Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Kinder Almentero

Re-engenharia do software C&L para plataforma Lua-Kepler utilizando princípios de transparência

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Julio Cesar Sampaio do Prado Leite

Rio de Janeiro, 08 abril de 2009

Eduardo Kinder Almentero

Re-engenharia do software C&L para plataforma Lua-Kepler utilizando princípios de transparência

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Julio Cesar Sampaio do Prado Leite
Orientador
Departamento de Informática – PUC-Rio

Prof. Carlos José Pereira de Lucena
Departamento de Informática – PUC-Rio

Prof. Roberto Ierusalimschy
Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 08 de abril de 2009

Eduardo Kinder Almentero

Graduou-se em Bacharelado em Informática na Universidade do Estado do Rio de Janeiro (UERJ) em 2007. Desde 2007 participa do grupo de pesquisa de engenharia de requisitos da PUC-Rio, coordenado pelo professor Julio Leite, e trabalha no Laboratório de Engenharia de Software (LES).

Ficha Catalográfica

Almentero, Eduardo Kinder

Re-engenharia do software C&L para plataforma Lua-Kepler utilizando princípios de transparência / Eduardo Kinder Almentero ; orientador: Julio Cesar Sampaio do Prado Leite. – 2009.
112 f. ; 30 cm

Dissertação (Mestrado em Informática)–Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.
Inclui bibliografia.

1. Informática – Teses. 2. Engenharia de Requisitos. 3. Transparência de Software. 4. Cenário. 5. Léxico Ampliado da Linguagem. I. Leite, Julio Cesar Sampaio do Prado. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Titulo.

CDD: 004

Agradecimentos

A meus pais, Emilio e Meire, pelo incentivo, motivação e carinho. Sem vocês nada disto seria possível.

Ao professor Julio Leite pela enorme paciência, dedicação, confiança e pelo conhecimento compartilhado ao longo destes anos.

Aos integrantes do Grupo de Requisitos da PUC-Rio, Antonio de Padua, Maurício, Claudia, Fillipe, Herbet, Vera e Elizabeth, pelas suas valiosíssimas opiniões e sugestões sempre construtivas.

Aos amigos, Andrew, Evelin, Camila, Isela, Baldoino, Elder e Manoel, pela ajuda indispensável, pela motivação e pelo companheirismo durante toda essa jornada.

Resumo

Almentero, Eduardo Kinder; Leite, Julio Cesar Sampaio do Prado. **Re-engenharia do software C&L para plataforma Lua-Kepler utilizando princípios de transparência.** Pontifícia Universidade Católica do Rio de Janeiro, 2009. 112p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A transparência é um termo chave que está presente em diversos contextos como o econômico e político, e atualmente um dos novos contextos em que se apresenta é a transparência de software. Software livre é um bom exemplo de transparência de software, onde a grande vantagem é que podemos acessar o código fonte e então escolher entre suas características as que desejamos, porém esta possibilidade está direcionada somente aqueles que entendem seu código fonte. Entender o código fonte de um software poder ser uma tarefa árdua, especialmente se nenhuma técnica foi utilizada para facilitar sua leitura. Neste trabalho exploramos um método de desenvolvimento para software livre baseado no uso de cenários. O resultado da aplicação deste método será um documento único, o código fonte, onde teremos os cenários integrados com o código, facilitando sua leitura e entendimento, trazendo assim mais transparência para o software. Este método foi refinado durante sua aplicação na re-engenharia do software C&L. Para produzir uma documentação complementar aos cenários inclusos no código fonte, utilizamos a técnica LAL (Léxico Ampliado da Linguagem) para mapear o espaço de nomes do novo software C&L.

Palavras-chave

Engenharia de requisitos, Transparência de software, Cenário, Léxico Ampliado da Linguagem.

Abstract

Almentero, Eduardo Kinder; Leite, Julio Cesar Sampaio do Prado (Advisor). **Reengineering the C&L software towards Lua-Kepler platform using principles of transparency.** Pontifícia Universidade Católica do Rio de Janeiro, 2009. 112p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Transparency is a keyword present in different contexts such as the economic and the political ones, and, currently, one of the new contexts, in which it stands, is software. Free (open source) software is a good example of transparency, where the great advantage is that one can access the source code and then choose the characteristics he/she wants, but, in this way, we will be serving only those who understand the source code. Understanding software source code can be an arduous task, especially if no technique has been used for facilitate reading. In this work we explore a method for developing free software based on the use of scenarios. The result of applying this method is a single document, the source code, in which the scenarios will be integrated within the code, making it easier to read and understand, thus bringing more transparency to the software. This method was refined during its application to reengineer the “C&L” software. In order to produce additional documentation, besides the scenarios embedded in the code, we used the LEL (Language Extended Lexicon) technique to map the namespace of the new “C&L”.

Keywords

Requirements Engineering, Software Transparency, Scenario, Language Extended Lexicon.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

A transparência é um termo chave que está presente em diversos contextos como o econômico e político, e atualmente um dos novos contextos em que se apresenta é a transparência de software. Sabemos que a comunicação através do software é realizada de maneira opaca, porém torna-se essencial que o software em si seja transparente, para que todos possam ter o conhecimento sobre o que exatamente o software faz [Fsf02]. Neste cenário a Transparência de Software torna-se um tema cada vez mais requisitado, seja pelos indivíduos, pela sociedade ou pelas organizações, já que o direito de ser informado e ter acesso a informação se constitui em um princípio democrático.

No futuro, a transparência vai ser um requisito exigido para toda aplicação, pois este é um conceito que está em processo de institucionalização, cabe portanto que pesquisas sejam realizadas no sentido de atender esta demanda para aqueles que usam o software ou são de alguma maneira afetados por ele [Leite06].

Software livre é um bom exemplo de transparência de software, onde a grande vantagem é a acessibilidade do código, que nos permite escolher, dentre as características do software, as que desejamos [Blank05]. Porém esta possibilidade está direcionada somente aqueles que possuem o conhecimento da programação.

A engenharia de software deve então propor mecanismos para tratar este novo requisito não funcional, a partir da utilização de métodos, técnicas e ferramentas apropriadas. A pesquisa sobre os meios para tornar o software mais transparente deve ser realizada com enfoque no ponto de vista do usuário, onde deve estar claro o que o software está fazendo ou pode fazer.

1.1.
Definições do Problema

O C&L é um software livre que propicia ao seu usuário um ambiente colaborativo que auxilia à edição de cenários e léxicos, descritos em linguagem natural semi-estruturada. Seu desenvolvimento foi realizado ao longo de anos,

de maneira incremental e com a participação de alunos da graduação e da pós-graduação [Silva04]. Este modo de desenvolvimento agregou bastante conhecimento ao grupo de engenharia de requisitos da PUC-Rio [GrupoER09], e a todos que se envolveram direta ou indiretamente no processo. Porém, a falta de uma arquitetura definida, de padronização e pouca documentação fez do C&L um software opaco a nível de código.

O problema de falta de documentação não é restrito ao C&L. Este é um problema comum na comunidade de desenvolvimento de software livre. Isso ocorre devido a informalidade destes projetos e a importância que os usuários finais dão apenas ao fato da instalação do software ser gratuita, sem se preocupar em cobrar dos desenvolvedores algum tipo de documentação. O resultado é que as únicas fontes de informação disponíveis sobre o software são, normalmente, o código fonte, fóruns e listas de e-mail contendo mensagens trocadas entre os integrantes do projeto. Esta falta de documentação acaba sendo um obstáculo a de entrada de novos participantes no desenvolvimento, manutenção e evolução do software. Como a entrada de novos membros é algo fundamental para projetos deste tipo, deve-se fazer um esforço para eliminar ou, se isto não for possível, pelo menos reduzir ao máximo este obstáculo.

1.2. Abordagem Proposta

Neste trabalho exploramos um método de desenvolvimento para software livre baseado no uso de cenários [Leite00]. O resultado da aplicação deste método será um documento único, o código fonte, onde teremos os cenários integrados com o código, facilitando sua leitura e entendimento, trazendo assim mais transparência para o software. Este método foi refinado durante sua aplicação na re-engenharia [Leite96] do software C&L, migrando da plataforma PHP [PHP09] para a plataforma Lua-Kepler [Kepler09] e adotando o *framework* MVC. Para produzir uma documentação complementar aos cenários inclusos no código fonte, utilizamos a técnica LAL [Leite 93] para mapear o espaço de nomes do software C&L.

O conceito de espaço de nomes é muito utilizado em diferentes áreas da ciência da computação. No nosso caso em particular, a nível de código fonte, o espaço de nomes nos fornece um contexto onde nomes são atribuídos a identificares únicos que comporão o código fonte do software (variáveis, funções, objetos) [Leite08].

As duas técnicas utilizadas neste trabalho, cenários e LAL, foram escolhidas porque são objeto de estudo do grupo de engenharia de requisitos da PUC-Rio [GrupoER09] há alguns anos, e trabalhos como [Christoph04], [Silva03] e [Leite08] comprovaram que seu uso neste contexto é promissor.

1.3. Trabalhos relacionados

Em um trabalho precursor sobre o uso de comentários como forma de documentação, [Knuth84] introduz o termo “programação literária”. Este termo identifica um método de desenvolvimento de programas bem documentados, utilizando a idéia de que os programas devem ser considerados uma obra literária. Neste trabalho, Knuth diz que devemos mudar nossa atitude tradicional na construção de programas, segundo ele “em vez de imaginarmos que nossa principal tarefa é instruir o computador sobre o que fazer, devemos nos concentrar em explicar a seres humanos o que queremos que o computador faça”.

Para dar suporte a sua idéia, Knuth desenvolveu um ambiente de programação literária chamado WEB. Neste ambiente os programadores devem programar em WEB, uma linguagem que mistura documentação com implementação, produzindo um único arquivo. Este arquivo será processado posteriormente por dois programas (*weave* e *tangle*), dando origem a dois novos arquivos, um contendo código, que poderá ser executado, e outro descrito em linguagem natural, que servirá de documentação.

Em um trabalho relacionado [Cassino96], Cassino propõe um *framework* para facilitar o desenvolvimento de ferramentas de suporte à programação literária. A motivação para criação deste *framework* veio da popularização da idéia de “programação literária”, que fez surgir um grande número de ferramentas de apóio a métodos semelhantes. O razão principal para o surgimento de tantas ferramentas foi o uso de diferentes linguagens. Cada ferramenta estava atrelada ao uso de uma linguagem específica.

Para demonstrar seu uso, foi desenvolvida uma ferramenta sobre o *framework* proposto. Esta ferramenta é dividida em dois programas. Um que extrai do arquivo fonte o código escrito pelo usuário e gera um arquivo que será posteriormente compilado para gerar o software desejado, e outro que processa o arquivo, adicionando informações e formatações para transformá-lo em um arquivo que possa ser processado por um formatador de textos e assim obter a

documentação final. Todo o processamento realizado pela ferramenta é baseado no uso de marcas pré-estabelecidas no arquivo fonte.

A idéia de uso de marcas no código fonte está por trás da criação do padrão JavaDoc [JavaDoc09]. Este padrão utiliza uma serie de marcadores para realizar a padronização de comentários no código Java e, posteriormente, produzir automaticamente um conjunto de páginas HTML navegáveis contendo as informações comentadas. Esta padronização também pode ser encontrada para as linguagens Lua [LuaDoc] e PHP [PHPDoc].

Em [Staa00] são estabelecidas regras para padronizar a utilização de comentários. Esta padronização visa uniformizar o estilo de programação, facilitando o entendimento, manuseio e evolução de código escrito por terceiros.

No trabalho [Christoph04] é sugerida a adoção da estrutura de cenários [Leite00] para padronizar o uso de comentários no código fonte. Segundo o autor, o uso de cenários ajudaria na organização e uniformidade dos comentários, facilitando a enumeração das funcionalidades e não-funcionalidades do software.

1.4. Organização do Documento

No Capítulo 2 detalharemos as duas técnicas de representação de conhecimento utilizadas neste trabalho, o LAL e cenários.

No Capítulo 3 apresentaremos a versão anterior do software C&L, a forma como foi desenvolvida, sua arquitetura e uma descrição detalhada de seu funcionamento.

No Capítulo 4 explicaremos o método baseado no uso de cenários e nossa proposta de mapeamento do espaço de nomes através do LAL.

No Capítulo 5 detalharemos passo a passo a aplicação do método proposto na re-arquitetura do software C&L.

No Capítulo 6 faremos uma comparação do trabalho realizado com os trabalhos relacionados, apresentaremos algumas conclusões e listaremos alguns trabalhos que precisam ser estudados.

2
Representações utilizadas

Este Capítulo apresenta em detalhes duas técnicas de representação muito utilizadas no contexto de engenharia de requisitos: Cenários e Léxico Ampliado da Linguagem. Estas duas técnicas são utilizadas pelo método proposto, que será detalhado no Capítulo 4.

2.1.
Léxico Ampliado da Linguagem (LAL)

O LAL [Leite93] é uma técnica criada para ajudar na elicitação e representação da linguagem utilizada na aplicação. Esta técnica é baseada na idéia de que cada aplicação possui uma linguagem específica. Portanto, a principal tarefa dos engenheiros de requisitos na construção do LAL é focar na identificação de símbolos peculiares ao domínio em que a aplicação está inserida e seus significados, sem se preocupar com o entendimento do problema. A identificação destes símbolos e de seus significados ajuda na compreensão do Universo de Informação (UdI) da aplicação.

Os símbolos do LAL são descritos através de uma noção e um impacto. A noção explica o significado literal do símbolo, e o impacto descreve os efeitos do uso ou ocorrência do símbolo no domínio sob estudo.

Podemos classificar os símbolos do léxico gramaticalmente como sujeito, verbo, estado ou objeto. A Tabela 1, criada com base na tabela existente no software C&L, sintetiza estes tipos, mostrando uma breve descrição de cada um, de acordo com a noção e o impacto.

	Noção	Impacto
Sujeito	• Quem é o sujeito?	• Quais ações podem ser executadas por este sujeito?
Verbo	• Quem executa a ação? • Quando ela acontece? • Quais os procedimentos envolvidos na ação?	• Quais os reflexos desta ação no ambiente? • Quais estados são alcançados após a execução da ação?
Objeto	• Definir o objeto e identificar com quais outros objetos ele se relaciona.	• Quais ações podem ser aplicadas a este objeto?
Estado	• O que este estado significa? • Quais as ações levaram a este estado?	• Quais outros estados e ações que podem ocorrer a partir deste estado?

Tabela 1 – Descrição dos tipos de entrada do LAL.

Na Figura 1 podemos ver o Modelo Entidade Relacionamento construído para o LAL [Leite00]. O modelo possui cinco entidades principais: LAL, símbolo, nome, noção e impacto. O LAL é composto de um ou mais símbolos, cada símbolo é identificado por um nome ou frase, é definido por sua noção e possui um impacto.

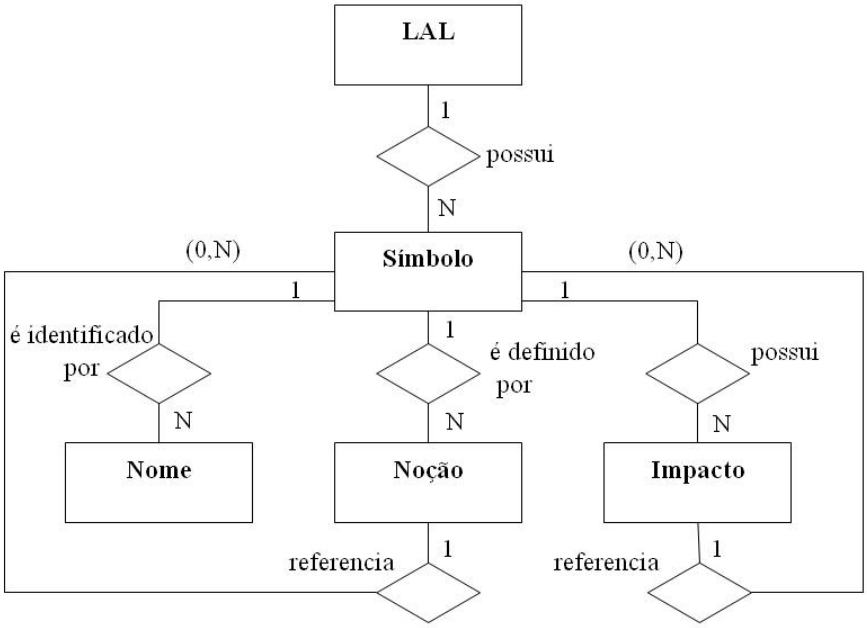


Figura 1 – Modelo entidade relacionamento do LAL.

Adicionalmente, a descrição de noção e impacto dos símbolos do LAL segue dois princípios, o da circularidade e do vocabulário mínimo. O primeiro afirma que os símbolos do léxico devem utilizar ao máximo a referência a outros símbolos do próprio domínio na descrição de sua noção e impacto. O segundo princípio afirma que para se fazer a descrição de noção e impacto, deve se utilizar o mínimo possível de termos externos a linguagem e, ao utilizarmos termos externos à linguagem, devemos escolher os mais freqüentes e de significado claro, pertencentes a um vocabulário restrito da linguagem natural. Por exemplo, se tratando da língua inglesa, o projeto Collins [Willis90] construiu um vocabulário com 700 palavras que formariam o núcleo da língua inglesa. Segundo os pesquisadores, 70% de todos os textos escritos em inglês utilizam apenas estas palavras.

Podemos ver na Figura 2 um exemplo de símbolo do léxico, baseado no sistema para agendar reuniões apresentado em [Leite98], que segue os dois princípios apresentados acima. As palavras que estão sublinhadas são símbolos do LAL que foram utilizados na descrição da noção e do impacto do símbolo “Agendador de Reuniões”.

Nome:	Agendador de Reuniões
Noção:	1- Sistema para organizar reuniões
Classificação:	Sujeito
Impactos:	1- Produz uma data de reunião e um local de reunião para um pedido de reunião. 2- Produz um conflito sério ou um conflito normal para um pedido de reunião. 3- Produz uma nova tentativa de agendamento se não houver local de reunião disponível.

Figura 2 – Exemplo de um símbolo do LAL [Leite98].

Ao seguirmos o princípio da circularidade automaticamente criamos uma rede de relacionamento entre os símbolos do LAL. Esta rede de relacionamentos permite um rastreamento interno dos símbolos e pode ser mapeada através de um grafo. A Figura 3 mostra o grafo de relacionamento do símbolo “Agendador de Reuniões”.

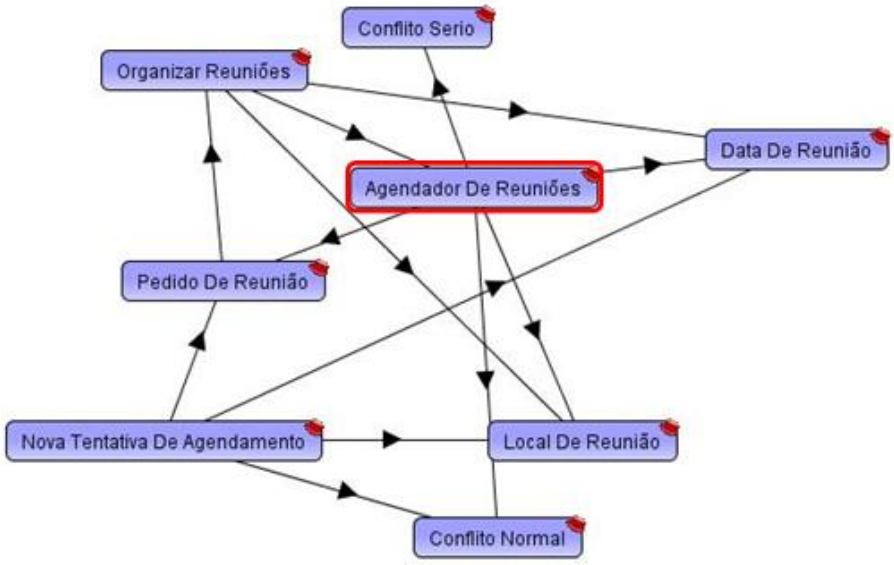


Figura 3 – Exemplo de um grafo de relacionamentos.

2.2. Cenários

A necessidade de facilitar a comunicação entre engenheiros e clientes motivou a pesquisa e o desenvolvimento de métodos que ajudassem na interação de todos os participantes, durante o processo de definição dos requisitos de um sistema. Para elicitar os requisitos os engenheiros devem entender, modelar e conhecer o domínio da aplicação do software. Os clientes devem interagir com os engenheiros para verificar se o entendimento foi correto e validar seus modelos propostos.

O uso de cenários na engenharia de requisitos ajuda tanto na compreensão do sistema que será desenvolvido quanto na validação do conhecimento do engenheiro pelo cliente. Isto é facilitado pelo fato de que os cenários são descritos em linguagem natural e modelam situações do sistema. Isto faz com que clientes se sintam mais a vontade em interagir com os engenheiros e entendam melhor o processo de descobrimento dos requisitos.

Existem vários modelos de cenários desde o mais formal, que permite a geração e validação automática de cenários [Hsia94], até um modelo informal como o mostrado em [Carroll94]. O modelo que adotamos para este trabalho é intermediário: ele visa descrever uma situação específica do software através de uma linguagem natural semi-estruturada [Leite00]. A estrutura desde modelo é composta das seguintes entidades: título, objetivo, contexto, recursos, atores, episódios, exceções e o atributo restrição, como mostra a Figura 4. Atores e cenários são uma lista. Título, objetivo, contexto e exceção são sentenças declarativas simples. Os episódios são sentenças escritas em linguagem simples, que dão uma descrição operacional do comportamento da situação modelada pelo cenário.

Um cenário deve satisfazer um objetivo, o caminho para alcançar este objetivo é descrito passo a passo pelos seus episódios. Os episódios representam o curso principal das ações de um cenário, mas também incluem variações e outras alternativas possíveis. Uma exceção pode ocorrer durante a execução dos episódios, sinalizando que existe um empecilho para satisfazer o objetivo. O tratamento para a exceção não precisa necessariamente satisfazer o objetivo principal.

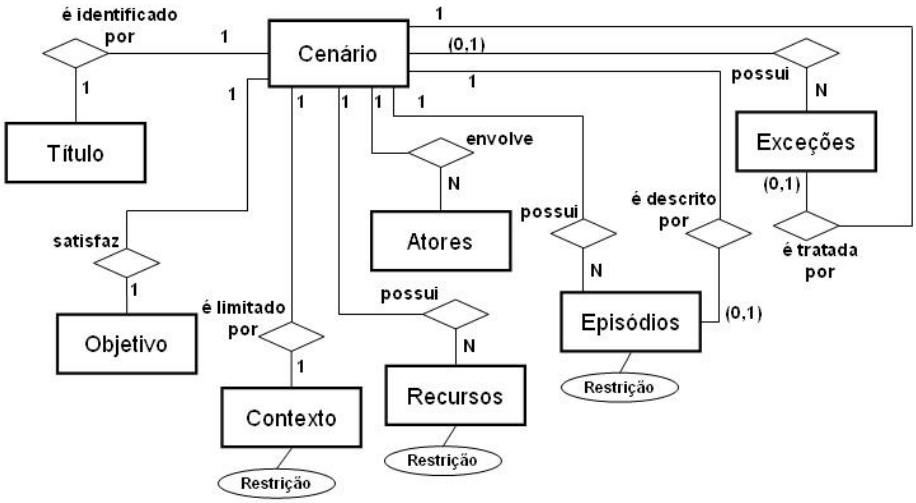


Figura 4 – Modelo entidade relacionamento para um cenário.

O atributo restrição é usado para caracterizar aspectos não funcionais que podem fazer com que o objetivo de um cenário não seja alcançado na qualidade desejada. Estes aspectos não funcionais podem estar relacionados ao contexto, recursos e episódios.

Uma exceção pode interromper o curso normal de um cenário. Cada exceção deve ser descrita por uma sentença simples que deve especificar a causa da interrupção e como tratá-la. Um outro cenário pode ser utilizado para este fim.

O contexto é descrito através da localização geográfica, localização temporal e precondições. Cada um desses elementos pode ser descrito por uma ou mais sentenças simples, ligadas pelos conectores lógicos: “e” e “ou”.

Os episódios são seqüenciais e podem ser simples, condicionais ou opcionais. Os simples são os necessários para completar o cenário. Os condicionais são aqueles cuja ocorrência depende de uma condição específica. As condições internas de um cenário podem ocorrer devido a pré-condições, alternativas, restrições de atores ou recursos e episódios anteriores. E por fim, os episódios opcionais são os que podem ocorrer ou não, dependendo de condições que não podem ser explicitamente detalhadas.

Independente de seu tipo, um episódio pode ser descrito através de uma sentença simples ou através de um outro cenário, o que nos permite decompor um cenário, facilitando seu entendimento.

Outro fator que contribui para a compreensão de cenários, além do uso da linguagem natural e descrição de situações bem definidas, é a existência de relacionamentos. De acordo com [Breitman00], cenários estão ligados a outros

cenários através de elos, formando uma complexa rede de relacionamentos. Estes elos podem ser de quatro tipos distintos: subcenário, pré-condição, exceção e restrição.

Um cenário é dito subcenário de outro quando aparece em pelo menos um de seus episódios. Este relacionamento é útil quando:

- Detectamos um comportamento comum em vários cenários. Quando isto acontece podemos isolar este comportamento em um novo cenário, reduzindo a redundância de informações.
- Há um curso de ação, alternativo ou condicional, complexo dentro de um sistema. Quando isto ocorre podemos separar este comportamento em um novo cenário para um maior detalhamento, facilitando sua compreensão.
- Desejamos melhorar a descrição de uma situação quando um objetivo importante e bem definido é encontrado dentro de um cenário. Diante desta situação, podemos utilizar um novo cenário para destacar e detalhar o novo objetivo detectado.

A pré-condição é um relacionamento definido dentro do componente contexto de um cenário. Um cenário que age como pré-condição para outro deve aparecer no seu contexto. Este relacionamento nos permite a definição de uma seqüência entre os cenários e a especificação de estágios que devem ser completados antes da execução de outros.

Uma relação de exceção acontece quando um cenário aparece no componente exceção de outro cenário. Este relacionamento nos permite descrever o tratamento de uma exceção utilizando outro cenário, o que nos permite um maior detalhamento.

Quando um cenário é utilizado para detalhar aspectos não funcionais que restringem o funcionamento correto de outro cenário, temos o relacionamento de restrição. Este relacionamento ocorre quando um cenário aparece no atributo restrição de outro. O atributo restrição pode ser utilizado nos campos contexto, recursos e episódios.

O processo de construção de cenários está intimamente ligado ao Léxico Ampliado da Linguagem, técnica abordada na seção anterior. Podemos relacionar os cenários com outros cenários e também com os símbolos do léxico do domínio em questão. Desta forma, além de elos entre cenários teremos elos entre cenários e símbolos do léxico, facilitando a compreensão do cenário através do entendimento dos termos próprios da linguagem da aplicação.

A Figura 5 nos mostra um exemplo de cenário baseado no modelo apresentado. As palavras sublinhadas escritas em letras maiúsculas são outros cenários utilizados pelo cenário principal. As palavras sublinhadas escritas em letras minúsculas são símbolos pertencentes ao Léxico Ampliado da Linguagem referenciados pelo cenário.

Título:	Agendar reunião.
Objetivo:	Marcar a reunião no melhor dia, horário e local para que todos os participantes possam comparecer.
Contexto:	O cenário REQUISITAR REUNIÃO deve ter sido executado.
Atores:	Sistema, participantes da reunião.
Recursos:	lista de participantes, assunto, datas preferenciais, horários preferenciais, locais preferenciais.
Episódios:	1- O sistema informa o assunto e requisita aos participantes suas datas preferenciais, horários preferenciais e locais preferenciais para a reunião. 2- O sistema analisa as datas de reunião, horários de reunião e locais de reunião fornecidos pelos participantes. Restrição: Todos os participantes da reunião devem fornecer os dados requisitados. 3- Sistema define a data de reunião, o horário de reunião e o local de reunião. 4- COMUNICAR DATA DA REUNIÃO
Exceção:	Se não houver uma data de reunião, horário de reunião e local de reunião comum, em que todos os participantes possam comparecer, então o sistema deve REALIZAR NOVA TENTATIVA DE AGENDAMENTO.

Figura 5 – Exemplo de cenário.

Utilizando a idéia apresentada anteriormente, podemos criar um grafo que representa os relacionamentos do cenário “agendar reunião”, apresentado na Figura 5. Cada cenário e símbolo do léxico representará um nó e os relacionamentos serão identificados pelas arestas. O grafo obtido pode ser visualizado na Figura 6. As arestas direcionadas identificam quem referencia e quem é referenciado. Neste exemplo temos os relacionamentos de subcenário, pré-condição e exceção, que estão explicitados na Figura através do nome junto à aresta.

