Uma ferramenta Web colaborativa para apoiar a engenharia de requisitos em software livre

Marco Aurélio Graciotto Silva
Renata Pontin de Mattos Fortes

Laboratório de Engenharia de Software
Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo
Av. do Trabalhador São-Carlense, 400 - Centro - Cx. Postal 668
São Carlos - São Paulo - Brasil CEP 13560-970
{magsilva,renata}@icmc.usp.br

Abstract. Requirement engineering is a second-class citizen within free software projects development when facing the current state of art. However, free software is acknowledged as a high quality product, status that would be impossible to achieve if its user's requirements were not satisfied. Recent researches about free software development processes state that free software requirements are publicly asserted after the code development, depending upon the developer's abilities for a correct elicitation, analysis and specification of the requirements. However, there is a constant concern about documentating the source code but not the requirements. One reason is that there isn't a suitable tool to register those requirements and make them public, requiring the developers to use plain text files or web pages, whose difficult management hinders the constant information updating required by the process. A possible solution to the problem is the use of web enabled agile and collaborative edition tools, allowing fast documentation updates by any stakeholder. The Wiki/RE provides an environment that satisfies those needs, along with others requirements desired for a requirement management tool. The Wiki/RE, tool developed under this work, is a wiki that supports the collaborative development of requirements hyperdocuments, supporting the management and quality assessment of the requirements document.

Keywords: Requirement engineering, free software, wiki.

1 Introdução

A Engenharia de Requisitos, uma área chave da Engenharia de Software, é responsável pela criação e aperfeiçoamento das técnicas que permitam o correto desenvolvimento dos requisitos do software. Através da clara e precisa definição dos objetivos que o software deve cumprir para satisfazer aos seus usuários, evitam-se, ou, ao menos, reduzem-se os riscos de fracassos para os novos projetos. Avanços na área, tais como as técnicas de pontos de vista, casos de uso, metas e separação de interesses, refletem essa meta, aumentando a chance de sucesso de novos projetos

Entretanto, a mera utilização de técnicas adequadas — boas soluções locais — não é uma garantia de bons resultados. O processo de engenharia de software, como um todo, deve ser adequado, bem como as pessoas envolvidas. Dentre os processos mais recentes, o modelo de desenvolvimento de software livre ganha destaque, focando no aspecto global do processo e gerando produtos de boa qualidade.

Reis realizou um levantamento sobre projetos de software livre buscando a caracterização do processo utilizado [16]. Estudaram-se duas hipóteses sobre as atividades relacionadas a engenharia de requisitos que fazem parte desse processo e contribuem para o seu sucesso: o desensenvolvedor como usuário e a utilização de padrões abertos, projetos anteriores e programas pré-existentes.

Existe a percepção, na comunidade de software, de que uma parcela significativa dos softwares é criada por uma necessidade do autor, não atendida por softwares já existentes. Defende-se que todo bom software começa por atender uma necessidade pessoal de seu desenvolvedor [15]. De fato, 75% dos projetos analisados por Reis afirmaram que o desenvolvedor é um dos principais usuários do software [16, questão 1.2, p. 106]. Um segundo possível fa-

cilitador à engenharia de requisitos é a reutilização de requisitos de outros projetos. Em software livre, argumenta-se que vários projetos tentam replicar softwares já existentes ou implementam especificações abertas [16].

Se, para uma parcela significativa dos projetos, os requisitos são obtidos de especificações definidas por organizações como a IEEE e ISO, ou da replicação de funcionalidades de outros softwares, para os restantes desconhece-se a presença de documentos de requisitos ou do processo utilizado para obtê-los. Esse fato foi confirmado em outras pesquisas [17, 21].

Isso, em nenhum momento, implica que o processo de engenharia de software desses projetos não inclui atividades relativas a requisitos e projetos: apenas não se encontram documentos, bem formados, que resumem os resultados dos respectivos esforços de engenharia. De fato, evidências sobre atividades de engenharia de requisitos são encontradas nas listas de discussões, pedidos de alteração, manuais de utilização e documentos detalhados sobre pontos importantes de cada software.

Verifica-se que, diferentemente do que se espera nos projetos tradicionais, os requisitos, quando descritos, o são após sua implementação [17]. Sua disponibilização é feita em linguagem natural, por arquivos textos ou páginas Web editados sem o apoio de ferramentas especializadas. Sempre que possível, são criadas ligações entre esses documentos e as discussões relativas à implementação (que se encontram arquivadas em gerenciadores de listas de email). Não existe um padrão para essa documentação de modo a caracterizá-la como uma especificação de requisitos: ela é organizada o suficiente para que os desenvolvedores consigam encontrá-la futuramente.

Esse formato impõe alguns obstáculos na atualização e recuperação da informação. Sua natureza estática dificulta a atualização das informações, além de restringir os tipos de dados que podem ser inseridos. A ausência de estruturas de acesso para os arquivos e o alto volume típico das listas de discussões impedem que usuários encontrem, ao menos facilmente, as informações necessárias.

O primeiro objetivo deste trabalho é propor um modelo de documento e um processo que permitam a criação de especificações de requisitos que, ao mesmo tempo que atendam às normas, reduzam o volume de informações, principalmente as duplicadas, inseridas pelos desenvolvedores e o custo de manutenção, especialmente quanto à rastreabilidade dos dados. A natureza híbrida e dispersa das informações, bem como o modelo de desenvolvimento de software livre, distribuído e aberto, torna os hiperdocumentos, disponibilizados via Web, uma solução adequada para o tipo de documento em questão.

Um problema em hiperdocumentos é a manutenção das ligações entre seus diferentes nós. Especificações de requisitos compartilham essa mesma dificuldade. A utilização de uma ferramenta que facilite a atualização e recuperação de requisitos e o gerenciamento de seus inter e intra-relacionamentos é essencial. Um tipo de aplicação, relativamente recente, que satisfaz esses requisitos são as *wikis*.

O segundo objetivo deste trabalho é desenvolver a Wiki/RE, uma *wiki* que estende o modelo de *wikis* para o registro de diversos tipos de dados, permitindo a criação de hiperdocumentos multimídia de requisitos.

Através da investigação sobre hiperdocumentos para apoiar o armazenamento e a recuperação de documentos, artefatos e experiências produzidos durante a engenharia de requisitos, espera-se o uso mais efetivo das informações durante o processo de engenharia de requisitos e de software.

O restante deste artigo está organizado da seguinte forma. As seções 2 e 3 apresentam uma revisão da literatura de Engenharia de Requisitos e a aplicação de hiperdocumentos. A seção 4 discute *wikis*. A Wiki/RE é detalhadamente descrita na seção 5. Os resultados sobre o trabalho e trabalhos futuros são apresentados na seção 6.

