DIEGO MARCZAL

UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS

CURITIBA

2010

DIEGO MARCZAL

UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2010

DIEGO MARCZAL

UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre no Programa de Pós-Graduação em Informática da Universidade Federal do Paraná, pela Comissão formada pelos professores:

Orientador: Prof. Dr. Alexandre Ibrahim Direne
Departamento de Informática, UFPR

Prof. Dr. Robinson Vida Noronha
Departamento de Eletrônica, UTFPR - Membro Externo

Prof. Dr. Dr. Andrey Ribardo Pimentel
Departamento de Informática, UFPR - Membro Interno

Prof. Dr. Fabiano Silva
Departamento de Informática, UFPR - Membro Suplente

Curitiba, 20 de maio de 2010

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Diego Marczal, avaliamos o trabalho intitulado, *"UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS"*, cuja defesa foi realizada no dia 20 de maio de 2010, às 14:00 horas, no Auditório do Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 20 de maio de 2010.

Prof. Dr. Alexandre Ibrahim Direne
DINF/UFPR – Orientador

Prof. Dr. Robinson Vida Noronha
UTFPR – Membro Externo

Prof. Dr. Andrey Ricardo Pimentel
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

"A excelência é uma habilidade conquistada
através do treinamento e da prática.
Nós somos aquilo que fazemos com freqüência,
portanto, a excelência não é um ato."
Aristóteles

CAPÍTULO 1

AGRADECIMENTOS

À Deus pelas oportunidades e desafios, por permitir a conclusão de mais uma etapa da minha vida, pela força concedida e por guiar meus passos durante toda minha caminhada.

Aos meus pais, Domingos e Danuta, pelo apoio, compreensão e incentivo, não só durante o período do mestrado, mas em todos os momentos importantes de minha vida.

À minha irmã, Denise, que além de irmã é uma grande amiga.

À minha namorada, Maísa, por todo amor, compreensão e carinho.

À toda minha família, pela ajuda oferecida.

Ao grande professor Alexandre Direne, orientador e grande amigo, pela paciência, competência e compreensão que conduziu este trabalho. Além de toda orientação e incentivo para meu crescimento acadêmico e pessoal. Um exemplo extraordinário de comprometimento com o ensino e a pesquisa. Muito Obrigado!

Ao amigo Eleandro Maschio, pelo incentivo e ajuda durante essa caminhada.

Aos amigos, pelo incentivo nos momentos difíceis e pelos momentos de alegria.

Aos membros da banca examinadora, pelas sugestões que contribuiram para melhoria do trabalho.

Aos professores da UFPR, por toda dedicação na realização de seu trabalho.

SUMÁRIO

LISTA DE SIGLAS

ACT	*Atomic Component of Thought*
ACT-R	*Adaptive Control of Thought¿Rational*
OA	Objetos de Aprendizagem
CARRIE	Controlador de Acesso Reflexivo e Retroativo Indexado por Erros
STI	Sistema Tutor Inteligente
MREs	Múltiplas Representações Externas
FNDE	Fundo Nacional de Desenvolvimento da Educação
SEED	Secretaria de Educação a Distância
MEC	Ministério da Educação e Cultura
MCT	Ministério da Ciência e Tecnologia
UFPR	Universidade Federal do Paraná
C3SL	Centro de Computação Científica e Software Livre
LACTEC	Instituto de Tecnologia para o Desenvolvimento
CETEPAR	Centro de Excelência em Tecnologia Educacional do Paraná
UEL	Universidade Estadual de Londrina
WYSIWYG	*What You See Is What You Get*

LISTA DE FIGURAS

RESUMO

Neste trabalho, são apresentados os aspectos que justificam a necessidade de se projetar a implementar um controlador genérico de interatividade que possa ser aplicado e reutilizado por diversos objetos de aprendizagem. Os principais objetivos do controlador são detalhados em termos de acesso ao conteúdo e como ele enfatiza os aspectos metacognitivos envolvidos nas tarefas típicas de aprendizagem. Além disso, apresenta-se uma nova abordagem para utilizar os erros cometidos pelo aprendiz como uma ferramenta eficiente para reparar e construir o seu conhecimento. A arquitetura e a implementação do arcabouço denominado controlador de acesso reflexivo e retroativo indexado por erros (CARRIE) têm seus módulos principais descritos e exemplificados. Ao final, destacam-se as limitações e as perspectivas sobre o mecanismo de retroação ao contexto de erros proposto.

Palavras-chave: Objetos de Aprendizagem, ambientes interativos de aprendizagem, retroação a contexto de erros.

ABSTRACT

In this paper, we present the aspects that justify the need to design and to implement a generic interactivity controller which can be reused for several learning objects. The main objectives of the controller are detailed in terms of access to content and how it emphasizes the meta-cognitive aspects involved in the typical tasks of learning. Moreover, it presents a new approach to use the errors made by the apprentice as a effective tool to repair and build your knowledge. The architecture and the implementation of the framework called Access Controller reflective and retrospective indexed by errors (CARRIE) have their main modules described and exemplified. Finally, we highlight the limitations and perspectives of the retroaction mechanism to context of errors proposed.

Keywords: Learning Objects, interactive learning environments, retroaction to the context of errors.

CAPÍTULO 2

INTRODUÇÃO

2.1 Problema Central

Atualmente vários objetos de aprendizagem (OA) e simuladores têm sido desenvolvidos para apoiar tarefas de visualização de conceitos e solução de problemas em diversas áreas do conhecimento humano (*e.g.*, Matemática, Física, etc). Por iniciativa coordenada do governo, boa quantidade desses objetos de aprendizagem estão disponíveis em bases de acesso público e gratuito do Ministério de Educação [9]. Além dos software educacionais, essas bases também mantêm outros objetos de aprendizagem na forma de conteúdos pedagógicos digitais como vídeos, arquivos de áudio e texto, imagens, mapas e simulações, os quais são destinados a todos os níveis de ensino, do fundamental ao superior. Todavia, a principal carência desses OA está na ausência de uma maneira de utilizar o erro do aprendiz como uma forma eficiente de mecanismo reparador e de aquisição de conhecimento.

Assim sendo, no processo de aquisição de conhecimento, mediado ou não pelo computador, é inevitável que o aprendiz cometa alguns erros. Porém, a partir dos erros é possível que o aprendiz amplie ainda mais seu conhecimento. Exemplos disso podem ser encontrados nos ambientes de aprendizagem baseados na arquitetura ACT* [5]. Nesses ambientes, um erro cometido leva o aprendiz a refletir e entender melhor suas ações e conceituações. A partir de uma abordagem sistemática, o ambiente consegue identificar se algo está incorreto no raciocínio do aprendiz durante a resolução de um problema. Caso algo inexato seja identificado, um *feedback* imediato é mostrado ao aprendiz para que ele então possa tentar novamente e encontrar soluções corretas a partir dos erros vistos.

Desse modo, o desenvolvimento de software educacionais, geralmente é fundamentado em teorias que investigam como as pessoas aprendem e adquirem competências, além de investigar também como e porque elas cometem erros. Dentre essas teorias, as que mais se

destacam são a ACT (*Adaptive Control of Thought*), REPAIR e STEP.

A teoria ACT destina-se a uma abordagem geral sobre a cognição humana, que logo foi chamada de ACT* [5] e mais recentemente de ACT-R (*Adaptive Control of Thought–Rational*) [6], que se fundamenta em manter o aprendiz em uma linha de aprendizagem ideal, mesmo que bem restrita. Para tal, essa teoria faz uso de uma estrutura hierárquica de objetivos em conjunto com os tutores automáticos do ACT* para fornecer aproximações contínuas durante as resoluções de problemas, que geralmente são respostas a erros cometidos pelo aprendiz. Nessa teoria, a detecção de erros acontece através de regras de produção apoiadas na resolução de um problema da forma mais contextualizada possível. Dessa maneira, deve-se fornecer respostas imediatas a erros cometidos, pois quanto maior a demora em apresentar o erro, maiores as chances de o aprendiz cometer tal erro novamente.

A REPAIR [12], também é uma teoria que tenta explicar como as pessoas aprendem. Seu foco maior está em como os aprendizes cometem erros. Ela propõe que quando o aprendiz não consegue formalizar um procedimento, por consequência de um conhecimento incompleto ou esquecido, um impasse (também chamado de falha) ocorre. Então, o aprendiz pode executar diversas estratégias chamadas de ações de recuperação, ou *REPAIR*, para contornar o impasse, podendo assim gerar soluções corretas ou ainda novos impasses. Nessa teoria assume-se que primeiro aprende-se procedimentos por indução e que os impasses ocorrem por causa de tendências indutivas que são apresentadas em novos exemplos. Dessa forma, uma repetição intercalada com impasses e recuperações pode revelar o procedimento incorreto, podendo então um tutor, seja ele humano ou máquina, fornecer explicações sobre os erros. Geralmente essas explicações quando são dadas por tutores automáticos são na forma textual.

Finalmente, a teoria STEP [28] é considerada uma extensão da teoria *REPAIR*, pois também aborda temas da origem de erros do aprendiz. A principal diferença é que a teoria STEP atribui os erros do aprendiz a um processo indutivo de aprendizagem através de exemplos, onde esses exemplos são uma sequência de ações planejadas pelo tutor. Por outro lado, a teoria REPAIR atribui os erros a ações que o aprendiz inventa para tentar

contornar um impasse durante a resolução de um problema. Alguns sistemas com base nessas teorias foram desenvolvidos com o objetivo de fornecer explicações ao aprendiz, Sierra [29] é um deles, no entanto ele apenas atuava na geração de explicações textuais para os erros cometidos.