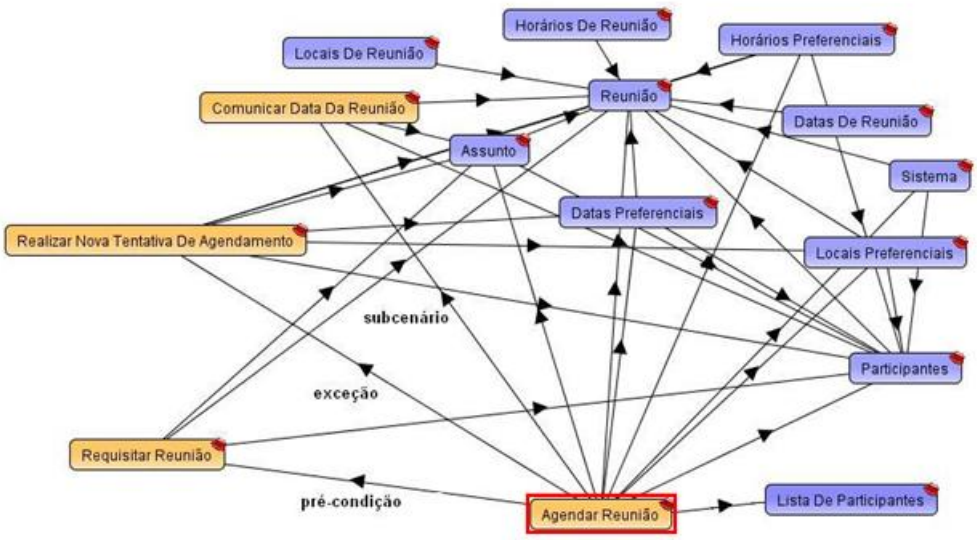


Figura 6 – Grafo de relacionamentos entre cenários e símbolos do léxico.

3
Software C&L

Este capítulo apresenta a versão anterior do software Cenários & Léxicos (C&L), que passou por um processo de re-engenharia durante este trabalho. Ao longo do capítulo descreveremos seu processo de desenvolvimento e a evolução do software, desde sua criação até atualmente. Também detalharemos sua arquitetura e os problemas decorrentes do seu processo de desenvolvimento, que tornaram o seu código fonte difícil de entender e, consequentemente, de manter e evoluir. Por fim, explicaremos cada uma das funcionalidades oferecidas pelo C&L, e como utilizá-las.

3.1.
Introdução

O C&L [Silva05] é um software para edição de símbolos do léxico e cenários. Este software disponibiliza um ambiente em que usuários podem interagir para construir, manter, evoluir e gerenciar projetos contendo cenários e símbolos do léxico.

O primeiro protótipo do C&L foi desenvolvido durante a disciplina Princípios de Engenharia de Software, oferecida pela PUC-Rio ao curso de Engenharia de Computação, no primeiro semestre de 2002. O software atual é o resultado da evolução deste protótipo por estudantes da graduação, mestrado e doutorado do Departamento de Informática da PUC-Rio ao longo de seis anos. O C&L foi desenvolvido desde o início com a filosofia de software livre e seu código fonte está disponível para quem quiser baixá-lo.

No ano de 2007 o C&L passou por um processo de manutenção e atualização, realizado pelos integrantes do grupo de engenharia de requisitos da PUC-Rio [GrupoER09]. Este processo corrigiu algumas de suas funcionalidades e adicionou uma nova ferramenta para geração de grafos, desenvolvida por um aluno da PUC-Rio em seu projeto final de graduação [Lecesne07].

3.2. Arquitetura

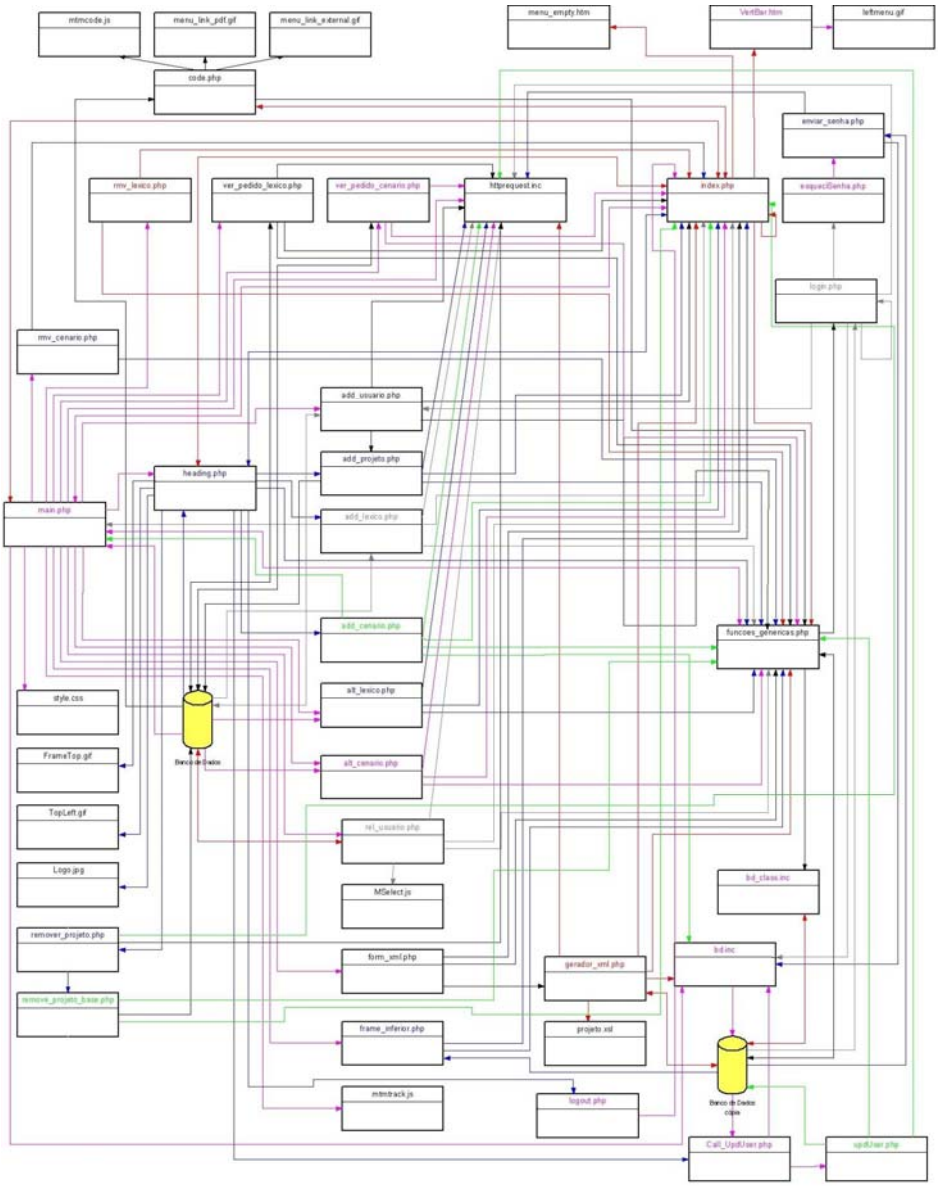


Figura 7 – Mapa do relacionamento entre arquivos [C&L09].

O desenvolvimento do C&L foi realizado utilizando-se software livre, são eles: PHP [PHP09] versão 4, como linguagem de implementação, o banco de dados MySQL [MySQL09], o servidor Web Apache http Server [Apache03] e a ferramenta de controle de versão e gerenciamento dos códigos-fonte CVS [CVS09]. O C&L possui por volta de onze mil linhas de código, distribuídas em noventa e cinco arquivos. Na Figura 7 pode-se ver um mapeamento de alguns dos arquivos que compõem o software.

O envolvimento de diversos grupos, com os mais variados níveis de conhecimento em programação, a falta da definição de padrões e de uma arquitetura no início do desenvolvimento refletiram negativamente na qualidade do código fonte do C&L. Seu código apresenta vários estilos de programação, falta de padronização e uso de mais de um de idioma na escolha dos nomes de variáveis e funções, mistura entre as diversas linguagens utilizadas no desenvolvimento do software (PHP, HTML, JavaScript e SQL), dentre outros problemas. O uso dos cenários como forma de documentação, apesar de se ser utilizado apenas em parte do código, ajuda bastante na sua compreensão. Ainda assim, o código do C&L é difícil de entender e, conseqüentemente, de evoluir.

3.3. Funcionamento

A descrição das funcionalidades que apresentaremos a seguir foi feita com base no uso do software C&L, que pode ser acessado através do endereço [C&L09].

Acessando a página inicial do C&L [C&L09], mostrada na Figura 8, o usuário encontra um pequeno texto explicativo do software e tem a sua disposição uma série de elos. Estes elos levam a artigos e apresentações que explicam os conceitos envolvidos no software e como utilizá-lo. Além disto, podemos encontrar elos para *download* do código fonte completo e *scripts* para construção da estrutura do banco de dados. Há também outros elos interessantes, dentre eles destacamos uma lista de projetos criados e tornados públicos pelos seus administradores, um manual de utilização, os casos de teste utilizados e o relacionamento entre os diversos cenários que são utilizados para sua documentação.

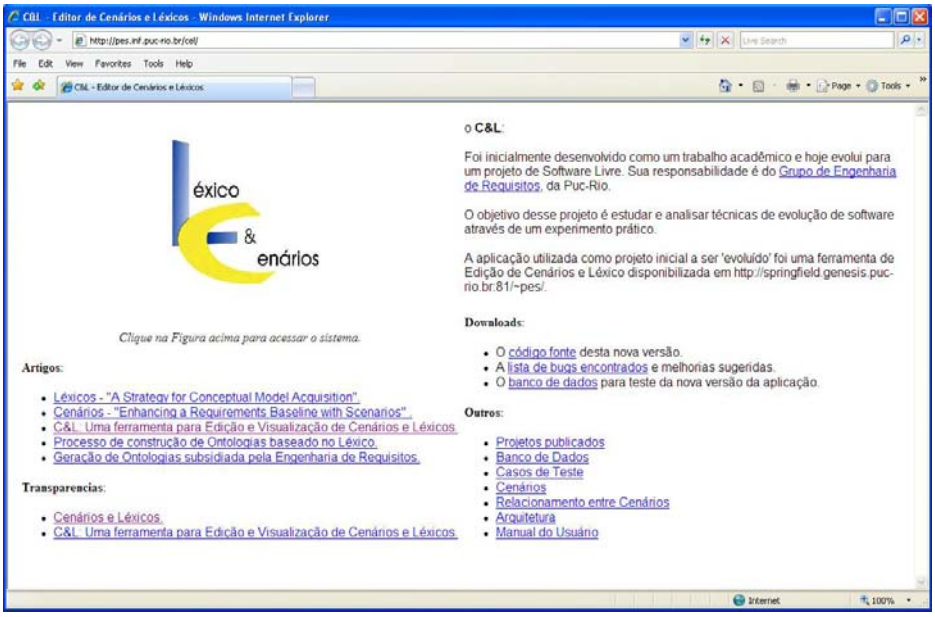


Figura 8 – Página inicial do sistema.

Para começar a utilizar o C&L é preciso cadastrar-se no sistema. O cadastro pode ser feito através do atalho “Cadastrar-se” presente na página de *login* do sistema (Figura 9), que é acessada através de um clique no logo do C&L que se encontra na página inicial. Após acessar o atalho o usuário será direcionado a uma tela contendo um formulário básico de cadastro (Figura 10). Dentre as informações solicitadas estão o *login* e senha que o usuário utilizará para acessar o sistema posteriormente, através da tela de *login*.

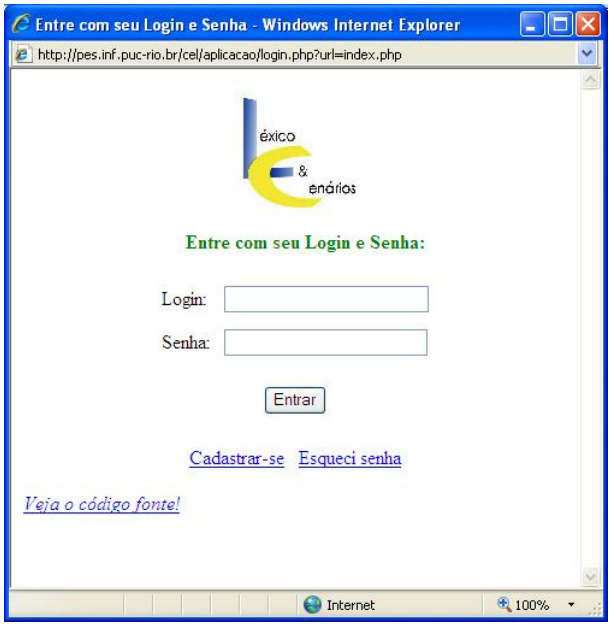


Figura 9 – Tela de login do sistema.

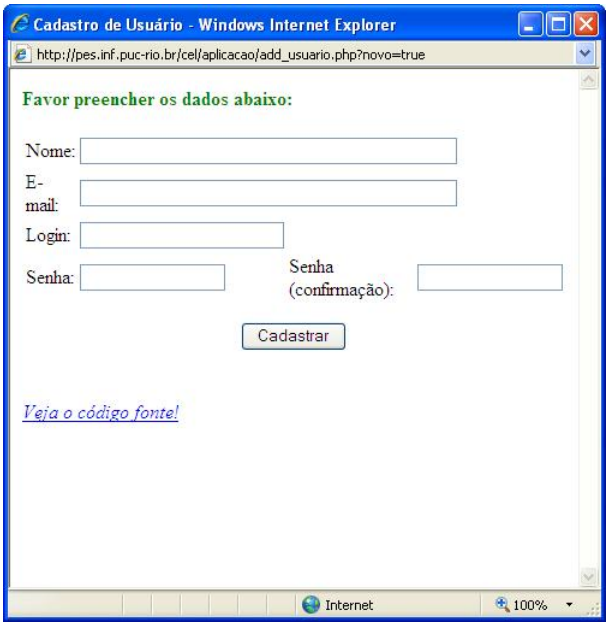


Figura 10 – Formulário de cadastro de usuário.

É também através da tela de *login* que o usuário pode solicitar uma nova senha. Ao clicar no atalho “Esqueci senha” o usuário é direcionado a uma nova tela onde deve informar seu *login* e e-mail. Ao confirmar que realmente quer mudar sua senha, e se o campo *login* e e-mail foram preenchidos corretamente, o sistema irá substituir a senha do usuário por uma nova, gerada randomicamente, e a enviará por e-mail para o usuário. O usuário poderá alterar a senha para uma de sua escolha no próximo acesso ao software.

Após realizar o cadastro no sistema o usuário é levado à página principal do software (Figura 11). Nesta página encontram-se diversos elementos importantes para a sua utilização. Estes elementos são: o menu de projetos, menu principal, menu de léxicos e cenários e área de trabalho. O primeiro destes elementos que destacamos é o menu de projetos, localizado na parte superior direita da página principal. Este menu exibe os projetos criados pelo usuário, identificados por um asterisco do lado esquerdo do seu nome, e os projetos que participa em colaboração com outros usuários, estes identificados pela ausência do asterisco. É através deste menu que o usuário selecionará com qual projeto deseja trabalhar no momento.

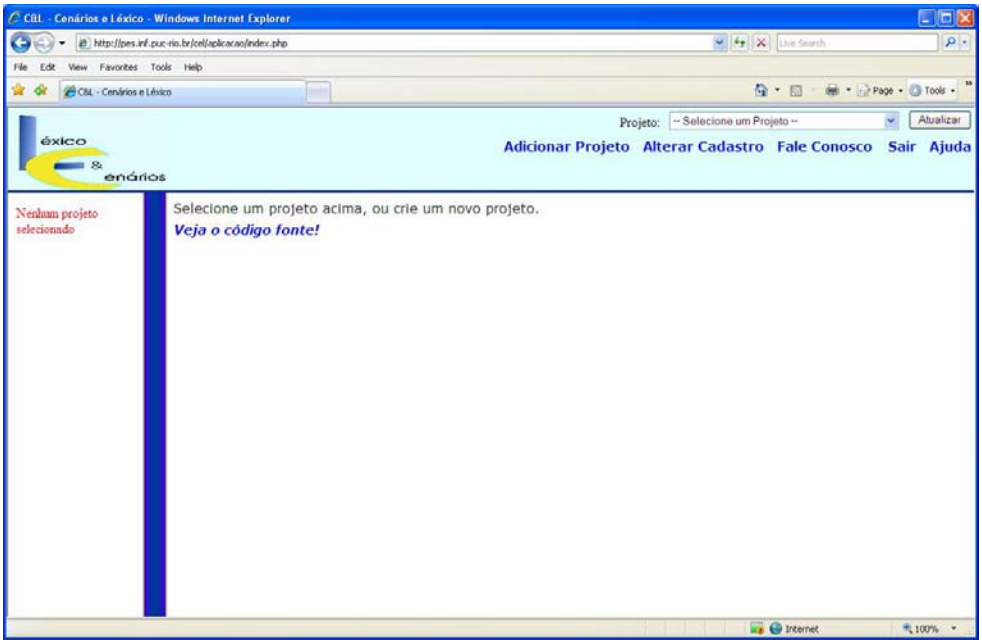


Figura 11 – Página principal do sistema.

Logo abaixo do menu de projetos há outro menu, que chamaremos de menu principal. As opções exibidas por este menu irão mudar caso um projeto seja selecionado pelo usuário. As seguintes opções serão exibidas, se nenhum projeto foi previamente selecionado: adicionar projeto, alterar cadastro, fale conosco e ajuda. A opção alterar cadastro levará o usuário a uma nova tela contendo um formulário já preenchido com as informações do usuário cadastradas no software, permitindo que sejam alteradas. A opção fale conosco permite que o usuário entre em contato com o grupo responsável pela manutenção e evolução do C&L. A opção ajuda exibe uma página com os tópicos de ajuda da ferramenta. A opção cadastrar projeto direciona o usuário para um pequeno formulário que deve ser preenchido para a criação de um novo projeto. Após criar um novo projeto o usuário deve clicar no botão “Atualizar”, que fica localizado do lado direito do menu de projetos, para que o novo projeto apareça no menu para seleção.

Ao selecionarmos um projeto do menu de projetos quatro novas opções aparecem no menu principal, são elas: adicionar cenário, adicionar léxico, remover projeto e “info”. A opção remover projeto permite que o projeto selecionado seja excluído. Ao excluirmos um projeto todos os símbolos do léxico e cenários vinculados a este projeto também serão excluídos. Ao clicarmos na opção “Adicionar cenário” o usuário é redirecionado para um formulário de inclusão de cenário (Figura 12), que deve ser preenchido para adição de um

novo cenário ao projeto. O mesmo acontece com a opção “Adicionar léxico”. Só que neste caso, o usuário é redirecionado para um formulário de adição de símbolo do léxico (Figura 12). Conforme os símbolos do léxico e os cenários são incluídos, eles vão aparecendo no menu de léxicos e cenários, que se encontra no lado esquerdo da página principal do sistema.

Quando o usuário seleciona um símbolo do léxico ou cenário presente no menu de léxicos e cenários a ferramenta monta, automaticamente, uma rede de relacionamentos, identificando quais cenários e símbolos do léxico são referenciados no corpo do elemento selecionado, e quais símbolos do léxico e cenários referenciam o elemento selecionado. Os relacionamentos identificados são utilizados para criar dois tipos de rastreamento, o rastreamento “para frente” e o rastreamento “para trás”. O rastreamento “para frente” utiliza atalhos no corpo do elemento selecionado para permitir que o usuário navegue entre os elementos referenciados. O rastreamento “para trás” é utilizado para criar atalhos que ficam dispostos abaixo do elemento selecionado, permitindo que o usuário navegue entre os elementos que o referenciam. Como existem dois tipos de elementos possíveis, cenários e símbolos do léxico, a ferramenta faz uma distinção dos atalhos, criando atalhos escritos em letras maiúsculas para cenários e em letras minúsculas para símbolos do léxico. Um exemplo desta característica do C&L pode ser visto na Figura 13.

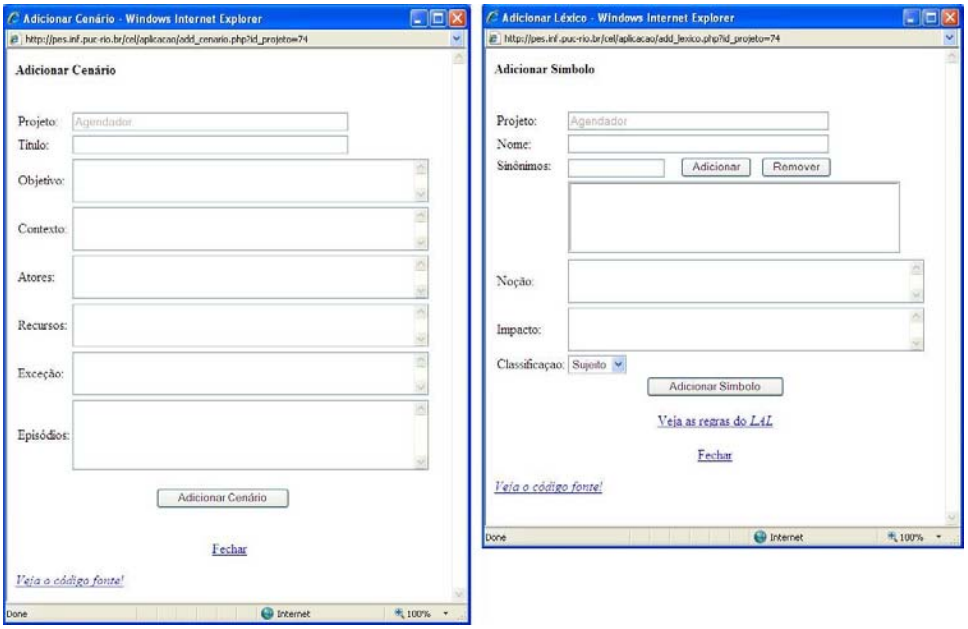


Figura 12 – Formulários de inclusão de cenário e símbolo do léxico.

A opção “info” do menu principal de projetos leva o usuário a uma nova tela que contém informações sobre o projeto, como nome, data da criação e descrição e um menu com as seguintes opções (Figura 14): adicionar usuário não cadastrado ao projeto, adicionar usuários já existentes ao projeto, verificar pedidos de alteração de cenários e símbolos do léxico, gerar grafo do projeto, gerar XML do projeto e recuperar XML do projeto. As outras opções do menu que não foram citadas são relacionadas à geração automática de ontologias, parte do C&L que ainda se encontra em desenvolvimento. As opções deste menu, exceto a geração de grafo do projeto, são restritas ao administrador do projeto.

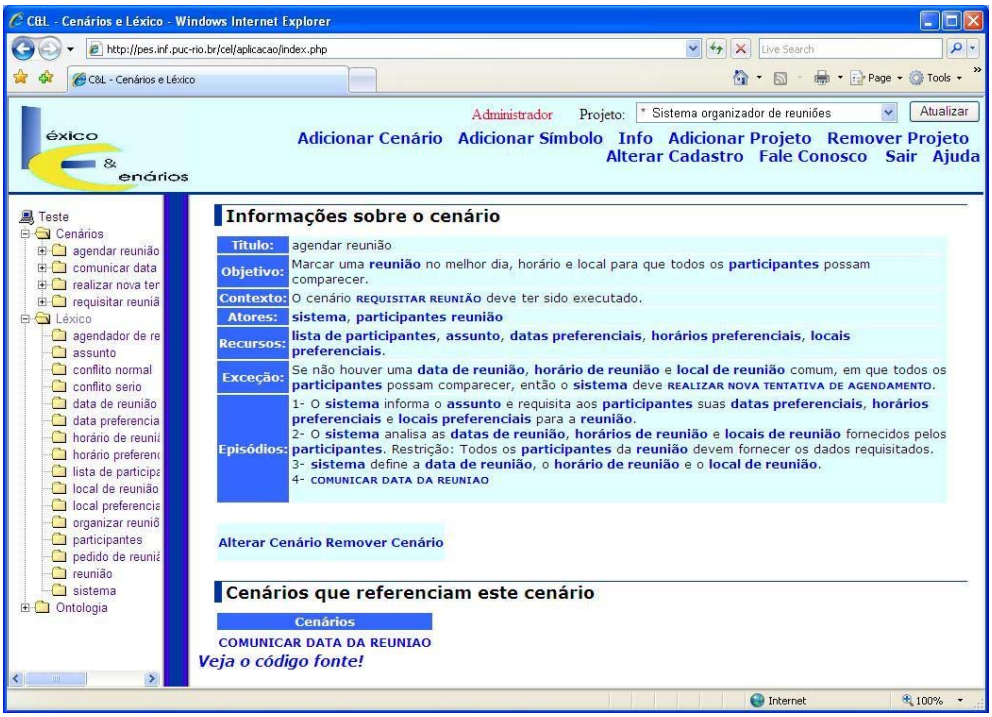


Figura 13 – Exemplos de atalhos criados pelo C&L.

A primeira opção permite a adição de um colaborador que não possui cadastro no sistema. Neste caso o próprio administrador do projeto irá preencher o formulário de cadastro do colaborador que deseja adicionar. O usuário cadastrado estará automaticamente vinculado ao projeto como colaborador. A segunda opção permite que o administrador adicione usuários já cadastrados no sistema como colaboradores do projeto. Isto é feito através da seleção dos usuários de uma lista, como pode ser visto na Figura 15.

As alterações feitas por colaboradores do projeto não são realmente efetivadas sem a autorização do administrador do projeto. Para visualizar as

alterações sugeridas pelos colaboradores e decidir se as autoriza ou não, os administradores contam com as opções verificar pedidos de alteração de cenários e símbolos do léxico. Ao clicar em uma destas opções será exibida uma lista com as alterações sugeridas pelos colaboradores do projeto e suas justificativas. Se o administrador quiser acatar a modificação sugerida basta selecionar a opção “aprovar” que as alterações serão realmente efetivadas. Caso contrário basta selecionar a opção “remover da lista” para ignorá-las.

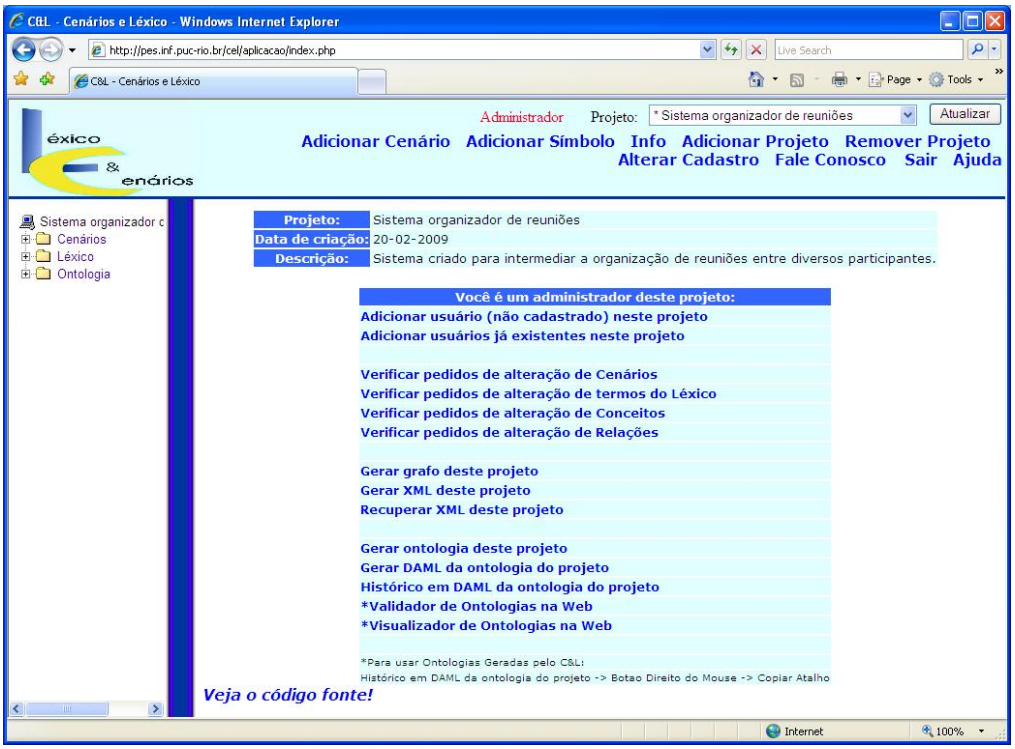


Figura 14 – Menu “info”.

