5 - Cenários no código livre

Um dos pontos críticos encontrados ao se realizar a evolução da ferramenta C&L, foi sem dúvida a falta de documentação adequada. Não existia na ferramenta anterior uma documentação específica que nos ajudasse a entender o código mais profundamente, toda a documentação disponível era superficial e infelizmente não estava atualizada corretamente, o que veio a dificultar a evolução do software, já que o primeiro mês de trabalho foi utilizado procurando-se entender o código fonte e fazendo engenharia reversa para atualização da documentação.

Esse fato não foi apenas um evento ao acaso, a verdade é que a qualidade da documentação em projetos de software livre como um todo ainda deixa muito a desejar. Essas documentações podem ser divididas em duas categorias, as externas que são apêndices do código como, por exemplo, diagramas, modelos, dicionários de dados e manuais, e as internas que são os comentários que se encontram juntamente com o código fonte do software.

Ambas essas categorias tendem a sofrer um pouco em um projeto de software livre, pois seus participantes o fazem por prazer e muitas vezes a documentação é considerada uma atividade não tão “nobre” quanto a codificação, o que acarreta com que estas sejam deixadas em segundo plano sendo, portanto normal que a documentação se encontre incompleta, desatualizada ou insuficiente.

Uma pesquisa recente realizada em [Robotton03] com vários projetos de software livre, mostra que existe uma preocupação por parte dos projetos com a documentação, mas esta documentação é produzida de maneira informal, concentrada mais no próprio código fonte e menos em documentação externas. Além disso, a documentação externa normalmente visa o usuário final e não o desenvolvedor. Alguns dados significativos da pesquisa são:

- apenas 30% dos projetos pesquisados responderam que produzem e mantém documentação formal para os desenvolvedores;

- apenas 24% dos projetos se utilizam algum tipo de padrão para a codificação do software;

- grande parte dos projetos (78%) possuem algum tipo de documentação para o usuário final;

- apenas 55% destes projetos afirmam que uma parte significativa de sua documentação é atualizada freqüentemente.

Um conseqüência desagradável da falta de cuidado com a documentação de um software livre é o fato de que isto pode limitar a entrada de novos desenvolvedores no projeto em questão. Entendemos como barreira de entrada para um projeto a dificuldade que um potencial usuário da comunidade encontra para se tornar um membro que contribua com código para este projeto.

Para que esse usuário participe no projeto como contribuinte de código, é necessário que as vantagens percebidas por este sejam maiores que a barreira de entrada deste projeto. Como visto anteriormente, em um projeto de software livre essas vantagens não têm um âmbito financeiro (pelo menos não na maioria dos projetos); o que atrai o usuário à participação são o interesse que este possui pelo projeto em si (principalmente se ele for usuário do software), interesse pela tecnologia envolvida, aumento de *status* dentro da comunidade e a possibilidade de ter seu código examinado por especialistas da área.

Vários fatores contribuem para a desestimular a entrada de novos desenvolvedores no projeto, mas aqui vamos nos ater a problemas ligados diretamente ao código, já que é de conhecimento geral que um código bem organizado é mais fácil de se trabalhar. Se o código do projeto não for suficientemente bem documentado ou escrito, aumentará a dificuldade para que sejam feitas evoluções no mesmo, aumentando conseqüentemente a barreira de entrada para qualquer pretendente.

Caso a barreira de entrada seja elevada de tal modo que supere as vantagens esperadas por um usuário, isso acarretará em menos um membro entrando no projeto. Um exemplo de quando a barreira de entrada supera os benefícios esperados, seria quando a expectativa de esforço necessário para se evoluir um determinado código fonte mal documentado e desestruturado, supera a vontade de se aprender uma nova tecnologia por parte de um usuário do software.

Como um projeto de software livre depende da comunidade para poder evoluir, é de vital importância que essa barreira de entrada seja a menor possível, e que sejam perdidos o mínimo possível de potenciais desenvolvedores devido a ela.

A proposta apresentada neste capítulo serve para diminuir essa barreira de entrada, trabalhando diretamente com a documentação do código (documentação interna). Essa proposta consiste de uma nova abordagem para a construção de código com base em cenários, de modo que estes cenários fiquem encapsulados no código fonte.

Esta abordagem foi usada no C&L, e seus resultados serão mostrados no decorrer deste capítulo. Esta abordagem foca no uso de cenários como comentários dentro do código, permitindo rastreabilidade entre requisitos e componentes, além das ligações de dependências entre esses requisitos.

5.1 - Padrões de comentários no código

Antes de apresentar a proposta em si, é necessário fazer um pequeno parêntese para falar de padrões de comentários em uso no mercado, já que este assunto está intimamente ligado com a proposta.

A idéia de padrões de comentários é antiga, a proposta apresentada em [Knuth84] já nos mostrava a importância de incluir assertivas e justificativas de

codificação no código fonte, mas sua proposta apresentava melhores resultados para algoritmos e mostrava algumas deficiências para grandes sistemas.

Hoje em dia, são muito poucos os padrões de comentários aceitos pelo mercado de desenvolvimento de sistemas, e a maioria dos padrões encontrados são bem distintos entre si, pois foram feitos por empresas que os utilizam apenas para projetos pertinentes à mesma e não tem relação com os demais padrões existentes.

Podemos citar como exemplo de um dos poucos padrões aceitos pelo mercado, o JavaDoc, que é usado pela linguagem Java e utiliza marcadores para realizar a padronização e a posterior transformação destes comentários em uma página HTML navegável. No mesmo estilo do JavaDoc, existem outros padrões que utilizam marcadores similares para outras linguagens, como o PHPDoc para a linguagem PHP.

O ideal de uma documentação detalhada é ela poder ser gerada e mantida junto com o código, e por isso a padronização dos comentários é tão importante, já que esta padronização torna possível gerar uma documentação através destes comentários, como é feito com o JavaDoc. Esse fato é ignorado por uma boa parte da comunidade de desenvolvedores, em parte porque se torna necessário que sejam feitas revisões constantes no código para se verificar a consistência com a documentação, e também se faz necessário um controle de versão para a documentação, já que o código deve ser atualizado constantemente.

Além disso, metodologias muito usadas no mercado como o RUP não estabelecem qualquer padrão específico para comentários, da mesma forma que CMM também não o faz, e alguns processos ágeis usados atualmente não encorajam sequer a realização de comentários. Estes fatos colaboram para a pouca popularidade que a padronização de comentários no código tem atualmente no mercado.

Tendo estes fatos em mente, a proposta que será apresentada adiante usa um padrão de comentários bem parecido com o JavaDoc, visando assim facilitar o aprendizado da mesma, e aproveitar a experiência do número de usuários que já fazem uso do JavaDoc e padrões semelhantes.

5.2 – Elementos de um cenário

A definição de cenário foi apresentada na seção 4.2.1, onde foi mostrado o diagrama de entidade-relacionamento da representação escolhida; para prosseguir é necessário um maior aprofundamento neste tópico, descrevendo cada uma das partes que compõe um cenário completo.

Uma grande vantagem de se trabalhar com cenários, é o fato de que estes possuem uma estrutura bem definida que oferece organização e uniformidade na apresentação da informação, facilitando assim a enumeração das funcionalidades, não-funcionalidades e a identificação de cenários relacionados [Silva03].

A estrutura de cenários aqui mostrada é uma estrutura intermediária entre uma representação formal como mostrada em [Hsia94] e uma informal encontrada em [Carroll94]. Esta representação visa descrever uma situação específica do software, através de uma linguagem natural semi-estruturada.

Os elementos que compõe essa estrutura são:

- título - é o identificador do cenário; deve ser sempre único.

- objetivo - é a descrição da finalidade do cenário. Deve-se incluir neste elemento, como esse objetivo é alcançado. Deve-se procurar ser o mais concreto e preciso possível nesta descrição.

- contexto - é a descrição do estado inicial do cenário, deve ser explicitada através de pré-condições, localização geográfica ou temporal.

- recursos - são as entidades passivas trabalhadas no software. Para ser um recurso válido do cenário, este deve aparecer em pelo menos um episódio deste mesmo cenário.

- atores - são entidades envolvidas diretamente com o sistema. Para ser um ator válido do cenário, este deve aparecer em pelo menos um episódio deste mesmo cenário.

- Episódios - são sentenças que correspondem a ações e decisões com participação dos atores e utilização de recursos. Essas sentenças devem aparecer em seqüência e serem sempre o mais simples possível.