De acordo com o relatado, fica claro que essas três teorias aqui apresentadas brevemente abordam a ocorrência de erros nas soluções do aprendiz. Todavia, todas essas teorias e modelos, apesar de bem sucedidas em ambientes de sistemas tutores inteligentes (STI), se limitam a condições nas quais: (a) o *feedback* é explícito e imediato; (b) as formas de *feedback* existem apenas em formato textual; (c) onde os erros do aprendiz possam a ser inteiramente identificados.

Um dos fundamentos para a construção de sistemas tutores cognitivos é o momento do envio do *feedback*. Se o aprendiz cometer um erro, imediatamente o sistema fornece um *feedback* e exige a sua correção, o que minimiza os problemas de incerteza mas restringe o aprendiz. Mesmo assim, no balanço geral, estudos demonstraram que um *feedback* é mais efetivo quanto mais próximo do erro ele for demonstrado ao aprendiz. Isso evita que o aprendiz construa toda uma solução em cima de uma suposição incorreta, tornando a sua aprendizagem mais eficiente [4].

Mas ao fornecer mensagens imediatas a erros cometidos, deve-se ter alguns cuidados. Um dos cuidados é na montagem do texto da mensagen, que deve ser formado para forçar o aprendiz a pensar e construir uma nova resposta, não fornecendo a solução do problema diretamente. Do mesmo modo, interrupções constantes podem atrapalhar o aprendiz ao invés de ajudá-lo. Também é complexo explicar escolhas erradas sem que o aprendiz tenha terminado seu raciocínio.

Em alguns casos, se não lhe fosse enviado um *feedback* imediato, o aprendiz poderia ter compreendido o seu erro sozinho apenas com a disponibilização de mais tempo. Horas, dias e até meses, dependendo de cada um, podem motivar a retroação ao contexto do erro quando o aprendiz achar necessário, seja no momento que ele entendeu a razão de ocorrência do erro, ou apenas para ele refletir sobre esse erro. A presente dissertação de Mestrado segue a linha de pensamento educacional que advoga em favor da livre decisão

do aprendiz sobre quando abordar um conceito [19], mesmo que seja para abordar um erro.

Um exemplo disso pode ocorrer quando o aprendiz está resolvendo uma equação matemática. Em algum momento, como parte da resolução da equação, ele pode precisar realizar uma soma de números com virgulas. Porém, por ele acreditar que possui conhecimento necessário para resolver esse tipo de soma, chega a um resultado incorreto, comprometendo assim o resultado final da equação. Por exemplo, em uma soma de 12,01 + 3 o aprendiz pode chegar ao resultado incorreto de 12,04 em vez de 15,01. Dessa forma, pode ser difícil para o aprendiz reproduzir manualmente esse tipo exato de erro por repetição de passos, pois ele está relacionado com um conhecimento subentendido do aprendiz sobre somas. Adicionalmente, durante a reprodução do erro, o aprendiz pode chegar a outras respostas para a soma, ou mesmo seguir outros passos que não exigem soma de números com virgulas.

Com base na busca bibliográfica realizada, verificou-se que ainda existe uma carência de abordagens e ferramentas de software destinadas a este propósito específico, o qual torna o contexto do erro uma forma rica de aprendizagem exploratória livre (*i*.e., dos micromundos), sem a intervenção frequente das máquinas de tutoria tais como os STI. Além disso, também ressalta-se aqui a carência de enfoque da aplicação dos conceitos de Múltiplas Representações Externas (MREs) para a revelação dos erros. Por si só, essa conjugação de conceitos aponta indícios de originalidade do presente projeto de pesquisa e desenvolvimento de Mestrado.

2.2 Objetivos

2.2.1 Objetivo Geral

O objetivo geral do presente projeto de pesquisa e desenvolvimento de Mestrado se concentra na determinação de conceitos importantes do plano meta-cognitivo, assim como, na implementação de mecanismos genéricos de interação que refletem a aplicação desses conceitos. Isso é concretizado por meio da criação de um controlador genérico de acesso

a conteúdos educacionais que serve como mecanismo exploratório para um aprendiz ter acesso aos conteúdos de um domínio específico. Para servir a esse propósito, o controlador possui diversos recursos, tais como: (a) paginação para frente e para trás; (b) acesso por índice geral; (c) glossário de termos; (d) calculadora; (e) teclado virtual para a entrada dos operandos de uma expressão de maneira mais natural; (f) botão de controle do tamanho do texto; (g) bloco de anotações; (h) construtor de introdução e de enunciao; (i) menu de acesso a erros cometidos anteriormente.

2.2.2 Objetivos Específicos

Figuraram ainda como objetivos específicos os seguintes itens:

- Desenvolvimento de um protótipo de controlador interativo de aprendizagem que dê suporte a aspectos meta-cognitivos envolvidos com as tarefas típicas de aprendizagem;
- Enfatizar a importância dos aspectos referentes á retroação ao contexto de erros cometidos no passado;
- Validar e aperfeiçoar grande parte desta pesquisa por meio da construção de 4 (quatro) softwares educacionais, todos da área de matemática do ensino médio, que seguem o padrão de interatividade do controlador desenvolvido por este trabalho;
- Investigar abordagens de desenvolvimento que tornam o ensino e aprendizagem mais direcionado através do erro do aprendiz;
- Preparar bases para uma arcabouço conceitual e arquitetural para um ambiente interativo de aprendizagem votado à ideia de múltiplas representações externas que possa ser reutilizado por diversos objetos de aprendizagem;
- Disseminar o conhecimento alcançado em publicações da área de Informática na Educação;
- Disponibilizar o referido arcabouço na forma de Software Livre para que outras pesquisas também venham a contribuir com o domínio em questão.

2.3 Contexto do projeto

O presente trabalho encontra-se inserido no projeto CONDIGITAL do grupo do estado do Paraná. Este grupo é financiado pelo Fundo Nacional de Desenvolvimento da Educação (FNDE) por meio do **Edital 001/07 MEC/MCT**. O projeto é uma iniciativa da Secretaria de Educação a Distância (SEED) do Ministério da Educação (MEC) e do Ministério da Ciência e Tecnologia (MCT).

Na Universidade Federal do Paraná (UFPR), este projeto é realizado pelo C3SL (Centro de Computação Científica e Software Livre) em parceria com o Instituto de Tecnologia para o Desenvolvimento (LACTEC), o Centro de Excelência em Tecnologia Educacional do Paraná (CETEPAR) e a Universidade Estadual de Londrina (UEL). Um de seus objetivos principais é de contribuir para a melhoria e a modernização dos processos de ensino e aprendizagem da área de Matemática na rede de escolas públicas (e mesmo privadas).

Neste projeto estão sendo desenvolvidos 4 (quatro) OA para apoiar o ensino de matemática do nível médio. Os OA devem ser utilizados apoiar atividades laboratoriais nos seguintes domínios: (a) funções de primeiro grau[1] com exemplos sobre o coeficiente de elasticidade de molas e composições de roldanas móveis; (b) progressões geométricas[2] que ocorrem em elementos da geometria fractal; (c) funções cíclicas ou trigonométricas[3], com exemplos sobre projeções de sombras e movimento de planetas no espaço; (d) matemática financeira[4], com exemplos de jogos e desafios dependentes do cálculo de juros simples e compostos.

Cada um desses domínios compõe um OA. A abordagem arquitetural de todos eles é baseada na existência de um núcleo comum de software, que é um arcabouço genérico chamado de Controlador de Acesso Reflexivo e Retroativo Indexado por Erros (CARRIE). O presente trabalho de pesquisa e desenvolvimento destinou-se prioritariamente

[1]Disponível em http://condigital.c3sl.ufpr.br/linear
Código fonte em http://git.c3sl.ufpr.br/gitweb?p=condigital/linear.git
[2]Disponível em http://condigital.c3sl.ufpr.br/fractal
Código fonte em http://git.c3sl.ufpr.br/gitweb?p=condigital/fractal.git
[3]Disponível em http://condigital.c3sl.ufpr.br/periodic
Código fonte em http://git.c3sl.ufpr.br/gitweb?p=condigital/periodic.git
[4]Disponível em http://condigital.c3sl.ufpr.br/finance
Código fonte em http://git.c3sl.ufpr.br/gitweb?p=condigital/finance.git

(mas não exclusivamente) ao desenvolvimento desse núcleo comum. O referido arcabouço genérico de acesso pode ser aplicado em diversos OA de maneira reutilizável e reconfigurável, visando fornecer uma maneira facilitada de desenvolver ambientes exploratórios de aprendizagem que torne fácil retroagir aos contextos do software onde um erro de solução de problema ocorreu. Cabe ainda observar que o controlador de acesso tem mecanismos genéricos o suficiente para ser utilizado não apenas em Matemática mas também no apoio ao ensino e a aprendizagem de conceitos em áreas como Física e Química.