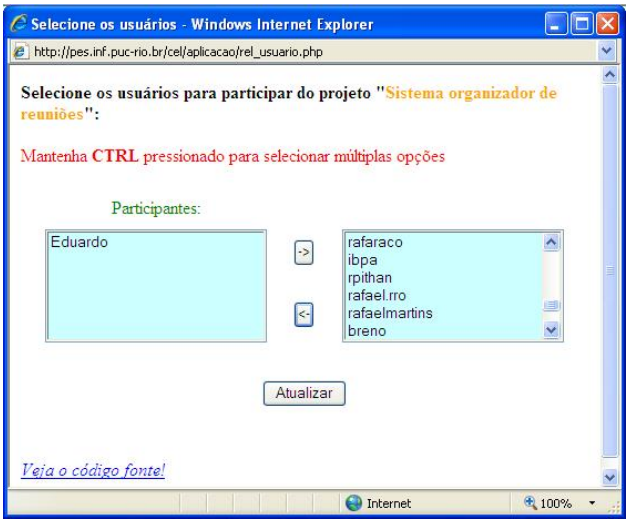


Figura 15 – Tela para adicionar colaboradores ao projeto.

Disponibilizar o código fonte de um software é interessante, pois permite que o usuário interessado visualize como foi codificada cada situação. Mas em códigos muito extensos, como a maioria, é difícil relacionar as situações do software com os trechos correspondentes do código fonte. Uma característica importante do C&L, que pode ser encontrada no final de cada página que compõe o sistema, propõe uma solução para este problema. Esta solução é o elo “Veja o código fonte!”. Esta característica é interessante, pois permite que visualizemos o código de um evento específico do software. Na Figura 16, pode-se ver um exemplo desta característica. Nesta Figura é exibido o código fonte da funcionalidade mostrada na Figura 15, este código foi exibido após um clique no elo “Veja o código fonte!”.

Figura 16 – Exemplo de visualização do código.

É possível representar todas as informações de um projeto criado no C&L em um arquivo, gerado automaticamente, no formato XML, através da opção “gerar XML do projeto”. Este arquivo pode ser exibido como um arquivo comum XML (Figura 17) ou como uma página HTML navegável, criada através de uma transformação do XML em HTML (XSLT). Uma vez criado o arquivo XML de um projeto, este ficará disponível para todos os usuários do sistema. Esta característica facilita a integração do C&L com outras ferramentas. Na Figura 17 podemos visualizar um exemplo de XML gerado pelo C&L.

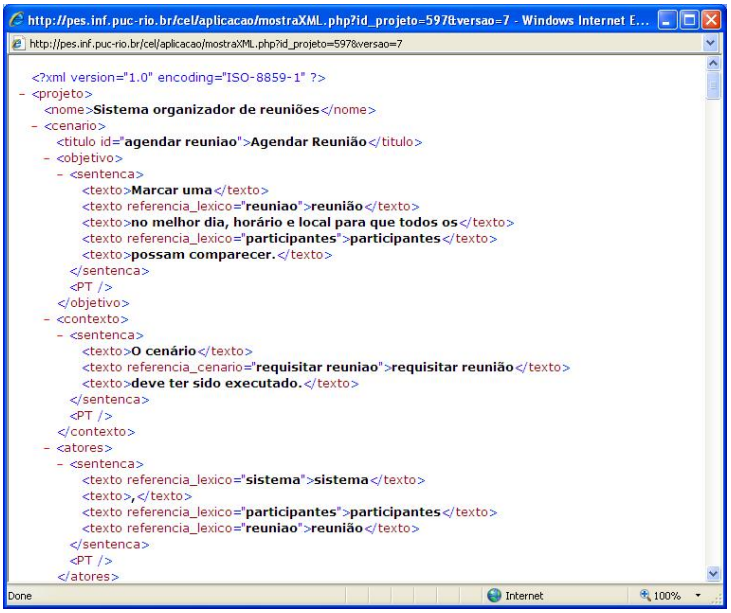


Figura 17 – Exemplo de XML gerado pelo C&L.

Uma das ferramentas integradas ao C&L através do XML é a de construção de grafos [Lecesne07] (Figura 18). Esta ferramenta pode ser acessada através da opção “gerar grafo do projeto”. Ao selecionar esta opção serão listadas todas as versões do XML geradas previamente, para que o usuário selecione a que deseja para gerar o grafo. A ferramenta de visualização de grafos tem uma série de características interessante, dentre elas podemos destacar a diferenciação entre cenários e símbolos do léxico através de cores (cenários são marcados com a cor laranja e os símbolos com a cor azul), possibilidade de escolher visualizar apenas cenários ou apenas símbolos ou ambos e possibilidade de focar em um elemento (Figura 19), destacando os elementos com que se relaciona e seus relacionamentos, enquanto os outros ficam em segundo plano.

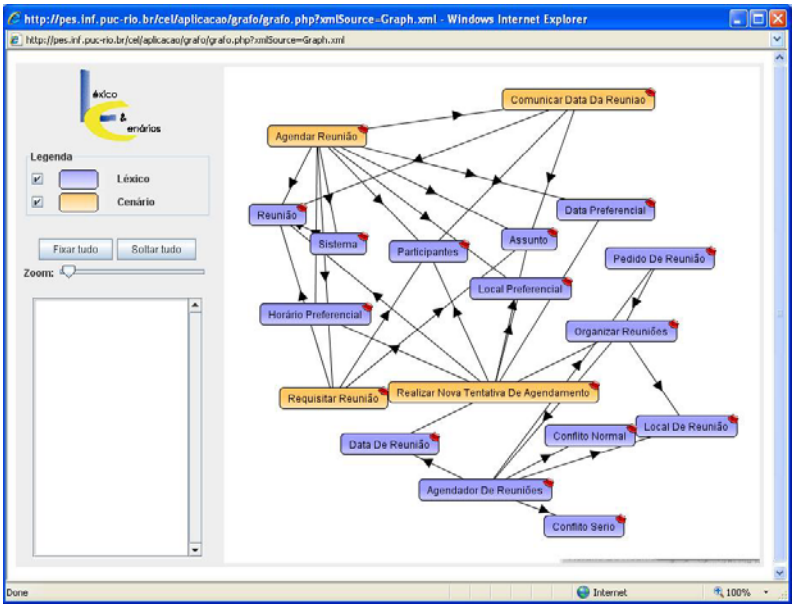


Figura 18 – Exemplo de grafo gerado pelo C&L.

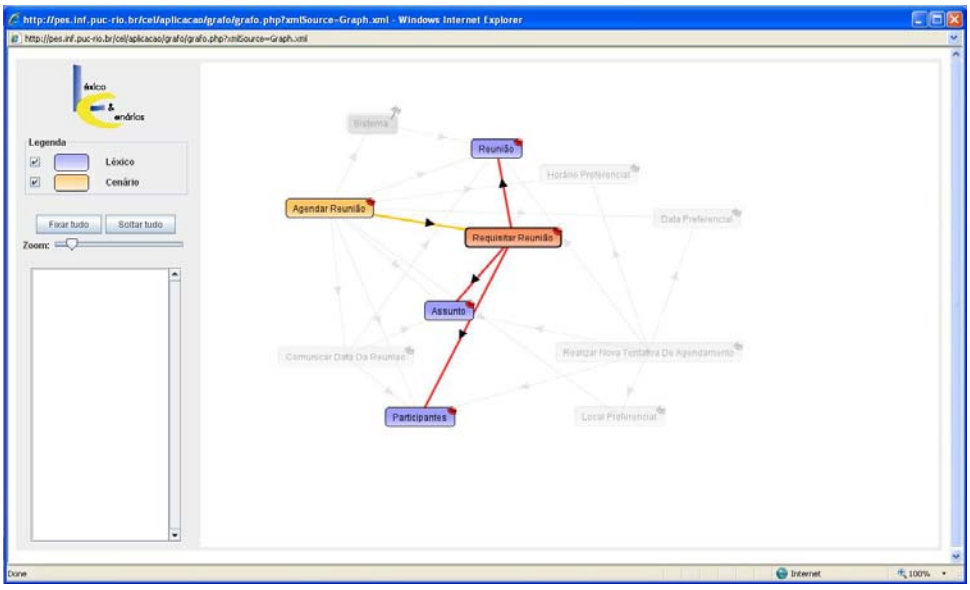


Figura 19 – Exemplo de foco no grafo gerado pelo C&L.

4
Construindo a nova arquitetura

Neste capítulo detalharemos o método de desenvolvimento proposto e mostraremos um exemplo da sua aplicação na re-arquitetura do software C&L. Também mostraremos o mapeamento do espaço de nomes que foi realizado utilizando a própria ferramenta.

4.1.
Método de desenvolvimento

Um dos principais problemas do desenvolvimento de software livre é a documentação do projeto. Isto se deve ao fato de que a maioria dos projetos deste tipo conta com a participação de uma grande quantidade de desenvolvedores, muitas vezes espalhados por diversos países. Aliado a isto temos a falta de cobrança de uma documentação de qualidade por parte dos usuários dos softwares, que dão importância apenas ao fato do software ser gratuito.

Entender o código fonte de um software, sem que nenhuma técnica tenha sido utilizada para facilitar sua leitura e sem dispor de documentação é uma tarefa trabalhosa e custosa [Biggerstaff94] [Meyrhauser94] [Brooks77].

A utilização de cenários integrados com o código fonte [Silva03] [Christoph04] é uma técnica que visa melhorar sua apresentação, de maneira estruturada e uniforme, facilitando sua leitura e entendimento. Esta técnica de apresentação melhora, sem dúvida, a leitura e o entendimento do código fonte do software, mas também exige um esforço extra do desenvolvedor para realizar sua anotação.

Para abordar estas questões propomos um método de desenvolvimento baseado no refinamento de cenários. A aplicação deste método tem como resultado um documento único, o código fonte, onde teremos cenários integrados com o próprio código, facilitando sua leitura e entendimento. Além disto, o método utiliza a idéia de integração de cenários por generalização, para proporcionar uma visão global do sistema, facilitando o entendimento e gerência dos grupos de cenários que comporão o sistema.

4.1.1.
Construção do LAL

O primeiro passo do método é a descrição dos símbolos que possuem significado específico dentro do domínio, utilizando a técnica LAL, apresentada na subseção 2.1. O processo de construção do LAL é composto pelas seguintes fases: identificação dos símbolos, identificação da semântica dos símbolos e validação junto ao usuário. Mais detalhes sobre estas fases e heurísticas para a construção do LAL podem ser encontradas em [Leite94].

Na Figura 20 podemos ver um exemplo de entrada do LAL. Este símbolo foi extraído do LAL do domínio de aplicação do C&L. As palavras sublinhadas também fazem parte do LAL.

Nome: Símbolo do léxico

Noção: São palavras chave que possuem um significado específico no contexto em que estão inseridas. São descritas através de sua noção e impacto.

Classificação: Objeto.

Impactos:

- Pode ser cadastrada em um projeto.
- Pode ser removida de um projeto.
- Pode ser alterada.
- Pode referenciar outros símbolos do léxico.
- Pode ser referenciado por outros símbolos e por cenários.

Sinônimos: símbolo, léxico, símbolos do léxico, léxicos, símbolos.

Figura 20 – Exemplo de uma entrada do LAL

4.1.2.
Descrição das situações do sistema através de cenários

Neste passo serão construídos os primeiros cenários, que descreverão o funcionamento do sistema. Para tanto, deve ser feito um levantamento das situações que compõem o sistema e cada situação identificada deve ser mapeada, de acordo com a representação de cenários definida em [Leite00]. A Tabela 2 resume a representação de cenários adotada, descrevendo o uso de cada entidade representativa do cenário.

Título	Identificador do cenário. Deve ser único.
Objetivo	Descrição da finalidade do cenário. Deve ser descrito também como este objetivo é alcançado.
Contexto	Descrição do estado inicial do cenário. Deve ser descrito através de pré-condições, localização geográfica ou temporal.
Atores	São entidades envolvidas diretamente com o software. Um ator, para ser válido, deve aparecer em pelo menos um dos episódios.
Recursos	São entidades passivas utilizadas pelo software. Um Recurso, para ser válido, deve aparecer em pelo menos um dos episódios.
Episódios	Sentenças seqüenciais que correspondem a ações e decisões com participação dos atores e utilização de recursos. Essas sentenças devem aparecer em seqüência cronológica e serem sempre o mais simples possível.
Restrições	São aspectos não funcionais que qualificam/restringem o correto funcionamento do software. Estes aspectos podem estar relacionados ao contexto, recursos ou episódios.
Exceções	Situações que impedem que o objetivo do cenário seja alcançado. O tratamento para tais situações deve ser descrito.

Tabela 2 – Representação de cenários adotada.

Na Figura 21 podemos ver, a titulo de exemplo, a descrição do cenário “Recuperar senha do usuário”, de acordo com a representação de cenários adotada pelo método. Este cenário descreve a situação onde o usuário solicita a recuperação de sua senha de acesso ao sistema. Como descrito na subseção 2.2, podemos aliar as técnicas de cenários e LAL. Um exemplo disto pode ser visto nesta mesma figura, onde as palavras sublinhadas fazem parte do LAL do domínio do C&L, construído no passo anterior.

Título: RECUPERAR SENHA DO USUÁRIO
Objetivo: Permitir que o usuário recupere sua senha. Ao solicitar recuperação de sua senha o usuário preencherá um formulário com seu login e e-mail. Após a verificação dos dados o sistema irá substituir a senha antiga por uma nova senha randômica e enviá-la para o e-mail do usuário.
Contexto: Usuário deve ter sido previamente cadastrado no sistema
Recursos: login e e-mail do usuário.
Atores: usuário e sistema.
Episódios:
1- Usuário clica no atalho "Esqueceu a senha?" na página inicial do sistema.
2- O sistema exibe um formulário contendo os campos e-mail e login para o usuário preencher.
3- O usuário submete o formulário preenchido. **Restrição:** Todos os campos devem ser preenchidos.
4- O sistema verifica se o login e e-mail informado são iguais aos do cadastro do usuário.
5- Sistema gera uma nova senha randômica e a envia para o e-mail do usuário.
6- Sistema exibe um aviso informando que a senha foi alterada e que o usuário deve verificar o e-mail informado para obter a nova senha.
Exceção:
O login e e-mail informado pelo usuário não estão corretos. O sistema informará ao usuário que o login e a senha não estão corretos e permitirá que ele tente mais duas vezes. Caso erre mais duas vezes o usuário terá que esperar por 20 minutos para tentar novamente.

Figura 21 – Exemplo de descrição de uma situação do sistema.

4.1.3. Integração dos cenários

Mesmo diante de sistemas de médio porte, a quantidade de cenários que o descrevem pode ser grande, chegando às centenas. Com isso, se os engenheiros de requisitos se prenderem muito aos detalhes podem acabar perdendo a visão global do sistema. Para evitar que isto aconteça, propomos a integração de cenários. Nossa idéia de integração é baseada no processo descrito em [Leite00] com algumas modificações, para atender as particularidades de nosso método.

A integração é feita através da criação de um novo cenário, o cenário de integração. Este cenário é uma descrição artificial com o único propósito de facilitar o entendimento de um conjunto de cenários. Sua idéia principal é estabelecer uma relação entre os cenários dispersos dando uma visão global do sistema, e ainda assim preservar o uso da linguagem natural adotada nos cenários. Para a construção do cenário de integração utilizaremos um processo que tem como base a integração por generalização. A idéia deste processo é relacionar um grupo de cenários através da criação de um novo cenário, criado artificialmente, que descreva as mesmas situações que os cenários pertencentes

ao grupo, com um grau de abstração maior. O processo proposto é composto de três passos (Figura 22): Formar grupos de cenários, identificar cenários raiz e criar cenário integrador. A seguir detalharemos cada um destes passos.

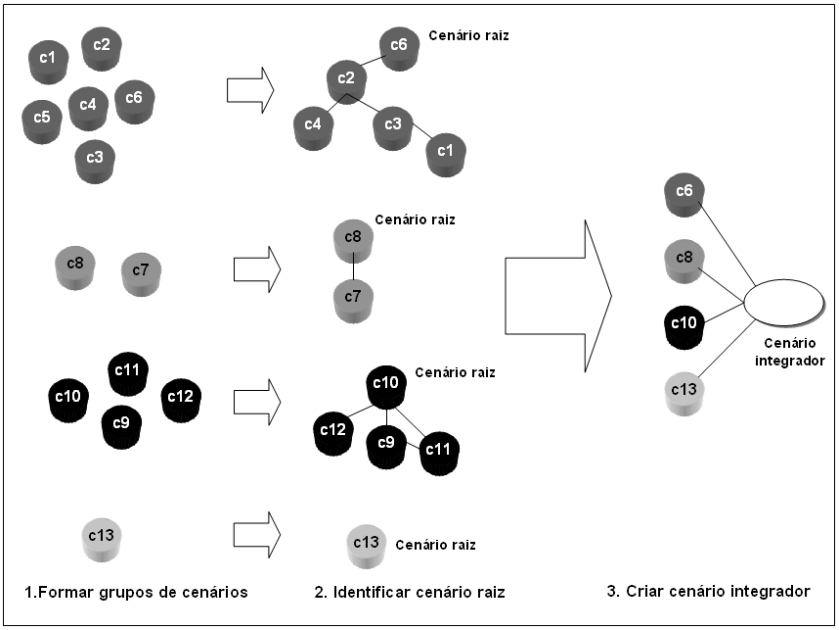
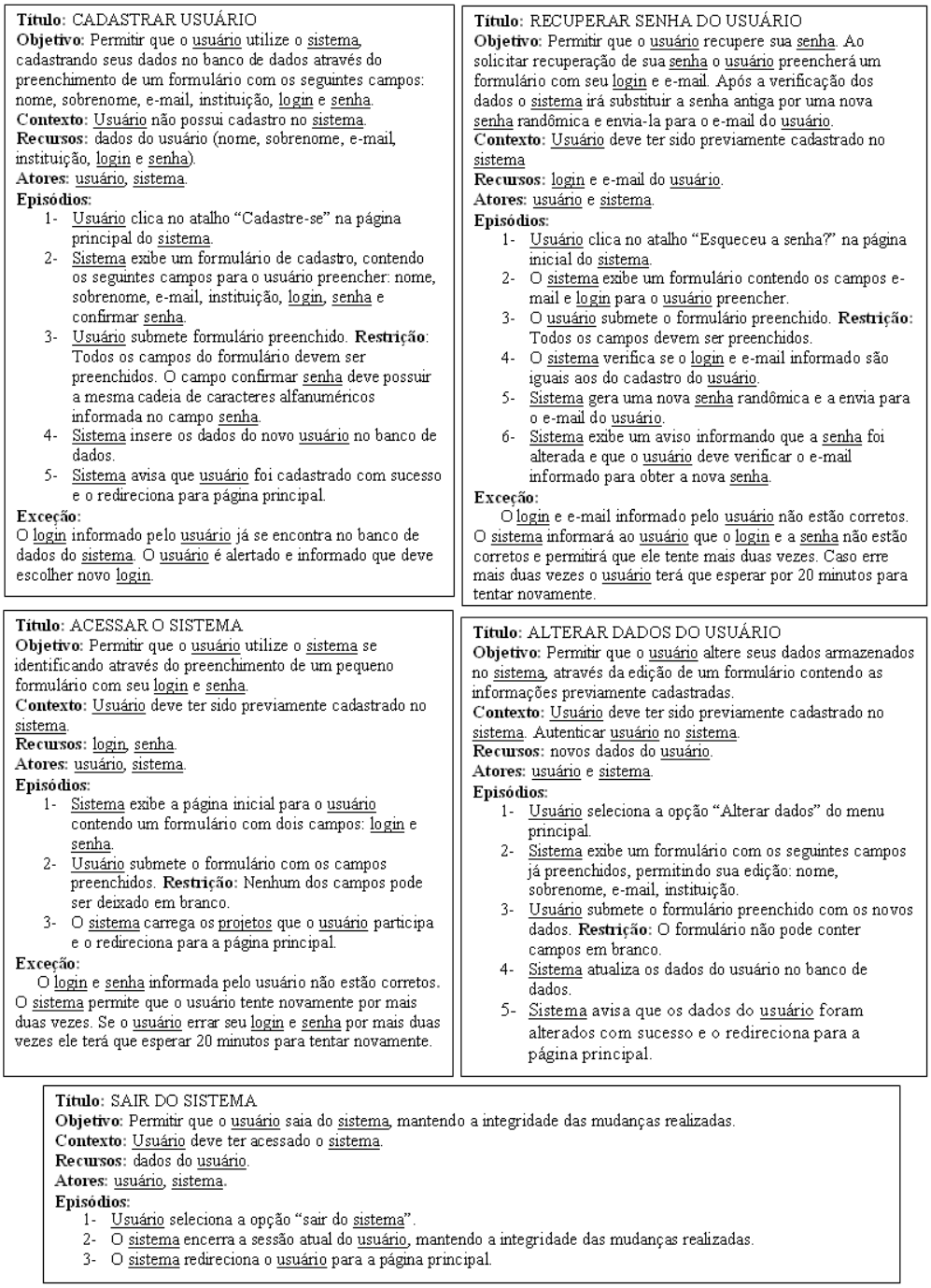


Figura 22 – Processo de integração de cenários.

4.1.3.1. Divisão dos cenários em grupos

A formação dos grupos de cenários deve ser feita considerando uma visão modular do sistema. Cenários que modelem funcionalidades comuns do sistema devem ser colocados no mesmo grupo. Um cenário que não se encaixar em nenhum outro grupo pode ser considerado um grupo por si só. Um exemplo da formação de um grupo pode ser visto na Figura 23. Neste exemplo todos os cenários estão relacionados, pois manipulam informações sobre o usuário.

.



Título: CADASTRAR USUÁRIO
Objetivo: Permitir que o usuário utilize o sistema, cadastrando seus dados no banco de dados através do preenchimento de um formulário com os seguintes campos: nome, sobrenome, e-mail, instituição, login e senha.
Contexto: Usuário não possui cadastro no sistema.
Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login e senha).
Atores: usuário, sistema.
Episódios:
1- Usuário clica no atalho "Cadastre-se" na página principal do sistema.
2- Sistema exibe um formulário de cadastro, contendo os seguintes campos para o usuário preencher: nome, sobrenome, e-mail, instituição, login, senha e confirmar senha.
3- Usuário submete formulário preenchido. **Restrição**: Todos os campos do formulário devem ser preenchidos. O campo confirmar senha deve possuir a mesma cadeia de caracteres alfanuméricos informada no campo senha.
4- Sistema insere os dados do novo usuário no banco de dados.
5- Sistema avisa que usuário foi cadastrado com sucesso e o redireciona para página principal.
Exceção:
O login informado pelo usuário já se encontra no banco de dados do sistema. O usuário é alertado e informado que deve escolher novo login.

Título: RECUPERAR SENHA DO USUÁRIO
Objetivo: Permitir que o usuário recupere sua senha. Ao solicitar recuperação de sua senha o usuário preencherá um formulário com seu login e e-mail. Após a verificação dos dados o sistema irá substituir a senha antiga por uma nova senha randômica e envia-la para o e-mail do usuário.
Contexto: Usuário deve ter sido previamente cadastrado no sistema
Recursos: login e e-mail do usuário.
Atores: usuário e sistema.
Episódios:
1- Usuário clica no atalho "Esqueceu a senha?" na página inicial do sistema.
2- O sistema exibe um formulário contendo os campos e-mail e login para o usuário preencher.
3- O usuário submete o formulário preenchido. **Restrição:** Todos os campos devem ser preenchidos.
4- O sistema verifica se o login e e-mail informado são iguais aos do cadastro do usuário.
5- Sistema gera uma nova senha randômica e a envia para o e-mail do usuário.
6- Sistema exibe um aviso informando que a senha foi alterada e que o usuário deve verificar o e-mail informado para obter a nova senha.
Exceção:
O login e e-mail informado pelo usuário não estão corretos. O sistema informará ao usuário que o login e a senha não estão corretos e permitirá que ele tente mais duas vezes. Caso erre mais duas vezes o usuário terá que esperar por 20 minutos para tentar novamente.

Título: ACESSAR O SISTEMA
Objetivo: Permitir que o usuário utilize o sistema se identificando através do preenchimento de um pequeno formulário com seu login e senha.
Contexto: Usuário deve ter sido previamente cadastrado no sistema.
Recursos: login, senha.
Atores: usuário, sistema.
Episódios:
1- Sistema exibe a página inicial para o usuário contendo um formulário com dois campos: login e senha.
2- Usuário submete o formulário com os campos preenchidos. **Restrição:** Nenhum dos campos pode ser deixado em branco.
3- O sistema carrega os projetos que o usuário participa e o redireciona para a página principal.
Exceção:
O login e senha informada pelo usuário não estão corretos. O sistema permite que o usuário tente novamente por mais duas vezes. Se o usuário errar seu login e senha por mais duas vezes ele terá que esperar 20 minutos para tentar novamente.

Título: ALTERAR DADOS DO USUÁRIO
Objetivo: Permitir que o usuário altere seus dados armazenados no sistema, através da edição de um formulário contendo as informações previamente cadastradas.
Contexto: Usuário deve ter sido previamente cadastrado no sistema. Autenticar usuário no sistema.
Recursos: novos dados do usuário.
Atores: usuário e sistema.
Episódios:
1- Usuário seleciona a opção "Alterar dados" do menu principal.
2- Sistema exibe um formulário com os seguintes campos já preenchidos, permitindo sua edição: nome, sobrenome, e-mail, instituição.
3- Usuário submete o formulário preenchido com os novos dados. **Restrição:** O formulário não pode conter campos em branco.
4- Sistema atualiza os dados do usuário no banco de dados.
5- Sistema avisa que os dados do usuário foram alterados com sucesso e o redireciona para a página principal.

Título: SAIR DO SISTEMA
Objetivo: Permitir que o usuário saia do sistema, mantendo a integridade das mudanças realizadas.
Contexto: Usuário deve ter acessado o sistema.
Recursos: dados do usuário.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "sair do sistema".
2- O sistema encerra a sessão atual do usuário, mantendo a integridade das mudanças realizadas.
3- O sistema redireciona o usuário para a página principal.

Figura 23 – Exemplo de um grupo de cenários.

4.1.3.2. Identificação dos cenários raiz

O primeiro passo para a identificação do cenário raiz de cada grupo é o mapeamento dos relacionamentos existentes entre os cenários que compõem o grupo. Estes relacionamentos podem ser de quatro tipos, como visto no capítulo

2: pré-condição, subcenário, exceção ou restrição. O mapeamento dos relacionamentos é baseado nas comparações entre pré-condições, recursos ou restrições de um cenário e título, objetivo ou episódios de outro. Se um cenário utiliza em um dos seus episódios outro cenário, então dizemos que este cenário é subcenário do primeiro. A seguir citamos as comparações que devem ser feitas para detectar os relacionamentos entre cenários:

- Contexto de um cenário e título dos outros cenários;
- Recursos de um cenário e título dos outros cenários;
- Recursos de um cenário e objetivo dos outros cenários;
- Recurso de um cenário e episódios dos outros cenários;
- Episódios de um cenário e título dos outros cenários;
- Exceção de um cenário e título dos outros cenários.

Podemos utilizar os relacionamentos identificados para estabelecer uma ordem entre os cenários do grupo. Por exemplo, se através da execução de um cenário alcançamos um estado que aparece como pré-condição em outro então há um relacionamento e também uma ordem entre estes dois cenários. O mesmo pode acontecer com um recurso, que é produzido por um cenário e utilizado por outro. Além disso, restrições que aparecem em episódios e recursos podem ser satisfeitas por outro cenário, estabelecendo uma ordem entre eles. As restrições presentes em um cenário nos fornecem pistas sobre a ordem sempre que apontam para a necessidade de algum recurso ou para condições que devem ser previamente satisfeitas.

Determinada a ordem entre os cenários, podemos identificar o cenário raiz do grupo. O cenário raiz é o cenário que não depende de nenhum outro cenário do grupo para sua correta execução. As pré-condições, recursos e restrições dos grupos não são obtidas apenas do cenário raiz, e sim de todos os cenários do grupo. Para estabelecer as pré-condições do grupo devemos reunir todas as pré-condições dos cenários que pertencem ao grupo e eliminar as que são satisfeitas pelos cenários do próprio grupo. Os recursos e restrições são obtidos da mesma maneira. Na Figura 24 podemos ver o cenário raiz identificado do grupo de cenários mostrado na Figura 23. Neste caso o cenário “cadastrar usuário” é pré-condição para os cenários “acessar o sistema” e “lembrar senha do usuário”. O cenário “acessar sistema”, por sua vez, é pré-condição para os cenários “alterar dados do usuário” e “sair do sistema”. Desta forma, o único cenário que não possui como pré-condição um cenário deste grupo é o “cadastrar usuário”, portanto este é o cenário raiz.