- restrições - servem para mostrar aspectos não funcionais que podem estar relacionados a contexto, recursos ou episódios.

- exceções - são situação que põe em risco o objetivo do cenário. O tratamento dado a exceção deve ser descrito neste elemento.

De acordo com [Leite00] uma estrutura válida de cenário deve possuir título, objetivo, contexto, ao menos um recurso e um ator, ao menos dois episódios e 0 ou mais restrições e exceções.

Como podemos ver na estrutura apresentada, as entidades passivas e ativas são representadas respectivamente pelos recursos e atores. O contexto é uma representação da situação em que o cenário ocorre, sendo que sua descrição juntamente com as restrições, episódios e exceções, dão suporte à elaboração de casos de teste.

Abaixo segue um exemplo de um cenário de inclusão de léxicos retirado da ferramenta C&L. Este exemplo possui todos os elementos de uma estrutura de cenário.

Objetivo	Permitir ao usuário a inclusão de uma nova palavra do léxico
Contexto	Usuário deseja incluir uma nova palavra no léxico. **Pré-Condição**: Login, palavra do léxico ainda não cadastrada
Atores	Usuário, Sistema
Recursos	Dados a serem cadastrados
Episódios	O sistema fornecerá para o usuário uma tela com os seguintes campos: - Entrada Léxico - Noção – **Restrição**: Caixa de texto com pelo menos 5 linhas de escrita visíveis - Sinônimos – **Restrição**: Caixa de texto para a entrada de 1 ou mais sinônimos. - Impacto - **Restrição**: Caixa de texto com pelo menos 5 linhas de escrita visíveis - Botão para confirmar a inclusão da nova entrada do léxico **Restrições**: Depois de clicar no botão de confirmação, o sistema verifica os campos nome e noção foram preenchidos.
Exceção	Se esses dois campos não foram preenchidos, retorna para o usuário uma mensagem avisando que estes campos devem ser preenchidos e um botão de voltar para a pagina anterior.

Figura 5.1 - Cenário de inclusão de léxicos retirado do C&L.

5.3 - Cenários inclusos no código

Como observado em [Robotton03], a maioria dos projetos de software livre não possui documentação formal, mas isso não significa que não existe uma preocupação com documentação por parte dos participantes. Apesar da documentação formal não ser muito usada, esta costuma ser substituída por uma mais informal, que se baseia em comentários de código.

Esses comentários não seguem um padrão específico e não existe garantia que expressem os recursos utilizados, as entidades envolvidas, as restrições e as exceções passíveis de ocorrer nesta parte do código. A falta desta informação no código aumenta o esforço para que este seja evoluído adequadamente, já que a pessoa responsável pela evolução terá de fazer um trabalho de engenharia reversa para conseguir obter essas informações e estimar o impacto que as novas alterações irão causar na aplicação como um todo.

A tentativa de se impor qualquer tipo de padrão de documentação mais formal dentro de um projeto de software livre pode esbarrar na resistência da comunidade, pois seus membros não consideram documentar uma atividade tão prazerosa quando codificar, o que pode levar a uma falta de interesse por parte da comunidade em ajudar um projeto que imponha tais normas.

Durante a vida de um projeto deste tipo, um determinado módulo pode ser modificados por dezenas ou mesmo por centenas de pessoas [Godfrey00]. O fato de que cada desenvolvedor colocará seus comentários informalmente dentro do código é preocupante, já que a compreensão deste módulo pode vir a ficar seriamente comprometida, até porque não há garantias que comentários antigos virão a ser alterados.

A proposta de se inserir cenários no código vem ao encontro da necessidade de existir um tipo de documentação mais formal para projetos de software livre, mas sem ferir o modo como a comunidade trabalha com o código. A documentação ainda seria feita dentro do código, mas respeitaria algumas regras de fácil aprendizado.

A utilização de cenários durante o desenvolvimento de software já existe há alguns anos, e possui bastante destaque na comunidade de Engenharia de Software [Weidenhaupt98], mas a idéia de mesclar os cenários com o código fonte é bem recente, e se apóia no fato de que a elaboração de cenários já é conhecida de parte da comunidade de Engenharia de Software.

Usando o modelo de cenários mostrado anteriormente, cada cenário descreverá uma situação específica através de uma linguagem natural semi-estruturada.

A seguir segue um exemplo de um cenário incluído no código fonte de uma página PHP da ferramenta C&L.

```
    <h2>Propriedades do Relatório a ser Gerado:</h2>
<?php

//Cenário - Gerar Relatórios XML

//Objetivo:    Permitir ao administrador gerar relatórios em formato XML de um projeto,
//          identificados por data.
//Contexto:    Gerente deseja gerar um relatório para um dos projetos da qual é administrador
//          Pré-Condição: Login, projeto cadastrado.
//Atores:    Administrador
//Recursos:    Sistema, dados do relatório, dados cadastrados do projeto, banco de dados.
//Episódios:O administrador clica na opção de Gerar Relatório XML.
//          Restrição: Somente o Administrador do projeto pode ter essa função visível.
//          O sistema fornece para o administrador uma tela onde deverá fornecer os dados
//          do relatório para sua posterior identificação, como data e versão.

   $today = getdate();
?>

     Data da Versão:
    <?= $today['mday'];?>/<?= $today['mon'];?>/<?= $today['year'];?>
    <p> <input type="hidden" name="data_dia" size="3" value="<?= $today['mday'];?>">
    <input  type="hidden" name="data_mes" size="3" value="<?= $today['mon'];?>">
    <input type="hidden" name="data_ano" size="6" value="<?= $today['year'];?>">

     </p>
    Versão do XML:  <input type="text" name="versao" size="15">
    <p>Exibir

    Formatado: <input type="checkbox" name="flag" value="ON"><br><br>

    <input type="submit" value="Gerar" onClick="return TestarBranco(this.form);"> </p>

</form>
```

Figura 5.2 - Cenário inserido no código.

Podemos perceber neste cenário os elementos Título, Objetivo, Contexto, Atores, Recursos e Episódios. Com a leitura da descrição do objetivo, é possível ao programador conhecer a funcionalidade que este módulo implementa: gerar um relatório XML do projeto. O contexto mostra em que situação este módulo é iniciado (situação para a geração do relatório), os atores apontam quais entidades estão envolvidas nesta tarefa (neste caso apenas o administrador), os recursos apresentam os dados ou módulos necessários para a realização da tarefa e os episódios fornecem em linguagem natural as ações do algoritmo executado.

Vale ressaltar que os cenários não precisam ficar no começo ou final do código, o ideal é que estes fiquem mesclados com o código, de forma que cada episódio ou exceção apareça perto da parte do código descrita. Abaixo segue um exemplo de cenário que tem esse comportamento.

```
//opener.parent.frames['text'].location.replace('main.php?id_projeto=<?=$_SESSION
['id_projeto_corrente']?>');
//self.close();
location.href = "http://<?php echo $ipValor ?>/cel/aplicacao/add_lexico.php?id_projeto=<?
=$id_projeto?>&sucesso=s";

</script>
<?php
}
/*Episodio 4: Mostrar o formulário para adição de léxico no projeto*/
/*Restrição: O sistema deve verificar se os campos nome e nocao foram preenchidos. */
/*Excecao: Um dos campos nome ou nocao não foi preenchido pelo usuário então o sistema deve pedir
para que o usuário preencha-os. */
else {        // Script chamado atraves do menu superior
    $q = "SELECT nome FROM projeto WHERE id_projeto = $id_projeto";
    $qrr = mysql_query($q) or die("Erro ao executar a query");
    $result = mysql_fetch_array($qrr);
    $nome_projeto = $result['nome'];
?>

<html>
    <head>
        <title>Adicionar Léxico</title>
    </head>
    <body>
```

Figura 5.3 - Partes do cenário mescladas no código.

O episódio 4 deste cenário apresenta a exibição do formulário para a adição de um léxico no C&L, tal formulário se encontrará logo abaixo deste episódio, assim como a restrição exibida também ocorre no pedaço de código logo abaixo da descrição do episódio. Desta forma o programador não apenas saberá o que ocorre no módulo, mas também saberá onde ocorre.

Acreditamos que os cenários possuem várias vantagens quando comparados a comentários não estruturados; algumas destas são:

- a estrutura bem definida do cenário oferece organização e padronização na apresentação da informação contida no código. Este fato é extremamente importante em projetos em que o código é compartilhado entre muitos desenvolvedores diferentes.

- a enumeração das funcionalidades do código podem ser facilitadas pela representação dos episódios de um cenário.

