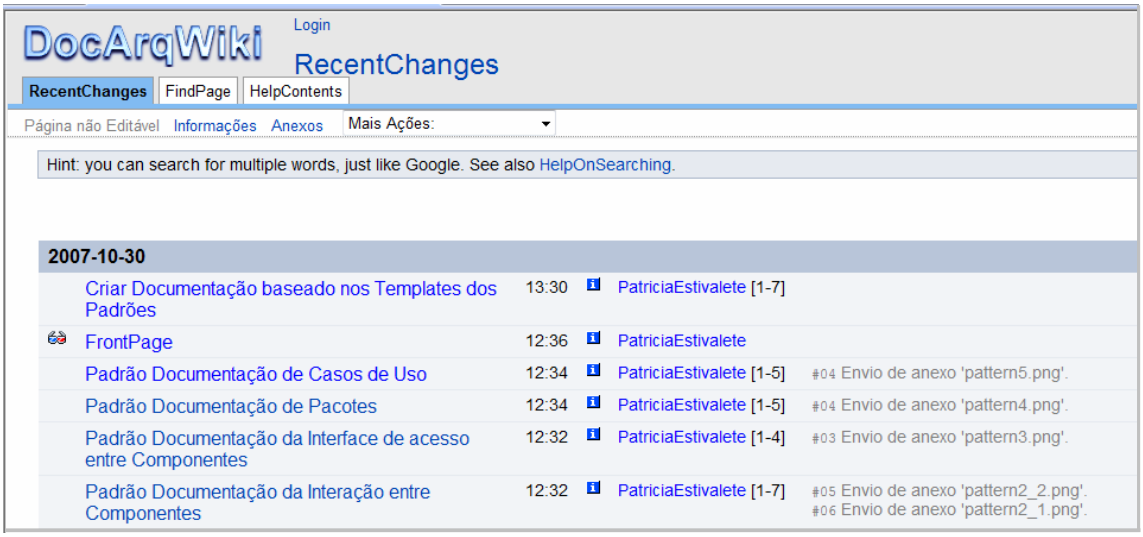

Figura 14: *Recentchanges* da tela principal do “DocArqWiki”

Nesta tela, apresentada na Figura 14, são especificadas as características utilizadas pela *recentchanges*, sendo elas: i) data, hora e nome do usuário, referente a cada alteração realizada; ii) visualização do envio dos anexos; iii) observação das ações executadas pelo usuário no ambiente, visualizadas através dos ícones e, iv) acesso aos conteúdos das páginas *wikis*.

6.2 Configurações do “DocArqWiki”

O “DocArqWiki” possibilita a criação de documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens por meio de *templates*, disponibilizando regras claras aos usuários e oferecendo um ambiente de trabalho colaborativo.

A configuração de usuários no “DocArqWiki” pode ser feita pela opção *UserPreferences*. Nesse módulo o usuário pode criar sua conta sem nenhuma restrição de acesso ao ambiente.

Dessa forma, são disponibilizados recursos de segurança das informações através de linhas de códigos inseridos no cabeçalho de cada página, permitindo que o autor do conteúdo estabeleça permissões, assegurando que usuários comuns não possam ter acesso a edição de páginas, de modo a proteger as informações publicadas no “DocArqWiki”.

A edição colaborativa proposta pelo “DocArqWiki” possibilita que os usuários autorizados, interajam com o ambiente, sempre que for necessário incluir novas informações a documentação da arquitetura, compartilhando assim, conhecimento a qualquer tempo.

As opções de edição, recuperação e exclusão das páginas wiki, bem como ações para salvar, podem ser executadas apenas pelos usuários que tem permissão de acesso, concedida pelo administrador da página. Essas contêm ícones representando as opções e ações que podem ser executadas, de modo que a interface seja amigável.

A página wiki referente ao Padrão de Documentação de Componentes de Software encontra-se preenchida com regras de proteção contra usuários maliciosos, podendo ser visualizada no início da página. Quando o usuário clicar em uma página que não possui permissão, o *link* de edição ficará desabilitado, autorizando apenas a leitura. Para efetivar a permissão é necessário inserir uma linha de código, como mostrado na Figura 15.

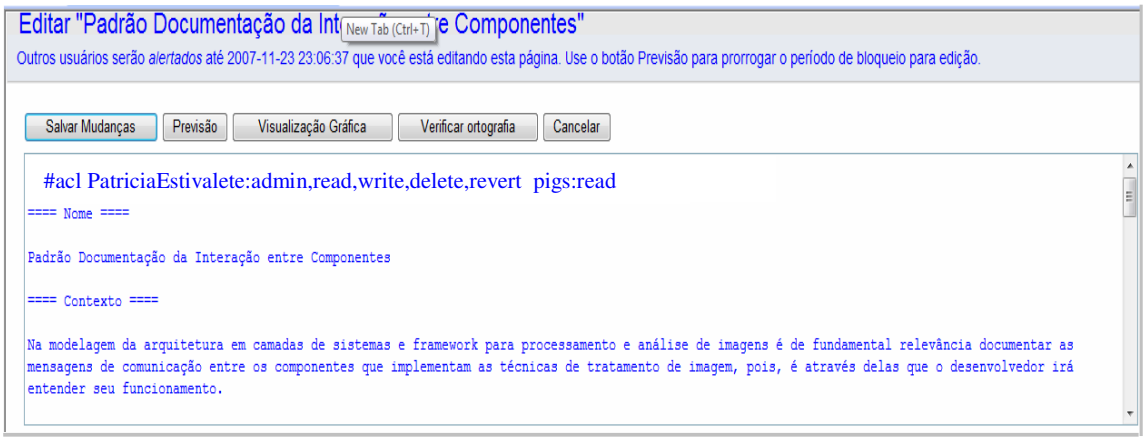

Figura 15: Tela de edição de página do Padrão Documentação de Componentes de Software

Nesta figura ainda são mostradas informações como previsão de publicação, verificação ortográfica e visualização gráfica referente ao conteúdo que está sendo editado, e ainda, aviso de bloqueio para edição, de modo que não permite a edição simultânea de conteúdos.

Na exclusão das páginas *wikis*, para que a ação possa ser efetivada é necessário que o usuário tenha permissão para executá-la. Após a exclusão de uma página, a ação fica armazenada na *recentchanges*.

Medidas de organização são tomadas para evitar que aconteçam problemas na coordenação das atividades do ambiente, como por exemplo: caso sejam feitas alterações pelos usuários e essas sejam salvas, será registrado no histórico de revisões a nova versão da página e os dados do usuário, a fim de que o coordenador possa analisar o processo de colaboração.

As ações e medidas de coordenação apresentadas na tela de histórico de revisões estão disponíveis aos usuários do ambiente, como visto na Figura 16.

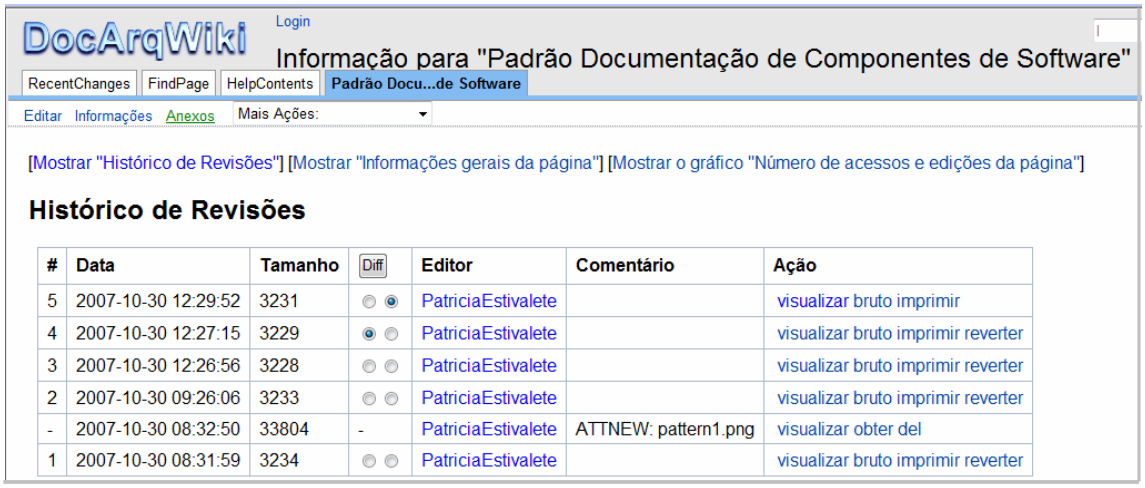

Figura 16: Tela de histórico de revisões do Padrão Documentação de Componentes de Software

Não serão detalhadas as funcionalidades do histórico de revisões, de modo a evitar redundância de informações, já que são as mesmas apresentadas pelas *recentchanges*, seguindo um mesmo padrão e lógica.

No momento de alteração de uma página, um registro é fornecido ao histórico de revisões, informando o nome da pessoa que executou as alterações. Isso é feito para permitir que seja identificado o responsável pelas informações modificadas, de modo a garantir a veracidade dessas.

A seguir são mostradas as telas da "DocArqWiki" para avaliação dos *templates* dos padrões de documentação da arquitetura. Essas são funcionalidades mais relevantes da

"DocArqWiki", pois serviram para validar de forma prática a utilização da visão lógica e de caso de uso da UML.

6.3 Templates dos padrões

Depois de já terem sido apresentados os passos seguidos para utilização do "DocArqWiki" e algumas de suas funcionalidades, nesta seção é apresentada a construção do módulo dos *templates* dos Padrões.

Através da "DocArqWiki" tornou-se possível apresentar aos usuários, de forma clara e objetiva, *templates* baseados nos padrões de documentação da arquitetura, verificando-se o trabalho colaborativo entre os integrantes do grupo.

A visão lógica e de caso de uso da UML foram utilizadas para padronização da Documentação da Arquitetura, mostrada na seção 5.3, de forma que, sabendo-se quais características da visão arquitetural o padrão contempla, é possível gerar documentos mais facilmente e com mais segurança.

Utiliza-se características de sistemas e *frameworks* para processamento e análise de imagens para estabelecer o domínio de uso dos padrões, pois através dessas características é mais fácil para o usuário entender os *templates* oferecidos.

Essas características são apresentadas aos usuários na forma de padrões para facilitar o entendimento do usuário e, conhecido o resultado da documentação, torna-se possível aos usuários o reuso dos componentes de forma consciente.

Na Figura 17 é mostrada a tela do "DocArqWiki", com a opção do usuário escolher qual *template* deseja utilizar para documentar a arquitetura.

Figura 17: Tela de *templates* dos padrões de documentação da arquitetura

A documentação da arquitetura feita pela “DocArqWiki” foi disposta de modo que os usuários pudessem criar novos documentos seguindo *templates* e oportuniza-se a interação através da edição colaborativa das páginas *wikis*.

A seguir é apresentada, de forma detalhada os *templates* dos padrões de documentação da arquitetura, sendo eles: “PadraoComponenteTemplate, PadraoInteracaoComponenteTemplate, PadraoInterfaceComponenteTemplate, PadraoPacotes Template e PadraoCasoUsoTemplate.

6.3.1 Template “PadraoComponenteTemplate”

Na Figura 18 é mostrado o funcionamento do “Padrão Componente Template”, exemplificando a maneira como o “DocArqWiki” foi desenvolvido para a documentação dos componentes de software.

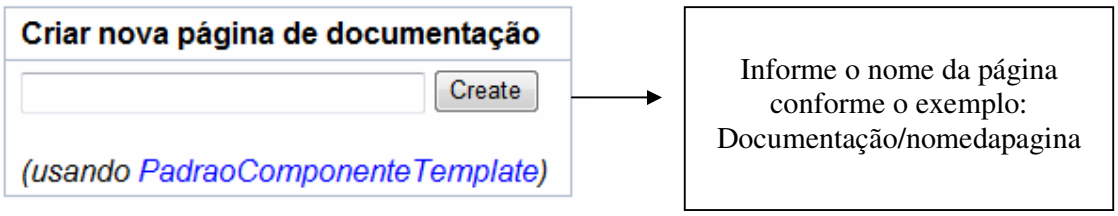

Figura 18: Exemplificação do “Padrão Componente Template”

Pelo “DocArqWiki” são apresentados ao usuário opções de *templates* dos padrões previamente cadastrados. De acordo com a escolha do usuário é retornada uma nova página, descrevendo as características oferecidas pelo mesmo.

Através do “PadraoComponenteTemplate” são apresentados aos usuários características de reuso dos componentes de sistemas e *frameworks* para processamento e análise de imagens. Conforme o preenchimento do usuário é retornado uma página *wiki* no ambiente, mostrando as características documentadas referente ao componente.

O componente, cujas informações sobre suas características foram documentadas na “DocArqWiki”, são disponibilizadas em uma página *wiki*, de maneira que os usuários possam cooperar com novas informações a qualquer tempo através da opção “Editar”, como mostrado na Figura 19.

