UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

EDITOR GRÁFICO PARA SEQUENCIAMENTO E NAVEGAÇÃO USANDO REDES DE PETRI

MAICON MUELLER

BLUMENAU
2012

2012/2-18

MAICON MUELLER

EDITOR GRÁFICO PARA SEQUENCIAMENTO E NAVEGAÇÃO USANDO REDES DE PETRI

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M Sc. - Orientador

BLUMENAU
2012

2012/2-18

EDITOR GRÁFICO PARA SEQUENCIAMENTO E NAVEGAÇÃO USANDO REDES DE PETRI

Por

MAICON MUELLER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M. Sc – Orientador, FURB
Membro:	Prof. Marcel Hugo, M. Sc – FURB
Membro:	Prof. Alexander Roberto Valdameri, M. Sc – FURB

Blumenau, 12 de dezembro de 2012

Dedico este trabalho a minha noiva, aos meus pais, meu irmão todos os amigos, especialmente aqueles que me ajudaram diretamente na realização desta obra.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha noiva pela paciência, cobrança e confiança depositada em mim.

À minha família, que sempre esteve presente dando apoio nos momentos difíceis.

Aos todos meus amigos, que me ajudaram direta ou indiretamente.

Ao meu orientador, Dalton Solano dos Reis, por ter acreditado na conclusão deste trabalho, mesmo diante das dificuldades.

No que diz respeito ao desempenho, ao compromisso, ao esforço, a dedicação, não existe meio termo. Ou você faz uma coisa bem feita ou não faz.

Ayrton Senna

RESUMO

Este trabalho apresenta o desenvolvimento de um ambiente web para criação de cursos online, a fim de permitir o acesso de qualquer lugar que tenha disponibilidade de acesso a internet. A ferramenta possui uma área para criação, edição e navegação pelo curso, que utiliza modelagem gráfica de redes de petri para montagem dos cursos. Além do mais há possibilidade de armazenamento das informações em banco de dados, upload e download dos cursos, utilizando os recursos de um webservice para a troca de mensagem. A ferramenta foi desenvolvida na linguagem Java, que utilizou o framework GWT que utiliza a metodologia AJAX, JSNI possibilitando escrever código Javascript em fonte Java, e o protocolo RPC fazendo a comunicação cliente e servidor, isso em conjunto com a biblioteca Smartgwt para facilitar na criação das telas.

Palavras-chave: SCORM. EAD. Redes de Petri. Cursos. Upload. Download. GWT. Smartgwt.

ABSTRACT

This work presents the development of a Web environment that allows the creation of online courses, to enable the access from anywhere with internet connection available. The tool has an area to creation, edition and navigation on the course which uses the graphical modeling of Petri's networks to model the courses. Beyond that, it also allows the storage of information in Database, upload and download of the courses, using the resources from a webservice to messages exchange. The tool was developed in Java using the GWT framework which makes use of the technologies AJAX and JSNI, enabling to write Javascript code in Java's source code, and, the RPC protocol does the communication between the client and the server together with the Smartgwt library to facilitate the creation of the screens.

Key-words: SCORM. EAD. Petri Nets. Courses. Upload. Download. GWT. Smartgwt.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ADL – *Advanced Distributed Learning*

AJAX – *Asynchronous* Javascript And XML

API – *Application Programming Interface*

GWT – *Google Web Toolkit*

JSNI – *Java Script Native Interface*

MER – Modelo de Entidade e Relacionamento

MIT – *Massachussetts Institute of Techonoly*

PIF – *Package Interchange File*

RF – Requisito Funcional

RNF – Requisito Não Funcional

RN – Regra de Negócio

RPC – *Remote Method Call*

SCORM – *Sharable Content Object Reference Model*

SGBD – Sistema de Gerenciamento de Banco de Dados

SN – Sequenciamento e Navegação

XML – *eXtensible Markup Language*

UC – Casos de Uso

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Com a expansão do ensino à distância foram necessários o estabelecimento de padrões e especificações internacionais visando a confecção de materiais instrucionais providos de certa inteligência e que sejam portáveis entre os ambientes (SU et al., 2005).

O *Sharable Content Object Reference Model* (SCORM) é um padrão internacional que usa metadados para especificar a estrutura de cada objeto de aprendizagem e propõe um sistema de agregação de conteúdo para compactar estes objetos com formato *eXtensible Markup Language* (XML) (SU et al., 2005). Além disso, ele possui a definição de Sequenciamento e Navegação (SN) que são regras que podem ser usadas para controlar a sequência, seleção e fornecimento de atividades de aprendizagem de acordo com a evolução do aluno.

Existem ferramentas de autoria que seguem a especificação SCORM, como por exemplo, MosSolo, ExeLearning e CourseLab. Algumas destas não implementam o SN, e aquelas que implementam, não possuem uma visualização do processo de regras como um todo. Portanto, é complexo rastrear qualquer situação relativa pelos criadores de conteúdo no modelo de definição de sequenciamento.

Lin et al. (2005) e Su et al. (2005) propuseram o uso de redes de Petri para representar o processo de SN. Uma rede de Petri é representada por um grafo dirigido bipartido em que os vértices ou são lugares ou transições, onde lugares representam condições e as transições representam atividades.

Visto o acima, o presente trabalho desenvolveu um ambiente web de acordo com as representações propostas por Lin et al. (2005) e Su et al. (2005), permitindo que se abram pacotes SCORM, editem o SN com redes de Petri, e por final reempacotem.

Este ambiente foi desenvolvido usando a *Application Programming Interface* (API) Smartgwt (Smartgwt, 2011) que facilita a aplicação de interfaces mais sofisticadas para o ambiente web, além da utilização de `applet` para criação do editor.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um editor gráfico para o SN de pacotes utilizando redes de Petri.

Os objetivos específicos do trabalho são:

a) visualizar pacotes de cursos, baseando-se na SN utilizada nas redes de Petri;

b) prover no ambiente a edição dos pacotes (desempacotar, editar e reempacotar).

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em quatro capítulos intitulados respectivamente como: introdução, fundamentação teórica, desenvolvimento e conclusões.

O capítulo dois apresenta os aspectos teóricos estudados para o desenvolvimento do trabalho, que permite um melhor entendimento. Também são relacionados alguns trabalhos correlatos.

No capítulo três é descrito como foi realizado o desenvolvimento deste trabalho, contemplando os requisitos da ferramenta, a especificação e a implementação. Por fim são apresentados os resultados e discussão encontrados com a finalização do trabalho.

O capítulo quarto traz conclusões do trabalho, bem como alguns aspectos que ficaram em aberto, servindo de sugestões para futuras extensões.

2 FUNDAMENTAÇÃO TEÓRICA

No capítulo são abortados os principais assuntos em relação ao trabalho. Na seção 2.1 é abordado o conceito de utilização de redes de Petri. A seção 2.2 é apresentado o conceito de SCORM. Na seção 2.3 é apresentado o Google web toolkit (GWT). A seção 2.4 apresenta a biblioteca SmartGWT. Por fim, na seção 2.5 são apresentados os trabalhos correlatos ao trabalho proposto.

2.1 REDES DE PETRI

As redes de Petri são um modelo gráfico e matemático que surgiu a partir de uma tese intitulada “Comunicação com Autômatos”, em 1962 no qual seu autor foi Carl Adam Petri, na universidade de Darmstadt, na Alemanha. Mas, somente entre os anos de 1968 e 1976 que despertou atenção de um grupo de pesquisadores da Massachussetts Institute of Technology (MIT) nos Estados Unidos. Este grupo se dedicou estudar o assunto, que mais tarde se tornaria conhecido como redes de Petri, sendo seu principal foco a modelagem de sistemas (CARDOSO; VALETTE, 1997, p. 13).

Segundo Cardoso e Valette (1997, p. 13), as redes de Petri são uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes. Além do mais afirmam que este modelo é poderoso para representar os diferentes processos existentes num sistema, permitindo estruturar de forma organizada a modelagem (CARDOSO; VALETTE, 1997, p. 19). Assim pode-se compreender que as redes de Petri possuem um grande campo de utilização, podendo detalhar os mais variados sistemas.

2.1.1 Elementos

Uma estrutura de rede de Petri é composta pelos seguintes elementos (PENHA, D. O. ; FREITAS, H. C. ; MARTINS, C. A. P. S., 2004):

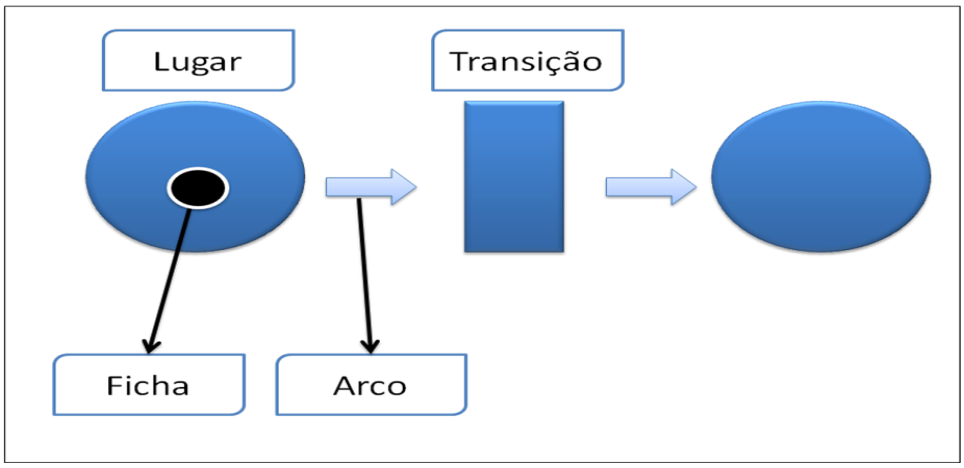
a) lugares: representam uma condição, uma atividade ou um recurso;

b) fichas: representam o estado de um sistema;

c) transições: representam um evento;

d) arcos: indicam os lugares de entrada ou saída para as transições.

Os elementos de uma rede de Petri estão representados graficamente na Figura 1.



Fonte: Penha; Freitas e Martins (2004).

Figura 1 – Elemento de uma rede de Petri

2.1.2 Vantagens e Desvantagens

De acordo com Cardoso e Valette (1997, p. 14), as vantagens na utilização das redes de Petri podem ser resumidas pelas seguintes considerações:

a) pode-se descrever uma ordem parcial entre vários eventos, o que possibilita levar-se em conta a flexibilidade;

b) os estados, bem como os eventos são representados explicitamente;

c) uma única família de ferramentas é utilizada através da especificação, da modelagem, da análise, da avaliação do desempenho e da implementação;

d) uma única família de ferramentas é utilizada nos diversos níveis da estrutura hierárquica do controle, o que facilita a integração destes níveis;

e) uma descrição precisa e formal das sincronizações torna-se possível, o que é essencial para alcançar-se a necessária segurança de funcionamento.

Porém, existem algumas desvantagens na utilização das redes de Petri, a grande quantidade de teoria que o assunto apresenta, assim como o fato da grande maioria de literaturas encontradas sobre o assunto ser muito matemática e pouco prática (REISIG, 1992, p. VIII).

2.2 SCORM

O Sharable Content Object Reference Model (SCORM), ou em português modelo de referência para objetos de conteúdo compartilhado, é um padrão internacional que usa metadados para especificar a estrutura de cada objeto de aprendizagem e propõe um sistema de agregação de conteúdo para compactar estes objetos com formato eXtensible Markup Language (XML) (SU et al., 2005). Além disso, ele possui a definição de Sequenciamento e Navegação (SN) que são regras que podem ser usadas para controlar a sequência, seleção e fornecimento de atividades de aprendizagem de acordo com a evolução do aluno. O padrão SCORM é mantido pela ADL (Advanced Distributed Learning) (ADL, 2012), que é responsável pelas adaptações e liberações das versões.

As principais características do SCORM (ADL, 2006) são:

a) Acessibilidade: a habilidade de alocar e acessar componentes instrucionais de uma localização remota e entregá-los para muitos outros destinos;
b) Adaptabilidade: a habilidade de modificar uma instrução de acordo com necessidades individuais e organizacionais;
c) Rentabilidade: a habilidade de se aumentar a eficiência e a produção ao reduzir o tempo e o custo envolvidos na entrega da instrução;
d) Durabilidade: a habilidade de acompanhar a evolução e mudança de uma tecnologia sem arcar com reprojeto, reconfiguração ou recodificação;
e) Interoperabilidade: a habilidade de capturar componentes desenvolvidos em um local com um conjunto de ferramentas ou plataforma e utilizá-los em outro local com um diferente conjunto de ferramentas ou plataforma;
f) Reusabilidade: a flexibilidade para se incorporar componentes instrucionais em múltiplos contextos.

A especificação SCORM é divida em três partes (VAHLDICK, 2008, p. 21):

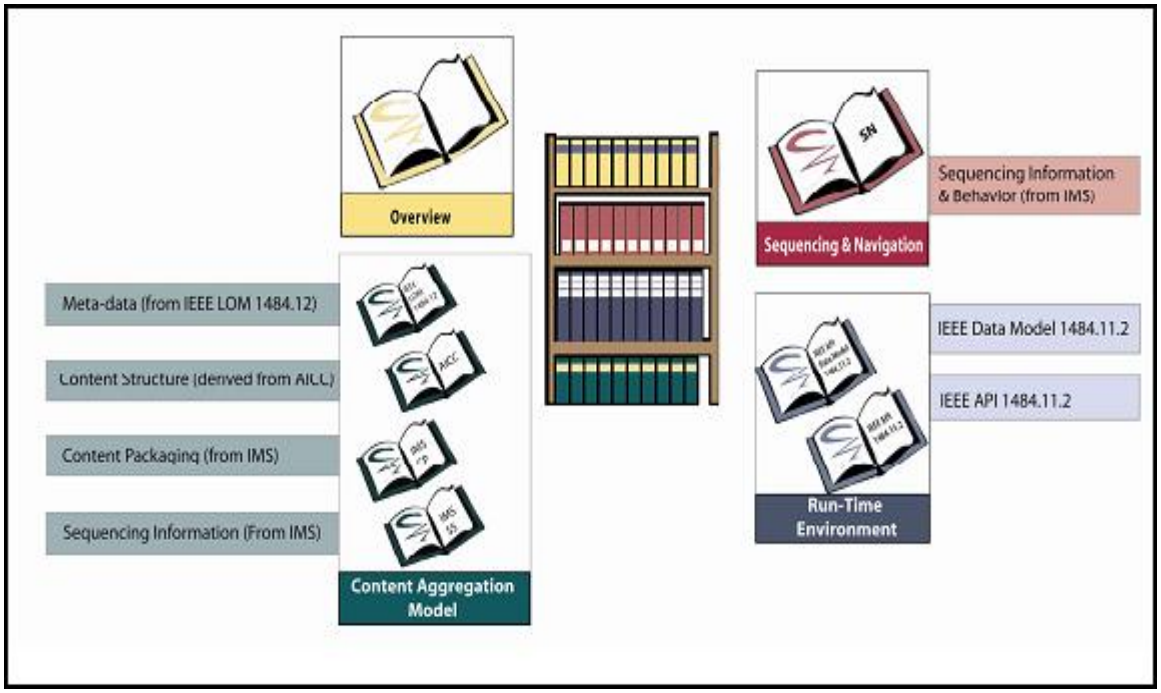
a) *Content Aggregation Model* (CAM) ou modelo de agregação do conteúdo: define como compactar o conteúdo instrucional para a troca de sistema para sistema;
b) *Run-Time Environment* (RTE) ou ambiente de execução: estabelece como o conteúdo instrucional se comunicará com o *Learning Management Systems*[1]

[1] Segundo Irlbeck e Mowat (2007, p. 163), o LMS tem como objetivo ajudar na administração das atividades relacionadas ao aprendizado. Com isso consegue-se montar cursos agrupando e organizando os Objetos de Apredizagem (OA) numa estrutura que representa a ordem em que o aluno deve acessá-los.

(LMS) e vice-versa;

c) *Sequencing and Navigation* (SN) ou sequenciamento e navegação: institui como ocorre a seleção do conteúdo instrucional baseado nas interações do usuário e na definição do CAM.

Além dos três livros técnicos, existe um quarto livro que descreve os fundamentos básicos do SCORM, proporcionando uma visão geral entre todos os livros na Figura 2.



Fonte: ADL (2006).

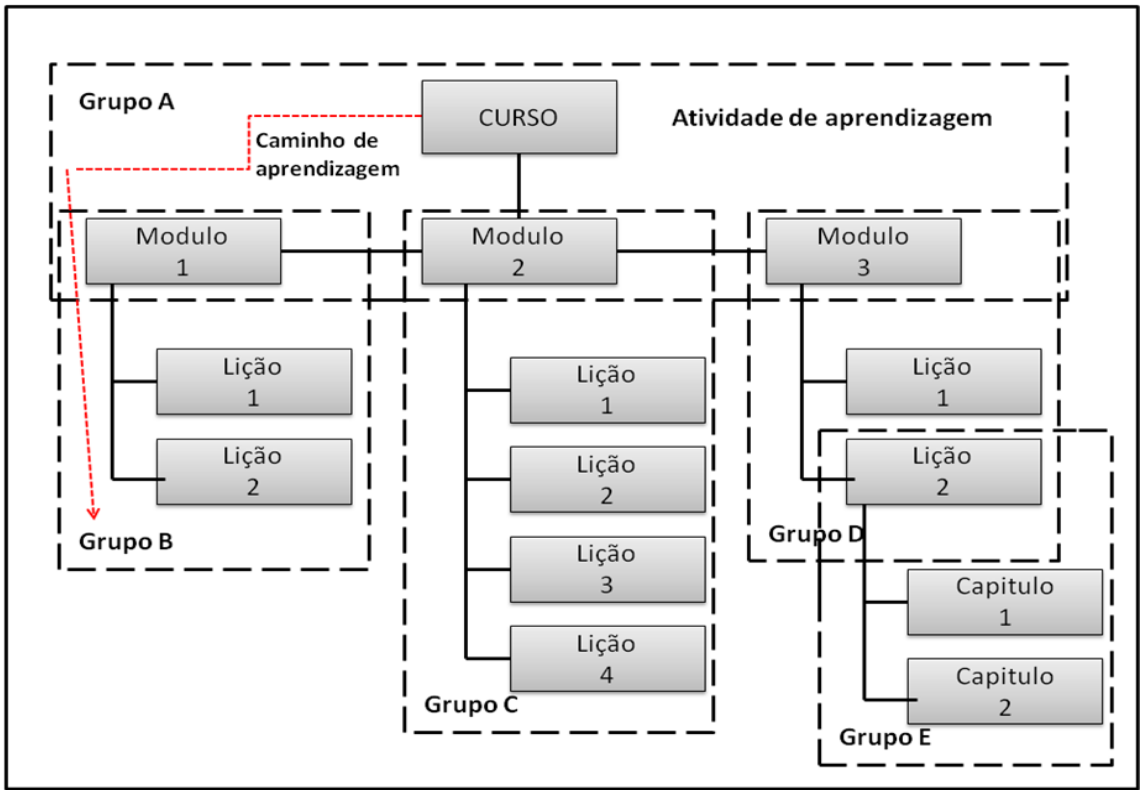
Figura 2 – Conjunto de especificações SCORM

2.2.1 Árvore de atividades, Sequenciamento e navegação

Com o SN tem-se a possibilidade de definir um conjunto de elementos, ou seja, regras que podem ser usadas para descrever e afetar diferentes comportamentos para disponibilização das atividades. Um exemplo seria que o aluno só poderia visualizar uma nova atividade somente se terminar a que estiver fazendo no momento, entre outras regras que podem estar estabelecidas.