Dessa forma, as principais contribuições deste trabalho para o projeto principal CONDIGITAL são: (a) fornecer uma arquitetura genérica de Objetos de Aprendizagem; (b) fornecer acesso padronizado a conteúdos educacionais por meio de tarefas reflexivas sobre domínios específicos (incluindo a reflexão sobre erros cometidos no passado); (c) fornecer métricas de desenvolvimento que facilitem a criação de OA. Todas essas contribuições foram alcançadas através do desenvolvimento do CARRIE.

CAPÍTULO 3

RESENHA LITERÁRIA

3.1 Ambientes Exploratórios para a Aprendizagem

Os ambientes exploratórios para a aprendizagem consistem em ferramentas de auto-estudo. Apesar de não interagirem pró-ativamente com o aprendiz, esses sistemas mostram-se semanticamente ricos, pois oferecem diversos recursos, os quais estão geralmente representados em formatos variados para aumentar a chance de assimilação do aprendiz por meio de metáforas computacionais bem construídas. Neles a aquisição de conhecimento está voltada a proporcionar atividades de exploração, investigação e descoberta para que o aprendiz construa individualmente seu conhecimento.

Um exemplo clássico de ambiente exploratório é o LOGO [19]. Nele, o aprendiz recebe o controle sobre o seu aprendizado, cabendo a ele decidir o quê e como aprender. Dessa forma, a responsabilidade do professor é transferida para o aprendiz, pois ele tem o direito de explorar o ambiente como quiser e não de uma maneira pré-definida. Outro aspecto importante no LOGO é que o erro não é considerado algo negativo, mas sim uma forma de aprendizagem. Com isso, diante do erro, o aprendiz tem a oportunidade de compreender um conceito incorreto durante a resolução do problema em foco. Dessa maneira, o conhecimento pode ser construído de uma forma sequencial, de acordo com as experiências realizadas no passado.

O desenvolvimento de qualquer software é uma tarefa complexa. Em se tratando de software educacional, ela torna-se ainda mais difícil por exigir equipe multidisciplinar. Nos ambientes exploratórios de aprendizagem, essa dificuldade é ainda maior, principalmente na hora da criação dos aspectos interativos controlados pelo aprendiz. Se eles não forem bem projetados do ponto de vista pedagógico, podem fazer com que o software tenha pouco valor educacional [16]. Tudo isso acaba caracterizando problemas de baixa eficiência [3] do ponto de vista de apoio à aprendizagem. Para que tais situações não aconteçam, deve

existir um equilíbrio entre o potencial educacional e a quantidade de recursos de acesso ao conteúdo do software.

Dessa forma, Santos [26] sugere algumas diretrizes para a construção de interfaces educacionais, as quais são: (a) empregar uma abordagem conceitualmente rica, permitindo que o aprendiz explore e encontre soluções de problemas a partir de metáforas do mundo real; (b) de posse da abordagem conceitual, integrar as tecnologias disponíveis no projeto de interface; (c) fornecer ao usuário a possibilidade de ajustar a interface a suas necessidades; (d) considerar, na medida do possível, os aspectos culturais no projeto da interface; (e) incorporar os princípios gerais do projeto de interface [7]; (f) considerar regras básicas para a formatação de telas [20].

Com base em tantos detalhes, parece razoável que muitos deles sejam embutidos em um controlador genérico de conteúdos educacionais para que sejam reutilizados de maneira facilitada. Em conjunto, controlador e conteúdo são comumente chamados de *objetos de aprendizagem* (OA). Além disso, esse controlador também deve fornecer aspectos meta-cognitivos que dão à navegação pelo conteúdo um maior potencial de reflexividade [22, 23, 24]. Um exemplo de potencial meta-cognitivo que ainda parece inexplorado em ambientes exploratórios de aprendizagem é a possibilidade de o aprendiz retroceder ao contexto de erros cometidos no passado.

De acordo com a busca bibliográfica realizada, nenhum ambiente exploratório de aprendizagem conhecido oferece mecanismos de retroação a erros cometidos no passado. É clara a situação em que o laboratório se presta bem como ferramenta virtual para a repetição exaustiva de tarefas. Da mesma forma, mais recentemente, dispositivos móveis também servem para que um erro seja repetido tantas vezes quantas sejam necessárias [30]. Todavia, parte da reprodução exata de um erro pode depender de ações inconscientes que o aprendiz realizou. Em outras palavras, o erro em si pode ter sido causado por fatores tácitos que dificultam sua detecção. Em alguns ambientes pró-ativos, como os STI, certos modelos e ferramentas de autoria foram construídos no passado para tentar lidar automaticamente com o erros do aprendiz [31]. Todavia, ainda parece muito incipiente o campo de pesquisa e desenvolvimento de ambientes exploratórios de aprendizagem capa-

zes de apoiar o aprendiz nos momentos de registrar o contexto de ocorrência de erros e de retornar a esses contextos.

3.2 Arcabouços (Shells) para Conteúdos Educacionais

As linguagens de autoria podem ser classificadas como linguagens de programação peculiares. Elas são projetadas com o objetivo de proporcionar formas claras e de rápida assimilação para a construção de software educacional. Essas formas claras, podem também ser chamadas de Múltiplas Representações Externas (MRE) [2], conhecidas por elevarem o grau de abstração e de cobertura da descrição de conceitos de um domínio específico. Com tais MREs, as linguagens de autoria são muito mais compreensíveis a um autor de material eletrônico que, em geral, também é leigo em Ciência da Computação [14].

As linguagens de autoria mais típicas são as empregadas no desenvolvimento de sistemas educacionais baseados em estruturas de hipertexto ou hipermídia. Um representante clássico dessa categoria de software é o HyperTalk [15]. O público-alvo do HyperTalk foi constituído por pessoas habitualmente chamadas de autores. Esses profissionais são capazes de identificar fragmentos simples em processos complexos (*e.g.*, didáticos) e de transformá-los em programas simplificados, chamados de "scripts". Um script é semelhante a um texto escrito em língua natural mas possui estrutura lógica que lembra a de linguagens da programação imperativa.

Criadas a partir de uma linguagem rica em MRE, as ferramentas de autoria são a implementação de compiladores ou interpretadores acoplados a ambientes de editoração. Em geral, tais ferramentas são operadas por meio de técnicas de programação visual (WYSIWYG[1]) de tal maneira que o autor não precise usar nenhum tipo de programação através de linhas de comandos. Em outras palavras, a ferramenta deve ajudar o autor a construir uma aplicação específica com recursos predominantemente gráficos. Apesar de essa abordagem ser mais interessante, ela provoca o surgimento de muitas limitações à sua aplicação [8]. Uma delas, por exemplo, é que o usuário precisa se adaptar ao projeto do sistema, tendo em vista que várias decisões já foram tomadas pelo construtor do aplicativo

[1]What You See Is What You Get

de autoria.

De forma resumida, os sistemas de autoria traduzem escolhas do autor do conteúdo enquanto um usuário de linguagens de programação toma todas as decisões e gera o código linha por linha. Como exemplos de sistemas de autoria voltados à criação de STI (Sistemas Tutores Inteligentes), pode-se citar os ambientes EON [17], RUI [13], SIMQUEST [27] e Demonstr8 [11].

O sistema RUI destina-se ao desenvolvimento de STI com o objetivo de ensinar conceitos visuais especializados. Foi desenvolvido para que um especialista em ensino de Radiologia médica auxiliado por um especialista em representação de conhecimento possa projetar e alterar com facilidade o conteúdo do STI sem a necessidade de conhecimentos aprofundados em informática. Apesar das facilidade de uso de uma das duas ferramentas de autoria do ambiente RUI por parte de Radiologistas, o artigo sobre o projeto relata dificuldades de uso da outra por causa da necessidade de introdução de sentenças em Lógica de Predicados de Primeira Ordem.

EON também é uma ferramenta de autoria para a construção de STI. Ele inclui ferramentas de autoria para a implementação de todos os modelos de tutores inteligentes, incluindo a aprendizagem, o domínio do conhecimento, estratégias de ensino e o modelo do aprendiz. Aparentemente, a modelagem do aprendiz é um diferencial do EON em relação à maioria das outras ferramentas de autoria para STI. Todavia, poucos detalhes são revelados no artigo acerca do funcionamento do mecanismo essencial da shell interpretadora do modelo do aprendiz.

O SIMQUEST por sua vez, é um sistema de autoria destinado à criação de simuladores baseados em ambientes de aprendizado. Ele permite facilmente a criação de simulações nas quais o aprendiz aprenderá através da exploração do ambiente. Apesar da facilidade que oferece para a geração de simulações, nada foi relatado sobre esse sistema que evidencie o potencial de seus aspectos meta-cognitivos que afetam o controle do usuário final (aprendiz) sobre o acesso aos conteúdos inseridos pelo autor das simulações.

Finalmente, o Demonstr8 é um software que destina-se ao desenvolvimento de STIs baseados na abordagem *Model-Tracing*. Nessa abordagem, o software educacional apenas

interfere os passos do aprendiz, durante a resolução de um problema, se ele não estiver no caminho da solução correta. Outro aspecto importante dessa ferramenta de autoria é automatizar a construção da interface, da introdução da lição; das regras de produção e do catálogo de erros sobre o domínio de aritmética.

Sendo assim, o autor do conteúdo pode criar softwares educacionais sem grandes conhecimentos em programação de computadores. Porém, um dos principais problemas enfrentados por esse ambiente é a construção do catálogo de erros, pois existe uma grande variedade de erros que o aprendiz pode cometer, tornando assim, impossível a cobertura completa de todos erros. Além disso, outro problema está em saber exatamente a resposta correta de um problema, o que acaba aumentando a complexidade de geração de *feedback* sobre erros do aprendiz.