- O conjunto dos cenários do projeto pode vir a servir como um documento explicativo para o usuário.

- Como é usada uma linguagem natural semi-estruturada para descrever os cenários, estes podem ser entendidos com facilidade até mesmo por pessoas não técnicas, como por exemplo, *designers*.

- As restrições e exceções de um cenário podem ajudar a formular casos de teste.

5.4 - O uso do LAL no código

Vale a pena abrir um parêntese neste capítulo para falar do uso do LAL no código, já que o LAL é freqüentemente usado em conjunto com os cenários e serve para empresar a denotação e a conotação de símbolos existentes nos cenários. O uso do LAL deve ser sempre que possível feito em conjunto com os cenários, já que este enriquece ainda mais os cenários, melhorando a compreensão destes por parte dos usuários.

O LAL é uma implementação de conceitos de semiótica num hipergrafo [Leite93], onde tanto a detonação quanto a conotação são expressas para cada símbolo individualmente, nas figuras dos elementos noção e impacto. O LAL obedece ao princípio da circularidade, e com isso cada um de seus elementos, seja de denotação (noção) ou conotação (impacto) pode fazer referência a outros símbolos da linguagem. Também é possível a criação de sinônimos de um termo do LAL, estes sinônimos teriam a mesma noção e impacto do termo.

A descrição dos elementos que compõem um LAL deve ser a mais objetiva possível, fazendo uso de um vocabulário mínimo, ou seja, um sub-conjunto reduzido de palavras com significado bem definido. A seguir temos um exemplo de um elemento do léxico retirado da ferramenta C&L.

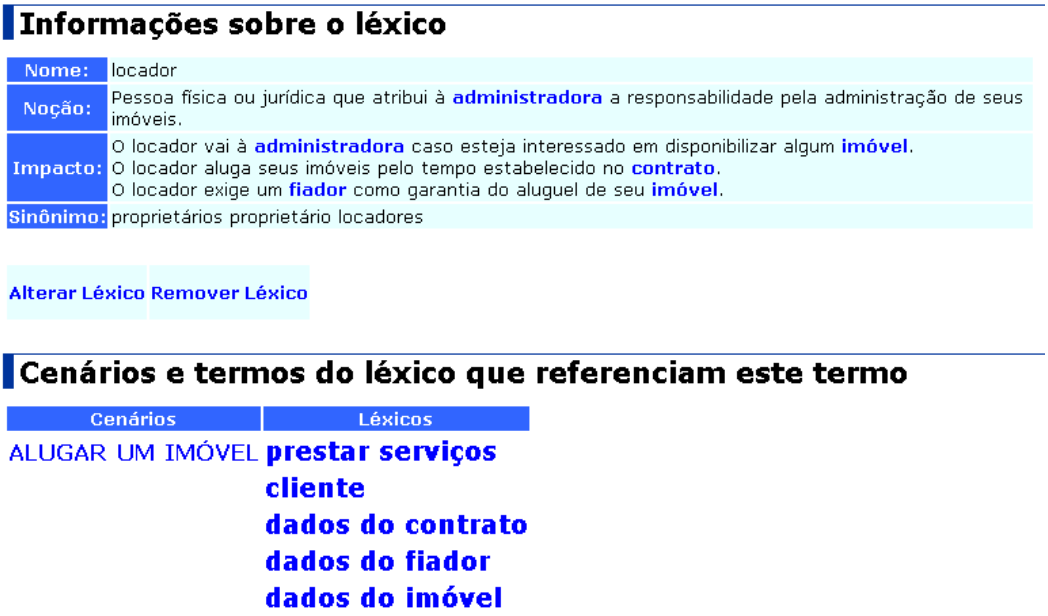


Figura 5.4 - Exemplo de um termo do LAL retirado da ferramenta C&L.

Podemos ver no exemplo que temos diversas ligações nos elementos noção e impacto do LAL, estes atalhos nos levam a outros termos deste mesmo LAL, implementando desta forma o princípio da circularidade. Também é possível notar no exemplo que existem outros termos do LAL referenciando este termo, assim como também existem cenários que fazem referência.

O uso de cenários em conjunto com o LAL é bastante comum, e do mesmo modo que os cenários, o LAL também possui bastante destaque na Engenharia de Software.

Tendo em vista esses fatos, é normal querer usar o LAL no código juntamente com os cenários. Desta forma poderíamos utilizá-lo para uma maior compreensão do cenário e do código em si, e a utilização de um vocabulário controlado facilitaria a implementação de mecanismos que permitissem a identificação de outros cenários que fazem uso dos mesmos termos, assim sendo, seria possível saber com relativa facilidade que partes do código interagem entre si.

Ainda não existe um modelo definido de como estes termos do LAL interagiriam com os cenários localizados dentro do código, mas existe uma idéia que foi implementada recentemente.

Foi desenvolvida uma ferramenta de extração de cenários, ou seja, esta ferramenta extrai os cenários diretamente do código fonte e os coloca em uma documentação pré-formatada própria para cenários feita em HTML. Para buscar o LAL do projeto, uniondo-o posteriormente aos cenários retirados, é necessário que esse LAL do projeto se encontre em um arquivo externo, no caso da ferramenta, esse arquivo é um XML com uma DTD própria para o LAL. Da união dos cenários localizados no código com o arquivo de léxico externos, são criados os arquivo da documentação com os atalhos correspondentes entre os dois. Esta documentação serve então como uma documentação externa deste projeto.

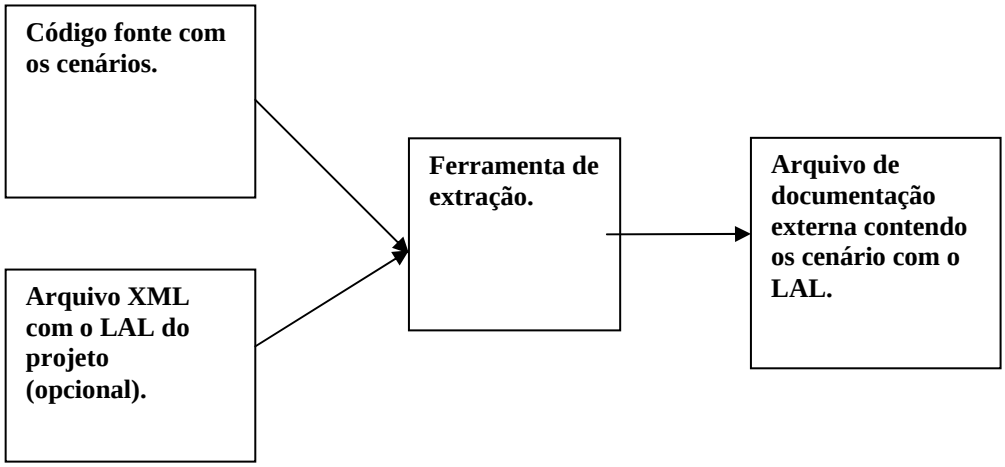


Figura 5.5 – Estrutura da ferramenta para extração de léxicos e cenários.

O capítulo a seguir apresenta e descreve em detalhes como funciona a ferramenta de extração de cenários e léxicos, bem como mostra um exemplo de um arquivos XML do LAL e a DTD que este deve seguir.

5.5 - Ferramenta de extração de cenários e léxicos

Juntamente com esse trabalho, foi desenvolvida uma ferramenta que tem como finalidade transformar os cenários descritos no código em uma documentação externa, e ainda uni-los a um possível léxico do projeto, formando assim uma documentação que pode ser entendida pelos membros do projeto, sem a obrigatoriedade de consultar o código fonte para tanto.

A ferramenta de extração de cenários e léxicos, assim como o C&L é um projeto de software livre, mas apenas agora ela está sendo disponibilizada para a comunidade, portanto até o presente momento, todo o código foi feito pelo autor dessa dissertação. Ela esta implementada em Java, consistindo de aproximadamente 10 módulos distintos. Como único membro do projeto, coube ao autor dessa dissertação a parte de elaboração, desenvolvimento e testes, mas com a disponibilização do código fonte para a comunidade de software livre, espera-se que a ferramenta seja testada, utilizada e evoluída por novos membros, ajudando assim a melhorar sua qualidade.

5.5.1 – Padrão dos cenários

Para que os cenários possam ser extraídos pela ferramenta, primeiramente é necessário que estes obedeçam a uma estrutura de comentário bem definida. Todo bloco de comentários onde se encontre o cenário ou parte dele deve começar com o *token* **/**** e terminar com *token* ****/** . Desta forma a ferramenta sabe exatamente que blocos examinar quando procurar pelos cenários.