2 Engenharia de Requisitos

A primeira etapa do processo de engenharia de software é o levantamento, a partir das metas atribuídas ao software, das operações a serem oferecidas e as condições e restrições aplicáveis. Elas são expressas por sentenças em linguagem natural, diagramas ou especificações formais obtidos pela aplicação de técnicas de engenharia. Esses produtos constituem os requisitos do software: uma propriedade que o software possui para resolver um problema no mundo real [9].

A Engenharia de Requisitos é responsável pelo estudo e desenvolvimento de técnicas que auxiliem e guiem o processo de definição de requisitos durante toda a vida do projeto. Através do desenvolvimento e aplicação de novas técnicas, torna-se possível a detecção prematura de erros, implicando em menores custos e prazos, e a satisfação das necessidades do usuário do software.

Os requisitos são usualmente classificados em funcionais e não-funcionais [19]. Os funcionais são aqueles que executam uma função, causando uma alteração no estado do sistema. Em uma primeira instância, esses requisitos seriam os mais importantes para os usuários, dado que os resultados daqueles

são necessários para atender às necessidades destes. No entanto, toda atividade é executada em um meio que impõe restrições. Essas condições que os requisitos funcionais devem observar são os requisitos não-funcionais.

Ambos os tipos de requisitos são obtidos ou definidos por pessoas interessadas nos resultados obtíveis da execução do software. Essas pessoas, denominadas *stakeholders*, incluem os usuários diretos e indiretos do programa, os desenvolvedores e as entidades que controlam o meio ao qual o software se aplica (governo, órgãos reguladores, etc).

A obtenção dos requisitos a partir dos interessados e sua correta definição em um formato útil ao desenvolvimento do software, a especificação de requisitos, requer a execução ordenada de diversas atividades, constituindo assim um processo. O processo de engenharia de requisitos é caracterizado pelas etapas de elicitação, análise, especificação e validação [9]:

Elicitação: Obtenção de dados e subseqüente descoberta de requisitos. As fontes investigadas são os próprios interessados, documentos, *softwares* similares, etc.

Análise: Organização e refinamento dos requisitos coletados na elicitação, resolução de requisitos conflitantes, criação de modelos conceituais ou de negócio.

Especificação: Definição precisa dos requisitos e dos atributos de qualidade aplicáveis.

Validação: Certificar-se que os requisitos foram corretamente definidos (verificação) e realmente satisfazem as necessidades de seus interessados (validação).

A execução do processo é acompanhada por métodos e técnicas, buscando um equilíbrio no esforço investido e as metas a serem alcançadas no desenvolvimento, mitigando os problemas intrínsecos e comuns à Engenharia de Requisitos: comunicação, determinação do domínio, imprecisão. Algumas das técnicas, descritas na literatura, que oferecem suporte às respectivas atividades de engenharia de requisitos são: léxico ampliado de linguagem, pontos de vista, metas, casos de uso, casos de mau uso, cenários, padrões, modelos de referência, métodos formais, inspeção, gerenciamento de requisitos, classificação e priorização de requisitos, dentre outros.

Os requisitos são reunidos em um documento denominado Especificação de Requisitos de Software. As seguintes características são desejáveis nesse documento [10]:

Corretitude: Uma especificação de requisitos é dita correta quando está de acordo com outros documentos previamente estabelecidos (como, por exemplo, a especificação de sistema), ou seja, apenas requisitos que o software deva cumprir devem estar definidos.

Ausência de ambigüidade: Um requisito não deve admitir mais de uma interpretação.

Completitude: Todo requisito do sistema deve estar contido na especificação ou devidamente referenciado, de maneira que esse documento seja suficiente para descrever o software. Cada requisito definido, por sua vez, deve ter suas entradas (válidas e inválidas) e saídas especificadas.

Consistência: Inconsistência é uma situação em que duas partes da especificação não obedecem a algum relacionamento mantido entre elas [6]. Esses relacionamentos podem se referir aos aspectos sintáticos e semânticos, além de relacionamentos inerentes ao próprio processo.

Classificação: Dentre os requisitos definidos na especificação, é natural uns terem importância maior do que outros. Essa classificação pode ser feita de acordo com os mais diversos fatores: volatilidade/estabilidade, importância, exigência legal e fatores organizacionais.

Verificabilidade: Para todo requisito da especificação, deve existir um mecanismo que permita avaliar se ele está sendo atendido.

Modificabilidade: Os requisitos são inerentemente voláteis. A estrutura da especificação de requisitos deve ser tal que mudanças possam ser acomodadas com o menor esforço possível. Isso implica uma boa modularização e organização, evitando redundâncias e buscando definir requisitos o mais atomicamente possível.

Rastreabilidade: Capacidade de saber as origens e os destinos dos requisitos.

A especificação de requisitos é um importante instrumento para unir todos os requisitos do software, produzidos pelas diversas técnicas empregadas durante o processo. Infelizmente, a união desses artefatos não é trivial, principalmente frente aos atributos de qualidade desejáveis ao documento e a diversidade dos artefatos gerados. Documentos hipermídia, hiperdocumentos, podem ser utilizados positivamente nesse ambiente heterogêneo.

3 Engenharia de Requisitos e Hiperdocumentos

Um conhecido problema nas especificações de requisitos é o custo de criação e manutenção da rastreabilidade entre os diversos elementos contidos na especificação. A identificação dos relacionamentos geral-

mente é uma tarefa manual, realizada após a criação dos artefatos. Sua atualização, necessidade constante frente à volatilidade dos requisitos, possui um custo considerável, agravado pela organização linear dos documentos.

Uma abordagem para a racionalização do gerenciamento de requisitos é a utilização de sistemas hipermídia. A representação dos artefatos em nós e os relacionamentos através de ligações e âncoras mostra-se apropriada para a engenharia de requisitos e para a geração e manutenção dos documentos de requisitos. Os hiperdocumentos produzidos também facilitam a leitura não seqüencial das especificações, contribuindo positivamente para o processo.

Os sistemas hipermídia ainda são pouco explorados em Engenharia de Requisitos. Na literatura científica, são descritos modelos e mecanismos para visualização de documentos de requisitos [2, 7], técnicas para para criação automática [2] ou semi-automática de ligações (rastreabilidade) [12, 8, 13] e ferramentas, como a RETH [11], REM [20] e HERE [14], que automatizam a aplicação dessas técnicas e modelos.

