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.

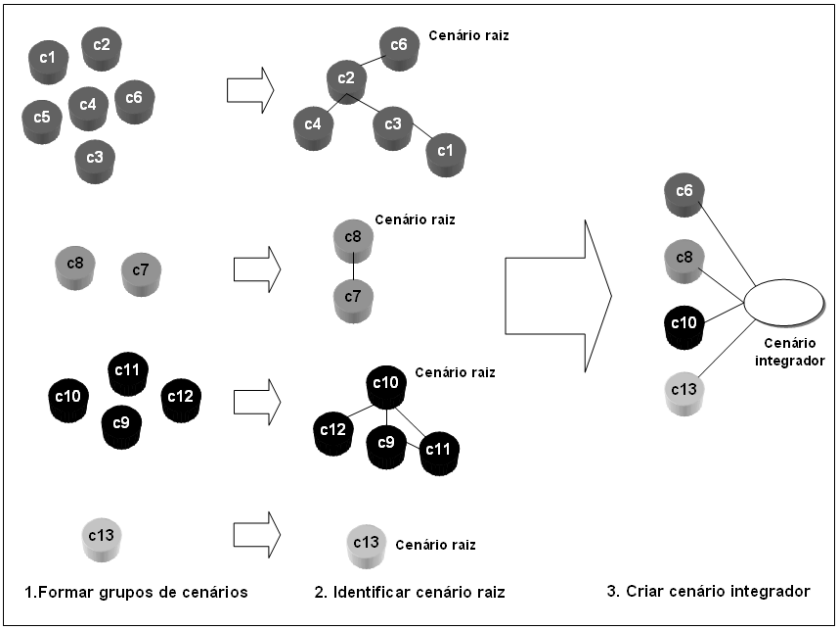
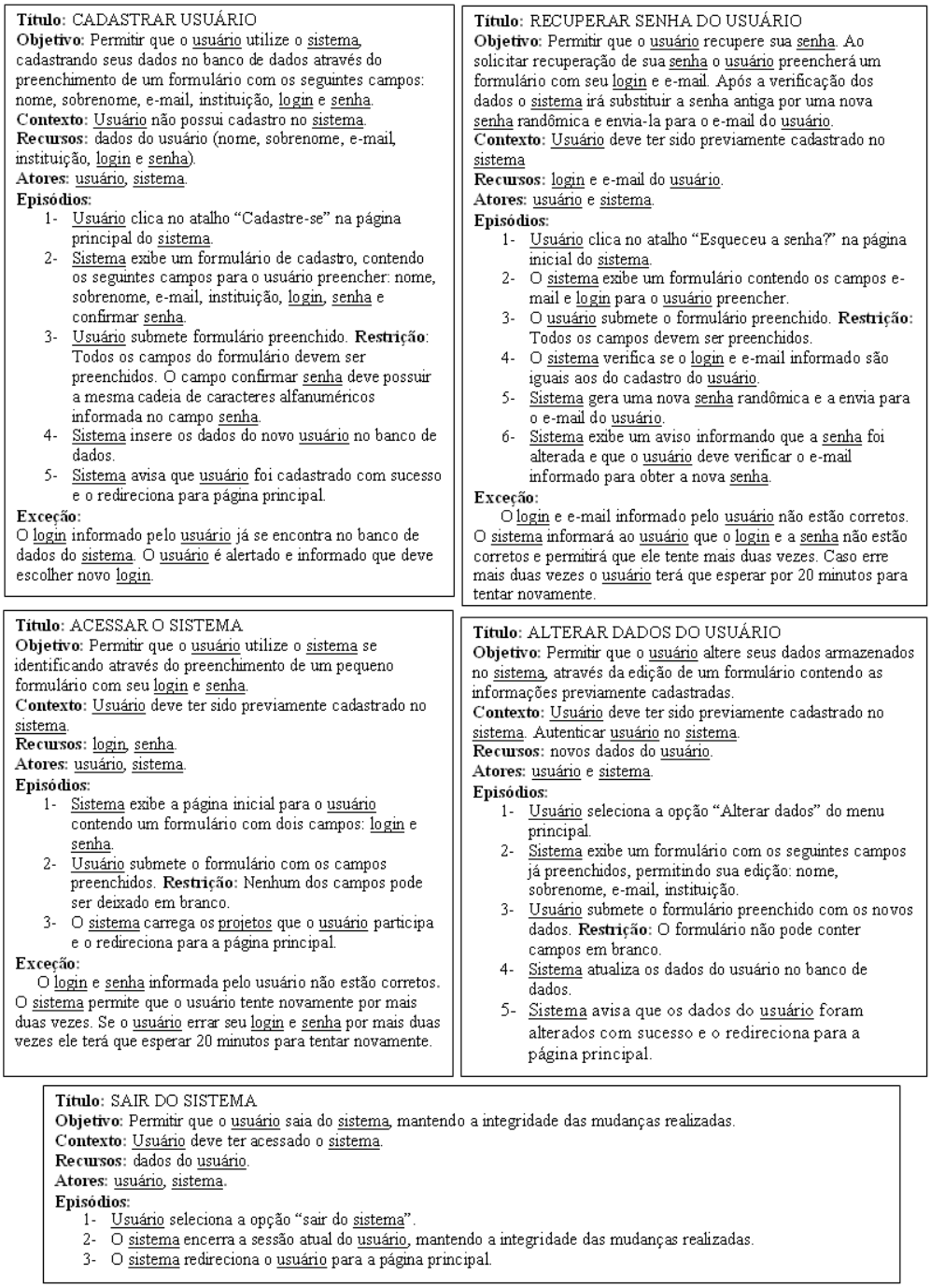


Figura 22 – Processo de integração de cenários.

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).

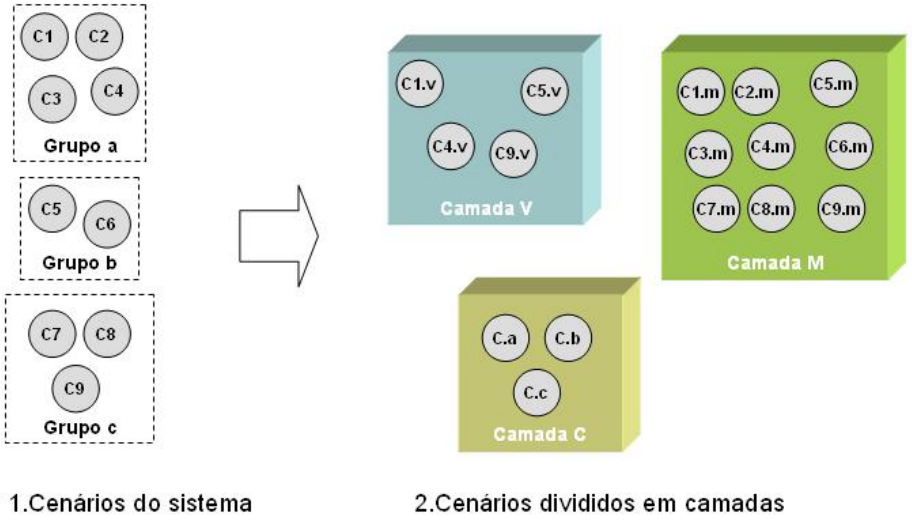


Figura 26 – Processo de refinamento dos cenários em camadas.

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.