Cada linha de um elemento do cenário também possui uma formatação específica, são elas:

Linha de Elemento	Token
Título	@title
Objetivo	@goal
Contexto	@context
Ator	@actor
Recurso	@resource
Exceção	@exception
Episódio	@episode

Tabela 5.1 - Formatação dos elementos de um cenário para que estes possam ser entendidas pela ferramenta de extração.

Usando esta formatação, um cenário ficaria desta forma em um arquivo fonte:

```
<?php

/**
@title Adicionar léxico
@goal Permitir que o usuário adicione um símbolo do léxico num projeto aberto.
@context o usuário acessa a função Adicionar léxico que fica ao lado do adicionar cenário
@context Pre-Requisito: a variável id_projeto é passada através da URL
@context Pre-Requisito: é necessário ter-se usado o adicionar projeto anteriormente
@actor usuário, add_lexico.php
@resource id_projeto, id_projeto_corrente, funcoes_genericas.php, httprequest.inc, index.php, nome, nocao,
@resource impacto, listSinonimo, id_usuario_corrente, sucesso, showSource.php
@episode Iniciar sessão
**/

session_start();

/* vim: set expandtab tabstop=4 shiftwidth=4: */
include("funcoes_genericas.php");
include("httprequest.inc");

/**
@episode CHECAR AUTENTICAÇÃO DO USUARIO - funcoes_genericas.php
**/
chkUser("index.php");        // Checa se o usuario foi autenticado

/**
@episode Conectar o SGBD
**/
$r = bd_connect() or die("Erro ao conectar ao SGBD");

/**
@episode Se o formulário tiver sido submetido então verificar se o nome do símbolo ou algum de seus sinônimos já existe
**/
if (isset($submit)) {

   $ret = checarLexicoExistente($_SESSION['id_projeto_corrente'],$nome);
```

Figura 5.6 - Exemplo de um cenário comentado no formato aceito pela ferramenta de extração.

Como já foi observado, os cenários não devem ficar apenas em um ponto fixo do código, mas sim espalhados por este, por exemplo, os cenários ficarão perto da parte do código fonte que o origina. A ferramenta de extração suporta esse tipo de comentário espalhado, assim como também suporta que um determinado código fonte possua mais de um cenário, mas é importante lembrar que quando isso ocorrer, a ordem dos seus elementos não devem se misturar (um exemplo seria um episódio de um cenário sendo logo seguido por um episódio de outro cenário), já que isso seria um uso errado do conceito de cenário.

De posse destes padrões de criação de cenários, estes devem ser colocados no código manualmente, através do seu editor de código de preferência. A ferramenta C&L pode ser de grande ajuda para editar os cenários e léxicos, mas ela ainda não permite que estes sejam incluídos diretamente no código, portanto caso esteja se usando o C&L para a edição, será necessário copiar os cenários manualmente e colocá-los no código para que a ferramenta de extração possa identificá-los. Pretende-se futuramente fazer com que o C&L possa automatizar esse processo e permitir a edição também do código, estuda-se inclusive a possibilidade de desenvolver um *plugin* do C&L no editor Java Eclipse para facilitar essa edição.

5.5.2 – Ligações entre cenários

O nome do cenário se encontra no *token* @title, portanto para que seja feita uma ligação entre dois cenários, basta que durante a composição do objetivo, contexto, ator, recurso, exceção ou episódio, apareça o nome de algum cenário localizado em outro código fonte, que a ferramenta se encarregará de criar uma ligação entre esses códigos fontes.

No caso da figura 5.6, caso exista um outro cenário com o nome “adicionar projeto”, e outro com o nome “adicionar cenário”, a ferramenta se encarregará de criar uma ligação entre o código fonte contendo esses cenários e o cenário do exemplo. A figura 5.7 demonstra como seria o resultado final.

```
<?php

Titulo: Adicionar léxico
Objetivo: Permitir que o usuário adicione um símbolo do léxico num projeto aberto.
Contexto: o usuário acessa a função Adicionar léxico que fica ao lado do ADICIONAR CENÁRIO
Contexto: Pre-Requisito: a variável id_projeto é passada através da URL
Contexto: Pre-Requisito: é necessário ter-se usado o ADICIONAR PROJETO anteriormente
Ator: usuário, add_lexico.php
Recurso: id_projeto, id_projeto_corrente, funcoes_genericas.php, httprequest.inc, index.php, nome, nocao,
Recurso: impacto, listSinonimo, id_usuario_corrente, sucesso, showSource.php
Episodio: Iniciar sessão

session_start();

/* vim: set expandtab tabstop=4 shiftwidth=4: */
include("funcoes_genericas.php");
include("httprequest.inc");

Episodio: CHECAR AUTENTICAÇÃO DO USUARIO - funcoes_genericas.php
chkUser("index.php"); // Checa se o usuario foi autenticado

Episodio: Conectar o SGBD
$r = bd_connect() or die("Erro ao conectar ao SGBD");

Episodio: Se o formulário tiver sido submetido então verificar se o nome do símbolo ou algum de seus sinônimos já existe
naquele projeto, CHECAR SE LÉXICO JÁ EXISTE - checarLexicoExistente() definido em funcoes_genericas.php.
if (isset($submit)) {
```

Figura 5.7 – Exemplo de um cenário com ligações para outros cenários.

5.5.3 – Ligações entre cenários e léxicos

A ferramenta não exige que exista um LAL para o projeto, mas é altamente recomendável a existência de um no projeto (a estrutura do léxico é descrito em detalhes na seção 4.2.1).

Para tanto é necessário a existência de um arquivo externo XML contendo o LAL. Este arquivo deve se encontrar na raiz do diretório onde se encontram os arquivos de código fontes a serem examinados, chamar-se **lexico.xml** e seguir uma DTD específica. Esta DTD está descrita a seguir:

```
<!ELEMENT project (lexicon)+>
<!ELEMENT lexicon ( (symbol) , (notion) , (behavioral_response) ,
(synonym)* )>
<!ELEMENT symbol              (#PCDATA)>
<!ELEMENT notion              (#PCDATA)>
<!ELEMENT behavioral_response (#PCDATA)>
<!ELEMENT synonym             (#PCDATA)>
```

Esta simples DTD diz que para um LAL ser válido, este deve ter um projeto e pelo menos um léxico. Este léxico por sua vez deve possuir obrigatoriamente um símbolo, e pode vir a ter os elementos noção, impacto e zero ou mais sinônimos.

Observa-se que esta DTD é extremamente simples, visando com que o usuário não tenha problemas para criar um XML contendo o léxico do projeto. No entanto, esta DTD é apenas temporária, pois ela futuramente será a mesma DTD usada pelo C&L, juntando desta forma as duas aplicações (o extrator de cenários e léxicos e o C&L). Como no caso dos cenários, o usuário que estiver usando o C&L para a edição de léxicos, terá de copiá-lo manualmente e colocá-lo no padrão XML entendido pela ferramenta, para facilitar esta tarefa, é aconselhável o uso de um editor de XML de livre escolha. Futuramente o usuário fará o léxico no ambiente C&L e usará a opção já existente no C&L para exportar o resultado para um XML externo que obedecerá a uma DTD comum a ambos.

A única razão para que isto não tenha sido ainda implementado, é o fato de que a ferramenta C&L está sofrendo alterações na parte relativa a geração do arquivo XML e, portanto sua DTD também será alterada. Em breve, quando essas alterações forem terminadas, a ferramenta de extração terá suporte a esta DTD, e ambas as ferramentas serão integradas.