Título: CADASTRAR USUÁRIO
Objetivo: Permitir que o usuário utilize o sistema, cadastrando seus dados no banco de dados através do preenchimento de um formulário com os seguintes campos: nome, sobrenome, e-mail, instituição, login e senha.
Contexto: Usuário não possui cadastro no sistema.
Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login e senha).
Atores: usuário, sistema.
Episódios:
1- Usuário clica no atalho "Cadastre-se" na página principal do sistema.
2- Sistema exibe um formulário de cadastro, contendo os seguintes campos para o usuário preencher: nome, sobrenome, e-mail, instituição, login, senha e confirmar senha.
3- Usuário submete formulário preenchido. **Restrição**: Todos os campos do formulário devem ser preenchidos. O campo confirmar senha deve possuir a mesma cadeia de caracteres alfanuméricos informada no campo senha.
4- Sistema insere os dados do novo usuário no banco de dados.
5- Sistema avisa que usuário foi cadastrado com sucesso e o redireciona para página principal.
Exceção:
O login informado pelo usuário já se encontra no banco de dados do sistema.
O usuário é alertado e informado que deve escolher novo login.

Figura 24 – Exemplo de cenário raiz.

4.1.3.3. Criar cenário integrador

O cenário integrador é um cenário artificialmente criado apenas para estabelecer uma relação entre os cenários raiz, permitindo uma visão global do sistema. Começaremos a construção deste cenário pelos seus episódios. Para isto precisaremos ordenar os cenários raiz identificados, da mesma forma que ordenamos os cenários de cada grupo. Os cenários raiz devem aparecer nos episódios do cenário integrador na ordem obtida. O objetivo de se fazer isto é dar uma idéia das funcionalidades oferecidas pelo sistema e a ordem em que elas ocorrem através dos episódios do cenário integrador. O título, objetivo e contexto devem seguir o modelo de cenários apresentado anteriormente. Como este é um cenário criado artificialmente não há necessidade de descrevermos os recursos e os atores, conforme [Leite 00]. Na Figura 25 podemos ver um exemplo de cenário integrador. Neste exemplo, o caractere "#" delimita um grupo de episódios não seqüenciais e os episódios entre colchetes são opcionais.

Título: Sistema C&L
Objetivo: Permitir a colaboração entre diversos usuários na elaboração de projetos utilizando cenários e símbolos do léxico, fornecendo meios para facilitar a integração com outras ferramentas.
Contexto: - Usuário deseja elicitar a linguagem e as situações de um domínio de aplicação específico.
Recursos: -
Atores: -
Episódios:

1- Se o usuário ainda não possuir cadastro então CADASTRAR USUÁRIO, senão ACESSAR SISTEMA.
2- Usuário pode CADASTRAR PROJETO ou SELECIONAR PROJETO se já houver algum cadastrado.
3- # [Usuário pode INCLUIR COLABORADOR NO PROJETO]
4- [Usuário pode CADASTRAR SÍMBOLO DO LÉXICO no projeto.]
5- [Usuário pode CADASTRAR CENÁRIO no projeto.]
6- Usuário pode SAIR DO SISTEMA. #

Figura 25 – Exemplo de cenário integrador.

4.1.4. Refinamento dos cenários

Para que o uso de cenários realmente ajude, tanto no desenvolvimento do software quanto na sua documentação, é necessário que o método proposto se adapte a arquitetura definida para o sistema. Nesta dissertação nos restringiremos a sistemas que podem ser descritos segundo o *framework* MVC. Descreveremos a seguir como derivar os cenários pertencentes às três camadas do *framework*, aqui identificadas como camada M(modelo), camada C(controle) e camada V (visão), a partir dos cenários originais produzidos no primeiro passo (Figura 26).

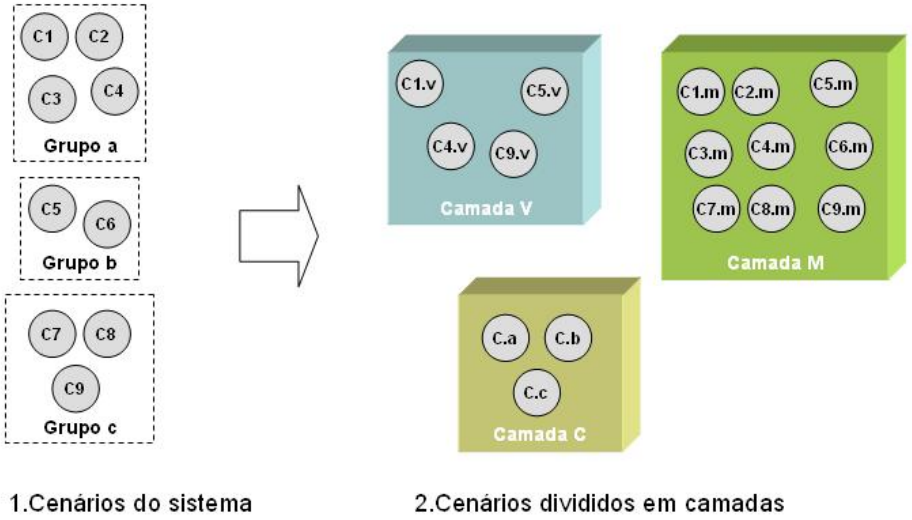


Figura 26 – Processo de refinamento dos cenários em camadas.

4.1.4.1. Divisão dos cenários em camadas

Nesta seção iremos detalhar como é feito o refinamento dos cenários em camadas. Cada camada possuirá uma heurística diferente para construção de seus cenários. Estas heurísticas serão detalhadas para cada uma das entidades que compõem o cenário.

4.1.4.1.1. Camada de visão

A maioria dos cenários que possuem como um de seus atores o usuário devem possuir uma interface de interação com ele. Para um maior detalhamento desta interface devemos separá-la do cenário original, gerado na subseção 4.1.2, e criar um novo cenário somente para descrevê-la. Na Tabela 3 descrevemos as heurísticas gerais para construção dos cenários da camada de visão, detalhando-as para cada elemento do cenário.

Título	Identificador da interface que está sendo descrita. Deve ser único
Objetivo	Descrição da finalidade da interface. A finalidade de uma interface pode ser, por exemplo, exibir dados estáticos para o usuário a título de informação ou pode ser exibir um formulário para ser preenchido.

Contexto	Descrição de como a interface descrita pode ser acessada. Através de uma URL ou de um elo em outra interface, por exemplo.
Atores	Identificar os tipos de usuários que podem acessar a interface. Um sistema pode ter diversos níveis de usuário, como administrador e usuário comum, por exemplo. E a interface pode ser acessada só por determinados níveis.
Recursos	Identificar recursos externos utilizados pela interface, como por exemplo, *javascript*, *css*, *applet* Java, animação em *flash,* etc.
Episódios	Cada episódio deve descrever em detalhes um componente da interface. Por exemplo, uma interface pode possuir diversos tipos de menu, que devem ser descritos detalhadamente, cada um por um episódio.
Restrições	Identificar as restrições presentes no uso da interface. Por exemplo, para poder preencher determinado campo de um formulário devo antes ter preenchido um determinado campo em um outro formulário.
Exceção	Detalhar comportamentos excepcionais da interface e seu tratamento. Por exemplo, ao retornar para uma página através do botão voltar do navegador a página não carregará corretamente.

Tabela 3 – Heurísticas para construção de um cenário da camada de visão.

A Figura 27 apresenta um exemplo de cenário da camada de visão, extraído do cenário “cadastrar usuário”. Este cenário descreve uma página contendo um formulário de cadastro de usuário. Podemos observar que a comunicação entre a camada de visão e a camada de controle é realizada no episódio 8, utilizando o subcenário “controle usuário”.

Titulo: EXIBIR PÁGINA DE CADASTRO DE NOVO USUÁRIO.
Objetivo: Permitir que o usuário informe seus dados pessoais para cadastro no sistema. Para tanto, a página exibida conterá um formulário com os seguintes campos: nome, sobrenome, e-mail, instituição, login, senha e confirmar senha.
Contexto: Pode ser acessado através do elo "Cadastrar-se" na página principal do sistema.
Atores: usuário ainda não cadastrado no sistema.
Recursos: scripts.js e estilo.css
Episódios:
1- Exibir campo Nome.
2- Exibir campo Sobrenome.
3- Exibir campo E-mail.
4- Exibir campo Instituição.
5- Exibir campo Login.
6- Exibir campo Senha.
7- Exibir campo Confirmar senha.
8- Ao clicar no botão cadastrar o sistema será redirecionado para CONTROLE USUARIO. Restrição: Antes de submeter o formulário VERIFICAR FOMULÁRIO DE CADASTRO DO USUÁRIO.

Figura 27 – Exemplo de cenário da camada de visão.

4.1.4.1.2. Camada de modelo

Após a separação da parte relacionada à interface com o usuário dos cenários iniciais, produzidos na subseção 4.1.2, a parte restante é referente à camada de modelo do sistema. Sem a presença da interface com o usuário podemos focar mais nas funcionalidades do sistema e aumentar seu nível de detalhamento. As heurísticas para isto podem ser encontradas na Tabela 4, detalhadas para cada elemento do cenário.

Título	Identificador do cenário da camada de modelo. Deve ser único.
Objetivo	Descrever qual a finalidade da funcionalidade modelada pelo usuário.
Contexto	Identificar as pré-condições necessárias para a execução do cenário. Por exemplo, se é necessária a execução de um cenário da camada de visão (preenchimento de um formulário).
Atores	Identificar os atores envolvidos no cenário. Note que o usuário não é mais um ator válido para este cenário. Neste caso os atores podem ser outras partes do sistema ou componentes externos.

Recursos	Identificar as informações que devem estar disponíveis durante a execução do cenário. Se o cenário utilizar o banco de dados do sistema ele deve aparecer na descrição de seus recursos.
Episódios	Descrição em ordem cronológica de ações e decisões com a participação dos atores e o uso dos recursos, que se executadas corretamente levam a satisfação do objetivo do cenário.
Restrições	Identificar requisitos não funcionais associados a recursos, contexto e episódios que possam interferir na correta execução do cenário.
Exceção	Identificar comportamentos excepcionais que possam impedir a correta execução do cenário e detalhar o seu tratamento.

Tabela 4 – Heurísticas para construção de um cenário da camada de modelo.

Na Figura 28 podemos observar o cenário “cadastrar usuário” um exemplo de cenário da camada de modelo, resultante da retirada da interface e do enfoque na descrição mais detalhada da funcionalidade modelada.

Título: CADASTRAR USUÁRIO
Objetivo: Cadastrar usuário no sistema através da inserção de seus dados no banco de dados.
Contexto: EXIBIR FORMULÁRIO DE CADASTRO DE USUÁRIO.
Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login e senha), banco de dados.
Atores: CONTROLE USUÁRIO.
Episódios:
1- Receber os dados do usuário repassados pela camada de CONTROLE USUÁRIO.
2- Conectar ao banco de dados. **Restrição:** Banco de dados deve estar disponível.
3- Inserir dados do usuário no banco de dados.
4- Desconectar do banco de dados.
5- Informar a camada de CONTROLE USUÁRIO que os dados foram inseridos com sucesso no banco de dados.
Exceção:
O login informado pelo usuário já se encontra no banco de dados do sistema. Informar a camada de CONTROLE USUÁRIO que os dados não foram inseridos porque o login já existe.

Figura 28 – Exemplo de cenário da camada de modelo.

4.1.4.1.3. Camada de controle

Para cada grupo de cenários formado na subseção 4.1.3.1, um novo cenário deverá ser criado. Estes cenários representarão a camada de controle, e serão responsáveis pela intermediação entre os cenários que compõem a camada de visão e os que compõem a camada de modelo de cada grupo. A heurística para criação destes cenários pode ser vista na Tabela 5.

Título	Identificador da camada de controle que está sendo descrita. Deve ser único
Objetivo	O objetivo da camada de controle é fixo. É sempre intermediar a comunicação entre a camada de visão e a camada de modelo.
Contexto	Descrever a partir de quais páginas da camada de visão a camada de controle pode ser acionada.
Atores	Identificar com quais outros cenários este se relaciona.
Recursos	Identificar os dados que são recebidos e repassados pela camada.
Episódios	Identificar cada tipo diferente de requisição que pode ser recebida pela camada e como ela é tratada e repassada. Explicar como é feita a validação dos dados recebidos pela camada.
Restrições	Identificar as restrições existentes para o repasse das requisições. Por exemplo, uma requisição deve possuir um formato específico de dado para que possa ser repassada.
Exceção	Detalhar comportamentos excepcionais da camada de controle e seu tratamento. Por exemplo, no caso de uma requisição que não pode ser repassada.

Tabela 5 – Heurísticas para construção de um cenário da camada de controle.

Na Figura 29 vemos um exemplo de cenário na camada de controle. A intermediação proporcionada pela camada pode ser vista nos episódios 1 e 2. No episódio 1 a camada de controle recebe uma requisição para cadastrar um

usuário. Esta requisição é dividida em duas partes. Primeiro é enviada uma requisição a camada de modelo, através do cenário “checar login”, para saber se o login esta disponível. Se o login estiver disponível a camada de controle envia a requisição para cadastrar o usuário, através do cenário “cadastrar usuário”. Dependendo do resultado comunicado pela camada de modelo sobre a execução deste cenário, a camada de controle irá repassar a requisição para o cenário “exibir página inicial” ou para o “exibir página de erro” da camada de visão.

Titulo: CONTROLE USUÁRIO.
Objetivo: Intermediar a comunicação entre a camada de visão e a camada de modelo.
Contexto: novo_usuario.html, lembrar_senha.html, alterar_dados_usuario.lp, index.html, erro.lp, página_principal.lp, sucesso.lp.
Atores: camada de modelo e camada de visão.
Recursos: dados do usuário, informações para as páginas de erro, informações para as páginas de sucesso.
Episódios:

1- Se a requisição recebida for para cadastrar usuário então CHECAR LOGIN. Se login disponível então CADASTRAR USUÁRIO, senão EXIBIR PÁGINA DE ERRO.
2- Se os dados foram cadastrados corretamente então EXIBIR PÁGINA PRINCIPAL DO SISTEMA, senão EXIBIR PÁGINA DE ERRO.
3- Se a requisição recebida for para autenticar o usuário, então CHECAR DADOS DO USUÁRIO. Se os dados estiverem corretos então criar sessão para identificar o usuário e EXIBIR PÁGINA PRINCIPAL DO SISTEMA, senão EXIBIR PÁGINA DE ERRO.
4- Se a requisição recebida for para alterar os dados do usuário, então ALTERAR DADOS DO USUÁRIO. Se os dados forem alterados com sucesso então EXIBIR PAGINA DE SUCESSO, senão EXIBIR PAGINA DE ERRO.
5- Se a requisição recebida for para lembrar senha, então VERIFICAR LOGIN E E-MAIL DO USUÁRIO. Se login e e-mail estiverem corretos então LEMBRAR SENHA DO USUÁRIO, senão EXIBIR PÁGINA DE ERRO.
6- Se a nova senha for enviada com sucesso para o usuário então EXIBIR PÁGINA DE SUCESSO, senão EXIBIR PÁGINA DE ERRO.
7- Se a requisição recebida for para sair do sistema, então a sessão referente ao usuário é excluída e EXIBIR PAGINA INICIAL.

Figura 29 – Exemplo de cenário da camada de controle.

4.1.4.2. Operacionalização dos cenários

De maneira geral, o processo de operacionalização dos cenários de um sistema deve ser acompanhado de seu refinamento. O desenvolvedor deve utilizar os episódios construídos anteriormente apenas como um guia para implementação. Após a construção do código fonte o cenário deve ser refinado

para que descreva, de forma adequada e detalhada, o que foi implementado, incluindo as modificações realizadas. A seguir descreveremos as particularidades da operacionalização dos cenários de cada camada.

4.1.4.2.1. Operacionalização da camada de visão

Para cada um dos cenários que descreve uma interface, deve ser criado um novo arquivo. Esteve arquivo pode possuir a extensão html, caso apenas a linguagem de marcação HTML [HTML 09] seja utilizada na sua construção, ou pode possuir uma extensão própria da linguagem de implementação utilizada, caso utilize trechos desta linguagem no interior do arquivo. Os elos existentes entre os arquivos que compõem a interface devem ser mapeados através de relacionamentos entre os cenários que os descrevem.

Aliada ao HTML, a linguagem de *script JavaScript*[JavaScript09] também é muito utilizada no desenvolvimento das interfaces para sistemas Web, pois permite uma interação mais dinâmica com o usuário. Da mesma forma que o restante do sistema, a parte codificada em *JavaScript* também deve ser documentada através de cenários. Para fazer esta documentação, o código *JavaScript* utilizado deve ser separado em funções e colocado em um arquivo com extensão js, específico para este fim. Para cada função *JavaScript* deve ser criado um cenário, a fim de descrevê-la. Este cenário deve possuir a estrutura mostrada na Tabela 6. A ligação entre os cenários da camada de visão e os que descrevem as funções *JavaScript*, deve ser feita através de um dos relacionamentos possíveis entre cenários (subcenário, exceção, restrição ou pré-condição).

Título	Identificador da função *JavaScript*
Objetivo	Descrição da finalidade da função.
Contexto	Em que ocasiões a função é chamada. Quais outras funções a chamam.
Atores	Outras funções *JavaScript* utilizadas pela que está sendo descrita ou o próprio usuário, caso interaja com a função.
Recursos	Dados necessários para a execução da função (parâmetros, dados recuperados de páginas HTML através de objetos *document*)

Episódios	Devem descrever detalhadamente cada trecho de código da função.
Restrições	Aspectos não funcionais que podem atrapalhar o funcionamento da função. Podem aparecer no contexto, recursos ou episódios.
Exceção	Identificar comportamentos excepcionais que podem ocorrer na função e seu respectivo tratamento.

Tabela 6 – Heurísticas para descrever funções *JavaScript*

A Figura 30 apresenta um trecho de código JavaScript documentado com o uso de cenários. Este trecho corresponde à função "verificar_formulario_usuario", responsável por validar o formulário contendo os dados do usuário antes que ele seja submetido para a camada de controle. Na Figura 31 podemos ver um trecho do cenário "Exibir página de cadastro de novo usuário" implementado.

```
/*
@Título: Validar formulário de cadastro de usuário.
@Objetivo: Assegurar que o formulário de cadastro de usuário seja preenchido
da maneira correta.
@Contexto: O formulário de cdastro é preenchido e submetido pelo usuário.
@Atores: cadastro_usuario.html
@Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login,
senha e confirmar senha)
*/
function verificar_formulario_usuario()
{
    formulario = document.frmCadastro;

/*
@Episódio 1: Verificar se o campo nome está vazio ou
contém apenas espaços em branco. Se o campo estiver
vazio, emite alerta para o usuário e põe o foco do
cursor nele.
*/
    if (formulario.nome.value.ltrim() == ""){
       alert("O campo " + formulario.nome.name + " deve ser preenchido!");
       formulario.nome.focus();
       return false;
    }

/*
@Episódio 2: Verificar se o campo sobrenome está vazio
ou se contém apenas espaços em branco. Se o campo
estiver vazio emite um alerta para o usuário e põe
o foco do cursor nele.
*/
    if (formulario.sobrenome.value.ltrim() == ""){
       alert("O campo " + formulario.sobrenome.name + " deve ser preenchido!");
       formulario.sobrenome.focus();
       return false;
    }
```

Figura 30 – Trecho de uma função *JavaScript* documentada através de um cenário.

```
<!--
@Titulo: Exibir página de cadastro de novo usuario.
@Objetivo: Permitir que o usuário informe seus dados pessoais para cadastro
no sistema. Para tanto, a página exibida conterá um formulário com os seguintes
campos: nome, sobrenome, e-mail, instituição, login, senha e confirmar senha.
@Contexto: Usuário clica no botão "Cadastrar-se" na página principal do sistema.
@Atores: usuário ainda não cadastrado no sistema.
@Recursos: scripts.js e estilo.css
-->
    <head>
        <script src="js/scripts.js" type="text/javascript"></script>
        <link rel="stylesheet" type="text/css" href="css/estilo.css" />
        <title>Cadastro de Usuário</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    </head>
    <body>
        <div align=center bgcolor="#ffffff">
        <form name="frmCadastro" method="post" action="../controle/controle_usuario.lp"
            onSubmit="return verificar_formulario_usuario();">
        <table width=457>
            <tr>
                <td align="left"><span align="left" class="pes">
                    C&L - Cenários e Léxicos</span></td></tr>
        </table>
        <table class="tableestilo" cellpadding="20" >
        <tr>
        <td>
            <table cellpadding="5">
                <tr>
                    <td class="labelestilo" colspan="4" align="center">
                    <span class="titulo">Cadastro de novo usuário<br><br><br></span></td>
                    </tr>
                <tr>
<!--
@Episódio 1: Exibir campo Nome.
-->
                    <td class="labelestilo">Nome:</td><td colspan="3"><input
                        class="inputestilo" id="nome" name="nome" maxlength="60"
                        size="40" type="text"></td>
                </tr>
                <tr>
```

Figura 31 – Trecho de cenário da camada de visão operacionalizado

4.1.4.2.2. Operacionalização da camada de controle

Cada cenário da camada de controle dará origem a um novo arquivo. Este arquivo possuirá a extensão própria da linguagem utilizada. A implementação deste cenário será basicamente através de uma estrutura de seleção, que identificará a requisição recebida, determinará quais ações deverão ser executadas na camada modelo e depois encaminhará a resposta para a camada de visão. Na Figura 32 podemos ver um trecho do cenário “controle usuário” já implementado.

```
<?lua
--[[
@Titulo: Controle usuário.
@Objetivo: Intermediar a comunicação entre a camda de visão e controle do
módulo usuário.
@Contexto: cadastrar_usuario.html, alterar_dados.lp, lembrar_senha.html,
index.html, principal.lp.
@Atores: CHECAR LOGIN, CADASTRAR_USUARIO, CHECAR_USUARIO, LEMBRAR_SENHA.
@Recursos: dados do usuário.
]]--
--@Episódio 1: Importa o arquivo modelo_usuario.lua e habilita o uso
-- de sessões
    dofile("../modelo/modelo_usuario.lua")
    cgilua.enablesession();
?>
<html>
<head>
</head>
<body>
<?lua
    if (cgilua.POST.comando == "cadastrar") then

--@Episódio 2: Se o comando for cadastrar CHECAR LOGIN.
        local ha_usuario = checar_login_usuario(cgilua.POST.login);
        if(ha_usuario) then
            cgilua.put("Login já existente, por favor escolha outro login!");
        else

--@Episódio 3: Se o login não se encontra no banco de dados então
-- CADASTRAR_USUARIO.
            local id_usuario = cadastrar_usuario(cgilua.POST.nome,
            cgilua.POST.sobrenome, cgilua.POST.email, cgilua.POST.instituicao,
            cgilua.POST.login, cgilua.POST.senha);

--@Episódio 4: Cria uma sessão e salva o id do usuário nela.
            cgilua.enablesession();
```

Figura 32 – Trecho de cenário da camada de controle operacionalizado.

4.1.4.2.3. Operacionalização da camada de modelo

Os cenários da camada de modelo devem ser implementados através de funções. Uma função deve ser criada para cada um dos cenários que compõem a camada. Se um cenário descrever uma funcionalidade muito complexa, ele pode ser desmembrado em um ou mais cenários. Também podem ser criados novos cenários para descrever um comportamento comum a muitos cenários, reduzindo assim a redundância. Em ambos os casos existirá uma ligação entre os cenários originais e os novos. Esta ligação deve ser mapeada através de um dos relacionamentos possíveis entre cenários (subcenário, pré-condição, restrição, exceção). A Figura 33 apresenta o trecho de um cenário da camada de modelo implementado.

```
--[[
@Titulo: Cadastrar usuário.
@Objetivo: Cadastrar os dados informados pelo usuário.
@Contexto: Exibir página de cadastro de novo usuário.
@Atores: inserir_usuario_bd, selecionar_usuario_bd.
@Recursos: banco de dados, dados do usuário (nome, sobrenome, instituição,
login, senha).
]]--

function cadastrar_usuario (nome, sobrenome, email, instituicao, login, senha)

--@Episódio 1: Criptografa a senha fornecida no formulário do usuário.
    senha_criptografada = md5.sum(senha);

--@Episódio 2: INSERIR USUARIO NO BANCO DE DADOS.
    inserir_usuario_bd(nome, sobrenome, email, instituicao, login,
        senha_criptografada);

--@Episódio 3: SELECIONAR USUARIO NO BANCO DE DADOS.
    usuarios = selecionar_usuario_bd (login);

--@Episódio 4: Recupera o id do usuário que acabou de ser inserido e o retorna.
    for index, usuario in pairs(usuarios) do
        id_usuario = usuario["ID_USUARIO"];
        return id_usuario;
    end
end
```

Figura 33 – Cenário da camada modelo operacionalizado.

Um comportamento complexo que deve ser destacado dos cenários desta camada são os acessos ao banco de dados. Quando for necessário fazer um acesso ao banco de dados um novo cenário e, consequentemente, uma nova função devem ser criadas. A Tabela 7 apresenta as heurísticas gerais para descrição de cada um dos elementos destes novos cenários. Um exemplo de comportamento comum que pode ser destacado é a conexão e desconexão com o banco de dados, que são realizadas por todos os cenários que o acessam. Na Figura 34 podemos ver um trecho de cenário que faz acesso ao banco de dados implementado, onde esta separação foi feita.

Título	Identificador do cenário. Deve ser único.
Objetivo	Descrição da finalidade do cenário. Neste caso existem três opções: Inserir um novo dado no banco de dados e remover ou consultar um dado já existente.
Contexto	Identificar os cenários da camada de modelo que utilizam o cenário.
Atores	Identificar outras funções que sejam utilizadas pela descrita. Por exemplo, a função de conexão com o banco

	de dados.
Recursos	Dados que serão manipulados pela função. Estes dados podem ser provenientes de parâmetros da função ou de consultas ao banco de dados.
Episódios	Devem descrever detalhadamente cada trecho de código da função de acesso ao banco de dados.
Restrições	Aspectos não funcionais que podem atrapalhar o funcionamento da função. Podem aparecer no contexto, recursos ou episódios. Por exemplo, restrições de chave primária.
Exceção	Identificar comportamentos excepcionais que podem ocorrer na função e seu respectivo tratamento.

Tabela 7 – Heurísticas para criação de cenários que descrevem acessos ao banco de dados.

```
--[[
@Titulo: Inserir usuário no banco de dados.
@Objetivo:Inserir os dados do usuário no banco de dados do sistema.
@Contexto: Cadastrar usuário.
@Atores: Conectar ao banco de dados.
@Recursos: dados do usuário.
]]--
function inserir_usuario_bd (nome, sobrenome, email, instituicao, login, senha)

--@Episódio 1: CONECTAR AO BANCO DE DADOS.
    local conexao = conectar_bd();

--@Episódio 2: Montar query de inserção de usuário no banco de dados,
-- com os valores passados por parâmetro.
    local stmt = ("insert into usuario (nome, sobrenome, email, instituicao,"..
        "login, senha) values (\""..nome.."\",\""..sobrenome.."\",\""..email.."\",\""
        ..instituicao.."\",\""..login.."\",\""..senha.."\")");

--@Episódio 3: Executar a query para inserção de usuário no banco de dados.
    local cursor, erro = conexao:execute (stmt)

--@Episódio 4: Tratamento de eventuais erros que possam ocorrer com a conexão.
    if not cursor then
        error (erro.." SQL = [=["..stmt.."]=]")
    end
--@Episódio 5: Fechar a conexão com o banco de dados.
    conexao:close();
end
```

Figura 34 – Cenário que descreve acesso ao banco de dados implementado.

4.2. Mapeamento do espaço de nomes

Para produzir uma anotação auxiliar, propomos a aplicação da técnica LAL no mapeamento do espaço de nomes durante e após a escrita do código fonte. Esta proposta é baseada no trabalho apresentado em [Leite08].

