Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Formalização de um Modelo de Processo de Reengenharia Centrado no Usuário para Conversão de Aplicações Desktop em RIAs

David Buzatto

São Carlos

2010

Formalização de um Modelo de Processo de Reengenharia Centrado no Usuário para Conversão de Aplicações Desktop em RIAs

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

David Buzatto

Formalização de um Modelo de Processo de Reengenharia Centrado no Usuário para Conversão de Aplicações Desktop em RIAs

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de mestre em Ciência da Computação.

Orientação: Profa. Dra. Junia Coutinho Anacleto

São Carlos

2010

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

B992fm

Buzatto, David.
Formalização de um modelo de processo de reengenharia centrado no usuário para conversão de aplicações desktop em RIAs / David Buzatto. -- São Carlos : UFSCar, 2010.
73 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2010.

1. Reengenharia de software. 2. Projeto centrado no usuário. 3. Interação homem - máquina. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Formalização de um Modelo de Processo de Reengenharia Centrado no Usuário para Conversão de Aplicações Desktop em RIAs"

DAVID BUZATTO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Profa. Dra. Junia Coutinho Anacleto
(Orientadora - DC/UFSCar)

Prof. Dr. Antonio Francisco do Prado
(DC/UFSCar)

Prof. Dr. Wilson Vicente Ruggiero
(POLI/USP)

São Carlos
Maio/2010

Dedico este trabalho a minha namorada Fernanda, a minha mãe Selma, aos meus amigos do mestrado, em especial aos companheiros do laboratório e a todos aqueles que me apoiaram e me incentivaram durante a pesquisa e desenvolvimento deste trabalho.

Agradecimentos

Agradeço primeiramente a minha namorada Fernanda, por sempre me ouvir e ficar ao meu lado durante o desenvolvimento deste trabalho e ao longo destes sete anos de namoro.

A minha família, principalmente minha mãe, que sempre apoiou minhas decisões e contribuiu para que eu sempre seguisse o caminho correto.

Aos meus amigos e companheiros de laboratório Marcos, Gilberto, Johana, Ana Luiza e Bruno, por sempre me ouvirem e me apoiarem a buscar o que eu queria.

A todos os amigos da turma de 2008.

A professora Junia, primeiramente por ter acreditado no meu potencial ao me selecionar no processo seletivo do mestrado e por sempre ter estado disposta a me apoiar no desenvolvimento deste trabalho, incentivando a maioria das minhas decisões em relação à reengenharia do Cognitor e à condução da minha pesquisa.

Aos professores que tive oportunidade de conviver e conhecer durante o período das disciplinas.

Agradeço também a CAPES pelo suporte financeiro que permitiu que eu me dedicasse exclusivamente à minha pesquisa.

Por fim, agradeço a todas as pessoas que estiveram ao meu lado me apoiando, direta ou indiretamente.

Muito obrigado a todos!

"*Any fool can write code that a computer can understand. Good programmers write code that humans can understand*".

Martin Fowler

Resumo

A reengenharia de software se faz importante devido à necessidade que as organizações têm em se adequar às novas tendências, tecnologias e exigências dos usuários. Inclui-se ao termo "organizações", empresas ou universidades que desenvolvem softwares para serem utilizados por um grande número de pessoas. Pensando na adequação das exigências dos usuários, este trabalho apresenta a reengenharia de um software chamado Cognitor, que é uma ferramenta criada para apoiar os educadores no processo de criação de material didático eletrônico. A percepção da necessidade da reengenharia do Cognitor se deu através de um estudo de caso onde foram relatadas várias alterações que o software deveria sofrer, tais como: melhoria no editor de texto, pré-visualização das imagens que são inseridas nas páginas de conteúdo, *feedback* ao usuário, entre outras. Durante a reengenharia desse software, consequentemente, foi formalizado um modelo de processo de reengenharia de software, centrado no usuário, para a conversão de aplicações *desktop* em RIAs (*Rich Internet Application*), denominado UC-RIA (*User Centered Rich Internet Application*). Foi dado o nome de UC-RIA ao modelo de processo devido à participação dos potenciais usuários da aplicação durante o processo de reengenharia, pois estes estiveram envolvidos tanto na prototipação, quanto na validação das interfaces gráficas da nova versão. Os resultados obtidos neste trabalho mostram a potencialidade do modelo de reengenharia de software UC-RIA em ser utilizado como apoio às organizações durante a reengenharia de seus softwares, principalmente por inserir os usuários no processo de reengenharia durante a fase de Prototipação da aplicação, aproximando os softwares às reais necessidades dos usuários.

Palavras-chave: Reengenharia de Software. UC-RIA. Cognitor. UCD. IHC. SPEM.

Abstract

The software reengineering becomes important because of the need that organizations have in adjusting to new trends, technologies and user requirements. The term "organization" should be understood like universities or companies that develop software for use by a large number of people. Thinking about the adequacy of users' requirements, is presented in this work the reengineering process of a software called Cognitor, which is a tool designed to support teachers in the process of creating electronic teaching materials. The need to reengineer Cognitor was perceived through a case study where several changes were pointed by users. Some of these changes are: text editor improvement, preview of the images that are inserted in the content pages, feedback to the users, among others. During the reengineering of this software, it was formalized a software reengineering process model, user-centered, for the conversion of desktop applications in RIAs (Rich Internet Application), called UC-RIA (User Centered Rich Internet Application). The process model was named as UC-RIA due to the participation of the potential users during the application's reengineering process, because they were involved in prototyping and in validation of the graphical interfaces of the new version. The results of this study show the capability of the proposed software reengineering model to be used as a support in organizations for the reengineering of their software, mainly because it inserts the users in the reengineering process during the application's Prototyping phase, bringing software to users' real needs.

Keywords: Software Reengineering. UC-RIA. Cognitor. UCD. IHC. SPEM.

Sumário

Lista de Abreviaturas

AJAX	*Asynchronous JavaScript and XML*
APEL	*Abstract Process Engine Language*
API	*Application Programming Interface*
EAD	Educação à Distância
EPF	*Eclipse Process Framework*
HTML	*Hypertext Markup Language*
IHC	Interação Humano Computador
LIA	Laboratório de Interação Avançada
LMS	*Learning Management System*
LOM	*Learning Object Metadata*
LP	Linguagem de Padrões
MIT	*Massachusetts Institute of Technology*
MOF	*Meta Object Facility*
OA	Objeto de Aprendizagem
OMCS	*Open Mind Common Sense*
OMCS-Br	*Open Mind Common Sense* no Brasil
OMG	*Object Management Group*
PML	*Process Modeling Language*
PROMENADE	*Process-oriented Modelling and Enactment of Software Developments*
RIA	*Rich Internet Application*
RUP	*Rational Unified Process*
SCORM	*Shareable Content Object Reference Model*
SIGDOC	*Special Interest Group on the Design of Communication*
SPEM	*Software Process Engineering Meta-model*
UC-RIA	*User Centered Rich Internet Application*
UCD	*User Centered Design*
UML	*Unified Modeling Language*
WYSIWYG	*What You See Is What You Get*
XP	*Extreme Programming*

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

Com o passar do tempo o mercado de trabalho vem se tornando cada vez mais competitivo, fazendo com que as empresas tenham a necessidade de mudar constantemente. Para aumentar a qualidade nas empresas, existe a necessidade de se pensar todo o tempo em como melhorar a forma de se fazer negócio. A necessidade de mudança também é visível na Computação, principalmente se tratando dos softwares. A partir do momento em que um software começa a ser utilizado, ele entra em um estado contínuo de mudança. Mesmo que tenha sido construído aplicando as melhores técnicas de projeto e codificação existentes, os softwares vão se tornando obsoletos em vista das novas tecnologias que são disponibilizadas. Além das correções de erros, as mudanças mais comuns que os softwares sofrem são migrações para novas plataformas, ajustes para mudanças de tecnologia de hardware ou sistema operacional e extensões em sua funcionalidade para atender os usuários.

Segundo Osborne e Chikofsky [7], a variedade de problemas que envolvem manutenção de software cresce constantemente, sendo que as soluções não acompanham essa evolução. Estes problemas são resultantes de código fonte e documentação mal elaborados, além da falta de compreensão do sistema. Quando um sistema não é fácil de ser mantido, mesmo que sua utilidade seja grande, ele deve ser reconstruído. Partindo-se do sistema existente (via código-fonte, interface ou ambiente), são abstraídas as suas funcionalidades e são construídos os modelos de análise e de projeto do software. Para esse processo, dá-se o nome de reengenharia de software.

No entanto, são escassos os trabalhos que propõem um modelo de processo de reengenharia, principalmente os que levam em consideração fatores relacionados à IHC (Interação Humano Computador). Díscola Junior e Silva [8, 9] propõem um processo de planejamento para a reengenharia de software guiado por avaliação de usabilidade, entretanto este processo trata do planejamento e não da reengenharia em si. Sendo assim,

existe a necessidade de se propor modelos de processo de reengenharia que tenham como objetivo formalizar a conversão de uma aplicação fonte[1] em uma aplicação alvo[2], levando em consideração os aspectos que competem à IHC, como usabilidade, por exemplo, e não somente os aspectos relacionados à (re)engenharia de software. Afinal, o termo reengenharia está relacionado com a reconstrução de algo do mundo real, e independentemente de sua aplicação, o seu principal propósito é a busca por melhorias que permitam produzir algo de qualidade melhor ou, pelo menos, de qualidade comparável ao produto inicial.

Desta maneira, neste trabalho é proposto o desenvolvimento de um modelo de processo de reengenharia de software centrado nos usuários, que possa ser utilizado pelas empresas, podendo aumentar as chances de sucesso na versão final do software que passou pelo processo de reengenharia. Para alcançar esse resultado, este trabalho baseou-se em dois modelos de reengenharia de software, juntamente com o UCD (*User Centered Design*).

1.2 Objetivos

O objetivo deste trabalho é formalizar um modelo de processo de reengenharia de software centrado no usuário, que tem como objetivo guiar a conversão de aplicações *desktop* em RIAs (*Rich Internet Application*), formalizado a partir da observação do modelo de processo que se configurou durante o processo de reengenharia de uma aplicação *desktop* – chamada Cognitor – para uma aplicação rica para a Internet. Para isso, neste trabalho é apresentado o modelo de processo UC-RIA (*User Centered Rich Internet Application*).

Além da formalização do modelo de processo citado, neste trabalho é apresentada a nova versão da aplicação Cognitor e também aborda as mudanças que foram feitas em sua interface, sendo que estas foram propostas pelos usuários que fizeram parte do processo de reengenharia. Também é explorada a questão do apoio ao preenchimento dos metadados dos OAs (Objeto de Aprendizagem) por meio da utilização do conhecimento cultural disponibilizado pelo Projeto OMCS-Br, que tem como objetivo coletar este tipo de conhecimento da população brasileira [10].

[1] A aplicação fonte é a versão da aplicação que ainda não passou pelo processo de reengenharia.
[2] A aplicação alvo é a versão da aplicação fonte que passou pelo processo de reengenharia.

1.3 Metodologia

Para verificar o quanto os usuários podem ser importantes durante uma reengenharia de software, neste trabalho foi desenvolvido um modelo de processo de reengenharia de software que inclui os usuários durante o processo de reengenharia. Para alcançar esse resultado, foi adotada neste trabalho uma abordagem centrada no usuário, baseada na proposta apresentada por Anacleto, Fels e Villena [5], que por sua vez se baseia no UCD, ou Projeto Centrado no Usuário, idealizado por Norman e Draper [11], onde os usuários de um determinado artefato (construção civil, objeto do dia a dia, software, etc) têm papel central na tomada de decisão relacionada ao desenvolvimento do mesmo, opinando e avaliando o que está sendo construído.

Desta maneira, durante todo o processo de formalização do UC-RIA, houve a preocupação de inserir o usuário no processo de reengenharia, tentando tornar assim a nova versão do software mais usável, agradável e dentro dos critérios elaborados por Shneiderman et al. [12], tais como: satisfação do usuário, facilidade de aprendizado, entre outros.

1.4 Organização do Trabalho

Além deste Capítulo introdutório, este trabalho encontra-se organizado em outros cinco Capítulos: no Capítulo 2 são discutidos dois modelos de processo de reengenharia de software que foram importantes para a formalização do modelo de processo proposto neste trabalho; no Capítulo 3 é apresentado o UC-RIA, detalhando suas características e todo o processo que culminou em sua formalização; a versão Web do Cognitor é apresentada no Capítulo 4, detalhando suas interfaces e funcionalidades, além de apresentar sucintamente o Projeto OMCS-Br que disponibiliza uma base de conhecimento cultural, coletada da população brasileira, que pode ser usada no apoio à contextualização de aplicações computacionais; no Capítulo 5 é feita uma comparação entre as diversas funcionalidades das versões *desktop* e Web do Cognitor, além de apresentar as novas funcionalidades que precisaram ser desenvolvidas na nova versão da aplicação; por fim, no Capítulo 6, é apresentada a conclusão deste trabalho e os possíveis trabalhos futuros.

2 *Modelos de Processo de Reengenharia de Software*

2.1 Considerações Iniciais

Existe na literatura diversos trabalhos que descrevem modelos de processo para a reengenharia de sistemas computacionais. Neste Capítulo é apresentado um estudo sobre os principais modelos de processo encontrados e suas respectivas características.

Este Capítulo está organizado da seguinte maneira: na Seção 2.2 são apresentadas as definições de modelo de processo e de reengenharia de software; na Seção 2.3 são discutidos alguns modelos formais para a definição de modelos de processo de engenharia de software, que por sua vez podem ser utilizados para a definição de modelos de processo de reengenharia de software; na Seção 2.4 é apresentado o estado da arte em relação aos modelos de processo de reengenharia de software existentes; na Seção 2.5 é feita uma análise do cenário apresentado na Seção 2.4; por fim, na Seção 2.6, são apresentadas as Considerações Finais.

2.2 Definições

A reengenharia de software consiste na análise e na alteração de um sistema de software com o objetivo de criar uma nova representação para o sistema em questão, sendo que esta nova representação é formada por todos os artefatos relacionados ao sistema, como sua documentação, código fonte, etc [13]. A reengenharia não serve apenas para criar uma nova representação do sistema, mas também para criar uma representação melhor do que a original. No entanto, para descrever o processo de reengenharia, é necessário um modelo que o guie.

Desta maneira, na próxima Seção, serão apresentados quais os caminhos que devem ser seguidos, bem como os recursos existentes, para que um modelo de processo de enge-

nharia ou de reengenharia de software, seja formalizado de acordo com um vocabulário padronizado.

2.3 Como Definir um Modelo de Processo para Engenharia de Software

Normalmente cada empresa mantém um modelo de processo específico que foi utilizado na criação de um determinado software [14], entretanto esta criação de modelos de processo específicos pode acarretar em certa confusão no entendimento dos modelos, pois cada equipe pode usar uma notação particular para modelá-los e nos casos onde a notação é padrão apenas dentro de uma determinada empresa, esta notação pode não ser a mesma que a usada por outra.

Com o objetivo de resolver este problema de padronização, diversas PMLs[1] (*Process Modeling Language*) têm sido propostas. Alguns exemplos de PMLs podem ser citadas, tais como: APEL [15] (*Abstract Process Engine Language*), PROMENADE [16] (*Process-oriented Modelling and Enactment of Software Developments*) e SPEM [1] (*Software Process Engineering Meta-model*), sendo que a SPEM é a iniciativa mais recente na tentativa de padronizar um vocabulário para a especificação de modelos de processo, além de ser uma especificação desenvolvida pelo OMG (*Object Management Group*) e também por existir uma implementação de referência, o EPF[2] (*Eclipse Process Framework*). Por estes motivos, neste trabalho será dado maior enfoque na PML SPEM, discutida na próxima Seção.

2.3.1 SPEM

A PML SPEM é uma iniciativa do OMG que tem como objetivo padronizar a notação para a especificação de modelos de processo, permitindo assim uma definição formal dos processos e de seus componentes. O metamodelo do SPEM é baseado no MOF (*Meta Object Facility*), um meta-metamodelo também mantido pelo OMG. Na Figura 2.1 pode ser vista uma arquitetura em camadas dos modelos do OMG, que vai desde a camada mais abstrata (M3) até a camada mais concreta (M0).

[1] Uma PML é uma linguagem utilizada para modelar e descrever modelos de processo.
[2] `http://www.eclipse.org/epf/`

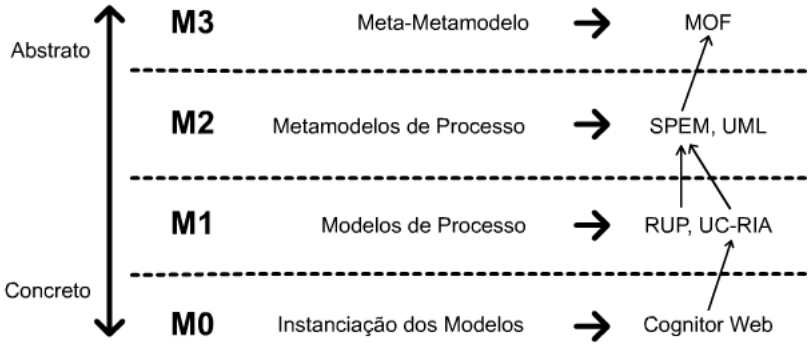


Figura 2.1: Camadas dos modelos OMG (adaptado de [1])

Como pode ser visto na Figura 2.1, o MOF se encontra na camada mais abstrata. Na camada seguinte (M2) estão situados o SPEM e a UML (*Unified Modeling Language*). Na M1, estão situados os modelos de processo, como RUP (*Rational Unified Process*), XP (*Extreme Programming*) e o UC-RIA (*User Centered Rich Internet Application*), modelo de processo criado como parte deste projeto e que será detalhado no Capítulo 3. Por fim, na camada M0, a mais concreta de todas, se encontram as aplicações criadas utilizando algum modelo de processo da camada M1. No exemplo da Figura 2.1 é mostrado que o Cognitor Web (que será apresentado no Capítulo 4) está situado na camada M0, pois ele é uma aplicação que foi criada com base no UC-RIA.

O SPEM é organizado em sete pacotes, apresentados na Figura 2.2.

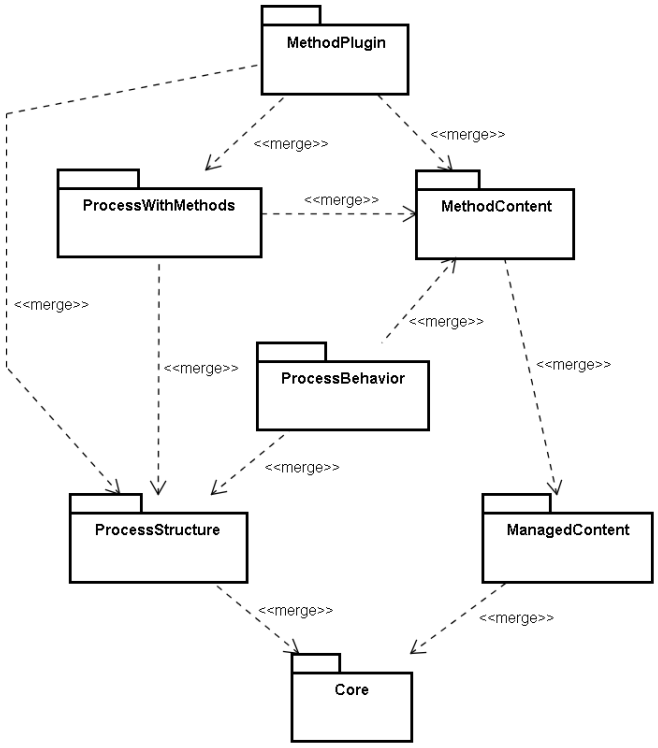


Figura 2.2: Estrutura do SPEM [1]

Cada pacote possui uma função particular, conforme descrito a seguir:

- ***Core***: O pacote *Core* fornece as classes base para todos os pacotes do metamodelo;
- ***Process Structure***: Este pacote é a base para todos os modelos de processo. Sua estrutura é formada por um conjunto de atividades (*Activities*) aninhadas, que por sua vez mantém referências para os papéis (*Roles*) que as executam, além dos artefatos (*Work Products*) gerados e consumidos em cada atividade;
- ***Process Behavior***: O pacote *Process Behavior* é utilizado para estender as estruturas do pacote *Process Structure* com o objetivo de prover comportamento para os modelos de processo. Um exemplo seria a ligação de um modelo com um diagrama de atividades da UML;
- ***Managed Content***: Este pacote permite que sejam gerenciadas descrições textuais do processo de desenvolvimento;
- ***Method Content***: O pacote *Method Content* descreve como alcançar os objetivos de desenvolvimento, especificando os papéis envolvidos, recursos e resultados correspondentes, independentemente da colocação destes passos dentro de um ciclo de vida de desenvolvimento específico;
- ***Process With Methods***: Este pacote provê a integração entre os processos criados através do pacote *Process Structure* com as instâncias dos conceitos gerados pelo uso do pacote *Method Content*;
- ***Method Plugin***: O pacote *Method Plugin* introduz os conceitos necessários para projetar e gerenciar bibliotecas manuteníveis e reusáveis de conteúdos (*Method Content*) e processos.

Toda esta estrutura é implementada computacionalmente no EPF, tornando fácil o uso do SPEM, pois o EPF provê diversos assistentes que permitem a criação dos componentes de um modelo de processo, bem como a organização do processo criado. O EPF foi utilizado neste trabalho para a formalização do UC-RIA, que será discutida no próximo Capítulo.

Com base no que foi apresentado, pode-se dizer que o SPEM é um metamodelo que permite que sejam criados modelos de processo, formalizando para isso os componentes dos processos (atividades, papéis e artefatos), a integração entre componentes e processos

e a estruturação dos processos através de diagramas de atividades da UML. Na próxima Seção, serão apresentados os modelos de processo de reengenharia considerados relevantes para este trabalho.

2.4 Estado da Arte em Modelos de Processo de Reengenharia de Software

Diversos modelos de processo de reengenharia de software vêm sendo propostos ao longo dos anos. Dentre alguns dos modelos existentes [2], [3], [17], [18] e [19], este trabalho descreve o Modelo de Byrne [2] e o Modelo em Espiral Dupla (*Dual-Spiral Reengineering Model*) [3], pois o primeiro apresenta uma base teórica do que é a reengenharia de software e as fases que a compõe e o segundo, propõe um modelo que usa duas espirais (baseado no Modelo em Espiral de Boehm [20]), sendo que uma destas espirais é usada para a engenharia reversa enquanto a outra – executada de forma síncrona com a primeira – é usada para a engenharia avante.

2.4.1 Modelo de Byrne

O Modelo de Byrne [2] foi desenvolvido por Eric J. Byrne na tentativa de formalizar uma base conceitual para a reengenharia de software. No Modelo de Byrne, a reengenharia de um sistema é apresentada como um processo formado por três princípios – Princípio da Abstração, Princípio da Alteração e Princípio do Refinamento.

1. **Princípio da Abstração**: O aumento gradual no nível de abstração de uma representação de um sistema é criado a partir da sucessiva troca de informações mais detalhadas por outras mais abstratas. A abstração é a engenharia reversa, ou seja, é a criação de níveis mais abstratos com base em níveis mais concretos. Um exemplo, seria a criação de um diagrama de classes de nível de projeto a partir do código da aplicação fonte. O Princípio da Abstração pode ser visto na parte azul da Figura 2.3;

2. **Princípio da Alteração**: Este princípio está relacionado à alteração de uma determinada representação do sistema, mas sem alterar o nível de abstração. Por exemplo, a adição de um atributo em uma determinada classe de um diagrama no nível do projeto. Este exemplo do Princípio da Alteração é representado pela flecha "reprojetar" na Figura 2.3;

3. **Princípio do Refinamento**: Este princípio é o inverso do Princípio da Alteração, pois está relacionado à diminuição do nível de abstração de uma determinada representação do sistema. Pode-se citar como exemplo a utilização de um diagrama de classes de nível de projeto para a implementação das classes representadas neste diagrama. O refinamento é a engenharia avante, podendo ser visto na parte verde da Figura 2.3.