A seguir tem-se um exemplo de um LAL contido em um arquivo XML seguindo esta DTD:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE project SYSTEM "lexico.dtd">
<project>
        <lexicon>
           <symbol>usuário</symbol>
           <notion>Pessoa que utiliza a aplicação</notion>
           <behavioral_response>O usuário utiliza a aplicação para a edição de cenários e léxicos</behavioral_response>
           <synonym>usuario</synonym>
           <synonym>usuarios</synonym>
           <synonym>usuários</synonym>
        </lexicon>
        <lexicon>
           <symbol>cenário</symbol>
           <notion>Descrição em linguagem natural de uma situação que ocorre no macrosistema, com ênfase em comportamentos.
Seções de interação entre o usuário final e o sistema.</notion>
           <behavioral_response>Simula tarefas reais, facilitando o entendimento do sistema.
Auxilia na identificação de pontos de vista conflitantes ou divergentes.</behavioral_response>
           <synonym>cenario</synonym>
           <synonym>cenários</synonym>
           <synonym>cenarios</synonym>
        </lexicon>
        <lexicon>
           <symbol>léxico</symbol>
           <notion>Hiperdocumento que descreve os símbolos de uma determinada linguagem;
no contexto da Engenharia de Requisitos, descreve palavras ou frases peculiares ao meio social da aplicação sob estudo.
Também utilizado na modelagem de requisitos.</notion>
           <behavioral_response>Deve ser auto-contido (vínculos ou links apenas para o próprio documento).
Utilizado para facilitar a comunicação e a compreensão de termos do Universo de Informações entre agentes do processo
de requisitos pelo usuário.</behavioral_response>
           <synonym>léxicos</synonym>
           <synonym>lal</synonym>
        </lexicon>
</project>
```

Figura 5.8 - Exemplo de um léxico em um arquivo XML, obedecendo a DTD mostrada.

De posse deste LAL, os cenários extraídos não mais conterão apenas ligações para outros cenário, mas também terão ligações para o termo correspondente do léxico, caso o nome deste termo ou algum de seus sinônimos apareça em algum momento nos elementos objetivo, contexto, ator, recurso, exceção ou episódio. Essas ligações levarão a um arquivo HTML contendo todos os termos do LAL com seus impactos, noções e sinônimos correspondentes.

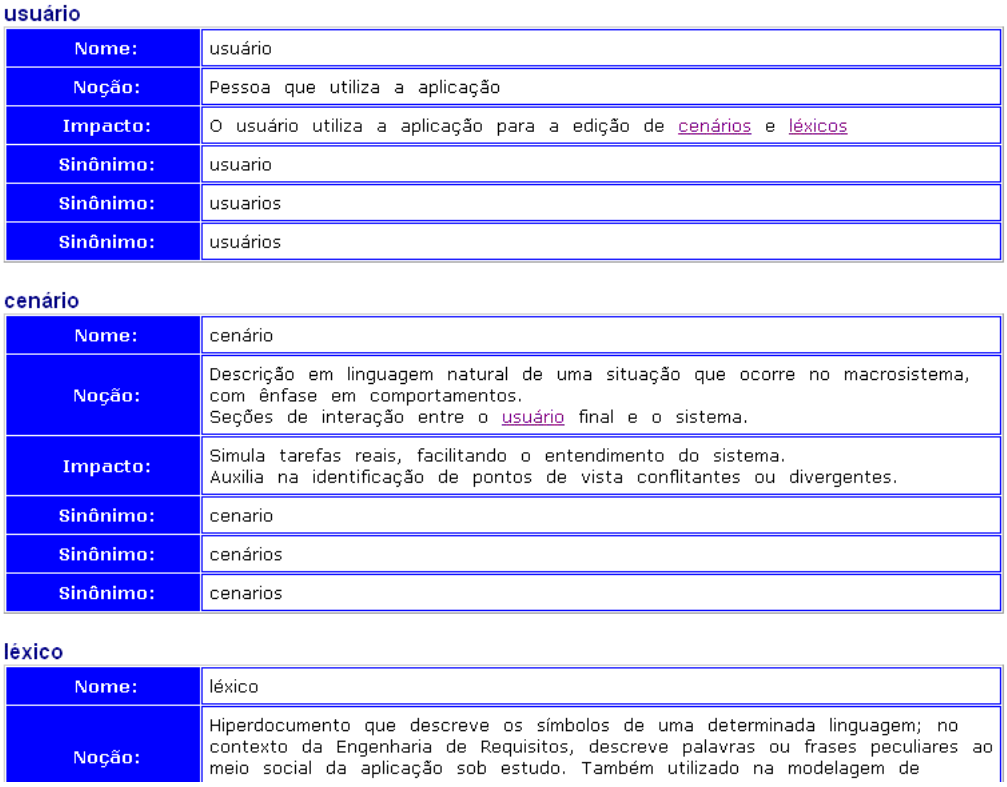
Deve-se lembrar que esse arquivo HTML contendo os temos do LAL, também terão ligações entre seus termos, implementando desta forma o princípio

da circularidade. A seguir segue um exemplo de como as ligações com o LAL aparecem no cenário mostrado anteriormente:

```
<?php

Titulo: Adicionar léxico
Objetivo: Permitir que o usuário adicione um símbolo do léxico num projeto aberto.
Contexto: o usuário acessa a função Adicionar léxico que fica ao lado do ADICIONAR CENÁRIO
Contexto: Pre-Requisito: a variável id_projeto é passada através da URL
Contexto: Pre-Requisito: é necessário ter-se usado o ADICIONAR PROJETO anteriormente
Ator: usuário, add_lexico.php
Recurso: id_projeto, id_projeto_corrente, funcoes_genericas.php, httprequest.inc, index.php, nome, nocao,
Recurso: impacto, listSinonimo, id_usuario_corrente, sucesso, showSource.php
Episodio: Iniciar sessão

session_start();

/* vim: set expandtab tabstop=4 shiftwidth=4: */
include("funcoes_genericas.php");
include("httprequest.inc");

Episodio: CHECAR AUTENTICAÇÃO DO usuario - funcoes_genericas.php
chkUser("index.php"); // Checa se o usuario foi autenticado

Episodio: Conectar o SGBD
$r = bd_connect() or die("Erro ao conectar ao SGBD");