O material de um SN é organizado em uma estrutura hierárquica, ou seja, uma árvore de atividades que possui um aprendizado mapeado. Na Figura 3 pode-se visualizar uma árvore de atividades, onde cada aprendizagem inclui uma ou mais atividades, sendo assim possui um conjunto de comportamentos associados sequencialmente, definida pelo

Sequencing Definition Model (SDM), que é um conjunto de atributos utilizados pelo SN (SU et al., 2005).



Fonte: Su et al. (2005, p. 02).

Figura 3– Exemplo de árvore de atividade com grupos

O SN utiliza a informação para atribuir um conjunto específico de dados a atributos que estarão associados na aprendizagem da árvore de atividades, seguindo o comportamento desejado, para controlar a sequenciação, seleção e entrega de atividades.

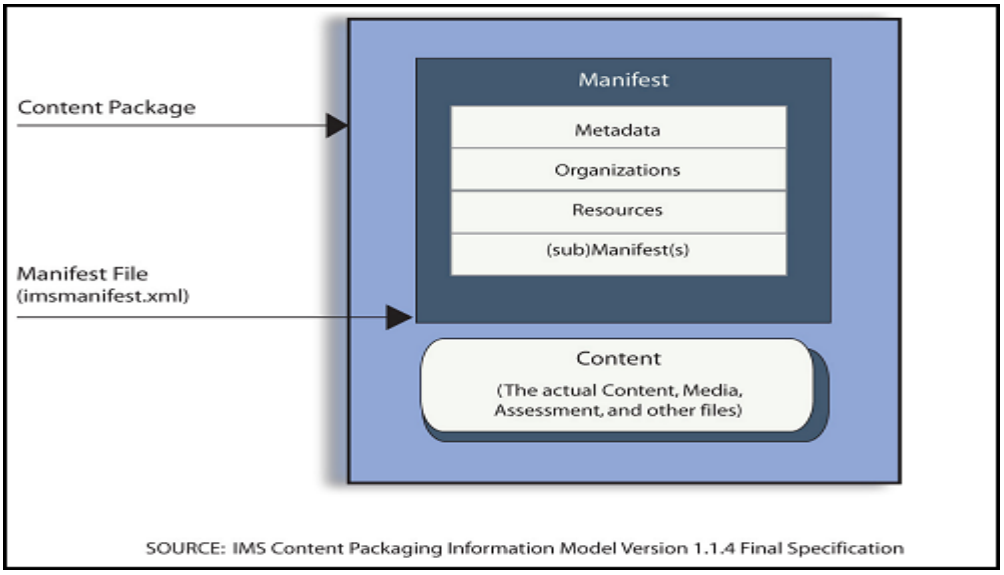
Conforme Figura 3, cada grupo possui um SDM para definir um conjunto de elementos que podem ser usados para descrever e afetar diferentes comportamentos da árvore de atividades de acordo com o sequenciamento imposto. O SDM inclui 4 definições (SU et al., 2005, p. 03):

a) *Sequencing Control Mode* (SCM) ou modo de controle de sequenciamento: afeta a forma como os pedidos de navegação são aplicados a um grupo e como as atividades do grupo são consideradas durante o processamento de pedidos de sequenciamento;
b) *Sequencing Rule* (SR) ou regras de sequenciamento: definem a forma de avaliar durante vários comportamentos o sequenciamento;
c) *Rollup Rule* (RR) ou regras de retropropagação: definem a forma de executar a ação de avaliar as informações de acompanhamento das atividades;
d) *Objective* (O) ou objetivo: define como objetivo avaliar o progresso de uma

atividade informada.

2.2.2 Empacotamento

O pacote SCORM é formado por um documento XML, chamado de manifesto, que descreve os conteúdos e a organização do bloco, e pelos arquivos físicos que constituem os recursos educacionais. Com o objetivo de facilitar a distribuição pela Web, todos esses componentes são armazenados em arquivos do tipo PIF (*Package Interchange File*) dividido em duas áreas, o manifesto e os arquivos físicos. Este pacote pode ter diversos formatos, sendo os mais comuns zip, jar, rar, arj, tar e cab (ADL, 2009). Na Figura 4 se faz a ilustração dos componentes de um pacote de conteúdo.



Fonte: ADL. (2009).

Figura 4– Diagrama conceitual de um pacote

Todo pacote SCORM deve conter em sua raiz um arquivo manifesto, e que este se chame obrigatoriamente de imsmanifest.xml. Caso não tenha o nome correto não será possível desempacotá-lo. Este arquivo, por sua vez, possui quatro seções (ADL, 2009):

a) Metadados: são os dados que descrevem o conteúdo do pacote como um todo;
b) Organizações: contém a estrutura, o conteúdo ou a organização dos recursos de aprendizagem tornando-se uma unidade independente ou unidades de ensino;
c) Recursos: define os recursos de aprendizagem incluídos no pacote de conteúdo, ou seja, contém referências dos artefatos que estão armazenados no pacote;
d) Submanifestos: é opcional. Contêm manifestos subordinados quando há um aninhamento de pacotes.

Além de possuir o manifesto, também estão presentes no pacote todos os demais artefatos digitais que o compõem.

2.3 GWT

O GWT é um *framework* de desenvolvimento web da Google cujo objetivo é facilitar o desenvolvimento de novas aplicações. Possui um conjunto de ferramentas, API e componentes visuais que facilitam a construção rápida de interfaces modernas, ricas e dinâmicas (GOOGLE, 2012).

Foi criado no intuito de ajudar os desenvolvedores a perder menos tempo na hora da criação da interface para os navegadores. Devido aos *browsers* não possuírem um padrão de utilização do JavaScript, os desenvolvedores podem chegar a gastar 90% do seu tempo trabalhando em particularidade de cada *browser* (GOOGLE, 2012).

Com o GWT é possível escrever o código Asynchronous Javascript And XML (AJAX) em linguagem de programação Java. Em seguida o GWT compila o código fonte Java para o equivalente em JavaScript, que funciona em todos principais navegadores do mercado (GOOGLE, 2012).

2.4 SMARTGWT

Smartgwt é uma poderosa biblioteca de *widgets*[2] que fornece *widgets* ricos, como vários tipos de grades, paginação e filtragem, árvores com suporte para arrastar e soltar, caixas de seleção altamente personalizáveis, painéis, menus e barras de ferramentas, diálogos e formulários (Smartgwt, 2011). O Smartgwt é uma biblioteca que extende os recursos da API GWT, principalmente relacionada a facilitar a construção de interfaces entre outros.

---

[2] *Widgets* são componentes de interface gráfica com o usuário. Podem representar textos, imagens ou pequenas janelas que ficam no *desktop*. Seu objetivo principal é entreter, ajudar ou informar o usuário (SPINER, 2010).

2.5 TRABALHOS CORRELATOS

Ressener (1997) desenvolveu um software para especificar e executar visualmente redes de Petri. O protótipo foi implementado no ambiente de programação Borland Delphi 2.0 para a plataforma Windows 95. Para realizar a execução da rede de Petri foram carregadas as matrizes de entrada e saída e um vetor de marcação. A matriz de entrada é formada por lugares (colunas) e por transições (linhas). O mesmo formato é utilizado pela matriz de saída. Assim, tendo as coordenadas do lugar e da transição, obtém o peso do arco em questão.

Giordani (1997) desenvolveu a continuação do software elaborado por Ressener (1997). Foi acrescentada a capacidade de análise da rede utilizando a técnica da árvore de alcançabilidade para obtenção de algumas propriedades da mesma. O método árvore de alcançabilidade cria uma estrutura em árvore das marcações alcançáveis a partir de uma marcação inicial M0, sendo os nós as marcações e os arcos as transições. A árvore gera um conjunto de alcançabilidade cujos elementos são todos nós da árvore. Os elementos deste conjunto são todos os estados possíveis que a rede pode alcançar.

Lin et al. (2005) comentam sobre um sistema denominado *MINA Authoring Tool*, o qual baseia-se em um sistema de criação de conteúdo de objetos. O sistema contém uma interface amigável no qual os usuários podem simplesmente arrastar e soltar para gerar um pacote SCORM de sequência e navegação. Porém, as regras de sequência são demasiadamente complexas para serem rastreadas. Para mostrar o resultado, o sistema gera uma nova janela para mostrar o objeto dentro do *Distance learning Color Petri Net* [3](DCPN), conforme as regras cadastradas.

Su et al. (2005) destacam uma ferramenta web chamada de *Visualized Online Simple Sequencing Authoring Tool* (VOSSAT), que fornece uma interface de fácil utilização para edição de pacotes SCORM. O próprio usuário pode colocar várias regras de sequenciamento clicando apenas nos ComboBox de regras de sequenciamento.

[3] Nome da aplicação destacado por Lin et al. (2005).

3 DESENVOLVIMENTO

Neste capitulo são abordados os requisitos, a especificação, a implementação e a operacionalidade da ferramenta, alem dos resultados e discussões.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Para alcançar os objetivos de utilização da ferramenta para criação de curso de conteúdo educacional, foram identificados os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF). No Quadro 1 pode ser observados os RF, no Quadro 2 os RNF e no Quadro 3 as Regras de negócio (RN).

Requisitos Funcionais (RF)	**Caso de Uso (UC)**
RF01: o sistema deve permitir criar atividades	UC01
RF02: o sistema deve permitir criar transição entre as atividades	UC05
RF03: o sistema deve permitir criar objetivos para cada atividade	UC02
RF04: o sistema deve permitir criar ações para cada objetivo	UC03
RF05: o sistema deve permitir inclusão de arquivos para cada ação	UC04
RF06: o sistema deve permitir a criação de grupos de atividades	UC01
RF07: o sistema deve permitir o armazenamento das informações do curso em banco de dados	UC01
RF08: o sistema deve permitir editar a rede de Petri	UC01
RF08: o sistema deve permitir navegação entre as atividades	UC06
RF10: o sistema deve permitir extrair os arquivos de um pacote	UC07
RF11: o sistema deve permitir criar um pacote de arquivos	UC07

Quadro 1 – Requisitos funcionais

Requisitos Não Funcionais (RNF)
RNF01: será implementada utilizando a linguagem de programação JAVA EE 6.0
RNF02: será implementada com a API SWARTGWT
RNF03: utilizar o banco de dados MySQL
RNF04: a área de edição será implementada com Applets

Quadro 2 – Requisitos não-funcionais

Regra de Negócio (RN)
RN01: ter SN para uma rede de Petri
RN02: será exibida a árvore de atividades da rede de Petri
RN03: a edição das atividades ocorre somente na rede de Petri

Quadro 3 – Regras de negócio

3.2 ESPECIFICAÇÃO

A especificação do trabalho foi modelada através da ferramenta Entreprise Architect. Utilizando o conceito de orientação a objetos, através da Unified Modeling Language (UML) para a criação dos diagramas de casos de uso, classe e de sequência, além do modelo de entidade e relacionamento (MER), para verificação da modelagem do banco.

3.2.1 Diagrama de casos de uso

Nesta sessão de são descritos os casos de uso do editor, que possui sete: Criar e editar objeto de aprendizagem, criar e editar objetivo, criar e editar ação, criar e editar material, criar e editar transição, efetuar sequenciamento e navegação, e `download` e `upload`, como pode ser observado na Figura 5.

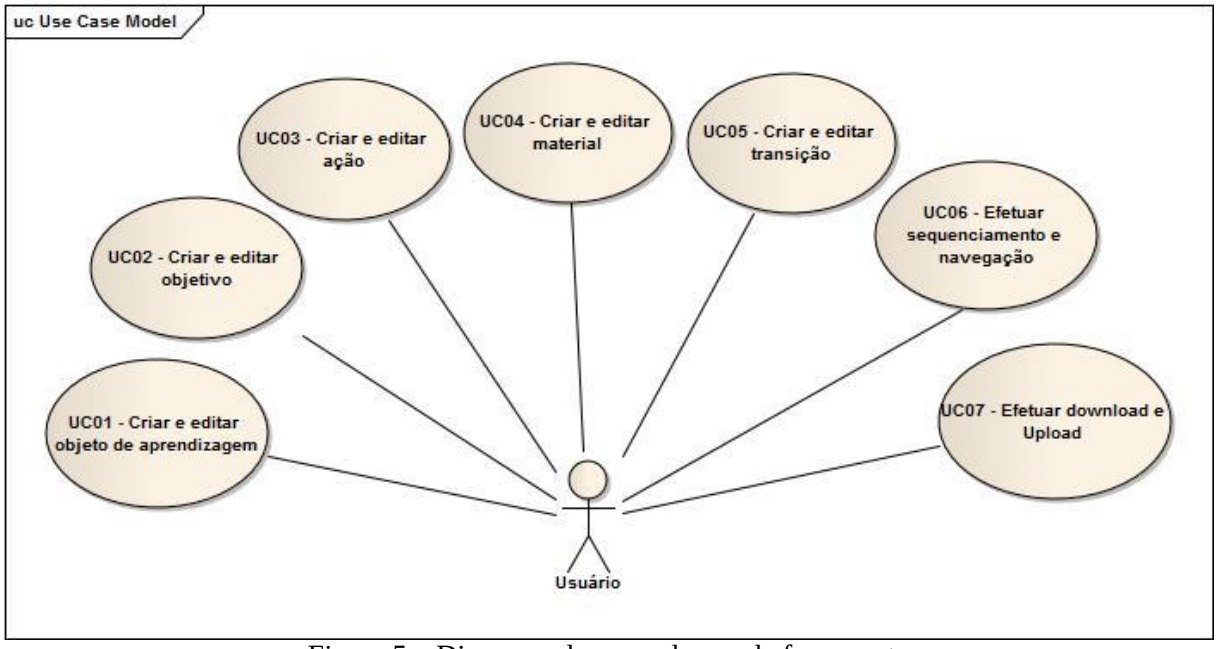


Figura 5 – Diagrama de casos de uso da ferramenta

O primeiro caso de uso (Quadro 4), designado `Criar e editar objeto de aprendizagem`, é como o usuário deve efetuar a criação do curso assim inserindo os objetos de aprendizagem.

UC01 – Criar e editar objeto de aprendizagem: permite a criação de atividades utilizando redes de Petrí	
Pré-condições	Clicar no botão novo
Cenário principal	01) O usuário clica no botão novo. 02) É efetua solicitação da descrição do novo projeto 03) O usuário preenche os dados e confirma. 04) O usuário clica no botão adiciona objeto de aprendizagem, assim efetua solicitação da descrição do objeto e confirma. 05) O usuário clica no botão selecionar objeto para posicionar o objeto de aprendizagem na posição desejada. 06) Para incluir novos objetos de aprendizagem retorna para o passo 04.
Fluxo alternativo 01	No passo 05, o usuário clica no botão selecionar objeto, escolhe objeto de aprendizagem, em seguida clica no botão direito do mouse sobre o objeto de aprendizagem, escolhe opção 1 (Alterar titulo): 05.01) O usuário informa novo titulo e confirma
Fluxo alternativo 02	No passo 05, o usuário clica no botão selecionar objeto, escolhe objeto de aprendizagem, em seguida clica no botão direito do mouse sobre o objeto de aprendizagem, escolhe opção 2 (Deletar objeto): 05.01) O usuário clica no botão ok.
	No passo 05, o usuário clica no botão selecionar objeto, escolhe objeto de aprendizagem, em seguida clica no botão direito do mouse sobre o objeto de aprendizagem, escolhe opção 3 (Definir lição final): 05.01) O usuário clica no botão ok.
Pós-condições	O curso é armazenado no banco.

Quadro 4 – Detalhamento do caso de uso `Criar e editar objeto de aprendizagem`

O segundo caso de uso (Quadro 5), denominado `criar e editar objetivo`, descreve os passos necessários para efetuar a inserção de objetivos para cada objeto de aprendizagem.

UC02 – Criar e editar objetivo: permite incluir e editar objetivos.	
Pré-condições	Ter um objeto de aprendizagem selecionado.
Cenário principal	01) O usuário clica no botão direito do mouse sobre o objeto de aprendizagem, escolhe opção 4 (Ver \ Objetivos). 02) Abre janela para inserção de objetivos. 03) O usuário deve clicar no botão novo objetivo. 04) Abre janela para preenchimento do objetivo, o usuário deve informar o titulo, descrição do objetivo e confirmar. 05) Para incluir novo objetivo retorna ao passo 01.
Fluxo alternativo 01	No passo 02, usuário deve efetuar duplo clique sobre objetivo em seguida escolher opção 1 (Alterar objetivo). 02.01) Abre janela para edição do objetivo, o usuário deve alterar o titulo, ou a descrição do objetivo e confirmar.
Fluxo alternativo 02	No passo 02, usuário deve selecionar o objetivo em seguida clicar no botão remove objetivo.
Pós-condições	Nenhuma

Quadro 1 – Detalhamento do caso de uso `Criar e editar objetivo`

O terceiro caso de uso (Quadro 6), denominado `criar e editar ação`, descreve os passos necessários para efetuar a inserção de ações para cada objetivo.

UC03 – Criar e editar ação: permite incluir e editar ação.	
Pré-condições	Ter um objetivo.
Cenário principal	01) O usuário deve efetuar duplo clique sobre objetivo em seguida escolher opção 2 (Inserir ação). 02) Abre janela de ações do objetivo. 03) O usuário deve clicar no botão nova ação. 04) Abre janela para preenchimento da ação, o usuário deve informar o titulo, descrição da ação e confirmar. 05) Para incluir nova ação retorna ao passo 01.
Fluxo alternativo 01	No passo 02, usuário deve efetuar duplo clique sobre ação em seguida escolher opção 1 (Alterar ação). 02.01) Abre janela para edição da ação, o usuário deve alterar o titulo, ou a descrição da ação e confirmar.
Fluxo alternativo 02	No passo 02, usuário deve selecionar a ação em seguida clicar no botão remove ação.
Pós-condições	Nenhuma

Quadro 6 – Detalhamento do caso de uso `Criar e editar ação`

O quarto caso de uso (Quadro 7), denominado `criar e editar material`, descreve os passos necessários para efetuar a inserção de materiais para cada ação.

UC04 – Criar e editar material: permite incluir e editar material.	
Pré-condições	Ter uma ação.
Cenário principal	01) O usuário deve efetuar duplo clique sobre ação em seguida escolher opção 2 (Inserir materiais). 02) Abre janela de materiais anexos da ação. 03) O usuário deve clicar no botão novo material. 04) Abre janela para encontrar o material a ser adicionado, o usuário deve encontrar o arquivo e salvar. 05) Para incluir novo material retorna ao passo 01.
Fluxo alternativo 01	No passo 02, usuário deve selecionar o material em seguida clicar no botão remove material.
Pós-condições	Nenhuma

Quadro 7 – Detalhamento do caso de uso `Criar e editar material`

O quinto caso de uso (Quadro 8), denominado `criar e editar transição`, descreve os passos necessários para efetuar a inserção de transições entre os objetos de aprendizagem.