3.1 Modelos

As primeiras pesquisas analisavam o uso de sistemas hipermídia para a representação de dados multimídia, permitindo assim a criação de definições ricas (sons e vídeos) para questões (cuja representações baseadas em texto e diagramas são deficientes), além da exploração dos relacionamentos entre os diversos tipos de dados [1]. O raciocínio fundamenta-se em um ponto: "a maneira com que a informação é representada é freqüentemente um fator crítico na determinação do grau de dificuldade da solução para o problema; ou na averiguação de uma aplicação funcionar ou não". Sistemas hipermídia provêem meios para a aplicação de métodos adequados para representar fielmente os requisitos da aplicação.

Posteriormente, com o advento da Internet e de tecnologias abertas para criação de hiperdocumentos, trabalhos como o RQML [7] e REM [**?**] foram desenvolvidos.

O RQML é uma definição de documento XML específica para documentação de requisitos. A utilização de um hiperdocumento visa superar as dificuldades quanto ao uso de documentos feitos em editores de texto comuns, ao mesmo tempo em que garante a interoperabilidade não encontrada em soluções baseadas em bancos de dados. A linguagem foi definida após após um minucioso estudo sobre características desejáveis para hiperdocumentos, observando cada etapa da engenharia de requisitos. As seguintes características são consideradas relevantes e oferecidas, em grau variado, pela RQML: integração de outros artefatos (casos de uso, glossário), rastreamento (fonte das informações e identificação de *stakeholder*), controle de versão, reutilização e determinação de prioridades.

A REM é uma ferramenta experimental de apoio à engenharia de requisitos. Ela define um projeto composto por quatro componentes: documento de requisitos orientado ao cliente, documento de requisitos orientado ao desenvolvedor, registro de conflitos e defeitos de requisitos e registro de solicitações de alteração de requisitos. Os documentos são compostos por requisitos voltados ao cliente, requisitos voltados ao desenvolvedor e informações sobre os interessados. Os dados são armazenados em um banco de dados relacional, mas a especificação de requisitos é gerada como um documento XML. Sobre esse documento são aplicadas transformações XSLT para gerar os documentos que compõem o projeto, apresentar os dados (como páginas em HTML), mostrar a rastreabilidade entre os itens do documento e aferir a qualidade do documento de requisitos (ausência de ambigüidade, completitude, rastreabilidade).

3.2 Criação de ligações

Analisada a questão de retratar corretamente os requisitos, é necessário abordar o problema do estabelecimento de ligações entre os dados armazenados nos hiperdocumentos. Essas ligações podem ser determinadas de modo manual, semi-automático e automático. As ligações manuais são aquelas criadas pelo engenheiro de requisitos de modo *ad hoc*. Considerando os engenheiros infalíveis, elas são sempre corretas. Porém, elas também são as mais custosas. A identificação automática de relacionamentos, por outro lado, possui um custo zero de utilização. No entanto, apesar de ela utilizar-se de regras que reproduzem aquelas empregues pelos engenheiros, não é possível garantir que o conjunto de regras definido esteja completo: a criação de relacionamentos geralmente requer o uso de conhecimento tácito. Além disso, a maneira como o requisito é expresso (texto em linguagem natural, por exemplo) pode dificultar a identificação do conteúdo tratado no nó, impedindo a aplicação automática das regras. Para esses casos, existe a opção de ligações semi-automáticas: determinam-se, automaticamente, os possíveis relacionamentos entre os nós e oferece-se, ao engenheiro, a decisão quanto à validade das ligações [12]. A ferramenta RETH [11], por exemplo, provê facilidades para criação semi-automática de ligações. Ela identifica automaticamente, a partir dos termos pre-

sentes no dicionário de dados, potenciais associações no documento de requisitos. Essas são oferecidas ao usuário, que as classifica como corretas ou não. Outra possibilidade é a criação de uma associação *ad hoc* pelo usuário. Nesse caso, o sistema tenta aprender esse novo relacionamento, utilizando-o em subseqüentes execuções.

3.3 Ferramentas

As diferentes ferramentas hipermídia encontradas na literatura foram analisadas de acordo com as funcionalidades esperadas de sistemas hipermídias, julgadas adequadas para a Engenharia de Requisitos, e a criação de hiperdocumentos que acomodem, de maneira flexível e extensível, os artefatos gerados pelas diferentes técnicas de Engenharia de Requisitos. As características analisadas foram:

Ligações globais: as ligações globais, criadas automaticamente preferencialmente, são um importante mecanismo para relacionar assuntos ortogonais aos diversos métodos de engenharia. No RETH, elas são empregues para relacionar termos encontrados no texto a um dicionário.

Metadados em ligações: metadados em ligações permitem adicionar informações aos relacionamentos: questões levantadas e tipo de ligação de que se trata. Na XMLC, apesar de existirem vários tipos de ligações, os metadados não são utilizados na implementação.

Integração de e para outros métodos: a integração deve ser observada sob dois aspectos: o da utilização de artefatos gerados por outros métodos e o da possibilidade de uso do próprio hiperdocumento em outras etapas da engenharia.

Gerenciamento de configuração: fator importante para gerenciar os requisitos e que não é abordado em nenhuma ferramenta. A REM e a RQML permitem identificar a versão dos componentes do hiperdocumento, mas não oferecem sequer um mecanismo de controle de versão.

Verificação de atributos de qualidade: algumas características desejáveis no documento de requisitos podem ser verificadas automaticamente. A utilização de documentos XML validados, no caso do REM e RQML, contribui para garantir a completitude do documento. A REM ainda verifica a presença de ambigüidades e completitude.

A tabela 1 relaciona as técnicas revisadas neste capítulo e as características adicionais de sistemas hipermídia para apoio à engenharia de requisitos. As características presentes estão marcadas com um **X** e aquelas não informadas ou ausentes estão identificadas com um hífen.

A REM é a mais completa das ferramentas avaliadas. A capacidade de geração de documentos XML facilita não apenas a integração de e em outros métodos, mas também a aferição da qualidade do documento, por intermédio de transformações XSLT. Esse mesmo arquivo poderia ter suas versões gerenciadas por um sistema qualquer de sistema de controle de versão baseado em arquivos. As ligações globais e com metadados seriam possíveis com outras técnicas compatíveis com o XML, como XLink e RDF. No entanto, cabe ressaltar que a REM não é uma ferramenta Web e, para criação de hiperdocumentos em software livre, ferramentas Web são essenciais. É necessária a busca por um tipo de ferramenta Web que permita a implementação das características positivas aqui analisadas e comparadas: as *wikis*.

4 Wikis

A comunidade de padrões de software, em meados da década de 90, criou um *site* Web que funcionava como um repositório para descrição de padrões: o *Portland Pattern Repository*. A natureza do conteúdo disponibilizado, constituído de colaborações enviadas por pesquisadores do mundo inteiro, e a experiência positiva da utilização da ferramenta *HyperCard* para a documentação de padrões motivaram o desenvolvendo de uma ferramenta Web para complementar o repositório: a WikiWikiWeb [4].