Episodio: Se o formulário tiver sido submetido então verificar se o nome do símbolo ou algum de seus sinônimos já existe
naquele projeto, CHECAR SE LÉXICO JÁ EXISTE - checarLexicoExistente() definido em funcoes_genericas.php.
if (isset($submit)) {
```

Figura 5.9 - Exemplo de um cenário com ligações para outros cenários e também para o LAL.

Pode-se observar que todos os termos apresentados no exemplo de LAL da figura 5.8 estão marcados como ligações. Ao se clicar nestas ligações, o usuário do projeto será levado para o arquivo HTML contendo o LAL, no exato ponto onde este se encontra. A seguir será mostrado outro exemplo em que esse arquivo será mostrado, assim como suas ligações que também ocorrem no próprio LAL:



usuário

Nome:	usuário
Noção:	Pessoa que utiliza a aplicação
Impacto:	O usuário utiliza a aplicação para a edição de cenários e léxicos
Sinônimo:	usuario
Sinônimo:	usuarios
Sinônimo:	usuários

cenário

Nome:	cenário
Noção:	Descrição em linguagem natural de uma situação que ocorre no macrosistema, com ênfase em comportamentos. Seções de interação entre o usuário final e o sistema.
Impacto:	Simula tarefas reais, facilitando o entendimento do sistema. Auxilia na identificação de pontos de vista conflitantes ou divergentes.
Sinônimo:	cenario
Sinônimo:	cenários
Sinônimo:	cenarios

léxico

Nome:	léxico
Noção:	Hiperdocumento que descreve os símbolos de uma determinada linguagem; no contexto da Engenharia de Requisitos, descreve palavras ou frases peculiares ao meio social da aplicação sob estudo. Também utilizado na modelagem de

Figura 5.10 - Exemplo do arquivo HTML com o LAL do projeto. Nota-se que este arquivo também possui ligações, implementando assim o princípio da circularidade.

5.5.4 - Usando a ferramenta de extração

A utilização da ferramenta é bem simples, tudo o que se deve fazer é rodar a aplicação, dando um duplo clique no arquivo celparser.jar. Como este é um arquivo Java, é necessário que exista uma Máquina Virtual Java instalada na máquina. Caso não exista, basta baixá-la gratuitamente em [JAVA03].

Após isso, basta clicar em **Menu** e posteriormente **Selecionar Diretório,** selecionando em seguida a pasta raiz que contém os arquivos fontes que serão

examinados pela ferramenta. Após isso, a ferramenta se encarregará de criar um arquivo HTML próprio para cada código fonte, outro para o léxico (caso exista), e uma estrutura de frames para facilitar a navegação entre os arquivos gerados. As figura a seguir demonstram esse simples processo.

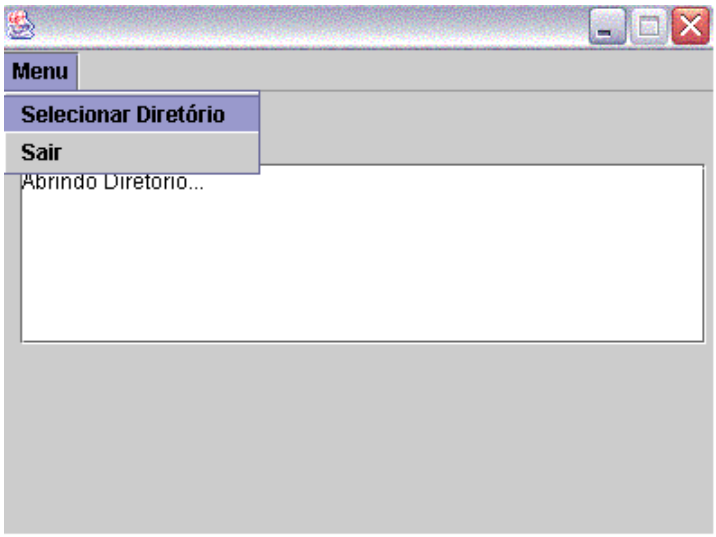


Figura 5.11 - Demonstração do uso da ferramenta.

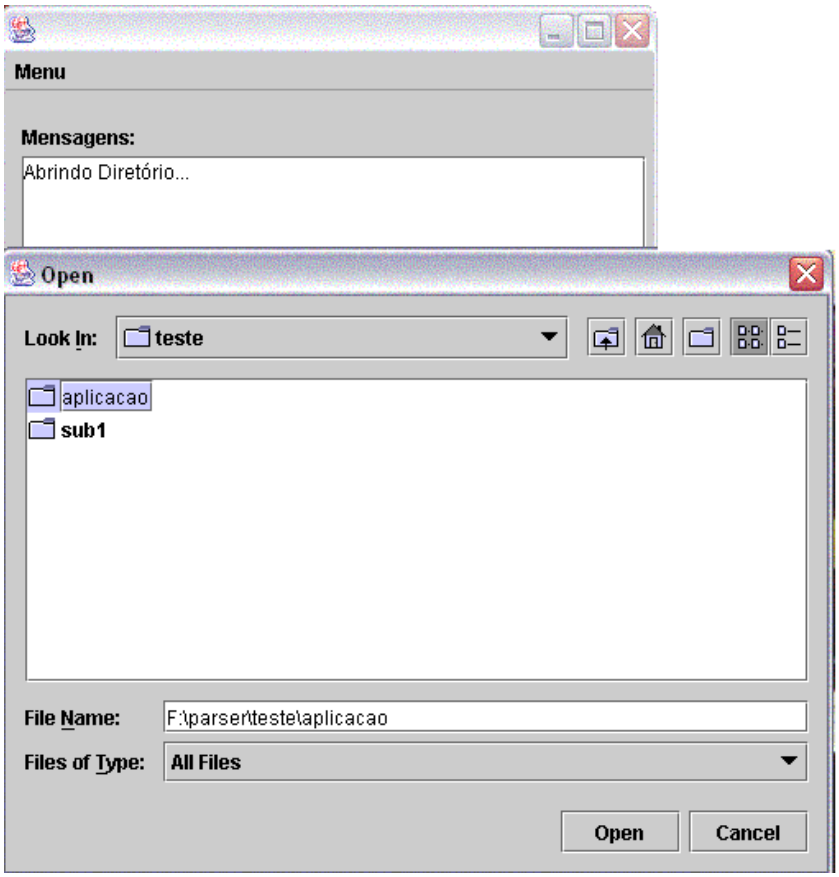


Figura 5.12 – Escolha do diretório raiz do código fonte do projeto.

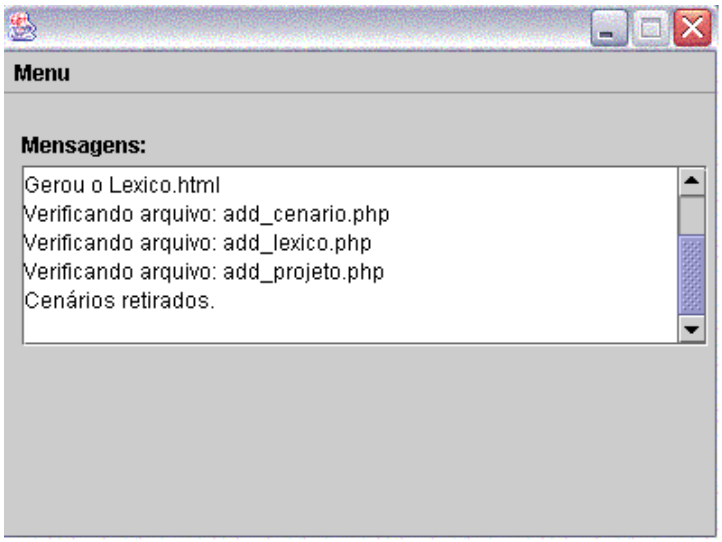


Figura 5.13 - A ferramenta retirando os cenários do código fonte.

Após esse processo, os cenários serão retirados de todos os arquivos de código fonte do diretório selecionado e seus subdiretórios. Caso seja encontrado o arquivo **lexico.xml**, na raiz do diretório escolhido, e este esteja de acordo com a DTD apresentada anteriormente, também será gerado o LAL do projeto.

Para visualizar o resultado da extração, basta clicar no arquivo **index.html** que aparecerá na raiz do diretório escolhido pelo usuário para realizar a extração. Este arquivo exibirá dois frames, o frame da esquerda contém todos os nomes dos arquivos fontes pesquisados pela ferramenta, e o frame central mostrará o resultado final da extração do cenários e léxico em cada um deles, bastando para tanto, selecionar o arquivo apropriado no frame esquerdo.

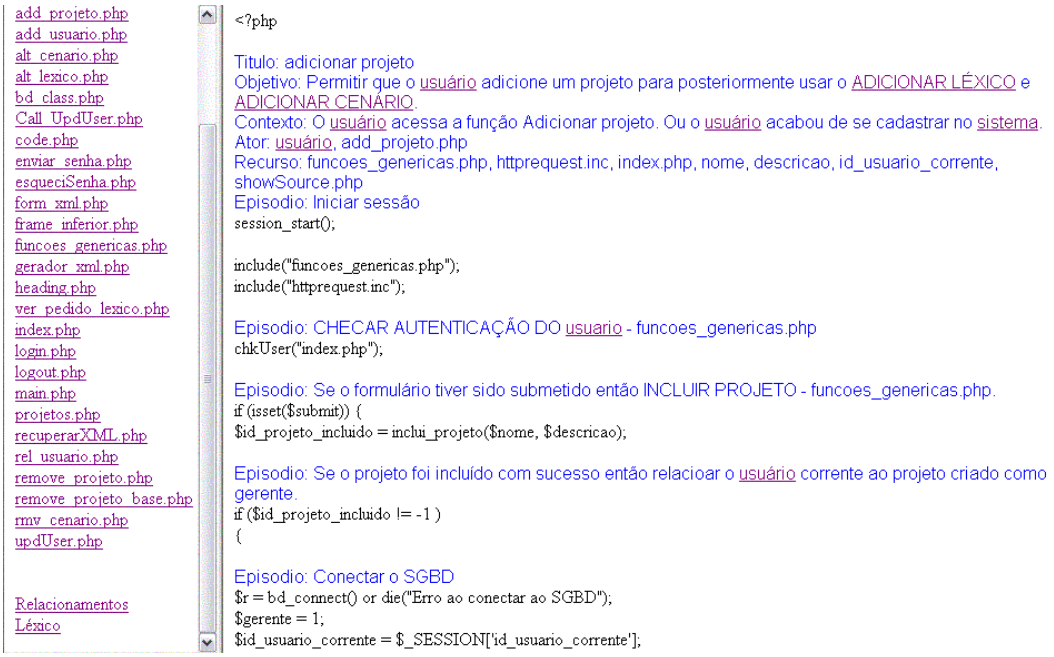
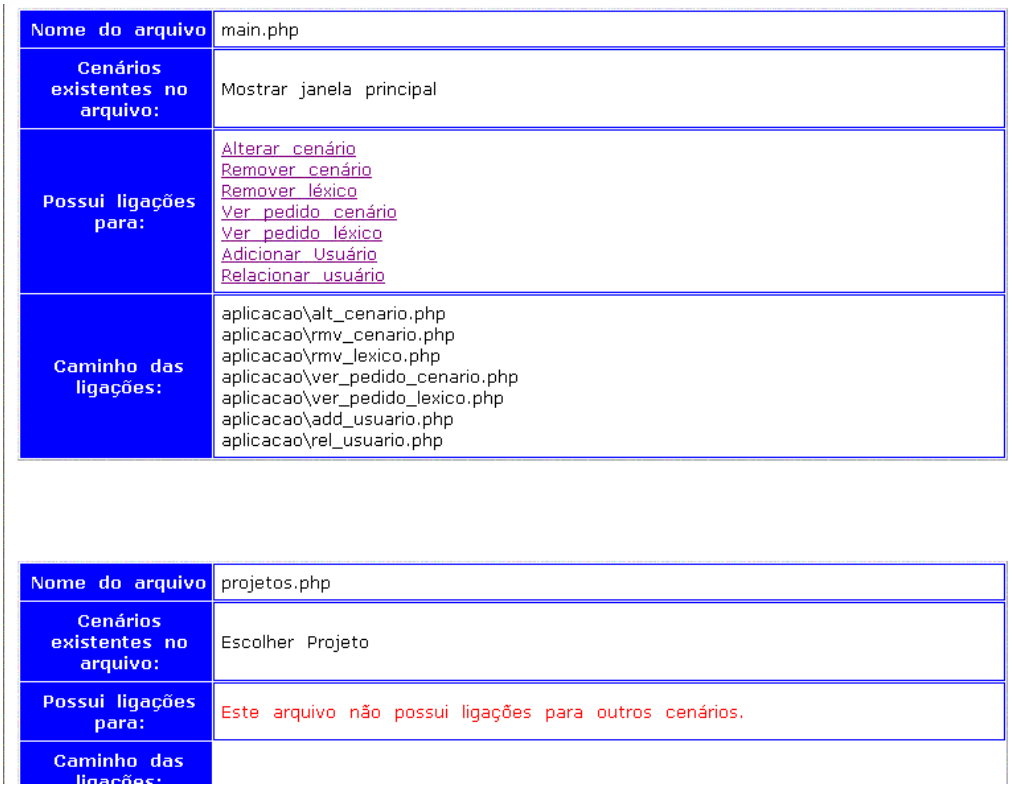


Figura 5.14 - Frames de exibição da ferramenta de extração.

5.5.5 - Relacionamentos entre cenários

Como podemos observar na figura 5.14, existem dois atalhos extras no frame de seleção, logo após a listagem dos arquivos do projeto. O segundo destes é autoexplicativo, sendo um atalho para o arquivo HTML contendo o LAL como é mostrado na figura 5.10, já o primeiro é um atalho que nos leva a um HTML contendo os relacionamentos entre os cenários do código.

O objetivo desta página é listar todos os arquivos do projeto, juntamente com algumas informações em relação a seus cenários e ligações. A figura abaixo mostra um exemplo desta página.



Nome do arquivo	main.php
Cenários existentes no arquivo:	Mostrar janela principal
Possui ligações para:	Alterar cenário Remover cenário Remover léxico Ver pedido cenário Ver pedido léxico Adicionar Usuário Relacionar usuário
Caminho das ligações:	aplicacao\alt_cenario.php aplicacao\rmv_cenario.php aplicacao\rmv_lexico.php aplicacao\ver_pedido_cenario.php aplicacao\ver_pedido_lexico.php aplicacao\add_usuario.php aplicacao\rel_usuario.php