UC05 – Criar e editar transição: permite incluir e editar material.	
Pré-condições	Clicar no botão adiciona objeto transição.
Cenário principal	01) O usuário clica no botão adiciona objeto transição. 02) O usuário deve posicionar o curso no editor e clicar. 03) É efetua solicitação da descrição da transição. 04) O usuário preenche os dados e confirma. 05) O usuário clica no botão selecionar objeto para posicionar a transição na posição desejada. 06) Para incluir nova transição retorna ao passo 01.
Fluxo alternativo 01	No passo 05, o usuário clica no botão selecionar objeto, escolhe objeto de transição, em seguida clica no botão direito do mouse sobre o objeto de transição, escolhe opção 1 (Alterar titulo): 05.01) O usuário informa novo titulo e confirma.
Fluxo alternativo 02	No passo 05, o usuário clica no botão selecionar objeto, escolhe objeto de transição, em seguida clica no botão direito do mouse sobre o objeto de transição, escolhe opção 2 (Deletar objeto): 05.01) O usuário clica no botão ok.
Pós-condições	Nenhuma

Quadro 8 – Detalhamento do caso de uso `Criar e editar transição`

O sexto caso de uso (Quadro 9), denominado `efetuar sequenciamento e navegação`, descreve os passos necessários para que através do sequenciamento, seja efetua a navegação do curso. O sequenciamento é definido pela inclusão dos itens na tela conforme os casos de uso anteriores, assim criando uma sequencia para navegação entre os objetos de aprendizagem.

UC06 – Efetuar sequenciamento e navegação: permite a navegação pelos objetos de aprendizagem.	
Pré-condições	Ter um curso montado.
Cenário principal	01) O usuário clica no botão iniciar navegação. 02) O editor posiciona no objeto de aprendizagem inicial. 03) O usuário clica no botão seguinte para prosseguir na navegação, abrindo respectivamente objetivos, ações e materiais, a cada clique. 04) O editor posiciona no próximo objeto de aprendizagem, efetuando passo 3 novamente. 05) Ao termino do curso o editor informa que a navegação foi encerrada. 06) Retorna ao passo 01.
Pós-condições	Nenhuma

Quadro 9 – Detalhamento do caso de uso efetuar sequenciamento e navegação

O setimo caso de uso (Quadro 10), designado `efetuar download e upload`, descreve o processo para desempacotar um curso com seus respectivos materiais e `upload` para empacotar um curso e seus respectivos materiais.

UC07 – Download e upload: permite compactar e descompactar cursos	
Pré-condições	Ter um curso montado para compactar ou arquivo para descompactar.
Cenário principal	01) O usuário clica no botão upload. 02) Abre janela de upload. 03) O usuário deve passar o caminho e o nome do local do arquivo. 04) A ferramenta pega o XML gerado pelo editor e os arquivos anexos aos objetos de aprendizagem e efetua o empacotamento. 05) Retorna ao passo 01.
Fluxo alternativo 01	No passo 01, o usuário opta por clicar no botão download. 01.01) O usuário deve clicar no botão procurar, assim encontrar o arquivo e clicar no botão abrir. 01.02) O usuário deve passar o caminho e o nome do local onde descompactar o arquivo. 01.03) A ferramenta pega o arquivo e desempacota no local escolhido pelo usuário e apresenta as informações conforme XML, apresentando as no editor. 01.04) O usuário efetua as alterações necessárias no curso. 01.05) Retorna ao passo 1 do cenário principal.
Pós-condições	Nenhuma

Quadro 10 – Detalhamento do caso de uso `download` e `upload`

3.2.2 Diagrama de classes

O diagrama de classes permite uma visão de como as classes estão estruturadas e relacionadas. De forma a facilitar a estruturação e a relação entre elas. Com a ferramenta dividida entre dois projetos, um projeto `applet` e outro utilizando o `framework` Smartgwt para efetuar a interação entre cliente / servidor, são apresentados os seguintes diagramas de classe.

3.2.2.1 Projeto `applet`

O primeiro diagrama, na Figura 06, fornece a visão da construção inicial de um `applet`. Classe `NewJFrame` apenas existe para execução do método `init`, que é executado quando a applet é carregada pela primeira vez e o método `start` é executado quando o navegador carrega ou volta à pagina com `applet`, ou seja, responsável pela inicialização da área de edição da ferramenta. A classe `JanelaApplet` efetua a montagem da tela de edição como botões e seus eventos, além de possuir métodos para interagir com a edição dos cursos e com o `framework` Smartgwt.

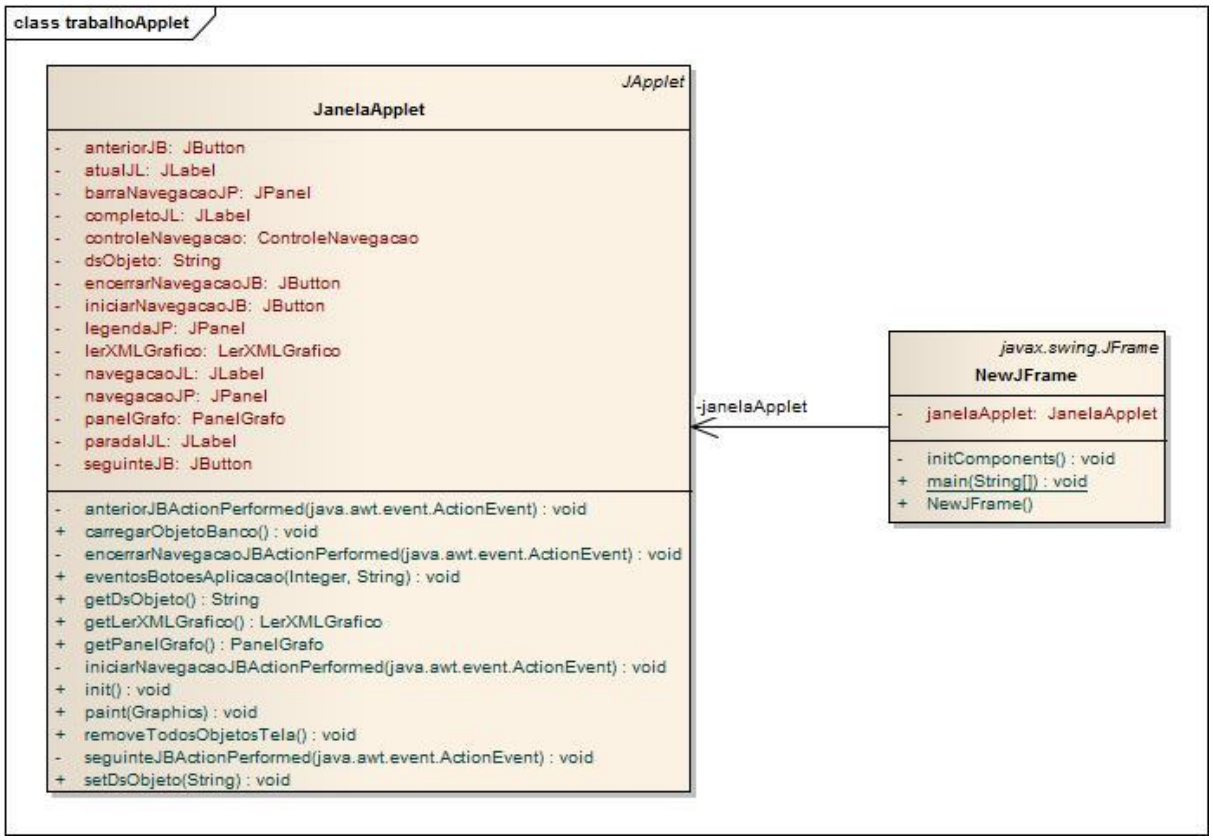


Figura 6 – Diagrama de classe do pacote trabalhoApplet

O pacote `trabalhoApplet.editor` apresentado na Figura 07, é responsável pelas classes que interagem com o editor para a montagem do curso, que vão dos objetos de aprendizagem, transições, grupos, objeto inicial e final, que o usuário efetua a montagem dos cursos que são:

a) `PanelGrafo`: classe principal do editor. Possui todas as chamadas de eventos de mouse, métodos de interação com as classes dos itens, grupos e armazenamento de informação da tela, além de possui e método `paint`, responsável pela montagem visual do elementos em tela;

b) `AcaoPanelGrafo`: classe do tipo enum, que contém somente constantes para ter controle de qual botão foi clicado pelo usuário;

c) `Item`: classe pai das classes `Noh`, `Rect`, `Arco` e `Grupo`. Possui atributos e métodos de comum utilização nas classes filhas;

d) `Noh`: classe responsável pelo armazenamento das informações de cada objeto de aprendizagem, que vão de coordenadas na tela à lista de objetivos;

e) `Rect`: classe responsável pelos elementos de transição entre os objetos de aprendizagem;

f) `Arco`: classe responsável pela ligação dos elementos objeto de aprendizagem e

transição;

g) `Grupo`: classe que contém os objetos de aprendizagem definidos em um grupo;
h) `Objetivos`: classe que contém a definição de objetivos de um objeto de aprendizagem e armazenamento de lista de ações para cada;
i) `Acoes`: classe que contém as definições das ações de um objetivo, e armazenamento de uma lista de arquivos para cada;
j) `ArquivoAnexo`: classe que contém informações relacionadas aos arquivos anexos a cada ação.

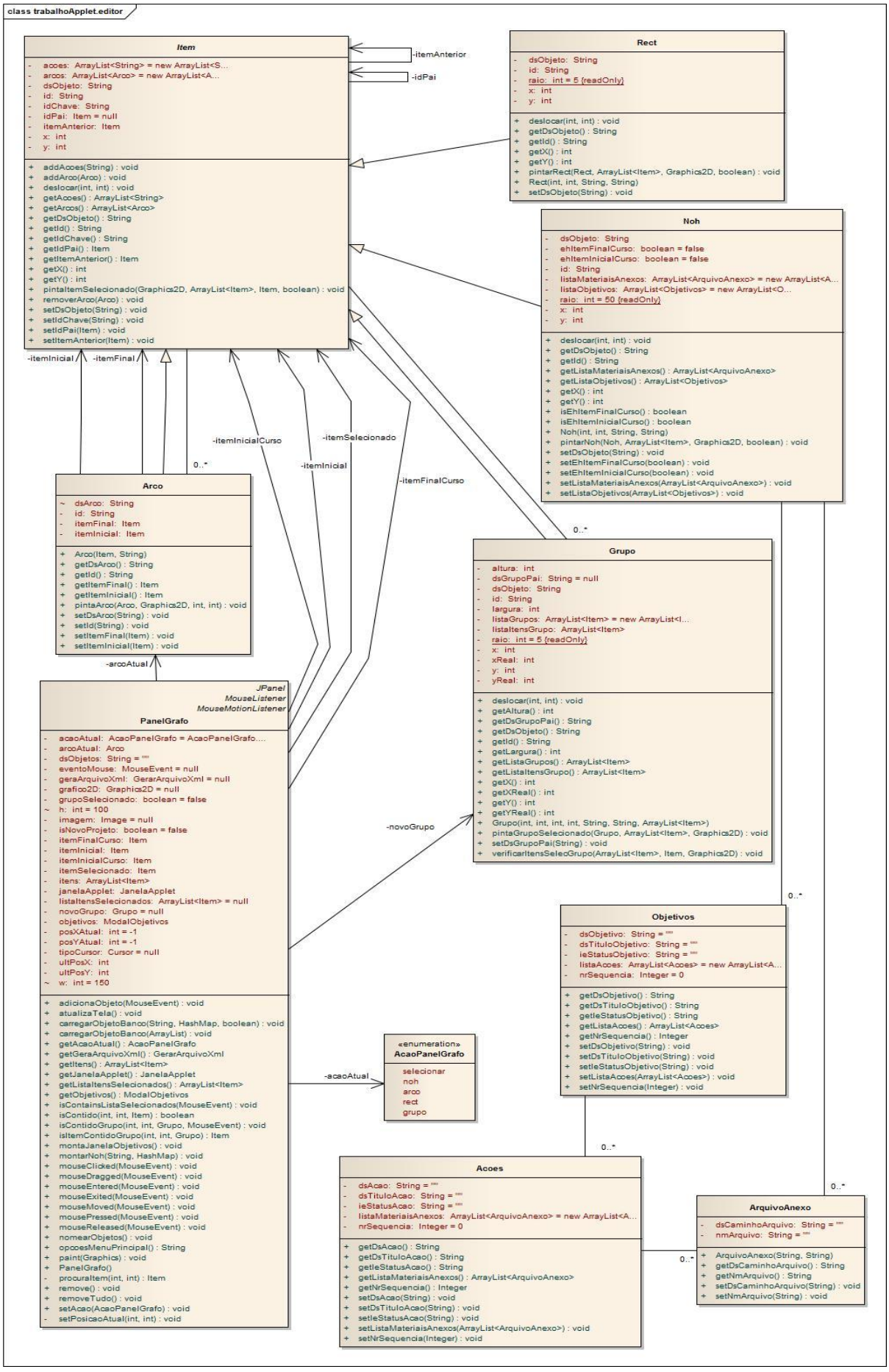

class trabalhoApplet.editor
Item
- acoes: ArrayList<String> = new ArrayList<S...
- arcos: ArrayList<Arco> = new ArrayList<A...
- dsObjeto: String
- id: String
- idChave: String
- idPai: Item = null
- itemAnterior: Item
- x: int
- y: int
+ addAcoes(String) : void
+ addArco(Arco) : void
+ deslocar(int, int) : void
+ getAcoes() : ArrayList<String>
+ getArcos() : ArrayList<Arco>
+ getDsObjeto() : String
+ getId() : String
+ getIdChave() : String
+ getIdPai() : Item
+ getItemAnterior() : Item
+ getX() : int
+ getY() : int
+ pintaItemSelecionado(Graphics2D, ArrayList<Item>, Item, boolean) : void
+ removerArco(Arco) : void
+ setDsObjeto(String) : void
+ setIdChave(String) : void
+ setIdPai(Item) : void
+ setItemAnterior(Item) : void
-itemAnterior
-idPai
-itemInicial
-itemFinal
Rect
- dsObjeto: String
- id: String
- raio: int = 5 {readOnly}
- x: int
- y: int
+ deslocar(int, int) : void
+ getDsObjeto() : String
+ getId() : String
+ getX() : int
+ getY() : int
+ pintarRect(Rect, ArrayList<Item>, Graphics2D, boolean) : void
+ Rect(int, int, String, String)
+ setDsObjeto(String) : void
Noh
- dsObjeto: String
- ehItemFinalCurso: boolean = false
- ehItemInicialCurso: boolean = false
- id: String
- listaMateriaisAnexos: ArrayList<ArquivoAnexo> = new ArrayList<A...
- listaObjetivos: ArrayList<Objetivos> = new ArrayList<O...
- raio: int = 50 {readOnly}
- x: int
- y: int
+ deslocar(int, int) : void
+ getDsObjeto() : String
+ getId() : String
+ getListaMateriaisAnexos() : ArrayList<ArquivoAnexo>
+ getListaObjetivos() : ArrayList<Objetivos>
+ getX() : int
+ getY() : int
+ isEhItemFinalCurso() : boolean
+ isEhItemInicialCurso() : boolean
+ Noh(int, int, String, String)
+ pintarNoh(Noh, ArrayList<Item>, Graphics2D, boolean) : void
+ setDsObjeto(String) : void
+ setEhItemFinalCurso(boolean) : void
+ setEhItemInicialCurso(boolean) : void
+ setListaMateriaisAnexos(ArrayList<ArquivoAnexo>) : void
+ setListaObjetivos(ArrayList<Objetivos>) : void
-itemInicialCurso
-itemSelecionado
-itemInicial
-itemFinalCurso
0..*
Arco
~ dsArco: String
- id: String
- itemFinal: Item
- itemInicial: Item
+ Arco(Item, String)
+ getDsArco() : String
+ getId() : String
+ getItemFinal() : Item
+ getItemInicial() : Item
+ pintaArco(Arco, Graphics2D, int, int) : void
+ setDsArco(String) : void
+ setId(String) : void
+ setItemFinal(Item) : void
+ setItemInicial(Item) : void
-arcoAtual
JPanel
MouseListener
MouseMotionListener
PanelGrafo
- acaoAtual: AcaoPanelGrafo = AcaoPanelGrafo....
- arcoAtual: Arco
- dsObjetos: String = ""
- eventoMouse: MouseEvent = null
- geraArquivoXml: GerarArquivoXml = null
- grafico2D: Graphics2D = null
- grupoSelecionado: boolean = false
~ h: int = 100
- imagem: Image = null
- isNovoProjeto: boolean = false
- itemFinalCurso: Item
- itemInicial: Item
- itemInicialCurso: Item
- itemSelecionado: Item
- itens: ArrayList<Item>
- janelaApplet: JanelaApplet
- listaItensSelecionados: ArrayList<Item> = null
- novoGrupo: Grupo = null
- objetivos: ModalObjetivos
- posXAtual: int = -1
- posYAtual: int = -1
- tipoCursor: Cursor = null
- ultPosX: int
- ultPosY: int
~ w: int = 150
+ adicionaObjeto(MouseEvent) : void
+ atualizaTela() : void
+ carregarObjetoBanco(String, HashMap, boolean) : void
+ carregarObjetoBanco(ArrayList) : void
+ getAcaoAtual() : AcaoPanelGrafo
+ getGeraArquivoXml() : GerarArquivoXml
+ getItens() : ArrayList<Item>
+ getJanelaApplet() : JanelaApplet
+ getListaItensSelecionados() : ArrayList<Item>
+ getObjetivos() : ModalObjetivos
+ isContainsListaSelecionados(MouseEvent) : void
+ isContido(int, int, Item) : boolean
+ isContidoGrupo(int, int, Grupo, MouseEvent) : void
+ isItemContidoGrupo(int, int, Grupo) : Item
+ montaJanelaObjetivos() : void
+ montarNoh(String, HashMap) : void
+ mouseClicked(MouseEvent) : void
+ mouseDragged(MouseEvent) : void
+ mouseEntered(MouseEvent) : void
+ mouseExited(MouseEvent) : void
+ mouseMoved(MouseEvent) : void
+ mousePressed(MouseEvent) : void
+ mouseReleased(MouseEvent) : void
+ nomearObjetos() : void
+ opcoesMenuPrincipal() : String
+ paint(Graphics) : void
+ PanelGrafo()
- procuraItem(int, int) : Item
+ remove() : void
+ removeTudo() : void
+ setAcao(AcaoPanelGrafo) : void
- setPosicaoAtual(int, int) : void
-novoGrupo
-acaoAtual
«enumeration»
AcaoPanelGrafo
selecionar
noh
arco
rect
grupo
Grupo
- altura: int
- dsGrupoPai: String = null
- dsObjeto: String
- id: String
- largura: int
- listaGrupos: ArrayList<Item> = new ArrayList<I...
- listaItensGrupo: ArrayList<Item>
- raio: int = 5 {readOnly}
- x: int
- xReal: int
- y: int
- yReal: int
+ deslocar(int, int) : void
+ getAltura() : int
+ getDsGrupoPai() : String
+ getDsObjeto() : String
+ getId() : String
+ getLargura() : int
+ getListaGrupos() : ArrayList<Item>
+ getListaItensGrupo() : ArrayList<Item>
+ getX() : int
+ getXReal() : int
+ getY() : int
+ getYReal() : int
+ Grupo(int, int, int, int, String, String, ArrayList<Item>)
+ pintaGrupoSelecionado(Grupo, ArrayList<Item>, Graphics2D) : void
+ setDsGrupoPai(String) : void
+ verificarItensSeleoGrupo(ArrayList<Item>, Item, Graphics2D) : void
Objetivos
- dsObjetivo: String = ""
- dsTituloObjetivo: String = ""
- ieStatusObjetivo: String = ""
- listaAcoes: ArrayList<Acoes> = new ArrayList<A...
- nrSequencia: Integer = 0
+ getDsObjetivo() : String
+ getDsTituloObjetivo() : String
+ getIeStatusObjetivo() : String
+ getListaAcoes() : ArrayList<Acoes>
+ getNrSequencia() : Integer
+ setDsObjetivo(String) : void
+ setDsTituloObjetivo(String) : void
+ setIeStatusObjetivo(String) : void
+ setListaAcoes(ArrayList<Acoes>) : void
+ setNrSequencia(Integer) : void
Acoes
- dsAcao: String = ""
- dsTituloAcao: String = ""
- ieStatusAcao: String = ""
- listaMateriaisAnexos: ArrayList<ArquivoAnexo> = new ArrayList<A...
- nrSequencia: Integer = 0
+ getDsAcao() : String
+ getDsTituloAcao() : String
+ getIeStatusAcao() : String
+ getListaMateriaisAnexos() : ArrayList<ArquivoAnexo>
+ getNrSequencia() : Integer
+ setDsAcao(String) : void
+ setDsTituloAcao(String) : void
+ setIeStatusAcao(String) : void
+ setListaMateriaisAnexos(ArrayList<ArquivoAnexo>) : void
+ setNrSequencia(Integer) : void
ArquivoAnexo
- dsCaminhoArquivo: String = ""
- nmArquivo: String = ""
+ ArquivoAnexo(String, String)
+ getDsCaminhoArquivo() : String
+ getNmArquivo() : String
+ setDsCaminhoArquivo(String) : void
+ setNmArquivo(String) : void

Figura 7 – Diagrama de classes do pacote trabalhoApplet.editor

O pacote `trabalhoApplet.janelas`, apresentado na Figura 09 possui todas as janelas relacionadas ao `applet`, utilizadas para acrescentar as informações dos objetos de aprendizagem que são:

a) `ModalObjetivos`: classe responsável por trazer as informações visuais de todos objetivos de um objeto de aprendizagem, além de inserir, editar e excluir os objetos de aprendizagem;
b) `ModalObjetivoFormulario`: classe responsável pela edição dos objetivos;
c) `ModalAcoes`: classe responsável por trazer as informações visuais de todos ações de um objetivo, além de inserir, editar e excluir as ações;
d) `ModalAcaoFormulario`: classe responsável para edição das ações;
e) `ModalMateriaisAnexos`: classe responsável por trazer as informações visuais de todos materiais de uma ação, além de inserir, editar e excluir novos materiais;
f) `ModalInserirMaterial`: classe responsável pela inclusão de arquivos relacionados às ações.