Espaço de nomes é um conceito aplicado em diferentes áreas da informática. Sua principal idéia é proporcionar um contexto em que nomes são atribuídos a identificadores únicos. A gerência do espaço de nomes para evitar a proliferação de nomes é um problema que traz diferentes tipos de solução em diferentes contextos [Neuman89] [Achermann00]. Nosso interesse particular é a nível de código fonte. Nós entendemos que a equipe de engenheiros de software deve ter controle sobre o espaço de nomes do código que produzir independente da infra-estrutura utilizada. Para que este controle seja possível deve ser feita uma enumeração e descrição dos identificadores utilizados (cadeias de caracteres para nomear variáveis, funções, parâmetros, arquivos), mantendo um rastro para sua localização no código fonte.

Existem ferramentas no mercado que fazem a analise estática e dinâmica do código fonte [Semantic Designs]. Estas ferramentas irão nos fornecer listas de identificadores, elos e grafos. Nossa proposta é construir uma rede de conceitos rastreável que mapeia o espaço de nomes e, ao mesmo tempo, fornece definição e rastros (fluxogramas estáticos).

Propomos a utilização do LAL, pois acreditamos que esta é uma representação razoável para mapearmos e descrevermos os identificadores utilizados no código fonte de um software. Nossa proposta é utilizar o C&L para dar suporte ao mapeamento feito através do LAL. Para tanto, os elementos do espaço de nomes devem ser inseridos no software como símbolos do léxico, seguindo um modelo pré-definido. Feito isto, podemos utilizar todas as funcionalidades oferecidas pelo software para o rastreamento e visualização dos elementos do espaço de nomes. O rastreamento é obtido através dos *elos* gerados automaticamente pela ferramenta. Eles permitem a navegação entre os diferentes elementos que se relacionam. A visualização pode ser feita através da ferramenta de construção de grafos ou do XML formatado do projeto. O grafo gerado nos dará uma visão global de todos os relacionamentos entre todos os elementos do espaço de nomes e permite o foco em elementos específicos, facilitando a identificação de seus relacionamentos. O XML formatado fornecerá

uma lista de todos os elementos do espaço de nomes, permitindo que o usuário navegue entre eles através de *elos*.

Cada elemento do espaço de nomes (variáveis, funções e arquivos) deverá ser inserido no LAL de acordo com um modelo diferente. A seguir detalharemos cada um destes modelos.

4.2.1. Modelo para inclusão de variável

A Tabela 8 mostra o modelo a ser seguido para inclusão de uma variável no LAL. Na Figura 35 podemos ver um exemplo de variável descrita através deste modelo, podemos observar que o software gerou automaticamente *elos* para a função e o arquivo onde a variável foi declarada.

Nome	Nome da variável.
Noção	Identificar sua localização (em que arquivo, linha e função é declarada). Identificar seu escopo, tipo e descrever sua utilidade.
Classificação	Uma variável é sempre classificada como objeto.
Impactos	Identificar quais outros elementos utilizam a variável.

Tabela 8 – Descrição de uma variável através do LAL.

Nome	senha_criptografada
Noção	Localização: - Arquivo: modelo_usuario.lua, linha 13. - Função: cadastrar_usuario. Escopo: local. Tipo: string. Utilidade: Armazena a senha criptografada.
Classificação	objeto
Impacto(s)	É utilizada pela função cadastrar_usuario.
Sinônimo(s)	

Figura 35 – Exemplo de variável descrita através do LAL.

4.2.2. Modelo para inclusão de função

A Tabela 9 mostra o modelo a ser seguido para inclusão de uma função no LAL. Na Figura 36 podemos ver um exemplo de função descrita através deste modelo. Neste exemplo o software gerou elos para o arquivo onde a função foi declarada, para seus parâmetros, valor de retorno e para outras funções que são utilizadas por ela.

Nome	Nome da função
Noção	- Identificar sua localização (nome do arquivo e linha onde foi declarada). - Identificar seus parâmetros e valor de retorno. - Descrever sua utilidade.
Classificação	- Uma função é sempre classificada como verbo.
Impactos	- Identificar outros elementos que a utilizam. - Identificar as funções que ela utiliza.

Tabela 9 – Descrição de uma função através do LAL.

Nome	cadastrar_usuario
Noção	Localização: - Arquivo modelo_usuario.lua, linha 11. Parâmetros: nome, sobrenome, email, instituicao, login, senha. Valor de retorno: id_usuario. Utilidade: Cadastrar o usuário no sistema.
Classificação	sujeito
Impacto(s)	Utilizada pela camada de controle. (arquivo controle_usuario.lp) Utiliza as funções: - inserir_usuario_bd. - selecionar_usuario_bd.
Sinônimo(s)	

Figura 36 – Exemplo de função mapeada através do LAL.

4.2.3. Modelo para inclusão de arquivo

A Tabela 10 mostra o modelo a ser seguido para inclusão de um arquivo no LAL. Na Figura 37 podemos ver um exemplo de arquivo descrito através deste modelo. No exemplo mostrado, foram gerados *elos* para as funções declaradas dentro do arquivo.

Nome	Nome do arquivo
Noção	- Identificar a relação entre as funções definidas dentro do arquivo.
Classificação	- Um arquivo é sempre classificado como objeto.
Impactos	- Identificar as funções definidas dentro do arquivo.

Tabela 10 – Descrição de um arquivo através do LAL.

Nome	modelo_usuario.lua
Noção	Contém as funções relacionadas a camada modelo do módulo usuário.
Classificação	objeto
Impacto(s)	As seguintes funções são definidas dentro do arquivo: - cadastrar_usuario. - checar_login. - checar_usuario. - alterar_usuario. - lembrar_senha.
Sinônimo(s)	

Figura 37 – Exemplo de arquivo mapeado através do LAL.

O LAL deve ser único para cada aplicação, mas neste trabalho propomos a construção de dois, um para definir os símbolos específicos da aplicação e outro para mapear o espaço de nomes. Porém, podemos enxergar claramente a possibilidade de integração deles, mas este assunto não será abordado neste trabalho.

5
Re-arquitetura do software C&L

Neste capítulo apresentamos as tecnologias utilizadas na construção do novo C&L e a sua nova arquitetura. Também mostraremos, passo a passo, a aplicação do método proposto na sua construção.

5.1.
Arquitetura do novo C&L e tecnologias utilizadas

Escolhemos a linguagem Lua [Ierusalimschy03] para implementação da ferramenta. Esta escolha foi motivada por três motivos principais. O primeiro deles é que Lua provê um excelente suporte a programação funcional e acreditamos que o uso de funções se adequa mais a descrição por cenários proposta pelo nosso método. O Segundo motivo é que Lua é uma linguagem cada vez mais utilizada, e acreditamos que suas principais características (livre, rápida, extensível e pequena, dentre outras) sejam bastante úteis quando aplicadas ao desenvolvimento de sistemas Web. E por último, o C&L faz muitas comparações baseadas em expressões regulares para determinar o relacionamento entre elementos. Lua oferece um excelente suporte ao uso de expressões regulares e comparações entre *strings.*

Lua não foi projetada para desenvolvimento de sistema Web, por isso utilizamos a plataforma Kepler [Kepler09]. Esta plataforma disponibiliza uma série de módulos que facilitam o desenvolvimento de código Lua para Web. Dentre os módulos podemos destacar o CGILua que permite a criação de páginas dinâmicas para Web e a manipulação de dados provenientes de formulário, o LuaSQL que fornece uma interface simples para interação de Lua com alguns dos principais sistemas gerenciadores de banco de dados e, por último, o Xavante, que é um poderoso servidor Web HTTP 1.1, que usa uma arquitetura modular baseada em tratadores URI mapeados.

Para a construção da nova arquitetura adotamos o framework MVC, pois acreditamos que a divisão em camadas proporcionada por ele favorece a organização e entendimento do código fonte do software, proporcionando uma maior transparência a nível de código. Em nossa arquitura existe a divisão física

do sistema em camadas e a divisão lógica em módulos. Cada um dos módulos está distribuído pelas camadas de visão, controle e modelo, como mostra a Figura 38.

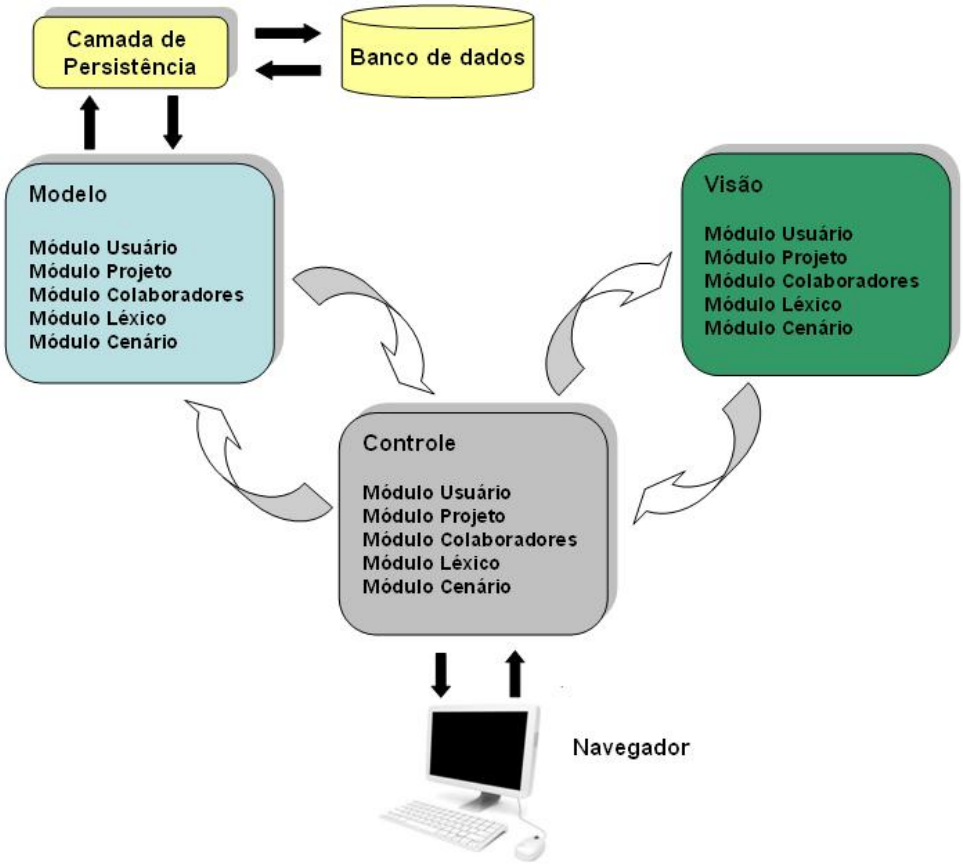


Figura 38 – Arquitetura do novo C&L.

Como ferramenta de apoio à aplicação do método proposto utilizamos o próprio C&L. No princípio, utilizamos a versão antiga para construir os cenários iniciais e o léxico ampliado da linguagem. Conforme o desenvolvimento do novo C&L foi avançando, pudemos migrar os cenários para a nova ferramenta. A partir daí, utilizamos o novo C&L para evoluir os cenários iniciais e os símbolos do léxico.

5.2. Aplicação do método na construção do novo C&L

Detalharemos a aplicação do método proposto, passo a passo, nas subseções a seguir.

5.2.1. Descrição das situações do sistema através de cenários

A seguir apresentamos a descrição das situações do software C&L com o uso do modelo de cenários apresentado anteriormente. Os termos sublinhados ou são referências a outros cenários ou a símbolos do léxico.

Título: CADASTRAR USUÁRIO

Objetivo: Cadastrar usuário no sistema através da inserção de seus dados no banco de dados.

Contexto: Usuário não possui cadastro no sistema.

Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login e senha), banco de dados.

Atores: camada de controle

Episódios:

1- Receber os dados do usuário repassados pela camada de controle.
2- Conectar ao banco de dados. **Restrição:** Banco de dados deve estar disponível.
3- Inserir dados do usuário no banco de dados.
4- Desconectar do banco de dados.
5- Informar a camada de controle que os dados foram inseridos com sucesso no banco de dados.

Exceção:

O login informado pelo usuário já se encontra no banco de dados do sistema. Informar a camada de controle que os dados não foram inseridos porque o login já existe.

Título: ALTERAR DADOS DO USUÁRIO

Objetivo: Permitir que o usuário altere seus dados armazenados no sistema, através da edição de um formulário contendo as informações previamente cadastradas.

Contexto: Usuário deve ter sido previamente cadastrado no sistema. AUTENTICAR USUÁRIO NO SISTEMA.

Recursos: novos dados do usuário.

Atores: usuário e sistema.

Episódios:

1- Sistema exibe um formulário com os seguintes campos já preenchidos, permitindo sua edição: nome, sobrenome, e-mail, instituição.
2- Usuário submete o formulário preenchido com os novos dados. **Restrição:** O formulário não pode conter campos em branco.
3- Sistema atualiza os dados do usuário no banco de dados.
4- Sistema avisa que os dados do usuário foram alterados com sucesso e o redireciona para a página principal.

Título: ACESSAR O SISTEMA

Objetivo: Permitir que o usuário utilize o sistema se identificando através do preenchimento de um pequeno formulário com seu login e senha.

Contexto: Usuário deve ter sido previamente cadastrado no sistema.

Recursos: login, senha.

Atores: usuário, sistema.

Episódios:

1- Sistema exibe para o usuário um formulário contendo dois campos: login e senha.
2- Usuário submete o formulário com os campos preenchidos. **Restrição:** Nenhum dos campos pode ser deixado em branco.
3- O sistema carrega os projetos que o usuário participa e o redireciona para a página principal.

Exceção:

O login e senha informada pelo usuário não estão corretos**.** O sistema permite que o usuário tente novamente por mais duas vezes. Se o usuário errar seu login e senha por mais duas vezes ele terá que esperar 20 minutos para tentar novamente.

Título: RECUPERAR SENHA DO USUÁRIO

Objetivo: Permitir que o usuário recupere sua senha. Ao solicitar recuperação de sua senha o usuário preencherá um formulário com seu login e e-mail. Após a verificação dos dados o sistema irá substituir a senha antiga por uma nova senha randômica e enviá-la para o e-mail do usuário.

Contexto: Usuário deve ter sido previamente cadastrado no sistema

Recursos: login e e-mail do usuário.

Atores: usuário e sistema.

Episódios:

1- O sistema exibe um formulário contendo os campos e-mail e login para o usuário preencher.
2- O usuário submete o formulário preenchido. **Restrição:** Todos os campos devem ser preenchidos.
3- O sistema verifica se o login e e-mail informado são iguais aos do cadastro do usuário.
4- Sistema gera uma nova senha randômica e a envia para o e-mail do usuário.
5- Sistema exibe um aviso informando que a senha foi alterada e que o usuário deve verificar o e-mail informado para obter a nova senha.

Exceção:

O login e e-mail informado pelo usuário não estão corretos. O sistema informará ao usuário que o login e o e-mail não estão corretos e permitirá que ele tente mais duas vezes. Caso erre mais duas vezes o usuário terá que esperar por 20 minutos para tentar novamente.

Título: SAIR DO SISTEMA

Objetivo: Permitir que o usuário saia do sistema, mantendo a integridade das mudanças realizadas.

Contexto: Usuário deve ter acessado o sistema.

Recursos: dados da sessão.

Atores: usuário, sistema**.**

Episódios:

1- Usuário seleciona a opção “sair do sistema”.
2- O sistema encerra a sessão atual do usuário, mantendo a integridade das mudanças realizadas.
3- O sistema redireciona o usuário para a página principal.

Título: CADASTRAR PROJETO.

Objetivo: Permitir que o usuário cadastre um novo projeto no sistema através da inclusão dos seguintes dados do projeto no banco de dados: nome, descrição e data da inclusão. Os campos nome e descrição devem ser informados pelo usuário através do preenchimento de um formulário. O sistema deve preencher automaticamente o campo data da inclusão com a data atual.

Contexto: Usuário deve estar autenticado no sistema.

Recursos: dados do projeto (nome, descrição e data da inclusão).

Atores: usuário, sistema.

Episódios:

1- Sistema exibe um formulário de cadastro de projeto, contendo os campos nome e descrição.
2- Usuário submete o formulário preenchido. **Restrição:** todos os campos devem estar preenchidos.
3- Sistema armazena a data atual, como data da inclusão do projeto.
4- Sistema insere os seguintes dados no banco de dados: nome, descrição e data da inclusão.
5- Sistema avisa ao usuário que o projeto foi inserido com sucesso e o redireciona para página principal.

Exceção:

O usuário já possui um projeto com o mesmo nome informado. O usuário é avisado que deve escolher um novo nome para o projeto.

Título: ALTERAR PROJETO.

Objetivo: Permitir que o usuário altere a descrição de um projeto já cadastrado, através da edição de um formulário contendo a descrição previamente cadastrada.

Contexto: Projeto deve ter sido cadastrado pelo usuário (usuário deve ser administrador).

Recursos: descrição do projeto.

Atores: usuário, sistema.

Episódios:

1- Sistema exibe para o usuário um formulário contendo os campos nome, data de inclusão e descrição do projeto, já preenchidos e permite a edição apenas do campo descrição.
2- Usuário submete o formulário com a nova descrição do projeto.
Restrição: O campo descrição não pode estar vazio e nem conter apenas espaços em branco.
3- Sistema atualiza a descrição do projeto no banco de dados.
4- Sistema comunica o usuário que os dados do projeto foram atualizados com sucesso e o redirecionam para a página principal do sistema.

Título: REMOVER PROJETO.

Objetivo: Permitir que o usuário remova um projeto, seus cenários, símbolos do léxico e arquivos XML vinculados.

Contexto: Projeto cadastrado anteriormente pelo usuário (usuário deve ser administrador).

Recursos: dados do projeto.

Atores: sistema, usuário.

Episódios:

1- O sistema exibe os dados do projeto e solicita uma confirmação de exclusão para o usuário.
2- Usuário confirma exclusão.
3- Sistema exclui o projeto, os símbolos do léxico, cenários e arquivos XML vinculados ao projeto.
4- Sistema comunica ao usuário que a exclusão foi efetuada com sucesso.

Título: SELECIONAR PROJETO.

Objetivo: Permitir que o usuário trabalhe com um projeto específico, previamente cadastrado no sistema. O Sistema irá carregar os dados referentes ao projeto e exibirá no menu lateral todos os símbolos do léxico e cenários já cadastrados no projeto.

Contexto: Usuário deve possuir um projeto cadastrado ou ser colaborador de um.

Recursos: projetos criados pelo usuário e projetos em que o usuário é colaborador.

Atores: usuário, sistema.

Episódios:

1- O sistema exibe uma lista dos projetos acessíveis para o usuário, destacando o seu nível de acesso em cada um deles.
2- Usuário seleciona um dos projetos da lista.
3- Os cenários e símbolos do léxico previamente cadastrados são exibidos em um menu lateral.
4- As seguintes opções de tornam disponíveis para o usuário: cadastrar léxico, cadastrar cenário e gerar grafo do projeto.
5- Caso o usuário possua nível de acesso de administrador ou gerente as seguintes opções surgiriam, além das anteriores: GERAR XML DO PROJETO, gerenciar colaboradores do projeto e verificar alterações propostas.

Título: GERAR XML DO PROJETO.

Objetivo: Permitir que o usuário crie um arquivo XML contendo todas as informações referentes ao projeto, seus símbolos do léxico e cenários e os relacionamentos entre eles.

Contexto: Usuário deve selecionar um projeto do menu de projetos.

Recursos: dados do projeto, cenários e símbolos do léxico pertencentes ao projeto.

Atores: usuário, sistema.

Episódios:

1- Usuário seleciona a opção “Gerar XML do projeto”
2- Sistema atribui automaticamente a data de criação e número de versão para o arquivo XML que será gerado.
3- Usuário escolhe se deseja visualizar o arquivo XML formatado através de um arquivo XSL ou o arquivo XML sem formatação.
4- Sistema gera o arquivo XML, com todas as informações referentes ao projeto.
5- Sistema armazena o arquivo XML gerado no banco de dados.
6- Sistema exibe o arquivo XML gerado para o usuário.

Título: GERAR GRAFO DO PROJETO.

Objetivo: Gerar um grafo do projeto a partir de um dos arquivos XML do projeto, onde os cenários e símbolos do léxico serão representados como nós e seus relacionamentos como arestas.

Contexto: Um arquivo XML do projeto deve ter sido gerado.

Recursos: arquivo XML.

Atores: usuário, sistema, ferramenta de construção de grafos.

Episódios:

1- Usuário seleciona a opção “Gerar grafo do projeto”.
2- Sistema mostra uma lista com as versões de arquivo XML geradas para o projeto selecionado.
3- Usuário seleciona uma das versões.
4- Sistema chama a ferramenta de construção de grafos e passa como parâmetro o arquivo XML escolhido. A ferramenta de construção de grafos constrói o grafo do projeto e o exibe para o usuário.

Título: INCLUIR COLABORADOR NO PROJETO.

Objetivo: Permitir a inclusão de colaboradores para ajudar na construção e evolução do projeto. A inclusão é feita através de uma lista contendo os usuários

cadastrados no sistema. Os colaboradores podem possuir os seguintes níveis de acesso: gerente, usuário participativo e usuário passivo. O gerente possui todos os privilégios do administrador do projeto, exceto adicionar outros usuários colaboradores como nível de acesso de gerente. Os usuários participativos podem sugerir alterações, que podem ser aceitas ou não pelo gerente ou administrador. O usuário passivo só pode visualizar os itens do projeto.

Contexto: O usuário dever estar autenticado como administrador do projeto.

Recursos: login dos usuários cadastrados no sistema, dados do projeto selecionado.

Atores: usuário, sistema.

Episódios:

1- Usuário seleciona a opção “Adicionar colaborador ao projeto”.
2- Sistema exibe uma tela com login de todos usuários cadastrados no sistema e com os níveis de acesso possíveis.
3- Usuário seleciona um usuário da lista e o inclui na lista de colaboradores do projeto. **Restrição:** O usuário deve selecionar um nível de acesso para o usuário escolhido.
4- Sistema inclui usuário como colaborador e permite que ele acesse o projeto.
5- Sistema emite um aviso informando o sucesso da operação.

Título: REMOVER COLABORADOR DO PROJETO.

Objetivo: Permitir a remoção de um colaborador do projeto.

Contexto: Usuário deve estar autenticado como gerente ou administrador do projeto.

Recursos: lista de colaboradores do projeto, dados do projeto.

Atores: sistema, usuário.

Episódios:

1- Sistema seleciona a opção “Remover colaborador do projeto”.
2- Sistema exibe uma lista com os colaboradores do projeto, permitindo que eles sejam removidos.
3- Usuário remove os colaboradores que desejar e clica em “confirmar atualizações”.
4- Sistema atualiza a lista de colaboradores.
5- Sistema emite um aviso informando o sucesso da operação.

Título: VERIFICAR ALTERAÇÕES SUGERIDAS PELOS COLABORADORES.
Objetivo: Permitir que o administrador ou gerente do projeto acate ou rejeite as alterações referentes a símbolos do léxico e cenários, sugeridas pelos usuários participativos.
Contexto: Usuário deve ter sido autenticado como administrador ou gerente do projeto.
Recursos: lista de alterações sugeridas e justificativas.
Atores: usuário, sistema.
Episódios:

1- Usuário seleciona a opção “Verificar alterações sugeridas”
2- O sistema exibe uma lista com todas as alterações sugeridas, tanto para léxicos quanto para cenários. Abaixo de cada alteração sugerida da lista são exibidas duas opções “Aceitar” e “Rejeitar”.
3- O usuário seleciona uma das duas opções para cada alteração e clica no botão “Processar”.
4- O sistema realmente efetua as alterações acatadas e rejeita as que não foram acatadas pelo usuário.
5- Sistema exibe uma tela de sucesso do processamento das alterações.

Título: CADASTRAR SÍMBOLO DO LÉXICO
Objetivo: Permitir que o usuário inclua um símbolo do léxico em um projeto.
Contexto: Projeto cadastrado no sistema.
Recursos: dados do léxico.
Atores: usuário, sistema.
Episódios:

1- Usuário seleciona a opção “Novo símbolo do léxico”.
2- O sistema exibe um formulário com os seguintes campos: nome, noção, impacto, classificação e sinônimo.
3- Usuário submete o formulário preenchido. **Restrição:** Os campos nome, noção e classificação não podem ser deixados em branco.
4- Sistema cadastra o novo símbolo do léxico.
5- Sistema comunica ao usuário que a inclusão foi realizada com sucesso.

Exceção: Se o nome do léxico escolhido ou algum de seus sinônimos já estiver cadastrado no sistema como nome de outro léxico ou sinônimo, então o sistema deve exibir uma mensagem de erro, identificando o que o ocasionou.

Título: EXIBIR SÍMBOLO DO LÉXICO E SEUS RELACIONAMENTOS.

Objetivo: Exibir as informações de um símbolo do léxico juntamente com seus relacionamentos, utilizando elos para permitir o acesso a outros símbolos referenciados pelo que esta sendo exibido e permitir o acesso a outros elementos do projeto que referenciem o símbolo exibido.

Contexto: Símbolo do léxico cadastrado no sistema.

Recursos: dados do léxico, nomes e sinônimos de símbolos do léxico e títulos de cenários do projeto.

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O sistema monta a rede de relacionamentos em torno do símbolo selecionado.
3- Sistema exibe as informações do símbolo e os relacionamentos mapeados através de elos.

Título: ALTERAR SÍMBOLO DO LÉXICO.

Objetivo: Permitir que o usuário altere os dados de um símbolo do léxico cadastrado no sistema, através da edição de um formulário preenchido com suas informações cadastradas.

Contexto: Símbolo do léxico cadastrado no sistema.

Recursos: novos dados do símbolo do léxico.

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O usuário seleciona a opção “Alterar símbolo”
3- Sistema exibe um formulário preenchido com dos dados do léxico selecionado.
4- Usuário faz as alterações que deseja nas informações do símbolo e submete o formulário. **Restrição:** O campo noção não pode ser deixado em branco.
5- O sistema comunica ao usuário que as alterações no símbolo foram feitas com sucesso.

Título: REMOVER SÍMBOLO DO LÉXICO.

Objetivo: Permitir que o usuário remova um símbolo do léxico do projeto selecionado.

Contexto: Símbolo do léxico cadastrado no sistema.

Recursos: dados do símbolo do léxico.

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O usuário seleciona a opção “Remover símbolo”
3- Sistema solicita a confirmação da exclusão.
4- Usuário confirma a exclusão.
5- Sistema comunica o usuário que a exclusão do símbolo foi realizada com sucesso.

Título: CADASTRAR CENÁRIO

Objetivo: Permitir que o usuário inclua um cenário em um projeto.

Contexto: Projeto cadastrado no sistema.

Recursos: dados do cenário.

Atores: usuário, sistema.

Episódios:

1- Usuário seleciona a opção “Novo cenário”.
2- O sistema exibe um formulário com os seguintes campos: título, objetivo, contexto, recursos, atores, episódios e exceção.
3- Usuário submete o formulário preenchido. **Restrição:** Os campos título, objetivo, contexto, recursos, atores e episódios não podem ser deixados em branco.
4- Sistema cadastra o novo cenário no projeto.
5- Sistema comunica ao usuário que a inclusão do cenário foi realizada com sucesso.

Exceção: Se o título escolhido para o cenário escolhido já estiver cadastrado no sistema como título de outro cenário ou nome/sinônimo de um léxico, então o sistema deve exibir uma mensagem de erro, identificando o que o ocasionou.

Título: EXIBIR CENÁRIO E SEUS RELACIONAMENTOS.

Objetivo: Exibir as informações de um cenário juntamente com seus relacionamentos, utilizando elos para permitir o acesso a outros elementos

referenciados pelo cenário que esta sendo exibido e permitir o acesso a outros elementos do projeto que referenciem o cenário exibido.

Contexto: Cenário cadastrado no sistema.

Recursos: dados do cenário, nomes e sinônimos de símbolos do léxico e títulos de cenários do projeto.

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um cenário do projeto, exibido no menu lateral.
2- O sistema monta a rede de relacionamentos em torno do cenário selecionado.
3- Sistema exibe as informações do cenário e os relacionamentos mapeados através de elos.

Título: ALTERAR CENÁRIO.

Objetivo: Permitir que o usuário altere os dados de um cenário cadastrado no sistema, através da edição de um formulário preenchido com suas informações cadastradas.

Contexto: Cenário cadastrado no sistema.

Recursos: novos dados do cenário.

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um cenário do projeto, exibido no menu lateral.
2- O usuário seleciona a opção “Alterar cenário”
3- Sistema exibe um formulário preenchido com os dados do cenário selecionado.
4- Usuário faz as alterações que deseja nas informações do cenário e submete o formulário. **Restrição:** Os campos objetivo, contexto, recursos, atores e episódios não podem ser deixados em branco.
5- O sistema comunica ao usuário que as alterações no cenário foram feitas com sucesso.