3.3 Limitações do Objetos de Aprendizagem para Ciências Físicas e Matemáticas

Atualmente, muitos dos OAs desenvolvidos atuam de uma maneira puramente somativa quando conduzem algum processo de avaliação [18]. Em outras palavras, a finalidade desses OAs é estabelecer a porção de conhecimento adquirida por um aprendiz em um determinado domínio de especialidade. Isso possibilita ao ambiente atribuir uma qualificação que pode ser empregada como o grau de credibilidade da aprendizagem realizada. Outro objetivo da avaliação somativa é apenas de classificar os aprendizes ao término de uma fase de estudo, como por exemplo, no final de um semestre, ano, mês ou curso, de acordo com o grau de aproveitamento do aprendiz [32].

Por outro lado, são raros os OAs encontrados que possuem a capacidade de constatar se realmente os aprendizes estão alcançando os objetivos pretendidos durante o processo de aprendizagem. Os OAs com essa característica realizam avaliações contínuas, sem se preocuparem com a atribuição de nota ou grau. Ao contrário, esses OAs tentam identificar as principais dificuldades do aprendiz e para tentar auxiliá-lo em sua aprendizagem. Por esse motivo, esses OAs podem ser caracterizados como ambientes de avaliação formativa

[21, 32].

Outra característica importante da avaliação formativa, é o mecanismo que fornece *feedback* tanto para o aluno quanto para o professor. Com ele, o professor consegue identificar deficiências na maneira como ensina, para então poder aperfeiçoar seu trabalho. Já o aprendiz recebe *feedback* sobre o que ele aprendeu e do que necessita aprender, além de suas necessidades individuais e quais aspectos devem ser melhorados para obter uma aprendizagem próxima do ideal.

De forma resumida, OAs com carater de avaliação somativa buscam auxiliar o julgamento de classificação dos aprendizes no final de uma determinada aprendizagem. Já os OAs com carater formativo, acompanham todo o processo de aprendizagem fornecendo *feedback* sempre que acharem necessário. Com isso, os de carater formativo induzem a reflexão do aprendiz sobre seu aprendizado e conduzem-no a um melhor desempenho [21]. Os OAs formativos, apesar de raros, são os que possuem maior potencial de utilidade para as áreas de ciências físicas e matemáticas.

O sub-conjunto de OAs que se classificam como simuladores, em especial, podem ajudar a aquisição de conhecimento para as áreas de ciências físicas e matemáticas. Isso ocorre pois eles são projetados para proporcionar um entendimento geral de um determinado domínio. Para isso, oferecem mecanismos de interação para gerar atividades de relevância cognitiva para o aprendiz, proporcionando assim um aprendizado mais bem planejado [1].

Um dos motivos do sucesso do uso de simuladores para aquisição de conhecimento está em fornecer variações sobre determinadas situações de um domínio específico. Dessa forma, vão muito além de software que simplesmente oferecem exercícios de múltipla escolha pré-formatados para realizarem avaliação somativa. Todavia, os simuladores não são aplicados em situações direcionadas de proposta de atividades com acompanhamento da solução e correção das mesmas.

Diante disso, nenhum dos objetos educacionais publicados como produção cientifica ou tecnológica possuem solução interativa de exercícios. Muitos deles até propõem exercícios aos usuários, no entanto são incapazes de adotar qualquer reação mais elaborada diante

de erros do que apenas comunicar que algo está incorreto.

CAPÍTULO 4

ARQUITETURA FUNCIONALISTA

4.1 Controlador de Acesso Reflexivo e Retroativo Indexado por Erros (CARRIE)

Para consolidar os fundamentos da aprendizagem adotados até então no presente projeto de pesquisa e desenvolvimento, diversas ferramentas de software educacional estão sendo implementadas e validadas no ambientes escolar. A abordagem arquitetural de todas elas é baseada na existência de um núcleo comum de software, o qual é um arcabouço chamado aqui de Controlador de Acesso Reflexivo e Retroativo Indexado por Erros (CARRIE).

4.1.1 Organização funcionalista do CARRIE

Conforme representado na Figura 4.1, quatro módulos principais compõem a arquitetura funcionalista do CARRIE, são eles: Módulo de acesso ao conteúdo, Módulo indexador de erros, Módulo Facilitador e a Interface com o aprendiz. Todos esses módulos serão detalhados na sequência. Para obter maior detalhamento de como utilizar esses módulos verifique o anexo C

Já foram implementados, completamente, três OA que fazem uso do arcabouço CARRIE, sendo o primeiro para o domínio de Progressões Geométricas em Fractais, o segundo para o domínio de Funções de Primeiro Grau e o terceiro para domínio da Matemática Financeira. Atualmente, diversos esforços têm sido realizados para a implementação de mais um OA que destina-se ao domínio da funções periódicas que também faz uso do CARRIE. Um total de quatro software educacionais farão parte da fase inicial de construção e aplicação de apoio computacional a atividades laboratoriais do ensino escolar de conceitos matemáticos do nível médio.

Para a implementação do CARRIE, foi utilizada a plataforma Java seguindo as técnicas

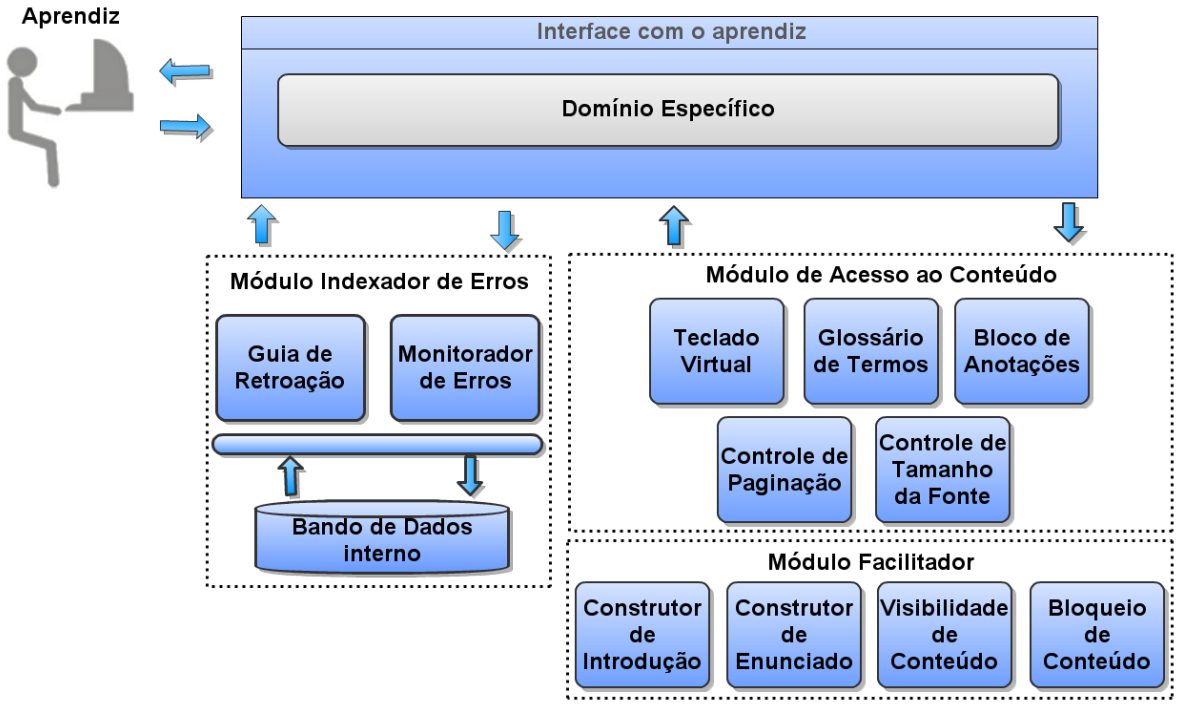

Figura 4.1: Arquitetura Funcionalista do CARRIE

do paradigma de programação Orientada a Objetos. O software foi projetado para ser executado em qualquer navegador *Web*, independente de sistema operacional ou hardware. Além disso, o código é divulgado como código livre sob licença GPL. Seu código fonte se encontra-se disponível no repositório *git*[1] da UFPR.

4.1.2 Módulo indexador de erros

Este módulo está dividido em 2 partes principais, sendo uma delas o monitorador de erros e a outra, o guia de retroação.

4.1.2.1 Monitorador de Erros

O monitorador de erros funciona com base em pontos de observação definidos pelo autor do conteúdo. Mais precisamente, o autor precisa necessariamente marcar[2], através da chamada de um método, onde o aprendiz comete um erro. Os parâmetros desse método

[1] *http://git.c3sl.ufpr.br/gitweb?p=condigital/shellcontroller.git*

[2] No anexo B pode ser visualizado um exemplo de código para marcar onde o aprendiz cometeu um erro

são o título do erro, o texto explicativo do erro e uma variável booleana. Quando essa variável for verdadeira, uma janela *popup* será mostrada, alertando o aprendiz para o erro. A área visual do *popup* na tela tem como conteúdo o título e o texto explicativo passado como parâmetro para o método. Caso seja falsa, nada será mostrado.