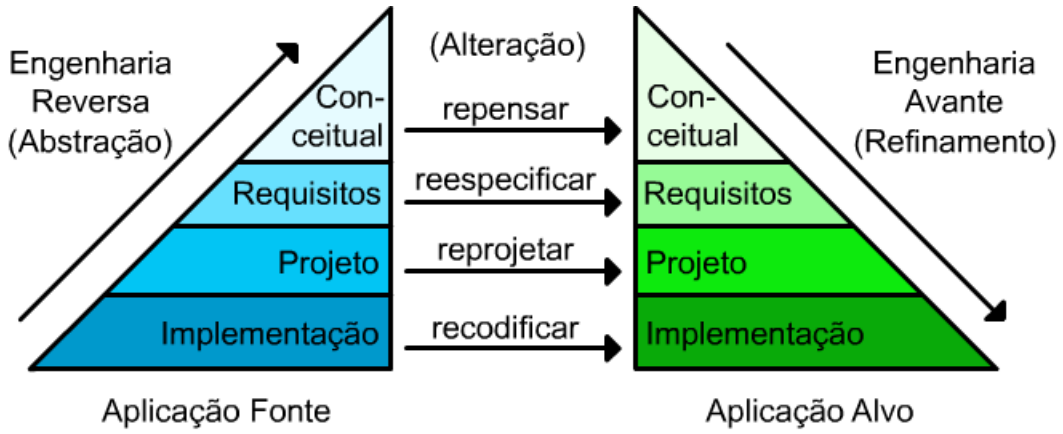


Figura 2.3: Estrutura do Modelo de Byrne (adaptado de [2])

Como pode ser observado, o Modelo de Byrne formaliza o ciclo de reengenharia de uma aplicação e, a partir do modelo, podem ser derivadas três estratégias de reengenharia:

1. **Estratégia "Reescrever"**: Esta estratégia incorpora apenas o Princípio da Alteração, sendo que para alterar um sistema, é preciso transformar a aplicação fonte – representada em algum nível de abstração – na aplicação alvo (no mesmo nível de abstração). Por exemplo: a partir do código da aplicação fonte, reescrevê-la, criando assim a aplicação alvo. Na Figura 2.4 pode ser vista a Estratégia "Reescrever";

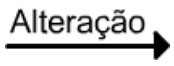


Figura 2.4: Estratégia de reengenharia "Reescrever" [2]

2. **Estratégia "Melhorar"**: A Estratégia "Melhorar" incorpora os três princípios do Modelo de Byrne: Abstração, Alteração e Refinamento. Para que uma característica da aplicação fonte seja alterada, é necessário, primeiramente, trabalhar a abstração até o nível desejado, fazer a alteração para o nível correspondente da aplicação alvo e, por fim, utilizar o refinamento para alcançar o nível de abstração correspondente na aplicação alvo em relação ao nível de início. Por exemplo, a alteração no código

da aplicação fonte deve ser refletida em todos os níveis mais abstratos. As alterações devem ser feitas para cada nível e, por fim, deve ser feito o refinamento até chegar ao nível de implementação. Na Figura 2.5 pode ser vista a Estratégia "Melhorar";

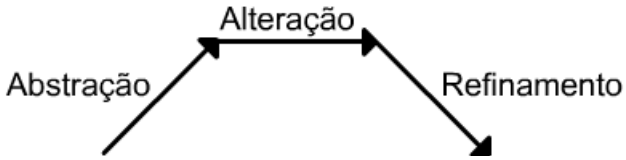


Figura 2.5: Estratégia de reengenharia "Melhorar" [2]

3. **Estratégia "Substituir"**: A Estratégia "Substituir" (composta pelos princípios Abstração e Refinamento), consiste na abstração de cada nível da aplicação fonte e em seguida no refinamento de cada um dos níveis da aplicação alvo até chegar ao nível de implementação. Na Figura 2.6 pode ser vista a Estratégia "Substituir";

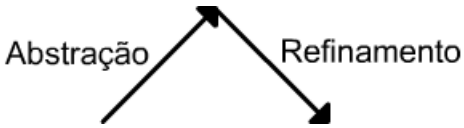


Figura 2.6: Estratégia de reengenharia "Substituir" [2]

Pelo exposto, pode-se notar que o Modelo de Byrne fornece um modelo conceitual para a reengenharia de software, entretanto ele não trata da mudança de tecnologias entre a aplicação fonte e a aplicação alvo (por exemplo, fazer a reengenharia de uma aplicação em C++ para uma aplicação em Java). Este modelo não se preocupa em sincronizar o processo de reengenharia entre as aplicações quando não se pode parar uma delas até que a outra fique pronta. Com o intuito de solucionar estes problemas, o Modelo de Reengenharia em Espiral Dupla foi proposto por Yang et al. [3].

2.4.2 Modelo de Reengenharia em Espiral Dupla

O Modelo de Reengenharia em Espiral Dupla [3] é representado por duas espirais, sendo que, uma delas modela a engenharia reversa da aplicação fonte, enquanto a outra, representa a engenharia avante da aplicação alvo. Ele pode ser aplicado nos processo de reengenharia onde a aplicação fonte não pode ser "aposentada" e a aplicação alvo precisa ser desenvolvida de forma concorrente. Neste modelo de processo, a reengenharia é feita por funcionalidade, isto é, o processo se inicia com a engenharia reversa de uma funcionalidade e, a partir da engenharia reversa, é feita então a engenharia avante. Quando

a funcionalidade passa totalmente pelo processo de reengenharia, ela é "desligada" na aplicação fonte e passa a ser utilizada exclusivamente na aplicação alvo. Um exemplo seria o de uma a aplicação fonte que permite o cadastro de usuários. O primeiro passo é fazer a engenharia reversa desta funcionalidade e ao fim desta etapa, fazer a engenharia avante, implementando assim a funcionalidade na aplicação alvo. Quando esta encontra-se implementada, ela é desativada na aplicação fonte e passa a ser utilizada na aplicação alvo. Um exemplo gráfico da reengenharia de uma aplicação – que para simplificar contém apenas três funcionalidades – pode ser visto na Figura 2.7. Uma característica importante deste modelo é permitir que cada uma das aplicações utilizem tecnologias e arquiteturas distintas, pois sua divisão é baseada em funcionalidade.

Como pode ser visto na Figura 2.7, as funcionalidades vão sendo migradas da aplicação fonte para a aplicação alvo. Durante o processo de reengenharia, é adicionada uma camada de comunicação (*proxy*) entre as duas aplicações e no final do processo esta camada é removida.

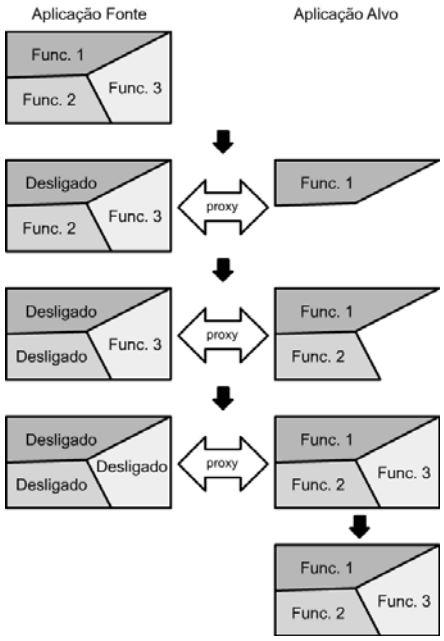


Figura 2.7: Estrutura do Modelo em Espiral Dupla, instanciado em uma aplicação com três funcionalidades (Adaptado de [3])

Na Figura 2.8, são apresentadas as duas espirais do modelo de processo, sendo que a espiral detalhada em azul é a espiral da engenharia reversa, enquanto a espiral em verde é a espiral da engenharia avante.

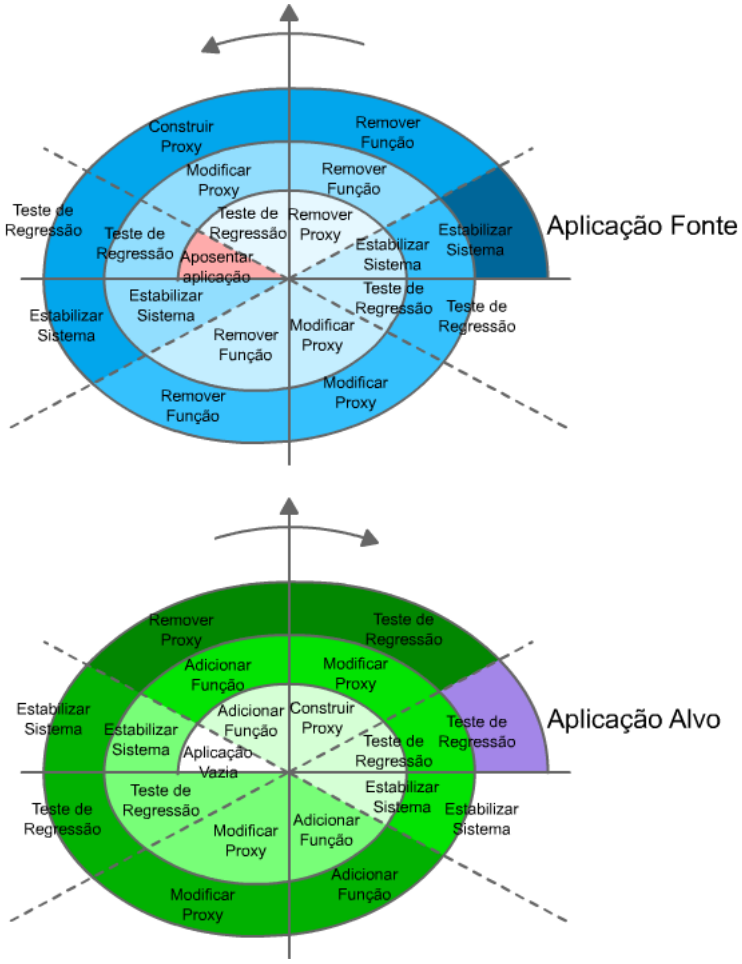


Figura 2.8: Estrutura do Modelo em Espiral Dupla (Adaptado de [3])

Na especificação formal do modelo, ele é dividido em três fases: na primeira fase, as funcionalidades da aplicação fonte são identificadas. Na segunda, o processo em espiral é realizado até que todas as funcionalidades tenham sido migradas. Por fim, a terceira fase especifica que quando o processo em espiral termina e a aplicação alvo é testada e validada pelos usuários, a aplicação fonte deve ser "aposentada" e a aplicação alvo deve entrar em produção na sua totalidade. Para as duas primeiras fases, os autores do modelo propõem uma série de equações que formalizam a divisão das funcionalidades e a conversão de cada uma delas.

A seguir será apresentada a comparação entre estes dois modelos estudados.

2.5 Análise do Cenário Apresentado

O estudo dos modelos de processo de reengenharia de software citados nas Seções anteriores, em especial o Modelo de Byrne e o Modelo em Espiral Dupla, viabilizou o en-

tendimento da estrutura dos modelos de processo de reengenharia de software. O ponto forte no Modelo de Byrne é a possibilidade dele ser usado como base para criar modelos de processo de reengenharia, pois ele formaliza as fases do processo, bem como lista as estratégias possíveis na aplicação de um modelo de reengenharia. O outro modelo de processo estudado (Espiral Dupla), utiliza o conceito do Modelo em Espiral proposto por Boehm [20], na construção de duas espirais, sendo uma responsável pela organização da engenharia reversa, enquanto a outra pela engenharia avante. As espirais são sincronizadas entre si, pois cada fase finalizada em uma das espirais (engenharia reversa) é ponto de partida para uma fase na outra espiral (engenharia avante). Além de modelar esta transição entre as espirais, o modelo também diz que a aplicação que está passando pelo processo de reengenharia deve ser convertida na aplicação alvo em partes pequenas (funcionalidades) e que cada uma das aplicações devem coexistir, comunicando-se entre si, até que todo o processo de reengenharia seja finalizado e a aplicação fonte possa ser definitivamente "aposentada". A seguir, na Tabela 2.1, são apresentadas as principais características dos dois modelos, identificando aquelas que são vantagens (prós) e as que são desvantagens (contras).

Tabela 2.1: Características dos modelos de processo de reengenharia de software estudados

Modelo	**Prós**	**Contras**
Modelo de Byrne	É um modelo genérico, que formaliza uma base conceitual para a reengenharia de software.	Não diz nada a respeito sobre a reengenharia onde a aplicação fonte usa uma tecnologia enquanto na aplicação alvo será utilizada outra tecnologia.
	Especifica diversos princípios que modelam as possíveis fases de um modelo de processo de reengenharia.	Por ser um modelo genérico, muitos detalhes do processo de reengenharia ficam subentendidos.
	Define uma série de estratégias para reengenharia de software.	
Modelo em Dupla Espiral	Possui um formalismo que especifica as fases do processo de reengenharia.	Não deixa claro onde a camada de comunicação entre as duas aplicações deve ficar.
	Permite a sincronização dos processos de engenharia reversa e engenharia avante.	Não é tratada a possibilidade de não se usar a camada de comunicação entre as aplicações nos casos onde não há necessidade desta camada existir. Por exemplo, uma aplicação que vai passar pelo processo de reengenharia, mas que pode ser usada normalmente enquanto a nova aplicação está sendo construída.
	A utilização do modelo em espiral facilita a adição de novas funcionalidades na aplicação alvo.	
	Abstrai as tecnologias utilizadas na aplicação fonte e na aplicação alvo.	
	Define que as aplicações devem conversar entre si durante o processo de reengenharia, migrando gradualmente as funcionalidades da aplicação fonte para a aplicação alvo.	

Pode-se dizer que tais modelos, mesmo com as desvantagens do Modelo de Byrne, podem ser utilizados como base para a criação de outros modelos de processo de reengenharia, que levem em consideração outras características (como a reengenharia das interfaces gráficas da aplicação fonte) que não são tratadas nestes modelos. Na próxima Seção serão discutidas as Considerações Finais deste Capítulo.

2.6 Considerações Finais

Neste Capítulo foi discutido o SPEM, uma PML mantida pela OMG, que tem como objetivo padronizar uma notação para a especificação de modelos de processo de engenharia de software, permitindo, assim, que os modelos de processo criados por qualquer equipe sejam entendidos facilmente por outras, pois existe uma padronização do vocabulário utilizado na modelagem do processo. Além do SPEM, foram abordados também dois modelos de processo de reengenharia de software: o Modelo de Byrne, um modelo genérico para reengenharia, e o Modelo em Espiral Dupla, que expande o conceito do desenvolvimento em espiral, proposto por Boehm [20], na forma de duas espirais, uma para a engenharia reversa da aplicação fonte e outra para a engenharia avante da aplicação alvo.

O estudo destes modelos, bem como do SPEM, foi importante para a formalização do modelo de processo de reengenharia proposto neste projeto, o UC-RIA. No próximo Capítulo, o processo de formalização UC-RIA será apresentado e discutido.

3 O Modelo de Processo UC-RIA

3.1 Considerações Iniciais

O UC-RIA (*User Centered Rich Internet Application*) é um modelo de processo de reengenharia de software centrado no usuário para conversão de aplicações *desktop* em RIAs (*Rich Internet Application*). A formalização do UC-RIA se deu à partir da percepção do modelo que surgiu depois do processo de reengenharia da versão *desktop* da ferramenta Cognitor. Neste Capítulo será descrito o processo de formalização do UC-RIA sendo que para isso ele está organizado da seguinte maneira: na Seção 3.2 são apresentadas as características do UC-RIA; na Seção 3.3 são descritas as fases que compuseram o processo de criação e formalização do UC-RIA; a comparação entre o UC-RIA e os modelos de reengenharia de software – apresentados no Capítulo 2 – é apresentada na Seção 3.4; por fim, na Seção 3.5, são apresentadas as Considerações Finais.

3.2 Características

O UC-RIA é um modelo de reengenharia de software criado com o objetivo de propor um modelo de processo para a conversão de aplicações *desktop* convencionais em aplicações ricas para a Internet, as chamadas RIAs. No UC-RIA a etapa de desenvolvimento das interfaces gráficas com o usuário é pautada pela prototipação, uma das técnicas que podem ser empregadas no UCD (*User Centered Design*). O UCD pode ser considerado como um método utilizado para projetar ferramentas computacionais – em forma de aplicações Web – para uso humano, sendo que o processo do projeto envolve a participação das pessoas que serão os usuários da ferramenta [21]. O UCD, portanto, se preocupa em colocar os usuários – e suas necessidades – como o centro das decisões de projeto.

O UCD é composto por três fases: na primeira, denominada *design research*, o objetivo do projetista é identificar quais são os usuários potenciais da aplicação que será criada. Na segunda fase, chamada *design*, o projetista, baseado nos resultados da fase anterior,

projeta as interfaces (utilizando protótipos, por exemplo) e outros artefatos que ele julgar serem importantes. Na terceira e última fase, *design evaluation*, o projetista avalia o *design* em conjunto com os usuários e revisa o que foi feito com base nos resultados obtidos. Um estudo mais detalhado sobre o UCD pode ser encontrado no trabalho de Williams [21].

Para que uma aplicação *desktop* seja convertida em uma aplicação Web convencional é necessário mudar o modelo de interação com o usuário, utilizando, para isso, formulários, requisições síncronas, etc, no entanto, esta mudança no modelo de interação – do usuário com a aplicação – pode não ser bem aceita, pois este tem que "reaprender" a utilizar a ferramenta por causa das mudanças que foram realizadas, principalmente na interface gráfica. Este problema começou a ser contornado a partir do momento em que o Google começou a utilizar requisições assíncronas em suas aplicações (Gmail e Google Maps), dando a estas aplicações uma cara de aplicação *desktop*, permitindo que o usuário interagisse com componentes de interface gráfica que até o momento não eram utilizados em uma aplicação Web.

Com isso, surge então o conceito de RIA, que são aplicações Web que oferecem responsividade[1] e "funcionalidades ricas" que as tornam parecidas com aplicações *desktop* [22]. Esta responsividade é alcançada pelo uso do AJAX (*Asynchronous JavaScript and XML*), que basicamente consiste em requisições assíncronas entre a aplicação Web e o servidor, não havendo a necessidade de recarregar as páginas HTML (*Hypertext Markup Language*). As "funcionalidades ricas" são alcançadas por meio da utilização de bibliotecas de componentes para a construção de interfaces gráficas. Sendo assim, no UC-RIA, as aplicações que serão criadas devem ser construídas da maneira mais parecida possível com a aplicação *desktop* original.

Pelo o que foi apresentado, a utilização da prototipação – apoiada sempre pelos usuários da ferramenta – e a criação de aplicações ricas são as principais características do UC-RIA. Na próxima Seção serão apresentadas as fases que permitiram a formalização do UC-RIA.

[1] O termo responsividade, tradução livre de "*responsiveness*", pode ser entendido como a velocidade da resposta dada por uma aplicação computacional.

3.3 O Processo de Criação do UC-RIA

O processo de criação e formalização do UC-RIA se deu a partir da percepção do modelo que surgiu durante o processo de reengenharia de uma ferramenta *desktop* chamada Cognitor. Originalmente o objetivo deste projeto era prover à ferramenta novas funcionalidades, sendo que a principal mudança estaria relacionada ao suporte dado ao preenchimento de metadados dos OAs (Objeto de Aprendizagem) gerados por ela. Mais detalhes sobre a nova versão do Cognitor, além de outros conceitos relacionados à ferramenta – metadados e OAs – serão discutidos no próximo Capítulo.

Além da implementação das novas funcionalidades relacionadas aos metadados, foi decidido também que:

1. A ferramenta deveria ser convertida em uma aplicação Web, permitindo assim que qualquer pessoa a utiliza-se, sem a necessidade de instalar a aplicação;
2. Os OAs gerados na ferramenta deveriam ser armazenados em um repositório central, pois a versão *desktop* não tinha tal funcionalidade (os OAs eram gerados como arquivos específicos da ferramenta);
3. A ferramenta deveria manter o perfil de cada usuário, além de prover a cada um deles uma área pessoal dentro do repositório de OAs.

Em razão destas mudanças na ferramenta, houve a necessidade de fazer a reengenharia da mesma. Todo o processo de reengenharia até a formalização do UC-RIA se deu em três fases que serão detalhadas nas próximas Seções.

3.3.1 Fase 1 – Análise e Geração de Código

A primeira fase do processo de criação do UC-RIA foi constituída da reengenharia da aplicação *desktop* (aplicação fonte). Nesta fase, foram analisadas as funcionalidades da aplicação fonte por meio da sua versão executável, do seu código fonte e de diversos artigos que descreviam a ferramenta. A partir disso, foram então analisadas diversas tecnologias que poderiam ser empregadas na construção da versão Web (aplicação alvo), sendo que o enfoque principal nesta análise era verificar se existiam bibliotecas de componentes gráficos (para criação de RIAs), que proveriam ao desenvolvedor os recursos necessários para criar a interface gráfica da nova aplicação de forma mais parecida com a original.

A escolha das possíveis bibliotecas se deu pela análise de algumas características: quantidade de componentes, quão ativa era a comunidade que dava suporte à biblioteca, quantidade de documentação e de exemplos disponíveis, compatibilidade entre navegadores, independência de plataforma e licença de uso. Com as bibliotecas escolhidas, diversos protótipos simples (janelas, diálogos, formulários, etc) foram criados para definir então, dentre a lista de possíveis bibliotecas, qual seria a biblioteca utilizada na nova versão da ferramenta. A biblioteca escolhida neste caso foi a ExtJS[2], pois é uma biblioteca bem documentada, que tem grande suporte da comunidade e principalmente por ser escrita totalmente utilizando JavaScript, sendo suportada pela maioria dos navegadores atuais sem a necessidade de instalar nenhum *plugin*. A partir deste momento, iniciou-se a concepção da nova versão da ferramenta, culminando assim, após aproximadamente cinco meses de desenvolvimento, na versão Web do Cognitor[3].

3.3.2 Fase 2 – Tabela com Percepção do Modelo

A segunda fase do processo de criação do UC-RIA se deu pela organização de todo o caminho percorrido, durante a reengenharia do Cognitor, em forma de uma tabela, derivada da metodologia formalizada por Anacleto, Fels e Villena em [5]. Esta metodologia, baseada no UCD, especifica que a tabela deve conter algumas colunas:

1. **Iteração ou Fase**: Um número para identificar uma determinada iteração ou fase;
2. **Estratégia de Projeto**: O que vai ser feito em uma determinada iteração ou fase;
3. **Perguntas a Serem Respondidas**: Quais as perguntas que devem ser respondidas em relação à estratégia adotada;
4. **Protótipo**: Um protótipo que usa a estratégia definida para responder às perguntas;
5. ***Stakeholders***: Quem está envolvido em uma iteração ou fase para ajudar a responder às perguntas;
6. **Análise ou Avaliação**: Quais as respostas que foram obtidas – com a ajuda dos *Stakeholders* – a partir das perguntas.