Título: REMOVER CENÁRIO

Objetivo: Permitir que o usuário remova um cenário do projeto selecionado.

Contexto: Cenário cadastrado no sistema.

Recursos: dados do cenário

Atores: usuário, sistema.

Episódios:

1- O usuário seleciona um cenário do projeto, exibido no menu lateral.
2- O usuário seleciona a opção “Remover cenário”
3- Sistema solicita a confirmação da exclusão.
4- Usuário confirma a exclusão.
5- Sistema comunica o usuário que a exclusão do cenário foi realizada com sucesso.

5.2.2. Divisão dos cenários em grupos

Foram formados cinco grupos a partir dos cenários que descrevem as situações do sistema. O primeiro grupo (Figura 39) é formado pelos cenários: cadastrar usuário, recuperar senha do usuário, acessar o sistema, alterar dados do usuário e sair do sistema. Estes cenários foram colocados no mesmo grupo, pois considerando uma visão modular do sistema pertenceriam ao módulo usuário, já que todos descrevem situações relacionadas aos dados do usuário.

O segundo grupo (Figura 40) é formado pelos cenários: cadastrar projeto, alterar projeto, remover projeto, selecionar projeto, gerar XML do projeto e gerar grafo do projeto. Estes cenários descrevem situações relacionadas a projetos do sistema, e corresponderiam ao módulo projeto, por isso foram colocados no mesmo grupo.

O terceiro grupo (Figura 41), formado pelos cenários: incluir colaborador no projeto, remover colaborador do projeto e verificar alterações sugeridas pelos colaboradores, corresponde ao módulo de gerência de colaboradores. As situações descritas pelos cenários destes grupos estão relacionadas ao controle da interação entre os usuários nos projetos.

O quarto grupo (Figura 42) corresponde ao módulo léxico do sistema. Os cenários que compõem este grupo são: cadastrar símbolo do léxico, exibir símbolo do léxico e seus relacionamentos, alterar símbolo do léxico e remover símbolo do léxico. Estes cenários descrevem situações relacionadas a símbolos do léxico.

O quinto e último grupo (Figura 43) corresponde ao módulo cenário. Este grupo é formado pelos cenários: cadastrar cenário, exibir cenário e seus relacionamentos, alterar cenário e remover cenário. Estes cenários foram colocados no mesmo grupo, pois descrevem situações relacionadas a cenários.

Título: CADASTRAR USUÁRIO
Objetivo: Permitir que o usuário utilize o sistema, cadastrando seus dados no banco de dados através do preenchimento de um formulário com os seguintes campos: nome, sobrenome, e-mail, instituição, login e senha.
Contexto: Usuário não possui cadastro no sistema.
Recursos: dados do usuário (nome, sobrenome, e-mail, instituição, login e senha).
Atores: usuário, sistema.
Episódios:
1- Usuário clica no atalho "Cadastre-se" na página principal do sistema.
2- Sistema exibe um formulário de cadastro, contendo os seguintes campos para o usuário preencher: nome, sobrenome, e-mail, instituição, login, senha e confirmar senha.
3- Usuário submete formulário preenchido. **Restrição**: Todos os campos do formulário devem ser preenchidos. O campo confirmar senha deve possuir a mesma cadeia de caracteres alfanuméricos informada no campo senha.
4- Sistema insere os dados do novo usuário no banco de dados.
5- Sistema avisa que usuário foi cadastrado com sucesso e o redireciona para página principal.
Exceção:
O login informado pelo usuário já se encontra no banco de dados do sistema. O usuário é alertado e informado que deve escolher novo login.

Título: RECUPERAR SENHA DO USUÁRIO
Objetivo: Permitir que o usuário recupere sua senha. Ao solicitar recuperação de sua senha o usuário preencherá um formulário com seu login e e-mail. Após a verificação dos dados o sistema irá substituir a senha antiga por uma nova senha randômica e envia-la para o e-mail do usuário.
Contexto: Usuário deve ter sido previamente cadastrado no sistema
Recursos: login e e-mail do usuário.
Atores: usuário e sistema.
Episódios:
1- Usuário clica no atalho "Esqueceu a senha?" na página inicial do sistema.
2- O sistema exibe um formulário contendo os campos e-mail e login para o usuário preencher.
3- O usuário submete o formulário preenchido. **Restrição:** Todos os campos devem ser preenchidos.
4- O sistema verifica se o login e e-mail informado são iguais aos do cadastro do usuário.
5- Sistema gera uma nova senha randômica e a envia para o e-mail do usuário.
6- Sistema exibe um aviso informando que a senha foi alterada e que o usuário deve verificar o e-mail informado para obter a nova senha.
Exceção:
O login e e-mail informado pelo usuário não estão corretos. O sistema informará ao usuário que o login e a senha não estão corretos e permitirá que ele tente mais duas vezes. Caso erre mais duas vezes o usuário terá que esperar por 20 minutos para tentar novamente.

Título: ACESSAR O SISTEMA
Objetivo: Permitir que o usuário utilize o sistema se identificando através do preenchimento de um pequeno formulário com seu login e senha.
Contexto: Usuário deve ter sido previamente cadastrado no sistema.
Recursos: login, senha.
Atores: usuário, sistema.
Episódios:
1- Sistema exibe a página inicial para o usuário contendo um formulário com dois campos: login e senha.
2- Usuário submete o formulário com os campos preenchidos. **Restrição:** Nenhum dos campos pode ser deixado em branco.
3- O sistema carrega os projetos que o usuário participa e o redireciona para a página principal.
Exceção:
O login e senha informada pelo usuário não estão corretos. O sistema permite que o usuário tente novamente por mais duas vezes. Se o usuário errar seu login e senha por mais duas vezes ele terá que esperar 20 minutos para tentar novamente.

Título: ALTERAR DADOS DO USUÁRIO
Objetivo: Permitir que o usuário altere seus dados armazenados no sistema, através da edição de um formulário contendo as informações previamente cadastradas.
Contexto: Usuário deve ter sido previamente cadastrado no sistema. Autenticar usuário no sistema.
Recursos: novos dados do usuário.
Atores: usuário e sistema.
Episódios:
1- Usuário seleciona a opção "Alterar dados" do menu principal.
2- Sistema exibe um formulário com os seguintes campos já preenchidos, permitindo sua edição: nome, sobrenome, e-mail, instituição.
3- Usuário submete o formulário preenchido com os novos dados. **Restrição:** O formulário não pode conter campos em branco.
4- Sistema atualiza os dados do usuário no banco de dados.
5- Sistema avisa que os dados do usuário foram alterados com sucesso e o redireciona para a página principal.

Título: SAIR DO SISTEMA
Objetivo: Permitir que o usuário saia do sistema, mantendo a integridade das mudanças realizadas.
Contexto: Usuário deve ter acessado o sistema.
Recursos: dados do usuário.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "sair do sistema".
2- O sistema encerra a sessão atual do usuário, mantendo a integridade das mudanças realizadas.
3- O sistema redireciona o usuário para a página principal.

Figura 39 – Grupo correspondente ao módulo usuário.

Título: CADASTRAR PROJETO.
Objetivo: Permitir que o usuário cadastre um novo projeto no sistema através da inclusão dos seguintes dados do projeto no banco de dados: nome, descrição e data da inclusão. Os campos nome e descrição devem ser informados pelo usuário através do preenchimento de um formulário. O sistema deve preencher automaticamente o campo data da inclusão com a data atual.
Contexto: Usuário deve estar autenticado no sistema.
Recursos: dados do projeto (nome, descrição e data da inclusão).
Atores: usuário, sistema.
Episódios:
1- Sistema exibe um formulário de cadastro de projeto, contendo os campos nome e descrição.
2- Usuário submete o formulário preenchido. **Restrição:** todos os campos devem estar preenchidos.
3- Sistema armazena a data atual, como data da inclusão do projeto.
4- Sistema insere os seguintes dados no banco de dados: nome, descrição e data da inclusão.
5- Sistema avisa que o projeto foi inserido com sucesso e o redireciona para página principal.

Exceção:
O usuário já possui um projeto com o mesmo nome informado.
O usuário é avisado que deve escolher um novo nome para o projeto.

Título: REMOVER PROJETO.
Objetivo: Permitir que o usuário remova um projeto, seus cenários, símbolos do léxico e arquivos XML vinculados.
Contexto: Projeto cadastrado anteriormente pelo usuário (usuário deve ser administrador).
Recursos: dados do projeto.
Atores: sistema, usuário.
Episódios:
1- O sistema exibe os dados do projeto e solicita uma confirmação de exclusão para o usuário.
2- Usuário confirma exclusão.
3- Sistema exclui o projeto, os símbolos do léxico, cenários e arquivos XML vinculados ao projeto.
4- Sistema comunica ao usuário que a exclusão foi efetuada com sucesso.

Título: GERAR XML DO PROJETO.
Objetivo: Permitir que o usuário crie um XML contendo todas as informações referentes ao projeto, seus símbolos do léxico e cenários e os relacionamentos entre eles.
Contexto: Usuário deve selecionar um projeto do menu de projetos.
Recursos: dados do projeto, cenários e símbolos do léxico pertencentes ao projeto.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "Gerar XML do projeto"
2- Sistema atribui automaticamente a data de criação e número de versão para o XML que será gerado.
3- Usuário escolhe se deseja visualizar o XML formatado através de um arquivo XSL ou sem nenhum tipo de formatação.
4- Sistema gera XML, com todas as informações referentes ao projeto.
5- Sistema armazena o XML gerado no banco de dados.
6- Sistema exibe o XML gerado para o usuário.

Título: ALTERAR PROJETO.
Objetivo: Permitir que o usuário altere a descrição de um projeto já cadastrado, através da edição de um formulário contendo a descrição previamente cadastrada.
Contexto: Projeto deve ter sido cadastrado pelo usuário (usuário deve ser administrador).
Recursos: descrição do projeto.
Atores: usuário, sistema.
Episódios:
1- Sistema exibe para o usuário um formulário contendo os campos nome, data de inclusão e descrição do projeto já preenchidos, mas permite a edição apenas do campo descrição.
2- Usuário submete o formulário com a nova descrição do projeto. **Restrição:** O campo descrição não pode estar vazio e nem conter apenas espaços em branco.
3- Sistema atualiza a descrição do projeto no banco de dados.
4- Sistema comunica o usuário que os dados do projeto foram atualizados com sucesso e o redirecionam para a página principal do sistema.

Título: SELECIONAR PROJETO.
Objetivo: Permitir que o usuário trabalhe com um projeto específico, previamente cadastrado no sistema. Sistema irá carregar os dados referentes ao projeto e exibirá no menu lateral todos os símbolos do léxico e cenários já cadastrados.
Contexto: Usuário deve possuir um projeto cadastrado ou ser colaborador de um.
Recursos: projetos criados pelo usuário e projetos em que o usuário é colaborador.
Atores: usuário, sistema.
Episódios:
1- O sistema exibe uma lista dos projetos acessíveis para o usuário, destacando o seu nível de acesso em cada um deles.
2- Usuário seleciona um dos projetos da lista.
3- Os cenários e símbolos do léxico previamente cadastrados são exibidos em um menu lateral.
4- As seguintes opções de tornam disponíveis para o usuário: cadastrar léxico, cadastrar cenário e gerar grafo do projeto.
5- Caso o usuário possua nível de acesso de administrador ou gerente as seguintes opções surgiriam, além das anteriores: gerar XML do projeto, gerenciar colaboradores do projeto e verificar alterações propostas.

Título: GERAR GRAFO DO PROJETO.
Objetivo: Gerar um grafo do projeto a partir de um dos arquivos XML do projeto, onde os cenários e símbolos do léxico serão representados como nós e os seus relacionamentos como arestas.
Contexto: Um arquivo XML do projeto deve ter sido gerado.
Recursos: arquivo XML.
Atores: usuário, sistema, ferramenta de construção de grafos.
Episódios:
1- Usuário seleciona a opção "Gerar grafo do projeto".
2- Sistema mostra uma lista com as versões de arquivo XML geradas para o projeto selecionado.
3- Usuário seleciona uma das versões.
4- Sistema chama a ferramenta de construção de grafos e passa como parâmetro o XML escolhido. A ferramenta constrói o grafo e o exibe para o usuário.

Figura 40 – Grupo correspondente ao módulo projeto.

Título: INCLUIR COLABORADOR NO PROJETO.
Objetivo: Permitir a inclusão de colaboradores para ajudar na construção e evolução do projeto. A inclusão é feita através de uma lista contendo os usuários cadastrados no sistema. Os colaboradores podem possuir os seguintes níveis de acesso: gerente, usuário participativo e usuário passivo. O gerente possui todos os privilégios do administrador do projeto, exceto adicionar outros usuários colaboradores como nível de acesso de gerente. Os usuários participativos podem sugerir alterações, que podem ser aceitas ou não pelo gerente ou administrador. O usuário passivo só pode visualizar os itens do projeto.
Contexto: O usuário dever estar autenticado como administrador do projeto.
Recursos: login dos usuários cadastrados no sistema, dados do projeto selecionado.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "Adicionar colaborador ao projeto".
2- Sistema exibe uma tela com login de todos usuários cadastrados no sistema e com os níveis de acesso possíveis.
3- Usuário seleciona um usuário da lista e o inclui na lista de colaboradores do projeto. **Restrição:** O usuário deve selecionar um nível de acesso para o usuário escolhido.
4- Sistema inclui usuário como colaborador e permite que ele acesse o projeto.
5- Sistema emite um aviso informando o sucesso da operação.

Título: REMOVER COLABORADOR DO PROJETO.
Objetivo: Permitir a remoção de um colaborador do projeto.
Contexto: Usuário deve estar autenticado como gerente ou administrador do projeto.
Recursos: lista de colaboradores do projeto, dados do projeto.
Atores: sistema, usuário.
Episódios:
1- Sistema seleciona a opção "Remover colaborador do projeto".
2- Sistema exibe uma lista com os colaboradores do projeto, permitindo que eles sejam removidos.
3- Usuário remove os colaboradores que desejar e clica em "confirmar atualizações".
4- Sistema atualiza a lista de colaboradores.
5- Sistema emite um aviso informando o sucesso da operação.

Título: VERIFICAR ALTERAÇÕES SUGERIDAS PELOS COLABORADORES.
Objetivo: Permitir que o administrador ou gerente do projeto acate ou rejeite as alterações referentes a símbolos do léxico e cenários, sugeridas pelos usuários participativos.
Contexto: Usuário deve ter sido autenticado como administrador ou gerente do projeto.
Recursos: lista de alterações sugeridas e justificativas.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "Verificar alterações sugeridas"
2- O sistema exibe uma lista com todas as alterações sugeridas, tanto para léxicos quanto para cenários. Abaixo de cada alteração sugerida da lista são exibidas duas opções "Aceitar" e "Rejeitar".
3- O usuário seleciona uma das duas opções para cada alteração e clica no botão "Processar".
4- O sistema realmente efetua as alterações acatadas e rejeita as que não foram acatadas pelo usuário.
5- Sistema exibe uma tela de sucesso do processamento das alterações.

Figura 41 – Grupo correspondente ao módulo administração de colaboradores.

Título: CADASTRAR SÍMBOLO DO LÉXICO
Objetivo: Permitir que o usuário inclua um símbolo do léxico em um projeto.
Contexto: Projeto cadastrado no sistema.
Recursos: dados do léxico.
Atores: usuário, sistema.
Episódios:
1- Usuário seleciona a opção "Novo símbolo do léxico".
2- O sistema exibe um formulário com os seguintes campos: nome, noção, impacto, classificação e sinônimo.
3- Usuário submete o formulário preenchido. **Restrição:** Os campos nome, noção e classificação não podem ser deixados em branco.
4- Sistema cadastra o novo símbolo do léxico.
5- Sistema comunica ao usuário que a inclusão foi realizada com sucesso.

Exceção: Se o nome do léxico escolhido ou algum de seus sinônimos já estiver cadastrado no sistema como nome de outro léxico ou sinônimo, então o sistema deve exibir uma mensagem de erro, identificando o que o ocasionou.

Título: ALTERAR SÍMBOLO DO LÉXICO.
Objetivo: Permitir que o usuário altere os dados de um símbolo do léxico cadastrado no sistema, através da edição de um formulário preenchido com suas informações cadastradas.
Contexto: Símbolo do léxico cadastrado no sistema.
Recursos: novos dados do símbolo do léxico.
Atores: usuário, sistema.
Episódios:
1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O usuário seleciona a opção "Alterar símbolo"
3- Sistema exibe um formulário preenchido com dos dados do léxico selecionado.
4- Usuário faz as alterações que deseja nas informações do símbolo e submete o formulário. **Restrição:** O campo noção não pode ser deixado em branco.
5- O sistema comunica ao usuário que as alterações no símbolo foram feitas com sucesso.

Título: REMOVER SÍMBOLO DO LÉXICO.
Objetivo: Permitir que o usuário remova um símbolo do léxico do projeto selecionado.
Contexto: Símbolo do léxico cadastrado no sistema.
Recursos: dados do símbolo do léxico.
Atores: usuário, sistema.
Episódios:
1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O usuário seleciona a opção "Remover símbolo"
3- Sistema solicita a confirmação da exclusão.
4- Usuário confirma a exclusão.
5- Sistema comunica o usuário que a exclusão do símbolo foi realizada com sucesso.

Título: EXIBIR SÍMBOLO DO LÉXICO E SEUS RELACIONAMENTOS.
Objetivo: Exibir as informações de um símbolo do léxico juntamente com seus relacionamentos, utilizando elos para permitir o acesso a outros símbolos referenciados pelo que esta sendo exibido e permitir o acesso a outros elementos do projeto que referenciem o símbolo exibido.
Contexto: Símbolo do léxico cadastrado no sistema.
Recursos: dados do léxico, nomes e sinônimos de símbolos do léxico e títulos de cenários do projeto.
Atores: usuário, sistema.
Episódios:
1- O usuário seleciona um símbolo do léxico do projeto, exibido no menu lateral.
2- O sistema monta a rede de relacionamentos em torno do símbolo selecionado.
3- Sistema exibe as informações do símbolo e os relacionamentos mapeados através de elos.

Figura 42 – Grupo correspondente ao módulo léxico.

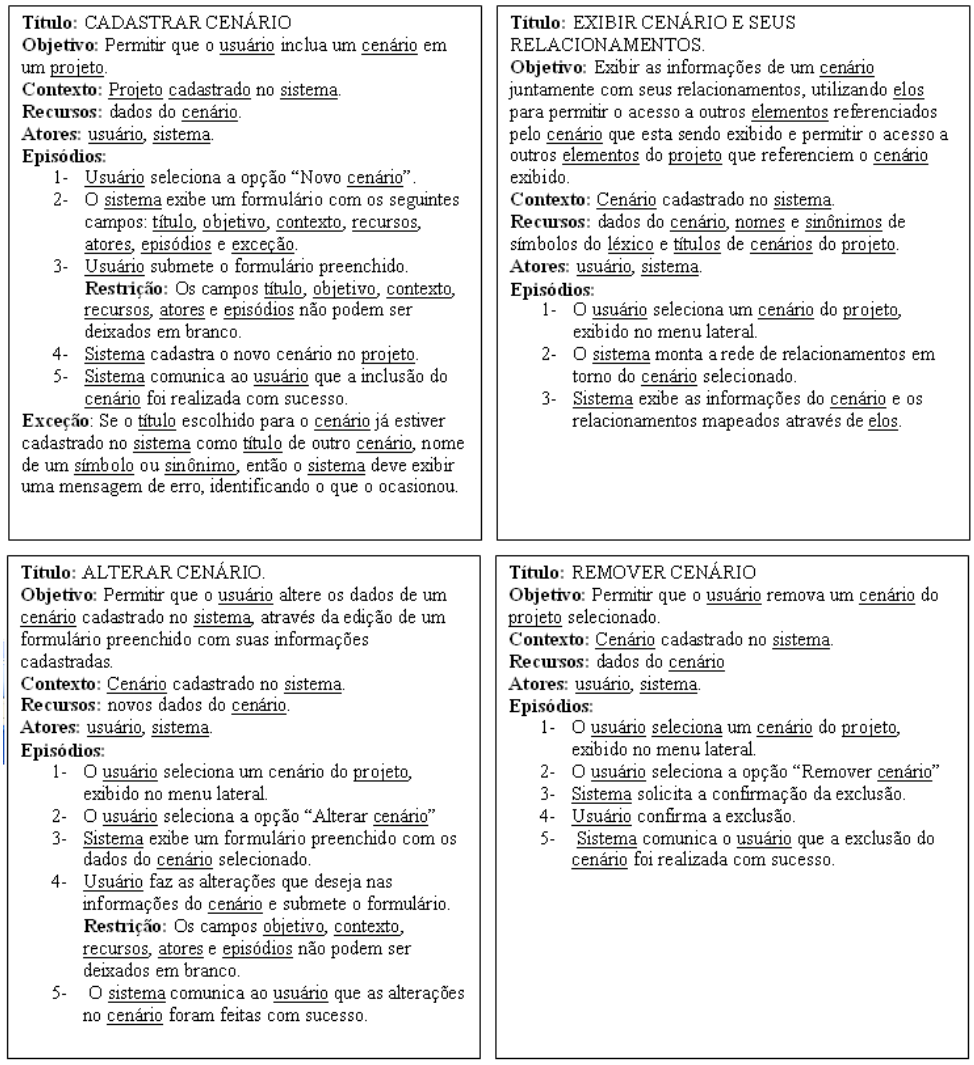


Figura 43 – Grupo correspondente ao módulo cenário.

5.2.3. Identificar os cenários raiz

Neste passo, primeiro iremos determinar o relacionamento entre os cenários pertencentes aos grupos identificados e, a partir dos relacionamentos identificados estabeleceremos uma ordem entre estes cenários. O cenário raiz será o cenário que não dependa de nenhum outro cenário do seu grupo.

No primeiro grupo identificamos os relacionamentos mostrados na Figura 44. Com base nestes relacionamentos podemos determinar que o cenário cadastrar usuário deve preceder todos os outros. Com isto, identificamos o cenário cadastrar usuário como cenário raiz deste grupo.

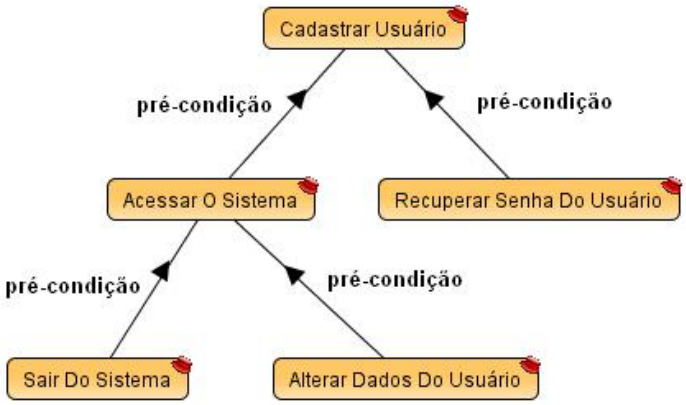


Figura 44 – Relacionamentos entre os cenários do grupo correspondente ao módulo usuário.

Os relacionamentos identificados no segundo grupo podem ser vistos na Figura 45. Podemos observar que o cenário raiz deste grupo é o cenários cadastrar projeto, pois não possui nenhuma pré-condição dentro do seu grupo.

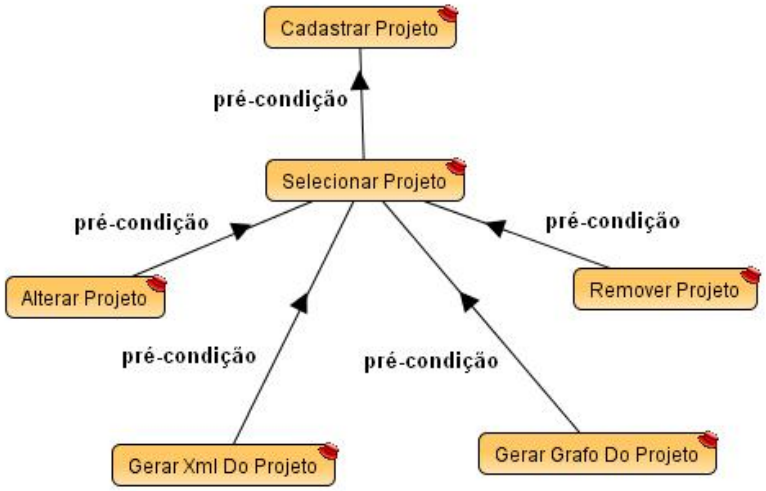


Figura 45 – Relacionamentos entre os cenários do grupo correspondente ao módulo projeto.

De acordo com os relacionamentos identificados entre os cenários do grupo correspondente ao módulo gerência de colaboradores (Figura 46), podemos identificar como cenário raiz o cenário incluir colaborador no projeto, pois este cenário não possui nenhuma pré-condição dentro de seu grupo.

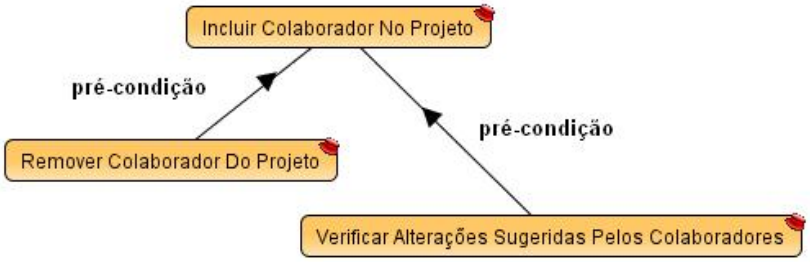


Figura 46 – Relacionamentos entre os cenários do grupo correspondente ao módulo colaboradores.

Na Figura 47 pode-se observar os relacionamentos entre os cenários do grupo referente ao módulo léxico. O cenário raiz deste grupo é o cenário cadastrar símbolo do léxico, pois não possui nenhuma pré-condição entre os cenários de seu grupo.

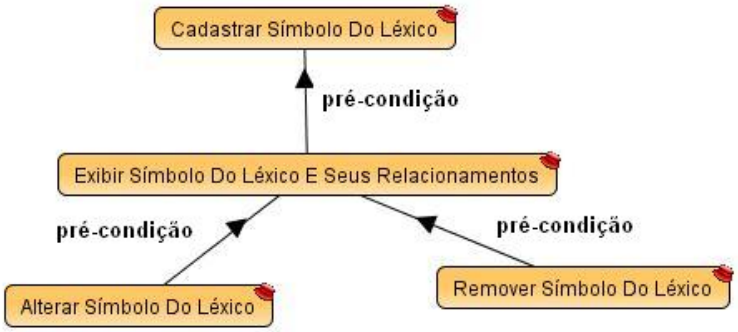


Figura 47 – Relacionamentos entre os cenários do grupo correspondente ao módulo léxico.

O cenário raiz do grupo correspondente ao módulo cenário é o cenário cadastrar cenário. Isto porque, como podemos observar na Figura 48, este cenário não possui nenhuma pré-condição entre os cenários do grupo.

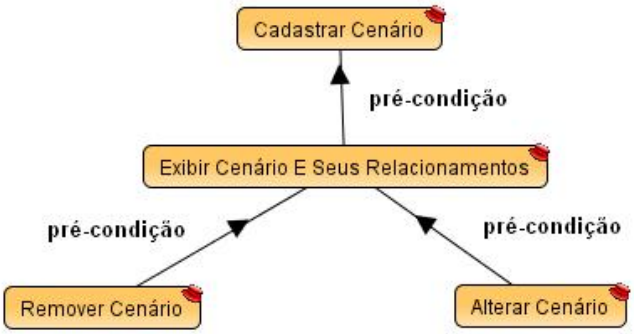


Figura 48 – Relacionamentos entre os cenários do grupo correspondente ao módulo cenário.

5.2.4. Construir cenário integrador

O primeiro passo para a construção do cenário integrador é a identificação dos relacionamentos entre os cenários raiz e sua ordem. Podemos observar na Figura 49 os relacionamentos entre os cenários raiz identificados. Com base na ordem determinada através destes relacionamentos e o modelo apresentado na subseção 4.1.3.3, criamos o cenário integrador para o software C&L. Este cenário pode ser visto na Figura 50.