No momento que esse método é chamado, o monitorador de erros recebe uma mensagem indicando um desvio conceitual cometido pelo aprendiz. Após o recebimento dessa mensagem, o monitorador salva o erro e o estado da aplicação em que o erro ocorreu em um banco de dados interno. Com essas informações salvas, o CARRIE estrutura o seu comportamento para que o aprendiz possa retroceder ao momento exato em que o erro foi cometido. Sendo assim, é apresentado ao aprendiz um guia de retroação, por meio do qual o aprendiz pode retroceder a qualquer erro cometido durante seu aprendizado.

Para que o estado da aplicação seja salvo, quando o aprendiz cometer um erro, é necessário que o autor do domínio específico também defina, na classe principal de cada exercício, os pontos do código que sofrem alterações [3]. Em um primeiro momento, o salvamento do estado da aplicação era feito automaticamente, sem nenhum tipo de configuração além da definição dos pontos de observação. Porém, isso deixou a aplicação bastante lenta cada vez que um erro era cometido, pois era necessário salvar todos os dados da aplicação.

Uma tentativa de solução para esse problema foi o uso de *threads*. Porém, devido ao fato do CARRIE ser desenvolvido em Java *Swing*, o mesmo objeto que se encontra em processo de arquivamento também pode estar disponível ao uso por parte do aprendiz. Tal situação leva à possibilidade do objeto sofrer novas alterações durante o seu próprio arquivamento. Esse fato, acabava ocasionando diversos erros na aplicação, comprometendo a execução da mesma. Por esse motivo, o uso das *threads* foi descartado.

Para contornar o problema de demora durante o arquivamento do objeto, a solução encontrada foi deixar a critério do autor do conteúdo definir quais partes do código sofrem alterações com a interação do aprendiz. Dessa forma, conseguiu-se otimizar o salvamento do erro, pois só o que realmente era necessário era salvo, tornando-o imperceptível ao

[3] O anexo B.2 apresenta um exemplo de configuração de pontos do código que podem sofrem alterações com a interação do usuário

aprendiz.

4.1.2.2 Guia de Retroação

O guia de retroação (Figura 4.2) permite que o aprendiz inspecione momentos exatos de erros passados e decida a quais deles vale a pena retroceder. Este guia faz a leitura do banco de dados interno de erros e cria um menu contendo um *link* para cada erro armazenado no banco de dados. Esse menu é mostrado ao aprendiz só depois de existir pelo menos um erro cometido. Quando o menu está visível, o aprendiz pode visualizar o título do erro juntamente com a data e horário e o seu texto explicativo. Isso pode ser atingido apenas passando-se o *mouse* sobre o *link* do erro. A qualquer momento pode-se retroceder ao erro cometido apenas clicando no *link*.

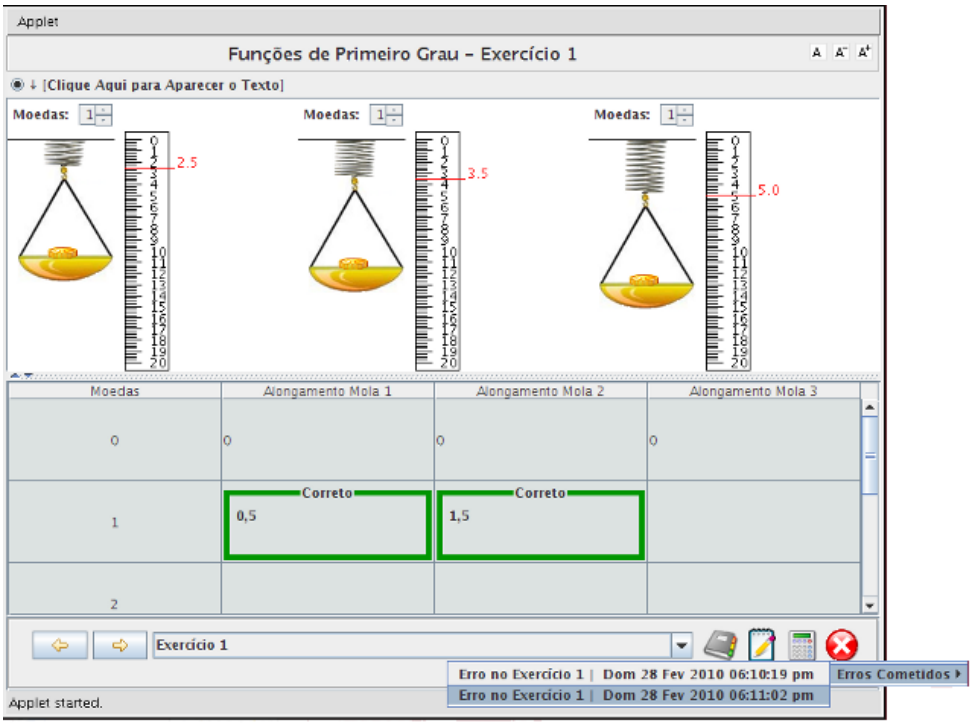

Figura 4.2: Guia de Retroação.

Quando o aprendiz retrocede a um erro, o CARRIE possibilita que ele refaça todo o exercício em que o erro foi cometido. Tal abertura de tarefas promove fundamentalmente o que é chamado de atividade reflexiva e permite que o aprendiz tente revisitar os aspectos que o levaram a esse erro. Caso o aprendiz não queira refazer todo o exercício novamente

(*e.g.,* por sentir que ainda tem dúvidas), ele também pode desistir da tentativa. Vale a pena ressaltar aqui que, ao contrário dos tradicionais recursos de desfazer simplesmente uma tarefa (*undo*, em inglês), o guia de retroação atinge um potencial meta-cognitivo muito mais adequado à finalidade pedagógica do CARRIE. Acredita-se que essa característica lhe confere um bom grau de originalidade em relação aos ambientes interativos de aprendizagem existentes até o presente momento.

4.1.3 Módulo de acesso ao conteúdo

É um conjunto de módulos responsáveis por apresentar o conteúdo ao aprendiz. Ele está dividido nos seguintes sub-módulos:

4.1.3.1 Glossário de Termos

O CARRIE oferece um módulo para a criação do glossário de termos do domínio de uma maneira simples e clara. Quando o autor achar necessário que um glossário de termos seja disponibilizado ao aprendiz, basta definir um arquivo respeitando uma indentação pré-estabelecida, um exemplo disso pode ser visualizado no anexo A. Quando o CARRIE identificar que um arquivo que segue essa estrutura foi criado, ele automaticamente disponibilizará a interface responsável pela apresentação do glossário por meio do *link* de acesso. Além disso, o glossário ainda conta com uma busca por palavra-chave baseada no recurso de *auto-completar*. A Figura 4.3 apresenta uma imagem da janela gráfica do glossário de termos do OA construído para o domínio de Funções de Primeiro Grau usando o CARRIE.

4.1.3.2 Controle de Paginação

O CARRIE tem como um dos objetivos fazer com que o aprendiz desenvolva seu conhecimento passo-a-passo para atingir abstração por generalização. Para que isso ocorra, há um módulo que é responsável por oferecer paginação de todo e qualquer conteúdo a ser apresentado. Em outras palavras, o autor de material eletrônico não precisa se preocupar

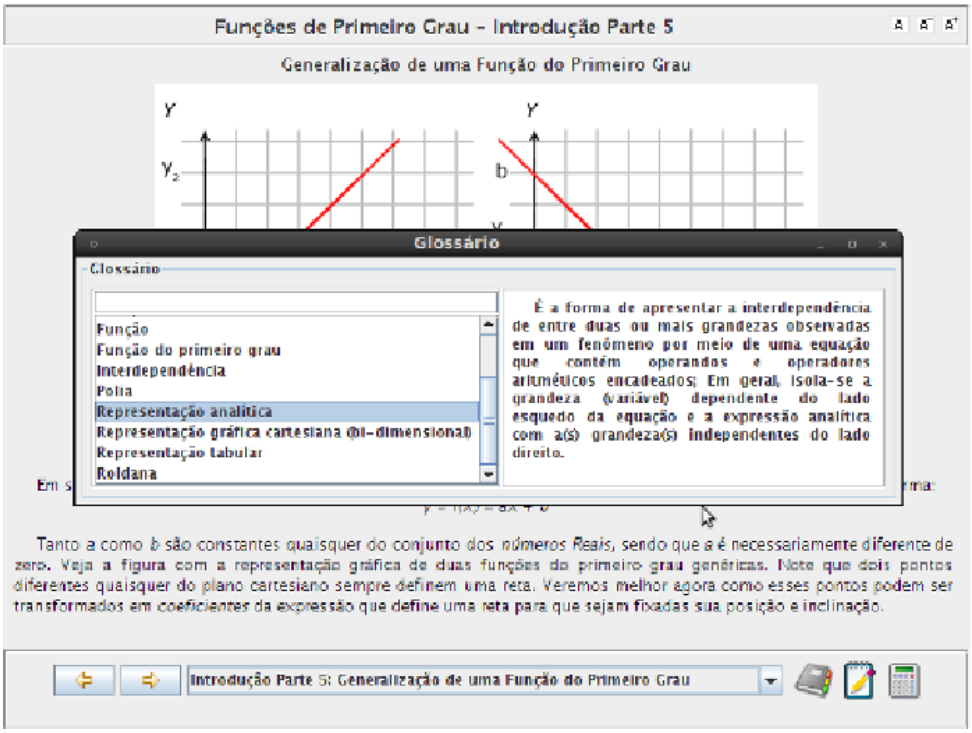

Figura 4.3: Glossário de termos.

com esse tipo de código.