Neste trabalho houve a necessidade de adaptar esta tabela para que o processo de reengenharia do Cognitor fosse encaixado. A primeira das alterações consistiu em trocar

[2] `http://www.extjs.com/`
[3] `http://lia.dc.ufscar.br/cognitorweb`

o título da coluna "Protótipo" para "Protótipos ou Recursos", pois em algumas iterações não foram gerados protótipos, mas sim outros artefatos (diagramas de classe, relatórios, artigos, etc). A segunda modificação foi a inclusão de uma coluna, chamada "Técnicas Adotadas", usada para especificar quais foram as técnicas empregadas (análise de requisitos, engenharia reversa, etc) em uma determinada iteração. A estrutura da tabela utilizada, bem como a primeira iteração do caminho percorrido, podem ser vistos na Tabela 3.1.

Tabela 3.1: Estrutura da tabela baseada no trabalho de Anacleto, Fels e Villena [5]

Iteração	**Estratégia de Projeto**	**Perguntas a Serem Respondidas**	**Técnicas Adotadas**	**Protótipos ou Recursos**	***Stakeholders***	**Análise ou Avaliação**
1	Análise das funcionalidades da aplicação *desktop* (aplicação fonte).	**P1.** É possível, a partir dos artefatos disponíveis (programa executável, código fonte e artigos), levantar as funcionalidades da aplicação fonte?	Análise de requisitos e Engenharia reversa.	Lista de funcionalidades da aplicação fonte.	Orientadora e alunos de mestrado do LIA.	**R1.** Sim, é possível obter a lista de funcionalidades da aplicação fonte a partir dos artefatos disponíveis.
2	...	...	...	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	...	...	...	...	...	...

A tabela completa, com todas as iterações detalhadas, pode ser encontrada no Apêndice A. A partir da análise da tabela foi possível perceber que um modelo de processo de reengenharia de software estava se configurando. Para formalizar este modelo de processo foi então utilizada a PML SPEM.

3.3.3 Fase 3 – Formalização do Modelo de Processo Usando SPEM

Como dito anteriormente, a percepção do UC-RIA se deu ao observar a tabela criada na fase anterior, onde foi percebida a configuração de um modelo de reengenharia. Decidiu-se então formalizar o modelo de processo utilizando a PML SPEM. É importante frisar que toda a construção do UC-RIA foi feita no EPF. O primeiro passo da construção do UC-RIA foi a definição dos papéis do modelo. Foram definidos oito papéis:

- **Analista**: É responsável em executar todas as tarefas relacionadas à análise do sistema, como o levantamento de requisitos, criação de diagramas, etc;
- **Arquiteto**: É responsável em decidir quais são as características da arquitetura do sistema, como a escolha do mecanismo de persistência, linguagem de programação a ser utilizada, bibliotecas de componentes gráficos que serão utilizadas, etc;

- **Avaliador de Testes**: É responsável em avaliar os testes realizados com os usuários durante a etapa de prototipação;
- **Desenvolvedor de Infraestrutura**: É responsável em converter as especificações criadas pelo Analista em código, ou seja, programar o sistema, sendo que a programação realizada por este desenvolvedor está relacionada a infraestrutura do sistema, como persistência, gerenciamento de usuários, etc;
- **Projetista de Interface Gráfica**: É responsável em projetar as interfaces gráficas do sistema, criando protótipos, conversando com usuários, etc;
- **Desenvolvedor de Interface Gráfica**: É responsável em desenvolver as interfaces gráficas especificadas pelo Projetista de Interface Gráfica;
- **Testador**: É responsável em executar os testes necessários para a aprovação de determinada funcionalidade;
- **Usuário**: Representa os usuários potenciais da aplicação.

Os oito papéis definidos executam diversas tarefas dentro do modelo de processo, sendo que basicamente cada uma destas tarefas recebe um ou vários artefatos (*Work Products*) como entrada e geram um ou vários artefatos como saída. Um exemplo de uma tarefa definida no UC-RIA é a tarefa "Identificar Funcionalidades da Aplicação Fonte". Na Tabela 3.2 são apresentados os detalhes desta tarefa.

É importante frisar que, durante o desenvolvimento deste trabalho, o aluno atuou em todos os papéis descritos anteriormente e que diversos alunos do laboratório, em especial os alunos de mestrado, contribuíram no desenvolvimento, atuando como Usuários. Essa dinâmica ocorreu, pois este trabalho se trata de uma pesquisa acadêmica, entretanto nada impede – e é recomendável – que diferentes pessoas (que tenham preferencialmente conhecimento especializado) atuem em cada papel.

Tabela 3.2: Detalhe da tarefa "Identificar Funcionalidades da Aplicação Fonte"

<table>
<tr><td colspan="3">Tarefa:
Identificar Funcionalidades da Aplicação Fonte</td></tr>
<tr><td colspan="3">Propósito:
Obter uma lista de funcionalidades disponibilizadas pela aplicação fonte.</td></tr>
<tr><td colspan="3">Descrição:
Esta tarefa consiste em analisar toda a base de conhecimento disponível relacionada com a aplicação fonte de modo a criar uma lista de funcionalidades, detalhando assim cada funcionalidade disponibilizada pela aplicação fonte.</td></tr>
<tr><td colspan="3">Passos:
1 – Analisar aplicação fonte;
2 – Analisar código fonte da aplicação fonte;
3 – Revisar literatura da aplicação fonte;
4 – Analisar a documentação disponível;
5 – Gerar lista de funcionalidades.</td></tr>
<tr><td colspan="3">Papéis:
Analista.</td></tr>
<tr><td colspan="3">Artefatos (Work Products)</td></tr>
<tr><td colspan="2">Entrada(s)</td><td>Saída(s)</td></tr>
<tr><td>Obrigatória(s)</td><td>Opcional(is)</td><td rowspan="4">Lista de Funcionalidades da Aplicação Fonte</td></tr>
<tr><td rowspan="3">Aplicação Fonte - Binário</td><td>Aplicação Fonte - Código Fonte</td></tr>
<tr><td>Artigos da Aplicação Fonte</td></tr>
<tr><td>Documentação da Aplicação Fonte</td></tr>
<tr><td colspan="3">Categorias:
Análise de Requisitos;
Engenharia Reversa.</td></tr>
</table>

No UC-RIA foram definidas diversas tarefas que tratam desde o levantamento de requisitos, como a tarefa apresentada na Tabela 3.2, até a implantação da versão final da aplicação que está sendo gerada. O fluxo do processo no SPEM é organizado utilizando diagramas de atividades da UML. O diagrama de atividades que contém o fluxo de nível mais alto do UC-RIA pode ser visto na Figura 3.1.

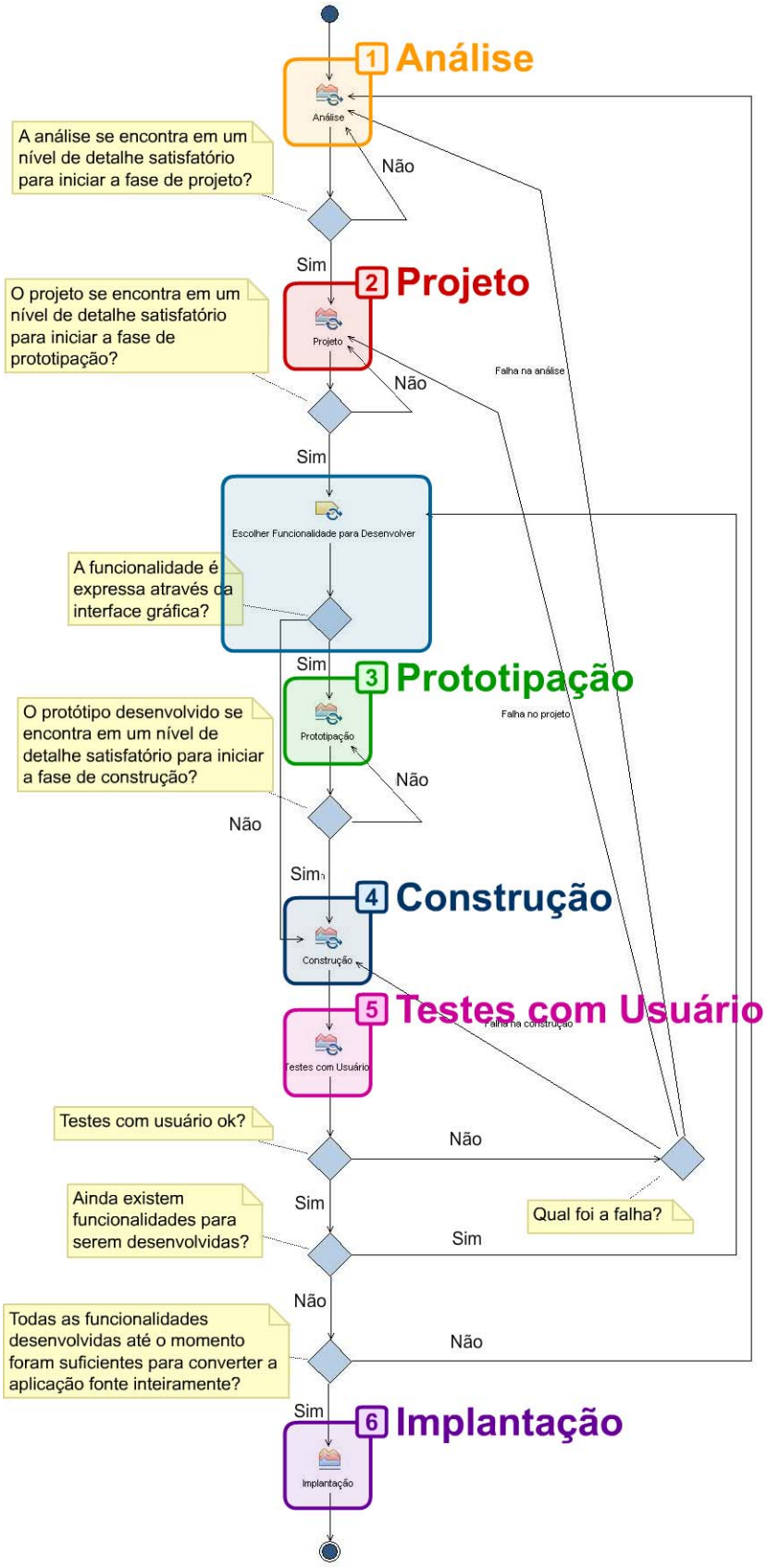


Figura 3.1: O UC-RIA

Como pode ser observado na Figura 3.1, o UC-RIA está dividido em seis fases:

1. **Análise**: Nesta fase (área 1 em laranja) são levantadas as funcionalidades existentes na aplicação fonte, as novas funcionalidades – se existirem – que devem ser implementadas na aplicação alvo, as bibliotecas de componentes gráficos para construção de RIAs são escolhidas e um diagrama de classes de nível conceitual é gerado;

2. **Projeto**: Na fase de Projeto (área 2 em vermelho) o diagrama de classes gerado na fase de Análise é refinado e é gerado um plano de desenvolvimento;

3. **Prototipação**: Com base na lista de funcionalidades gerada, uma é escolhida para ser implementada. Caso a funcionalidade envolva a construção de interfaces gráficas (área destacada em azul claro), a fase de Prototipação (área 3 em verde) é utilizada para gerar protótipos da interface em questão e validar esta interface junto aos potenciais usuários da aplicação alvo, até que este protótipo esteja em um nível de detalhe totalmente aceito pelos usuários;

4. **Construção**: Nesta fase (área 4 em azul escuro) a funcionalidade escolhida é implementada;

5. **Testes com Usuário**: Na fase de Testes com Usuários (área 5 em rosa) a funcionalidade implementada é testada pelos usuários. Se houverem falhas nesta fase, é decidido, com base no tipo da falha (análise, projeto ou construção), para qual fase o processo deve retornar. Se tudo estiver correto é então escolhida uma nova funcionalidade para ser implementada. Caso não haja mais funcionalidades para serem implementadas, é verificado se a aplicação alvo contém todas as funcionalidades da aplicação fonte, caso não tenha, retorna-se a fase de Análise;

6. **Implantação**: Quando todo o processo termina (todas as funcionalidades estão implementadas), a aplicação alvo é implantada (área 6 em roxo).

A fase de Prototipação é o principal diferencial do UC-RIA, em comparação com os modelos de processo de reengenharia apresentados, pois neste modelo de processo existe a preocupação de fazer com que os usuários da aplicação avaliem cada protótipo criado, possibilitando assim que as interfaces criadas estejam de acordo com o que os usuários desejam e que sejam viáveis de desenvolver. Na Figura 3.2, a fase de Prototipação é detalhada.

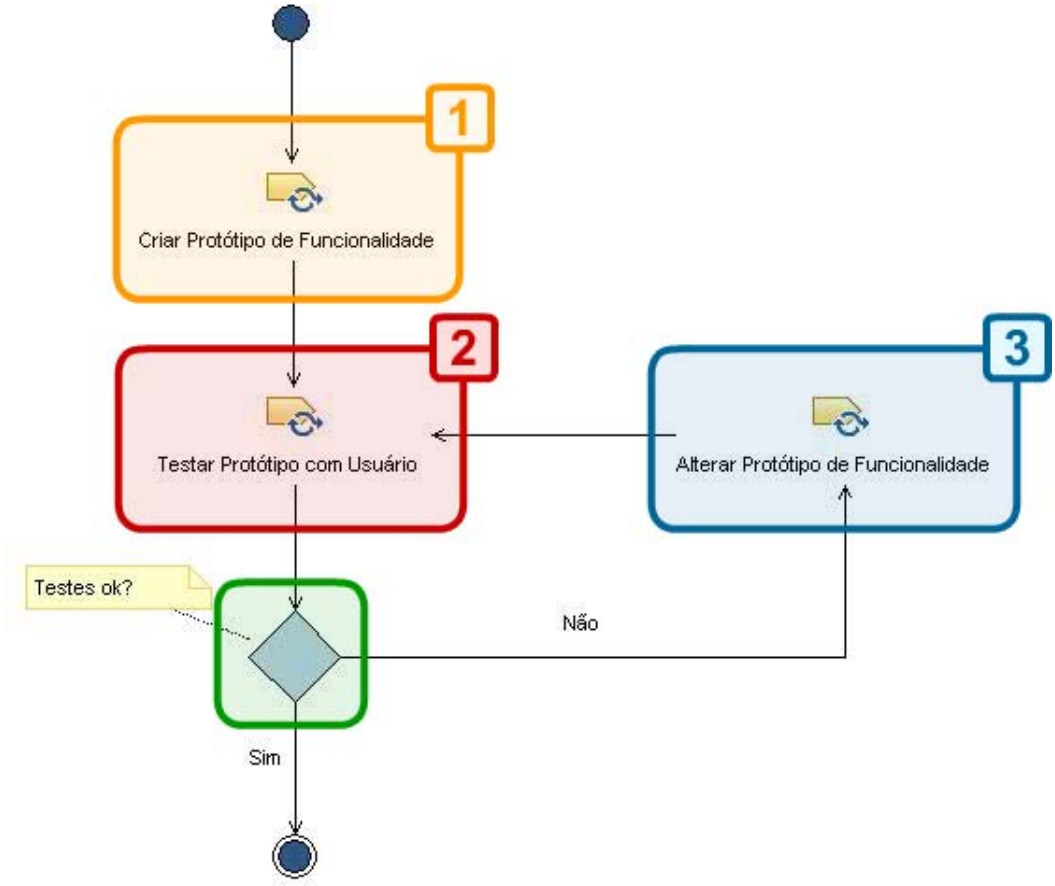


Figura 3.2: Fase "Prototipação" do UC-RIA

A fase de Prototipação é composta por três tarefas, destacadas na Figura 3.2. Na tarefa "Criar Protótipo de Funcionalidade" (área 1 em laranja) o projetista de interface gráfica desenvolve o protótipo da funcionalidade escolhida. Na segunda tarefa, "Testar Protótipo com Usuário" (área 2 em vermelho), o projetista de interface gráfica e o avaliador de testes testam o protótipo gerado em conjunto com o(s) usuário(s). Se o protótipo for aprovado, passa-se para fase de construção (área 4 na Figura 3.1). Caso não tenha sido aprovado, o projetista de interface gráfica, fazendo uso de uma lista de alterações propostas pelo(s) usuário(s), inicia a tarefa "Alterar Protótipo de Funcionalidade" (área 3 em azul), alterando assim o protótipo. O ciclo de testes e alterações (áreas 2 e 3) é executado até que a funcionalidade esteja de acordo com o que o usuário espera. A área destacada em verde representa o resultado do teste com o usuário. As três tarefas da fase de Prototipação estão descritas respectivamente nas Tabelas 3.3, 3.4 e 3.5.

Tabela 3.3: Detalhe da tarefa “Criar Protótipo de Funcionalidade”

<table>
<tr><td colspan="3">Tarefa:
Criar Protótipo de Funcionalidade</td></tr>
<tr><td colspan="3">Propósito:
Criar o protótipo de uma funcionalidade.</td></tr>
<tr><td colspan="3">Descrição:
Esta tarefa tem como objetivo criar o protótipo de uma funcionalidade da aplicação alvo para que este protótipo possa ser submetido a avaliação de um usuário da ferramenta.</td></tr>
<tr><td colspan="3">Passos:
1 – Desenvolver protótipo.</td></tr>
<tr><td colspan="3">Papéis:
Projetista de Interface Gráfica.</td></tr>
<tr><td colspan="3">Artefatos (Work Products)</td></tr>
<tr><td colspan="2">Entrada(s)</td><td>Saída(s)</td></tr>
<tr><td>Obrigatória(s)</td><td>Opcional(is)</td><td rowspan="5">Protótipo de Funcionalidade</td></tr>
<tr><td>Diagrama de Classes de Projeto</td><td rowspan="4"></td></tr>
<tr><td>Lista de Funcionalidades Gerais</td></tr>
<tr><td>Lista de Funcionalidades Adaptadas</td></tr>
<tr><td>Requisitos da Aplicação Alvo</td></tr>
<tr><td colspan="3">Categorias:
Prototipação;
Engenharia Avante.</td></tr>
</table>

Tabela 3.4: Detalhe da tarefa “Testar Protótipo com Usuário”

<table>
<tr><td colspan="3">Tarefa:
Testar Protótipo com Usuário</td></tr>
<tr><td colspan="3">Propósito:
Testar com o(s) usuário(s) o protótipo criado.</td></tr>
<tr><td colspan="3">Descrição:
Esta tarefa tem como objetivo testar com o(s) usuários(s) o protótipo criado.</td></tr>
<tr><td colspan="3">Passos:
1 – Avaliar o uso do protótipo;
2 – Criar lista de alterações do protótipo.</td></tr>
<tr><td colspan="3">Papéis:
Avaliador de Testes;
Usuário;
Projetista de Interface Gráfica.</td></tr>
<tr><td colspan="3">Artefatos (Work Products)</td></tr>
<tr><td colspan="2">Entrada(s)</td><td>Saída(s)</td></tr>
<tr><td>Obrigatória(s)</td><td>Opcional(is)</td><td rowspan="2">Lista de Alterações do Protótipo</td></tr>
<tr><td>Protótipo de Funcionalidade</td><td></td></tr>
<tr><td colspan="3">Categorias:
Prototipação.</td></tr>
</table>

Tabela 3.5: Detalhe da tarefa "Alterar Protótipo de Funcionalidade"

<table>
<tr><td colspan="3">Tarefa:
Alterar Protótipo de Funcionalidade</td></tr>
<tr><td colspan="3">Propósito:
Alterar o protótipo de uma funcionalidade.</td></tr>
<tr><td colspan="3">Descrição:
Esta tarefa tem como objetivo alterar o protótipo de uma funcionalidade.</td></tr>
<tr><td colspan="3">Passos:
1 – Analisar lista de alterações;
2 – Fazer alterações.</td></tr>
<tr><td colspan="3">Papéis:
Projetista de Interface Gráfica.</td></tr>
<tr><td colspan="3">Artefatos (Work Products)</td></tr>
<tr><td colspan="2">Entrada(s)</td><td>Saída(s)</td></tr>
<tr><td>Obrigatória(s)</td><td>Opcional(is)</td><td rowspan="2">Protótipo de Funcionalidade</td></tr>
<tr><td>Lista de Alterações do Protótipo</td><td></td></tr>
<tr><td colspan="3">Categorias:
Prototipação;
Engenharia Avante.</td></tr>
</table>

Pode-se notar que o UC-RIA é um modelo de processo de reengenharia de software que:

1. Preocupa-se com o processo de engenharia reversa da aplicação fonte;
2. Utiliza todo o fluxo normal de desenvolvimento (análise, projeto, implementação, testes e implantação) para a construção da aplicação alvo;
3. Tem o diferencial de formalizar a prototipação de interfaces gráficas das funcionalidades que são expressas por meio delas.

A especificação completa do UC-RIA pode ser encontrada *online* no endereço `http://lia.dc.ufscar.br/uc-ria` ou no Apêndice B. A seguir, será apresentada uma comparação entre o UC-RIA e os dois modelos de processo de reengenharia de software que foram apresentados no Capítulo anterior.

3.4 Comparação entre os Modelos de Reengenharia de Software e o UC-RIA

O UC-RIA é um modelo de processo de reengenharia de software que propõe a inserção de uma fase de Prototipação, advinda do UCD, dentro do ciclo normal de reengenharia.

Ele faz uso, de forma implícita, das ideias do Modelo de Byrne e do Modelo em Espiral Dupla.

Do Modelo de Byrne, o UC-RIA herda o conceito dos princípios da Abstração, Alteração e Refinamento, que são as fases básicas para a maioria dos modelos de processo de reengenharia de software. Analisando o UC-RIA, percebe-se que ele se parece com o Modelo em Espiral Dupla, pois existe a necessidade de se obter as funcionalidades da aplicação alvo, que por sua vez serão migradas aos poucos para a aplicação fonte. No entanto, o UC-RIA não força nem especifica nenhuma forma de comunicação entre a aplicação fonte e a aplicação alvo, pois a ideia do modelo é ser um pouco mais genérico neste quesito, permitindo que aplicações que não precisem de tal recurso de comunicação possam passar pelo processo de reengenharia. É importante frisar que isso não implica em não poder existir esta camada.

Assim como no Modelo em Espiral Dupla, o UC-RIA também não limita a utilização de tecnologias diferentes para a conversão da aplicação fonte para a aplicação alvo, pelo contrário, pois no escopo em que ele se aplica – conversão de aplicações *desktop* em RIAs – existe a necessidade de se utilizar tecnologias diferentes, principalmente no que diz respeito aos componentes para a construção de interfaces gráficas.

O UC-RIA também formaliza, na fase de Análise, uma série de tarefas que têm como objetivo guiar a escolha das bibliotecas de componentes gráficos que serão usadas na construção da aplicação alvo.

Retornando à discussão relativa à fase de Prototipação, pode-se notar que a principal contribuição do UC-RIA para o conjunto de modelos de processo de reengenharia de software existentes, é a inserção do usuário não somente na fase de testes com usuários, mas também na tomada de decisão relacionada a todas as funcionalidades que serão expressas através das interfaces gráficas na aplicação alvo. Considerando o que foi discutido e apresentado neste Capítulo, na próxima Seção são discutidas as Considerações Finais.

3.5 Considerações Finais

Neste Capítulo, foi apresentado o processo de criação e formalização de um modelo de processo de reengenharia de software, denominado UC-RIA. O diferencial do UC-RIA, em relação aos outros modelos de processo de reengenharia, é o de colocar os possíveis usuários da ferramenta como entidades que fazem parte da tomada de decisões no projeto das interfaces gráficas da aplicação que está sendo criada. No próximo Capítulo, será

apresentada a nova versão da ferramenta Cognitor, que é o produto da reengenharia da versão *desktop* da mesma e que foi o ponto de partida para a formalização do UC-RIA.