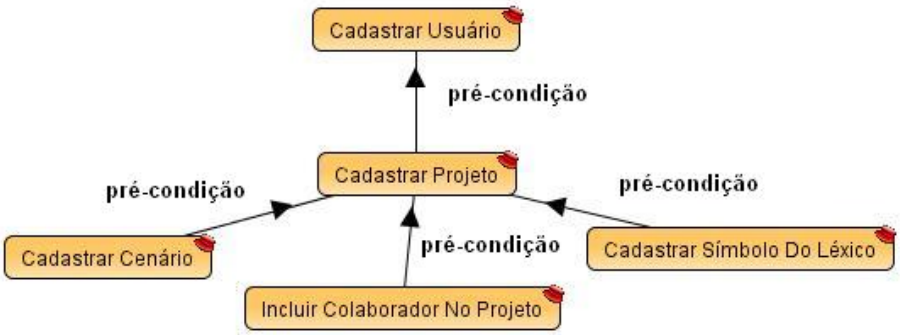


Figura 49 – Relacionamentos entre os cenários raiz.

Título: Sistema C&L
Objetivo: Permitir a colaboração entre diversos usuários na elaboração de projetos utilizando cenários e símbolos do léxico, fornecendo meios para facilitar a integração com outras ferramentas.
Contexto: - Usuário deseja elicitar a linguagem e as situações de um domínio de aplicação específico.
Recursos: -
Atores: -
Episódios:

1- Se o usuário ainda não possuir cadastro então CADASTRAR USUÁRIO, senão ACESSAR SISTEMA.
2- Usuário pode CADASTRAR PROJETO ou SELECIONAR PROJETO se já houver algum cadastrado.
3- # [Usuário pode INCLUIR COLABORADOR NO PROJETO]
4- [Usuário pode CADASTRAR SÍMBOLO DO LÉXICO no projeto.]
5- [Usuário pode CADASTRAR CENÁRIO no projeto.]
6- Usuário pode SAIR DO SISTEMA. #

Figura 50 – Cenário integrador do software C&L.

5.2.5. Refinamento dos cenários

A seguir mostraremos os cenários divididos em camadas, seguindo o *framework* MVC, obtidos através do refinamento dos cenários construídos inicialmente. Como o número de cenários resultante deste refinamento é muito grande, mostraremos apenas alguns cenários de cada camada do software. Os cenários que mostraremos a seguir são resultantes do refinamento dos seguintes cenários, pertencentes ao módulo léxico: “cadastrar símbolo do léxico”, “exibir símbolo do léxico e seus relacionamentos”, “alterar símbolo do léxico”, “remover símbolo do léxico”.

5.2.5.1. Cenários da camada de visão

Título: EXIBIR PÁGINA DE CADASTRO DE NOVO SÍMBOLO DO LÉXICO
Objetivo: Permitir que o usuário informe os dados do símbolo do léxico para seu cadastro no projeto. Para tanto, a página exibida conterá um formulário com os seguintes campos: nome, noção, classificação, impacto e sinônimo.
Contexto: Pode ser acessado através da opção “Novo símbolo” no menu superior da página principal.
Atores: usuário autenticado no sistema.
Recursos: script.js e estilo.css

Episódios:

1- Exibir campo Nome.
2- Exibir campo Noção.
3- Exibir *combobox* Classificação, com as opções: sujeito, verbo, estado e objeto.
4- Exibir campo Impacto.
5- Exibir campo Sinônimo(s)
6- Ao clicar no botão cadastrar o formulário será submetido e o sistema será redirecionado para o CONTROLE LÉXICO. **Restrição:** Antes de submeter o formulário VERIFICAR FORMULÁRIO DE CADASTRO DE LÉXICO.

Título: EXIBIR SÍMBOLO DO LÉXICO E SEUS RELACIONAMENTOS

Objetivo: Permitir a visualização dos dados de um símbolo do léxico. Para tanto, a página exibida conterá as seguintes informações: nome, noção, impacto, classificação e sinônimos. Os campos noção e impacto podem conter elos para outros símbolos do léxico.

Contexto: Pode ser acessado através da seleção de um símbolo do léxico do menu lateral.

Atores: usuário autenticado no sistema.

Recursos: script.js e estilo.css

Episódios:

1- Exibir nome do símbolo.
2- Exibir noção do símbolo, já com os elos montados.
3- Exibir classificação do símbolo.
4- Exibir impacto do símbolo.
5- Exibir sinônimos do símbolo.
6- Exibir botão “Alterar símbolo”.
7- Exibir botão “Remover símbolo”.
8- Ao clicar no botão “Alterar símbolo”, o formulário será submetido com o comando alterar e o sistema será redirecionado para o CONTROLE USUÁRIO.
9- Ao clicar no botão “Remover símbolo”, o formulário será submetido com o comando remover e o sistema será redirecionado para o CONTROLE USUÁRIO.

Título: EXIBIR PÁGINA DE ALTERAÇÃO DOS DADOS DO SÍMBOLO DO LÉXICO.

Objetivo: Permitir que o usuário altere os dados do símbolo do léxico. Para isto, será exibido um formulário com os dados do símbolo do léxico já preenchido, permitindo a edição dos campos noção, impacto, classificação e sinônimos.

Contexto: Pode ser acessado através do atalho "Alterar símbolo" da página de exibição dos dados do símbolo do léxico.

Atores: usuário autenticado no sistema.

Recursos: script.js e estilo.css

Episódios:

1- Exibir campo nome já preenchido.
2- Exibir campo noção já preenchido, permitindo sua edição.
3- Exibir campo classificação já preenchido, permitindo sua edição.
4- Exibir campo impacto já preenchido, permitindo sua edição.
5- Exibir campo sinônimo já preenchido, permitindo sua edição.
6- Ao clicar no botão "Alterar", o formulário será submetido com o comando alterar e o sistema será redirecionado para o CONTROLE USUÁRIO. **Restrição:** Antes de submeter o formulário, VERIFICAR FORMULÁRIO DE ALTERAÇÃO DE SÍMBOLO DO LÉXICO.

5.2.5.2. Cenário da camada de controle

Titulo: CONTROLE LÉXICO

Objetivo: Intermediar a comunicação entre a camada de visão e a camada de modelo do módulo léxico.

Contexto: novo_léxico.html, alterar_dados_léxico.lp, erro.lp, página_principal.lp, sucesso.lp.

Atores: camada de modelo e camada de visão.

Recursos: dados do léxico, informações para as páginas de erro, informações para as páginas de sucesso.

Episódios:

1- Se a requisição recebida for para cadastrar símbolo do léxico então CHECAR NOME DO LÉXICO. Se nome disponível então CADASTRAR SÍMBOLO DO LÉXICO.
2- Se o símbolo for cadastrado corretamente então EXIBIR PÁGINA DE SUCESSO, senão EXIBIR PÁGINA DE ERRO.

3- Se a requisição recebida for para remover símbolo então REMOVER SÍMBOLO DO LÉXICO.
4- Se o símbolo for removido com sucesso então EXIBIR PÁGINA DE SUCESSO, senão EXIBIR PÁGINA DE ERRO.
5- Se a requisição recebida for para alterar símbolo do léxico então ALTERAR SÍMBOLO DO LÉXICO.
6- Se a requisição recebida for para exibir dados do símbolo e seus relacionamentos, então RECUPERAR DADOS DO SÍMBOLO E MONTAR SEUS RELACIONAMENTOS. Após a recuperação dos dados, EXIBIR SÍMBOLO DO LÉXICO E SEUS RELACIONAMENTOS.

5.2.5.3. Cenários da camada modelo

Titulo: CADASTRAR SÍMBOLO DO LÉXICO

Objetivo: Cadastrar símbolo do léxico no projeto do usuário, através da inserção de seus dados no banco de dados.

Contexto: É necessário o preenchimento do formulário presente em EXIBIR PÁGINA DE CADASTRO DE NOVO SÍMBOLO DO LÉXICO.

Atores: CONTROLE LÉXICO**.**

Recursos: dados do símbolo do léxico, nome do projeto (nome, noção, classificação, impacto e sinônimos), banco de dados.

Episódios:

1- Receber os dados do símbolo do léxico e o nome do projeto repassados pela CAMADA LÉXICO.
2- Conectar ao banco de dados. **Restrição:** Banco de dados deve estar disponível.
3- Inserir dados do símbolo do léxico no banco de dados.
4- Inserir sinônimos do símbolo do léxico no banco de dados.
5- Desconectar do banco de dados.
6- Informar a camada CONTROLE LÉXICO que os dados foram inseridos.

Exceção:

O nome do símbolo do léxico informado já existe no banco de dados. Informamos a camada CONTROLE LÉXICO que não foi possível inserir os dados porque o nome já existe.

Titulo: REMOVER SÍMBOLO DO LÉXICO

Objetivo: Remover um símbolo do léxico do projeto do usuário, através da remoção de seus dados do banco de dados.

Contexto: Dados do símbolo devem estar cadastrados no sistema.

Atores: CONTROLE LÉXICO.

Recursos: nome do símbolo, nome do projeto em que está cadastrado, banco de dados.

Episódios:

1- Receber o nome do símbolo e do projeto da camada CONTROLE LÉXICO.
2- Conectar ao banco de dados. **Restrição:** Banco de dados deve estar disponível.
3- Remover símbolo do léxico do banco de dados.
4- Desconectar do banco de dados.
5- Informar a camada CONTROLE LÉXICO que o símbolo foi removido.

Titulo: ALTERAR SÍMBOLO DO LÉXICO

Objetivo: Alterar os dados de um símbolo do léxico cadastrado no projeto, através de uma atualização de seus dados cadastrados no banco de dados.

Contexto: Símbolo do léxico cadastrado no sistema.

Atores: CONTROLE LÉXICO.

Recursos: dados alterados do símbolo do léxico, nome do projeto, banco de dados.

Episódios:

1- Receber os dados do símbolo do léxico alterados e o projeto a que pertence.
2- Conectar ao banco de dados. **Restrição:** Banco de dados deve estar disponível.
3- Alterar os dados do símbolo no do banco de dados.
4- Desconectar do banco de dados.
5- Informar a camada CONTROLE LÉXICO que o símbolo foi alterado.

Titulo: RECUPERAR DADOS DO SÍMBOLO E MONTAR SEUS RELACIONAMENTOS.

Objetivo: Recuperar os dados do símbolo do léxico do banco de dados e criar elos nos textos recuperados dos campos noção e impacto, através da identificação da ocorrência de nomes ou sinônimos de outros símbolos do léxico nestes textos.

Contexto:

Atores: CONTROLE LÉXICO.

Recursos: nome do símbolo, nome do projeto, nome dos símbolos do projeto, sinônimos dos símbolos do léxico.

Episódios:

1- Receber o nome do símbolo e do projeto da CAMADA CONTROLE.
2- Recuperar o nome dos símbolos do léxico do projeto.
3- Recuperar os sinônimos dos símbolos do léxico do projeto.
4- COLOCAR ELOS NO TEXTO impacto.
5- COLOCAR ELOS NO TEXTO noção.

Encaminhar os dados do símbolo do léxico, já com os elos montados, a camada CONTROLE LÉXICO.

5.2.6. Operacionalização dos cenários

Se fossemos mostrar todos os cenários operacionalizados nesta subseção teríamos que mostrar o código fonte de todo o software, o que a tornaria muito extensa, por isso decidimos mostrar um exemplo através da operacionalização do cenário “exibir símbolo do léxico e seus relacionamentos”, que é um dos principais do software. Este cenário abrange a principal característica do C&L, que é uma estratégia de rastreamento automático [Sayão06]. Esta estratégia se baseia na identificação automática dos relacionamentos entre símbolos do léxico e cenários e a montagem de elos, que são fundamentais na implementação da estratégia, pois permitem a navegação entre os símbolos e cenários relacionados.

Antes de mostrarmos a operacionalização do cenário iremos apresentar o problema que é resolvido por ele. Depois mostraremos o algoritmo que era utilizado antes da re-engenharia do C&L e a falha que apresentava. Por fim, apresentaremos o novo algoritmo, que foi implementado no novo C&L.

5.2.6.1. Apresentação do problema

Um projeto do C&L pode possuir vários elementos cadastrados, que podem ser símbolos do léxico ou cenários. Estes elementos possuem seus identificadores únicos dentro do projeto. No caso de símbolos do léxico, este identificador é o seu nome e no caso de cenários, este identificador é seu título.

O desejado é que o C&L identifique automaticamente a ocorrência destes identificadores no texto dos elementos que estão sendo exibidos e que, caso encontre algum identificador, o substituía por um elo para o elemento correspondente. Estes elos representam os relacionamentos que podem ocorrer entre dois cenários, entre dois símbolos do léxico ou entre um cenário e um símbolo do léxico. Os identificadores dos elementos podem ser compostos de várias palavras. Neste caso o C&L deve dar preferência aos identificadores compostos de mais palavras.

Para exemplificar, suponhamos que exista um projeto com apenas um símbolo do léxico cadastrado, cujo nome seja “software”, e que estejamos visualizando o texto da Figura 51.

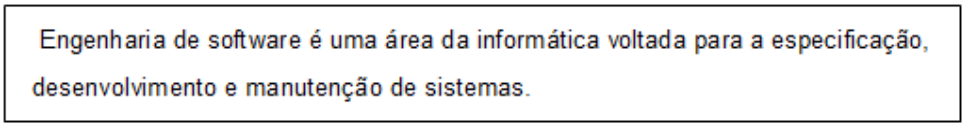


Figura 51 – Exemplo de texto sendo visualizado.

Neste caso, o termo “software” deveria ser substituído por um elo para o símbolo software (Figura 52).

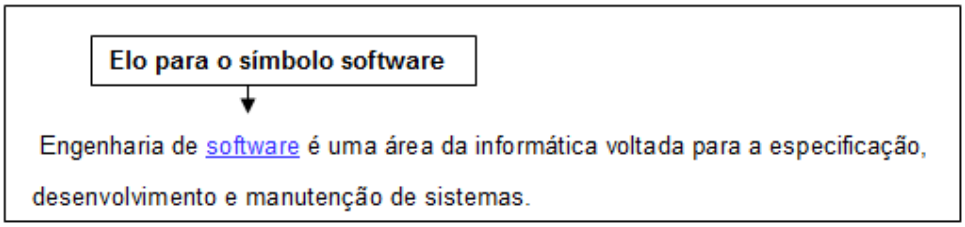


Figura 52 – Exemplo de texto com o elo software.

Suponhamos que mais adiante, o usuário cadastre um novo símbolo de nome “engenharia de software” no projeto, e que novamente estejamos visualizando o texto da Figura 51. Agora, o C&L deve substituir o termo “engenharia de software” por um elo para o símbolo “engenharia de software” e não teríamos mais uma referência ao termo “software” (Figura 53).

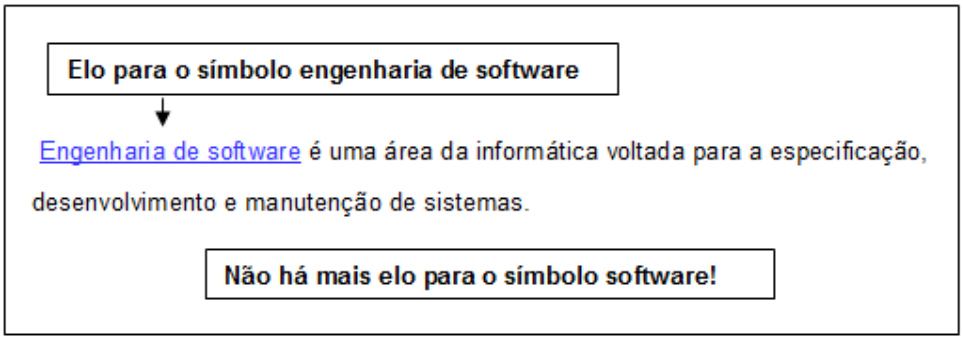


Figura 53 – Exemplo de texto com o elo engenharia de software.

5.2.6.2. Algoritmo antigo

Para facilitar, mostraremos, através de um exemplo, o funcionamento do algoritmo utilizado pelo C&L antes da re-engenharia e o erro que ocorria. Neste exemplo, iremos considerar um projeto com os seguintes símbolos do léxico cadastrados: “engenheiro”, “engenheiro de software”, “engenheiro de software experiente”, “software” e “engenharia de software”. Vamos considerar também que estamos visualizando o texto apresentado na Figura 54.

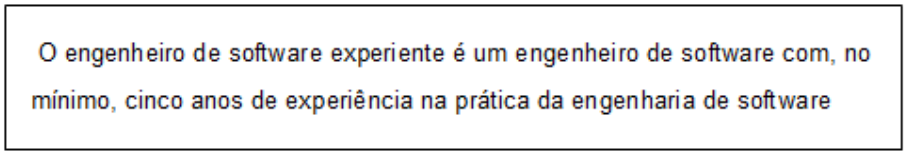


Figura 54 – Exemplo de texto sendo visualizado.

O primeiro passo do algoritmo é criar um vetor, colocar os identificadores dos elementos cadastrados no projeto neste vetor e ordená-los pelo número de palavras (Figura 55).

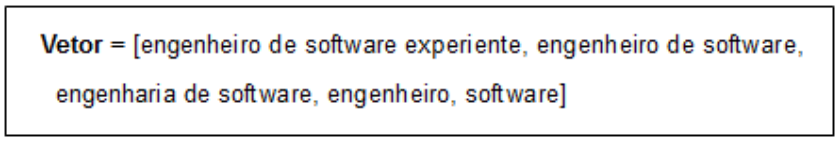


Figura 55 – Vetor de identificadores ordenado.

Depois da ordenação, o algoritmo percorre o vetor, começando do termo composto de mais palavras, e cada termo do vetor encontrado no texto é substituído por um elo para o elemento correspondente (Figura 56).

Figura 56 – Passos da aplicação do algoritmo antigo.

Podemos ver que no segundo e quarto passo, a *tag* HTML <a> (aqui simplificada sem seus atributos, apenas a título de exemplo), inserida para criar

o elo, fez com que o algoritmo funcionasse corretamente. Entretanto, no quinto passo a palavra “software” aparece no meio do termo “engenheiro de software experiente”, consequentemente sem a *tag* <a> para diferenciá-la. Por isso é criado um elo dentro de outro elo, o que ocasiona um erro. Este erro demorou muito para ser percebido, pois é uma situação que ocorre raramente no uso normal do software.

5.2.6.3. Algoritmo novo

Assim como feito anteriormente, também usaremos um exemplo para explicar o funcionamento do novo algoritmo. Para evidenciar as diferenças entre os dois algoritmos, utilizaremos o mesmo exemplo do algoritmo antigo.

O primeiro passo do novo algoritmo é, assim como o antigo, montar um vetor com todos os identificadores dos elementos cadastrados no projeto e ordená-los de acordo com seu número de palavras. Também devemos criar um vetor auxiliar inicialmente vazio. Este vetor será utilizado no decorrer do algoritmo.

Após a criação do vetor auxiliar, iremos percorrer o vetor ordenado a partir do identificador com maior número de palavras até os com menor número de palavras, procurando a ocorrência deles no texto. Caso seja encontrada uma ocorrência, o identificador do símbolo do léxico será colocado no vetor auxiliar e sua ocorrência no texto será substituída pelo código xzzxk*kxy, onde o * deverá ser substituído pela posição que o identificador ocupará no vetor auxiliar. Caso o identificador seja o título de um cenário, o procedimento será o mesmo só mudará o código utilizado, que deverá ser wzczxk*kxyyc, onde o * deve ser substituído pela posição que o identificador ocupará no vetor auxiliar. Os passos da aplicação deste algoritmo podem ser vistos na Figura 57.

Primeiro passo:

Vetor = [engenheiro de software experiente, engenheiro de software, engenharia de software, engenheiro, software]

Vetor auxiliar = [engenheiro de software experiente]

O *xzzxk1kxy* é um engenheiro de software com, no mínimo, cinco anos de experiência na prática da engenharia de software.

Segundo passo:

Vetor = [engenheiro de software experiente, engenheiro de software, engenharia de software, engenheiro, software]

Vetor auxiliar = [engenheiro de software experiente, engenheiro de software]

O *xzzxk1kxy* é um *xzzxk2kxy* com, no mínimo, cinco anos de experiência na prática da engenharia de software.

Terceiro passo:

Vetor = [engenheiro de software experiente, engenheiro de software, engenharia de software, engenheiro, software]

Vetor auxiliar = [engenheiro de software experiente, engenheiro de software, engenharia de software]

O *xzzxk1kxy* é um *xzzxk2kxy* com, no mínimo, cinco anos de experiência na prática da *xzzxk3kxy*.

Quarto passo:

Vetor = [engenheiro de software experiente, engenheiro de software, engenharia de software, engenheiro, software]

Vetor auxiliar = [engenheiro de software experiente, engenheiro de software, engenharia de software]

O *xzzxk1kxy* é um *xzzxk2kxy* com, no mínimo, cinco anos de experiência na prática da *xzzxk3kxy*.

Quinto passo:

Vetor = [engenheiro de software experiente, engenheiro de software, engenharia de software, engenheiro, software]

Vetor auxiliar = [engenheiro de software experiente, engenheiro de software, engenharia de software]

O *xzzxk1kxy* é um *xzzxk2kxy* com, no mínimo, cinco anos de experiência na prática da *xzzxk3kxy*.

Figura 57 – Passos da aplicação do algoritmo novo.

Após a execução destes cinco passos teremos um texto com códigos e o vetor auxiliar com os identificadores correspondentes a estes códigos. Para finalizar basta substituir os códigos por elos para os elementos correspondentes, como mostra a Figura 58.

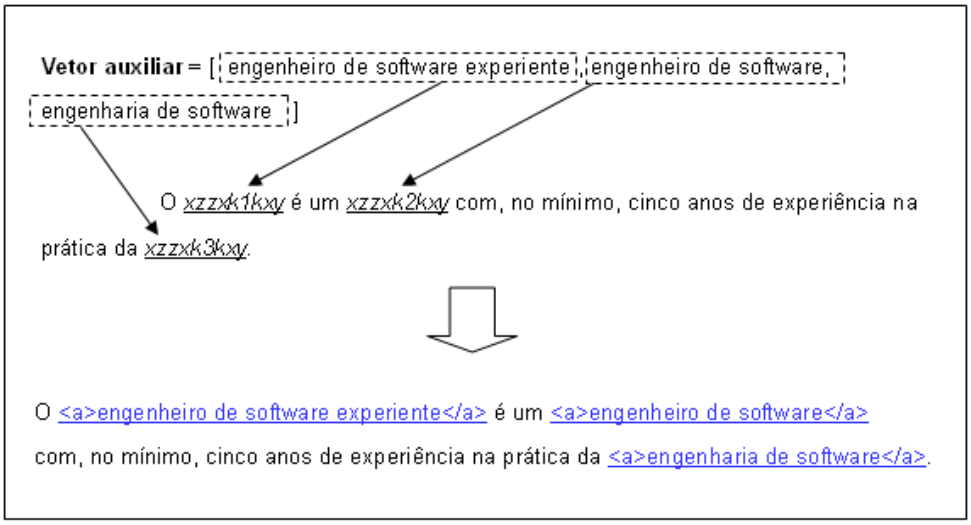


Figura 58 – Montagem dos elos do algoritmo novo.

A grande diferença do algoritmo novo para o antigo é a substituição das ocorrências dos identificadores no texto por códigos. Desta forma, ao realizarmos uma nova busca no texto, já não consideramos mais os identificadores que já foram encontrados, como acontecia anteriormente.

5.2.6.4. Implementação do algoritmo

A seguir mostraremos a operacionalização do cenário “Colocar atalhos no texto”, responsável pela descrição do algoritmo apresentado na subseção anterior. Este cenário faz uso do relacionamento de subcenários para destacar algumas funcionalidades e recuperar dados do banco de dados. Os subcenários “selecionar todos os léxicos do banco de dados”, “selecionar sinônimos do banco de dados” e “selecionar cenários do banco de dados” recuperam, respectivamente, todos os símbolos do léxico, sinônimos de símbolos e cenários de um determinado projeto. O subcenário “organizar tabela” organiza uma tabela de acordo com o número de palavras de seus elementos. E o subcenário “contar palavras” para determina o número de palavras um elemento específico. O relacionamento entre o cenário e seus subcenários pode ser visto na Figura 59.

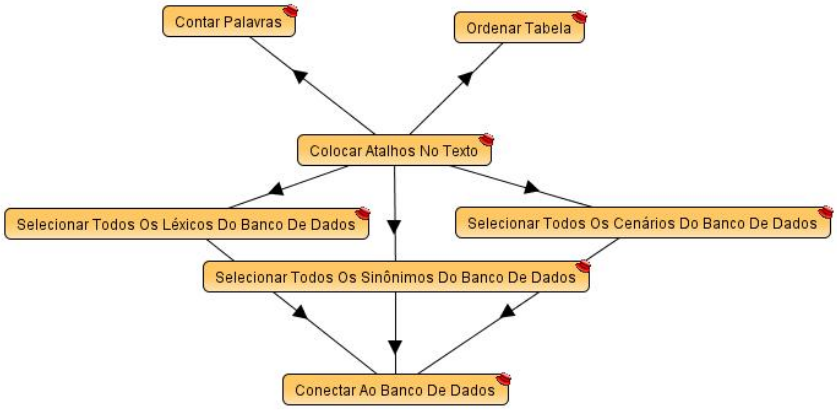


Figura 59 – Relacionamentos entre os cenários do algoritmo.

```
--[[
@Titulo: Colocar elos no texto
@Objetivo: Substituir a ocorrência de nomes de termos do léxico e seus sinônimos
ou de títulos de cenários, já cadastrado no projeto, por um atalho que leve para
definição do termo ou para descrição do cenário. Para tanto, cada elemento
cadastrado no projeto (termos do léxico e cenários) é procurado no texto, caso
alguma ocorrência seja encontrada é criado um atalho no texto para definição deste
elemento.
@Contexto: O usuário deve possuir um projeto com pelo menos um elemento
cadastrado.
@Atores: selecionar_todos_lexicos_bd, selecionar_sinonimos_projeto_bd,
selecionar_todos_cenarios_bd, ordenar_tabela, contar_palavras.
@Recursos: Identificador do projeto (parâmetro id_projeto), texto em que os
elementos serão procurados (parâmetro texto), tipo de elemento que foi selecionado
pelo usuário: termo do léxico ou cenário (parâmetro tipo) e banco de dados.
]]–
function colocar_elos (id_projeto, texto, tipo)

--@Episódio 1: SELECIONAR TODOS OS LEXICOS DO BANCO DE DADOS
       local lexicos = selecionar_todos_lexicos_bd(id_projeto);

--@Episódio 2: SELECIONAR TODOS OS SINÔNIMOS DO BANCO DE DADOS
       local sinonimos = selecionar_sinonimos_projeto_bd(id_projeto);

--@Episódio 3: SELECIONAR TODOS OS CENÁRIOS DO BANCO DE DADOS
       local cenarios = selecionar_todos_cenarios_bd(id_projeto);

--@Episódio 4: Criar uma tabela única, contendo nome dos termos do léxico e seus
siônimos.
       for index, sinonimo in pairs(sinonimos) do
              table.insert(lexicos, sinonimo);
       end
       local lexicos_e_sinonimos = lexicos;

--@Episódio 5: ORDENAR TABELA sinonimos.
       sinonimos = ordenar_tabela(sinonimos, 1, table.maxn(sinonimos) ,"lexico" )

--@Episódio 6: ORDENAR TABELA cenários.
       cenarios =  ordenar_tabela(cenarios, 1, table.maxn(cenarios) ,"cenario" )

--@Episódio 7: ORDENAR TABELA lexicos_e_sinonimos.
       lexicos_e_sinonimos = ordenar_tabela(lexicos_e_sinonimos, 1,
       table.maxn(lexicos_e_sinonimos),"lexico" )

--@Episódio 8: Obter o tamanho da tabela lexicos_e_sinonimos, da tabela cenários e
o total, que corresponde a soma dos tamanhos das duas tabelas.
       local tam_lexicos_e_sinonimos = #lexicos_e_sinonimos;
       local tam_cenarios = #cenarios;
       local tam_total = tam_lexicos_e_sinonimos + tam_cenarios ;

--@Episódio 9: Criar tabelas que irão armazenar os links criados para termos do
léxico e para cenários.
```

```
       local tabela_links_lexico = {};
       local tabela_links_cenario = {};

       if ((tipo == "lexico") or (tam_cenarios == 0)) then
--@Episódio 10: Se não há cenários cadastrados no projeto então apenas a lista de
léxicos e sinônimos será percorrida.
              for index, lexico in pairs(lexicos_e_sinonimos) do

--@Episódio 11: A variável nome_lexico recebe o nome do termo do léxico ou
sinônimo que está sendo verificado no momento.
                     local nome_lexico = lexico["NOME"];

--@Episódio 12: Uma expressão regular é montada com o nome do termo do léxico ou
sinônimo. Esta expressão regular evita erros devido a pontuação e espaços em
branco na hora da busca.
                     local expressao_regular =
                     "([%p%s])"..nome_lexico.."([%p%s])";

                     if (string.find(texto, expressao_regular) ~= nil) then

                            local nome_lexico_link = "";
--@Episódio 13: Se alguma ocorrência da expressão regular for encontrada no texto,
então esta expressão é substituída pelo código (wzzxk*kxy), onde o * será
substituído por um número que identificará a posição do termo do léxico ou
sinônimo na tabela léxicos e sinônimos.
                            texto = string.gsub(texto, expressao_regular,
                            "%1".."wzzxk"..index.."kxy".."%2");

--@Episódio 14: Verifica se a expressão encontrada é realmente um termo do léxico
ou é um sinônimo de um termo do léxico.
                            if (lexico["LEXICO"] ~= nil) then

--@Episódio 15: Se a expressão encontrada for um sinônimo então o nome_lexico_link
deve armazenar o nome do termo do léxico a que pertence o sinônimo. Isto é feito
para que possamos criar um atalho para o termo do léxico a que o sinônimo
corresponde.
                                   nome_lexico_link = lexico["LEXICO"]
                            else

--@Episódio 16: Se a expressão encontrada for um termo do léxico então o
nome_lexico_link deve armazenar o nome do termo do léxico.
                                   nome_lexico_link = nome_lexico;
                            end
--@Episódio 17: Criar o atalho para a expressão encontrada no texto.
                            link = "<a title=\"Léxico\"
                            href=\"../visao/exibe_lexico.lp?id_projeto=".."id_pro
                            jeto.."&nome=".."nome_lexico_link.."\">"..nome_lexico
                            .."</a>";

--@Episódio 18: Inserir o atalho criado na tabela_links_lexico na mesma posição
que o elemento encontrado ocupa na tabela lexicos_e_sinonimos. Isto permitirá que
identifiquemos a que código (wzzxk*kxy) do texto este atalho corresponde.
                            table.insert(tabela_links_lexico, index, link);
                     end --if

           end --for

       else

              if (tam_lexicos_e_sinonimos == 0 and tam_cenarios > 0) then

--@Episódio 19: Se não há léxicos no projeto então apenas a lista de cenários será
percorrida.
                     for index, cenario in pairs(cenarios) do

--@Episódio 20: A variável nome_cenario recebe o título do cenário que está sendo
verificado no momento.
                      nome_cenario = cenario["TITULO"];

--@Episódio 21: Uma expressão regular é montada com o título do cenário. Esta
expressão regular evita erros devido à pontuação e espaços em branco na hora da
busca.
                            expressao_regular =
"([%p%s])"..nome_cenario.."([%p%s])";

                            if (string.find(texto, expressao_regular) ~= nil)
then
```