4.1.3.3 Controle de Tamanho da Fonte

Um aspecto importante para os software atuais é a acessibilidade. Devido a isso, o CARRIE oferece um controle para que o aprendiz atue sobre o tamanho de letras de qualquer texto do OA.

4.1.3.4 Teclado Virtual

Alguns OAs necessitam manipular uma entrada de dados do tipo numérico-analítica. Para isso, é necessário um estilo diferente de mecanismo de interação. Esse mecanismo é um teclado virtual, que é disponibilizado pelo CARRIE para o autor do conteúdo fazer uso em seu OA. Ele está ilustrado na Figura 4. O teclado virtual usa a metáfora de uma calculadora científica estendida, onde é possível a entrada de expressões mais complexas com a visualização bidimensional imediata da mesma. No caso de todos os OA deste padrão, é possível a entrada de expressões em função das variáveis l e n, usando, além das

quatro operações normais e parênteses, potenciação e radiciação.

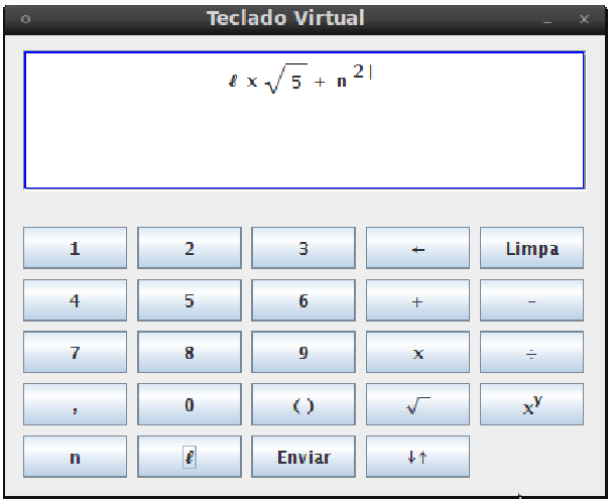

Figura 4.4: Teclado Virtual para a entrada de dados numérico-analíticos

4.1.3.5 Bloco de anotações

Um aspecto que pode ser considerado importante para o aprendizado é o bloco de anotações. Nele o aprendiz pode realizar anotações de todo o conteúdo que achar importante e, dessa forma, pode construir sua própria percepção do tema em foco. Além disso, um bloco de anotações pode facilitar a organização dos conteúdos que merecem mais atenção por parte do aprendiz. A partir disso, o CARRIE oferece um bloco de anotações ao aprendiz com recursos para destacar o texto em diversas cores.

4.1.3.6 Calculadora

Outro recurso importante para os software educacionais voltados às áreas de ciências físicas e matemáticas é a calculadora. Com isso, o CARRIE oferece uma calculadora para o aprendiz. Ela pode ser usada de maneira clara e simples.

4.1.4 Módulo Facilitador

Os sub-módulos contidos no módulo facilitador são destinados a tornar o desenvolvimento de um OA, com o uso do CARRIE, mais simples e com menos preocupação em codificação. Para isso, o CARRIE oferece três submódulos: (a) Construtor de Introdução; (b) Construtor de Enunciado; (c) Bloqueio de conteúdo e Visibilidade de Conteúdo. Cada um desses sub-módulos está descrito em seguida.

4.1.4.1 Construtor de Introdução

De um modo geral, todos os OAs possuem, de alguma maneira, uma introdução ao conteúdo que será tratado nos exercícios. Para facilitar a construção dessa introdução, o CARRIE oferece um módulo no qual o autor do conteúdo pode construir a introdução através de uma codificação em html. Dessa forma, o autor consegue ampliar as formas de formatação de textos e imagens do que se apenas usasse código Java puro. Outro aspecto importante desse módulo é que o autor do conteúdo pode conectar palavras que estão na introdução diretamente com sua definição no glossário de termos. Mais precisamente, ele

pode definir links de palavras, que quando clicadas abrem o glossário de termos com a sua respectiva definição em foco.

4.1.4.2 Construtor de Enunciado

Da mesma maneira que o autor do conteúdo pode formatar seu texto para introdução, ele também pode formatar seu texto para o enunciado. Através do CARRIE, é possível construir um enunciado seguindo o padrão de formatação e apresentação de texto em html e também conectar palavras com sua definição no glossário. Além disso, este módulo oferece uma opção na qual o aprendiz pode esconder e mostrar o texto do enunciado a qualquer momento. Dessa forma, pode-se aproveitar melhor o espaço disponível na tela do computador para resolver o exercício.

4.1.4.3 Bloqueio e Visibilidade de um Conteúdo

Certos conteúdos podem ser bloqueados ou escondidos pelo autor para que o aprendiz os acesse somente quando for apropriado. Isso oferece controle indireto do professor sobre os alunos espalhados no laboratório por meio de um módulo que garante que um exercício mais fácil é resolvido antes de outro substancialmente mais difícil.

4.1.5 Interface com o aprendiz

Uma interface com o aprendiz é pré-estabelecida pelo CARRIE e está dividida em três partes principais. A primeira é destinada ao título do OA, juntamente com o título da página atual. Em seu canto superior direito, também estão as opções para aumentar e diminuir o tamanho das letras. A segunda parte é destinada ao domínio específico. A outra é reservada às opções da dinâmica de controle oferecida ao aprendiz. Exemplos de tais controles são: paginar para frente ou para trás, botão de acesso ao glossário, botão de acesso ao guia de retroação e o botão de acesso ao bloco de anotações.

A Figura 4.5 mostra aspectos gráficos da interface em um certo instante da interação com o OA sobre força e deslocamento de molas. No quadro estão ressaltados como as

constantes elásticas das molas influenciam os gráficos das funções de primeiro grau de cada uma delas.

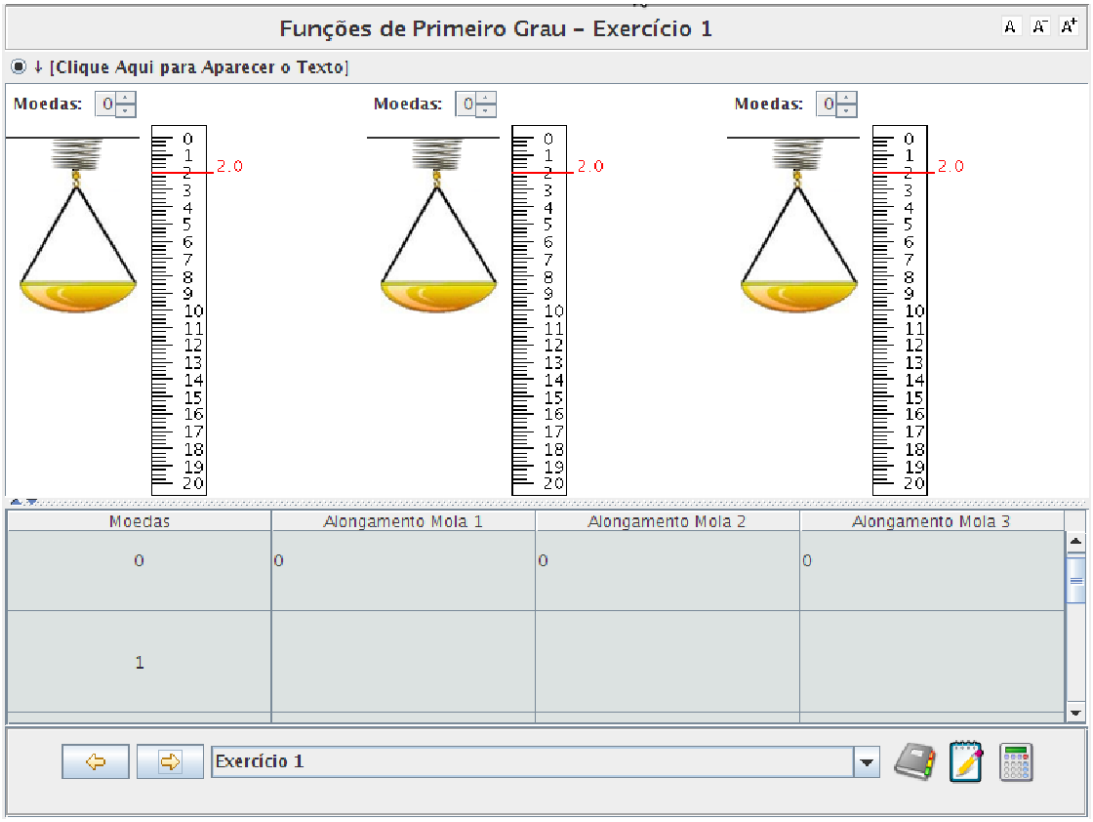

Figura 4.5: Interface do OA sobre funções de primeiro grau

CAPÍTULO 5

PERSPECTIVAS SOBRE A RETROAÇÃO A ERROS

Este capítulo tem o objetivo de mostrar as limitações do sistema de retroação ao contexto de erros oferecido pelo CARRIE, principalmente quando o seu tempo de uso torna-se muito maior do que a duração de uma aula em laboratório, como no exemplo de sua utilização por dias ou meses. Na mesma sequência de idéias, são oferecidas as possíveis soluções para contornar tais limitações visando o aperfeiçoamento desse mecanismo retroativo.