O objetivo da WikiWikiWeb era fornecer um ambiente simples e ágil para edição colaborativa de hiperdocumentos. O nome "WikiWiki" é proveniente do havaiano *wikiwiki*, cujo significado é rápido, depressa. As aplicações Web que se assemelham a WikiWikiWeb são classificadas como *wikis*, em homenagem ao nome do primeiro software dessa categoria já construído.

4.1 Princípios

As *wikis* atendem um conjunto de princípios estruturados ao redor de um único princípio, considerado fundamental: confiança. Crer nas pessoas, para a adição e alteração de páginas, e no processo implícito das *wikis*, que permite a criação de um hiperdocumento complexo espontaneamente. Neste alicerce, é construído o seguinte conjunto de princípios [5]:

Aberta: Qualquer usuário poderá editar qualquer página.

Tabela 1: Características em hipermídia para apoio à engenharia de requisitos

Características	HERE	RETH	XMLC	RQML	REM
Ligações globais	-	X	-	-	-
Metadados nas ligações	-	-	-	-	-
Integração de outros métodos	-	-	X	X	X
Integração em outros métodos	-	-	X	X	X
Gerenciamento de configuração	-	-	-	-	-
Verificação de atributos de qualidade	-	-	-	-	X

Incremental: Uma página poderá referenciar outra página, mesmo que essa página não exista.

Orgânica: A estrutura e conteúdo do site estarão abertos para edição e evolução.

Mundana: Um número reduzido de convenções textuais proverá acesso para as marcações de página mais úteis.

Universal: Os mecanismos de edição e organização serão os mesmos.

Evidente: O conteúdo apresentado deverá sugerir a formatação de entrada dos dados.

Unificada: Os nomes das páginas estarão em um espaço de nomes único.

Precisa: Os nomes das páginas serão definidos de maneira precisa, evitando o conflito com outras páginas, e compostos por substantivos.

Tolerante: O comportamento interpretável será preferível ao uso de mensagens de erro.

Observável: Atividades serão observáveis e revisáveis por qualquer pessoa.

Convergente: Duplicações de conteúdo serão desencorajadas e removidas através da busca e referência a materiais similares ou relacionados.

4.2 Análise de *wikis*

Para a implementação da Wiki/RE, algumas *wikis* foram escolhidas para estudo, com o objetivo de facilitar a definição dos requisitos e sua posterior implementação. Em meados de 2003, realizou-se um levantamento de *wikis* livres, posteriormente avaliadas de acordo com critérios apropriados [3, 3] para a avaliação de softwares livres:

Software livre: A ferramenta deverá ser um software livre, disponibilizada preferencialmente por uma licença compatível com a GPL.

Qualidade do código: Quantidade de linhas de código, documentação da API, testabilidade (presença de casos de testes especificados e cuja execução seja automatizada por ferramentas como a jUnit), portabilidade (linguagem de implementação que facilite a portabilidade entre sistemas, tais como a linguagem Java), clareza do código, manutenibilidade (padrão de codificação, arquitetura bem definida, reutilização de código de outros projetos).

Uso: Dependências para e de outros softwares, facilidade de instalação.

Produtos do projeto: Alcance de metas planejadas, lançamento de versões alpha, beta e estáveis.

Processo de desenvolvimento: Quantidade de desenvolvedores, tempo entre lançamento de versões, verificação quanto a utilização de ferramentas de controle de versão e controle de alterações.

Seguimento dos princípios *wiki*: A linguagem utilizada na definição das páginas e o mecanismo de ligações são os itens de maior interesse quanto a esse quesito.

Expansibilidade: Possibilidade da adição de novas funcionalidades para edição ou apresentação dos dados contidos na página.

Controle de versão: Várias ferramentas implementam controle de versão para as páginas, possibilitando a recuperação de versões anteriores.

Adotou-se, como critério de pré-seleção, a linguagem utilizada na implementação. Definiu-se Java com a linguagem preferencial devido a facilidades para desenvolvimento de aplicações Web e portabilidade por ela oferecida. Foram escolhidas, de acordo com a popularidade da ferramenta, aferida por meio do *site* `http://freshmeat.net` à epoca do levantamento, três *wikis* Java para estudo: a JSPWiki, a Snipsnap e a VeryQuickWiki.

A comparação entre as *wikis* mostra um resultado favorável à JSPWiki. A ausência de controle de versões de páginas da Snipsnap teve uma importante influência no resultado, bem como o seu modelo de desenvolvimento, com lançamento inconstante de novas versões. A JSPWiki, apesar de sua equipe de desenvolvimento reduzida e sem dedicação exclusiva a mesma (ao contrário da Snipsnap), possui todas as características desejáveis de uma *wiki*, falhando apenas quanto ao modo utilizado para criação de ligações e a ausência de controle de alterações. A VeryQuickWiki, por sua vez, não possui destaques

importantes, mas também não possui deficiências notáveis.

Entretanto, o uso de linguagens ricas para descrição das páginas, presente em todas as ferramentas *wiki* analisadas, representa uma questão delicada. Apesar delas permitirem a edição de conteúdo mais rico, prejudica-se o quão evidente e mundana é a edição das páginas. A sintaxe da linguagem em si não é um problema, dado que os usuários podem ignorar as estruturas mais complexas. O problema reside no fato de o único modo de inserir tal conteúdo é através da linguagem definida.

Uma outra questão é a maneira com que os conteúdos ricos (não-textuais) são referenciados nas páginas, utilizando uma sintaxe diferente daquela empregada para as páginas. Isso afeta diretamente o princípio de universalidade e do desenvolvimento incremental. Um ponto forte das *wikis* é a criação da referência antes da criação do conteúdo, estabelecendo, transparentemente, a rastreabilidade entre os diferentes elementos do hiperdocumento. Para os elementos em diferentes mídias, o fluxo é invertido, sendo necessário criar o elemento antes de estabelecer o relacionamento.

Observadas essas consierações quanto atendimento aos princípios pelas *wikis* analisadas, verificou-se um problema não trivial no modo com que são tratados dados em diferentes mídias e tipos. Essa característica é essencial para a Wiki/RE, cujos diversos nós conterão artefatos gerados por técnicas diferentes de engenharia de requisitos. Decidiu-se, portanto, pela construção de uma nova *wiki*, sem tomar uma já existente como base.