O pacote `trabalhoApplet.navegacao` apresentado na Figura 08 é responsável por proporcionar a navegação entre os objetos de aprendizagem. A classe `ControleNavegação` proporciona a manutenção de navegação entre os objetos de aprendizagem.

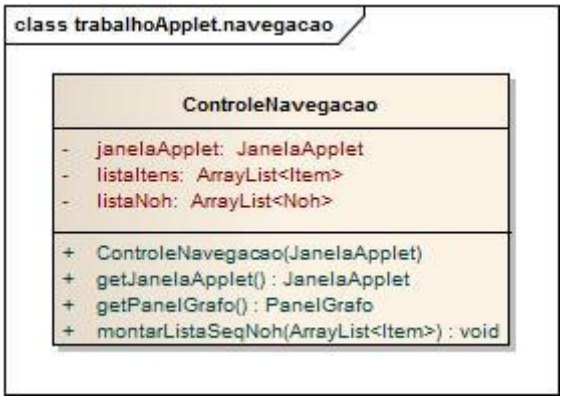


Figura 8 – Diagrama de classe do pacote trabalhoApplet.navegacao

Figura 9 – Diagrama de classes do pacote `trabalhoApplet.janelas`

O pacote `XMLArvore`, apresentado na Figura 10, contém as classes responsáveis por efetuar a manutenção entre as informações do `applet` e o `framework` Smartgwt, assim criando um arquivo XML com as informações do curso e outra para efetuar a leitura destas informações que são:

a) `GerarArquivoXml`: classe responsável por cada manifestação do usuário no editor. Cria e salva as informações em um arquivo XML para interação entre a

aplicação no seguinte caminho C:\temp\ArvoreSCORM.xml;

b) `LerXMLGrafico`: classe responsável pela leitura do XML quando estas são carregadas pelo banco de dados, garantindo que as informações sejam montadas de forma correta no arquivo de XML encontrado no caminho C:\temp\ArvoreSCORM.xml.

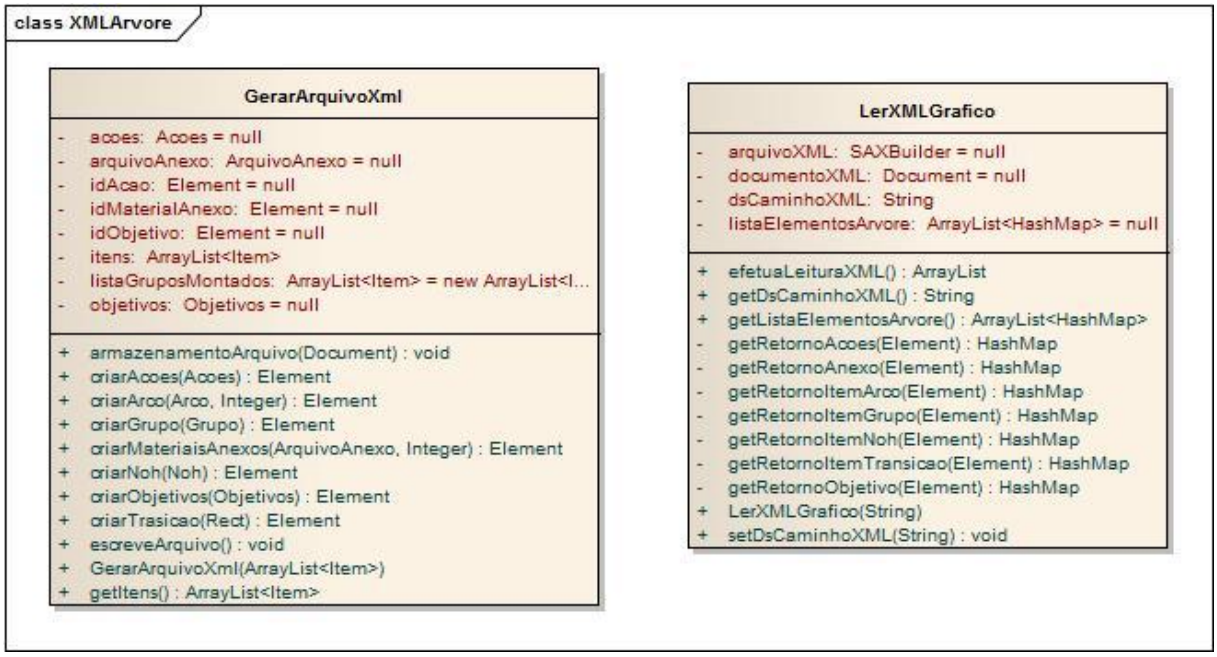


Figura 10 – Diagrama de classes do pacote `XMLArvore`

3.2.2.2 Projeto Smartgwt

Com o projeto usufruindo da utilização de cliente / servidor para transmissão das informações, dividido em dois pacotes:

a) `client` que traz informações de tela, montagem, e chamadas para o servidor;

b) `server` responsável pela regra de negócio da aplicação que traz as informações conforme as requisições feitas pelo usuário ao utilizar a ferramenta.

Partindo da definição dos componentes do lado do cliente, foram organizados em pacotes de acordo sua utilização janelas (`view`), organização (`model`) na raiz a classe principal e as classes de requisição ao servidor, vistos no pacote `org.ProjetoTCCV3.client` apresentado na Figura 11 são:

a) `MainTCC`: classe principal da ferramenta. Contém a chamada para as principais tarefas. Através dela são chamadas as classes de montagem da tela e a requisição para o servidor;

b) `ComunicacaoServiceAsync`: classe responsável para que os mesmos métodos sejam implementados no cliente e servidor;

c) `ComunicacaoService`: classe responsável para trazer o tipo de retorno das informações vindas do servidor.

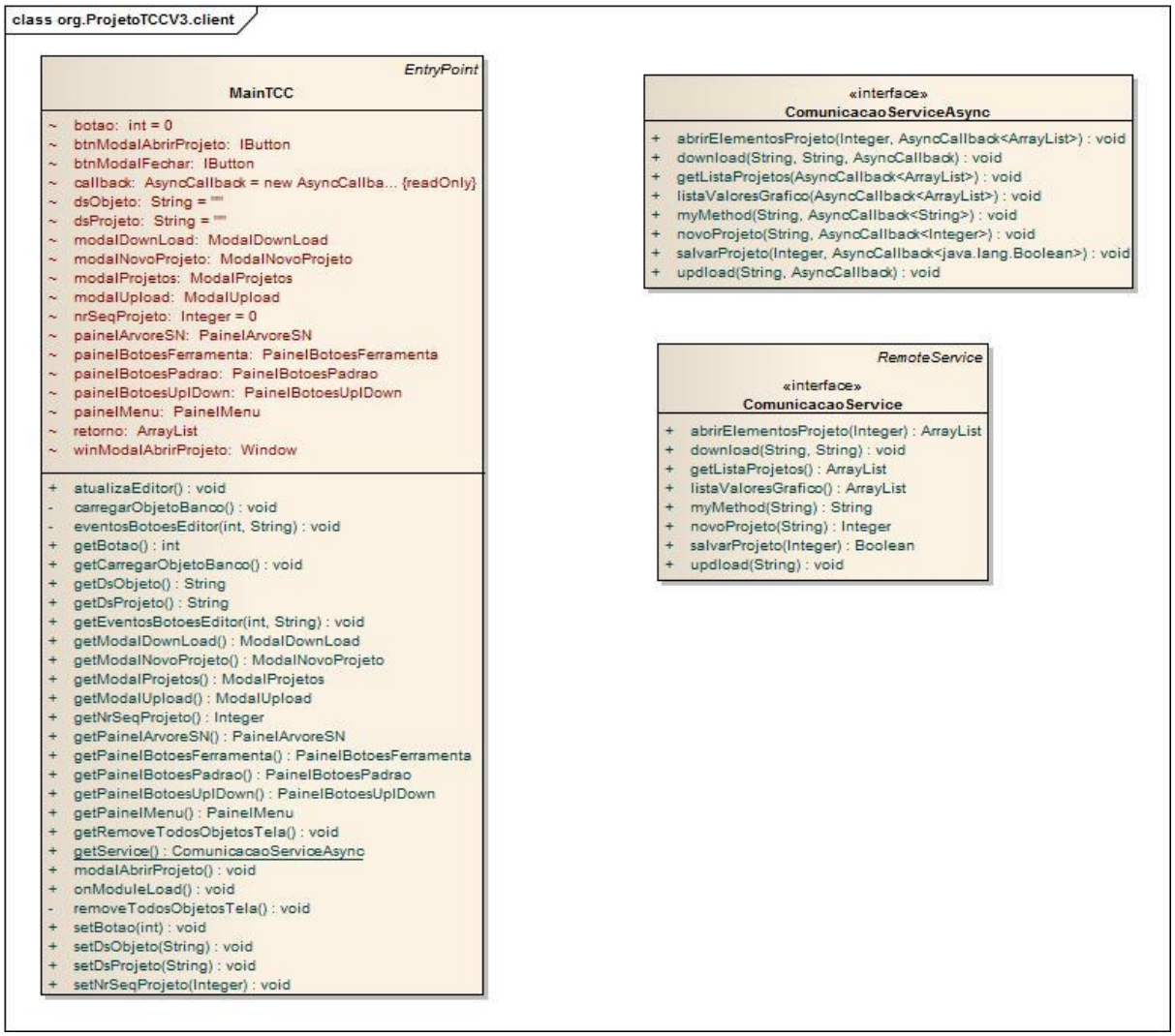


Figura 11 – Diagrama de classes do pacote `org.ProjetoTCCV3.client`

O pacote `org.ProjetoTCCV3.client.model`, apresentado na Figura 12, possui a classe para a montagem da árvore de atividades que somente tem o intuito de armazenar as informações de apresentação.

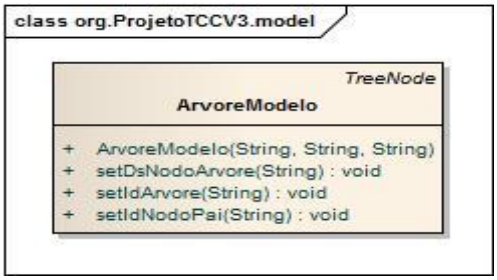


Figura 12 – Diagrama de classe do pacote `org.ProjetoTCCV3.client.model`

O pacote `org.ProjetoTCCV3.client.view` tem as classes que compõem a montagem da ferramenta como a árvore, menus, janelas e painel de botões com seus eventos interações e chamadas para o servidor, apresentados na Figura 13:

a) `PainelArvoreSN`: classe que apresenta as atividades do curso, efetua toda montagem da parte visual;
b) `PainelMenu`: classe que consiste em montar toda a estrutura visual dos menus da ferramenta e eventos de cada;
c) `PainelBotoesFerramenta`: classe que consiste em montar a estrutura visual dos botões de interação da ferramenta selecionar objeto, adiciona objeto, adiciona objeto de transição, ligação de objeto e criar grupo para construção dos cursos, além dos eventos e efetuar interação com o `applet`;
d) `PainelBotoesUplDown`: classe que consiste em montar a estrutura de visual dos botões de `upload` e `download` além dos eventos para a abertura das respectivas janelas;
e) `PainelBotoesPadrao`: classe que consiste em montar a estrutura de visual dos botões padrões novo, salvar e abrir , além de seus eventos;
f) `ModalDownLoad`: classe que consiste em montar a estrutura de visual da janela de `download`, que tem como finalidade o usuário passar um caminho de origem e um de destino para descompactação das informações de um arquivo zip;
g) `ModalUpload`: classe que consiste em montar a estrutura de visual da janela de `upload`, que tem como finalidade o usuário passar um caminho onde será criado o arquivo zip que contém os arquivos do curso;
h) `ModalProjetos`: classe que consiste em montar a estrutura de visual da janela que trará todos os cursos salvos no banco de dados numa lista, para escolha do usuário, assim carregar as informações do curso de escolha na ferramenta;
i) `ModalNovoProjeto`: classe que consiste em montar a estrutura de visual da janela para inserção de um novo curso.

Figura 13 – Diagrama de classes do pacote `org.ProjetoTCCV3.client.view`

No pacote `org.ProjetoTCCV3.server` (Figura 14), estão as classes de utilização no servidor:

a) `ComunicacaoServiceImpl` classe que contém os métodos de interação com o cliente, e que fazem a chamada para outros métodos ou classes que possuem a regras de negócio da aplicação;

b) `Dao`: classe responsável pela execução dos métodos executados no banco de dados da aplicação;

c) `GerarProjetoArquivoXml`: classe designada a montar o XML para construção dos cursos, através da escolha feita pelo usuário que curso será carregado pelas informações do banco;

d) `LerXMLGrafico`: classe responsável em efetuar a leitura do XML, montar a lista de elementos do curso, além de utilização para povoar o banco de dados ao salvar e informações para efetuar `download` e `upload` da ferramenta;

e) `Download`: classe responsável em descompactar um curso no formato zip, conforme passado pelo usuário o caminho de origem e o caminho de destino;

f) `Upload`: classe responsável em compactar um curso com seus respectivos arquivos além do arquivo `imsmanifest.xml`;

g) `GerarIMSManifest`: classe responsável pela criação do arquivo XML de manifesto `imsmanifest.xml`.

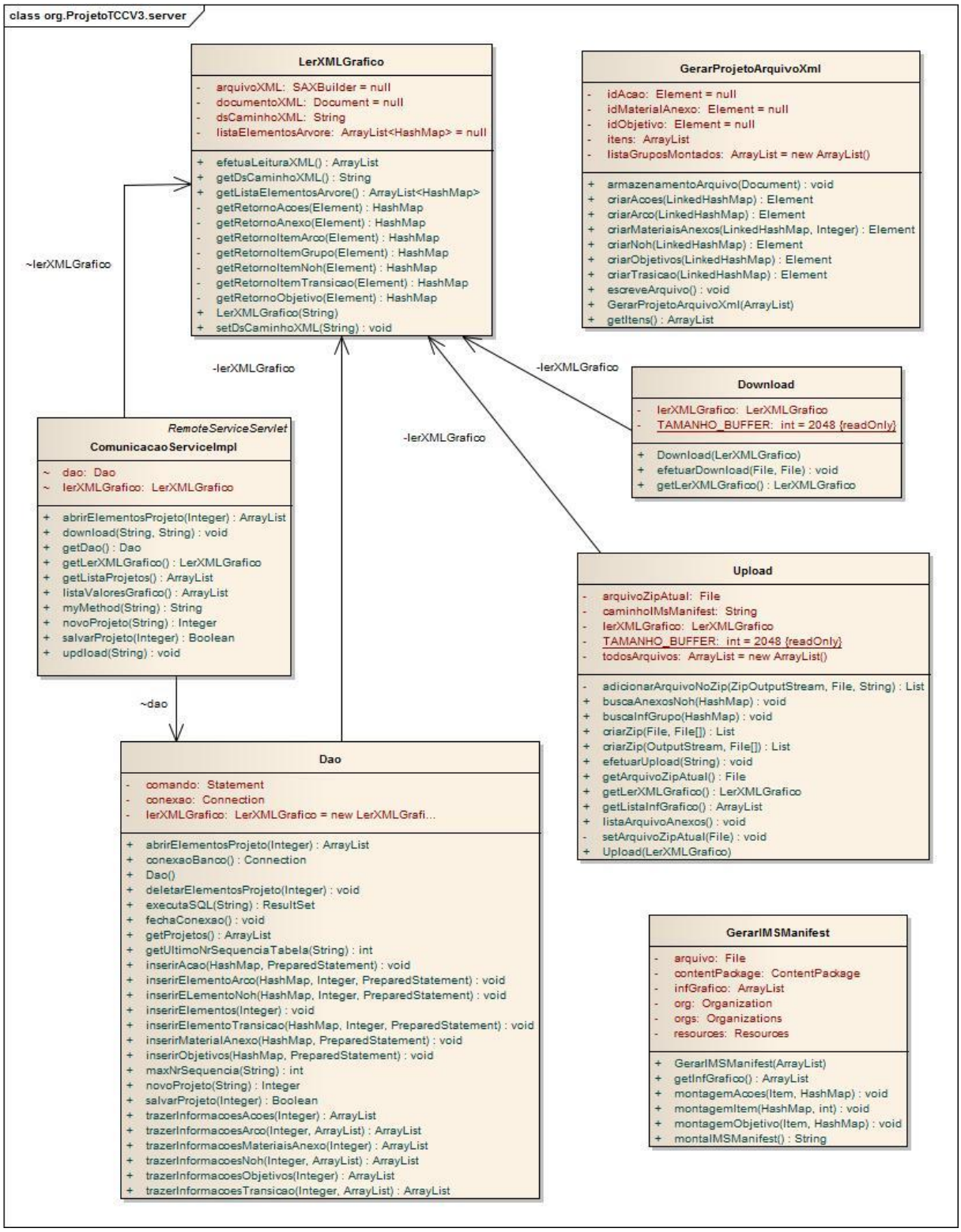


Figura 14 – Diagrama de classes do pacote `org.ProjetoTCCV3.server`

3.2.3 Diagrama de Sequência

O diagrama de sequência tem como finalidade demonstrar a troca de mensagens entre as classes criadas, com base nas ações do usuário.

A Figura 14 apresenta a sequência de chamada dos métodos pela ferramenta quando as informações carregadas no banco são expostas no editor. Este diagrama corresponde ao caso de uso `Criar e editar curso`.