4 Cognitor

4.1 Considerações Iniciais

Neste Capítulo será apresentada a ferramenta Cognitor, um *framework* que tem como objetivo apoiar a utilização da Cog-Learn, uma linguagem de padrões para e-Learning, que ajuda os educadores no projeto e na edição de material educacional de qualidade para EAD (Educação à Distância) [23]. Este Capítulo está dividido da seguinte maneira: na Seção 4.2 é apresentado o Cognitor; a utilização do Cognitor como um *framework* para a linguagem de padrões Cog-Learn é discutida na Seção 4.3; na Seção 4.4 é descrito o uso do conhecimento cultural disponibilizado pelo Projeto OMCS-Br (*Open Mind Common Sense* no Brasil) no Cognitor; na Seção 4.5, discute-se a utilização do conhecimento cultural no apoio ao preenchimento de metadados dos OAs gerados pela ferramenta; por fim, na Seção 4.6, são apresentadas as Considerações Finais.

4.2 Visão Geral da Ferramenta

O Cognitor é uma ferramenta que foi desenvolvida com o objetivo de apoiar os educadores na tarefa de projetar e editar material educacional de qualidade para EAD na forma de hiperdocumentos. Os materiais gerados no Cognitor podem ser reutilizados em vários contextos de aprendizagem, pois estes materiais são criados com base no conceito de OA, ou seja, são artefatos que podem ser usados, reusados ou referenciados durante o aprendizado apoiado por tecnologia [24]. A interface principal da versão Web do Cognitor, que foi desenvolvida como parte deste trabalho, pode ser visualizada na Figura 4.1. A seguir cada uma das áreas destacadas será detalhada.

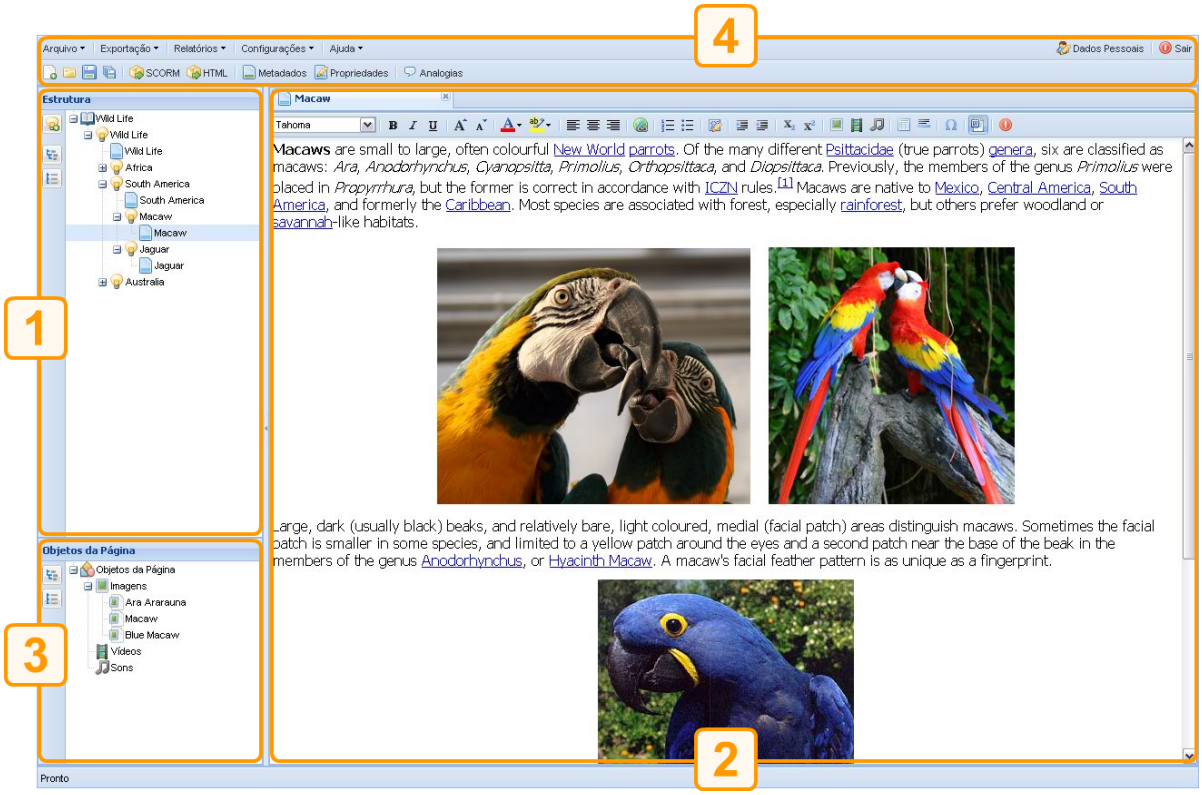


Figura 4.1: Interface Principal da Versão Web do Cognitor

1. **Área de Planejamento e Organização de Material Educacional:** Esta área é utilizada pelo professor para planejar e organizar seu material;

2. **Área de Edição de Página:** Permite ao professor editar seu material didático. A edição e visualização do material são realizadas em tempo real por meio de um editor HTML e de um agregador de mídias, sendo que ambos utilizam o paradigma WYSIWYG (*What You See Is What You Get*);

3. **Área de Controle de Objetos:** O propósito desta área é exibir os objetos (imagens, sons e vídeos) que compõem a página no momento de sua edição;

4. **Menu e Barra de Ferramentas:** O menu e a barra de ferramentas permitem que o usuário da ferramenta execute tarefas como criar um novo material ou salvar um material que está sendo editado, acesso ao assistente de inserção de metadados, criação de pacotes do material, entre outras funcionalidades.

Utilizando estas áreas, o professor pode então criar o seu material didático, indo desde a concepção e estruturação até o empacotamento deste material nos formatos HTML ou SCORM (*Shareable Content Object Reference Model*).

4.2.1 Cognitor para Geração de Objetos de Aprendizagem

De acordo com a norma IEEE LOM (*Learning Object Metadata*) [24], um OA é "qualquer entidade, digital ou não, a qual pode ser usada, reusada ou referenciada durante o aprendizado apoiado por tecnologia". Na mesma norma são citadas algumas situações onde os OAs podem ser usados:

- Sistemas de ensino baseados em computador;
- Ambientes de aprendizagem interativos;
- Sistemas educacionais inteligentes apoiados por computador;
- Sistemas de ensino à distância.

Complementando a definição da IEEE, Farrell et al. [25] definem que os OAs são formados por recursos de mídia agregados, os quais podem ter sido desenvolvidos de forma independente uns dos outros e que são disponibilizados para reuso através da Web ou de um repositório de conteúdo.

Para que um material didático eletrônico se torne um OA, ele precisa ser empacotado em algum formato específico. Para criar OAs a partir de materiais gerados no Cognitor utiliza-se a norma SCORM, que provê uma série de especificações que definem como o conteúdo de um material deve ser organizado e empacotado de forma a ser executado corretamente em LMSs (*Learning Management System*) que implementam a especificação do ambiente de execução SCORM [26]. Uma característica importante de um OA é que ele pode ser reutilizado, mas para que esta reutilização seja possível é necessário que o conteúdo do OA seja anotado através de metadados, ou seja, o conteúdo e seus componentes, uma imagem por exemplo, precisam ser classificados através de uma série de atributos, fazendo com que a busca e a obtenção do pacote seja facilitada. Existem diversos padrões para a organização dos metadados, como por exemplo, o Dublin Core[1], o CanCore[2], o IMS[3] e o IEEE LOM[4], sendo que este último é utilizado internamente pelo SCORM e o preenchimento de tais campos é suportado pelo Cognitor. Esta funcionalidade da ferramenta será explorada em maior detalhe na Seção 4.5.

[1] `http://dublincore.org/`
[2] `http://cancore.athabascau.ca/en/`
[3] `http://www.imsglobal.org/metadata/`
[4] `http://ltsc.ieee.org/wg12/`

4.3 Cognitor como *Framework* para a Linguagem de Padrões Cog-Learn

O conceito original de padrões foi concebido por Alexander et al. [27] para representar soluções de sucesso para problemas recorrentes de um certo contexto. Baseado na definição de Alexander, especialistas de IHC, tais como Welie e Veer [28], Fincher [29] e Borchers [30], criaram vários padrões que são utilizados não apenas para compartilhar soluções de sucesso entre os profissionais da área, mas também para fornecer um vocabulário comum entre os especialistas e o usuário final do sistema.

Segundo Talarico Neto et al. [23], um padrão pode ser parte de uma LP (Linguagem de Padrões), sendo que cada padrão que a LP contém se relaciona com outros padrões que detalham soluções para outros problemas de projeto no mesmo domínio. Na Figura 4.2 a LP Cog-Learn é apresentada para exemplificar a estrutura de uma LP, mostrando o relacionamento existente entre vários de seus padrões.

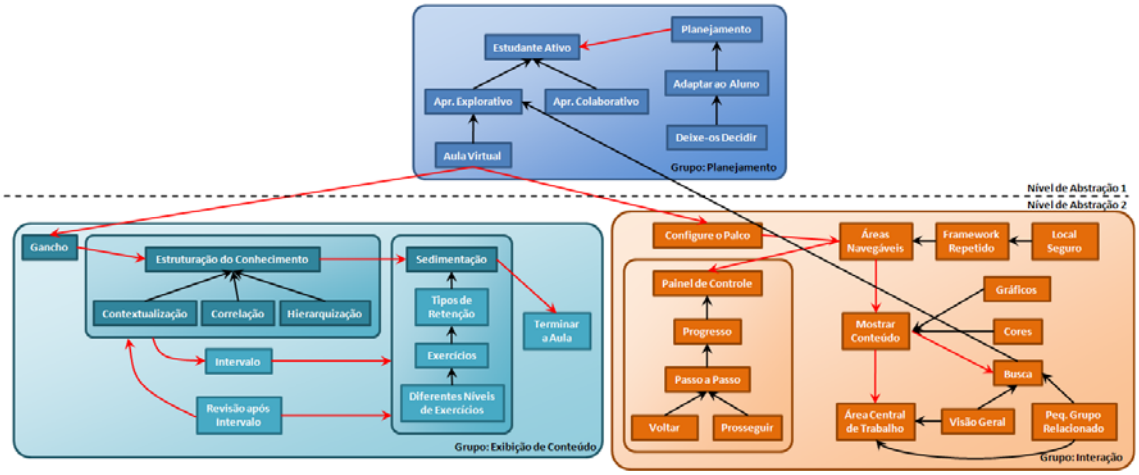


Figura 4.2: LP Cog-Learn [4]

A criação de uma LP que orienta o desenvolvimento de projetos de material educacional para EAD, segundo Talarico Neto et al. [23], deve conter padrões que orientem os professores na maneira de elaborar a estrutura e o conteúdo de uma aula. Desta maneira, os padrões podem: apoiar na concepção de um projeto ajudando a criar a sequência de ações do aluno, oferecendo auxílio durante a realização de um curso com estratégias de acesso e tratando de questões relativas ao projeto da interação, como navegação e *layout*.

Com base nestes fatos, foi proposta a criação da LP para EAD denominada Cog-Learn, a qual foi identificada e formalizada por especialistas em padrões de projeto durante a condução de três estudos de caso, apresentados no trabalho de Neris et al. [31]. Os estudos

de caso permitiram verificar que a utilização de um conjunto selecionado de estratégias cognitivas, em conjunto com padrões de IHC e pedagógicos, fazem com que a usabilidade dos materiais educacionais elaborados na forma de hiperdocumentos seja aumentada.

O Cognitor oferece auxílio na criação de material educacional por meio da representação computacional da Cog-Learn, oferecendo funcionalidades que são disponibilizadas ao professor através da ferramenta, sendo que estas funcionalidades expressam a utilização dos padrões contidos na LP, dando ao professor a capacidade de projetar e elaborar de maneira automatizada seu material educacional [23].

Na próxima Seção serão apresentados os padrões da Cog-Learn que são instanciados no Cognitor, bem como a utilização do conhecimento cultural disponibilizado pelo Projeto OMCS-Br para apoiar a instanciação destes padrões.

4.4 Uso de Conhecimento Cultural para Instanciar os Padrões da Cog-Learn

Antes de apresentar a instanciação dos padrões da Cog-Learn no Cognitor, é necessário entender o que é conhecimento cultural e como ele pode ser obtido, armazenado e disponibilizado para aplicações computacionais, pois a instância de alguns dos padrões implementados no Cognitor utilizam este conhecimento cultural. A implementação destes padrões na versão *desktop*, apoiados pelo uso do conhecimento cultural, foi proposta por Carlos [32] e publicada em Anacleto et al. [33].

4.4.1 Projeto OMCS-Br

O Projeto OMCS no Brasil (OMCS-Br) teve início a partir de uma parceria do LIA (Laboratório de Interação Avançada) com o Media Lab do MIT (*Massachusetts Institute of Technology*), com objetivo de registrar os fatos de senso comum da população brasileira.

Assim como o OMCS (*Open Mind Common Sense*), o Projeto OMCS-Br leva em consideração que qualquer pessoa possui o senso comum que se deseja fornecer às máquinas [34], tornando a construção da base de conhecimento um trabalho colaborativo, envolvendo as pessoas ao redor do mundo neste desafio, usando os recursos providos pela Internet e adoção da representação do conhecimento em língua natural. Deste modo, aproveitando os recursos da Internet e dos avanços das pesquisas na área de processamento de língua natural, foi lançado o projeto brasileiro com a disponibilização de um

site de coleta de senso comum [35].

4.4.1.1 O Site

O site do Projeto OMCS-Br pode ser acessado por qualquer pessoa através do endereço `http://www.sensocomum.ufscar.br`. Ao entrar no site, qualquer pessoa pode se cadastrar e ter acesso a diversas atividades e temas disponíveis. Atualmente o site do Projeto OMCS-Br conta com 8 temas distintos e 20 atividades que abordam os diferentes tipos de conhecimento que compõem o senso comum das pessoas (Figura 4.3).

As atividades foram propostas para coletar fatos relacionados a assuntos gerais, como por exemplo, o uso dos objetos que se tem contato no dia-a-dia, o local onde tais objetos podem ser encontrados, o material de que são feitos, etc.

Os temas foram propostos para coletar conhecimento relacionado a domínios específicos, como Sexualidade e Saúde. Esta abordagem, que tem como objetivo agilizar a coleta dos dados em domínios de interesse para as pesquisas desenvolvidas no LIA, foi uma inovação na versão brasileira do Projeto OMCS.

Figura 4.3: Interface do site do Projeto OMCS-Br

A coleta no site é feita através de *templates*, que são sentenças com estruturas gramaticais simples, que possuem lacunas que devem ser preenchidas pelo usuário de forma a compor uma sentença que, para ele, seja verdadeira, considerando as experiências do seu dia-a-dia e conhecimento, isto é, que represente para ele um fato de senso comum.

Os *templates* utilizados no site, de acordo com a Figura 4.4, possuem uma parte estática (destacada em azul) e uma parte dinâmica (destacada em verde). O conteúdo da parte dinâmica muda cada vez que o template é apresentado para o usuário, considerando as informações que outros já forneceram ao site em interações anteriores. Sendo assim, a base de conhecimento se retroalimenta; em outras palavras, usa o conhecimento que já possui para coletar novos fatos.

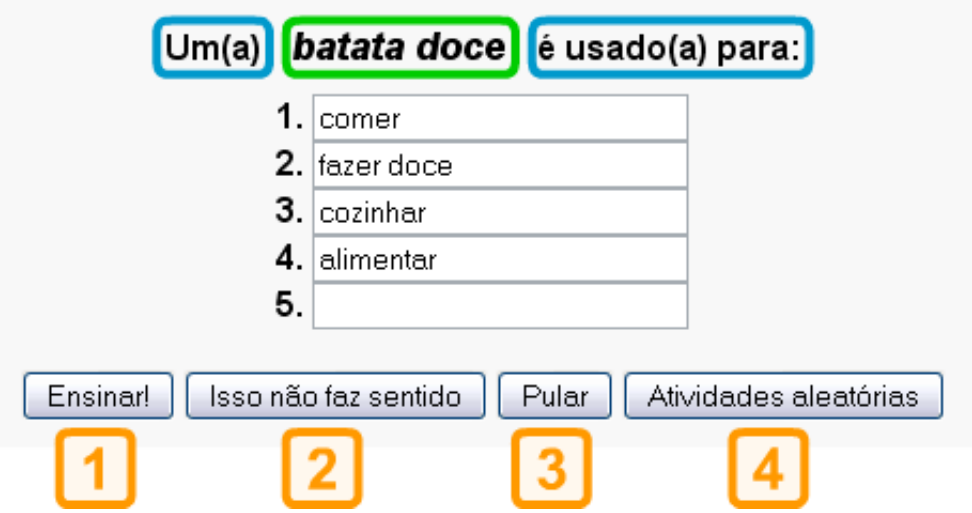


Figura 4.4: Exemplo de um *template* da atividade "Usos" do Projeto OMCS-Br

Em cada *template* o usuário possui quatro opções:

1. **Ensinar**: Enviar as informações que ele digitou para o Projeto OMCS-Br;
2. **Isso não faz sentido**: Informar que o conteúdo da sentença, parte estática e dinâmica, não tem uma relação lógica;
3. **Pular**: Pular para outro *template* sem ter que preencher o atual;
4. **Atividades aleatórias**: Permitir que *templates* de todas as atividades existentes no site possam aparecer.

Todas as informações coletadas através dos *templates* passam por um processamento para gerar uma rede semântica. Esta rede, que tem como característica representar o conhecimento e ser uma ferramenta de suporte para sistemas automatizados de inferência

sobre o conhecimento [36], é a forma que o Projeto OMCS-Br utiliza para armazenar e conectar todas as informações coletadas.

Na Figura 4.5 é ilustrada uma pequena parte desta rede semântica. As informações coletadas são os conceitos, representados pelos nós, interligados pelos arcos que representam as relações semânticas entre os mesmos. Esta rede é denominada ConceptNet – Rede de Conceitos.

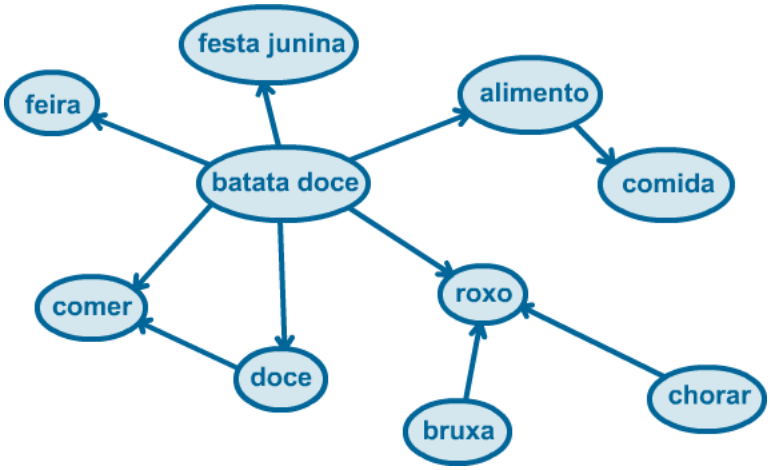


Figura 4.5: Exemplo de parte da ConceptNet

Como pode ser observado na Figura 4.5, os conceitos: “batata doce”, “festa junina”, “alimento”, “feira”, “comida”, “comer”, “doce”, “roxo”, “bruxa” e “chorar” estão conectados. Estes conceitos foram inseridos em algum momento por algum usuário que utilizou o site do Projeto OMCS-Br.

Com o intuito de mapear o conhecimento humano, os conceitos são conectados através de arcos, que representam as relações do modelo de conhecimento proposto por Marvin Minsky [37], de tal forma que a estrutura do conhecimento obtida na rede de conceitos esteja próxima à estrutura cognitiva humana.

Na Tabela 4.1 são apresentadas as 20 relações de Minsky utilizadas atualmente no projeto brasileiro do OMCS.

Tabela 4.1: Exemplo de parte da ConceptNet com as relações de Minsky [6]

Classificação	Relação	Exemplo
K-Lines	ConceptuallyRelatedTo	(ConceptuallyRelatedTo 'bad breath' 'mint' 'f=4;i=0')
	ThematicKLine	(ThematicKLine 'weding dress' 'veil' 'f=9;i=0')
	SuperThematicKLine	(SuperThematicKLine 'western civilisation' 'civilisation' 'f=0;i=12')
Things	IsA	(IsA 'horse' 'animal' 'f=17;i=3')
	PropertyOf	(PropertyOf 'fire' 'dangerous' 'f=17;i=1')
	PartOf	(PartOf 'butterfly' 'wing' 'f=3;i=0')
	MadeOf	(MadeOf 'bacon' 'pig' 'f=3;i=0')
	DefinedAs	(DefinedAs 'meat' 'flesh of animal' 'f=2;i=1')
Agents	CapableOf	(CapableOf 'dentist' 'pull tooth' 'f=4;i=0')
Events	PrerequisiteEventOf	(PrerequisiteEventOf 'read letter' 'open envelope' 'f=2;i=0')
	FirstSubeventOf	(FirstSubEventOf 'start fire' 'light match' 'f=2;i=3')
	SubEventeOf	(SubEventOf 'play sport' 'score goal' 'f=2;i=0')
	LastSubeventOf	(LastSubEventOf 'attend classical concert' 'aplaud' 'f=2;i=1')
Spatial	LocationOf	(LocationOf 'army' 'in war' 'f=3;i=0')
Causal	EffectOf	(EffectOf 'view video' 'entertainment' 'f=2;i=0')
	DesirousEffectOf	(DesirousEffectOf 'sweat' 'take a shower' 'f=3;i=1')
Functional	UsedFor	(UsedFor 'fire place' 'burn' 'f=1;i=2')
	CapableOfReceivingAction	(CapableOfReceivingAction 'drink' 'serve' 'f=0;i=14')
Affective	MotivationOf	(MotivationOf 'play game' 'compete' 'f=3;i=0')
	DesireOf	(DesireOf 'person' 'not be depressed' 'f=2;i=0')

A Figura 4.6 ilustra a rede semântica apresentada na Figura 4.5 com os conceitos conectados através das relações de Minsky.

Figura 4.6: Exemplo de parte da ConceptNet com as relações de Minsky

Para que as aplicações computacionais possam acessar diretamente a ConceptNet foi implementada uma API (*Application Programming Interface*). Esta API possui diversas funções que permitem acessar o conteúdo da ConceptNet. Na Figura 4.7 é apresentado um exemplo do uso da API através de uma aplicação de navegação da ConceptNet. No exemplo é executada uma busca (função Navegar) pelo conceito "batata doce". Ao executar a busca, os resultados são apresentados em uma lista, destacada em laranja.