5 A Ferramenta Wiki/RE

A Wiki/RE é uma *wiki* para engenharia de requisitos que disponibiliza um ambiente para construção de hiperdocumentos, com diferentes modos para a edição de conteúdo. Para isso, propõe-se um modelo em que os elementos do hiperdocumento não são sempre páginas. A escolha do tipo do elemento é definida no momento de sua criação e os mecanismos para criação de referências aos elementos independem de seu tipo.

O processo utilizado no desenvolvimento da Wiki/RE inspirou-se no *Unified Process*. No que tange à engenharia de requisitos, empregaram-se técnicas para descrição de requisitos não-funcionais (interesses e casos de mau uso) que não constam nos demais processos. Os produtos gerados pelo processo foram documentados em uma *wiki* (uma instância da CoTeia). O uso de uma ferramenta *wiki* demonstrou duas vantagens: a manutenção da CoTeia, realizada durante o desenvolvimento da Wiki/RE, permitiu a compreensão de importantes conceitos de *wikis* e auxiliou na definição da arquitetura da Wiki/RE; o desenvolvimento da documentação em uma *wiki* permitiu a identificação dos problemas existentes na documentação de requisitos em *wikis*.

5.1 Casos de Uso

Os casos de uso básicos da Wiki/RE são `Create page`, `Edit page` e `View page`. Eles são executados durante o processo de engenharia de requisitos, mais precisamente durante a aplicação de uma determinada técnica, seja para o registro e atualização dos resultados para a consulta de dados disponíveis na *wiki*.

O diagrama da figura 1 aprofunda os casos de uso básicos da Wiki/RE, relacionando todos aqueles que contribuem para a documentação de requisitos provenientes de técnicas diversas e de emprego de autoria colaborativa para a escrita dos documentos de requisitos.

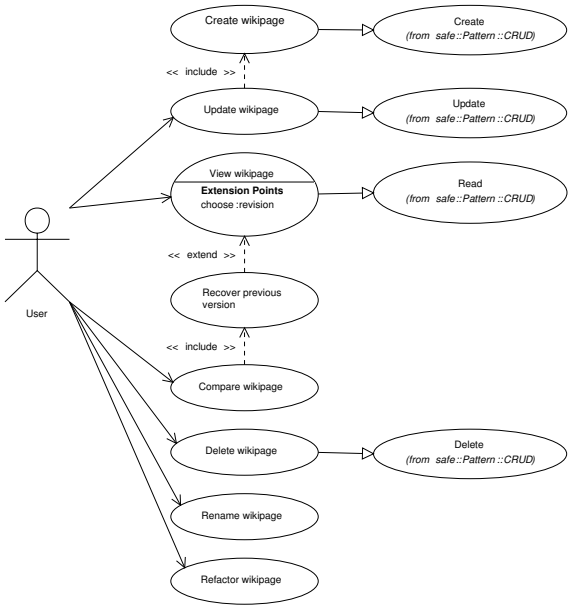

Figura 1: Diagrama de casos de uso da Wiki/RE quanto às suas funções *wiki*.

Os primeiros casos de uso para qualquer hiperdocumento são os responsáveis pela visualização ou pela atualização de uma página *wiki* (ou simplesmente *wikipage*). Os demais casos de uso não podem ser ativados sem que exista uma página definida.

O fato de qualquer desses dois casos de uso ser o inicial decorre do fato de que, em *wikis*, existe o conceito de que, no hiperdocumento, estão presentes todos os nós possíveis (no casa da Wiki/RE, infinitos). A diferença é que alguns possuem um

conteúdo definido, enquanto outros não. O acesso a um nó cujo conteúdo não esteja definido, ao invés de informar que tal nó não existe (como ocorre nas aplicações Web, apresentando-se uma resposta HTTP com código 404), redireciona, automaticamente, para o caso de uso de atualização do conteúdo do nó. Atende-se, com essa estratégia, o princípio incremental das *wikis*.

A Wiki/RE permite a criação de páginas com diferentes tipos de artefatos, possibilitando que os produtos de diversas técnicas de engenharia de requisitos sejam inseridos no hiperdocumento. Antes da definição do conteúdo de um nó, define-se o tipo de dado que nele será armazenado (um caso de uso, uma meta, etc). A referência do hiperdocumento que acionou o caso de uso em questão, o `Update wikipage` não precisa especificar o tipo de nó desejado. Nesse caso, durante a visualização da página, conforme descrito no caso de uso `View wikipage`, o acionamento da referência exibirá todos os nós do hiperdocumento com o nome especificado. Dessa forma, preserva-se o princípio de unificação e precisão.

A mesma concepção que justifica a relação entre os casos de uso de visualização e alteração, aplica-se ao caso de uso de criação de uma página *wiki*, o `Create wikipage`. Segundo o modelo de *wikis*, não existiria necessidade de criar um *wikipage*. No entanto, a aplicação precisa distinguir entre alteração e criação de um novo nó, para efeito de controle de versão.

A Wiki/RE preserva todas as versões de uma página *wiki*. Essas versões são identificadas única e globalmente: uma alteração de uma página não incrementa a versão apenas da página em questão, mas de todo o hiperdocumento. O objetivo é manter uma configuração consistente do hiperdocumento sem grandes custos computacionais. O caso de uso `Recover previous version`, que estende o `View wikipage`, permite a navegação no hiperdocumento, recuperando apenas as páginas com versões compatíveis (iguais ou inferiores) à solicitada pelo usuário.

Em Engenharia de Requisitos, esse recurso é importante para a rastreabilidade. Ele permite identificar, em um artefato que utilize o requisito (a página *wiki*), a identificação da exata versão consultada. Se o requisito for posteriormente alterado, é possível identificar os artefatos que devem ser revisados. Durante a validação do artefato (ou qualquer outra atividade de garantia de qualidade, como criação de casos de teste e revisões), a atividade poderá ser executada com os dados efetivamente utilizados para a criação do artefato.

Em software livre, as iterações rápidas e lançamentos freqüentes de novas versões compelem os desenvolvedores, mesmo aqueles que realizam o desenvolvimento em árvores ou ramos diferentes do principal, a utilizar a versão mais recente dos artefatos sempre que possível[1]. Comparar as alterações feitas a cada versão facilita a identificação de erros e o acompanhamento da evolução dos artefatos. O caso de uso `Compare wikipage` descreve essa funcionalidade, aplicada para páginas *wiki*.

Uma conseqüência da evolução do hiperdocumento é a obsolescência de páginas antigas. A remoção de um página em *wikis* é um tema controverso: páginas que citam a página apagada precisam ser atualizadas e a recuperação do conteúdo da página removida, caso necessário, não é trivial. Por outro lado, existem páginas que ficam "órfãs", ou seja, sem nenhuma ligação a partir de outras páginas do hiperdocumento. Fazendo uma analogia com algumas linguagens de programação (LISP, Java), a página seria uma candidata a ser removida pelo "coletor de lixo". Argumenta-se, no entanto, que alguma página poderia, futuramente, criar uma referência para uma página com o mesmo nome daquela apagada automaticamente e, conseqüentemente, perder-se-ia uma oportunidade de reutilização do conteúdo da mesma.