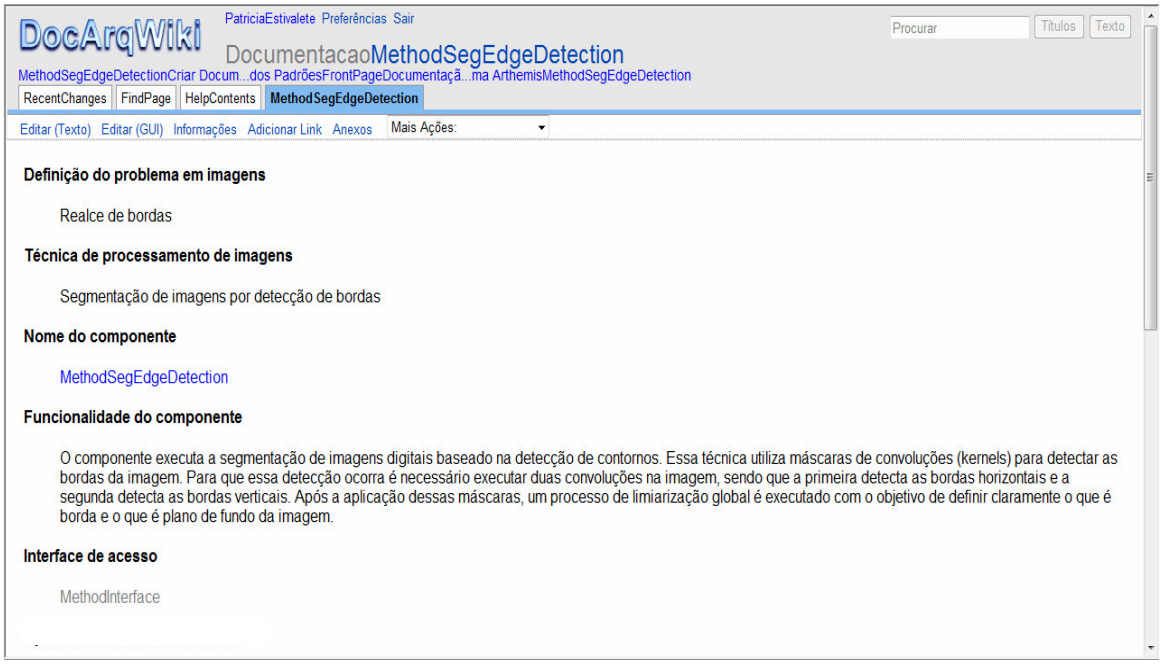

Figura 19: Tela de documentação do componente “MethodSegEdgeDetection”

A partir do conhecimento de quais características de reuso o Padrão contempla, torna-se possível retornar ao usuário a documentação do componente de forma que seja compreendido e aplicável em outros sistemas de mesmo domínio.

Pode-se observar que o “Padrão Componente Template” apresenta as características abordadas no elemento, Solução, do padrão documentação de componentes de software, sendo elas: definição do problema em imagens; técnica de processamento de imagem utilizada para resolvê-lo; nome do componente; funcionalidade do componente; operadores do método/técnica; parâmetros de entrada da técnica; atributo da imagem fonte; métodos matemáticos; atributo da imagem resultado e resultado da aplicação da técnica. O *template* ainda aborda a característica, Interface de acesso, do padrão documentação da interface de acesso entre componentes, como mostrado na Figura 20.

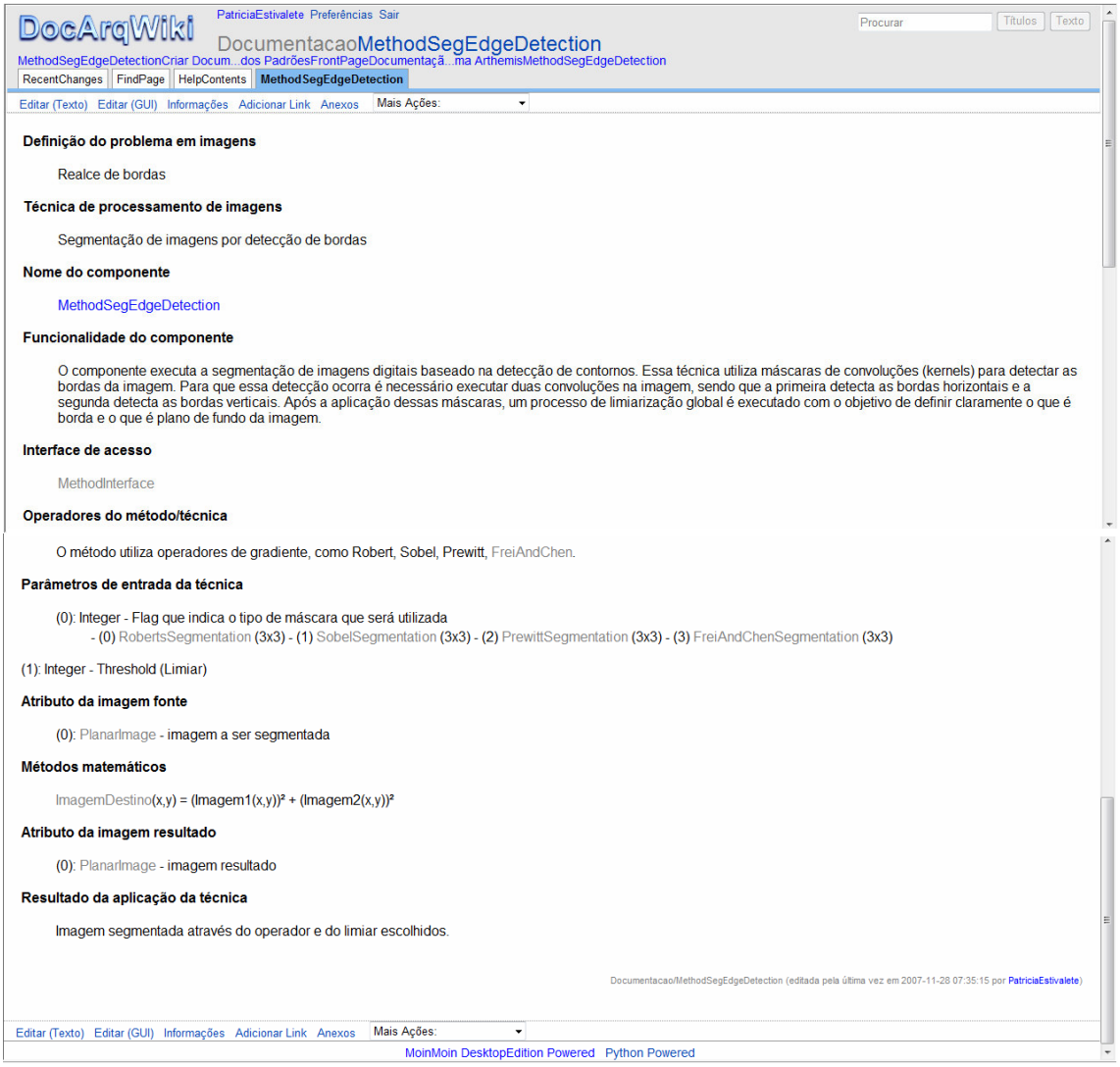

Figura 20: Tela de documentação do componente “MethodSegEdgeDetection”

Tendo conhecimento dos componentes documentados, o usuário pode tomar decisões estando mais seguro quanto a qual componente utilizar. A partir do resultado da documentação dos componentes, o usuário ainda pode ver detalhes de suas informações, bem como qual *template* foi utilizado e o nome do responsável pela criação da página wiki no “DocArqWiki”.

6.3.2 Template “PadraoInteracaoComponenteTemplate”

Nesta seção é detalhado o “PadraoInteracaoComponenteTemplate”, abordando as características de interação entre componentes e ainda, sua funcionalidade no ambiente “DocArqWiki”.

Para documentação das interações entre os componentes através do *template* é necessário que o usuário esteja cadastrado no ambiente e possua permissão para "edição", de maneira que possa criar novas páginas *wikis* e ainda, cooperar com novas informações em páginas já publicadas.

Na Figura 21 é mostrada uma exemplificação do "Padrão Interação Componente Template", onde são especificadas as mensagens de comunicação entre componentes, representada pelo fluxo de controle.

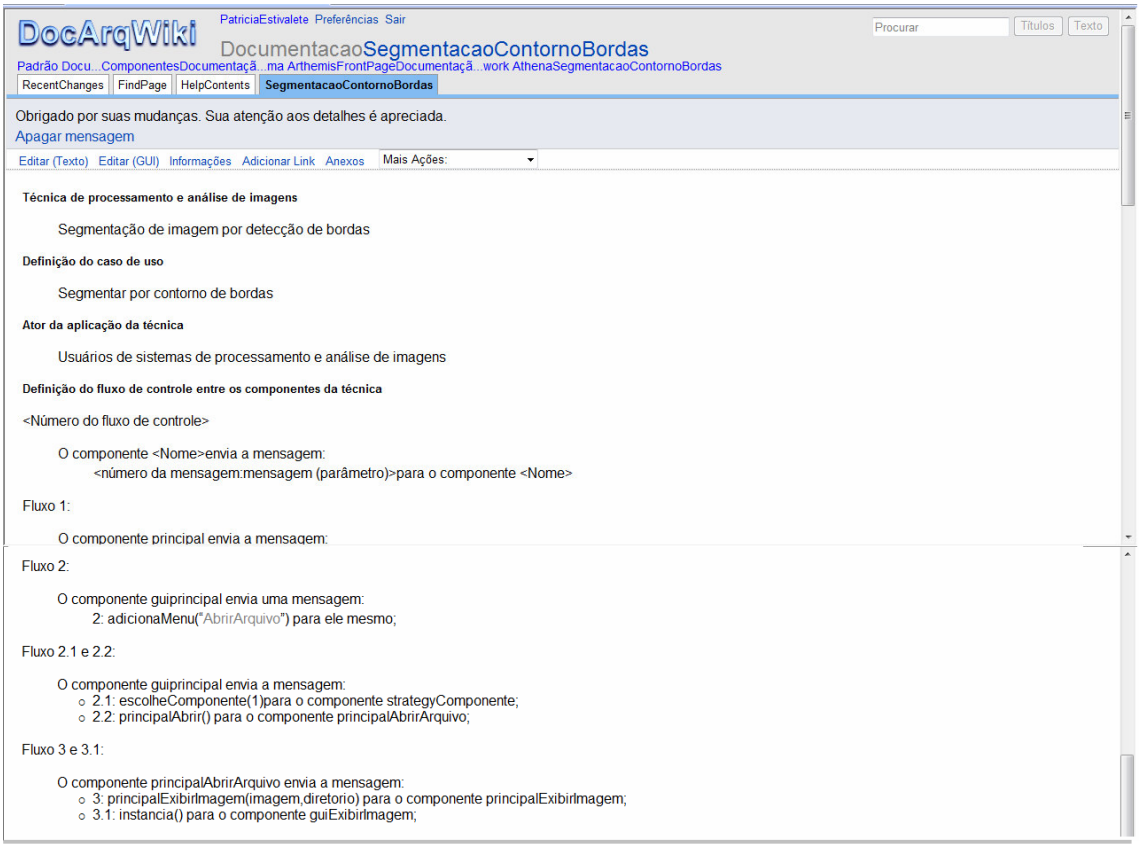

Figura 21: Tela de documentação da técnica "SegmentacaoContornoBordas"

Através do "PadraoInteracaoComponenteTemplate" são apresentados aos usuários às mensagens de comunicação, representada pelo fluxo de controle da técnica de processamento e análise de imagem, possibilitando o entendimento sobre seu processo de funcionamento.

Para que a documentação da interação entre componentes possa ser realizada de maneira satisfatória, o *template* deve referenciar todas as regras que identificam as características do fluxo de controle e, a partir daí, analisar e documentar os relacionamentos existentes em cada técnica.

Durante a documentação da interação entre componentes algumas regras impostas na sua criação devem ser cumpridas. Um exemplo são as mensagens de comunicação que

especificam as operações que cada componente deve executar, a fim de que a técnica de processamento e análise de imagem seja implementada corretamente.

Para cada mensagem de comunicação a ser documentada, existem algumas que são relevantes para a estrutura básica da técnica e outras que especificam sua usabilidade na aplicação. Dessa forma, a documentação apresenta-se mais eficiente quando é possível aplicar soluções já testadas e comprovadas, otimizando o processo de desenvolvimento do sistema.

6.3.3 Template "PadraoInterfaceComponenteTemplate"

Para a escolha do "PadraoInterfaceComponenteTemplate", utiliza-se a página wiki "Templates dos Padrões de Documentação da Arquitetura", a qual contempla todas as opções de padronização de documentos. Essas opções foram disponibilizadas na "DocArqWiki" em uma caixa de texto, onde o usuário digita o nome da página que deseja criar para sua documentação, gerando-a automaticamente com o padrão selecionado. A Figura 22 mostra uma exemplificação do conteúdo do *template* adicionado à nova página.