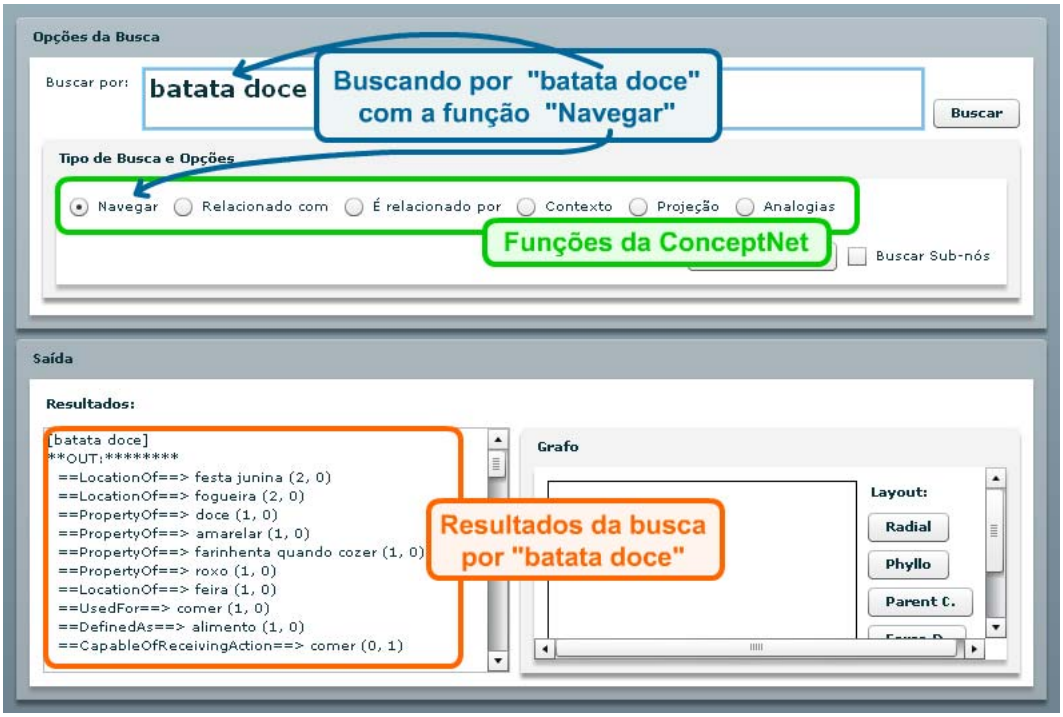


Figura 4.7: Aplicação para uso da API da ConceptNet

Pelo exposto, nota-se que o conhecimento cultural da população brasileira está expresso na base de conhecimento de senso comum do Projeto OMCS-Br, sendo que esta base de conhecimento é estruturada na forma de uma rede de conceitos denominada ConceptNet. O conhecimento cultural expresso na rede de conceitos pode ser acessado pelas aplicações computacionais desenvolvidas no LIA através de uma API, permitindo assim que estas aplicações sejam contextualizadas.

Na próxima Seção serão mostrados os padrões da Cog-Learn instanciados no Cognitor e que têm apoio do conhecimento cultural.

4.4.2 O Padrão Estruturação do Conhecimento

O padrão "Estruturação do Conhecimento" é implementado no Cognitor na "Área de Planejamento e Organização de Material Educacional" (área 1 da Figura 4.1). Como já dito, esta área é utilizada pelo professor para planejar e organizar seu material, sendo que isso pode ser feito de duas maneiras: a primeira é definindo uma nova estrutura de organização do material, por exemplo, qual a sequência das páginas ou então quais páginas pertencem a um determinado tópico. A segunda consiste na utilização do padrão "Estruturação do Conhecimento", que faz parte da LP Cog-Learn. Este padrão considera a teoria dos mapas conceituais de Novak [38], que representa a proposta de Ausubel [39]

no planejamento e organização de um material. O Cognitor possui um assistente para a utilização deste padrão, com o apoio de um módulo que permite a recuperação de conhecimento de senso comum para ajudar na identificação de conceitos [32]. Na versão Web do Cognitor, desenvolvida neste projeto, o assistente é composto pelos dois passos descritos a seguir:

- **Primeiro passo**: O assistente permite ao professor inserir (área 1 da Figura 4.8), remover e organizar de forma hierárquica os conceitos que deseja abordar no material educacional. Para cada conceito inserido no material, é realizada uma consulta na ConceptNet (por meio da API), sugerindo ao professor conceitos relacionados ao conceito inserido (área 2 da Figura 4.8).

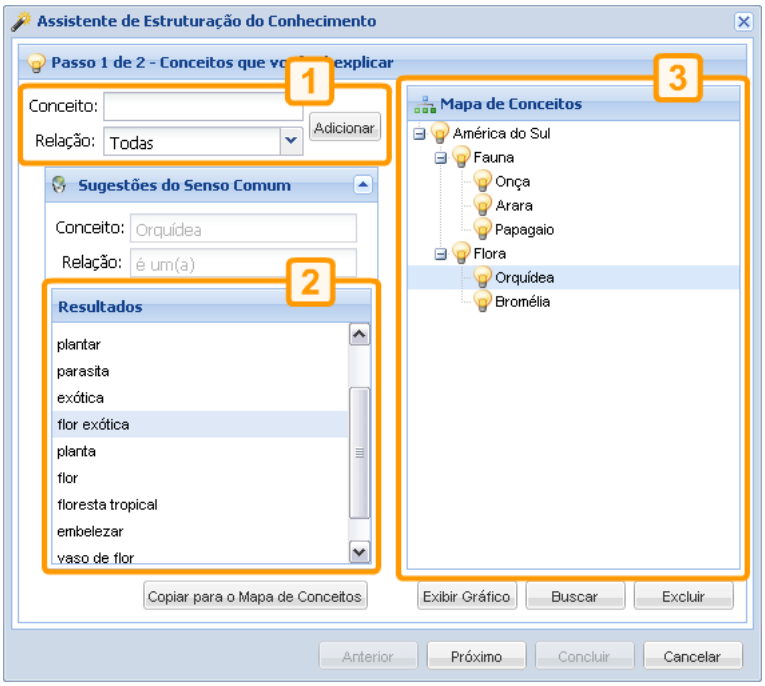


Figura 4.8: Primeiro passo do assistente de estruturação do conhecimento com sugestões de conhecimento cultural

Os conceitos utilizados pelo professor formam um mapa de conceitos que servirá como estrutura do material didático. Esta estrutura pode ser vista na área 3 da Figura 4.8.

- **Segundo passo**: Exibe ao professor as relações existentes entre os conceitos, permitindo renomear cada relação (área 1 da Figura 4.9). Estas relações são obtidas através da organização hierárquica entre os conceitos que foram definidos pelo professor no primeiro passo. Por exemplo, o expressão "América do Sul tem grande variedade de Flora", destacada em roxo na área 2 da Figura 4.9. Esta expressão é

formada pelos conceitos “América do Sul” e “Flora”, sendo ligados pela relação “tem grande variedade de”. Na área 3 (destacada em azul) da Figura 4.9, pode-se ver a edição da relação entre os conceitos “Flora” e “Orquídea” sendo realizada.

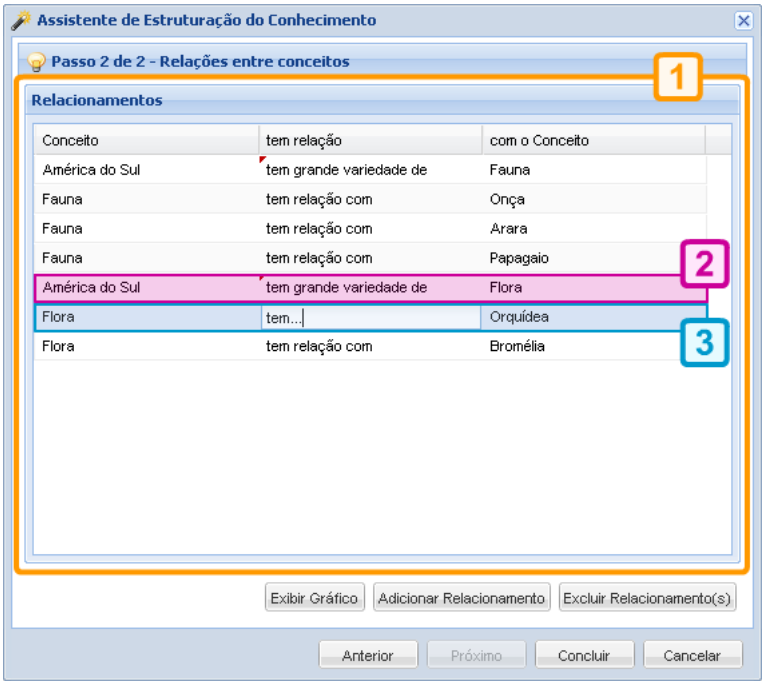


Figura 4.9: Segundo passo do assistente de estruturação do conhecimento

Ao finalizar este assistente, o professor obterá o mapa conceitual. Este mapa será utilizado como a estrutura das páginas e dos conceitos que serão trabalhados no material didático. O resultado obtido pelo assistente é exibido pelo Cognitor em forma de uma árvore, composta por um conjunto de páginas, organizadas em tópicos, sendo que estas páginas são consideradas OAs dentro da ferramenta. Um exemplo de uma árvore criada (estrutura do material) pode ser vista na área 1 da Figura 4.1.

4.4.3 O Padrão Correlação

O padrão “Correlação” da LP Cog-Learn é implementado no Cognitor através de um mecanismo de sugestão de analogias, sendo que este padrão tem como objetivo sugerir o uso de conceitos análogos para explicar um novo conceito que está sendo ensinado e que não faz parte do contexto comum dos aprendizes [33]. Os passos do mecanismo de busca, escolha e inserção de analogias podem ser vistos na Figura 4.10 e na Figura 4.11.

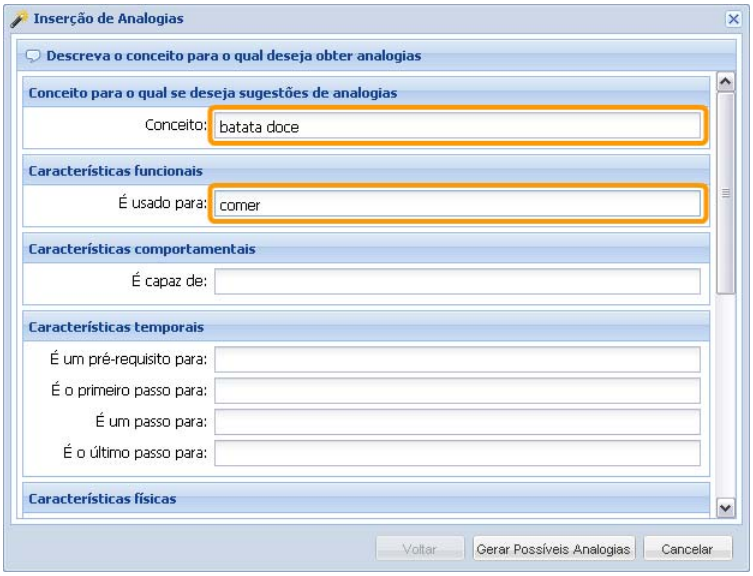


Figura 4.10: Primeiro passo do assistente de inserção de analogias

No primeiro passo do assistente (Figura 4.10) o professor define o conceito desejado para gerar as analogias (no exemplo, "batata doce") bem como as características que julgar importantes para este conceito (no exemplo, é usado para "comer"). Ao terminar de preencher os campos desejados, o professor clica no botão "Gerar Possíveis Analogias". Assim é invocada a função de geração de analogias da API da ConceptNet. Após a geração das analogias, é apresentado ao professor o segundo passo do assistente (Figura 4.11).

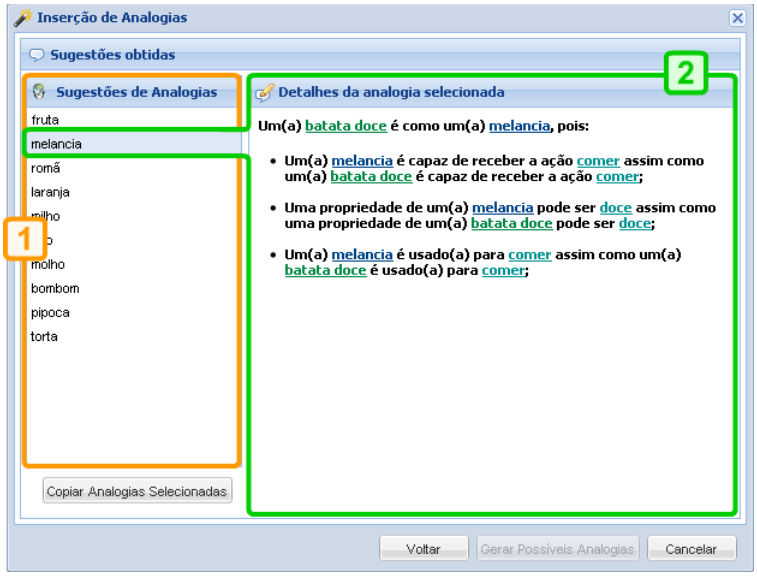


Figura 4.11: Segundo passo do assistente de inserção de analogias

No segundo passo, o professor decide qual ou quais analogias ele deseja inserir no seu material (área 1 da Figura 4.11), clica no botão "Copiar Analogias Selecionadas", copiando-as assim para a página de conteúdo que está sendo editada no momento. No

exemplo da Figura 4.11, destacado em verde na área 2, são mostradas as analogias que "batata doce" tem com "melancia".

Além do suporte para a estruturação do material didático e a inserção de analogias, o conhecimento cultural pode também apoiar o professor a preencher de forma contextualizada os metadados dos OAs gerados no Cognitor. Esta funcionalidade será explorada na próxima Seção.

4.5 Uso de Conhecimento Cultural para Apoiar o Preenchimento de Metadados

Para que um conteúdo digital se torne um OA é necessário anotá-lo com metadados. Para permitir que os materiais gerados dentro do Cognitor sejam anotados, a ferramenta fornece um assistente para este propósito. A versão atual da ferramenta, um dos produtos deste projeto, permite que diversos campos de metadados sejam apoiados durante o processo de preenchimento pelo uso do conhecimento cultural disponibilizado pelo Projeto OMCS-Br através da API da ConceptNet. A proposta para o suporte destes campos, bem como os protótipos da interface gráfica do assistente de preenchimento de metadados, foram validados através de um experimento. Este experimento resultou na publicação do artigo "*Providing Culturally Contextualized Metadata to Promote Sharing and Reuse of Learning Objects*" no evento SIGDOC (*Special Interest Group on the Design of Communication*) do ano de 2009 [40]. Na Figura 4.12 pode ser visto o assistente sendo utilizado para preencher os metadados da página de conteúdo "DST".

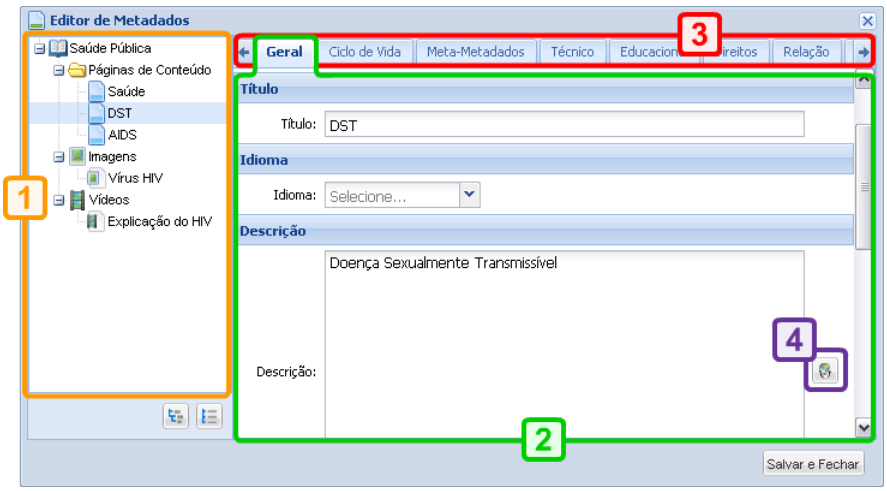


Figura 4.12: Editor de Metadados

Na área 1 da Figura 4.12, é destacada em laranja a lista de OAs que podem ser

anotados com os metadados, enquanto na área 2 em verde são mostrados os campos de metadados da categoria "Geral", sendo que esta faz parte de uma lista de nove categorias, destacadas na área 3 em vermelho. Como pode ser visto na Figura 4.12, dois campos de metadados estão preenchidos: "título" e "descrição". O campo "descrição" possui suporte ao uso do conhecimento cultural para o seu preenchimento, sendo que este recurso pode ser acessado através do botão destacado em roxo na área 4. Ao clicar neste botão, é exibido ao usuário da ferramenta o assistente de suporte do senso comum, permitindo ao professor executar buscas na base de conhecimento cultural (Figura 4.13).

Figura 4.13: Assistente de busca de conhecimento cultural

Na Figura 4.13 é apresentado um exemplo onde foi executada uma busca pelo conceito "doença" – destacado em azul – e os resultados desta busca foram listados na área 1 em laranja. Dos resultados obtidos, foram selecionados quatro conceitos que o professor julgou serem relevantes: "evitar", "curar", "transmitir" e "diagnosticar". Ao selecionar os conceitos desejados, o professor clica no botão "Copiar para o campo", e assim os conceitos selecionados são copiados para o campo que teve o assistente utilizado, neste caso, o campo "descrição". Na Figura 4.14 podem ser vistos os conceitos – destacados em laranja – que foram selecionados e copiados para o campo "descrição".

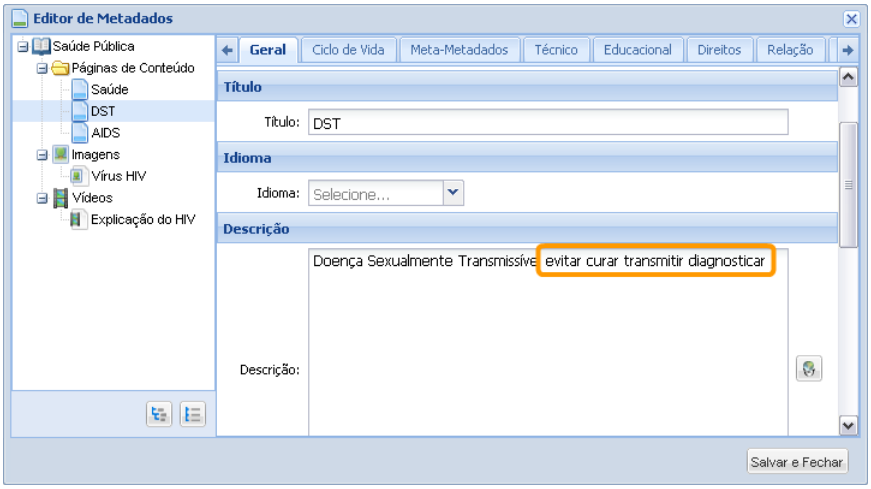


Figura 4.14: Conceitos copiados para o campo de metadados "descrição"

4.6 Considerações Finais

Neste Capítulo a versão Web do Cognitor, desenvolvida como parte deste projeto, foi apresentada. Foram abordadas suas principais funcionalidades, destacando suas interfaces e o funcionamento básico de cada uma delas. Foi visto também que o uso do conhecimento cultural disponibilizado pelo Projeto OMCS-Br pode apoiar o professor durante o processo de criação do material didático. Este apoio pode ocorrer de diversas formas como: na estruturação do material por meio do padrão "Estruturação do Conhecimento"; na inserção de analogias e no preenchimento dos metadados. Esta última funcionalidade é necessária para que o conteúdo gerado na ferramenta possa ser disponibilizado como OAs, facilitando assim o armazenamento e a obtenção de tais artefatos, permitindo que estes sejam reusados em outros contextos.

No próximo Capítulo, serão apresentadas as modificações que foram feitas na nova versão do Cognitor, comparando as interfaces da versão *desktop* com as da versão Web.

5 *Cognitor* Desktop *X Cognitor Web*

5.1 Considerações Iniciais

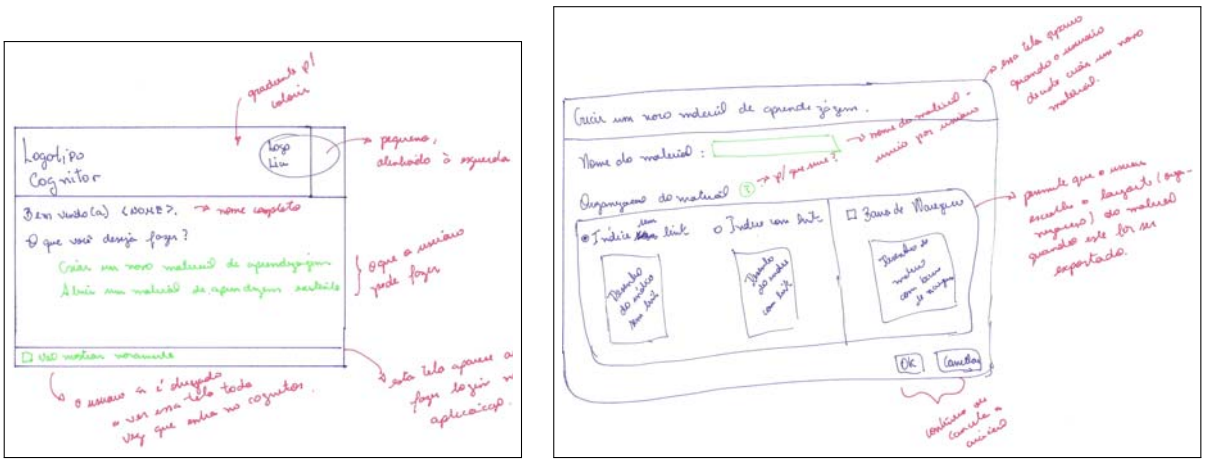
Neste Capítulo são apresentadas e comparadas as interfaces das versões *desktop* e Web do Cognitor, pois durante a prototipação das interfaces da nova versão, diversas alterações foram feitas de acordo com a participação de usuários. Além das comparações entre as duas versões, também são apresentadas as novas funcionalidades do Cognitor Web, por exemplo, o gerenciamento de usuários. Sendo assim, este Capítulo está organizado da seguinte maneira: na Seção 5.2 são apresentadas as comparações entre as duas versões da aplicação; na Seção 5.3 são discutidas as novas funcionalidades da versão Web; por fim, na Seção 5.3, são apresentadas as Considerações Finais.

5.2 Comparação Entre as Versões *Desktop* e Web do Cognitor

Durante o desenvolvimento da versão Web do Cognitor, a interface sofreu várias alterações a fim de aperfeiçoar a interação com a aplicação e a satisfação de uso do usuário. O ponto de partida, na maioria dos casos, foi a criação de um protótipo que representava determinada interface da versão *desktop*. Após a criação do protótipo, ele passava pelo processo de refinamento (testes/alteração), até que o usuário o aprovasse. É importante lembrar que este processo corresponde à fase de Prototipação do UC-RIA. Nas próximas Seções são apresentadas as comparações entre as interfaces das versões *desktop* e Web do Cognitor e, em alguns casos, também são apresentados alguns protótipos em papel que foram criados como parte do processo. Vale observar também que não são tratados os detalhes do funcionamento das funcionalidades, pois já foram apresentados no Capítulo 4.

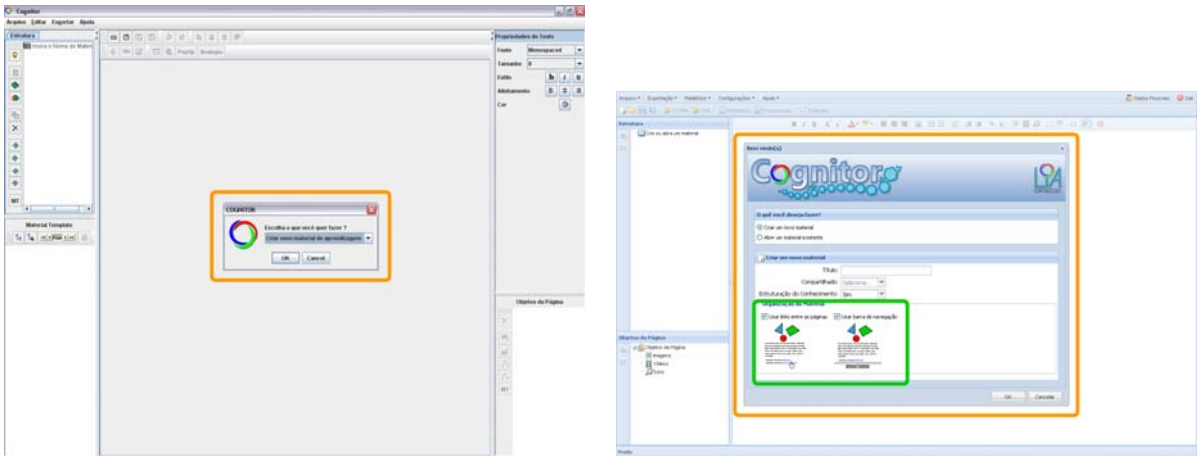
5.2.1 Interface Principal

Na Figura 5.1 são apresentados dois protótipos em papel da tela inicial da versão Web da ferramenta. Estes protótipos em papel foram criados a partir da necessidade de melhorar a tela inicial da versão *desktop* do Cognitor, destacada em laranja na Figura 5.2a, onde o usuário podia escolher se queria criar um novo material ou abrir um material existente. Na versão Web da tela inicial, destacada em laranja na Figura 5.2b, foram adicionadas as funcionalidades previstas no protótipo, onde além do usuário poder escolher se quer um novo material ou abrir um material existente, ele também pode definir algumas outras propriedades do mesmo, como o *layout* que será usado no material exportado.