O interesse em remover páginas na Wiki/RE advém justamente da presença de páginas órfãs. Nenhuma informação do hiperdocumento deveria estar desprovida de rastreabilidade, uma razão para sua existência, em qualquer momento. Se a perda de rastreabilidade foi uma conseqüência natural da evolução do documento, deve-se revisar a página órfã, para ter certeza de que seu conteúdo é desnecessário, e removê-la. Se foi um equívoco, deve-se identificar em que momento a rastreabilidade foi quebrada e restaurá-la devidamente.

A Wiki/RE identifica as páginas removidas com uma ligação específica, que torna possível a recuperação da página removida, caso necessário. Desta forma, páginas que contêm ligações para páginas identificadas como apagadas ainda dispõem de acesso à sua última versão disponível.

Uma outra operação típica durante a evolução de um hiperdocumento de requisitos é a mudança de nome de um artefato. Além de alterar o nome e atualizar as ligações em outras páginas, é preciso garantir que, na ocorrência de uma requisição para uma página que fora renomeada, o usuário seja redirecionado para a página com o novo nome. Esse recurso é necessário para preservar a rastreabilidade das páginas perante referências contidas em docu-

[1] A utilização da versão mais recente facilita a integração do novo código à linha de desenvolvimento principal, o que é um importante marco no desenvolvimento de software livre.

mentos que não são controlados pela Wiki/RE (por exemplo, arquivos de projeto ou casos de teste que são criados a partir dos requisitos).

Durante a engenharia de requisitos, é esperado que ocorra o refinamento dos artefatos e a eventual divisão dos artefatos em outros. A Wiki/RE dispõe de um mecanismo de refatoração, que permite a divisão de uma página em duas páginas diferentes. Uma ligação é, então, especificamente criada, garantindo que, caso ocorra a requisição de uma página que sofreu refatoração, sejam oferecidas as páginas geradas após o processo de divisão, preservando-se assim a rastreabilidade.

5.2 Arquitetura

A arquitetura da Wiki/RE foi elaborada considerando-se a separação em camadas. Assim, a arquitetura consiste, inicialmente, em três camadas: persistência (`Storage`), negócio (`Business`) e apresentação (`Presentation`). Cada camada comunica-se apenas com a sua camada imediatamente vizinha. Essa separação permite uma clara divisão dos interesses da ferramenta, além de ser uma solução bem sucedida para aplicações Web. A camada de persistência é responsável por estender o tempo de vida dos objetos para além do tempo de execução da aplicação, armazenando seus dados em memória permanente. A camada de negócio contém a lógica da aplicação, representando e manipulando o modelo de dados dos requisitos. Enfim, a camada de apresentação permite diferentes visualizações desses dados do modelo de requisitos.

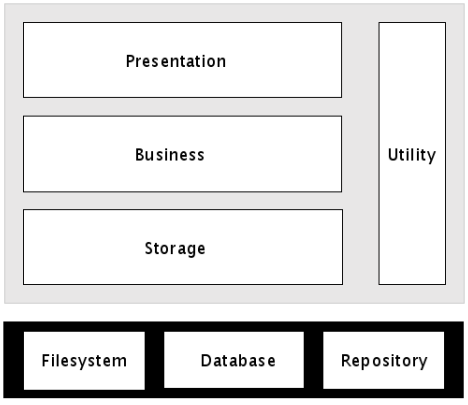

Figura 2: Arquitetura da Wiki/RE

Além das três camadas principais, existem elementos globalmente visíveis, que consistem em classes de uso geral e recorrente (como manipulação de arquivos e estruturas de dados) agrupadas na camada de utilitários (`Utility`). Adicionou-se a restrição de que uma camada não pode comunicar-se com uma camada não vizinha por meio dessa camada utilitária, garantindo-se assim a coesão da arquitetura.

Essas quatro camadas definem o núcleo da arquitetura da Wiki/RE. A última camada, a de persistência, ainda utiliza alguns serviços do sistema, como sistema de arquivos (`Filesystem`), sistemas gerenciadores de bancos de dados (`Database`) e sistemas de controle de versão (`Repository`).

5.2.1 Camada de Persistência

Idéias e conceitos, discutidos e ampliados durante a engenharia de requisitos, devem ser armazenados para posterior utilização no desenvolvimento de software. A camada de persistência mantém todo o histórico dos artefatos gerados pela aplicação. Para isso, utiliza-se o Hibernate, uma implementação, livre e de qualidade, de persistência. Ele utiliza uma base de dados para o armazenamento dos atributos dos objetos, acessando-a via JDBC.

A questão da preservação das diversas edições do documento é sanada com o emprego de sistemas de controle de versão. A Wiki/RE suporta sistemas que operam com arquivos (tal como na maioria das outras *wikis*). Para isso, ela implementa uma camada de controle de versão, abstraindo assim que implementação será efetivamente utilizada. Atualmente, apenas o Subversion é suportado, mas seria possível utilizar qualquer solução que implemente a API proposta.

A Wiki/RE utiliza sistemas que definem arquivos como itens de configuração. Porém, até o presente momento, os dados são armazenados apenas na base de dados. Torna-se mandatório, portanto, criar uma representação em arquivo dos objetos armazenados. Para isso, escolheu-se transformar cada objeto em um documento XML. Esses mesmos documentos serão utilizados, posteriormente, para apresentar uma visão do documento de requisitos.

5.2.2 Camada de Negócio

A camada de negócio é responsável pela manutenção do hiperdocumento de requisitos, oferecendo uma interface de alto nível para manipulação do modelo do documento de requisitos. A comunicação entre a camada de negócio e a camada de apresentação realiza-se segundo o modelo MVC. As interfaces gráficas oferecidas para o usuário representam um estado do modelo de negócio da aplicação: são as visões. O controlador recebe estímulos do usuário, geralmente conseqüências da interação do usuário com a visão oferecida da aplicação, e realiza alterações no modelo e gera novas visões, cumprindo assim um papel

de intermediador entre as visões (usuário) e o modelo de negócio (aplicação). O modelo representa as classes que implementam o modelo de negócios da aplicação.