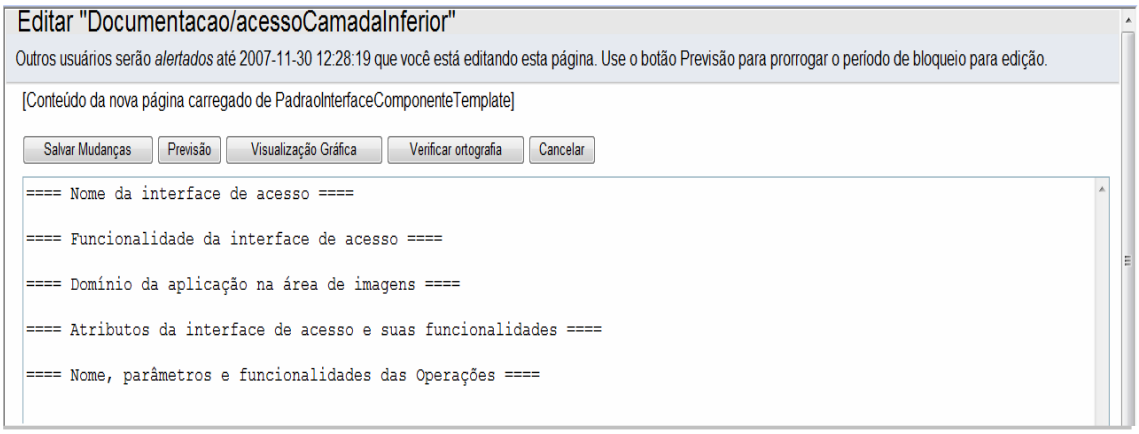

Figura 22: Exemplificação do "PadraoInterfaceComponenteTemplate"

Nessa figura pode-se visualizar que na resposta da criação da nova página de documentação usando o *template*, retornada ao usuário, são identificadas regras relevantes à interface de acesso que devem ser catalogadas.

A seguir, na Figura 23, é apresentada a documentação da interface, acessoCamadaInferior, no "DocArqWiki", seguindo a padronização do "PadraoInterfaceComponenteTemplate".

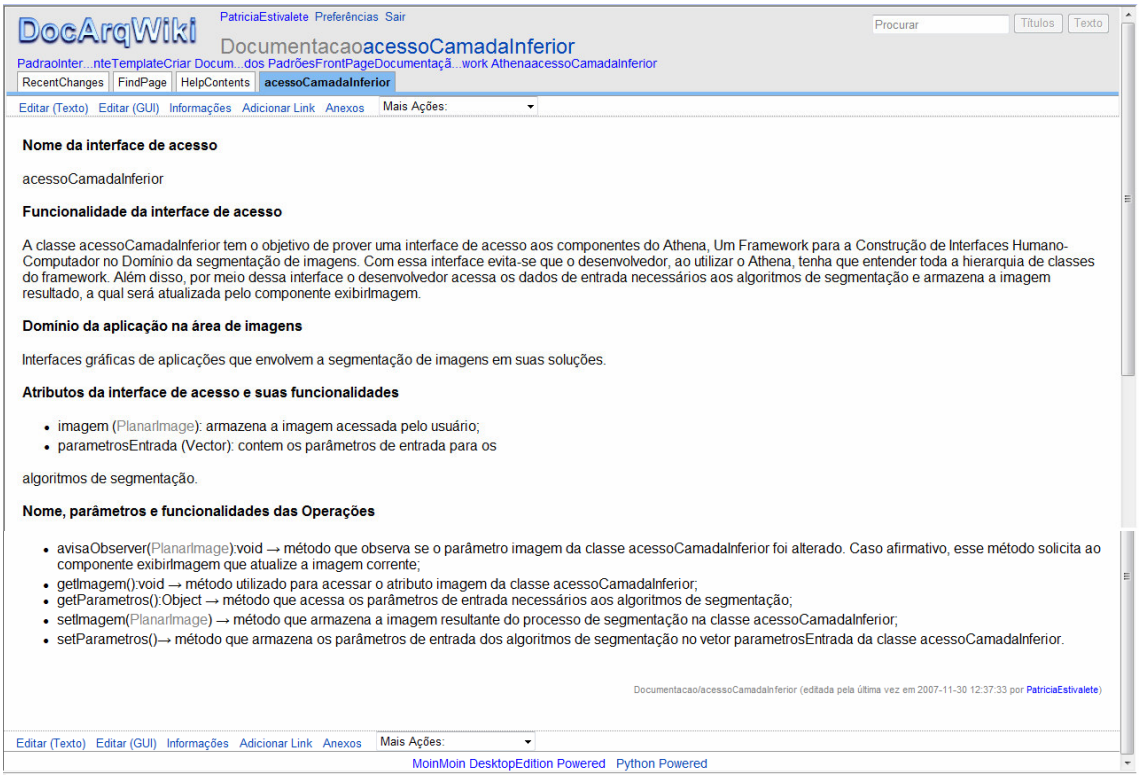

Figura 23: Tela de documentação da interface “acessoCamadaInferior”

Nessa tela, é possível observar que são retornados aos usuários informações sobre a documentação da interface de acesso, divididas por suas características, sendo elas: nome, funcionalidade, domínio, atributos e, ainda nome, parâmetros e funcionalidades das operações.

Através do “PadraoInterfaceComponenteTemplate” são disponibilizados aos usuários regras claras para catalogar as características da interface de acesso entre componentes. A partir de sua documentação, torna-se possível retornar ao usuário os métodos oferecidos e suas funcionalidades, de maneira a ajudá-los a compreender como integrar novos componentes ao sistema.

6.3.4 Template “PadrãoPacotesTemplate”

Na Figura 24 é mostrado o funcionamento do “PadrãoPacotesTemplate”, exemplificando a maneira que as características dos pacotes foram distribuídas no “DocArqWiki”, possibilitando a documentação dos pacotes lógicos de sistemas e frameworks para processamento e análise de imagens.

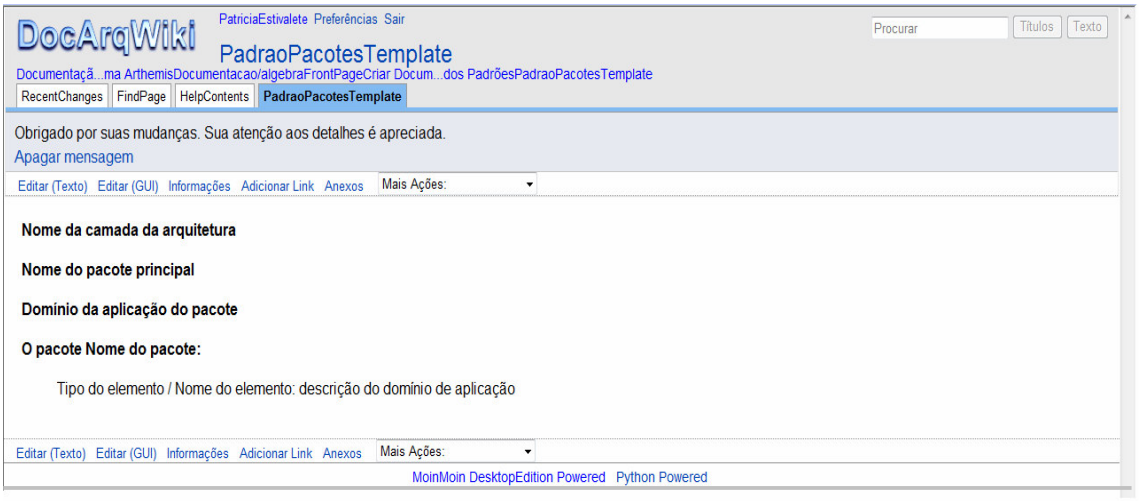

Figura 24: Exemplificação do "PadraoPacotesTemplate"

Nessa figura pode-se observar que as características estabelecidas no *template* para a documentação de pacotes lógicos são organizadas de maneira que os componentes semanticamente relacionados pertençam ao mesmo grupo, de forma a hierarquizar a arquitetura do sistema.

A documentação do pacote, álgebra, no "DocArqWiki" usando o "PadraoPacotesTemplate" pode ser visualizado na Figura 25.

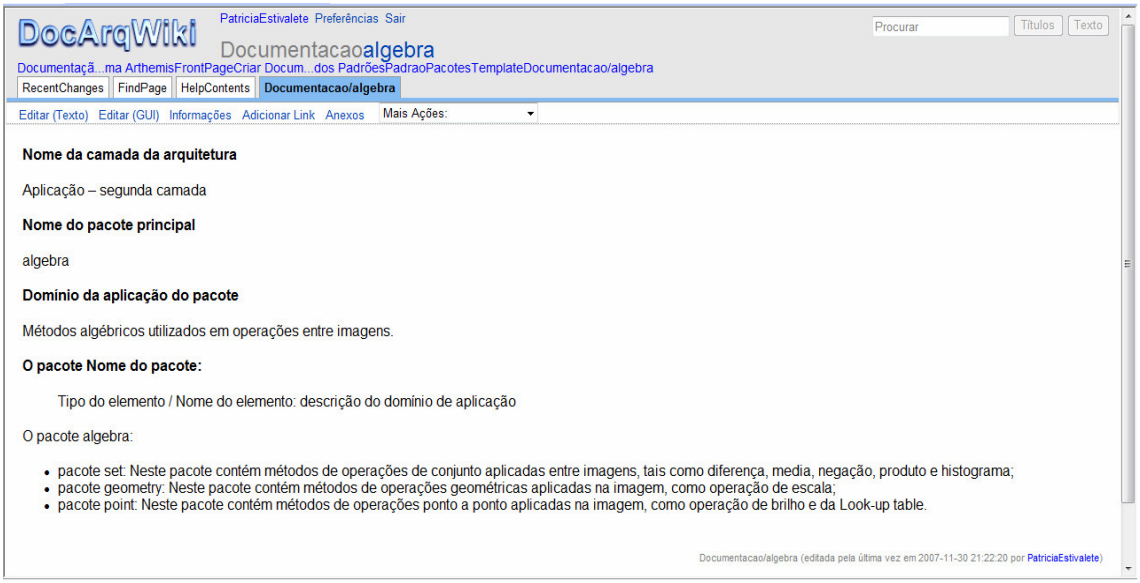

Figura 25: Tela de documentação do pacote "algebra"

Nessa tela, é possível observar que são mostrados aos usuários informações sobre as características dos pacotes, sendo elas: nome do pacote principal, tipo do elemento, nome do elemento, descrição do domínio do elemento, disponibilizando aos projetistas o registro da aplicabilidade do pacote e o domínio de seus elementos.

Através do “PadraoPacotesTemplate” é proposto uma padronização para catalogar as características dos pacotes lógicos. A partir de sua documentação, torna-se possível retornar ao usuário a organização dos pacotes e de seus elementos, construindo uma estrutura hierárquica dos componentes de mesma semântica.

6.3.5 Template “PadrãoCasoUsoTemplate”

Na Figura 26 é detalhado o “PadrãoCasoUsoTemplate”, exemplificando a maneira que os fluxos de eventos foram distribuídas no “DocArqWiki”, possibilitando a documentação das ações dos atores interagindo com as funcionalidades do sistema.

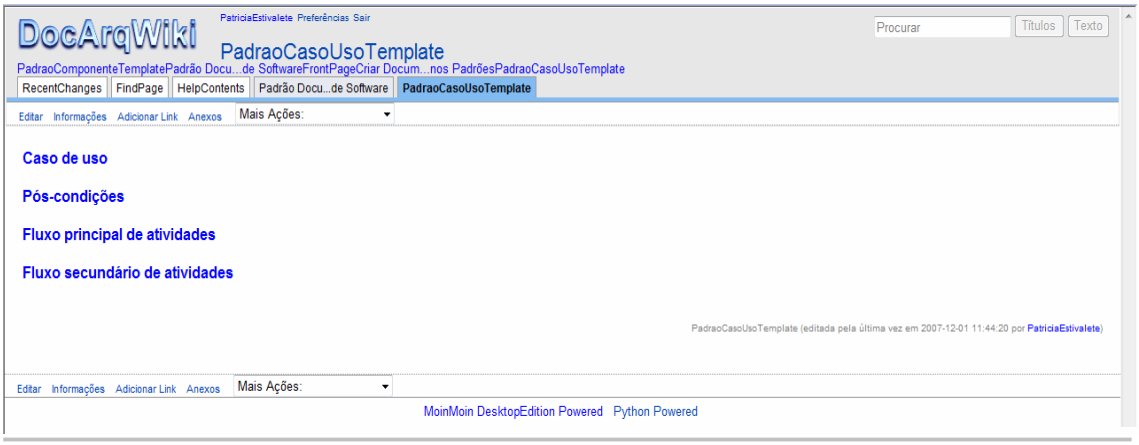

Figura 26: Exemplificação do “PadraoCasoUsoTemplate”

Nessa figura pode-se visualizar que as características estabelecidas no *template* para a documentação de casos de uso são organizadas de maneira que sejam definidas as pós-condições e as ações do fluxo principal e secundário, observados pelos atores.