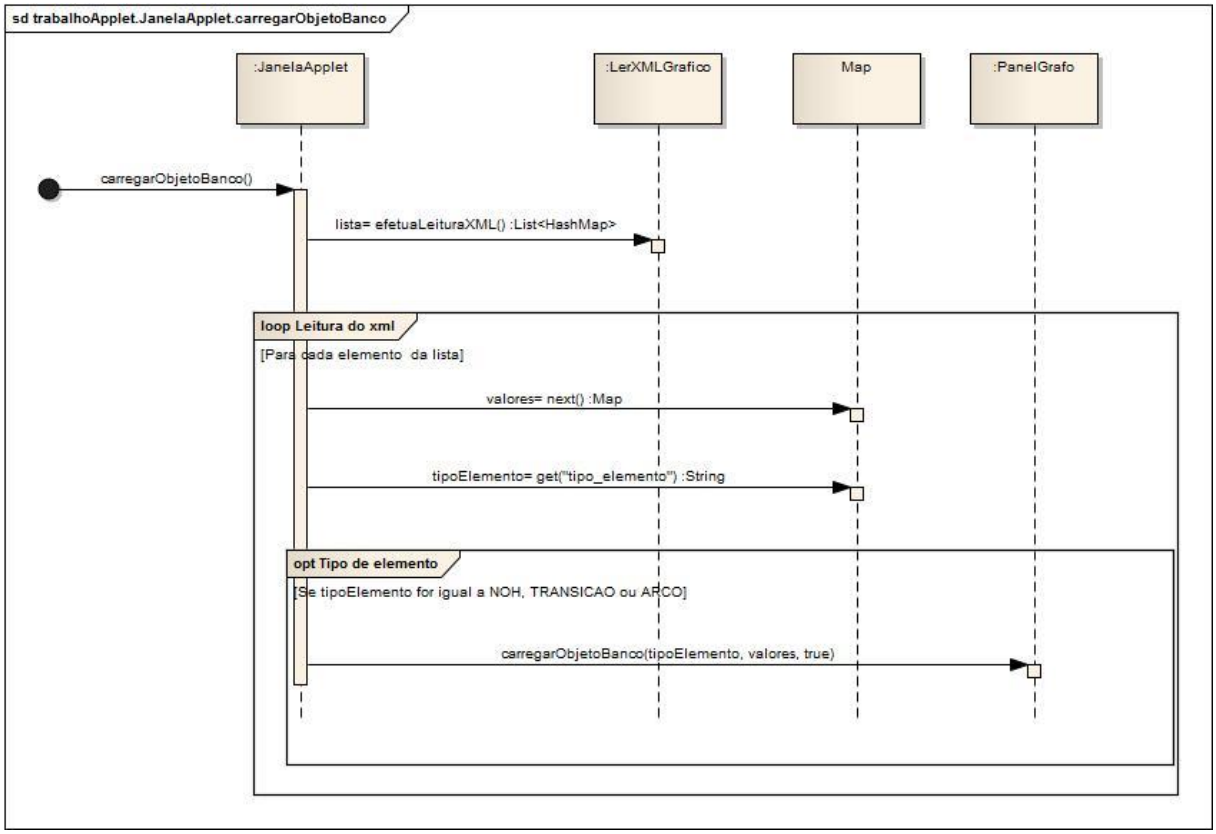


Figura 15 – Diagrama de sequência para carregamento de informações do banco

3.2.4 Banco de dados

As informações dos cursos criados podem ser armazenadas em banco de dados MySQL. A única classe que interage com as tabelas é a `Dao` que pertence ao projeto do `framework` Smartgwt. A Figura 16 apresenta o MER das tabelas do banco de dados da ferramenta.

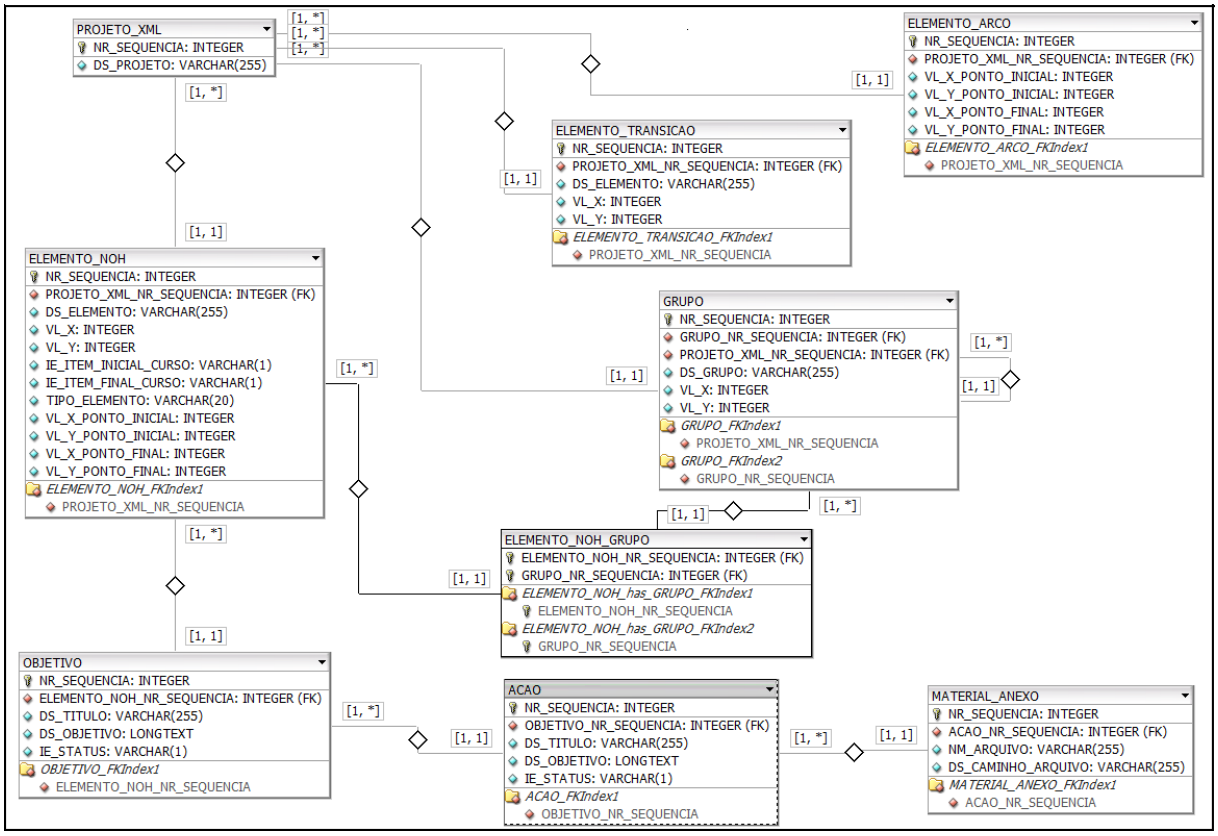


Figura 16 – MER das tabelas do banco de dados do sistema

Verificando a utilização das tabelas do banco são:

a) `Projeto_XML` tabela que armazena os dados do nome do curso;

b) `ELEMENTO_TRANSICAO` tabela que armazena os dados do elemento de transição entre os objetos de aprendizagem;

c) `ELEMENTO_ARCO` tabela que armazena os dados das ligações entre os objetos em tela, que vão de objetos de aprendizagem a transição;

d) `GRUPO` tabela responsável pelo armazenamento de vários objetos de aprendizagem no específico grupo;

e) `ELEMENTO_NOH_GRUPO` tabela responsável pelo armazenamento de dados quando um mesmo objeto de aprendizagem se encontra em mais de um grupo;

f) `ELEMENTO_NOH` tabela responsável pelo armazenamento de dados de um objeto de aprendizagem;

g) `OBJETIVO` tabela responsável pelo armazenamento de dados dos objetivos de um objeto de aprendizagem;

h) `ACAO` tabela responsável pelo armazenamento de dados das ações de um objetivo de um objeto de aprendizagem;

i) `MATERIAL_ANEXO` tabela responsável pelo armazenamento de dados dos materiais anexos de uma ação, que a ação consiste de um objetivo de um objeto de

aprendizagem.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas informações sobre as técnicas e ferramentas utilizadas para a implementação do editor, bem como a operacionalidade de implementação.

3.3.1 Técnicas e ferramentas utilizadas

O editor foi implementado na linguagem Java, seguindo o paradigma da orientação a objetos. A montagem do editor foi efetuada em 2 partes, de forma paralela. Para a codificação foi utilizado o ambiente de desenvolvimento Netbeans versão 7.1, juntamente com os `frameworks` de desenvolvimento web (Smartgwt versão 2.5 e GWT versão 2.3.0), além da utilização de `applet` para criação do editor. Já na geração do arquivo `imsmafest.xml` foi utilizado o `framework` CELINE e para armazenamento de dados dos cursos, assim utilizado o Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL.

3.3.1.1 Implementação da ferramenta

A utilização do `framework` Smartgwt se faz pela grande quantidade de componentes disponíveis para utilização em aplicações web, sendo compatível com o `framework` GWT. No qual toda a parte visual da aplicação é construída utilizando seus componentes.

O `framework` GWT trabalha com invocação remota de serviços, isso devido todas as classes dentro do diretório `client` ao serem compiladas se tornam javascript, fazendo que as regras de negócio da aplicação sejam todas implementadas no servidor.

Para efetuar a chamada de um serviço do servidor, deve-se inicialmente interface que estende a interface `RemoteService` do pacote `com.google.gwt.user.client.rpc.RemoteService` do `framework` GWT, no qual seus métodos são invocáveis remotamente. Está interface é `ComunicacaoService` representada na

Figura 17.

```
1  package org.ProjetoTCCV3.client;
2
3  import com.google.gwt.user.client.rpc.RemoteService;
4  import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;
5  import java.util.ArrayList;
6
7  /**
8   *
9   * @author mmueller
10  */
11 @RemoteServiceRelativePath("comunicacaoservice")
12 public interface ComunicacaoService extends RemoteService {
13
14     public String myMethod(String s);
15
16     public ArrayList getListaProjetos();
17
18     public ArrayList abrirElementosProjeto(Integer nrSeqProjeto);
19
20     public Integer novoProjeto(String novoProjeto);
21
22     public Boolean salvarProjeto(Integer nrSeqProjeto);
23
24     public ArrayList listaValoresGrafico();
25
26     public void download(String dsArquivoOrigem, String dsArquivoDestino);
27
28     public void updload(String dsLocalArquivo);
29 }
30
```

Figura 17 – Interface `ComunicacaoService`

Definida a interface de comunicação com o servidor, também é preciso criar uma interface que seja para efetuar a execução do serviço no cliente. A criação desta interface se faz necessária porque a invocação ao serviço utilizando-se AJAX, que para o cliente a aplicação não pára de executar.

Esta interface assíncrona possui os métodos praticamente iguais aos da interface de serviço, mas estes métodos devem ter seu retorno do tipo *void* e o ultimo parâmetro de cada deve ser do tipo `AsyncCallback` que se encontra no pacote `com.google.gwt.user.client.rpc.AsyncCallback`, que funciona como um `listener` do evento de execução do serviço. `AsyncCallback` é uma interface que ao ser invocada implementa dois métodos, `onSucess` que é chamado quando o método executa corretamente e tem como parâmetro o resultado da invocação do serviço e `onFailure`() que é executado quando a chamada ao serviço não acontece de forma correta e recebe como parâmetro um objeto do tipo `Throwable` com informações sobre o erro acontecido. O nome desta interface deve ter o mesmo nome da interface de serviço, acrescentado a seu final a palavra `Async`, que também deve ser encontrar no pacote `client` da ferramenta. Na Figura 18 é possível verificar

a estrutura da interface.

```
1  package org.ProjetoTCCV3.client;
2
3  import com.google.gwt.user.client.rpc.AsyncCallback;
4  import java.util.ArrayList;
5
6  /**
7   *
8   * @author mmueller
9   */
10 public interface ComunicacaoServiceAsync {
11
12     public void myMethod(String s, AsyncCallback<String> callback);
13
14     public void getListaProjetos(AsyncCallback<ArrayList> asyncCallback);
15
16     public void abrirElementosProjeto(Integer nrSeqProjeto, AsyncCallback<ArrayList> asyncCallback);
17
18     public void novoProjeto(String novoProjeto, AsyncCallback<Integer> asyncCallback);
19
20     public void salvarProjeto(Integer nrSeqProjeto, AsyncCallback<java.lang.Boolean> asyncCallback);
21
22     public void listaValoresGrafico(AsyncCallback<ArrayList> asyncCallback);
23
24     public void download(String dsArquivoOrigem, String dsArquivoDestino, AsyncCallback asyncCallback);
25
26     public void updload(String dsLocalArquivo,AsyncCallback asyncCallback);
27 }
28
```

Figura 18 – Classe `ComunicacaoServiceAsync`

Com as interfaces de serviço definidas, já podem ser implementadas no servidor. Criando uma classe que se estende da classe `RemoteServiceServlet` do pacote `com.google.gwt.user.server.rpc.RemoteServiceServlet` que implementa a interface `ComunicacaoService`, assim os métodos serão acrescentados a classe com seus respectivos tipos de retorno, e podendo ser acrescentados as regras e seus respectivos blocos para execução. Na Figura 19 é possível verificar a estrutura da classe.

```java
package org.ProjetoTCCV3.server;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.ZipException;
import org.ProjetoTCCV3.client.ComunicacaoService;

/**
 *
 * @author mmueller
 */
public class ComunicacaoServiceImpl extends RemoteServiceServlet implements ComunicacaoService {

    Dao dao;
    LerXMLGrafico lerXMLGrafico;

    public String myMethod(String s) {
        // Do something interesting with 's' here on the server.
        return "Server says: " + s;
    }

    public ArrayList getListaProjetos() {
        return getDao().getProjetos();
    }

    public ArrayList abrirElementosProjeto(Integer nrSeqProjeto) {
        return getDao().abrirElementosProjeto(nrSeqProjeto);
    }

    public Integer novoProjeto(String nomeProjeto) {
        return getDao().novoProjeto(nomeProjeto);
    }

    public Boolean salvarProjeto(Integer nrSeqProjeto) {
        return getDao().salvarProjeto(nrSeqProjeto);
    }

    public ArrayList listaValoresGrafico() {
        return getLerXMLGrafico().efetuaLeituraXML();
    }

    public LerXMLGrafico getLerXMLGrafico() {
        if (lerXMLGrafico == null) {
            lerXMLGrafico = new LerXMLGrafico("C:/temp/ArvoreSCORM.xml");
        }
        return lerXMLGrafico;
    }

    public Dao getDao() {
        if (dao == null) {
            dao = new Dao();
        }
        return dao;
    }

    public void download(String dsArquivoOrigem, String dsArquivoDestino) {
        try {
            Download download = new Download(getLerXMLGrafico());
            download.efetuarDownload(new File(dsArquivoOrigem), new File(dsArquivoDestino));

        } catch (ZipException ex) {
            Logger.getLogger(ComunicacaoServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(ComunicacaoServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void updload(String dsLocalArquivo) {
        try {
            Upload upload = new Upload(getLerXMLGrafico());
            upload.efetuarUpload(dsLocalArquivo);
        } catch (ZipException ex) {
            Logger.getLogger(ComunicacaoServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(ComunicacaoServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Figura 19 – Classe `ComunicacaoServiceImpl`

Para construção da parte de edição foi necessária a utilização de um `applet` que disponibilizava a utilização da biblioteca `Graphics` para criação de formas geométricas complexas no editor da ferramenta. Na Figura 20 é apresentado o método `paint` da biblioteca `Graphics` que se encontra na classe `PanelGrafo` que permite a inserção de elementos na tela.

```
55      @Override
56      public void paint(Graphics g) {
57          super.paintComponent(g);
58
59          Graphics2D g2d = (Graphics2D) g;
60
61          Dimension d = getSize();
62          if (imagem == null || imagem.getWidth(null) != d.width) {
63
64              imagem = createImage(d.width, d.height);
65              grafico2D = (Graphics2D) imagem.getGraphics();
66          }
67
68          grafico2D.setColor(new Color(142, 109, 245));//252, 254, 167
69          grafico2D.fillRect(0, 0, d.width, d.height);
70
71          for (int i = 0; i < itens.size(); i++) {
72
73              Item item = itens.get(i);
74
75              if (item.getClass() == Noh.class) {
76
77                  Noh noh = (Noh) item;
78                  noh.pintarNoh(noh, listaItensSelecionados, grafico2D, false);
79
80              } else if (item.getClass() == Rect.class) {
81
82                  Rect rect = (Rect) item;
83                  rect.pintarRect(rect, listaItensSelecionados, grafico2D, false);
84
85              } else if (item.getClass() == Grupo.class) {
86
87                  Grupo grupo = (Grupo) item;
88                  grupo.pintaGrupoSelecionado(grupo, listaItensSelecionados, grafico2D);
89
90              } else if (item.getClass() == Arco.class) {
91
92                  Arco arco = (Arco) item;
93                  arco.pintaArco(arco, grafico2D, posXAtual, posYAtual);
94              }
95          }
96
97          g2d.drawImage(imagem, 0, 0, null);
98
99          if (tipoCursor != null) {
100             this.setCursor(tipoCursor);
101         }
102
103         if (getItens() != null
104                 && !getItens().isEmpty()
105                 && !isNovoProjeto) {
106             getGeraArquivoXml().escreveArquivo();
107         }
108
109         isNovoProjeto = false;
        }
```

Figura 20 – Método `paint` da classe `PanelGrafo`

Para efetuar o compartilhamento das informações entre a área de edição feita em

`applet` e o restante da ferramenta que utiliza o `framework` Smartgwt, para salvar as informações do curso e para exposição das atividades na árvore. Portanto foi criada a classe `GerarArquivoXML`, já explicada na seção 3.2.2.1, e apresentada na figura 21.