O controlador, na Wiki/RE, é implementado com o *framework* Struts. A classe do *framework* que exerce a função de controlador, a `RequestProcessor`, foi especializada, criando-se a `WikiRequestProcessor`. Ela registra automaticamente o histórico de navegação, controla a internacionalização dos recursos da *wiki*, gerencia a sessão do usuário, inicializando e concluindo transações, e delega a um controlador de caso de uso o tratamento da requisição.

Os controladores de caso de uso são implementados através de especializações da classe `WikiAction`, que, por sua vez, é uma especialização da classe `DispatchAction` do Struts. Existe uma `WikiAction` para cada recurso suportado pela Wiki/RE e cada uma dessas classes contém um método que representa um caso de uso.

Após a realização de operações pelo controlador, uma nova visão é criada. Para cada caso de uso, existe, pelo menos, uma visão. Definem-se também visões globais, utilizadas pelo controlador `WikiRequestProcessor` para tratamento de erros e autenticação.

5.2.3 Camada de Apresentação

As visões são criadas a partir de estímulos enviados à camada de negócios e processados pelo controlador (padrão MVC). As visões dos diferentes tipos de recursos disponíveis na Wiki/RE possuem liberdade para a apresentação de seus dados, principalmente para operações de alteração. A única operação que é comum a todos os recursos é a apresentação do conteúdo do recurso (caso de uso `View wikipage`).

Todo recurso possui um arquivo XML, que o representa, conforme mencionado na seção 5.2.1. Esse documento é utilizado para a geração do formato de saída desejado pelo usuário, utilizando-se transformações XSLT:

1. Obtêm-se todos os documentos XML do projeto correspondentes à revisão do documento XML alvo da renderização.
2. Aplica-se a transformação XSLT, de acordo com o tipo de saída especificado pelo usuário (HTML, PDF, etc).

Assim, garante-se que, quando apresentado um documento, todos os elementos utilizados são da mesma versão. Neste ponto, a utilização do *SubVersion*, cujo sistema de controle de versão institui versões globais, resume o problema em obter uma cópia do repositório na revisão desejada. No caso do CVS, tal comportamento seria simulado através de marcações (*tags*), uma tarefa custosa se comparada com o SubVersion.

5.3 Projeto

O modelo de hiperdocumento da Wiki/RE é constituído de vários objetos do tipo `Resource`. Um `Resource` é uma abstração para os vários tipos de conteúdos, que podem ser armazenados em uma *Wiki* (páginas de texto simples, figuras, arquivos binários, etc) e são identificados unicamente pelo nome e o tipo concreto. Desse modo, estabelece-se um espaço de nomes global, que reúne todos os nomes de todos os `Resources`, e um espaço de nomes para cada tipo concreto de `Resource`. Para os espaços de nomes de cada tipo concreto, não se admite nomes repetidos. O espaço de nomes global, no entanto, admite-os. O mecanismo de resolução de nomes de uma `Wiki`, conseqüentemente, pode retornar vários recursos (de tipos distintos) quando inquirido por um nome.

O ciclo de vida de um `Resource` inicia-se pela sua criação por um `Registered User`. A esse usuário atribui-se o direito de cópia (*copyright*) do recurso. Os recursos podem ser acessados por qualquer usuário, mas alterados apenas por usuários registrados a fim de evitar as adulterações e os vandalismos nos recursos.

Os `Resources` podem ser associados com outros `Resources` através de ligações (`Links`). As âncoras-origem, quando existem, são definidas nos `Resources`, mas não é possível designar uma âncora-destino. Se um trecho do `Resource` é importante o suficiente para ser citado, o trecho do `Resource` deve ser transformado em um `Resource` distinto.

A Wiki/RE permite a criação de hiperdocumentos de requisitos que são uma representação dos modelos de requisitos que constituem o projeto de software. O diagrama da figura 3 associa os elementos de ambos os modelos, hiperdocumentos e projeto de software, evidenciando a capacidade para a completa documentação dos artefatos de requisitos de um projeto de software.

O objetivo da paridade entre elementos de ambos os modelos é permitir que a documentação dos elementos dos modelos de um software usufrua das mesmas facilidades disponíveis no desenvolvimento de hiperdocumentos *wiki*: geração incremental, unificação e convergência.

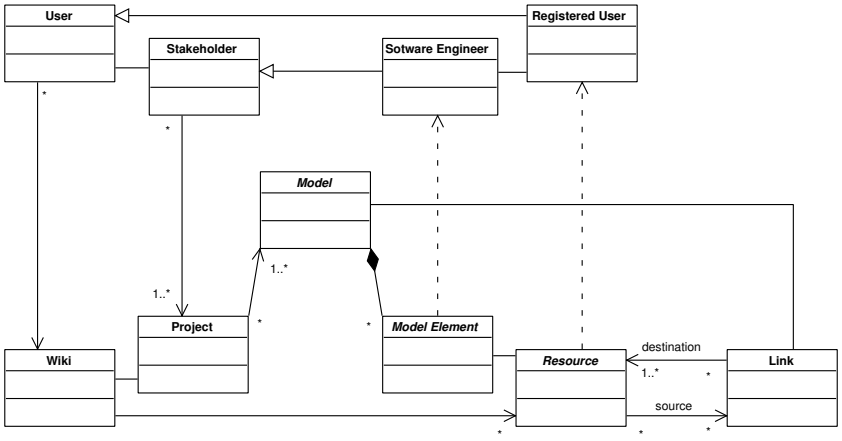

Figura 3: Diagrama conceitual do mapeamento entre hiperdocumentos e projetos de software.

6 Conclusões

Observa-se, em projetos de software livre, a existência de documentação sobre requisitos, porém a mesma se encontra dispersa e fragmentada. São os requisitos e respectivo código-fonte, os emails, os arquivos no repositório, as especificações abertas. Reunir, de maneira ordenada, esses artefatos não é uma tarefa simples. Shneiderman definiu três regras [18] que estabelecem as condições às quais um sistema hipertexto é apropriado para a tarefa de organização de informações: existência de muita informação, informações compostas de pedaços menores de informação e relacionamentos entre os pedaços de informação podem ser definidos e representados.

O volume de informações envolvido no desenvolvimento de software é inegável. Em software livre, todo esse contingente deve ser público, livre para redistribuição. Não existem grandes documentos que reúnam todas as informações: elas são geradas incrementalmente, em um esforço colaborativo em um ambiente distribuído coordenado por ferramentas díspares. Entretanto, todas elas corroboram em um único produto.

A coincidência entre as "três regras de ouro de hipertextos" e a disposição dos artefatos dos projetos de software livre não pode ser desprezada. E não é. A maioria dos projetos possuem *sites* Web, sistemas hipermídias bem simples, que tentam, se não reunir, ao menos indicar a localização das informações desejadas.