A documentação dos casos de uso de sistemas para processamento e análise de imagens através do “PadraoCasoUsoTemplate”, possibilita registrar os fluxos de eventos referentes ao resultado da observação do ator. Dessa forma, constrói-se um conjunto de situações de sucesso e de erros nas atividades de interações entre os atores e as funcionalidades do sistema, chamados de fluxo principal e fluxo secundário;

A seguir, na Figura 27, é mostrado a documentação do caso de uso, Gerar Mascara, no “DocArqWiki” usando o “PadraoCasoUsoTemplate”.

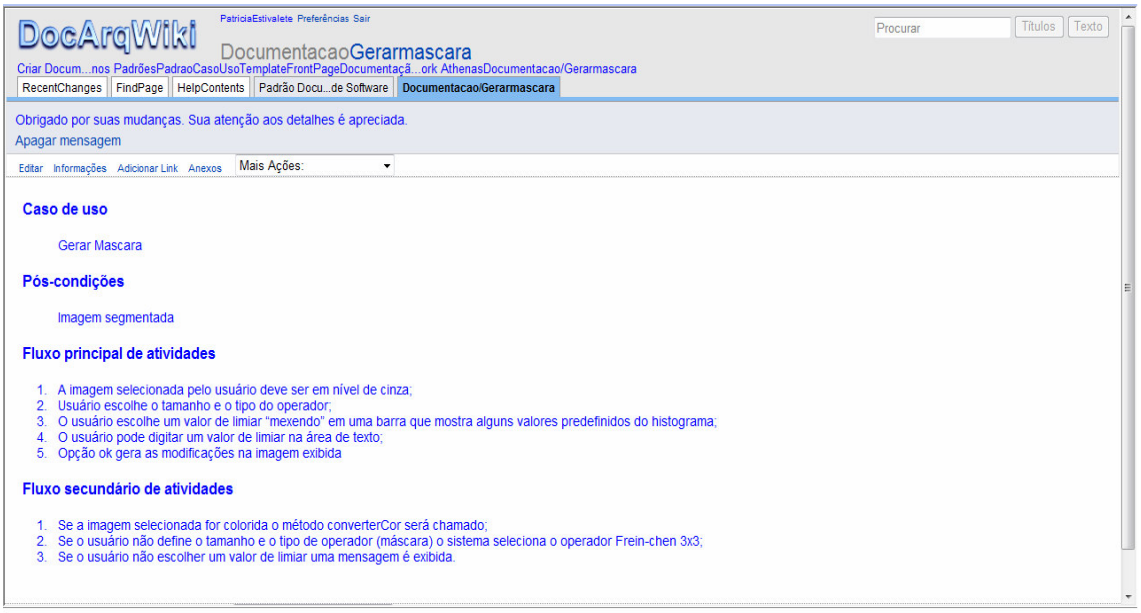

Figura 27: Tela de documentação do caso de uso “Gerar Mascara”

Nessa tela, é possível observar que é mostrada aos usuários a seqüência de ações efetuadas pelos atores no momento que interagem com a funcionalidade, gerar mascara, proporcionado um histórico de ocorrências de situações de sucesso e de erro.

Tendo conhecimento da documentação dos casos de uso, o usuário sente-se mais seguro ao executar uma funcionalidade, sabendo que pode ocorrer situações de erro caso ele não informe os dados certos ao sistema.

6.4 Conclusões

O desenvolvimento do “DocArqWiki” foi importante para validar conceitos sobre os *templates* dos Padrões de Documentação da Arquitetura oferecidos durante a interação dos usuários com o ambiente. Através do uso do “DocArqWiki” pode-se utilizar, de forma prática, os *templates* definidos.

Seu desenvolvimento ainda foi necessário devido ao fato de poder se ter um repositório de informações sobre a documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, facilitando dessa forma o estudo dos desenvolvedores e projetistas de software dessa área. Além de ser também um repositório de *templates* dos Padrões de Documentação e de suas respectivas características.

Outra vantagem apresentada pelo “DocArqWiki” é a sua capacidade de propor um ambiente de trabalho colaborativo, principalmente em relação a documentação da arquitetura

as quais estão sujeitas a constantes modificações, o que oportuniza a cooperação de vários integrantes em relação a um determinado assunto.

O "DocArqWiki" aborda temas referentes aos *templates* dos Padrões de Documentação da Arquitetura, permitindo que os usuários estejam integrados do assunto e percebam o quão importante é documentar a arquitetura de sistemas e os benefícios oferecidos pelo ambiente colaborativo.

O ambiente de trabalho colaborativo "DocArqWiki", encontra-se no endereço: *http://laca2.inf.ufsm.br:8080/*.

7 Estudo de Caso – Documentação da Arquitetura do "Arthemis"

Apresenta-se neste capítulo um Estudo de Caso no sistema de processamento e análise de imagens "Arthemis" que tem como objetivo a documentação da arquitetura através dos padrões de documentação disponibilizados no ambiente colaborativo "DocArqWiki" na forma de *templates*.

Esses *templates* representam o elemento, Solução, dos padrões de documentação da arquitetura, evidenciando as características que devem ser documentadas pelos desenvolvedores e projetistas, a fim de criarem uma documentação padronizada. O resultado deste Estudo de Caso serviu como base para documentar a arquitetura do sistema "Arthemis" através da utilização dos *templates* dos padrões de documentação propostos pelo ambiente "DocArqWiki", sendo esse o objetivo principal desse capítulo.

De acordo com Yin (2005) são empregados Estudos de Caso quando se quer investigar e compreender em profundidade fenômenos sociais, através de abordagens empíricas e holísticas de problemas contemporâneos.

Além disso, Estudos de Caso são especialmente pertinentes quando as perguntas de pesquisa buscam a compreensão de como se desenvolvem determinados processos, suas causas e motivações (como e por quê) (YIN, 2005).

Seu objetivo é compreender o evento em estudo e ao mesmo tempo desenvolver teorias mais genéricas a respeito dos aspectos característicos do fenômeno observado (FIDEL, 1993). A abordagem de Estudo de Caso não é um método propriamente dito, mas uma estratégia de pesquisa.

7.1 O sistema "Arthemis"

Tendo a decisão pela utilização da linguagem Java para a implementação do "Arthemis" foi tomado como ambiente uma API que oferece suporte ao Java e disponibiliza recursos visuais, de forma a facilitar a criação de interfaces amigáveis.

Para isso utilizou-se a API NetBeans IDE 5.5 como ferramenta de desenvolvimento, o compilador Java e o ambiente de execução presente no pacote de desenvolvimento *Java Development Kit* (JDK) versão 1.5.0.09, a API JAI como biblioteca para processamento e análise de imagens, aliado aos recursos da arquitetura baseada em componentes e padrões de projeto.

O “Arthemis” tem a funcionalidade de implementar métodos de processamento e análise de imagens em microscopia quantitativa e anatomopatologia, evidenciando os seguintes pontos: a filtragem, a segmentação e o reconhecimento, o qual busca classificar os objetos a partir de informações encontradas na imagem.

7.2 Levantamento de dados

A questão da documentação da arquitetura é um dos pontos relevantes no desenvolvimento de software. Os sistemas, especificamente os de processamento e análise de imagens que são retratados neste trabalho, dispõem de informações referente a suas técnicas como segmentação de imagens, realce de imagens, filtragem de ruídos, etc., que precisam ser catalogadas para posterior entendimento de seu funcionamento.

Técnicas de processamento e análise de imagens estão auxiliando diversas áreas do conhecimento, como exemplo, na medicina, as técnicas auxiliam os médicos a diagnosticarem com maior precisão patologias e no campo geoespacial permitem visualizar o estado climático de uma região ou quantificar o desmatamento em uma determinada área.

Do ponto de vista da engenharia de software, tanto a análise quanto a especificação dos requisitos desses sistemas devem ser bem definidos e documentados a fim de atender as necessidades dos usuários e oferecer maior segurança para os projetistas no desenvolvimento das funcionalidades.

Os sistemas devem disponibilizar uma documentação com regras claras e características relevantes de modo que os desenvolvedores possam entendê-los e adicionar novas funções de maneira confiável. Como por exemplo, padrões para documentação dos componentes, interação entre componentes, interface de acesso, pacotes e casos de uso, de modo que auxiliam a catalogação no desenvolvimento do software.

Pensando nisso, definiu-se um cenário de estudo para a documentação da arquitetura do “Arthemis”, sendo esse o sistema voltado ao domínio de processamento e análise de imagens.

A seguir é apresentada a metodologia empregada para o estudo de caso no sistema “Arthemis”, definindo-se para isso algumas etapas a serem seguidas, de forma a conduzir a realização das tarefas.

Objetivou-se conhecer informações precisas sobre os componentes a serem documentados no “Arthemis”, de forma que fossem úteis na comprovação da eficácia dos

padrões de documentação da arquitetura. Dessa forma, com a utilização desses padrões pelo "Arthemis", tornou-se possível criar um repositório de documentos referente a sua arquitetura.

Essas informações são adquiridas através do conhecimento científico, que se dá pela investigação planejada, de acordo com normas guiadas por uma metodologia bem definida. Dessa forma, busca-se compreender ou explicar a realidade pela prática de um conjunto de procedimentos técnicos e intelectuais denominados métodos científicos (GIL, 2002).

7.3 Metodologia utilizada

De acordo com Russ (1999), técnicas de processamento de imagens consistem em melhorar a aparência visual da imagem para a visão humana e/ou preparar a imagem para análise e mensuração de características e estruturas presentes.

A escolha pelo sistema "Arthemis" para o estudo deu-se ao fato de suas características serem voltadas ao processamento e análise de imagens. Essas características se bem documentadas podem ser utilizadas em outros sistemas de mesmo domínio.

Para tanto, utilizou-se uma metodologia composta por duas etapas, de modo a ampliar o entendimento dos resultados na formação do conhecimento.

Sabido do número de componentes desenvolvidos e em desenvolvimento no "Arthemis" viu-se necessário utilizar uma metodologia de amostragem para escolha de quais componentes seriam documentados.

A Figura 28 ilustra, de forma sintética as etapas gerais desta pesquisa, sendo que os detalhes dessas etapas, constituídos do projeto e desenvolvimento do Estudo de Caso, são apresentados nas próximas seções.

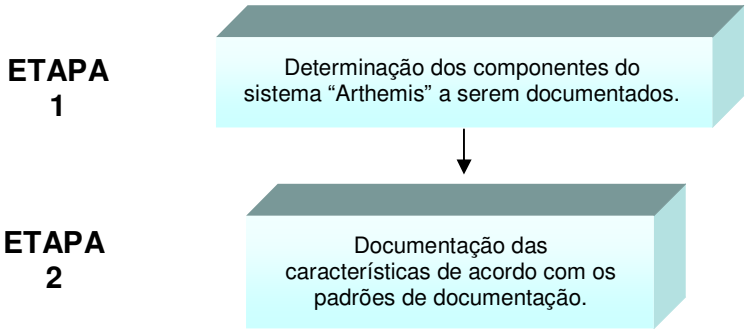

Figura 28: Ilustração das etapas seguidas para o levantamento de dados

As etapas compreendem um levantamento de dados para fornecer auxílio na identificação dos componentes relevantes a serem analisados no “Arthemis”, para a documentação da arquitetura conforme os padrões sugeridos.

Tais padrões podem auxiliar na determinação de regras claras de documentação da arquitetura, a partir das visões da UML já propostas, podendo esses serem utilizados como recurso para a documentação em outros sistemas de mesmo domínio.

Etapa 1 - Determinação dos componentes do sistema “Arthemis” a serem documentados.

De acordo com a nova reestruturação do “Arthemis”, é inviável a documentação de todos os componentes em função que não foram adicionados na versão atual do sistema. Para suprir essa dificuldade, fez-se uma amostragem selecionando parcelas menores desse conjunto de componentes que representam as funcionalidades principais do sistema.

Para a amostra dos componentes selecionados foram levadas em consideração algumas técnicas de processamento e análise de imagens que fazem parte do “Arthemis”. Cada componente da amostra foi documentado com a utilização dos *templates* do ambiente “DocArqWiki”.

Etapa 2 - Documentação das características de acordo com os padrões de documentação.

Após a determinação da amostra, nesse passo foram realizadas as atividades relacionadas a catalogação da arquitetura do “Arthemis” baseados nos padrões de documentação disponibilizados no ambiente “DocArqWiki”.