```
package XMLArvore;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;
import trabalhoApplet.editor.Acoes;
import trabalhoApplet.editor.Arco;
import trabalhoApplet.editor.ArquivoAnexo;
import trabalhoApplet.editor.Grupo;
import trabalhoApplet.editor.Item;
import trabalhoApplet.editor.Noh;
import trabalhoApplet.editor.Objetivos;
import trabalhoApplet.editor.Rect;

/**
 *
 * @author mmueller
 */
public class GerarArquivoXml {

    private ArrayList<Item> itens;
    private ArrayList<Item> listaGruposMontados = new ArrayList<Item>();
    private Element idObjetivo = null;
    private Element idAcao = null;
    private Element idMaterialAnexo = null;
    private Objetivos objetivos = null;
    private Acoes acoes = null;
    private ArquivoAnexo arquivoAnexo = null;

    public GerarArquivoXml(ArrayList<Item> itens) {
        this.itens = itens;
    }

    public void escreveArquivo() {
        try {
            listaGruposMontados.clear();
            Element arvore = new Element("ARVORE");
            Document documento = new Document(arvore);//Define a arvore como root do arquivo xml



            armazenamentoArquivo(documento);

            if (!getItens().isEmpty()) {
                for (int i = 0; i < getItens().size(); i++) {
                    Item item = getItens().get(i);

                    if (item.getClass() == Noh.class) {
                        arvore.addContent(criarNoh((Noh) item));
                    } else if (item.getClass() == Rect.class) {
                        arvore.addContent(criarTrasicao((Rect) item));
                    } else if (item.getClass() == Grupo.class
                            && ((Grupo) item).getDsGrupoPai() == null) {
                        arvore.addContent(criarGrupo((Grupo) item));
                    } else if (item.getClass() == Arco.class) {
                        arvore.addContent(criarArco((Arco) item, i));
                    }
                }
                armazenamentoArquivo(documento);
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void armazenamentoArquivo(Document documento) throws IOException {
        XMLOutputter xout = new XMLOutputter();
        Format formatoXML = Format.getPrettyFormat();
        xout.setFormat(formatoXML.setEncoding("ISO-8859-1"));
        FileWriter caminhoArquivo = new FileWriter(new File("C:/temp/ArvoreSCORM.xml"));
        xout.output(documento, caminhoArquivo);
    }

    public Element criarNoh(Noh item) {
```

```
81
82          Element nohXML = new Element("NOH");
83
84          nohXML.setAttribute("id", item.getId());
85          nohXML.setAttribute("ds_objeto", item.getDsObjeto());
86          nohXML.setAttribute("x", String.valueOf(item.getX()));
87          nohXML.setAttribute("y", String.valueOf(item.getY()));
88          nohXML.setAttribute("ie_item_inicial_curso", ((Noh) item).isEhItemInicialCurso() ? "S" : "N");
89          nohXML.setAttribute("ie_item_final_curso", ((Noh) item).isEhItemFinalCurso() ? "S" : "N");
90
91          Element idPai = new Element("ID_PAI");
92          idPai.setText(item.getIdPai() == null ? " " : item.getIdPai().getDsObjeto());
93
94          Element idItemAnterior = new Element("ID_ITEM_ANTERIOR");
95          idItemAnterior.setText(item.getItemAnterior() == null ? " " : item.getItemAnterior().getDsObjeto());
96
97          // Verificação dos objetivos do item
98          if (item.getListaObjetivos() != null
99                  && !item.getListaObjetivos().isEmpty()) {
100             idObjetivo = new Element("OBJETIVOS");
101
102             for (int j = 0; j < item.getListaObjetivos().size(); j++) {
103                 objetivos = item.getListaObjetivos().get(j);
104
105                 Element obj = criarObjetivos(objetivos);
106
107                 idObjetivo.addContent(obj);
108
109
110                 // Verificação das ações do objetivo
111                 if (objetivos.getListaAcoes() != null
112                         && !objetivos.getListaAcoes().isEmpty()) {
113
114                     idAcao = new Element("ACOES");
115                     obj.addContent(idAcao);
116                     for (int k = 0; k < objetivos.getListaAcoes().size(); k++) {
117                         acoes = objetivos.getListaAcoes().get(k);
118
119                         Element acao = criarAcoes(acoes);
120
121                         idAcao.addContent(acao);
122
123                         // VerificaÃ§Ã£o de materiais anexos
124                         if (acoes.getListaMateriaisAnexos() != null
125                                 && !acoes.getListaMateriaisAnexos().isEmpty()) {
126
127                             idMaterialAnexo = new Element("MATERIAL_ANEXO");
128                             acao.addContent(idMaterialAnexo);
129                             for (int z = 0; z < acoes.getListaMateriaisAnexos().size(); z++) {
130                                 arquivoAnexo = acoes.getListaMateriaisAnexos().get(z);
131
132                                 idMaterialAnexo.addContent(criarMateriaisAnexos(arquivoAnexo, z));
133                             }
134                         }
135
136                     }
137                 }
138
139             }
140
141         }
142
143         nohXML.addContent(idPai);
144         nohXML.addContent(idItemAnterior);
145
146         if (item.getListaObjetivos() != null
147                 && !item.getListaObjetivos().isEmpty()) {
148             nohXML.addContent(idObjetivo);
149         }
150
151         return nohXML;
152     }
```

Figura 21 – Classe `GerarArquivoXml`

Na Figura 22 é demonstrado como fica a estrutura do arquivo XML gerado pela classe `GerarArquivoXml`.