5.1 Problemas no acesso de longo prazo a erros

Atualmente o CARRIE tem o objetivo de oferecer um arcabouço para o desenvolvimento de OAs destinados à utilização em laboratórios durante o período de uma ou duas aulas de 50 minutos. Devido a isso, o registro do estado exato do erro pode ser feito apenas na memória de curto prazo. Ou seja, o aprendiz e o professor terão acesso aos erros cometidos apenas durante a execução corrente do OA. Em outras palavras, o sistema não permite que esses erros sejam acessados e analisados em uma outra sessão de execução do OA.

Consequentemente, tanto o aprendiz quanto o professor ficam limitados a um período pequeno de tempo para a análise e discussão dos erros. O que de fato seria mais interessante e produtivo do ponto de vista pedagógico é oferecer acesso aos erros a qualquer momento além do tempo de execução, como por exemplo, dias ou meses depois deles terem ocorridos. Com isso, o professor poderia reavaliar o aprendizado dos alunos, acessando seus erros, e identificar em quais partes do conteúdo apresentado o aprendiz teve mais dificuldade. Além disso, de posse dessas informações, o professor também poderia oferecer mecanismos mais direcionados para recuperar as falhas do aprendiz para que este consiga ampliar seu conhecimento de maneira mais sistemática.

Nesse mesmo sentido, o aprendiz também não ficaria limitado a rever seus erros apenas no período da execução do OA. Isso permitiria um aumento do potencial de auto-

identificação de falhas conceituais por parte do aprendiz, ou ainda com ajuda do professor. Com isso, o aprendiz poderia até se aprofundar mais do que apenas identificar as causas de seus erros. Ele teria a possibilidade de refazer, a qualquer momento, todo o exercício no qual o erro ocorreu, certificando-se assim que realmente compreendeu o conteúdo apresentado.

5.1.1 Limitações do mecanismo atual de busca de erros

Até o presente momento, o mecanismo de busca por erros cometidos que o CARRIE oferece ao aprendiz é puramente sequencial. Ele é constituído por apenas uma lista ordena pelo tempo de ocorrência do erro. A Figura 5.1 apresenta a forma como a busca por erros é conduzida no CARRIE.

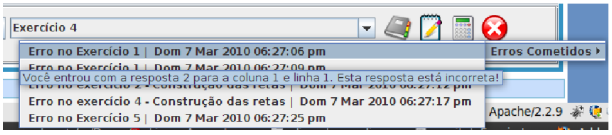

Figura 5.1: Menu do Guia de Retroação.

Essa forma de apresentação torna-se bastante limitada quando existe uma grande ocorrência de erros a ser listada para o aprendiz. Quando isso acontece, uma grande quantidade de *links* precisa ser mostrada, o que torna confusa a visualização e a busca por erros. Por exemplo, para encontrar um determinado erro, o aprendiz/professor teria que visualizar cada *link*, o que tomaria muito tempo de ambas as partes. Uma maneira de tornar a busca mais abreviada, mesmo que menos objetiva, seria oferecer um mecanismo de busca por erros no qual o aprendiz/professor apenas digitaria palavras chaves como entrada do processo. Todavia, uma nova maneira de buscar erros só pode ser implantada se a representação dos erros for expandida.

5.1.2 Limitações da representação atual de erros

A representação atual dos erros está dividida em duas partes principais: a interna e a externa. Elas têm o objetivo de proporcionar facilidades para o aprendiz adquirir percepção

sobre suas falhas. A estrutura interna, conforme apresentado no capítulo 4, é composta, até o momento, apenas pelos três seguintes itens: (a) data e hora da ocorrência do erro; (b) título do erro tal qual informado pelo autor; (c) uma breve descrição informada pelo autor.

Essa estrutura também é utilizada para a representação externa dos erros, pois essas informações podem ser vistas tanto durante a busca por um erro quanto por uma opção na janela que é aberta com a reprodução do exato estado em que o erro procurado ocorreu (*i.e.*, retroação ao erro). Porém, apesar de alguns benefícios, essa forma de estruturação ignora uma vasta gama de descrições que o aprendiz pode ter acerca de seus erros. Por exemplo, uma descrição sobre a classificação de erros em diferentes categorias significativas poderia promover maior precisão sobre o assunto abordado no momento de ocorrência desse erro. Isso parece levar a um quadro plausível de atenção do aprendiz para ele não cometer o mesmo erro novamente.

Porém, para se ampliar a linguagem de descrição externa de erros utilizada pelo aprendiz, seria necessário estender também a representação interna desses erros. Isso ocorre pois a representação atual é limitada e exclui detalhes da associação entre uma linguagem mais expressiva de interface com o aprendiz e uma taxonomia interna de categorização para guiar as intervenções da máquina.

5.2 Perspectiva de expansão da representação de erros

Um dos principais gargalos que torna complexa a ampliação do potencial do software para acompanhar o aprendiz no longo prazo está na representação interna dos erros. Esse mesmo gargalo também dificulta o aperfeiçoamento do mecanismo de acesso a erros cometidos no passado durante a solução de problemas.

Com isso, para eliminar esse gargalo, seria necessário uma expansão da atual representação interna de erros. Dessa maneira, além das três estruturas já existentes, uma possível expansão da estrutura interna poderia ser constituída de: (a) Representação taxonômica de erros; (b) Representação taxonômica para modelagem de aprendiz.

5.2.1 Representação taxonômica de erros para visualização

Até o momento, o CARRIE é destinado a diversos domínios de especialidade matemática tais como o de progressões geométricas em fractais, funções de primeiro grau, matemática financeira e trigonometria. Esses domínios são geralmente interligados entre si. Com isso, conhecimentos adquiridos em um domínio podem ser essenciais para o aprendizado em outro. Dessa forma, em um aprendizado de longo prazo, seria interessante possuir uma taxonomia para descrever e identificar erros não apenas desses domínios mas também de conteúdos mais básicos os quais seriam fundamentais para atividades planejadas de solução de problemas nos domínios em questão.

Uma organização taxonômica para os erros matemáticos pode trazer uma vasta quantidade de benefícios, tanto para o aprendiz como para o professor. Para o aprendiz, a taxonomia poderia oferecer respostas mais precisas e imediatas sobre suas falhas no decorrer do seu aprendizado com um software educacional. Já o professor, a partir dela, poderia realizar análises sobre quais pontos de determinado conteúdo os aprendizes de um turma estão com mais dificuldades, por exemplo. Com isso, seria possível um aperfeiçoamento mais conciso de suas metodologias de ensino. Além disso, pesquisadores da área poderiam determinar, para uma escola, cidade e até mesmo um estado, em quais assuntos da área de Matemática os aprendizes estão com mais dificuldades. Tal quadro de análise também é, de fato, plausível se for assumido que os OA utilizados por esses aprendizes fazem uso do CARRIE com seu conteúdo de longo prazo sendo composto por uma taxonomia de erros.

A Figura 5.2 apresenta uma proposta de uma taxonomia para descrição e classificação dos erros cometidos pelo aprendiz. Apesar de, nesta figura, ela ser exposta de maneira limitada, pois apresenta apenas alguns nodos, seria relativamente fácil expandi-la com o apoio de especialistas em pedagogia do domínio em foco. O uso dessa taxonomia durante a busca por uma descrição e classificação de um erro tornaria possível construir manualmente mensagens de explicações (*feedback*) com algum grau de precisão. Por exemplo, o grau de precisão poderia ser constituído três camadas: (a) genérico, no caso do erro ser

classificado por um nodo raiz; (b) intermediário, em situações onde o erro está catalogado por nodos do nível médio da hierarquia taxonômica; (c) específico, expondo descrições bem granulares sobre o erro, em condições onde é possível o erro ser qualificado para residir nos nodos folha.

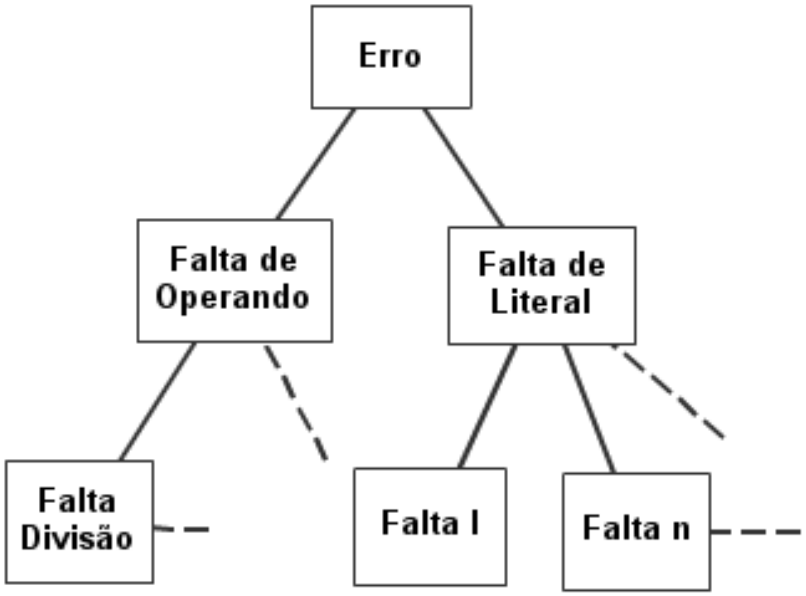

Figura 5.2: Exemplo de uma Taxonomia de Erros

Um aspecto importante para a taxonomia servir corretamente é a identificação de possíveis erros que podem ser catalogados por meio de pontos de observação do conteúdo em que o autor codificou no software. Por exemplo, através de uma linguagem pré definida, o autor entraria com ”falta de l”em uma expressão de resposta sobre o tamanho do lado de um fractal em uma dada iteração. Com isso, o CARRIE saberia que nesse ponto o aprendiz entrará com uma resposta e um possível erro para ela seria a falta do literal ”l”na composição da expressão.