Para a determinação desses *templates* foi levado em consideração o elemento, Solução, dos padrões de documentação desenvolvidos, os quais subsidiaram o desenvolvimento de regras que registram as características relevantes da arquitetura.

Tais *templates* foram dispostos em um ambiente de trabalho colaborativo, chamado “DocArqWiki”, de modo que a arquitetura possa ser documentada de forma cooperativa pelos desenvolvedores do sistema.

7.4 Catalogação dos componentes do “Arthemis” no ambiente “DocArqWiki”

Nesta seção é apresentada a catalogação dos componentes do “Arthemis” que fizeram parte do estudo de caso, através dos *templates* dos padrões de documentação disponíveis no ambiente “DocArqWiki”.

Esses *templates* são importantes para facilitar a interação dos usuários com a documentação da arquitetura, garantindo e disponibilizando regras claras para a catalogação das informações. Ainda são relevantes pois através deles tornou-se possível identificar os elementos que compõem a visão lógica e de caso de uso da UML.

De acordo com a metodologia proposta, com os *templates* dos padrões de documentação já apresentados, foram documentadas as características dos componentes seguindo os modelos sugeridos.

A seguir, apresenta-se a documentação da arquitetura do “Arthemis” no ambiente “DocArqWiki”, utilizando o “PadraoComponenteTemplate”, o “PadraoInteracaoComponenteTemplate” e o “PadraoCasoUsoTemplate”.

7.4.1 Uso do “PadraoComponenteTemplate”

Através do “PadraoComponenteTemplate” é apresentado aos usuários a documentação das características de reuso dos componentes do sistema “Arthemis”.

7.4.1.1 Componente de software “MethodCorrectSub”

O componente tem a funcionalidade de receber duas imagens e um offset (deslocamento) e executar a operação de subtração entre as imagens utilizando o deslocamento para ajustar as intensidades dos pixels da imagem resultado. O método tem como entrada uma imagem com iluminação irregular e uma imagem que representa o padrão de iluminação irregular. O resultado é uma imagem corrigida através da operação de subtração.

A seguir, na Figura 29, é mostrado a documentação do componente de software, MethodCorrectSub , no “DocArqWiki” usando o “PadraoComponenteTemplate”.

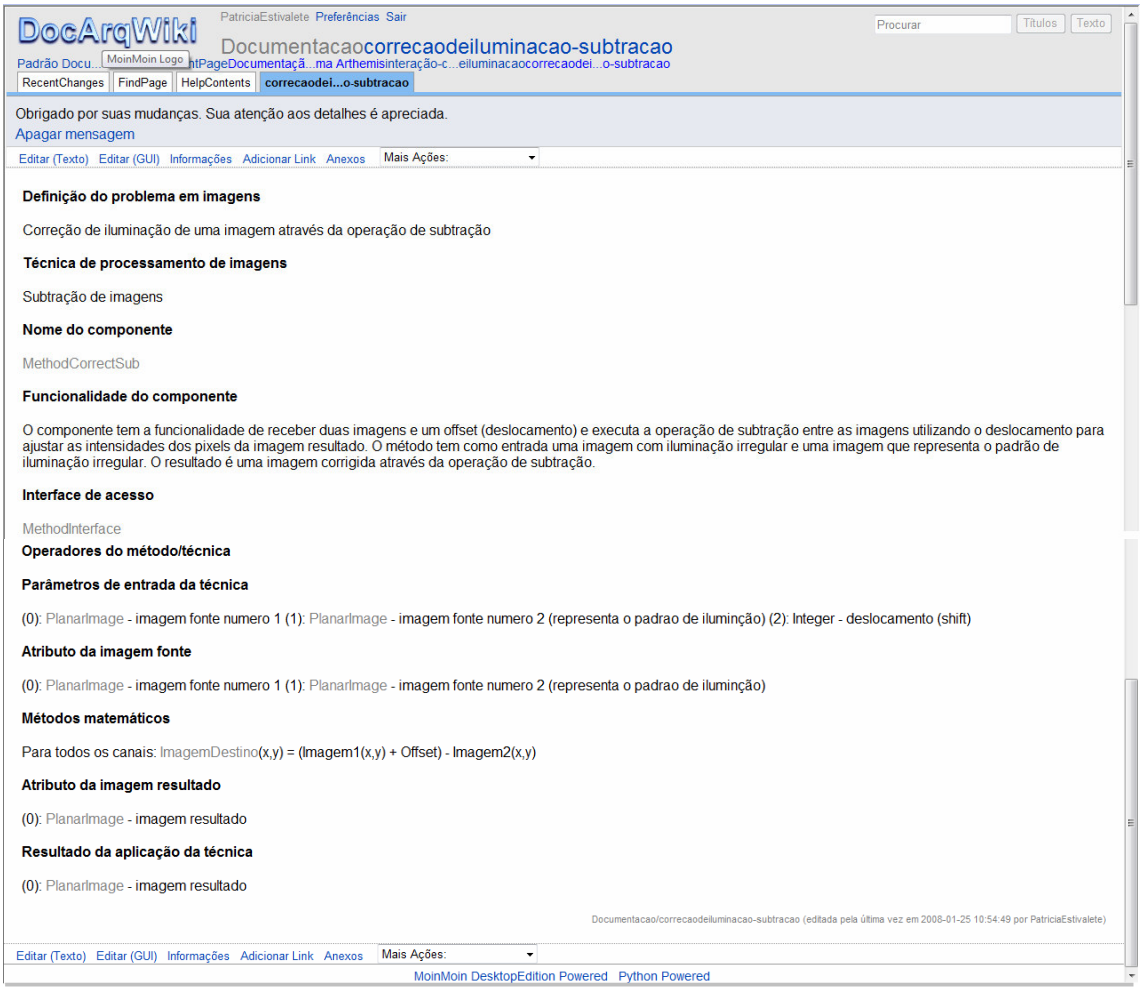

Figura 29: Tela de documentação do componente “MethodCorrectSub”

7.4.1.2 Componente de software “MethodCorrectDiv”

O componente tem a função de receber duas imagens e um offset (deslocamento) e executar a operação de divisão entre as imagens utilizando o deslocamento para ajustar as intensidades dos pixels da imagem resultado. O método tem como entrada uma imagem com iluminação irregular e uma imagem que representa o padrão de iluminação irregular. O resultado é uma imagem corrigida através da operação de divisão.

A seguir, na Figura 30, é mostrado a documentação do componente de software, MethodCorrectDiv , no “DocArqWiki” usando o “PadraoComponenteTemplate”.

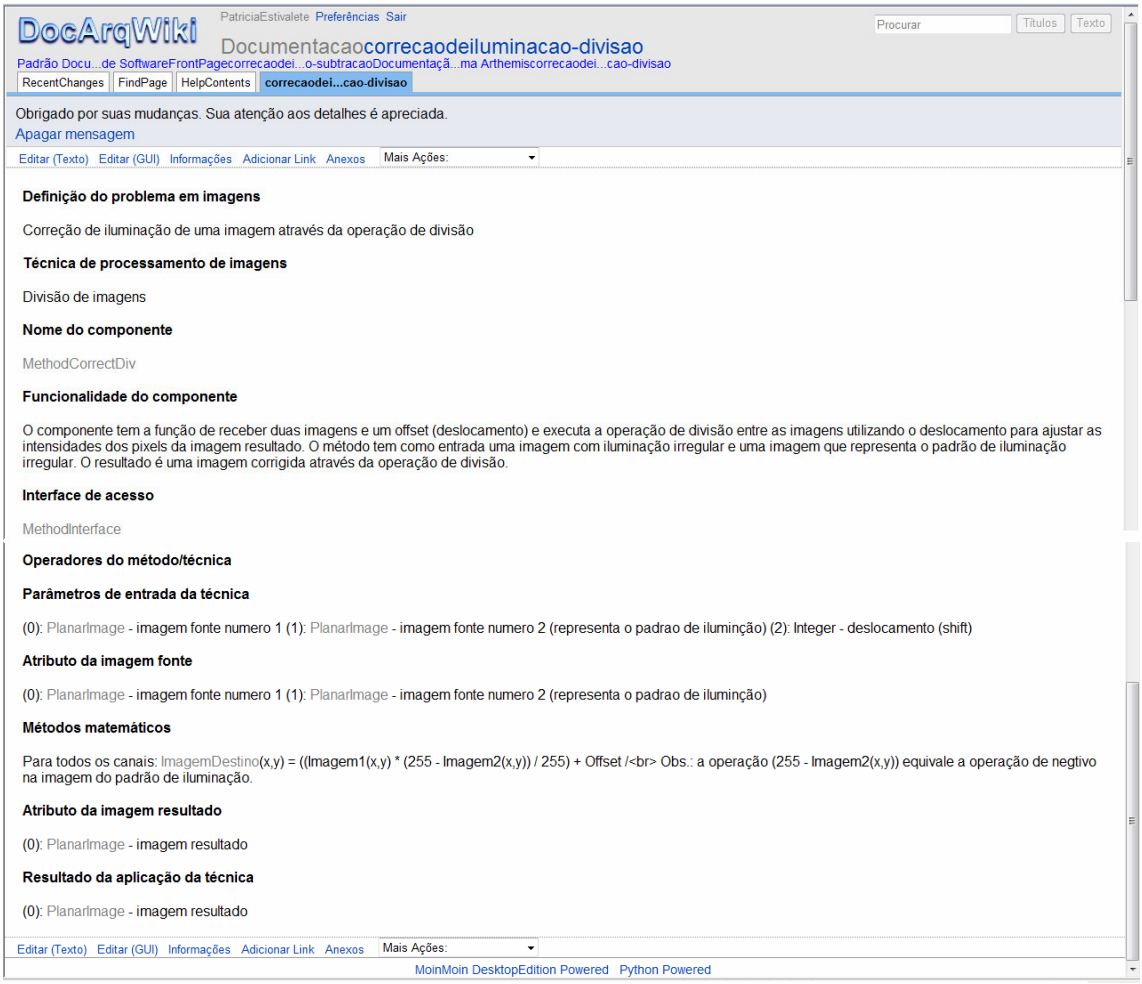

Figura 30: Tela de documentação do componente “MethodCorrectDiv”

7.4.1.3 Componente de software “MethodGenGauss”

O componente tem a finalidade de receber uma imagem, o tamanho do kernel e o número de passagens do filtro. O kernel é gerado pela equação de Gauss, sendo aplicado através da operação de convolução na imagem definida como fonte. O número de passagens do kernel é definido na entrada. A operação de convolução utiliza a "copia de bordas" para tratar as bordas da imagem. A imagem resultado da operação é colocada no vetor de saída.

A seguir, na Figura 31, é mostrado a documentação do componente de software, MethodGenGauss , no “DocArqWiki” usando o “PadraoComponenteTemplate”.

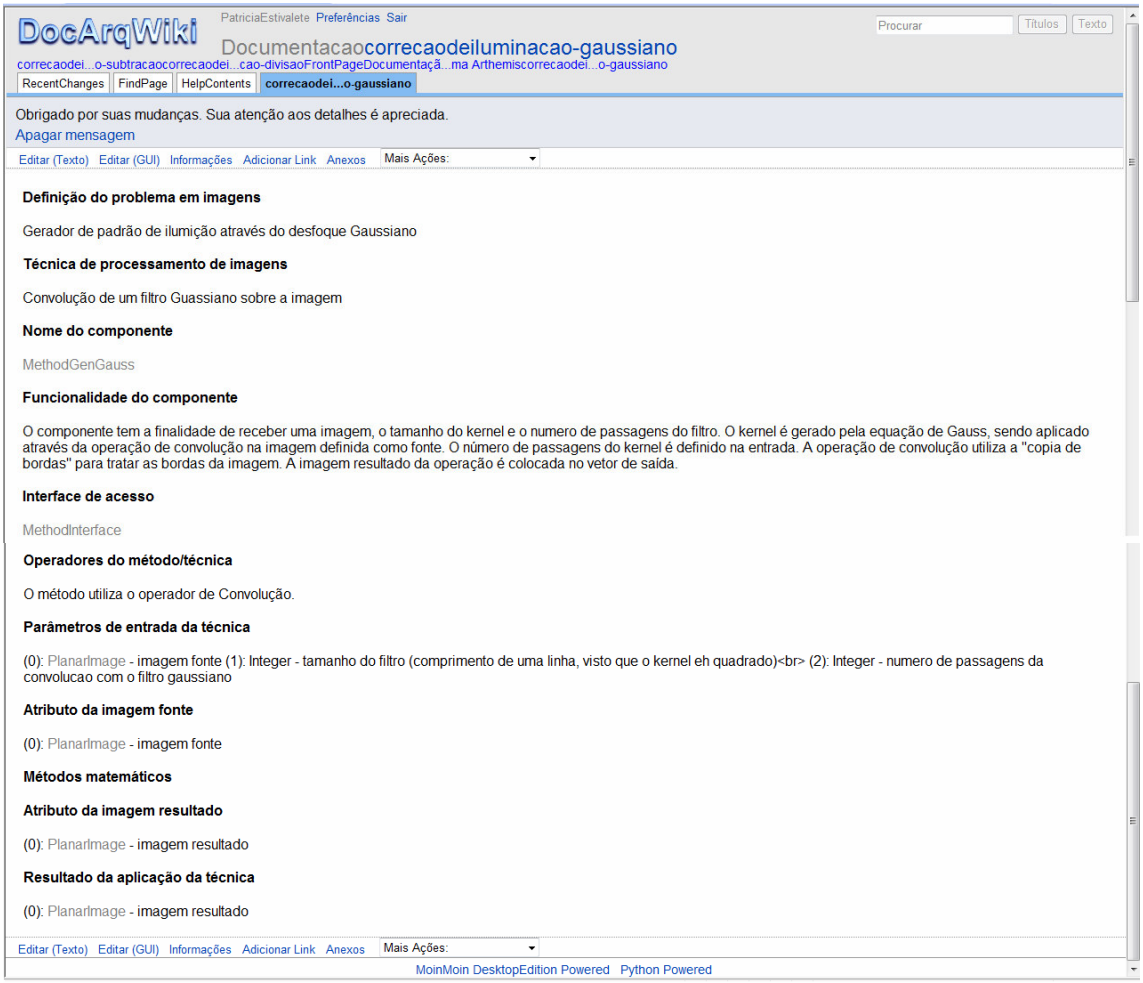

Figura 31: Tela de documentação do componente “MethodGenGauss”