(a) Tela inicial (b) Criação do novo material

Figura 5.1: Protótipos em papel da tela inicial da versão Web



(a) Versão *desktop* (b) Versão Web

Figura 5.2: Comparação da tela inicial

Da mesma forma que ocorreu com a tela inicial, a interface principal da ferramenta também foi criada primeiramente utilizando um protótipo em papel, apresentado na Fi-

gura 5.3. Este protótipo foi criado para permitir a visualização de uma versão mais simples da interface principal, removendo a opção de alterar o *layout* do material, que agora está presente na tela inicial da versão Web (área destacada em verde na Figura 5.2b) e modificando a localização de algumas áreas da aplicação.

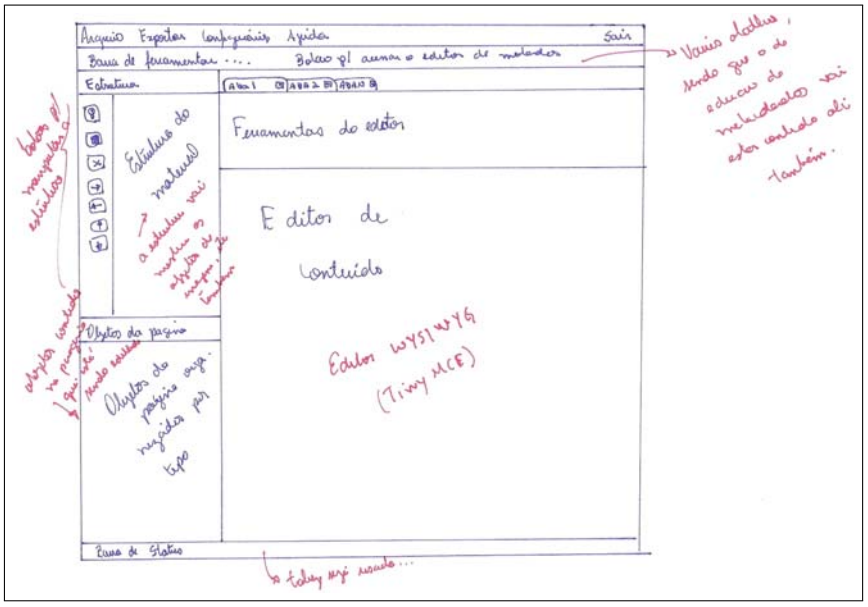


Figura 5.3: Protótipo em papel da interface principal da versão Web

As alterações realizadas na interface principal podem ser vistas comparando-se as Figuras 5.4 e 5.5.

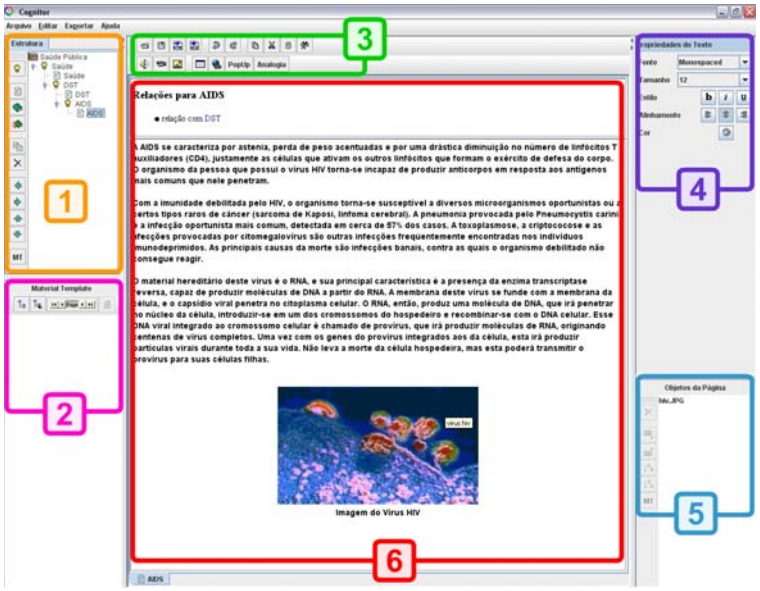


Figura 5.4: Interface principal da versão *desktop*

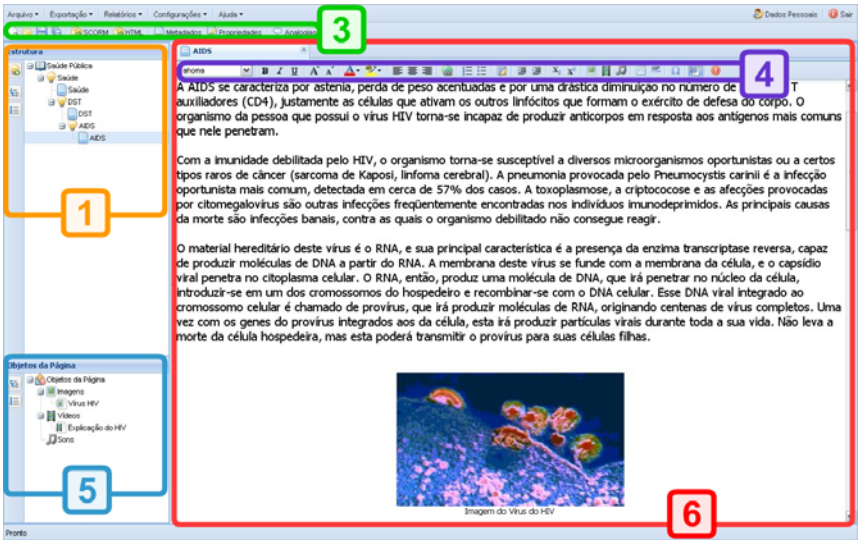
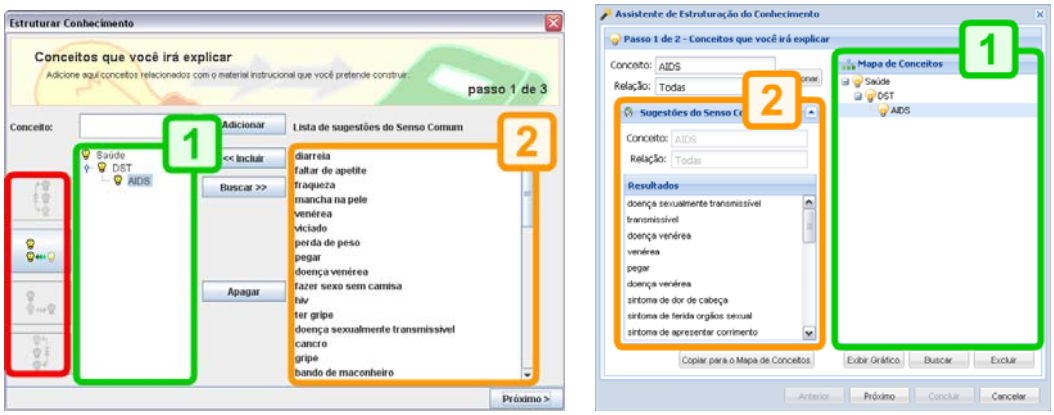


Figura 5.5: Interface principal da versão Web

A área para alteração do *layout*, destacada em rosa na Figura 5.4 foi removida, passando a fazer parte da tela inicial, destacada em verde na Figura 5.2b. A área de objetos da página (área 5 destacada em azul) foi movida para o lado esquerdo da interface. As ferramentas para edição de texto e de mídias (área 4 em roxo) foram colocadas dentro do editor WYSYWYG (área 6 em vermelho).

5.2.2 Estruturação do Conhecimento

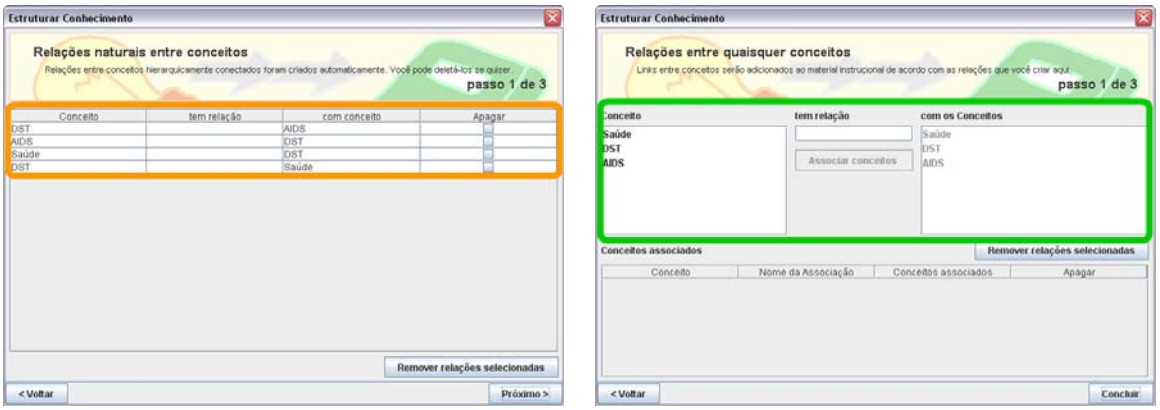
No primeiro passo do assistente de estruturação do conhecimento foram feitas algumas modificações. A primeira delas foi a troca do lugar do mapa de conceitos. Na versão *desktop*, o mapa de conceitos (área 1 em verde da Figura 5.6a) ficava abaixo do campo utilizado para inserir um conceito. De acordo com os usuários, esta característica era confusa, pois ao clicar no botão "Adicionar" eles esperavam que o conceito fosse copiado para o lado direito do assistente e não para a área logo abaixo. Pode-se ver a nova localização do mapa de conceitos na área 1 em verde da Figura 5.6b. Devido a mudança de local, a lista que exibe as sugestões de conceitos relacionados ao conceito inserido (vindas da base de conhecimento cultural disponibilizada pelo Projeto OMCS-Br) foi colocada do lado esquerdo. Esta mudança pode ser vista na área 2, destacada em laranja, da Figura 5.6a e da Figura 5.6b. Outra modificação foi a remoção dos botões que controlam a ordem dos conceitos dentro do mapa de conceitos (área destacada em vermelho na Figura 5.6a). Na versão Web, a organização dos conceitos é realizada utilizando *drag-and-drop* (arrastar e soltar), onde o usuário seleciona e arrasta o conceito escolhido – utilizando o mouse – para onde quiser dentro da estrutura do mapa.



(a) Versão *desktop* (b) Versão Web

Figura 5.6: Comparação do primeiro passo do assistente de estruturação do conhecimento

Na versão *desktop* da ferramenta, o assistente de estruturação do conhecimento é dividido em três passos. O terceiro passo, segundo os usuários, poderia ser eliminado e inserido como algo opcional no segundo passo da versão Web. Sendo assim, o segundo e o terceiro passos – representados respectivamente na Figura 5.7a e na Figura 5.7b – foram convertidos em um único passo na versão Web (Figura 5.8). Pode-se ver que o segundo passo do assistente na versão Web é basicamente igual ao segundo passo da versão *desktop* (áreas em laranja na Figura 5.7a e na Figura 5.8), sendo que a funcionalidade do terceiro passo da versão *desktop* (destacada em verde na Figura 5.7b), que é criar um novo relacionamento que não foi sugerido pela ferramenta, é alcançada por meio de um botão, destacado em verde na Figura 5.8.



(a) Passo 2 – Versão *desktop* (b) Passo 3 – Versão *desktop*

Figura 5.7: Segundo e terceiro passos do assistente de estruturação do conhecimento da versão *desktop*

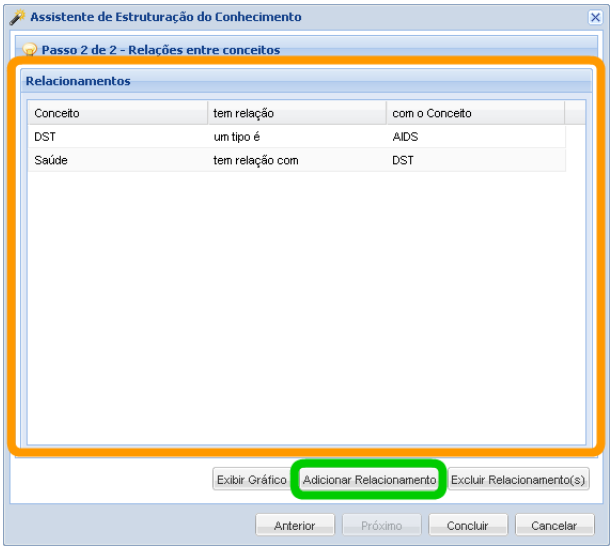
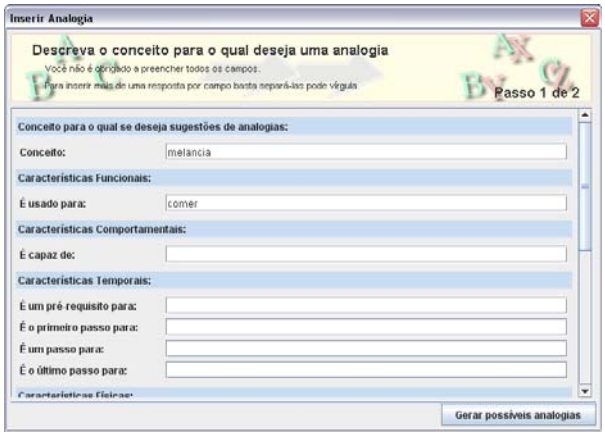


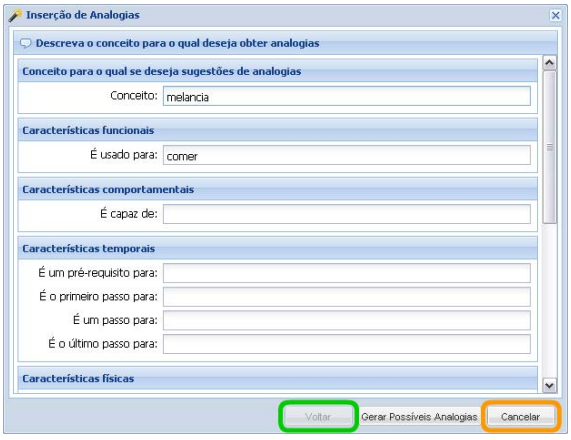
Figura 5.8: Segundo passo do assistente de estruturação do conhecimento da versão Web

5.2.3 Inserção de Analogias

O assistente de inserção de analogias, já explorado no Capítulo 4, é utilizado para criar analogias relacionadas a um determinado conceito. O assistente é composto por dois passos, sendo que no primeiro o usuário define o conceito desejado e suas características e no segundo são apresentadas as analogias geradas. Na versão Web deste assistente, foram inseridos alguns botões para permitir uma maior liberdade dos usuários. A comparação entre as interfaces do primeiro passo pode ser vista na Figura 5.9a e na Figura 5.9b.



(a) Versão *desktop*

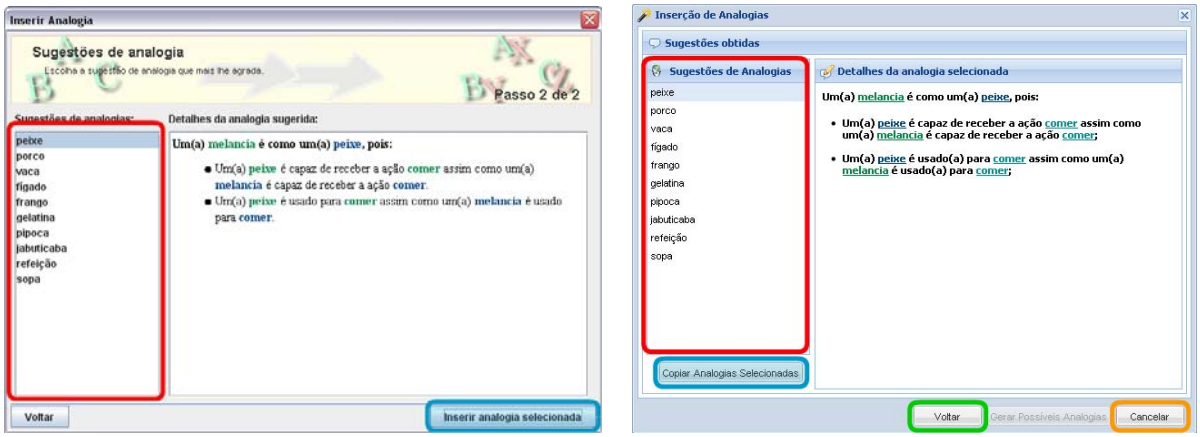


(b) Versão Web

Figura 5.9: Comparação do primeiro passo do assistente de inserção de analogias

Percebe-se na Figura 5.9b a inserção do botão "Cancelar" (destacado em laranja) que permite ao usuário cancelar o assistente. Nota-se que o botão "Voltar" (destacado em verde), no primeiro passo, fica desabilitado. A comparação do segundo passo está

representada na Figura 5.10a e na Figura 5.10b, sendo que os botões "Voltar" e "Cancelar" estão destacados respectivamente em verde e laranja na Figura 5.10b. Percebe-se também, que o botão "Copiar Analogias Selecionadas" (destacado em azul na Figura 5.10b e na Figura 5.10a) foi trocado de lugar, sendo que na versão Web, ele fica localizado abaixo da lista de analogias geradas (destacada em vermelho na Figura 5.10b). Esta mudança foi feita de acordo com os relatos dos usuários, que informaram que seria melhor colocar o botão "Copiar Analogias Selecionadas" perto da lista de analogias.



(a) Versão *desktop* (b) Versão Web

Figura 5.10: Comparação do segundo passo do assistente de inserção de analogias

5.2.4 Preenchimento de Metadados

O assistente de preenchimento de metadados foi a funcionalidade da nova versão da aplicação que mais sofreu alterações. Com o objetivo de manter o assistente mais organizado na versão Web do que na versão *desktop*, a prototipação da nova versão do mesmo foi iniciada a partir da criação de um protótipo em papel, da mesma maneira como foi realizado nos outros assistentes, que foram apresentados nas Seções anteriores. Dentre as principais modificações, pode-se destacar: I) a utilização de abas para organizar as categorias dos campos de metadados do IEEE LOM [24]; II) a utilização de diálogos para a formatação de alguns campos de metadados que possuem formato específico; e III) a simplificação do suporte ao preenchimento de alguns campos através do uso do conhecimento cultural disponibilizado pelo Projeto OMCS-Br. O primeiro protótipo é apresentado na Figura 5.11.

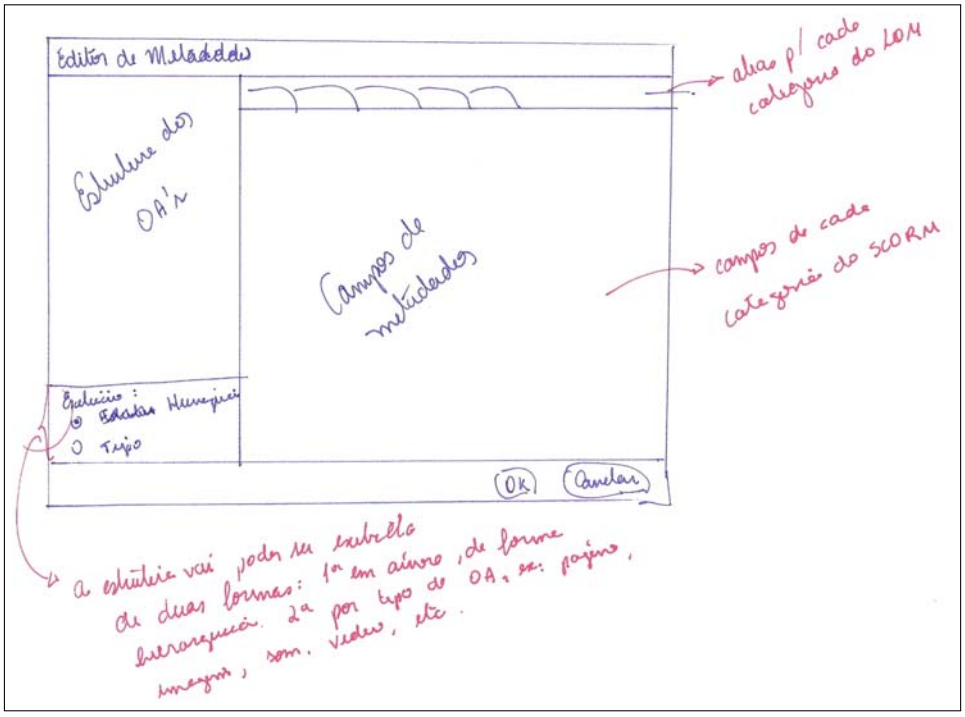


Figura 5.11: Protótipo da interface do assistente de preenchimento de metadados

A primeira modificação que será detalhada é a organização dos campos de metadados em categorias. Na versão *desktop*, todos os campos eram apresentados de forma contínua (destacado em laranja na Figura 5.12). Na versão Web, foram criadas nove abas para organizar cada uma das nove categorias – "Geral, "Ciclo de Vida", "Meta-Metadados", "Técnico", "Educacional", "Direitos", "Relação", "Anotação", "Classificação" – dos campos de metadados do IEEE LOM. Estas abas estão destacadas em laranja na Figura 5.13. Quando uma aba está ativa, seus respectivos campos são exibidos logo abaixo (área destacada em roxo na Figura 5.13).

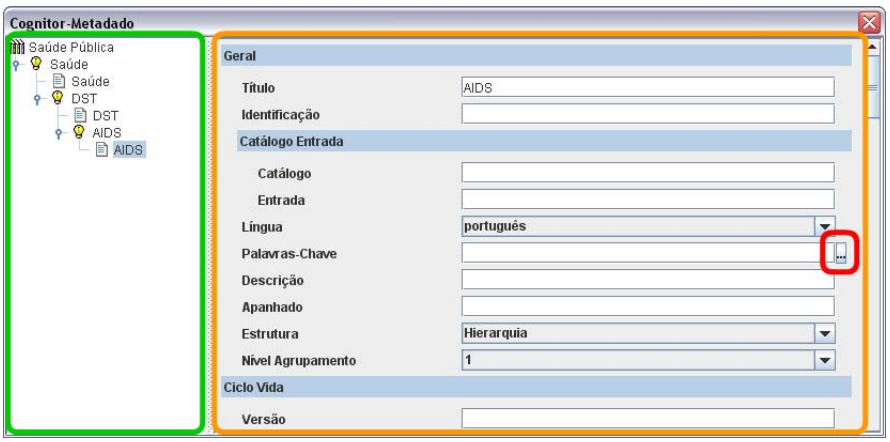


Figura 5.12: Versão *desktop* do assistente de preenchimento de metadados

Outra modificação foi a exibição organizada dos OAs contidos dentro do material. Na

versão *desktop* eram mostradas apenas as páginas de conteúdo que podiam ser anotadas usando metadados (área destacada em verde na Figura 5.12). Na versão Web, como já dito, todos os OAs do material são exibidos de forma organizada. No exemplo da Figura 5.13, na área destacada em verde, pode-se ver que existem cinco OAs no material, sendo três páginas de conteúdo ("Saúde", "DST" e "AIDS"), uma imagem ("Vírus HIV") e um vídeo ("Explicação do HIV").