A manutenção de tais *sites*, no entanto, não acompanha o ritmo de desenvolvimento do software. Eles são desprovidos de capacidades de edição colaborativa, centralizando a responsabilidade pela atualização a um grupo restrito de pessoas. Para contornar esse problema, diversos projetos de software livre começaram a adotar sistemas *wikis* para realizar essa tarefa.

As *wikis* resolvem satisfatoriamente a questão de atualização das informações, mas deixam outras questões ainda em aberto. Elas são mecanismos genéricos de documentação e os tipos de dados que suportam restringem-se a textos, figuras, tabelas, etc. Não existe a representação de objetos do domínio de negócio: casos de uso, requisitos não-funcionais, documentação de APIs. A representação dos artefatos de engenharia de requisitos como tais permitiria o uso mais eficiente de *wikis* e possibilitaria o tratamento diferenciado das relações entre as informações, como, por exemplo, na forma de rastreabilidade entre os artefatos.

Neste trabalho, aperfeiçoou-se o conceito de *wikis*, relacionando-se o modelo de hipertexto, intrínseco das mesmas, ao modelo de engenharia de requisitos. A Wiki/RE, ferramenta desenvolvida neste trabalho, permite a elaboração de hiperdocumentos de requisitos de acordo com esse modelo, utilizando a estratégia incremental de *wikis* para permitir que a rastreabilidade, tão desejada em documentos de requisitos, seja obtida sem custos adicionais.

Uma parcela significativa da documentação gerada durante o desenvolvimento da Wiki/RE foi criada em um sistema hipermídia tipo *wiki*, a ferramenta CoTeia. A construção desse hiperdocumento evidenciou as capacidades de rastreabilidade provenientes de documentos *wiki*, mas também destacou algumas deficiências na aplicação de *wikis* ao domínio de Engenharia de Requisitos: a perda de rastreabilidade ao dividir uma página *wiki*, a dificuldade imposta para renomear uma página *wiki*, um único espaço de nome (independente do tipo do dado inserido), os

poucos tipos de dados suportados.

As vantagens de rastreabilidade, obtidas pelo uso de *wikis*, seriam perdidas após períodos longos de desenvolvimento caso permanecessem esses problemas. A Wiki/RE elabora soluções que permitem a manutenção da rastreabilidade sem ônus aos seus usuários.

6.1 Trabalhos Futuros

Duas vertentes de trabalhos futuros são facilmente identificáveis. Uma é a evolução da ferramenta Wiki/RE. No presente momento, ela não explora todo o seu potencial para apoiar o processo de engenharia de requisitos. O objetivo, neste ciclo, foi permitir a rastreabilidade dos artefatos gerados pelas diversas técnicas de Engenharia de Requisitos, o que exigiu mudanças importantes no modelo de *wikis*. Aspectos como auxílio na criação de relacionamentos, usabilidade e verificação de atributos de qualidade do documento, não foram explorados.

A segunda vertente de trabalho é o estudo sobre o processo de engenharia de requisitos em softwares livres. São poucas e insuficientes as pesquisas sobre a engenharia de requisitos nos projetos de software livre, abreviando a compreensão sobre seus mecanismos. Essa ignorância traz inconvenientes, econômicos e tecnológicos, tanto para empresas que desejam utilizar softwares livres em seus produtos, quanto para a comunidade que os mantém.

Referências

[1] Peter Aiken. *Advanced Technology for Command and Control Systems Engineering*, chapter Hypermedia-based Requirements Engineering. AFCEA International Press, 1990.

[2] Luca Bompani, Paolo Ciancarini, and Fabio Vitali. Sophisticated hypertext functionalities for software engineering. In *3rd International Workshop on Software Engineering over the Internet*, pages 67–79, Limerick, Ireland, 2000. ACM Press.

[3] Kevin Crowston, Hala Annabi, and James Howison. Defining open source software project success. In *Proc. of International Conference on Information Systems (ICIS 2003)*, 2003.

[4] Ward Cunnigham. Wikiwikiweb. Programa de Computador, March 1995.

[5] Ward Cunningham. Wiki design principles. 2005.

[6] Steve Easterbrook and Bashar Nuseibeh. Using viewpoints for inconsistency management. *IEE Software Engineering Journal*, 11(1):31–43, January 1996.

[7] Gardar Gudgeirsson. Requirements engineering and xml, September 2000. Disponível em http://www.raqoon.is/rqml/rqml-spec.htm. Acesso em 16 de outubro de 2003).

[8] Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram, and Sarah Howard. Helping analysts trace requirements: An objective look. In *International Requirements Engineering Conference*, pages 249–259, Kyoto, Japão, 2004. IEEE Computer Society.

[9] IEEE Computer Society. *Software Engineering Body of Knowledge (SWEBOK)*. Angela Burgess, EUA, 2004.

[10] Institute of Electrical and Electronics Engineers (IEEE). Ieee recommended practice for software requirements specifications (ieee std 830-1998), June 1998.

[11] Hermann Kaindl. Active tool support for requirements engineering through reth. In *International Requirements Engineering Conference*, pages 362 – 363, 2004.

[12] Hermann Kaindl and Stefan Kramer. Semiautomatic generation of dictionary links in hypertext, 1995.

[13] Johan Natt och Dag, Björn Regnell, Vincenzo Gervasi, and Sjaak Brinkkemper. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22(1):32–39, jan 2005.

[14] Vaious Papaioannou and Babis Theodoulidis. Here: Hypermedia environment for requirements engineering. In *7ty Workshop on the Next Generation of CASE Tools (NGCT'96)*, pages 20–21, Heraklion, Crete, May 1996.

[15] Eric Steven Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc, Sebastopol, CA, USA, 1 edition, February 2001.

[16] Christian Robottom Reis. Caracterização de um modelo de processo para projetos de software livre. Master's thesis, Universidade de São Paulo, São Carlos, São Paulo, June 2003.

[17] Walt Scacchi. Understanding the requirements for developing open source software systems. In *IEE Proceedings - Software*, volume 149, pages 24–39, 2002.

[18] B. Shneiderman and G. Kearsley. *Hypertext Hands-On! An introduction to a New Way of Organizing and Accessing Information*. Addison-Wesley, 1989.

[19] Ian Sommerville. *Software Engineering*. International computer science series. Addison-Wesley, New York, USA, 6 edition, 2001.

[20] Amador Durán Toro. Rem (requisite management). Programa de Computador, 2004.

[21] Yutaka Yamauchi, Makoto Yokozawa, Takeshi Shinohara, and Toru Ishida. Collaboration with lean media: How open-source software succeeds. In *Computer Supported Cooperative Work*, pages 329–338, Philadelphia, EUA, December 2000.