7.4.1.4 Componente de software “MethodGenInt”

O componente tem a funcionalidade de receber uma imagem, o número de linhas e colunas em que a imagem foi dividida e retângulos correspondentes as seleções do fundo da imagem. Para cada posição (linha, coluna), os retângulos correspondentes são recortados e calculados o pixel mediano dos mesmos. A matriz (linha, coluna) é preenchida com os pixels medianos formando uma imagem, e essa será ampliada através de uma interpolação bicúbica gerando o padrão de fundo da imagem. A operação de escala utiliza a "copia de bordas" para tratar as bordas da imagem. A imagem resultado da operação é colocada no vetor de saída.

A seguir, na Figura 32, é mostrado a documentação do componente de software, MethodGenInt , no “DocArqWiki” usando o “PadraoComponenteTemplate”.

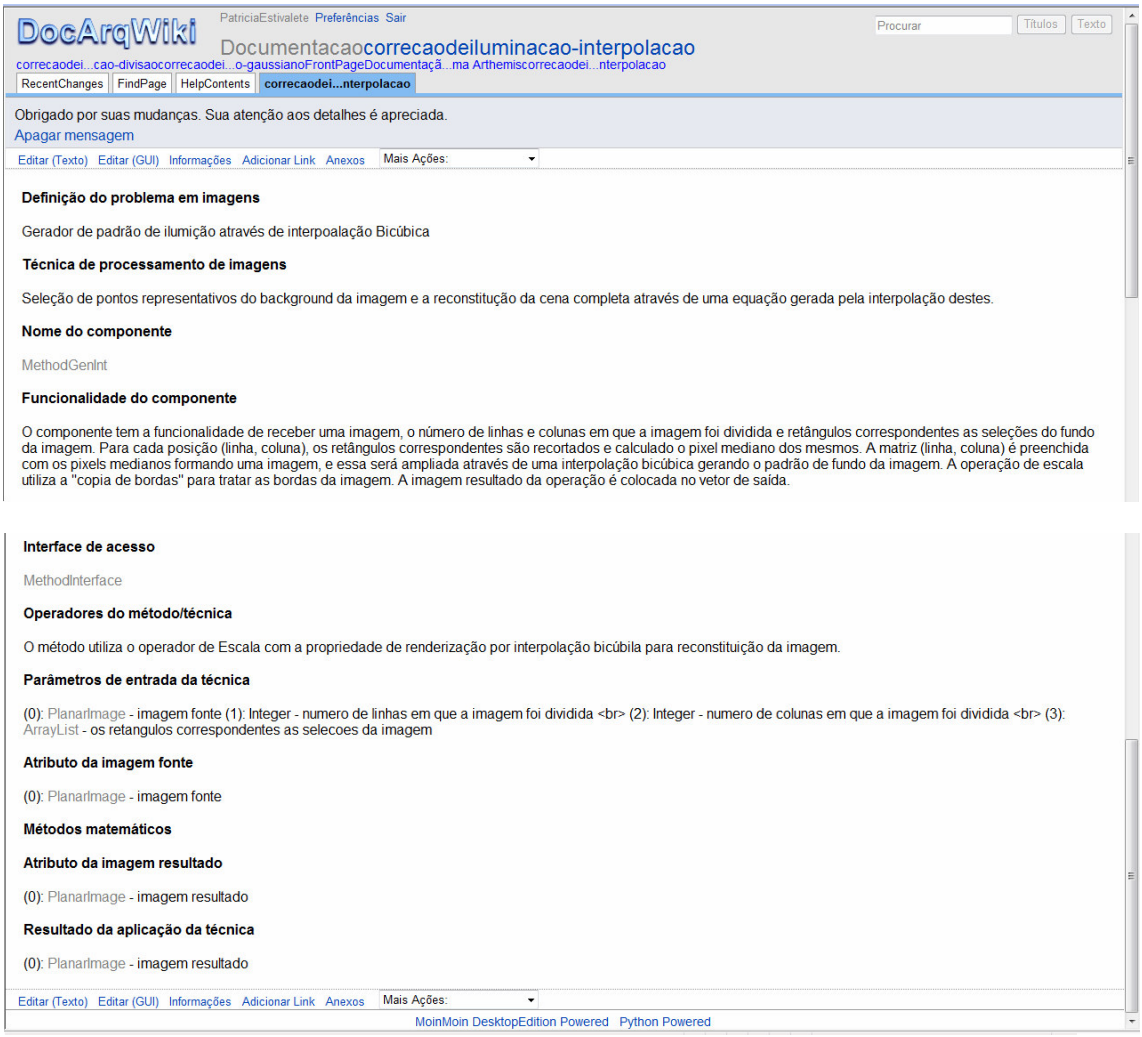

Figura 32: Tela de documentação do componente “MethodGenInt”

7.4.1.5 Componente de software “CalcPhase”

O componente CalcPhase, com base em um padrão de histogramas aproximados de componentes homogêneos presentes em uma ou mais imagens, juntamente com o histograma de cada imagem na qual quer-se calcular as frações de área dos componentes presentes, utiliza a análise linear de histogramas para a determinação destas frações.

A seguir, na Figura 33, é mostrado a documentação do componente de software, CalcPhase, no “DocArqWiki” usando o “PadraoComponenteTemplate”.

Figura 33: Tela de documentação do componente “CalcPhase”

7.4.2 Uso do “PadraoInteracaoComponenteTemplate”

Através do uso do “PadraoInteracaoComponenteTemplate” é possível catalogar às mensagens de comunicação que ocorrem entre os componentes para o funcionamento das técnicas de processamento e análise de imagens existentes no sistema “Arthemis”.

7.4.2.1 Interação entre os componentes da técnica “Correção de Iluminação”

Na Figura 34, é mostrada a documentação da interação entre os componentes da técnica, correção de iluminação, no “DocArqWiki” usando o “PadraoInteracaoComponenteTemplate”.

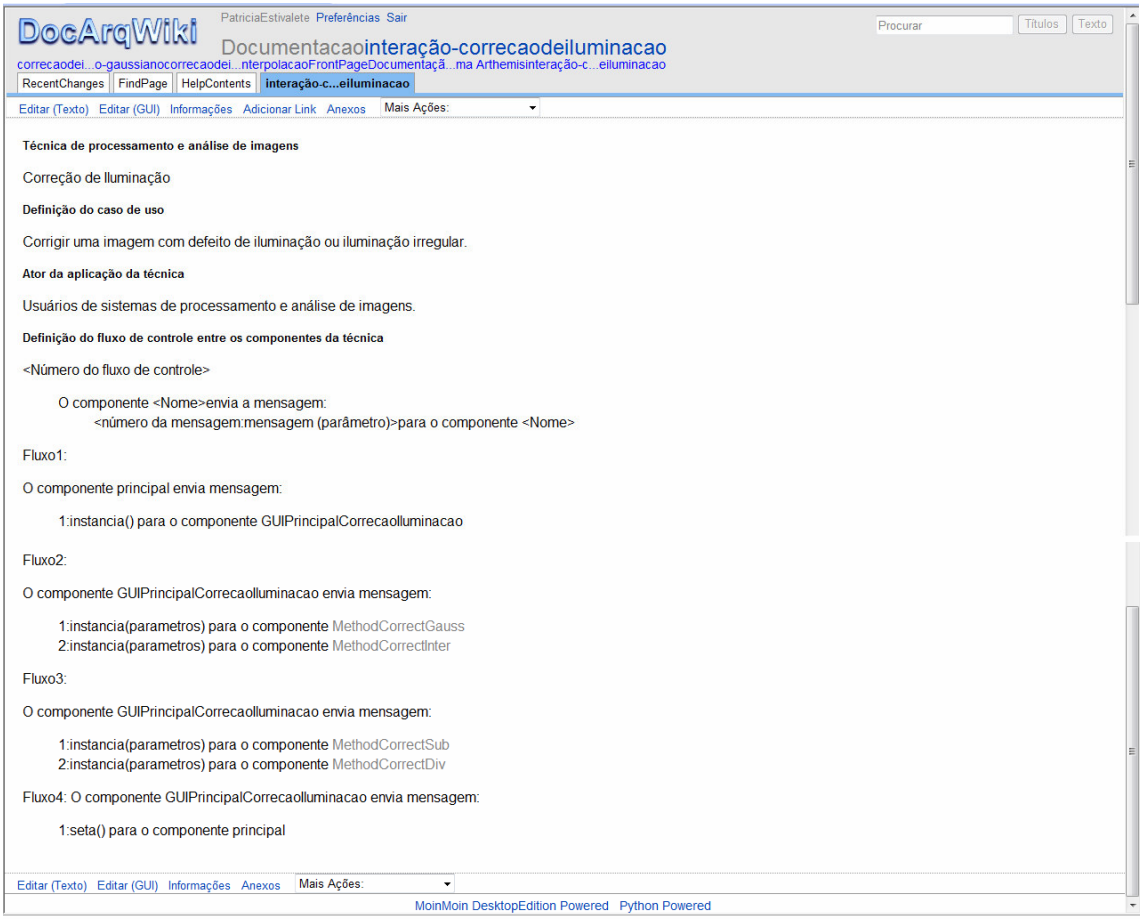

Figura 34: Tela de documentação da técnica “Correção de Iluminação”

7.4.2.2 Interação entre os componentes da técnica “Cálculo de frações de área de um conjunto de imagens”

Na Figura 35, é mostrada a documentação da interação entre os componentes da técnica, cálculo de frações de área de um conjunto de imagens, no “DocArqWiki” usando o “PadraoInteracaoComponenteTemplate”.

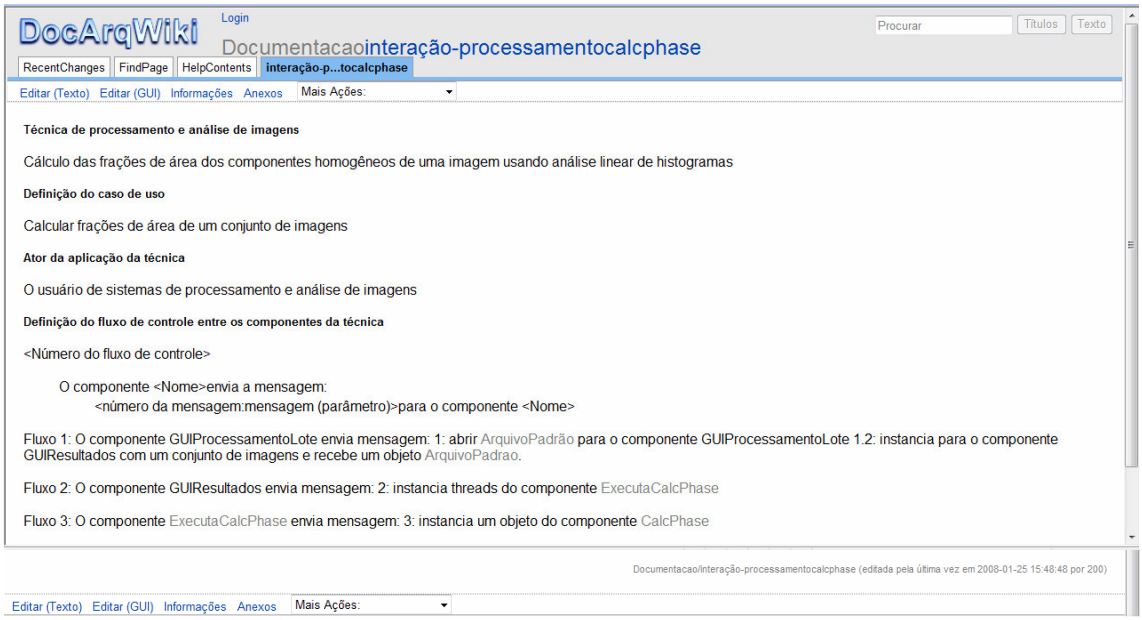

Figura 35: Tela de documentação da técnica “Cálculo de frações de área de um conjunto de imagens”

7.4.3 Uso do “PadraoCasoUsoTemplate”

Através do uso do “PadraoCasoUsoTemplate” é possível catalogar os fluxos de eventos, isto é, as ações dos atores com as técnicas de processamento e análise de imagens existentes no sistema “Arthemis”.