Na nova versão, também foram inseridas "dicas de ferramenta" (*tooltips*) para cada um dos campos de metadados, permitindo que os usuários da aplicação entendam para que serve cada um dos campos. A dica de ferramenta do campo "descrição" pode ser vista na área destacada em rosa na Figura 5.13. Todas estas modificações foram sugeridas pelos usuários durante a condução de um experimento, apresentado no artigo "*Providing Culturally Contextualized Metadata to Promote Sharing and Reuse of Learning Objects*" [40].

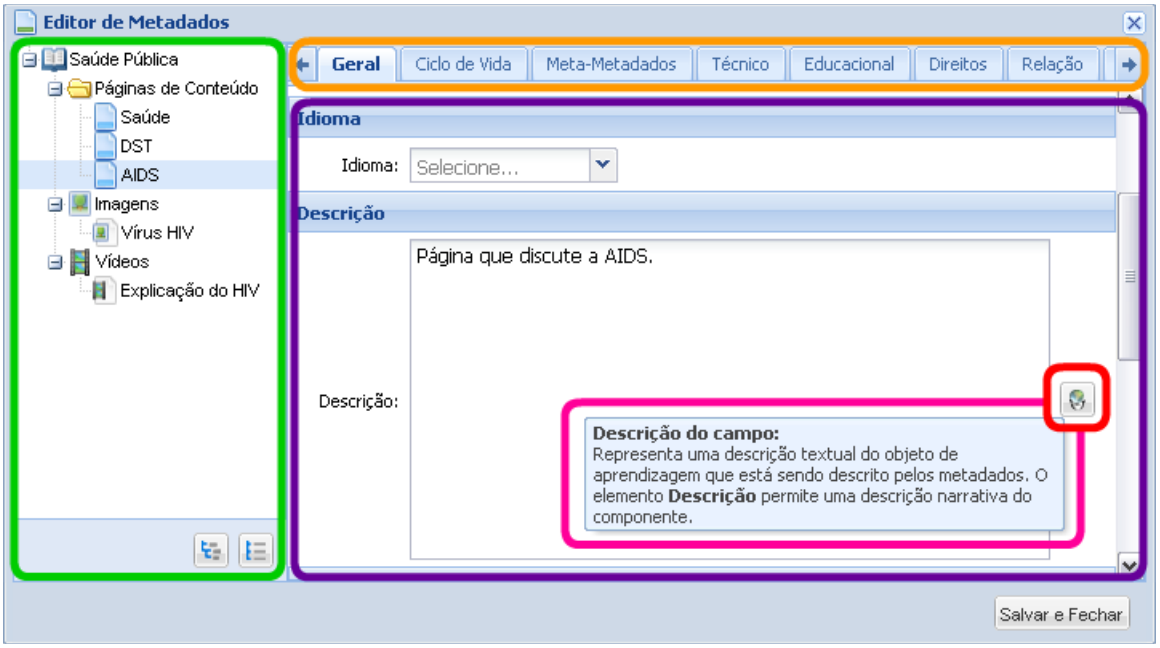


Figura 5.13: Versão Web do assistente de preenchimento de metadados

No mesmo artigo foi proposto também que diversos campos de metadados poderiam ser apoiados, em seu preenchimento, pelo conhecimento cultural disponibilizado pelo Projeto OMCS-Br. Na versão *desktop*, apenas o campo "palavras-chave" possuía este suporte, sendo acessado por meio de um botão, destacado em vermelho na Figura 5.12. Ao clicar neste botão, o assistente de preenchimento do campo "palavras-chave" (Figura 5.14) é exibido. Os usuários reportaram que este assistente era muito confuso, então, para a versão Web, o assistente foi criado de forma a simplificar o seu uso e para poder ser usado no apoio ao preenchimento de diversos campos de metadados que não somente o campo "palavras-chave". O novo assistente pode ser visto na Figura 5.15, destacado em laranja.

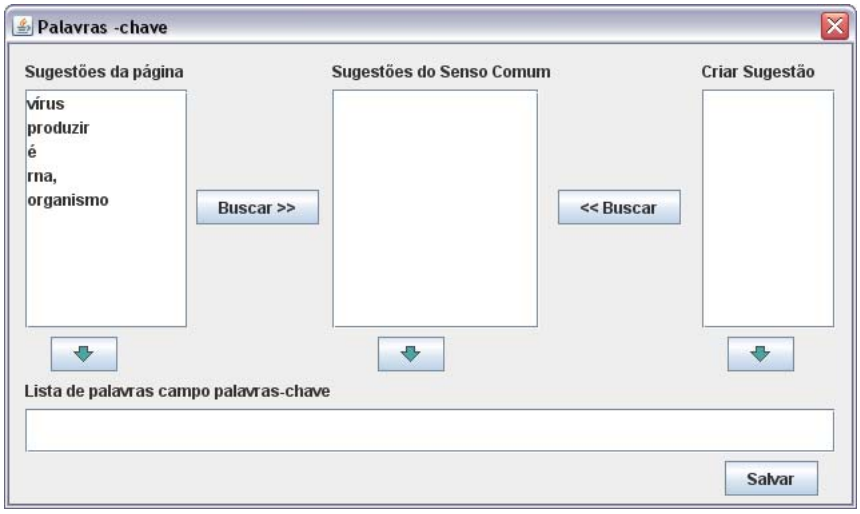


Figura 5.14: Assistente de preenchimento do campo "palavras-chave"

Para acessar o assistente da Figura 5.15, basta clicar no botão que está localizado na frente do campo desejado (destacado em vermelho na Figura 5.13). No exemplo, o campo "descrição" está recebendo o suporte do assistente. Como dito anteriormente, diversos outros campos recebem este suporte, sendo que todos estes têm como característica serem preenchidos com texto livre. Todos os campos que podem receber apoio deste assistente estão organizados na Tabela 5.1.

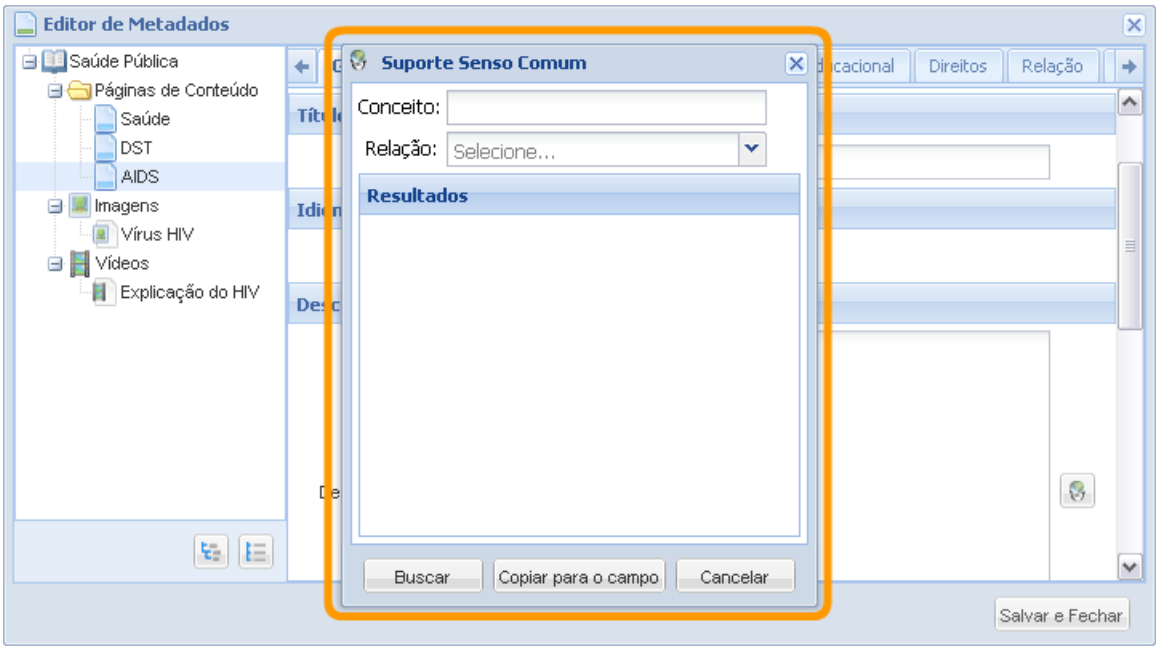


Figura 5.15: Assistente de suporte ao preenchimento de metadados utilizando conhecimento cultural

Tabela 5.1: Campos de metadados com suporte ao preenchimento apoiado pelo conhecimento cultural

Categoria	**Campo**
Geral	descrição
	palavras-chave
	cobertura
Ciclo de Vida	descrição
Meta-Metadados	contribuição/data/descrição*
Técnico	observações de instalação
	outros requisitos da plataforma
	duração/descrição*
Educacional	descrição
Direitos	descrição
Relação	recurso/descrição*
Anotação	data/descrição*
	descrição
Classificação	descrição
	palavras-chave

*A notação "campo1/campo2/.../campoN" significa que "campo1" é composto por partes, sendo "campo2" uma delas e assim sucessivamente.

Por fim, a última modificação realizada no assistente de inserção de metadados, foi tratar a formatação de alguns campos que utilizam um formato específico. Quando um campo precisa ser preenchido de acordo com um determinado formato, utiliza-se um botão situado à sua frente. Na Figura 5.16a pode-se ver um destes botões (destacado em laranja), que, no exemplo ilustrado, é utilizado para preencher o campo de metadados "entidade". Quando o usuário clica no botão, é exibido um diálogo que permite o preenchimento daquele campo. Foram desenvolvidos quatro diálogos diferentes:

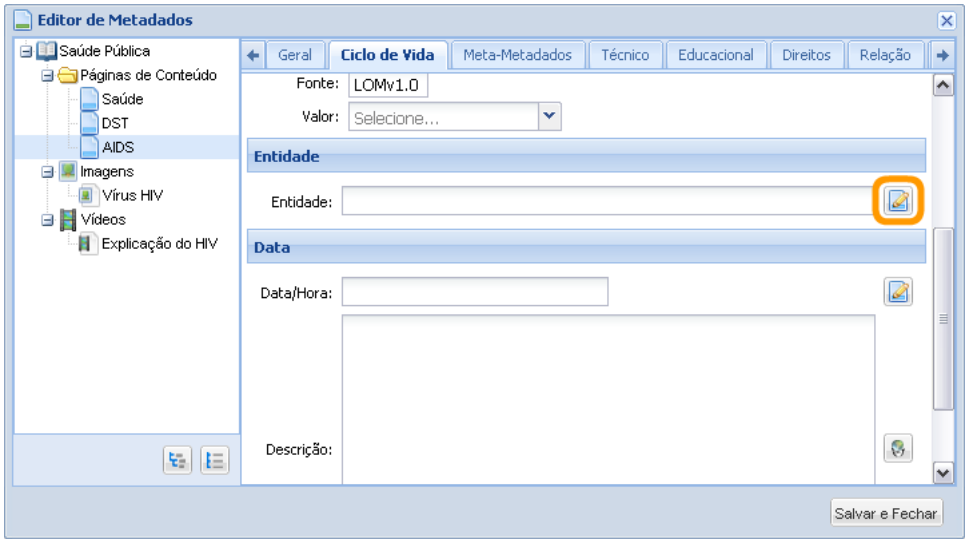
- **vCard**: Formata a descrição de uma entidade (pessoa, empresa, etc), utilizando um formato denominado vCard[1]. É utilizado no campo "entidade" das categorias "Geral", "Ciclo de Vida", "Meta-Metadados" e "Anotação". O diálogo para edição de valores do tipo vCard pode ser visto na Figura 5.16b;

- **Data e Hora**: Formata datas e horas. É utilizado no campo "data/hora" das categorias "Ciclo de Vida", "Meta-Metadados" e "Anotação". Este diálogo é apresentado na Figura 5.16c;

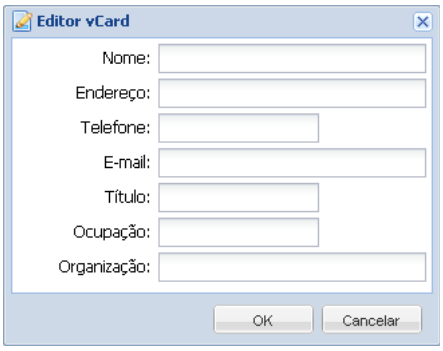
- **Duração**: Formata um intervalo de tempo. É utilizado no campo "duração" das categorias "Técnico" e "Educacional". Este diálogo pode ser visto na Figura 5.16d;

[1] `http://tools.ietf.org/html/rfc2425`

- **Intervalo de Idade**: Formata um intervalo de idades. É utilizado no campo "intervalo" da categoria "Educacional". Este diálogo pode ser visto na Figura 5.16e;

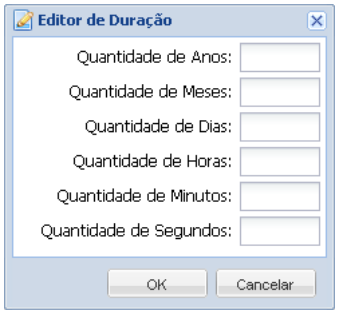


(a) Assistente de preenchimento de metadados

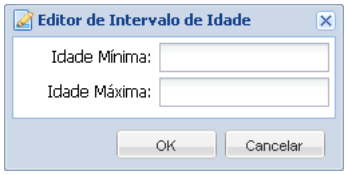


(b) vCard

(c) Data e Hora



(d) Duração



(e) Intervalo de Idade

Figura 5.16: Apoio no preenchimento de campos de metadados de formato específico

5.3 Novas Funcionalidades do Cognitor Web

Para a nova versão do Cognitor, houve a necessidade de implementar algumas funcionalidades que não existiam na versão *desktop*. A primeira delas é o gerenciamento

de usuários. Os administradores da aplicação podem gerenciar os usuários que estão cadastrados, alterando seus dados pessoais, como por exemplo, as permissões que um determinado usuário tem dentro da aplicação. A tela de gerenciamento de usuários pode ser vista na Figura 5.17, onde do lado direito são listados os usuários cadastrados, enquanto do lado esquerdo são preenchidos os dados do usuário selecionado.

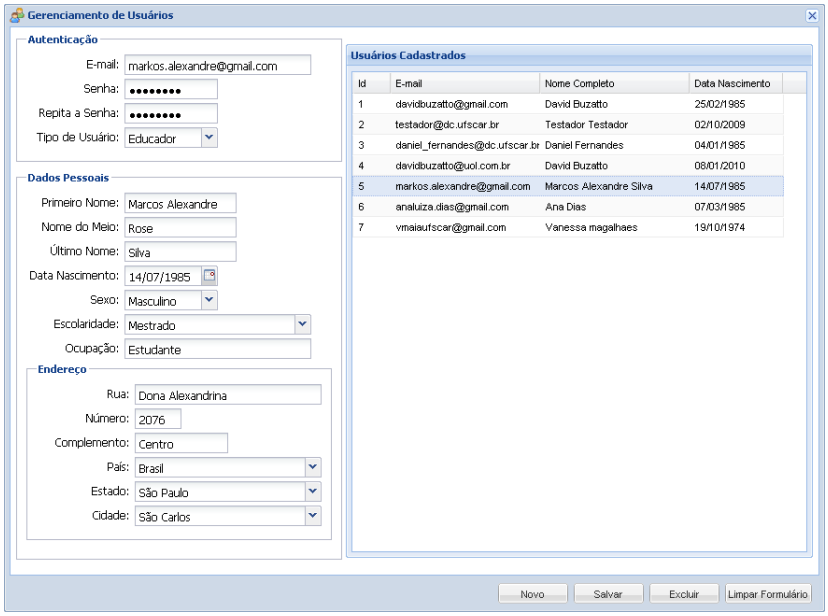


Figura 5.17: Gerenciamento de usuários

Além do gerenciamento de usuários, os administradores podem também gerenciar as configurações globais da ferramenta, como por exemplo, o número da versão e se a ferramenta deve executar em modo de *debug*. A ferramenta de gerenciamento de configurações pode ser vista na Figura 5.18, onde a propriedade "versão" está sendo editada.

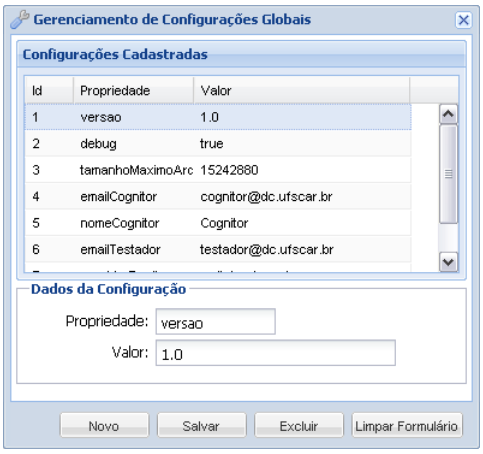


Figura 5.18: Gerenciamento de configurações

Quando um determinado usuário entra na aplicação, ele pode alterar seus dados pessoais, como endereço, senha, etc. Para permitir essas alterações foi desenvolvido um

formulário, sendo este apresentado na Figura 5.19, onde o usuário que tem como e-mail "davidbuzatto@gmail.com" está alterando seus dados pessoais.

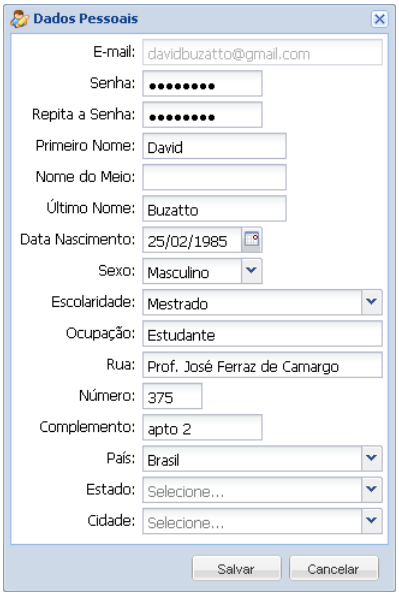


Figura 5.19: Alteração dos dados pessoais

Para permitir que o usuário da aplicação insira mídias dentro de uma página de conteúdo, foi necessário desenvolver três *plugins* para o editor WYSIWYG usado no Cognitor Web. Em cada um desses *plugins*, o usuário pode enviar arquivos de determinadas mídias e, após o envio, inserir a mídia enviada em uma página de conteúdo. Na Figura 5.20 pode ser visto o *plugin* de inserção de imagens, que permite ao usuário enviar para o repositório arquivos de imagem de diversos formatos, além de permitir inseri-los nas páginas de conteúdo.

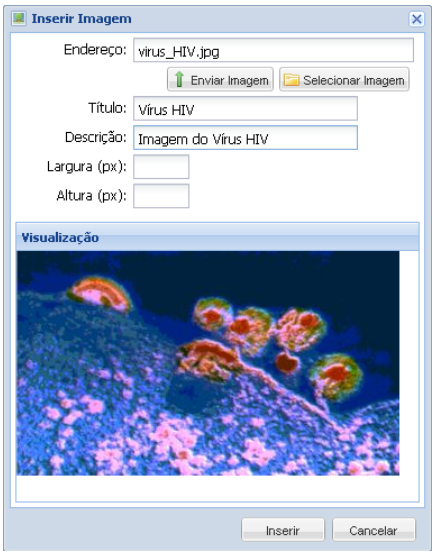
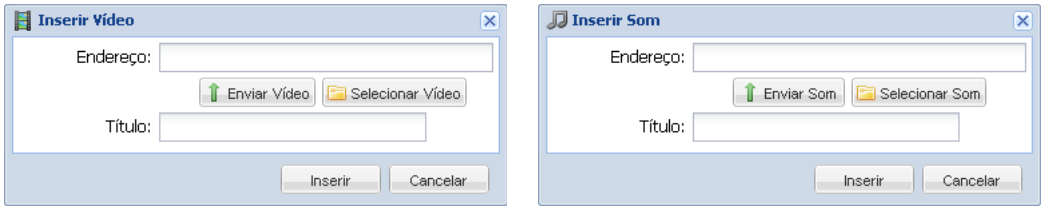


Figura 5.20: Inserção de imagens

Os outros dois *plugins* são utilizados para enviar e inserir vídeos e sons nas páginas de conteúdo. O *plugin* de inserção de vídeos pode ser visto na Figura 5.21a, enquanto o do envio de sons na Figura 5.21b.



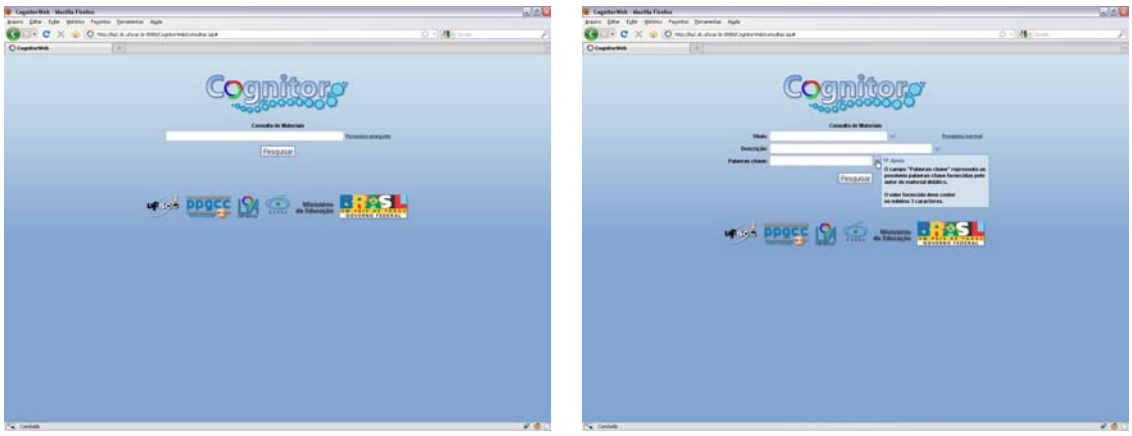
(a) Inserção de vídeos (b) Inserção de sons

Figura 5.21: Inserção de vídeos e sons

Por fim, a última funcionalidade que foi desenvolvida, teve como objetivo permitir a busca e a obtenção dos OAs armazenados no repositório do Cognitor, permitindo assim, que qualquer usuário realize uma consulta neste repositório, podendo então fazer o *download* – nos formatos HTML e/ou SCORM – dos materiais obtidos. A pesquisa pelos OAs pode ser realizada de duas maneiras:

1. **Pesquisa normal**: É realizada uma busca baseada somente no título do material armazenado. A interface deste tipo de pesquisa pode ser vista na Figura 5.22a;

2. **Pesquisa avançada**: É realizada uma busca pelo(s) campo(s) de metadados que se deseja pesquisar. Está disponível, nesta versão, a busca pelos campos "título", "descrição" e "palavras-chave", sendo que estes três fazem parte da categoria "Geral". Esta interface pode ser vista na Figura 5.22b.



(a) Interface para pesquisa normal (b) Interface para pesquisa avançada

Figura 5.22: Interfaces para pesquisa de OAs

Percebe-se aqui a importância do preenchimento dos metadados, pois isto impacta diretamente nos resultados da pesquisa realizada pelo usuário. Um exemplo de pesquisa pode ser visto na Figura 5.23, onde é realizada uma pesquisa avançada por OAs que têm como parte do título o termo "saúde". Podem ser vistos, destacados respectivamente em laranja e verde, os dois OAs que foram obtidos. É possível realizar o *download* de cada OA obtido, sendo que o usuário pode escolher baixar o conteúdo no formato de um pacote de arquivos HTML ou em um pacote SCORM. Os botões que permitem esse *download* podem ser vistos, destacados em vermelho, dentro da área destacada em laranja. A interface de pesquisa de OAs do Cognitor pode ser acessada no endereço `http://lia.dc.ufscar.br/cognitorweb/consultas/`.