```
▼<ARVORE>
 ▼<NOH id="Noh" ds_objeto="Introdução" x="58" y="235" ie_item_inicial_curso="S" ie_item_final_curso="N">
    <ID_PAI/>
    <ID_ITEM_ANTERIOR/>
   ▼<OBJETIVOS>
    ▼<OBJETIVO id="1" nr_squencia="1" ds_titulo="Inicia word" ds_objetivo="Abrir o aplicativo word" ie_status="P">
     ▼<ACOES>
      ▼<ACAO id="1" nr_squencia="1" ds_titulo="Barra de tarefas Windows" ds_acao="Clicar no botão iniciar do sistema operacional e seguida se dirigir programas -> micros
       office -> Word" ie_status="P">
        ▼<MATERIAL_ANEXO>
           <ANEXO id="0" nome="WINWORD.EXE" caminho="C:\Program Files\Microsoft Office\Office12\WINWORD.EXE"/>
         </MATERIAL_ANEXO>
        </ACAO>
      </ACOES>
     </OBJETIVO>
    </OBJETIVOS>
  </NOH>
 ▼<TRANSICAO id="TRANSICAO" ds_objeto="T1" x="199" y="234">
    <ID_PAI/>
    <ID_ITEM_ANTERIOR>Introdução</ID_ITEM_ANTERIOR>
  </TRANSICAO>
  <Arco id="Arco" nr_sequencia="2" vl_x_ponto_inicial="58" vl_y_ponto_inicial="235" vl_x_ponto_final="199" vl_y_ponto_final="234"/>
  <Arco id="Arco" nr_sequencia="3" vl_x_ponto_inicial="199" vl_y_ponto_inicial="234" vl_x_ponto_final="330" vl_y_ponto_final="133"/>
  <Arco id="Arco" nr_sequencia="4" vl_x_ponto_inicial="199" vl_y_ponto_inicial="234" vl_x_ponto_final="296" vl_y_ponto_final="310"/>
 ▼<TRANSICAO id="TRANSICAO" ds_objeto="T2" x="437" y="243">
    <ID_PAI/>
    <ID_ITEM_ANTERIOR>Lição 01</ID_ITEM_ANTERIOR>
  </TRANSICAO>
 ▼<NOH id="Noh" ds_objeto="Prova" x="539" y="243" ie_item_inicial_curso="N" ie_item_final_curso="S">
    <ID_PAI/>
    <ID_ITEM_ANTERIOR>T2</ID_ITEM_ANTERIOR>
  </NOH>
  <Arco id="Arco" nr_sequencia="7" vl_x_ponto_inicial="330" vl_y_ponto_inicial="133" vl_x_ponto_final="437" vl_y_ponto_final="243"/>
  <Arco id="Arco" nr_sequencia="8" vl_x_ponto_inicial="296" vl_y_ponto_inicial="310" vl_x_ponto_final="437" vl_y_ponto_final="243"/>
  <Arco id="Arco" nr_sequencia="9" vl_x_ponto_inicial="437" vl_y_ponto_inicial="243" vl_x_ponto_final="539" vl_y_ponto_final="243"/>
 ▼<GRUPO id="GRUPO" ds_objeto="Lições" x="281" y="118">
    <ID_PAI/>
    <ID_ITEM_ANTERIOR/>
   ▼<NOH id="Noh" ds_objeto="Lição 01" x="330" y="133" ie_item_inicial_curso="N" ie_item_final_curso="N">
      <ID_PAI>Lições</ID_PAI>
      <ID_ITEM_ANTERIOR>T1</ID_ITEM_ANTERIOR>
    </NOH>
   ▼<NOH id="Noh" ds_objeto="Lição 02" x="296" y="310" ie_item_inicial_curso="N" ie_item_final_curso="N">
      <ID_PAI>Lições</ID_PAI>
      <ID_ITEM_ANTERIOR>T1</ID_ITEM_ANTERIOR>
    </NOH>
  </GRUPO>
</ARVORE>
```

Figura 22 – XML `ArvoreSCORM.xml`

3.3.2 Operacionalidade da implementação

Nesta seção são apresentadas todas as funcionalidades da ferramenta. Ao efetuar a abertura da página o usuário irá encontrar a tela apresentada na Figura 23, com a parte de edição e a árvore de atividades em branco. Nesta figura são apresentadas as informações no editor e árvore de atividades para demonstração. Ainda, é possível verificar áreas distintas numeradas para identificação assim possibilitando o detalhamento de suas funcionalidades.

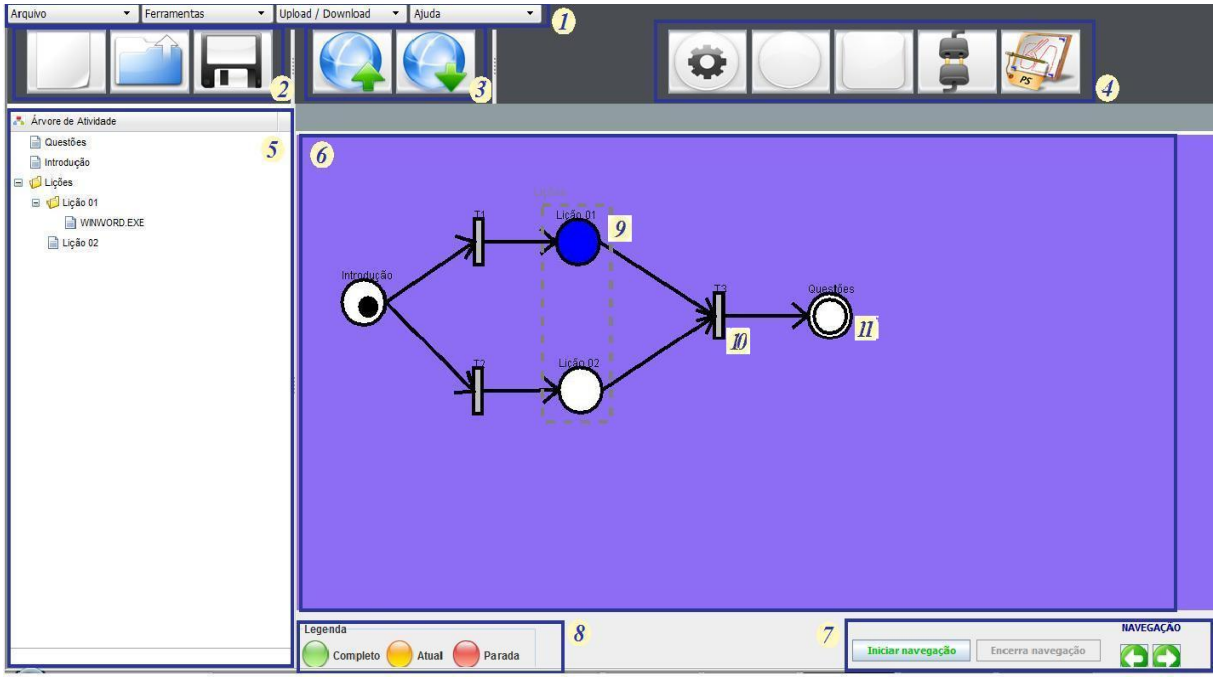


Figura 23 – Tela do editor

A área de menus (Figura 23(1)) e área de botões (Figura 23(2, 3, 4)) possuem praticamente as mesmas funcionalidades, uma única diferença que no menu existe a opção de `menu ajuda`, que contem os itens `ajuda do editor` e `sobre editor`.

Figura 24 – Botões padrões

Na Figura 24 são apresentados os botões padrões da ferramenta que praticamente todas as aplicações possuem, da esquerda para a direita, tem-se:

a) Botão `novo` que permite a criação de um novo curso, abrindo janela (Figura 25) para inclusão de um nome ao curso;
b) `Abertura` de um curso já existente na base de dados, abrindo janela (Figura 26) para escolha do carregamento do curso;
c) `Salvar` as informações do curso atual.

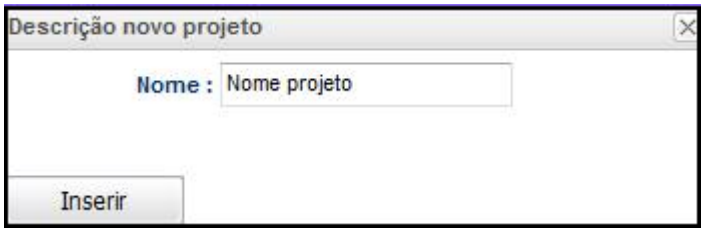


Figura 25 – Janela descrição de novo projeto

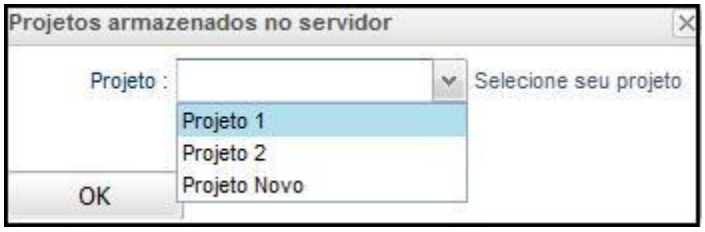


Figura 26 – Janela projetos armazenados no servidor

Figura 27 – Botões `upload` e `download`

Na Figura 27 são apresentados os botões de `upload` e `download` da ferramenta, da esquerda para a direita, tem-se:

1) Botão `upload` que tem por finalidade empacotar o curso atual com seus respectivos arquivos, além do arquivo `imsmanifest.xml`, abrindo janela (Figura 28) para escolha de destino onde deve ser embalado;
2) Botão `download` permite a procura de curso empacotado, para efetuar o desempacotamento para utilização na ferramenta, abrindo janela (Figura 29) para encontrar a origem do pacote e escolha de destino onde deve ser desempacotado.

Figura 28 – Janela de *upload*

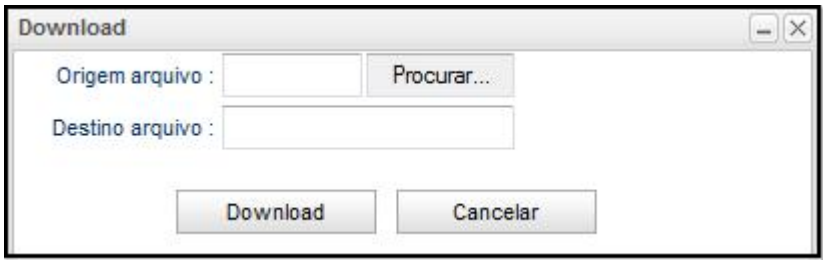


Figura 29 – Janela de *download*

Figura 30 – Botões de edição da ferramenta

Na Figura 30 são apresentados os botões de edição da ferramenta, da esquerda para a direita, tem-se:

a) Botão `selecionar objeto`, responsável pela seleção de qualquer objeto da tela permitindo movimentação de qualquer objeto da tela, além de utilização das opções do mouse botão lado direito para escolha de opções;
b) Botão `adiciona objeto de aprendizagem`, responsável pela inserção de um novo objeto de aprendizagem, assim representado pela circunferência no editor;
c) Botão `adiciona objeto de transição`, responsável pela inserção de uma nova transição que tem por finalidade criar uma ponte entre outro objeto de aprendizagem, assim representado pelo retângulo no editor;
d) Botão `ligação entre objetos`, responsável pela ligação entre os elementos da tela garantindo uma sequência no curso, assim representado pelas flechas no editor;
e) Botão `criar grupo de objetos`, responsável pela criação de conjuntos de objetos de aprendizagem que serão filhas do objeto de aprendizagem anterior da sequência Para sua utilização deve-se, clicar no botão selecionar objeto em seguida pressionar o botão `shift` do teclado e com o botão esquerdo do mouse selecionar os objetos de aprendizagem da tela que farão parte do grupo, ao finalizar a escolha clicar no botão criar grupo de objetos.

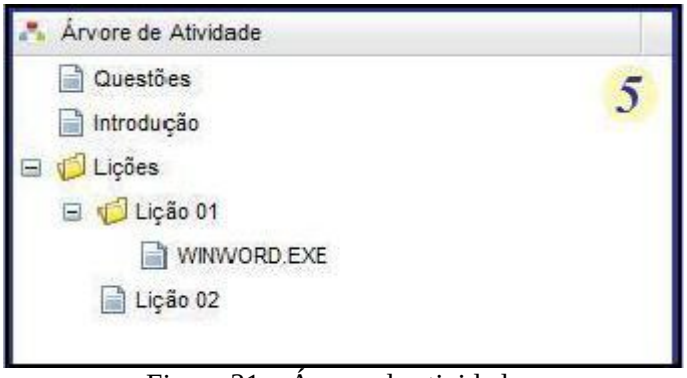


Figura 31 – Árvore de atividades

Na Figura 31 é apresentada a árvore de atividade que tem por finalidade representar os objetos de aprendizagem na sua sequência, e respectivamente seus objetivos e materiais.

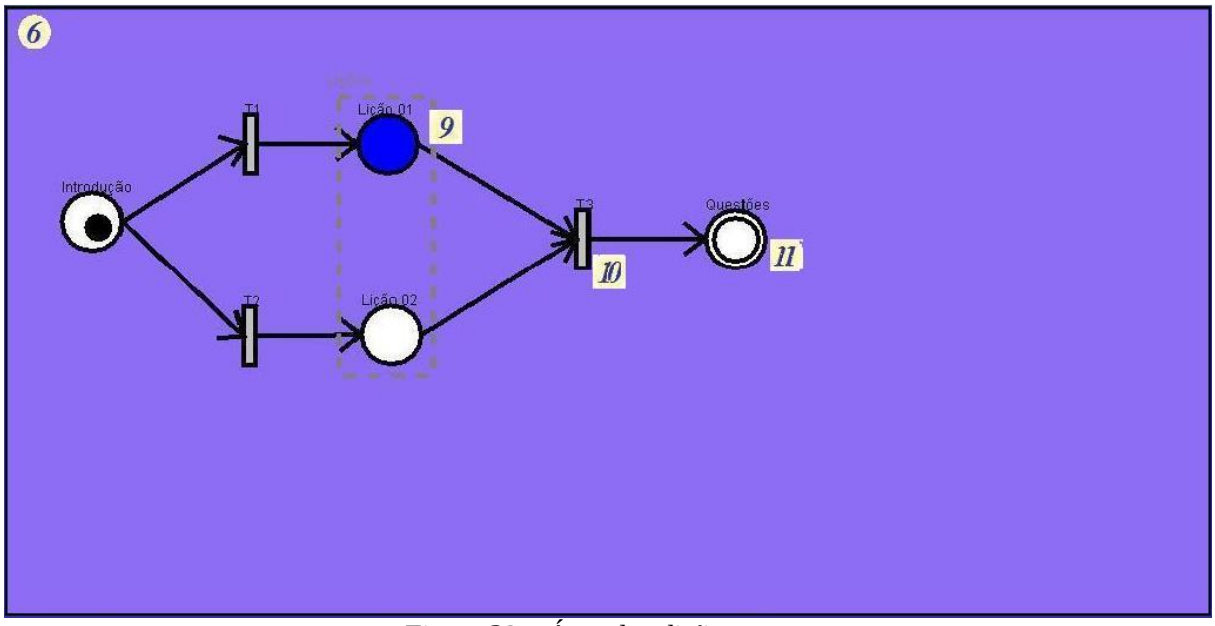


Figura 32 – Área de edição

Na Figura 32 é apresentado o editor que tem como finalidade a montagem do sequênciamento, conforme a utilização das opções anteriores.

Figura 33 – Botões de navegação

Na Figura 33 são apresentados os botões de navegação que tem a finalidade de navegação do curso conforme a sequência de montagem, da esquerda para a direita, tem-se:

a) Botão `iniciar navegação`, responsável pelo posicionamento no objeto de aprendizagem inicial e demarcar a inicialização da navegação;
b) Botão `encerra navegação`, responsável pelo fim da navegação pelos objetos de aprendizagem, que pode ser feita pelo próprio usuário e quando chegar ao último objeto de aprendizagem;
c) Botão `anterior`, responsável pelo posicionamento no elemento anterior ao atual na navegação;
d) Botão `seguinte`, responsável pelo posicionamento no elemento seguinte ao atual na navegação.

Figura 34 – Botões da legenda

Na Figura 34 é apresentada a legenda de navegação do editor, da esquerda para a direita, tem-se:

a) `Completo`, indica quando a navegação já passou pela opção e a mesma já foi executada;
b) `Atual`, indica qual a opção atual está ocorrendo a execução;
c) `Parada`, indica onde foi efetuada a parada da navegação feita pelo usuário ou pelo último elemento da sequência.

Na (Figura 23(9)) é apresentado um objeto de aprendizagem representado pelo círculo, ao utilizar a opção de mouse botão lado direito sobre o objeto selecionado é apresentado uma caixa de diálogo com uma série de opções para edição conforme na Figura 28, de cima para baixo, tem-se:

a) `Alterar titulo`, responsável pela alteração da descrição do título do objeto de aprendizagem, assim efetuando abertura de caixa de diálogo para alteração (Figura 29);
b) `Deletar objeto`, efetuada a exclusão do objeto de aprendizagem selecionada da tela;
c) `Definir lição final`, responsável pela definição do ltimo objeto de aprendizagem do curso, conforme Figura 23(11);
d) `Ver \ Objetivos`, efetuada abertura da janela (Figura 28), para verificação dos objetivos do objeto de aprendizagem, além da inclusão de `novo` ou `remoção` dos existentes.

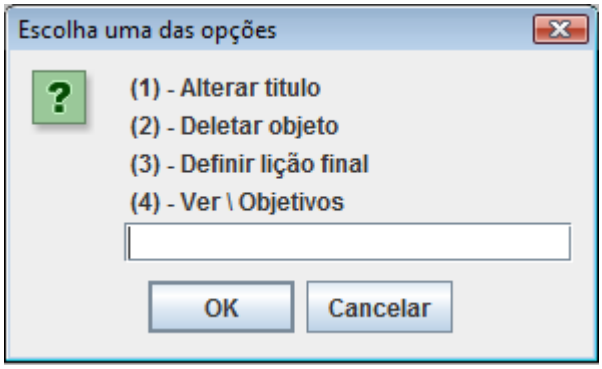


Figura 35 – Opções de edição do objeto de aprendizagem

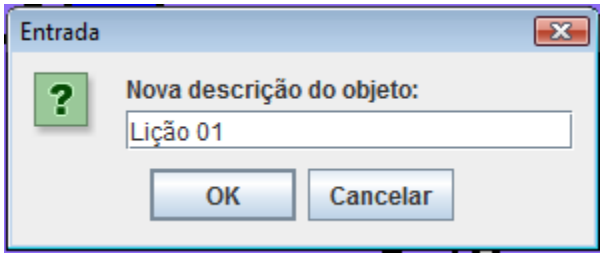


Figura 36 – Alterar descrição do título do objeto de aprendizagem

Ao utilizar a opção 4 conforme Figura 28 é efetuada abertura da janela Figura 30 para verificação dos objetivos do objeto de aprendizagem, também é possível a inclusão e remoção

de objetivos, respectivamente os botões `novo objetivo` e `remove objetivo` da Figura 30. Quando utilizado o botão `novo objetivo` é efetuada abertura de uma janela de fórmula do objetivo (Figura 31), para inserção do título e descrição, já o botão remove objetivo efetua a exclusão do registro selecionado.

Para efetuar alguma alteração no objetivo ou para verificação e inclusão de ações é preciso efetuar duplo clique ou pressionar a tecla enter do teclado, assim é apresentado uma caixa de diálogo com as opções alterar objetivo e inserir ação conforme na Figura 32. Utilizando a opção `alterar objetivo` é efetuado abertura da janela formulário objetivo (Figura 31), com as informações atuais para edição. Já opção inserir ação é efetuado abertura da janela ações do objetivo (Figura 33).

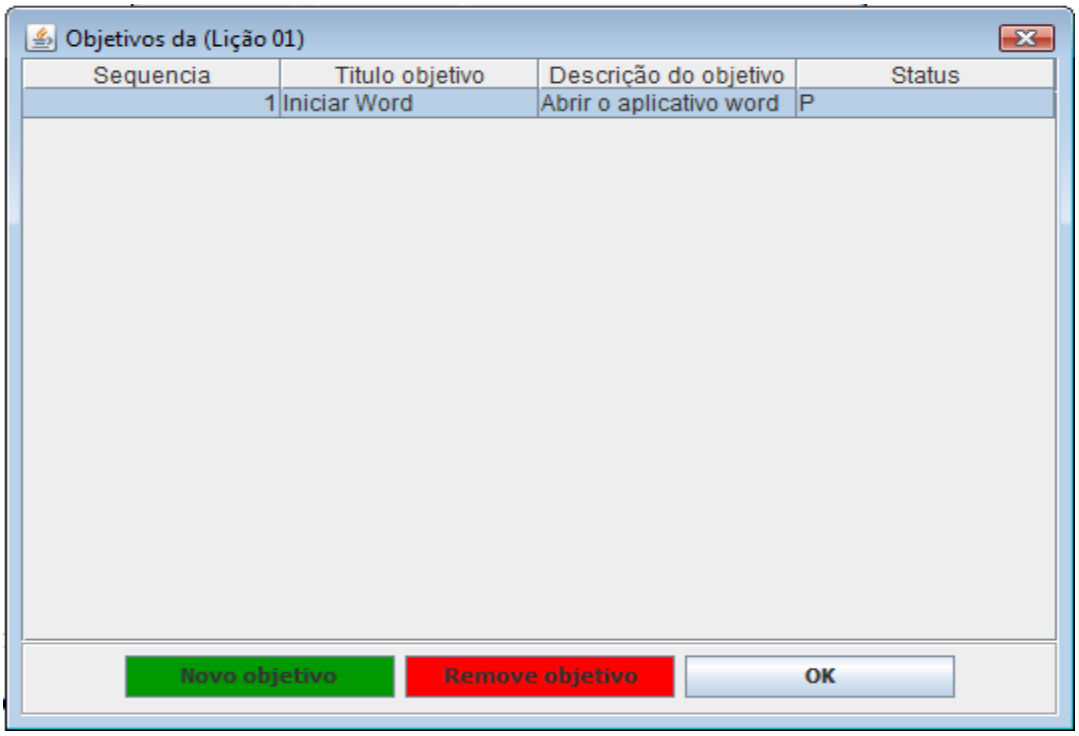


Figura 37 – Janela objetivos

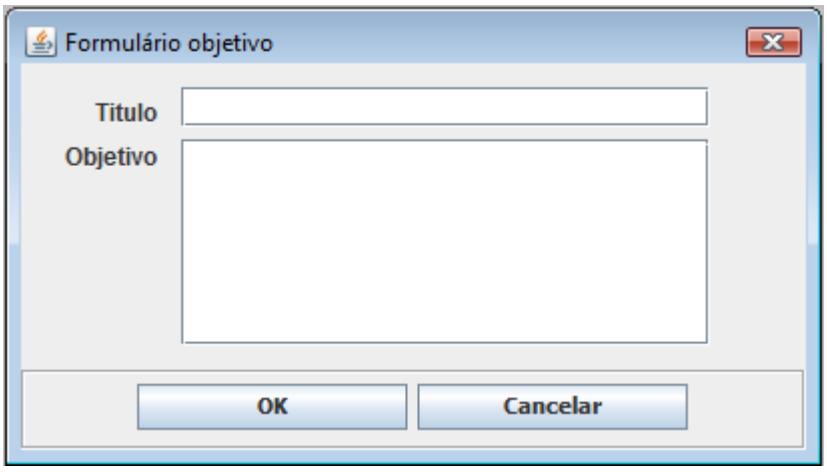


Figura 38 – Janela formulário objetivo

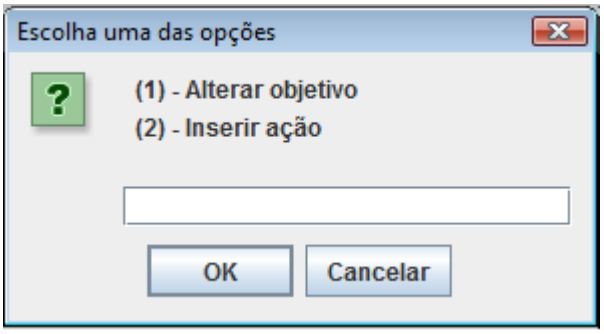


Figura 39 – Opções de edição do objetivo

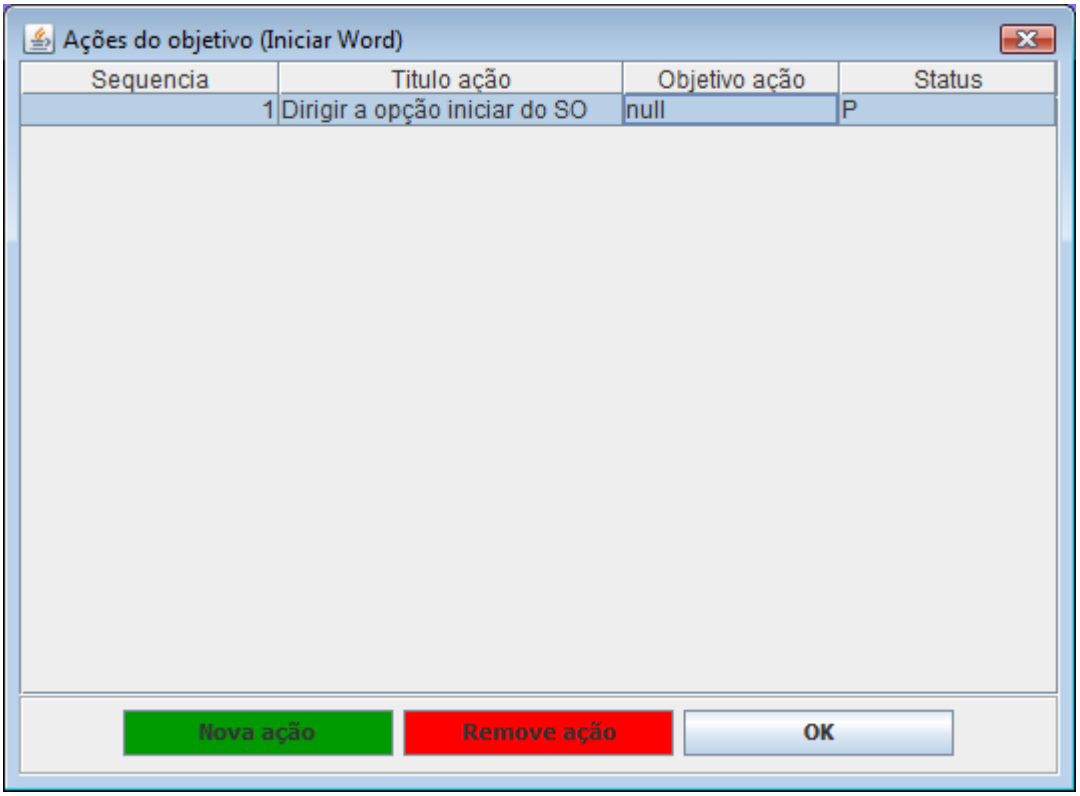


Figura 40 – Janela ações

Efetuada abertura da janela Figura 33 para verificação das ações dos objetivos, também possível a inclusão e remoção de ações, respectivamente os botões `nova ação` e `remove ação` da Figura 33. Quando utilizado o botão `novo objetivo` é efetuada abertura de uma janela de fórmula da ação (Figura 34), para inserção do titulo e descrição. Já o botão `remove ação` efetua a exclusão do registro selecionado.

Para efetuar alguma alteração de uma ação ou para verificação e inclusão de ações é preciso efetuar duplo clique ou pressionar a tecla enter do teclado, é apresentado uma caixa de diálogo com as opções `alterar ação` e `inserir materiais` conforme Figura 35.

Utilizando a opção `alterar ação` é aberta a janela `formulário ação` (Figura 34), com as informações atuais para edição. Já opção inserir materiais é efetuado abertura da `janela dos materiais` anexos a cada ação (Figura 36).

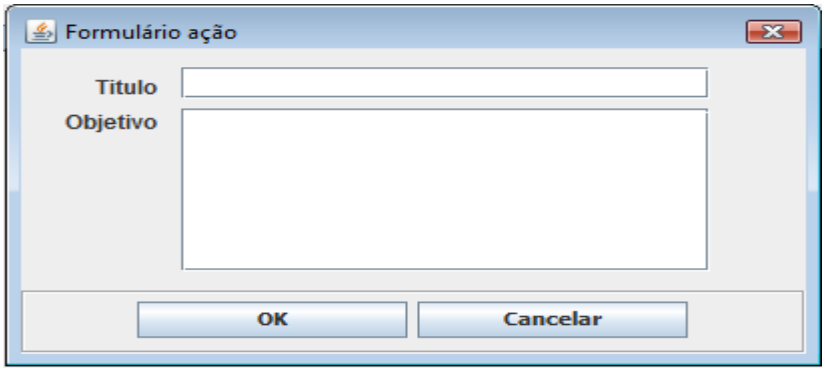


Figura 41 – Janela formulário ação

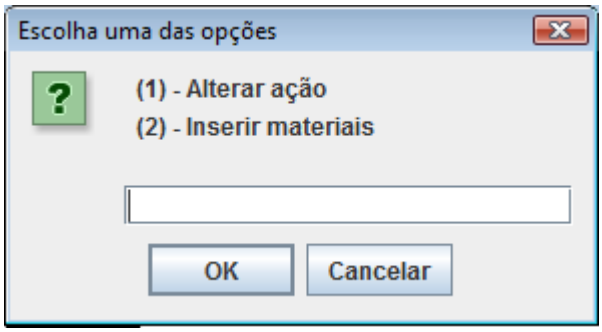


Figura 42 – Opções de edição da ação

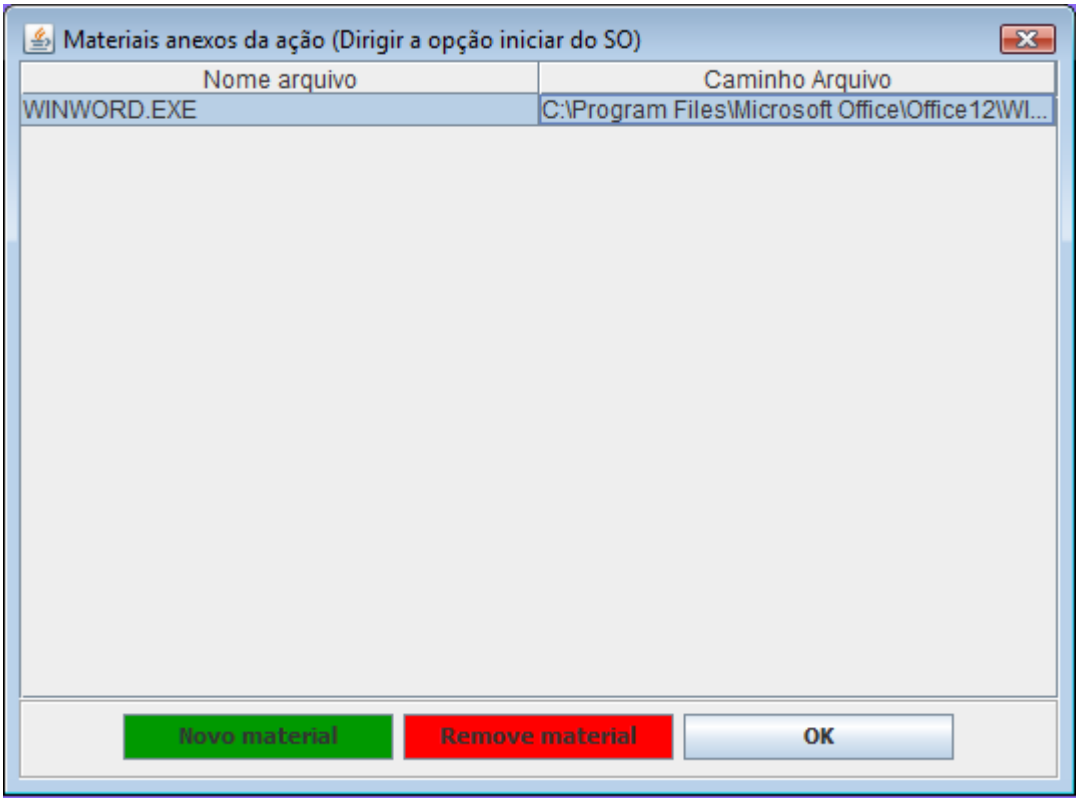


Figura 43 – Janela de materiais anexos

Com abertura da `janela de materiais anexos da ação` (Figura 36), é possível a inclusão e remoção de materiais, respectivamente os botões `novo material` e `remove material`. Ao clicar no botão `novo material` é efetuada abertura de uma janela para encontrar o arquivo (Figura 37), que fará vinculo com a ação ao clicar no `botão salvar`. Já o botão `remove material`

efetua a exclusão do registro selecionado. Para que seja executada abertura do material anexo somente é preciso efetuar duplo clique no material (Figura 36) ou pressionar a tecla enter do teclado.

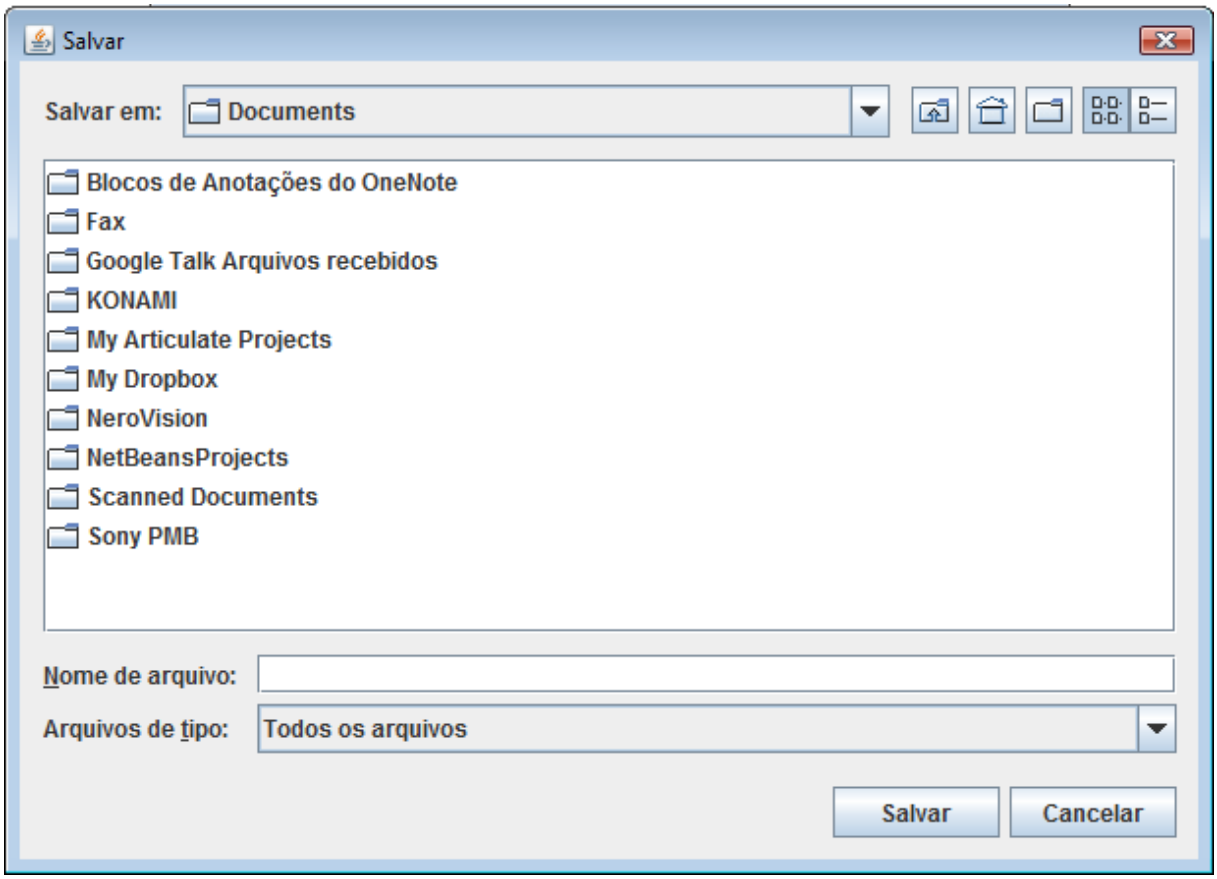


Figura 44 – Janela para inclusão de materiais

Na (Figura 23(10)) é apresentado um objeto de transição representado pelo retângulo, ao utilizar a opção de mouse botão lado direito sobre o objeto selecionado é apresentado uma caixa de diálogo com uma série de opções para edição conforme Figura 38, de cima para baixo, tem-se:

a) `Alterar titulo`, responsável pela alteração da descrição do titulo da transição entre objetos, assim efetuando abertura de caixa de diálogo para alteração (Figura 39);

b) `Deletar objeto`, efetuada a exclusão da transição selecionada da tela.

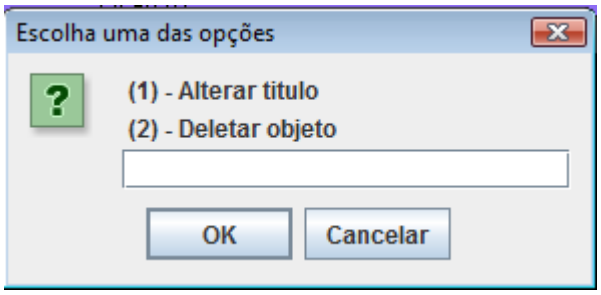


Figura 45 – Opções de edição da transição

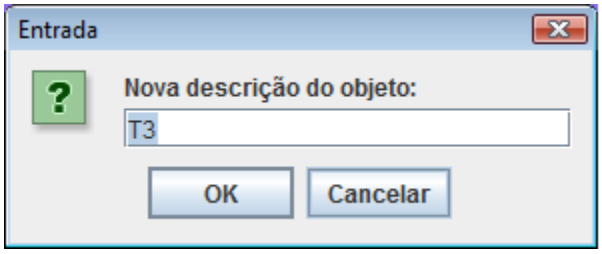


Figura 46 – Alterar descrição da transição

3.4 RESULTADOS E DISCUSSÃO

No presente trabalho foi desenvolvida uma ferramenta para utilização em navegadores e para analisar a aplicação foram efetuados testes de memória e transmissão de dados entre cliente e servidor que podem ser vistos nas seções 3.4.1 e 3.4.2.

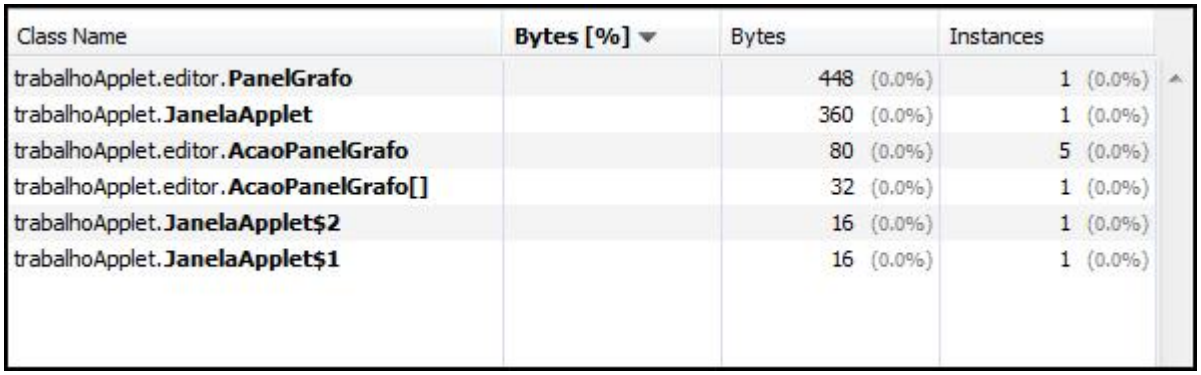
Durante os testes não foi identificado qualquer travamento na aplicação. Importante relatar que não houve nenhuma outra aplicação utilizando a conexão com a internet, no qual foi utilizado a máquina local para efetuar os testes.

O hardware utilizado possui os seguintes configurações:

a) *notebook* Dell vostro 1510;
b) processador Intel Core 2 Duo 2.10 Gigahertz, com 2 Megabytes de cachê L2;
c) quatro gigabytes de memória RAM DDR2 ;
d) sistema operacional Windows Vista Business.

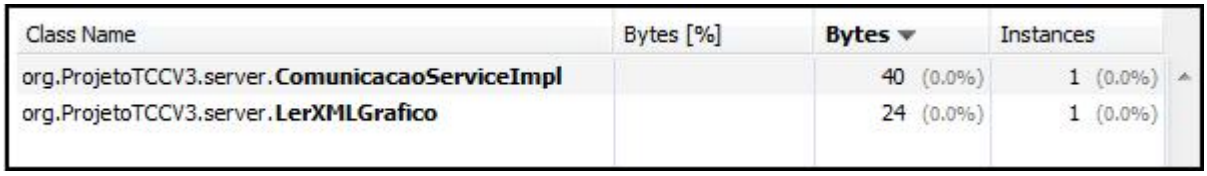
3.4.1 Memória

Para a análise de uso de memória os testes de desempenho foram efetuados utilizando o simulador VisualVM disponível pela Oracle, para monitorar e solucionar problemas de aplicativos Java. Figura 40 demonstra a primeira amostra da leitura de quantos bytes cada objeto utiliza quando a ferramenta é inicializada, isso pela abertura do editor em `applet` e na sequencia Figura 41 pelo *framework* Smartgwt. Visto que a classe `AcaoPanelGrafo` (Figura 40) já começa com cinco instâncias com valor total de 80 bytes o que significa 16 bytes para cada instâncias.



Class Name	Bytes [%]	Bytes	Instances
trabalhoApplet.editor.**PanelGrafo**		448 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet**		360 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**AcaoPanelGrafo**		80 (0.0%)	5 (0.0%)
trabalhoApplet.editor.**AcaoPanelGrafo[]**		32 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet$1**		16 (0.0%)	1 (0.0%)

Figura 47 – Quantidade de bytes por classe na inicialização do editor



Class Name	Bytes [%]	Bytes	Instances
org.ProjetoTCCV3.server.**ComunicacaoServiceImpl**		40 (0.0%)	1 (0.0%)
org.ProjetoTCCV3.server.**LerXMLGrafico**		24 (0.0%)	1 (0.0%)

Figura 48 – Quantidade de bytes por classe na inicialização no *framework*

Já com a inclusão de um curso conforme na Figura Figura 23(6), além da abertura de todas as janelas disponíveis no editor `applet`, ao clicar nas opções de um objeto e aprendizagem é tirada a amostra da leitura de quantos bytes cada objeto utiliza. Verificado que a cada exclusão de qualquer elemento do editor, o mesmo ainda se encontra na memória até que o *garbage colletor* da máquina virtual do Java o destrua, e cada elemento novo inserido na tela, vai acrescentando bytes e instâncias no objeto que se encontra em memória. Observando a Figura 42 é possível verificar que existem 7 instâncias da classe `Arco` que ocupam 448 (64 bytes para cada objeto), 4 instâncias de classe `Noh` que ocupam 288 bytes (72 bytes para cada objeto), 3 instâncias da classe `Rect` que ocupam 448 (64 bytes para cada objeto). Em relação aos arquivos incluídos no curso não foi efetuada alguma medição, portanto não foi verificado tamanho padrão do pacote.

Class Name	Bytes [%]	Bytes	Instances
trabalhoApplet.janelas.**ModalAcaoFormulario**		480 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes**		480 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivoFormulario**		480 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos**		472 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos**		464 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**Arco**		448 (0.0%)	7 (0.0%)
trabalhoApplet.editor.**PanelGrafo**		448 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**InserirMaterial**		424 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet**		360 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**Noh**		288 (0.0%)	4 (0.0%)
trabalhoApplet.editor.**Rect**		192 (0.0%)	3 (0.0%)
trabalhoApplet.editor.**Grupo**		88 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**AcaoPanelGrafo**		80 (0.0%)	5 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$5**		32 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$5**		32 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**AcaoPanelGrafo[]**		32 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**Objetivos**		32 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$5**		32 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**Acoes**		32 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$6**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$7**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$3**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivos$4**		16 (0.0%)	1 (0.0%)
trabalhoApplet.editor.**ArquivoAnexo**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$6**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$7**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$4**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalMateriaisAnexos$3**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivoFormulario$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivoFormulario$3**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalObjetivoFormulario$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcaoFormulario$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcaoFormulario$3**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcaoFormulario$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.**JanelaApplet$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$3**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$4**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$1**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$2**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$6**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**ModalAcoes$7**		16 (0.0%)	1 (0.0%)
trabalhoApplet.janelas.**InserirMaterial$1**		16 (0.0%)	1 (0.0%)
XMLArvore.**GeraArquivoXml**		40 (0.0%)	1 (0.0%)
XMLArvore.**LerXMLGrafico**		24 (0.0%)	1 (0.0%)

Figura 49 – Quantidade de bytes por classe utilizando todos os elementos do editor

3.4.2 Servidor

A utilização do servidor é para armazenamento da ferramenta, e para interação com o banco, executando as informações conforme ação do usuário. Para a marcação do tempo médio que a ferramenta leva para enviar e receber as informações do *WebService*, foram colocadas marcações antes de chamar um método do *webservice* e após o retorno, utilizando método de depuração para apresentar tais informações na saída do compilador, conforme na Figura 43. Para efetuar os testes foi utilizado uma rede de Petri que continha 4 objetos de aprendizagem, 3 transições e 7 ligações entre os elementos da tela, no qual 2 objetos de aprendizagem possuíam cada um 2 objetivos e que cada possuía uma ação, tal que cada ação possui um material anexo. Em seguida foi efetuado o empacotamento da rede. Um fator interessante para efetuação de testes no servidor, e que pode ser incluído como extensão seria a utilização de uma rede de Petri com proporções maiores com 15 ou mais objetos de aprendizagem, objetivos, ações, materiais, além das transições e ligações entre os elementos da tela.