7.4.3.1 Caso de uso da técnica “Correção de Iluminação”

Na Figura 36, é mostrada a documentação do caso de uso da técnica, correção de iluminação, no “DocArqWiki” usando o “PadraoCasoUsoTemplate”.

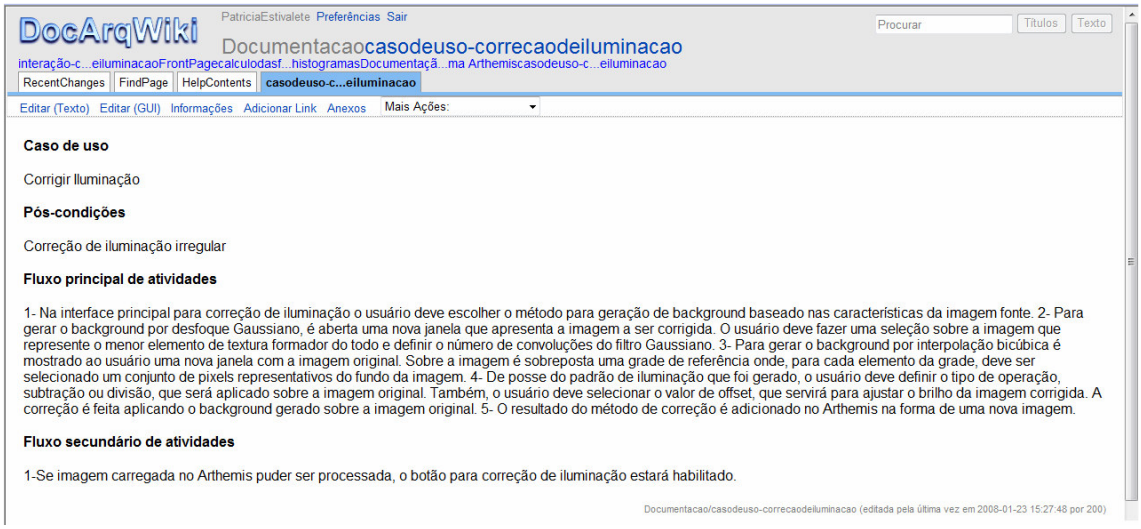

Figura 36: Tela de documentação do caso de uso da técnica “Correção de Iluminação”

7.4.3.2 Caso de uso da técnica “Cálculo de frações de área de um conjunto de imagens”

Na Figura 37, é mostrada a documentação do caso de uso da técnica, cálculo de frações de área de um conjunto de imagens, no “DocArqWiki” usando o “PadraoCasoUsoTemplate”.

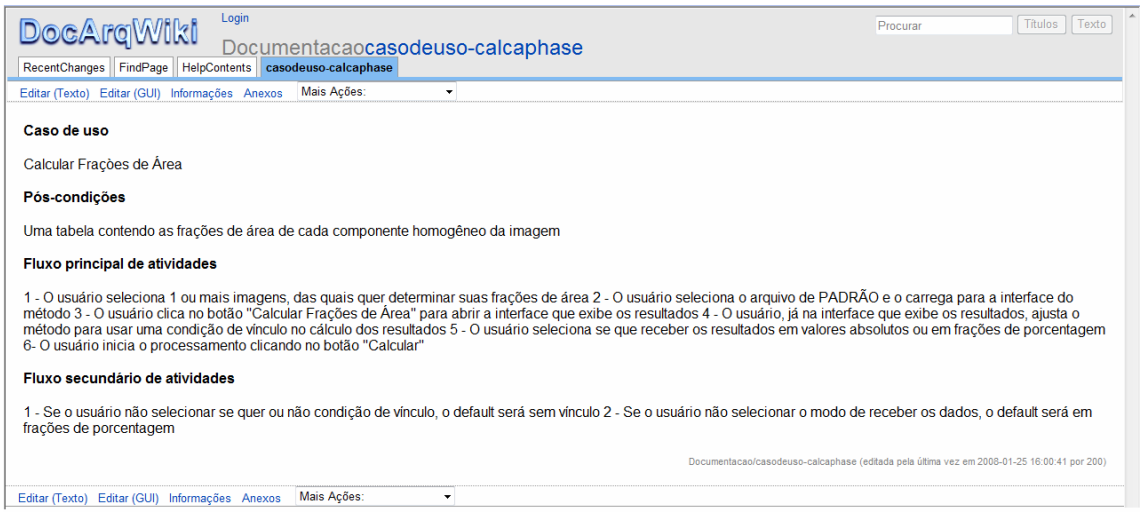

Figura 37: Tela de documentação do caso de uso da técnica “Cálculo de frações de área de um conjunto de imagens”

8 RESULTADOS FINAIS

A mais relevante contribuição apresentada pelo trabalho realizado é a definição de Padrões de Documentação da Arquitetura de sistemas e frameworks para processamento e análise de imagens. Esses são constituídos através da visão lógica e de caso de uso da UML, as quais representam características importantes que devem ser documentadas na arquitetura de sistemas.

Esses Padrões foram propostos de modo que sejam utilizados para documentar a arquitetura de sistemas e frameworks na área de imagens, otimizando a criação de documentos de forma padronizada.

Desenvolveu-se ainda neste trabalho um ambiente colaborativo, chamado "DocArqWiki", o qual dispõem de um conjunto de *templates* dos Padrões de Documentação a serem seguidos pelos desenvolvedores, contemplando as características de sistemas e frameworks na área de imagens.

Para criação desses *templates*, foram contempladas as características do elemento, Solução, de cada Padrão de Documentação, de modo que essa possa ser usada inúmeras vezes em situações semelhantes.

Outra grande relevância apresentada pela pesquisa é o desenvolvimento de um ambiente de trabalho colaborativo, o qual, através das características de cooperação, coordenação e comunicação, proporcionam aos usuários compartilhar conhecimentos na construção da documentação da arquitetura.

Tendo os *templates* disponíveis em um ambiente colaborativo via web, como o "DocArqwiki", tornou-se possível construir a documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, através de regras claras e objetivas e, ainda proporcionar aos usuários a colaboração a qualquer tempo.

Ainda como resultado desta pesquisa foi possível adicionar novas contribuições à área de *design patterns*, documentação da arquitetura de software e processamento e análise de imagens.

8.1 Conclusões e trabalhos futuros

O trabalho desenvolvido contribui de modo significativo para a área de documentação da arquitetura de sistemas e frameworks para processamento e análise de imagens, trazendo

uma fonte de pesquisa, de experimentos realizados e resultados obtidos, onde os usuários e pesquisadores poderão basear-se para o desenvolvimento de outras propostas similares.

Neste trabalho, importantes decisões de construção de Padrões de Documentação da Arquitetura de sistemas e frameworks na área de imagens foram tomadas e colocadas em prática, tendo a "DocArqWiki" como grande aliada ao tema em questão.

Os Padrões apresentados apresentaram uma relevante utilização para a documentação da arquitetura e oferta de serviços padronizados aos usuários, e com a utilização desses pelo "DocArqWiki", pode-se propor a criação de documentos baseados nos *templates* oferecidos pelo ambiente.

Através do uso do "DocArqWiki" pode-se aplicar de forma prática a importância do uso de Padrões de documentação da arquitetura para domínios específicos, caracterizando cada área e, ainda o quanto de benefícios ela oferece, mesmo que em uma versão inicial, a qual, no entanto, já contempla e supre o trabalho de mestrado proposto.

O "DocArqWiki" foi desenvolvida em uma ferramenta wiki *open source*, estando disponível para testes e maiores contribuições de funcionalidades no endereço *http://laca2.inf.ufsm.br:8080/*.

No entanto, deve ser observado que para sua utilização deve-se seguir a licença GPL (*General Public License*) a ela atribuída. É permitida sua utilização, ou modificação se necessário, sem que os autores percam o direito autoral e de forma que a ferramenta continue estando aberta à comunidade de pesquisa e desenvolvimento.

Neste trabalho foram utilizadas *templates* para documentação da arquitetura, no entanto, podem ser incluídas novas padronizações para facilitar e otimizar a geração de documentos. Essas novas funcionalidades, acopladas à "DocArqWiki" irão otimizar os resultados tanto em cumprimento de regras a serem seguidas quanto em repositório de informações aos usuários.

Como trabalhos futuros pode-se citar um melhoramento nas funcionalidades atribuídas ao "DocArqWiki", de modo que a documentação da arquitetura seja melhorada. Algumas sugestões de trabalhos futuros são:

• Permitir que a "DocArqWiki" disponha recursos, como administração de direitos, a fim de cadastrar as preferências dos usuários, selecionando o tipo de usuário, o grupo que será incluído e as permissões de acesso;

• Implementar novos Padrões de documentação no "DocArqWiki" que contemple as outras fases do ciclo de vida de desenvolvimento de software;

• Possibilitar aos usuários a organização de seus documentos por domínio de conteúdo, facilitando a administração das informações e os direitos de usuários e grupos e, ainda otimizando as buscas por assunto;

• Possibilitar que os usuários utilizem um sistema de troca de mensagens através do ambiente e que estas fiquem armazenadas para que o coordenador possa consultá-las;

• Propõe-se ainda um melhoramento na interface da "DocArqWiki" com vistas a cada vez torná-la mais amigável ao usuário, facilitando sua utilização;

• Ainda é sugerido como trabalho futuro o desenvolvimento de um *plug-in* acoplado ao DocArqWiki", possibilitando a modelagem da arquitetura através dos diagramas UML.

Tal pesquisa traz como benefício a colaboração para o meio acadêmico, contribuindo para um avanço tecnológico, onde toda a comunidade de desenvolvedores de sistemas e frameworks para área de imagens e áreas correlatas, poderão se beneficiar com a pesquisa.

Durante o desenvolvimento do trabalho de mestrado e posterior escrita da dissertação, foi possível obter a aprovação de artigos referentes às pesquisas desenvolvidas, validando dessa forma o estudo apresentado.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M. A Pattern Language. Oxford University Press, New York. 1977.

ASSIS, R. L. Facilitando a Percepção em Ambientes Virtuais de Aprendizagem através da Abordagem de Groupware. (Dissertação de Mestrado). Pontífica Universidade Católica do Rio de Janeiro, PUC-RJ, Rio de Janeiro, 2000.

BACHMANN, F. et al. Software Architecture Documentation in Practice: Documenting Architectural Layers. 2000. CMU/SEI. Disponível em: http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00sr004.pdf. Acesso em: 14 ago. 2006.

BAKER, K.; GREENBERG, S.; GUTWIN, C. (2001) Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. 8th IFIP International Conference (EHCI 2001). Lecture Notes in Computer Science Vol. 2254, pp. 123-139, Springer-Verlag.

BASS, L., CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice, Addison Wesley, 2003.

BLOIS, A. P., BECKER, K., 2002, A Component-based Architecture to Support Collaborative Application Design. Lecture Notes in Computer Science. Berlin: Springer Verlag, v. 2440, pp. 134-144. , La Serena - Chile.

BOOCH, G.; RUMBAUGH, A.; JACOBSON, I. UML: Guia do Usuário. v 1. São Paulo: Editora Campus. 2005. 474 p.

BORCHES, J. A Pattern Approach to Interaction Design. In: Conference on Designing interactive systems: processes, practices, methods, and techniques. New York, United States: 2000. Disponível em: <http://portal.acm.org/citation.cfm?id=558433&coll=Portal&dl=GUIDE&CFID=3720139 &C FTOKEN=24769028#>. Acesso em: 16 jul. 2007.

BORGES, M. R. S.; CAVALCANTI, M. C. R.; CAMPOS, M. L. M. Suporte por computador ao trabalho cooperativo, XV Congresso da Sociedade Brasileira de Computação, Canela, RS, 1995.

BUSCHMANN, F. et al. Pattern-Oriented Software Architecture. John Wiley and Sons, New York, NY, 1996.