Nome do arquivo	projetos.php
Cenários existentes no arquivo:	Escolher Projeto
Possui ligações para:	Este arquivo não possui ligações para outros cenários.
Caminho das ligações:	

Figura 5.15 - Relacionamentos entre arquivos e cenários

Como podemos observar no exemplo da figura 5.14, esta página nos mostra para cada arquivo do projeto, seu nome, quais cenários existem dentro dele, quais cenários este arquivo faz referência (existe um atalho para ele em algum lugar do código), e os caminhos relativos dentro do projeto para estes arquivos referenciados.

Estas informações são úteis para o desenvolvedor, na medida que será possível saber com mais precisão, onde mudanças em um determinado arquivo

fonte vão impactar no projeto como um todo. Apesar disso, espera-se que em uma versão futura da ferramenta, existam mais informações e que estas venham em forma gráfica ao invés da puramente textual.

5.5.6 - Utilidade da ferramenta

A utilidade desta ferramenta vem de encontro com uma das maiores necessidades de um projeto de software livre, que é a documentação externa voltada para os desenvolvedores do software.

Como foi mostrado no início deste capítulo, uma pesquisa recente com diversos projetos de software livres mostrou que bem menos da metade de todos os projetos pesquisados (30%) mantém algum tipo de documentação formal para os desenvolvedores, e mesmo assim, nem sempre essas documentações são atualizadas freqüentemente. A maior parte da documentação encontrada pelos desenvolvedores, é interna e se encontra na forma de comentários de código. Esta forma de se desenvolver um software é bem característica do desenvolvimento de software livre, e vem do fato de que o desenvolvedor quer gastar seu tempo livre programando e não trabalhando na documentação.

O uso desta ferramenta, juntamente com os cenários no código, são uma forma de não ir contra essa natureza do desenvolvedor de software livre, e ainda ajudá-lo na confecção da documentação externa visando outros desenvolvedores participantes do projeto. Desta forma, a barreira de entrada para que outros desenvolvedores possam entrar no projeto irá diminuir, já que estes terão a sua disposição uma documentação externa que irá auxiliá-los na hora de desenvolver e submeter seu próprio código, beneficiando a si mesmos e ao projeto.

5.6 - Dificuldades da proposta

Existem algumas dificuldades na proposta de inserção de cenários no código que devem ser abordadas.

A primeira e mais imediata é a aceitação destas normas por parte da comunidade de software livre. Fazer com que uma comunidade que está muito mais preocupada com a implementação, despender esforço com a criação dos cenários pode parecer problemático a princípio, mas acreditamos que o aumento de conhecimento do domínio de informação e a diminuição da barreira de entrada do projeto irão compensar o esforço extra. Para ajudar na tarefa de edição, a ferramenta C&L oferece facilidades para a edição de léxicos e cenários, bem como uma ajuda teórica sobre o assunto.

O segundo problema é estrutural, já que nem sempre um cenário está implementado em um módulo, assim como um módulo pode implementar mais de um cenário. Este item necessita de um estudo maior, já que em certos casos é comum se ter mais de um cenário por módulo, como por exemplo, um módulo de funções que seja usado como biblioteca, mas no caso de um cenário estar implementado em mais de um módulo, nossa experiência indica que pode ser decorrente de uma má decomposição do sistema. Assim, o uso de cenários pode ajudar o programador a perceber que algum módulo não obedece às propriedades de coesão e acoplamento.

O terceiro problema tem relação com o fato de que nada garante que um desenvolvedor ao evoluir o código, também evolua os cenários do mesmo. Para resolver essa situação, basta que um pré-requisito para aceitação da submissão seja a coerência entre cenários e códigos.

5.7 - Trabalhos relacionados

Existem diversas propostas para a documentação de código fonte, várias delas baseadas no trabalho exposto por Knuth em [Knuth84]. Neste trabalho, Knuth propõe regras para comentários estruturados de forma que sejam inseridos no código fonte e posteriormente tratados por um pré-processador de sua autoria. Knuth já enxergava a importância dos comentários estruturados e a possibilidade de através deles montar uma rede de ligações entre os módulos.

Comentários estruturados vêm se tornando uma tendência também para as novas linguagens de programação. A linguagem Java faz uso da documentação JavaDoc, esta documentação é gerada através de comentários estruturados dentro do código. A linguagem é acompanhada de uma ferramenta que se encarrega de extrair estes comentários do código e colocá-los de forma estruturada dentro de uma documentação em formato HTML, que possa ser consultada pelo usuário. O aplicativo não somente extrai os comentários, mas também identifica os seus componentes (classes, interfaces, atributos, métodos, exceções), colocando desta forma toda a estrutura do código fonte no documento gerado.