--Episódio 22: *Se alguma ocorrência de expressão regular for encontrada no texto, então um atalho é montado com o nome do cenário.*

```
                                    link = "<a title=\"Cenário\"
                                    href=\"../visao/exibe_cenario.lp?id_projeto="
                                    ..id_projeto.."&titulo="..titulo_cenario.."&c
                                    omando=exibir\">"..titulo_cenario.."</a>";
```

--Episódio 23: *O atalho montado montado é inserido na tabela_links_cenario na mesma posição que o título do cenário encontrado ocupa na tabela cenarios.*

```
                                    table.insert(tabela_links_cenario, index,
link);
```

--@Episódio 24: *A expressão é substituída pelo código (wzczxk*kxyyc), onde o * será substituído por um número que identificará a posição do título do cenário na tabela cenários.*

```
                                    texto = string.gsub(texto, expressao_regular,
                                    "%1".."wzczxk"..i.."kxyyc".."%2");
                                end --if

                    end -- for

                elseif (tam_total > 0) then
```

--@Episódio 25: *Se há léxicos e cenários no projeto então tanto a tabela cenarios quanto a lexicos_e_sinonimos serão percorridas.*

```
                        i = 1;
                        j = 1;
                        contador = 1;
                        while (contador <= tam_total) do
```

--@Episódio 26: *Verifica se a tabela cenarios e a tabela lexicos_e_sinonimos já foram percorridas.*

```
                                if ( ( i <= tam_lexicos_e_sinonimos ) and (j <=
                                tam_cenarios) ) then
```

--@Episódio 27:*Se a tabela cenarios ainda não foi completamente percorrida e a tabela lexicos_e_sinonimos também não foi completamente percorrida então CONTAR PALAVRAS do título do cenário e CONTAR PALAVRAS do nome do léxico ou sinônimo.*

```
                                    if ( contar_palavras(cenarios[j]["TITULO"])
                                    <=
                                    contar_palavras(lexicos_e_sinonimos[i]["NOME"
                                    ]) ) then
```

--@Episódio 28: *Se o título do cenário atual possui um número menor ou igual de palavras que o nome do termo do léxico, então procuraremos a ocorrência deste léxico no texto.*

```
                                        nome_lexico =
                                        lexicos_e_sinonimos[i]["NOME"];
```

--@Episódio 29: *Uma expressão regular é criada com o nome do termo do léxico atual. Esta expressão regular impede que erros sejam causados por causa da pontuação e espaços em branco.*

```
                                        expressao_regular =
                                        "([%p%s])"..nome_lexico.."([%p%s])";

                                        if( string.gfind(texto,
                                        expressao_regular) ~= nil ) then
```

--@Episódio 30: *Se uma ocorrência da expressão regular é encontrada no texto então a expressão é substituída pelo código (wzzxk*kxy), onde o * será substituído por um número que identificará a posição do nome do termo do léxico ou sinônimo na tabela lexicos_e_sinonimos.*

```
                                            texto = string.gsub(texto,
                                            expressao_regular,
                                            "%1".."wzzxk"..i.."kxy".."%2")
                                            ;
```

--@Episódio 31: *É feita um verificação para saber se o termo encontrado é um termo do léxico ou um sinônimo de um termo do léxico.*

```
                                            if
                                            (lexicos_e_sinonimos[i]["LEXIC
                                            O"] ~= nil) then
```

--@Episódio 32: *Se o elemento encontrado for um sinônimo então o termo do léxico a que este sinônimo corresponde é armazenado na variável nome_lexico_link. Caso*

contrário, se o elemento encontrado for um termo do léxico, então ele é armazenado na variável nome_léxico.

```
                                                    nome_lexico_link =
                                                    lexicos_e_sinonimos[i]
                                                    ["LEXICO"];
                                                else
                                                    nome_lexico_link =
                                                    nome_lexico;
                                                end
```

--@Episódio 33: *Um atalho para o termo encontrado é criado e colocado na tabela links_lexico, na mesma posição que o léxico atual ocupa na tabela lexicos e sinônimos. Isto permitirá que identifiquemos a que código (wzzxk*kxy) do texto este atalho corresponde.*

```
                                                    link = "<a title=\"Léxico\"
                                                    href=\"../visao/exibe_lexico.l
                                                    p?id_projeto="..id_projeto.."&
                                                    nome="..nome_lexico_link.."\">
                                                    "..nome_lexico.."</a>";
                                                    table.insert(tabela_links_lexi
                                                    co, i, link);
                                            end --if
                                            i = i + 1;

                                        else --if
```

--@Episódio 34: *Se o título do cenário atual possui um maior de palavras que o nome do léxico atual, então procuraremos a ocorrência do título deste cenário no texto.*

```
                                            titulo_cenario =
cenarios[j]["TITULO"];
```

--@Episódio 35: *Uma expressão regular é criada com o título do cenário atual. Esta expressão regular impede que erros sejam causados por causa da pontuação e espaços em branco*.

```
                                            expressao_regular =
                                            "([%p%s])"..titulo_cenario.."([%p%s])
                                            ";


                                            if( string.gfind(texto,
                                            expressao_regular) ~= nil ) then
```

--@Episódio 36: *Caso encontremos uma ocorrência do cenário atual no texto, um atalho será criado e armazenado no tabela tabela_links_cenario, na mesma posição que o cenário atual ocupa na tabela cenários.*

```
                                                    link = "<a title=\"Cenário\"
                                                    href=\"../visao/exibe_cenario.
                                                    lp?id_projeto="..id_projeto.."
                                                    &titulo="..titulo_cenario.."\"
                                                    >"..titulo_cenario.."</a>";
                                                    table.insert(tabela_links_cena
                                                    rio, j, link);
```

--@Episódio 37: *A expressão é substituída pelo código (wzczxk*kxyyc), onde o * será substituído por um número que identificará a posição do título do cenário na tabela cenarios.*

```
                                                    texto = string.gsub(texto,
                                                    expressao_regular,
                                                    "%1".."wzczxk"..j.."kxyyc"..."%
                                                    2");

                                            end --if
                                            j = j + 1;

                                        end --if

                                    elseif( tam_lexicos_e_sinonimos == i-1 ) then
```

--@Episódio 38: *Se a tabela de termos do léxico chegou ao fim e a tabela de cenários ainda possui cenários que ainda não foram procurados, então devemos continuar percorrendo apenas a tabela cenários.*

```
                                        titulo_cenario = cenarios[j]["TITULO"];
```

--@Episódio 39: *Uma expressão regular é criada com o título do cenário atual. Esta expressão regular impede que erros sejam causados por causa da pontuação e espaços em branco.*

```
                                    expressao_regular =
                                    "([%p%s])"..titulo_cenario.."([%p%s])";

                                    if( string.gfind(texto, expressao_regular) ~=
                                    nil ) then

--@Episódio 40: Caso seja encontrada uma ocorrência do cenário atual no texto, um
atalho será criado e armazenado no tabela tabela_links_cenario, na mesma posição
que o cenário atual ocupa na tabela cenários.
                                            link = "<a title=\"Cenário\"
                                            href=\"../visao/exibe_cenario.lp?id_p
                                            rojeto="..id_projeto.."&titulo="..tit
                                            ulo_cenario.."\">"..titulo_cenario.."
                                            </a>";
                                            table.insert(tabela_links_cenario, j,
link);

--@Episódio 41: A expressão é substituída pelo código (wzczxk*kxyyc), onde o *
será substituído por um número que identificará a posição do título do cenário na
tabela cenarios.
                                            texto = string.gsub(texto,
                                            expressao_regular,
                                            "%1".."wzczxk"..j.."kxyyc".."%2");


                                    end --if

                                elseif( tam_cenarios == j-1 ) then

--@Episódio 42: Se a tabela de cenários chegou ao fim e a tabela
lexicos_e_sinonimos ainda possui elementos que ainda não foram procurados, então
devemos continuar percorrendo apenas a tabela lexicos_e _sinonimos.

                                    nome_lexico = lexicos_e_sinonimos[i]["NOME"];

--@Episódio 43: Uma expressão regular é criada com o termo do léxico atual. Esta
expressão regular impede que erros sejam causados por causa da pontuação e espaços
em branco.
                                    expressao_regular =
                                    "([%p%s])"..nome_lexico.."([%p%s])";

                                    if( string.gfind(texto, expressao_regular) ~=
                                    nil ) then

--@Episódio 44: Caso seja encontrada uma ocorrência do termo do léxico atual no
texto, A expressão é substituída pelo código (wzzxk*kxy), onde o * será
substituído por um número que identificará a posição do nome do termo do léxico na
tabela léxicos.
                                            texto = string.gsub(texto,
                                            expressao_regular,
                                            "%1".."wzzxk"..i.."kxy".."%2");


                                            if (lexicos_e_sinonimos[i]["LEXICO"]
                                            ~= nil) then

--@Episódio 45: Se o elemento encontrado for um sinônimo então o termo do léxico a
que este sinônimo corresponde é armazenado na variável nome_lexico_link. Caso
contrário, se o elemento encontrado for um termo do léxico, então ele é armazenado
na variável nome_léxico.
                                            nome_lexico_link =
                                            lexicos_e_sinonimos[i]["LEXICO"];
                                            else
                                                    nome_lexico_link =
nome_lexico;
                                            end

--@Episódio 46: Um atalho para o termo do léxico encontrado é criado e armazenado
no tabela tabela_links_lexico, na mesma posição que o léxico atual.
                                            link = "<a title=\"Léxico\"
                                            href=\"../visao/exibe_lexico.lp?id_pr
                                            ojeto="..id_projeto.."&nome="..nome_l
                                            exico_link.."\">"..nome_lexico.."</a>
                                            ";
                                            table.insert(tabela_links_lexico, i,
link);
```

```
                                    end --if

                                    i = i + 1;

                                end --if
                                contador = contador + 1;

                        end --while

                end --if
        end--if

        contador = 1;

--@Episódio 47: A tabela links lexico é percorrida e os códigos inseridos
anteriormente nos texto são substituídos pelos links armazenados na tabela. O
número que se encontra no meio do código corresponde a posição na tabela de links
que o atalho que será inserido se encontra.
        for i, link in pairs(tabela_links_lexico) do
                expressao_regular = ("([%p%s])".."wzzxk"..i.."kxy".."([%p%s])");
                texto = string.gsub(texto, expressao_regular, "%1"..link.."%2");
        end
--@Episódio 48: A tabela links cenários é percorrida e os códigos inseridos
anteriormente nos texto são substituídos pelos links armazenados na tabela. O
número que está no meio do código corresponde a posição na tabela de links que o
atalho que será inserido se encontra.
        for i, link in pairs(tabela_links_cenario) do
                expressao_regular = ("([%p%s])".."wzczxk"..i.."kxyyc".."([%p%s])");
                texto = string.gsub(texto, expressao_regular, "%1"..link.."%2");
        end
--@Episódio 49: O texto com os links é retornado para o usuário.
        cgilua.put(texto);

end
```

6
Conclusão

O termo transparência é muitas vezes utilizado no contexto da ciência da computação no sentido de ocultar informações e processos. Por exemplo, neste contexto, tornar uma funcionalidade transparente para o usuário significa deixá-la disponível sem que o usuário precise saber como ela funciona, apenas que faz o que o usuário espera. Nesse sentindo transparência é utilizada para enfatizar a facilidade de uso sem que haja preocupação com detalhes que poderiam demandar investimento de tempo pelos usuários. Neste trabalho utilizamos o termo transparência no sentido real da palavra, que é trazer luz ao que está oculto.

A transparência de software é um temo muito abrangente [Oliveira07], por isso neste trabalho procuramos focar especificamente na transparência do código fonte. Para atendermos ao requisito de transparência de código não basta disponibilizar o código fonte do software, também devemos fornecer meios que facilitem sua compreensão.

Ao procurarmos trabalhos relacionados na literatura descobrimos que a documentação de código fonte é baseada na adoção de estruturas pré-definidas, que armazenam descrições em linguagem natural, inseridas no código fonte. Estas estruturas são padronizadas para que permitam a atuação de ferramentas automatizadas na extração, tanto do código, quanto da documentação.

Nossa principal contribuição com este trabalho é um método de desenvolvimento baseado no refinamento de cenários. O objetivo deste método é produzir um documento único, o código fonte, onde teremos a nossa disposição a documentação, em forma de cenários integrados com o código, e o código propriamente dito. Este método também utiliza o *framework* MVC para uma melhor organização do código e o LAL para uma documentação complementar..

Outra contribuição importante foi o novo C&L, resultado do processo de re-engenharia realizado. Um software livre desenvolvido inteiramente no ambiente Lua-Kepler, com sua arquitetura dividida em camadas de acordo com o *framework* MVC e com cenários documentando seu código fonte, o que o tornou mais transparente a nível de código e, consequentemente, mais fácil de se

evoluir. Esperamos que o software mais transparente a nível de código atraia mais desenvolvedores interessados em participar de alguma forma do projeto.

E finalmente, acreditamos que esta experiência bem sucedida servirá de estímulo para futuros projetos de desenvolvimento de sistemas Web com o uso de Lua.

6.1. Comparação com trabalhos relacionados

Em [Knuth84] é proposta a linguagem WEB, que mistura uma linguagem de implementação com comentários. Com o uso de dois programas específicos é possível extrair de um arquivo WEB o código que será compilado e a documentação. Mas, segundo o próprio autor, esta linguagem foi propositalmente projetada para ser complexa, de modo que não pudesse ser usada por qualquer um, apenas por cientistas da computação.

Nosso objetivo ao desenvolver este trabalho foi totalmente o oposto. Adotamos o uso de cenários em linguagem natural como forma de anotação e de tecnologias como o *framework* MVC e Lua, para torná-lo simples e fácil de ser usado por qualquer um. Mas, diferente de [Knuth84], não possuímos uma ferramenta que extraia de forma automática a documentação do código fonte.

Seguindo a mesma linha, o trabalho exposto em [Cassino96] propõe um *framework* para construção de ferramentas de apoio a programação literária, e exemplifica a instanciação deste *framework* através do desenvolvimento de uma ferramenta chamada “nome”. A documentação final é apresentada em LaTeX [Lamport86].

Acreditamos que esta forma de documentação é menos dinâmica do que a que propomos, que permite a navegação através de elos. Além disto, como os cenários são inclusos no código em forma de comentários, eles podem ser compilados e executados juntamente com o código. Assim, só precisamos de um programa para extrair os cenários para a documentação, não há a necessidade de extração do código fonte.

O trabalho [Christoph04] propõe o uso de cenários [Leite00] como forma de anotar o código. Estes cenários podem ser construídos antes, depois ou concomitantemente com o código. A forma de anotação abordada é genérica, isto é, pode ser aplicada a softwares com qualquer arquitetura, cabe ao desenvolvedor adaptá-la as suas necessidades.

Em nosso trabalho utilizamos o mesmo modelo de cenários [Leite00], mas diferente de [Christoph04] nosso método é restrito a sistemas que pode ser descritos segundo o *framework* MVC. Além disto, em nossa proposta, os cenários são construídos antes do código fonte e então refinados durante o processo de escrita do código.

Os padrões [JavaDoc] e [PHPDoc] utilizam cabeçalhos com *tags* pré-definidas para anotar cada função. Nós utilizamos cenários para anotar não só funções, mas também arquivos e código HTML. Além disto, a utilização de cenários é muito mais abrangente, pois não se limita apenas a um cabeçalho. Os cenários, através de seus episódios, detalham passo a passo o funcionamento de todo o código.

Estes padrões permitem a transformação automática destes comentários em páginas HTML navegáveis. No nosso trabalho isto só é possível se houver um esforço extra para que os cenários inclusos no código sejam manualmente cadastrados na ferramenta C&L.

Em [Staa00] são definidos padrões para documentação de código através de comentários, para facilitar sua compreensão. Nosso trabalho também utiliza padrões como forma de documentação, estes padrões são os cenários. Esta padronização visa uniformizar o estilo de programação, facilitando o entendimento, manuseio e evolução de código escrito por terceiros.

6.2. Dificuldades encontradas

Encontramos dificuldades no início da codificação do novo C&L devido a falta de documentação e exemplos do uso da linguagem Lua para desenvolvimento de sistemas Web. Diferente de outras linguagens, como Java e PHP, que podemos obter respostas para a grande maioria de nossas dúvidas, de várias fontes diferentes, através de simples consultas na Web, as únicas fontes de documentação disponíveis são a página do próprio projeto Kepler e listas de discussão. Pouquíssima documentação foi encontrada de outras fontes e os exemplos são bem escassos e limitados.

Quando o nosso trabalho de re-engenharia do C&L começou, o projeto Kepler ainda estava em uma versão beta. Durante a codificação na versão beta encontramos falhas em alguns dos módulos do ambiente. Para resolvê-los, interagimos com a comunidade de desenvolvimento e outros usuários através da lista de discussão, reportando os problemas encontrados. Acompanhamos as

soluções adotadas e o lançamento da nova versão do ambiente já com as correções. Destacamos que a comunidade de desenvolvimento interage bastante com os usuários, e trabalha arduamente no sentido de resolver os problemas que aparecem o mais rapidamente possível, mas mesmo assim ainda leva um tempo considerável.

Durante a construção da nova arquitetura tivemos dificuldade em manter a integridade entre os cenários cadastrados no C&L e os cenários inclusos do código fonte. Toda vez que evoluíamos o cenário do código fonte, tínhamos que evoluir o mesmo cenário cadastrado no C&L e vice-versa. Desta forma, toda alteração teve que ser feita duas vezes, o que gerou um considerável trabalho extra.

6.3. Trabalhos Futuros

Como descrito na subseção anterior, uma das dificuldades encontradas no decorrer deste projeto foi manter a integridade dos cenários inclusos no código fonte e os cenários cadastrados no software C&L. Para resolver este problema pretendemos estender o C&L, permitindo que os cenários sejam importados automaticamente do código fonte para dentro do software.

A navegabilidade entre os cenários que compõem o software só é possível quando se utiliza o C&L para visualizá-los. Um de nossos projetos futuros é a criação de um ambiente de programação que desse suporte a utilização de cenários, permitindo a navegação através dos cenários no próprio código fonte.

Pretendemos também fazer a integração entre o LAL do domínio da aplicação e o LAL do espaço de nomes. Esta integração permitirá um rastreamento que possibilitará a identificação dos conceitos do domínio da aplicação que deram origem aos nomes que compõem o espaço de nomes. Um dos benefícios desta integração seria a possibilidade de verificar se os nomes foram escolhidos adequadamente.

Por fim, dois trabalhos importantes que pretendemos realizar futuramente são a utilização do método por outras pessoas e a realização de experimentos comparativos. Com isto esperamos determinar o nível de complexidade do método proposto e se há algum ganho com a sua utilização, comparado com o desenvolvimento sem o uso de nenhum método e com o auxílio de outros métodos.

Referências bibliográficas

[Apache09] The Apache Software Foundation - Disponível em: http://www.apache.org. Acesso em: 02/02/2009.

[Archermann00] Achermann F. Nierstrasz O.; Explicit Namespaces, In Modular programming Languages, volume 1897 of LNCS.

[Biggerstaff94] Biggerstaff, T. J., Bharat, G. M., Webster, D. E.; Program Understanding and the Concept Assignment Problem. Commun. ACM 37(5):72-82(1994).

[Blank05] Blankenhorn, D.; Open Source Transparency; Buscado em:28/outubro/2008;URL:http://mooreslore.corante.com/archives/2005/04/19/open_source_transparency.php

[Breitman00] Breitman, K.K; Leite, J.C.S.P.; Scenario Evolution: A Close View on Relationships – In: 4 International Conference on Requirements Engineering (ICRE'00), Schaumburg, illinois, 2000 - IEEE Computer Society. pp 102-111.

[Brooks77] Brooks. R; "Towards a theory of the cognitive processes in computer Programming", In: International Journal of Man-Machine Studies, Vol. 9, pp. 737-75I, 1977.

[Carroll94] Carroll, J., Alpert, S., Karat, J., Van Deusen, M., Rosson, M. Raison d'etre; Capturing design history and rationale in multimedia narratives. In: Human Factors in Computing Systems (CHI94) - Boston, USA: ACM Press, 1994. pp 192-197.

[Cassino96] Cassino, C; Uma Ferramenta para Programação Literária Modular. Tese de Mestrado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, 1996.

[Christoph04] Christoph R. H. (2004); Engenharia de Software para Software Livre. Tese de Mestrado. Pontifícia Universidade Católica do Estado do Rio de Janeiro, Departamento de Informática, Brasil.

[C&L09] C&L – Cenários e Léxicos – Disponível em: http://pes.inf.puc-rio.br/cel. Acesso em: 14/03/2009.

[CVS09] Concurrent Version System – Disponível em: http://www.nongnu.org/cvs/. Acesso em: 14/03/2009.

[Fsf02] Free Software Foundation Europe; Carta de Direitos Fundamentais da UE; Disponível em: http://www.fsfeurope.org/documents/fp6/recommendation.pt.pdf Acesso em: 28/novembro/2006.

[GrupoER09] Grupo de Engenharia de Requisitos da PUC-Rio – Disponível em: http://www.er.les.inf.puc-rio.br/grupoER/. Acesso em: 10/03/2009.

[Hsia94] Hsia, P.et al; Formal Approach to Scenario Analysis – IEEE Software, vol. 11 no.2, 1994. pp 33-41.

[Ierusalimschy03] Ierusalimschy, R; “Programming in Lua.” Published by Lua.org ISBN 85-903798-1-7, Paperback, 288 pages, Distributed by Ingram and Baker & Taylor, December 2003. Available in http://www.lua.org/pil/ (2008)

[JavaDoc09] JavaDoc – Disponível em: http://java.sun.com/j2se/javadoc/ Acesso em: 13/03/2009

[JavaScript09] JavaScript – Disponível em: http://www.w3schools.com/JS/ Acesso em: 15/0302009.

[Kepler09] Kepler Project – Disponível em: http://keplerproject.org. Acesso em: 15/03/2009.

[Knuth84] Knuth, D. E.; Literate Programming. The Computer Journal, vol. 27, nº 2, pp 97-111.

[Lamport86] L. Lamport. LaTeX - a document preparation system. Addison-Wesley Publishing Company, 1986.

[Leite93] Leite, J.C.S.P. and Franco, A.P.M.; A Strategy for Conceptual Model Acquisition. First International Symposium on Requirements Engineering - IEEE Computer Society Press, 1993. pp 243-246.

[Leite94] Leite, J.C.S.P. – Engenharia de requisitos: Notas de aula. Disponível em: http://livrodeengenhariaderequisitos.googlepages.com/ERNOTASDEAULA.pdf

[Leite96] Leite, J.C.S.P.ACM Software Engineering Notes: Vol. 21, N. 2, Pags. 39 - 44, (1996),Working Results on Software Re-Engineering.

[Leite98] Leite, J.C.S.P. and Leonardi, M.C.; Business Rules as Organizational Polices, 9th IWSSD, Proceedings of the International Workshop of Specification and Design, IEEE Computer Society Press, pp. 68-76 (1998).

[Leite00] Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.; "A Scenario Construction Process" - Requirements Engineering Journal, Vol.5, N° 1, 2000, Pages 38-61.

[Leite06] J.C.S.P. Leite, "Transparência: Desafios para a Engenharia de Software", Sexta-Feira, May 19th, 2006 http://jcspl.wordpress.com/2006/05/19/transparencia-desafios-para-aengenharia-de-software/

[Leite08] Leite, J.C.S.P.; Serrano, M; Almentero, E. K. Taming the Namespace Problem. - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, 2008.

[Lecesne07] Lecesne, Pierre Carlos Eduardo. Desenho de grafos em um ambiente Web. 2007. Projeto final de graduação - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, 2007.

[LuaDoc] LuaDoc – Disponível em: http://www.tecgraf.puc-rio.br/~tomas/luadoc/luadoc. html

[Meyrhauser94] Meyrhauser, A and Vans A. M. “Program Understanding - A Survey.” CS94-120, Department of Computer Science, Colorado State University, August 1994.

[MySQL09] MySQL – Disponível em: http://mysql.com. Acesso em: 10/03/2009.

[Neuman89] Neuman, B.C. 1989. The need for closure in large distributed systems. SIGOPS Oper. Syst. Rev. 23, 4 (Oct. 1989), 28-30.

[Oliveira07] Oliveira, A.P, et. al., Engenharia de requisitos intencional: tornando o software mais transparente, Tutorial presented at SBES´07, http://www.sbbd-sbes2007.ufpb.br/tuto3.pdf

[PHP09] PHP: Hypertext Preprocessor – Disponível em: http://php.net. Acesso em: 20/01/2009.

[PHPDoc09] PHPDoc – Disponível em: http://www.phpdoc.org/ Acesso em: 13/03/2009

[Semantic Designs09] Disponível em: http://www.semdesigns.com/products/DMS/DMSTTollkit.html?Home=SoftwareReengineering. Acesso em 10/03/2009.

[Sayão06] Sayão, M., Leite, J.C.S.P.; Rastreabilidade de Requisitos - Revista de Informática Teórica e Aplicada RITA 13(1): 57-86 (2006).

[Silva03] Silva, L. F., Sayão, M., Leite, J.C.S.P., Breitman, K. K.; Enriquecendo o Código com Cenários; XVII Brazilian Symposium on Software Engineering (SBES2003), Manaus- AM, 2003.

[Silva04] Silva, L. F.; Leite J.C.S.P.; Breitman, K.K.; Ensino de engenharia de software: Relato de experiências; Workshop de Educação em Informática (WEI – 2004), 12, Salvador, 2004, pp. 994-10055.

[Silva05] L. F. Silva, J.C.S.P. Leite, K.K. Breitman – C&L uma Ferramenta de Apoio à Engenharia de Requisitos – Revista de Informática Teórica e Aplicada (RITA), Vol.XII, Número 1, Junho 2005, ISSN 0103-4308.

[Staa00] Staa, A. Programação Modular: Desenvolvendo Programas Complexos de Forma Organizada e Segura., Apêndice 4, Campus, 2000.

[Willis90] Willis, D. 1990. The Lexical Syllabus London: Collins ELT.