BRINCK, T.; MCDANIEL, S. E. (1997), "Awareness in Colaborative Systems", Workshop Report, SIGCHI Bulletin.

BRINCK, T. Groupware: Introduction.1998. Disponível em: <http://www.usabilityfirst.com/groupware/intro.txl>. Acesso em: 15 jun. 2007.

BROWN, A. Large-Scale Component-Base Development, Prentice Hall. 2000.

CHANG, C. J; ZHANG, J, JIANG, T. M. “Formalization of Computer Supported Cooperative Work Applications”. Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS’01). 7 pg. 2001.

CARVALHO, S. Um Design Pattern Para a Configuração de Arquiteturas de Software. Dissertação de Mestrado. IC/UFF, Niterói-RJ, 2001.

CHEESMAN, J., DANIELS, J., 2000, UML components: a Simple Process for Specifying Component-based Software, Addison-Wesley Longman Publishing Co., Inc.

CLEMENTS, P. C., NORTHROP, L. M.: Software Architecture: An Executive Overview. Technical report CMU/SEI-96-TR-003 (1996)

COLLINS-COPE, M., MATTHEWS, H., 2000, "A Reference Architecture for Component-Based Development". In: 6th International Conference on Object Oriented Information Systems (OOIS'2000), pp. 225-237, London, UK, 18-20 December.

COPLIEN, J. O. e HARRISON, N. B. Organizational Patterns of Agile Software Development. Prentice Hall PTR. 2004.

DAVIS, L.; GAMBLE, R. F.; and PAYTON, J. (2002); The impact of component architectures on interoperability. System & Software; 61(1):31–45. Disponível em: < http://www.seat.utulsa.edu/papers/DGP02.pdf>. Acesso em: 21 jun 2007.

DAVENPORT, T. H.; PRUSAK, L. Conhecimento empresarial: como as organizações gerenciam o seu capital intelectual. 8. ed. Rio de Janeiro: Campus, 2004. 237p.

DRON J.; BOYNE C.; MITCHELL R. (2001) “Footpaths in the stuff swamp” Proceedings of WebNet'2001, Fowler, W. & Hasebrook, J. (eds.), WorldConference of the WWW and Internet, Orlando, FL, AACE, pp. 323- 328.

D'SOUZA, D. F., WILLS, A.C., 1999, Objects, components, and frameworks with UML: the catalysis approach, Addison-Wesley Longman Publishing Co., Inc.

ELLIS, C. A.; GIBBS, S. J.; REIN, G. L. Groupware: Some issues and experiences. In: Communications of the ACM. 34 (1): 38-58 p., Jan. 1991.

EMMERICH, W., ELLMER, E., FIEGLEIN, H., 2001, "TIGRA – An architectural style for enterprise application integration". In: 23rd International Conference on Software Engineering (ICSE, 2001), pp. 567-576, Toronto, Canada, May.

FIDEL, R. Quality Methods in Information Retrieval Research. 1993. Disponível em: <http://www.ischool.washington.edu/fidelr/RayaPubs/QualitativeMethodsinInformationRetrievalResearch.pdf>. Acesso em: 07 jan. 2008.

FUKS, H.; RAPOSO, A. B.; GEROSA, M. A. Do Modelo de Colaboração 3C à Engenharia de Groupware. Disponível em: <http://www.tecgraf.pucrio.br/publications/artigo_2003_colaboracao_3c_engenharia_groupware.pdf>. Acesso em: 05 jun. 2007.

FUKS, H.; RAPOSO, A.; GEROSA, M.A. "Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas", XXI Jornada de Atualização em Informática, Anais do XXII Congresso da Sociedade Brasileira de Computação, v.2, Cap. 3, ISBN 85-88442-24-8, pp 89-128. 2002

FUSSELL, S.R.; KRAUT, R. E.; LEARCH, F.J.; SCHERLIS, W.L.; McNally, M.M.; CADIZ, J.J. (1998) "Coordination, overload and team performance: effects of team communication strategies", Proceedings of CSCW '98, Seattle, USA, p. 275- 284. ISBN 1-58113-009-0.

FREITAS, G. D. Athena: Um Framework para a Construção de Interfaces Humano-Computador no Domínio da segmentação de imagens. Dissertação de Mestrado — Programa de Pós- Graduação em Engenharia de Produção (PPGEP) - Universidade Federal de Santa Maria, 2006.

GAMMA, E., et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.

GAMMA, E. et al. Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos. Bookman, Porto Alegre. 2000.

GIL, A. C. Como elaborar projetos de pesquisa. 4ª Ed. São Paulo: Atlas. 2002. 171 p.

GIMENES, I. M. S., BARROCA, L., HUZITA, E. H. M., and CARNIELLO, A. (2000). O processo de desenvolvimento baseado em componentes através de exemplos. VIII Escola de Informática da SBC-Sul, pages 147–178.

GUTWIN, C.; STARK, G.; GREENBERG, S. Support for Workspace Awareness in Educational Groupware. Pro ACM Conference on Computer Supported Collaborative Learning. Indiana, USA: 147-156 p., 1995.

HAETINGER, D., ABEGG, I., COSTA., J., TAROUCO, L., FOOHS, M., SANTOS, F., SCHIMITT, M., RUDUIT, R., LINDEMANN, V. Twiki, uma ferramenta de co-autoria livre. Novas Tecnologias na Educação, Porto Alegre, v.3, nº2, nov. 2005.

HEINEMAN, G. T., COUNCIL, W. T. Component-Based Software Engineering: Putting the Pieces Together. 1ª. Ed, Addison-Wesley, 2001.

HERZUM, P., SIMS, O., 2000, Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, OMG Press.

HILLIARD, R.: Building Blocks for Extensibility in the UML: Response to UML 2.0 Request for Informatio n. Integrated Systems and Internet Solutions Inc, Concord, Massachusetts. December. 1999.

HOPKINS, J., Component Primer: laying the Foundation. Communications of the ACM, v.43, n.10, p. 27- 30, Oct. 2000. Disponível em: < http://doi.acm.org/10.1145/352183.352198> Acesso em: 20 jun 2007. IEEE Std 1471-2000.: Recommended Practice for Architectural Description of Software Intensive Systems. 2000 JAZAYERI, M., RAN, A., LINDEN, F. V. D. Software Architecture for Product Families. Addison Wesley, 2000.

KRAFZIG, Dirk; BANKE, Karl; SLAMA, Dirk. Enterprise SOA: Service-Oriented Architecture Best Practices. Indianapolis: Prentice Hall, 2004.

LEUF, B., CUNNINGHAM, W. The Wiki Way: Quick Collaboration on the Web. BostonAddison Wesley Longman, 2001.

LOBATO, L. L. e ZORZO, S. D. Estudo de caso da avaliação por inspeção em sites de comércio eletrônico. DC - UFSCar (Universidade Federal de São Carlos). São Carlos, SP, p.94. 2007b. (02/2007).

LOUREGA, L. V. MeSegHi - Um Método de Segmentação que Utiliza o Processamento Linear e Não-Linear em Imagens. Dissertação de Mestrado — Programa de Pós-Graduação em Engenharia de Produção (PPGEP) - Universidade Federal de Santa Maria, 2006.

MELODY, W. (1994), "Electronic Networks, Social Relations and the Changing Structure of Knowledge", Communication Theory Today, pp. 255-273, Stanford University Press. 1994.

MENDES, A., 2002, Arquitetura de Software: desenvolvimento orientado a arquitetura, Rio de Janeiro, Editora Campus.

MERSON, P. Como Documentar Arquitetura de Software. 2001. Simpósio Brasileiro de Engenharia de Software SBES. Disponível em: http://www.sbbdsbes2005.ufu.br/arquivos/Merson05_minicurso_SBES1.pdf. Acesso em: 5 jul. 2006.

MESZAROS, G. e DOBLE, J. MetaPatterns: A Pattern Language for Writing Patterns. 1996. Conference on Pattern Languages of Programming PLoP. Disponível em: <http://www.hillside.net/patterns/writing/patternwritingpaper.htm>. Acesso em: 3 dez. 2006.

MILER, N., 2000, A Engenharia de Aplicações no Contexto da Reutilização baseada em Modelos de Domínio, Dissertação de M.Sc, COPPE, UFRJ, Rio de Janeiro, Brasil.

MÖLLER, Erik. Autoria Coletiva: Instalando e Mantendo o Mediawiki. Publicado em Linux Magazine, 16 julho 2005. Disponível em: <http://www.linuxmagazine.com.br/issue/01/mediawiki.html>. Acesso em: 2 de Jun de 2007.

OLIVEIRA, I. R. D.; BALBY, L. e GIRARDI, R. Padrões baseados em Agentes para a Modelagem de Usuários e Adaptação de Sistemas. In: SugarLoafPLoP – Conferência Latino-Americana em Linguagens de Padrão para Programação. Fortaleza, Ceará: 2004. Disponível em: <sugarloafplop2004.ufc.br/acceptedPapers/ww/WW_18.pdf> Acesso em: 19 fev. 2007.

OMG. Unified Modeling Language Specification Version 2.0. 2006. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em: 10 mar. 2006.

PONTES, A. M., SOUZA; C. S.; BARBOSA, S. D. J. Organização Conversacional: Inspeção das Representações na Wikipédia. CLIHC'05, Cuernavaca, México, out. 2005.

PRIMO, A. F. T., RECUERO, R. C. Hipertexto cooperativo: uma análise da escrita coletiva a partir dos Blogs e da Wikipédia. Revista FAMECOS: Mídia, Cultura e Tecnologia, Porto Alegre, n.22, p.54-65, dez. 2003.

RAPOSO, A.B., MAGALHÃES, L.P., RICARTE, I.L.M., FUKS, H. 2001. Coordination of collaborative activities: A framework for the definition of tasks interdependencies. 7th International Workshop on Groupware - CRIWG 2001, 170-179.

RAPOSO, A.; FUKS, H. "Defining Task Interdependencies and Coordination Mechanisms for Collaborative Systems". in: Blay- Fornarino, M., Pinna-Dery, A. M., Schmidt, K. & Zaraté, P.; Cooperative Systems Design (vol 74 of Frontiers in Artificial Intelligence and Applications), ISBN 1-58603-244-5 IOS Press, Amsterdam, pp 88-103. 2002

RUSS, J. C. The image processing handbook. 3.ed. Raleigh, North Carolina: CRC Press LLC, 1999.

SHAW, M.: The Coming-of-Age of Software Architecture Research. Proceedings of the 23rd International Conference on Software Engineering - ICSE 2001. Keynote address. Toronto, Canada: 2001. Disponível em: < http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/shaw-keynote-rev.pdf>. Acesso em 30 jul. 2007.

SHAW, M., GARLAN, D., 1996, Software Architecture: Perspectives on an Emerging Discipline, New Jersey, Prentice-Hall.

SCHMITT, M. A. R. Dificuldades apresentadas pelo modelo *wiki* para a implementação de um ambiente colaborativo de aprendizagem. 2006. Disponível em: < http://www.cinted.ufrgs.br/renote/nov2005/artigosrenote/a77_TWiki.pdf>. Acesso em 3 jun 2007.

SIEBRA, S.A.; SALGADO, A.C.; TEDESCO, P.A.; BRÉZILLON, P. (2005) "Identifying the Interaction Context in CSCLE", Proceedings of the 5th International and Interdisciplinary Conference (CONTEXT 2005), Paris, France, July 5-8, Lecture Notes in Computer Science, Vol 3554, Springer-Verlag.

SIEMENS, G. Uma teoria de aprendizagem para a idade digital. Competências profissionais. 2004. Disponível em: <http://www.webcompetencias.com/textos/conectivismo.htm>. Acesso em 10 jul 2007.

SMYSER, B. M. Active and Cooperative Learning, 1993.

SOMMERVILLE, I. Engenharia de Software**.** São Paulo: Addison Wesley, 2004.

STAFFORD, J.A.; WOLF, A.L. (2001) Software Architecture, in: Component-Based Software Engineering: Putting the Pieces Together, Hineman, G.T. & Councill, W.T. (eds), Addison-Wesley, ISBN 0-201-70485-4.

TELES, V.M. (2004) Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade, ed. Novatec, ISBN 8575220470.

VITHARANA, P., ZAHEDI, F., JAIN, H., 2003, "Design, Retrieval, and Assembly in Component-based Software Development", Communication of the ACM, v. 46 N 11. November, 2003.

XIA, C.L., MICHAEL. R.; WONG, K., "Component-Based Software Engineering: Technologies, Quality Assurance Schemes, and Risk Analysis Tools". In: *7th Asia-Pacific Software Engineering Conference (APSEC'2000)*, pp. 372 – 379 Singapura: 2000. Disponível em: < http://citeseer.ist.psu.edu/492347.html> Acesso em 30 jul. 2007.

YIN, R. K. Estudo de Caso: Planejamento e método. 3ª Ed. Porto Alegre: Bookman. 2005.