Existem uma forma semelhante de documentação para PHP, chamada PHPDoc, que é uma adaptação do JavaDoc para a linguagem PHP.

A proposta aqui apresentada é fundamentada no trabalho de Knuth, além de usar algumas das idéias apresentadas em documentações como o JavaDoc, principalmente na maneira como os cenários estão estruturados no código, e sua extração para um documento externo. Da mesma forma que Knuth usava sua estrutura de comentários para fazer um mapeamento entre os módulos, usaremos o cenário para fazer a rastreabilidade entre os requisitos e os módulos. A ferramenta de extração de cenário no código também teve forte influência no trabalho de Knuth e pretendemos futuramente que a ferramenta monte juntamente com sua documentação, uma estrutura gráfica do cenário e sua relação com os demais, facilitando assim a visualização destas relações para os membros do projeto.

5.8 - Experiência no C&L

Ao se iniciar o processo de evolução de software que deu origem ao C&L, uma das principais dificuldades foi a compreensão do código, já que mesmo quando havia comentários, não se fazia menção à interação entre os módulos. Devido a dificuldade inicial, gasto-se muito tempo fazendo-se a engenharia reversa do sistema. Características como controle de sessão, que se propagam por

diversos módulos, são críticas e difíceis de se perceber caso não haja algum comentário explícito no código.

Os comentários também eram vagos com relação aos recursos usados, restrições existentes e exceções geradas. Estes fatos acabavam por afetar o desempenho da equipe, e precisaram ser corrigidos para a continuação do projeto.

A opção encontrada pela equipe para resolver esse problema foi a substituição destes comentários não estruturados por cenários inclusos no código. As normas usadas para a estruturação dos cenários foram as apresentadas anteriormente neste capítulo, sendo que os exemplos aqui mostrados foram todos retirados da ferramenta C&L.

Além da inclusão dos cenários no código, foram feitos dois mapas externos ao código. Um destes mostrava o relacionamento entre cenários baseado em [Breitman00] e outro apresentava relacionamentos entre módulos.

O mapa de relacionamento entre módulos fornece uma visão geral de como os módulos estão estruturados e suas relações, o segundo mapa tratava os cenários e suas relações, facilitando assim o trabalho no processo de evolução do software quando novas funcionalidades são incluídas ou quando erros são corrigidos [Breitman00]. Planeja-se que o mapa de relacionamento de cenários seja gerado automaticamente pela ferramenta de extração de cenários no futuro.

Essa estruturação da documentação interna e externa foi de grande importância para o processo evolutivo da ferramenta, já que espera-se que esta facilite a compreensão do software para processos de evolução futuros.

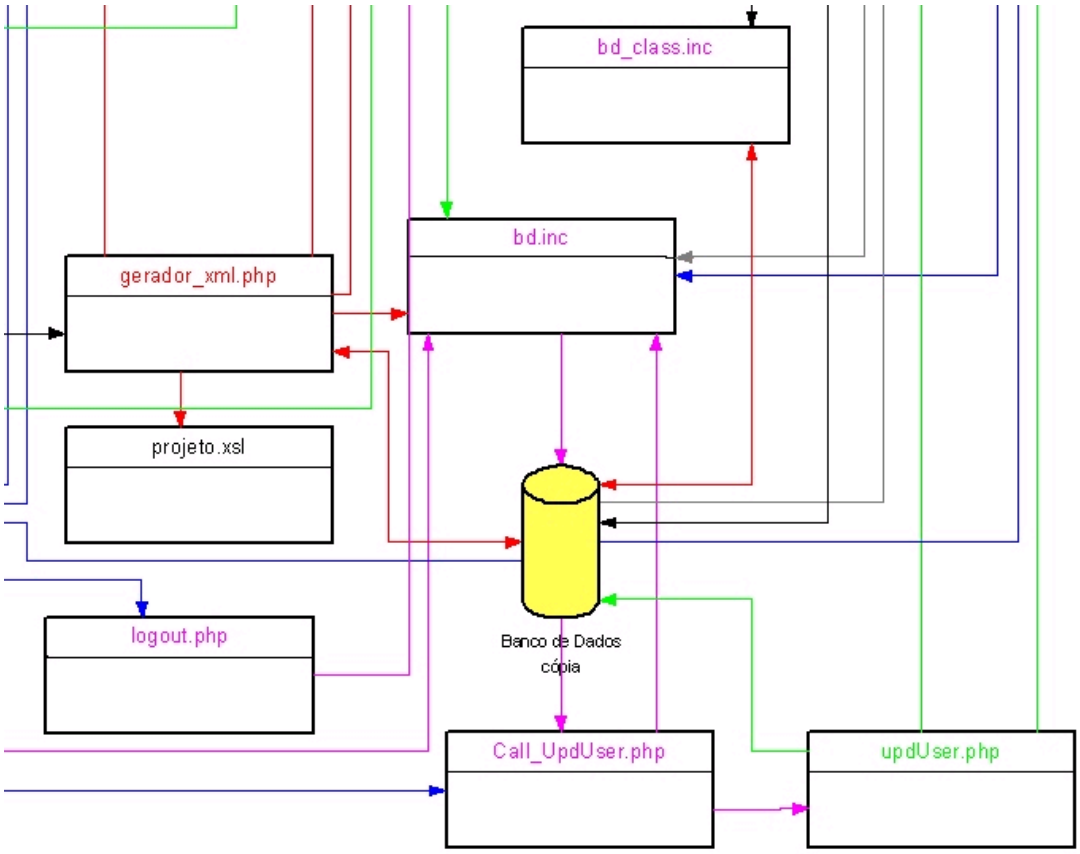


Figura 5.16 - Parte do mapa de relacionamento entre módulos do C&L.

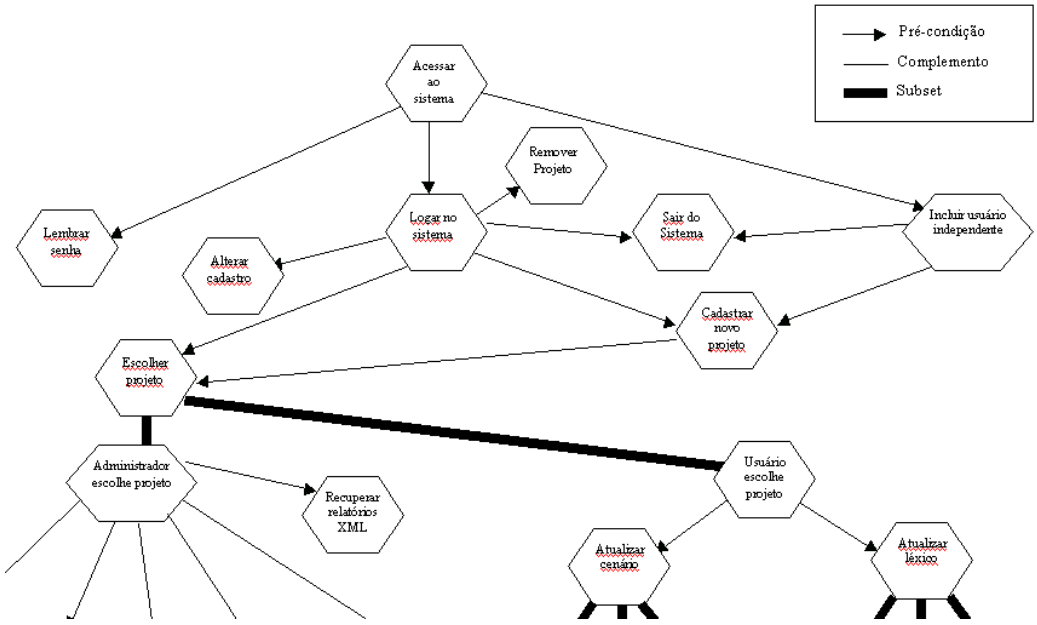


Figura 5.17 – Parte do mapa de relacionamento entre cenários.

5.9 - Trabalhos futuros

A ferramenta C&L ajuda na edição de cenários e léxicos, e a ferramenta de extração retira cenários do código fonte de um projeto, e os coloca em um documento externo HTML juntamente com o LAL do projeto, mas até o momento não foi feita a integração entre ambas as ferramentas. Quando as ferramentas estiverem integradas, será possível gerar todo o léxico do projeto na ferramenta de edição (C&L) e gerar instantaneamente o XML correspondente que será entendido e usado pela ferramenta de extração. Esta integração entre as ferramentas está sendo trabalhada no momento.

Pretende-se também que futuramente o C&L tenha capacidade de edição de código, desta forma seria possível incluir os cenários editados pelo C&L diretamente no código. Estuda-se inclusive a possibilidade de desenvolver um *plugin* do C&L no editor Java Eclipse para facilitar essa edição de código e a inclusão dos cenários.

Outro trabalho futuro que está sendo desenvolvido é a geração de diagramas com os relacionamentos entre os cenários existentes no projeto, como o mostrado na figura 5.17. Espera-se que futuramente a ferramenta de extração implemente estes diagramas, ao invés de apenas uma versão textual como é feito hoje em dia, já que estes diagramas seriam importantes adições para a documentação externa de um projeto de software livre. Uma outra linha que precisa ser mais bem pesquisada é a geração de casos de teste com base em cenários.