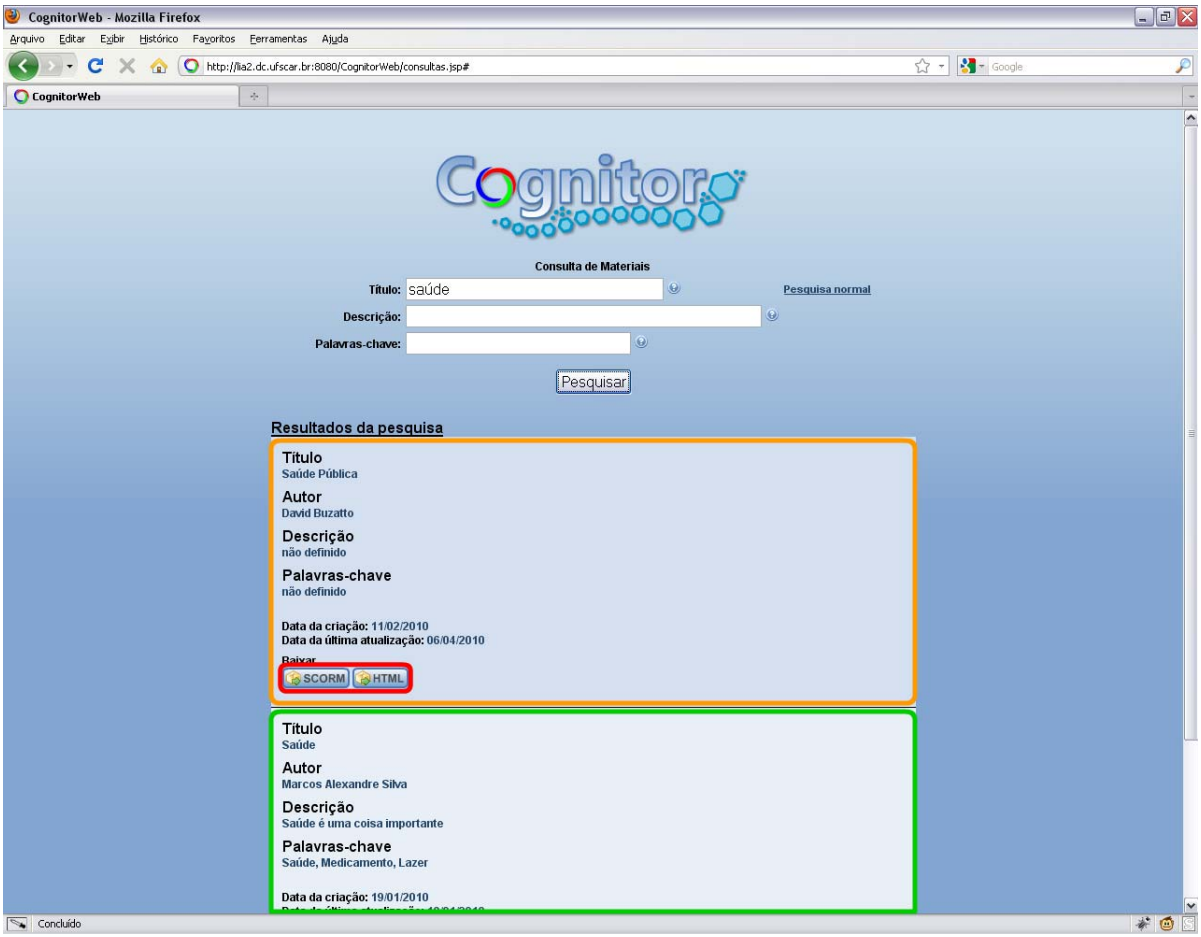


Figura 5.23: Resultados da pesquisa por OAs que têm no título o termo "saúde"

5.4 Considerações Finais

Neste Capítulo foram apresentadas as diferenças entre as versões *desktop* e Web do Cognitor, destacando o processo de recriação das interfaces gráficas de determinadas fun-

cionalidades. Estas alterações foram validadas, em todos os casos, pelos usuários da ferramenta, durante a fase de Prototipação. Além das funcionalidades existentes na versão *desktop* que foram migradas para a versão Web, foram também discutidas as novas funcionalidades que precisaram ser desenvolvidas na versão Web. Foi discutida também a interface de consulta de OAs, criada com o objetivo de permitir que qualquer usuário consiga consultar e obter os OAs existentes dentro do repositório do Cognitor. No próximo Capítulo, a conclusão deste projeto e os possíveis trabalhos futuros serão apresentados.

6 Conclusão e Trabalhos Futuros

6.1 Síntese dos Principais Resultados

Este trabalho apresentou o UC-RIA, um modelo de processo de reengenharia de software centrado no usuário, que tem como objetivo guiar a conversão de aplicações *desktop* em RIAs, sendo que, durante este processo, os potenciais usuários finais da nova aplicação têm papel central na tomada de decisões relacionadas à construção das interfaces gráficas.

O UC-RIA foi formalizado – utilizando a PML SPEM – a partir do modelo percebido durante a organização dos passos realizados na reengenharia da ferramenta Cognitor, sendo que essa organização se deu por meio do uso de uma tabela, baseada no UCD, proposta por Anacleto, Fels e Villena [5].

Os usuários participam do desenvolvimento da aplicação durante a fase de Prototipação do UC-RIA, nos casos onde a funcionalidade escolhida para ser implementada é expressa por meio de interfaces gráficas. Durante este processo, um protótipo da funcionalidade é desenvolvido e em seguida, este protótipo é avaliado pelo usuário, que por sua vez o aprova ou o rejeita. Caso o protótipo seja aprovado, o desenvolvimento prossegue, caso seja rejeitado, ele é alterado – com base nas observações do usuário – e um novo teste é realizado. Este processo de teste/alteração se repete até que o protótipo seja aprovado.

Pelo exposto, pode-se então perceber que o UC-RIA é um modelo de processo de reengenharia de software que pode ser aplicado com sucesso durante o processo de conversão de aplicações *desktop* em RIAs.

6.1.1 Contribuições ao Estado da Arte

O desenvolvimento deste projeto resultou em duas importantes contribuições para o estado da arte. A primeira delas, o objetivo principal do projeto, foi a criação do UC-RIA, permitindo que equipes de desenvolvimento de software insiram os usuários das aplicações no processo de criação das mesmas, apoiando assim as tomadas de decisões relacionadas

à construção das interfaces gráficas da aplicação.

Outra contribuição foi a validação – por meio do artigo "*Providing Culturally Contextualized Metadata to Promote Sharing and Reuse of Learning Objects*" [40] – da utilização do conhecimento cultural no apoio ao preenchimento de diversos campos de metadados (norma IEEE LOM) dos OAs criados pelo Cognitor.

6.1.2 Artigos Publicados

Durante o processo de elaboração deste trabalho foram publicados quatro artigos. Dois destes artigos estão relacionados diretamente a este projeto de mestrado e tratam principalmente da importância do preenchimento dos metadados dos OAs, que podem, por sua vez, serem apoiados em seu preenchimento pelo uso do conhecimento cultural disponibilizado pelo Projeto OMCS-Br. A publicação dos outros dois artigos foi produto da contribuição do aluno, em parceria com outros pesquisados do LIA, durante a elaboração de seus respectivos projetos. Apesar destes dois últimos artigos não terem ligação direta com este projeto, eles estão relacionados ao Projeto OMCS-Br, base da maioria dos projetos do laboratório. A seguir, a lista dos artigos publicados.

- Artigos relacionados a este projeto:
  - "*Providing Culturally Contextualized Metadata to Promote Sharing and Reuse of Learning Objects*" [40];
  - "*Filling out Learning Object Metadata Considering Cultural Contextualization*" [41].
- Artigos relacionados ao Projeto OMCS-Br:
  - "*Web Collaboration Motivated by Colors Emotionally Based on Common Sense*" [42];
  - "*A Game to Support Childrens' Expression and Socialization Considering their Cultural Background*" [43].

6.1.3 Problemas e Limitações

Alguns problemas surgiram durante o processo de reengenharia do Cognitor, sendo que a maioria deles estavam relacionados justamente à migração da aplicação *desktop* para a aplicação rica, dada a necessidade de construir interfaces iguais as da aplicação original. Todos esses problemas puderam ser contornados satisfatoriamente graças a escolha da

biblioteca de componentes – ExtJS – para construção de interfaces gráficas. Talvez, com o advento da versão 5 do HTML, os problemas inerentes ao desenvolvimento de aplicações ricas sejam suavizados.

Outro problema foi o de compatibilidade entre navegadores, pois não houve tempo suficiente para fazer com que o Cognitor Web funcionasse corretamente no navegador Internet Explorer. Nos navegadores Firefox e Chrome a ferramenta funciona de forma satisfatória, no navegador Opera alguns problemas ainda existem e no navegador Safari a ferramenta ainda não foi testada.

6.2 Trabalhos Futuros

Apesar do funcionamento satisfatório do UC-RIA e da ferramenta criada a partir deste processo, algumas melhorias podem ser feitas tanto no modelo de processo, quanto no Cognitor Web.

O UC-RIA formaliza a escolha das bibliotecas para construção de interfaces gráficas e especifica que deve existir uma fase de prototipação antes do desenvolvimento de alguma funcionalidade, que é expressa por meio de interfaces gráficas, entretanto, na fase de testes com usuários (após a implementação da funcionalidade), não é especificada nenhuma estratégia de testes. Outro trabalho futuro relacionado ao modelo de processo é expandi-lo de forma a permitir que seja feita a conversão de aplicações Web convencionais em RIAs. Pretende-se também validar o UC-RIA em eventos, através da publicação de artigos.

Quanto à ferramenta criada, o Cognitor Web, existem diversas frentes que podem ser analisadas com o objetivo de melhorá-la. Uma delas é aumentar o nível de portabilidade entre os navegadores Web, permitindo que uma maior quantidade de usuários possa utilizá-la. Outra possibilidade de melhoria é a implementação de um mecanismo de edição colaborativa dos OAs, permitindo que vários usuários sejam coautores dos materiais didáticos. Por fim, uma funcionalidade importante que pode se abordada, é a criação de um sistema de recomendação para os OAs criados dentro na ferramenta.

Referências

[1] OMG. *Software & Systems Process Engineering Meta-Model Specification, version 2.0*. 2008. Disponível em: <http://www.omg.org/cgi-bin/doc?formal/08-04-01.pdf>. Acesso em: 16 de março de 2010.

[2] BYRNE, E. A conceptual foundation for software re-engineering. In: *Proceedings of the Conference on Software Maintenance, 1992*. Los Alamitos, CA, USA: IEEE Computer Society, 1992. p. 226–235.

[3] YANG, X. et al. A dual-spiral reengineering model for legacy system. In: *TENCON 2005 IEEE Region 10*. Melbourne, Victoria, Australia: Swinburne Press, 2005. p. 1–5.

[4] TALARICO NETO, A. *Linguagem de Padrões para Apoiar o Projeto de Material Instrucional para EAD*. 137 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação, UFSCar, 2005.

[5] ANACLETO, J. C.; FELS, S.; VILLENA, J. M. R. Design of a web-based therapist tool to promote emotional closeness. In: *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*. New York, NY, USA: ACM, 2010. p. 3565–3570.

[6] LIU, H.; SINGH, P. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, Kluwer Academic Publishers, Hingham, MA, USA, v. 22, n. 4, p. 211–226, 2004.

[7] OSBORNE, W. M.; CHIKOFSKY, E. J. Fitting pieces to the maintenance puzzle. *IEEE Software*, v. 7, n. 1, p. 11–12, jan. 1990.

[8] DÍSCOLA JUNIOR, S. L.; SILVA, J. C. A. Processes of software reengineering planning supported by usability principles. In: *CLIHC '03: Proceedings of the Latin American conference on Human-computer interaction*. New York, NY, USA: ACM, 2003. p. 223–226.

[9] DÍSCOLA JUNIOR, S. L.; SILVA, J. C. A. Reengineering planning process guided by usability evaluation. In: *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2004. p. 1565–1565.

[10] ANACLETO, J. C. et al. Ambiente para criação de jogos educacionais de adivinhação baseados em cartas contextualizadas. In: *WIE - Workshop sobre Informática na Escola*. Porto Alegre, RS, Brasil: SBC, 2008. p. 29–38.

[11] NORMAN, D. A.; DRAPER, S. W. *User Centered System Design: New Perspectives on Human-computer Interaction*. Boca Raton, FL, USA: CRC Press, 1986. 544 p.

[12] SHNEIDERMAN, B. et al. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 5. ed. Reading, MA, USA: Addison Wesley, 2009. 624 p.

[13] CHIKOFSKY, E. J.; CROSS, J. H. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, v. 7, n. 1, p. 13–17, jan. 1990.

[14] MACIEL, R. S. P. et al. An integrated approach for model driven process modeling and enactment. In: *SBES '09. XXIII Brazilian Symposium on Software Engineering, 2009.* Porto Alegre, RS, Brasil: SBC, 2009. p. 104–114.

[15] DAMI, S.; ESTUBLIER, J.; M., A. Apel: A graphical yet executable formalism for process modeling. *Automated Software Engineering*, v. 5, n. 1, p. 61–96, jan. 1998.

[16] FRANCH, X.; RIBó, J. M. Using uml for modelling the static part of a software process. In: *UML '99: Beyond the Standard*. Berlin, Germany: Springer-Verlag, 1999. p. 292–307.

[17] BATTAGLIA, M.; SAVOIA, G.; FAVARO, J. Renaissance: a method to migrate from legacy to immortal software systems. In: *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering, 1998.* Stoughton, WI, USA: The Printing House, 1998. p. 197–200.

[18] CHU, W. C. et al. Pattern-based software reengineering: a case study. *Journal of Software Maintenance*, John Wiley & Sons, Inc., New York, NY, USA, v. 12, n. 2, p. 121–141, 2000.

[19] BIANCHI, A. et al. Iterative reengineering of legacy systems. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 29, n. 3, p. 225–241, 2003.

[20] BOEHM, B. W. A spiral model of software development and enhancement. *Computer*, v. 21, n. 5, p. 61–72, may. 1988.

[21] WILLIAMS, A. User-centered design, activity-centered design, and goal-directed design: a review of three methods for designing web applications. In: *SIGDOC '09: Proceedings of the 27th ACM international conference on Design of communication*. New York, NY, USA: ACM, 2009. p. 1–8.

[22] DEITEL, P. J.; DEITEL, H. M. *Ajax, Rich Internet Applications, and Web Development for Programmers*. Boston, MA, USA: Prentice Hall, 2008. 1040 p. (Deitel Developer Series).

[23] TALARICO NETO, A. et al. A framework to support the design of learning objects based on the cog-learn pattern language. In: *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2006. p. 128–137.

[24] IEEE. *IEEE 1484.12.1-2002 Draft Standard for Learning Object*. 2002. Disponível em: <http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf>. Acesso em: 15 de março de 2010.

[25] FARRELL, R. G.; LIBURD, S. D.; THOMAS, J. C. Dynamic assembly of learning objects. In: *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM, 2004. p. 162–169.

[26] SCORM. *SCORM 2004 4th Edition Version 1.1 Documentation*. 2009. Disponível em: <http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/SCORM 2004 4th Ed V1.1/Documentation Suite/SCORM_2004_4ED_v1_1_Doc_Suite.zip>. Acesso em: 16 de março de 2010.

[27] ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M. *A Pattern Language: towns, buildings, construction*. New York, NY, USA: Oxford University Press, 1977. 1171 p.

[28] WELIE, M. van; VEER, G. C. van der. Pattern languages in interaction design: Structure and organization. In: *Human-Computer Interaction – INTERACT'03*. Amsterdam, The Netherlands: IOS Press, 2003. p. 527–534.

[29] FINCHER, S. Perspectives on HCI patterns: concepts and tools (introducing PLML). In: *20th CHI - Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2003. p. 26–28.

[30] BORCHERS, J. *A Pattern Approach to Interaction Design*. Chichester, UK: John Wiley & Sons, 2001. 268 p.

[31] NERIS, V. P. A. et al. Hyper documents with quality for distance learning: cognitive strategies to help teachers in the navigational project and content organization. In: *WebMedia '05: Proceedings of the 11th Brazilian Symposium on Multimedia and the web*. New York, NY, USA: ACM, 2005. p. 1–7.

[32] CARLOS, A. J. F. *Aplicando Senso Comum na Edição de Objetos de Aprendizagem*. 71 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação, UFSCar, 2008.

[33] ANACLETO, J. C. et al. Using common sense knowledge to support learning objects edition and discovery for reuse. In: *WebMedia '07: Proceedings of the 13th Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2007. p. 290–297.

[34] SINGH, P.; BARRY, B.; LIU, H. Teaching machines about everyday life. *BT Technology Journal*, Kluwer Academic Publishers, Hingham, MA, USA, v. 22, n. 4, p. 227–240, 2004.

[35] TSUTSUMI, M. *Uso de senso comum na detecção das diferenças culturais no contexto do projeto Open Mind Common Sense*. 137 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação, UFSCar, 2006.

[36] FALBO, R. A. et al. Ontologias e ambientes de desenvolvimento de software semânticos. In: *JIISIC - Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento*. [s.n.], 2004. p. 16. Disponível em: <http://www.inf.ufes.br/ falbo/download/pub/2004-JIISIC-1.pdf>. Acesso em: 16 de março de 2010.

[37] MINSKY, M. *The Society of Mind*. New York, NY, USA: Simon and Schuster, 1988. 336 p.

[38] NOVAK, J. D. *A Theory of Education*. New York, NY, USA: Cornell University Press, 1986. 296 p.

[39] AUSUBEL, D. P. *Educational Psychology: A Cognitive View*. New York, NY, USA: Holt, Rinehart and Winston, 1978. 733 p.

[40] BUZATTO, D.; ANACLETO, J. C.; DIAS, A. L. Providing culturally contextualized metadata to promote sharing and reuse of learning objects. In: *SIGDOC '09: Proceedings of the 27th ACM international conference on Design of communication*. New York, NY, USA: ACM, 2009. p. 163–170.

[41] BUZATTO, D. et al. Filling out learning object metadata considering cultural contextualization. In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. Red Hook, NY, USA: Curran Associates, 2009. p. 424–429.

[42] DIAS, A. et al. Web collaboration motivated by colors emotionally based on common sense. In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. Red Hook, NY, USA: Curran Associates, 2009. p. 801–806.

[43] SILVA, M.; ANACLETO, J.; BUZATTO, D. A game to support childrens' expression and socialization considering their cultural background. In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. Red Hook, NY, USA: Curran Associates, 2009. p. 1230–1235.

APÊNDICE A – Caminho Percorrido Durante o Processo de Reengenharia

A tabela que contém todo o caminho percorrido durante o processo de reengenharia do Cognitor pode ser encontrada no diretório “apêndices” do CD, com o nome de CaminhoPercorrido.pdf.

APÊNDICE B – Especificação do UC-RIA

A especificação do UC-RIA pode ser encontrada no diretório “apêndices/uc-ria” do CD. Para abrir a especificação basta executar o arquivo index.html.

APÊNDICE C – Arquitetura do Cognitor Web

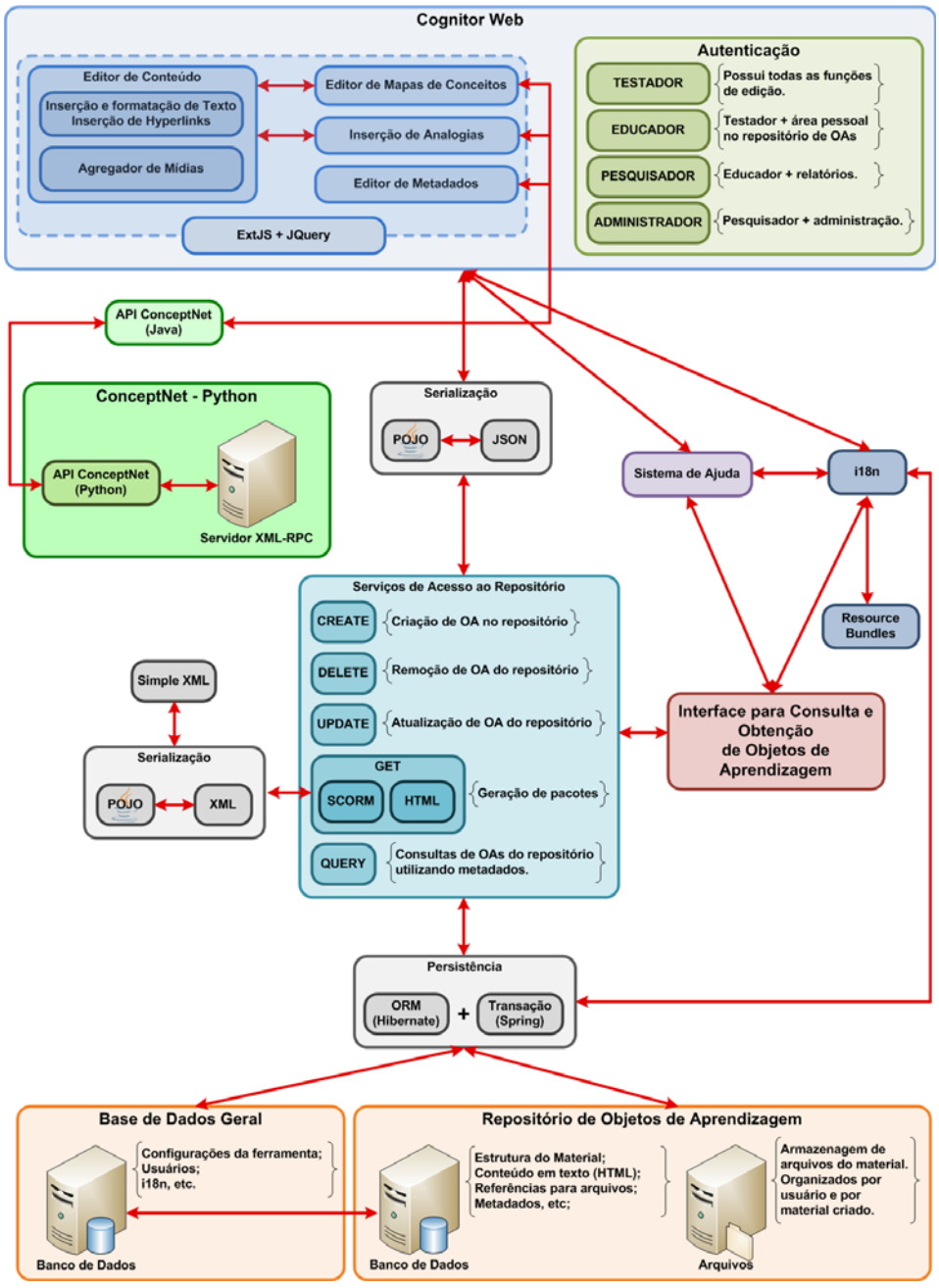


Figura C.1: Arquitetura do Cognitor Web

APÊNDICE D – Código Fonte do Cognitor Web

O código fonte do Cognitor Web pode ser encontrado no diretório "fontesCognitorWeb" do CD na forma de um projeto Web da IDE NetBeans[1]. Para abrir o projeto, recomenda-se utilizar o NetBeans 6.8.

[1] `http://www.netbeans.org/`