```
Início da chamada do web service para busca dos curso: 30/11/2012 12:20:32.777
 Fim da chamada do web service para busca dos curso: 30/11/2012 12:20:33.293
Início da chamada do web service para buscas dos elementos do curso: 30/11/2012 12:20:36.677
 Fim da chamada do web service para busca carregamento dos elementos do curso: 30/11/2012 12:20:37.396
Início da chamada do web service para salvar curso: 30/11/2012 12:20:48.317
 Fim da chamada do web service para salvar curso: 30/11/2012 12:20:49.969
Início da chamada do web service para efetuar empacotamento de arquivos: 30/11/2012 12:20:59.995
 Fim da chamada do web service para efetuar empacotamento de arquivos: 30/11/2012 12:21:00.188
Início da chamada do web service para inserção de novo curso: 30/11/2012 12:21:32.718
 Fim da chamada do web service para inserção de novo curso: 30/11/2012 12:21:32.857
```

Figura 50 – Marcações utilizadas para medir o tempo

Desta forma são analisadas as marcações pares subtraindo o tempo da segunda pelo tempo da primeira, resultando no tempo em que aplicação leva para enviar e receber informações do *webservice*. Para realização dos testes foi utilizado um curso com quatro objetos de aprendizagem, que dois possuem materiais anexos. Na Tabela 1 é possível ver o tempo médio de cada etapa levou para ser concluída.

Tabela 1 – Tempo médio para comunicação com *webservice*

TEMPO MÉDIO DAS ETAPAS PARA COMUNICAÇÃO COM WEBSERVICE			
Etapa	**Tempo em milissegundos**		
	Teste 1	**Teste 2**	**Teste 3**
Busca curso	2360	415	430
Busca elementos do curso	1040	674	642
Salvar curso	1595	1588	1395
Empacotamento de arquivos	203	168	174
Novo curso	186	145	142

3.4.3 Relação dos trabalhos correlatos

Uma vez construída e testada a ferramenta, pode-se fazer o comparativo com os trabalhos correlatos apresentados, conforme apresentado no Quadro 07. Em relação a Ressener (1997) e Giordani (1997), não foram incluídos na comparação pelo fato das ferramentas somente ter como disponibilidade a criação e edição de redes de Petri. O item de sequenciamento e navegação (SCORM), não foi implementado na ferramenta utilizando o padrão SCORM, a geração das informações do curso são geradas num arquivo XML especifico da ferramenta não sendo possível seu compartilhamento.

	Ferramenta desenvolvida	**MINA**	**VOSSAT**
Interface web	✓	×	✓
Permissões de acesso	×	×	✓
Criação e edição de redes de Petri	✓	✓	✓
Criação de objetivos em relação ao objeto de aprendizagem	✓	✓	✓
Armazenamento SGDB	✓	×	×
Apresentação da árvore de atividades	✓	✓	✓
Sequencionamento e navegação (SCORM)	×	✓	✓
Compactar\ Extrair pacote (SCORM)	×	✓	✓

Quadro 11– Características da ferramenta desenvolvida e trabalhos correlatos

4 CONCLUSÕES

O desenvolvimento deste ambiente web teve como objetivos desenvolver um editor de sequenciamento e navegação utilizando redes de Petri no padrão SCORM, conforme apresentado em Lin et al. (2005) e Su et al. (2005), para criação de cursos disponibilizados on-line, sendo a ferramenta disponibilizada por um servidor de aplicações acessado por qualquer computador que tenha acesso à internet.

A utilização de SN permite criar um conteúdo que possua inteligência com regras simples para que o material seja apresentado de acordo com a evolução do aluno. As ferramentas de autoria que possuem o recurso de editar o SN pecam quando não apresentam uma representação gráfica destas regras. Uma dificuldade encontrada foi a busca por materiais didáticos (livros) que abordassem o tema. Além disso, a maioria dos materiais encontrados estavam disponíveis on-line em inglês de difícil coompreensão. Sendo um fator determinante para não conclusão dos objetivos do trabalho que seria a criação da ferramenta no padrão SCORM.

A utilização de redes de Petri no editor seria a fácil representação gráfica que facilita o usuário a compreender o comportamento das regras de sequência, sendo assim aplicando as características de redes de Petri, para modelar e diminuir a complexidade do SN.

Para o usuário, a ferramenta facilita sua mobilidade, pois não é preciso instalar na máquina em que vai trabalhar, bastando ter apenas acesso a internet.

As tecnologias utilizadas para a construção da ferramenta se mostraram eficientes cada uma na sua área. GWT e SmartGwt pela utilização do processo cliente servidor e criação de interfaces amigáveis, e o applet por sua utilização da biblioteca `Graphics` para criação de formas geométricas complexas no editor da ferramenta. Mas houve dificuldades para a transmissão de informações entre as tecnologias, sendo assim criado um arquivo XML com as informações do curso para efetuar a troca de informações entre ambas, tanto para mostrar a árvore de atividades, carregar as informações no banco e descarregá-las, empacotar e desempacotar arquivos.

A ferramenta possui algumas limitações. A primeira seria a não criação do padrão SCORM, não podendo haver compartilhamento de pacotes no padrão, a não utilização das quatro regras do SN: modo de controle de sequenciamento, regras de sequenciamento, regras de retropropagação, e também a não atualização instantânea da árvore de atividades quando incluso um novo elemento no editor.

4.1 EXTENSÕES

Sugerem-se as seguintes extensões para continuidade do trabalho:

a) criar janela de acesso para distinguir o usuário professor e aluno;
b) permitir criar um arquivo de manifesto SCORM;
c) permitir extrair o arquivo de manifesto de um pacote SCORM;
d) utilizar as quatro regras do SN: modo de controle de sequenciamento, regras de sequenciamento, regras de retropropagação e regras de objetivos do SCORM;
e) traduzir as regras de SN SCORM para uma rede de Petri;
f) exibir a árvore de atividades do arquivo de manifesto do SCORM.

REFERÊNCIAS BIBLIOGRÁFICAS

ADL. **Advanced distributed learning**. [S.1.], 2012. Disponível em: <http://www.adlnet.org>. Acesso em: 10 out. 2012.

_____. **SCORM® 2004 3rd edition overview version 1.0**. ADL: Virginia, 2006.

_____. **SCORM® 2004 4rd content aggregation model (CAM) version1.1**. ADL: Virginia, 2009.

CARDOSO, Janette; VALETTE, Robert. **Redes de Petri**. Florianópolis: Editora Universidade Federal de Santa Catarina, 1997.

GIORDANI, Rodrigo G. **Software para análise e execução de redes de Petri órdinarias**. 1997. 88 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GOOGLE. **Google web toolkit**. [S.l.], 2010. Disponível em: <http://code.google.com/intl/en/webtoolkit/overview.html/>. Acesso em: 19 jul. 2012.

IRLBECK, Sonja; MOWAT, Joanne. Learning content management systems (LCMS). In: Harman, Keith; Koohang, Alex. **Learning objects:** standards, metadata, repositories & LCMS. California: Informing Science Press, 2007. cap. 6, p. 157-184.

LIN, H. W. et al. Applying Petri nets to model learning sequence with SCORM specification in collaborative learning. In: OF THE 2005 INTERNACIONAL CONFERENCE ON ACTIVE MEDIA TECHNOLOGY, 2005, Kagawa. **Proceedings…** Kagawa: Kagawa University, 2005. p. 181-186.

PENHA, Dulcinéia O.; FREITAS, Henrique C.; MARTINS, Carlos A. P. S. Modelagem de sistemas computacionais usando Redes de Petri: aplicação em projeto, análise e avaliação. In: ESCOLA REGIONAL DE INFORMÁTICA, 4., 2004, Vitória. **Anais**... Vitória: SBC, 2004. p. 1-40. Disponível em: <http://portal.sbc.org.br/bibliotecadigital/download.php?paper=33>. Acesso em: 20 jul. 2012.

REISIG, Wolfgang. **A primer in Petri net design**. Berlim: Springer-Verlag, 1992.

RESSENER, Fabio A. **Software para simulação visual de redes de Petri ordinárias**. 1997. 125 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SmartGwt. [S.l.], [2011]. Disponível em: <http://code.google.com/p/smartgwt/>. Acesso em: 19 jul. 2012.

SPINER. **O que são widgets?** [S.l.], [2010]. Disponível em: <http://www.spiner.com.br/modules.php?name=News&file=article&sid=1257>. Acesso em: 19 jul. 2012.

SU, Jun M. et al. An object based authoring tool for creating SCORM compliant. In: INTERNACIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 19., 2005, Taipe. **Proceedings…** Taipe: Tamkang University, 2005. p. 209-214.

VAHLDICK, Adilson. **CELINE**: um modelo para utilização e adaptação de conteúdo SCORM em ambientes inteligentes de aprendizagem. 2008. 129 f. Dissertação (Mestrado em Computação Aplicada) - Programa de Mestrado Acadêmico em Computação Aplicada, Universidade do Vale do Itajaí, São José.