Outro aspecto que deve-se levar em consideração é que uma resposta correta pode ter inúmeras variações. Com isso, seria fundamental para o uso da taxonomia que o CARRIE possuísse um identificador de erros capaz de analisar uma resposta e afirmar com precisão se ela é realmente incorreta. Ou ainda, se a expressão é correta mas têm algumas características de uma resposta que pode ser mais simplificada. Por exemplo, uma resposta considerada correta poderia ser $\frac{l}{n^2}$ a qual também poderia ser escrita como

$l.(n.n)^{-1}$.

Como exemplo para o uso dessa taxonomia, pode-se tomar por base o OA desenvolvido com o CARRIE para o ensino de PG em fractais (Figura 5.3).

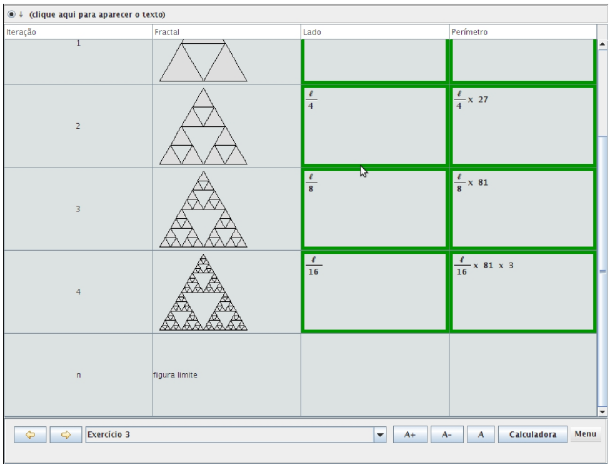

Figura 5.3: Exercício do Objeto de Aprendizagem PG em Fractais

Nessa figura, é apresentado um dos exercícios que o software educacional propõe aos aprendizes. Nesse exercício, é necessário que o aprendiz complete uma tabela com uma gama de expressões analíticas, as quais representam propriedades do fractal conhecido como Triângulo de Sierpinsky. Cada linha da tabela representa uma iteração da construção do fractal. O conjunto ordenado de todas a linhas, com exceção da última, forma uma PG. Dessa forma, é pedido que aprendiz preencha a última linha com o termo geral da PG formada, que também pode ser chamada de generalização das expressões dos termos preenchidos nas linhas anteriores.

Certamente, o aprendiz pode cometer os mais variados tipos de erros em qualquer uma das células de respostas do exercício. Todavia, para ressaltar mais a importância dessa última linha da terceira coluna, note-se que a resposta correta esperada deveria ser $\frac{l}{2^n}$. Porém o aprendiz pode preencher respostas como $\frac{l}{2}$ ou $l \times n$, entre outras, as quais seriam todas consideradas erradas.

Com o apoio da taxonomia, esses erros poderiam ser pré-classificados manualmente e descritos com suas explicações. Com isso, seria possível retornar mensagens mais precisas sobre o erro. Por exemplo, no erro $\frac{l}{2}$, caracterizado pelo nodo ”falta o literal n”, a

mensagem poderia ser a seguinte:

- *Na composição da expressão de resposta, é esperado que a variável n apareça*

Já para o erro $l \times n$, qualificado pelo nodo "falta de divisão", a explicação seria a seguinte:

- *Você notou que existe uma razão entre os elementos anteriores dessa coluna? Tente adicionar essa razão.*

Dessa forma, poderiam ser enviadas ao aprendiz mensagens mais específicas sobre os erros que ele cometeu. Além disso, o autor do conteúdo terá menos preocupação com a classificação e descrição dos erros.

Além dessas mensagens mais específicas, mensagens mais genéricas poderiam ser utilizadas como explicações. De acordo com a hierarquia taxonômica, as mais genéricas seriam caracterizadas pelos nodos mais acima. Por exemplo, para o erro $\frac{l}{2}$, uma mensagem mais genérica seria *Na expressão preenchida está faltando a presença de um literal.* Porém, não pode-se afirmar qual das explicações, mais genérica ou mais específica, seria mais apropriada para que o aprendiz compreendesse a razão pela qual cometeu o erro. Mas, de uma forma empírica, pode-se supor que uma informação mais específica sobre o erro é mais apropriada para a compreensão do mesmo, principalmente quando o contexto temporal desse erro está próximo. No entanto, para comprovar isso, seria necessário um estudo mais aprofundado por meio de coleta e análise estatística de dados operacionais em ambientes reais da prática pedagógica.

Finalmente, além dos benefícios supra-citados que uma taxonomia de erros pode oferecer, ela também poderia trazer vantagens para a construção de uma nova estrutura externa. Tal estrutura diz respeito a como um erro será percebido pelo aprendiz quando acusado por uma explicação. Com a taxonomia sendo também utilizada como uma representação externa, o aprendiz poderia ter uma nova forma de visualização de seus erros, o que facilitaria sua compreensão sobre eles para que pudessem ser descritos pelo aprendiz mais tarde, caso quisesse retroagir ao contexto de ocorrência do mesmo.

Como uma forma de ampliar ainda mais a compreensão do aprendiz, essas mensagens de erros ainda poderiam ser personalizadas pelo autor do conteúdo. Através de uma linguagem pré-estabelecida, o autor poderia identificar os nodos e atribuir a eles mensagens mais elaboradas, de acordo com domínio que será apresentado. Os desdobramentos desta idéia serão discutidos na subseção que segue.

5.2.2 Representação taxonômica para modelagem de aprendiz

A modelagem do aprendiz poder ser definida como um sistema que realiza aquisição e gerenciamento de informações sobre os aprendizes. Como o CARRIE registra os momentos exatos onde os aprendizes cometeram erros, pode-se dizer que fica registrada uma grande quantidade de dados sobre o processo de aprendizagem do aprendiz. E a partir desses dados, seria possível compor informações relevantes para a correspondência entre o modelo do aprendiz e o que seria mais adequado para esse aprendiz tentar como solução de problema no passo seguinte.

Com isso, o CARRIE poderia ir além e, a partir da taxonomia de erros, ele seria capaz de criar uma descrição bastante completa do perfil do aprendiz. Além disso, ele também teria a capacidade de apontar o perfil cognitivo do aprendiz e consequentemente, oferecer estratégias de ensino e/ou de aprendizagem a serem utilizadas por professores e alunos. Dessa maneira, o autor do conteúdo também poderia se beneficiar dessas característica para personalizar ações que o software deve tomar durante um determinado processo de aprendizagem.

5.3 Perspectiva de expansão do mecanismo de busca de erros

De posse de tantas novas características, seria essencial expandir o mecanismo de busca de erros oferecido pelo CARRIE. Essa expansão poderia ser composta de dois módulos principais: (a) busca através de linguagem natural; (b) busca através de uma hierarquia taxonômica de erros.

5.3.1 Busca com descrição em língua natural

Essa busca seria realizada a partir de um *string* inserido pelo aprendiz ou pelo professor. De posse desse *string*, o sistema poderia realizar diversas comparações com base em uma estrutura interna composta por todos os dados dos erros cometidos. Com isso, várias heurísticas de busca poderiam ser implementadas e testadas.

As heurísticas permitiriam que o motor de busca do CARRIE consiguisse reconhecer alterações na maneira em que os aprendizes realizassem suas buscas por meio de combinações de termos e palavras-chaves. Dessa forma, o sistema seria capaz de identificar tais variações e apresentar com mais precisão, e em ordem de relevância, os erros encontrados tal qual foram cometidos no passado. É importante notar que a tecnologia para esse mecanismo não é mais uma barreira difícil de ser transposta pois atualmente, até mesmo sob a forma de software livre estão disponíveis os códigos de diversas máquina de busca textual que operam na Web.

5.3.2 Busca com descrição taxonômica

Como forma complementar à textual, a busca guiada pela descrição taxonômica seria ainda mais avançada por combinar as informações estrutural (da hierarquia explícita) e textual (em língua natural). A partir dela, o aprendiz/professor seria capaz de visualizar todos os erros caracterizados pela taxonomia, organizando-os por grupos. Para isso, os modos de entrada dos argumentos de busca seriam compostos pelos erros caracterizados por nodos dos diversos níveis da hierarquia taxonômica. Além disso, o aprendiz/professor também seria capaz de visualizar os erros de qualquer nodo da taxonomia de forma isolada.

Outra característica oferecida seria a busca através de uma breve descrição do erro. Mais especificamente, o aprendiz poderia compor a descrição de um erro ocorrido no passado e, a partir dela, o sistema realizaria uma busca e apresentaria ao aprendiz os erros relacionados com a descrição. Uma combinação dessa busca com a busca em língua natural ofereceria um poder expressivo diferenciado. Com tais recursos, o aprendiz/professor seria capaz de fornecer uma palavra chave em todos, em alguns ou ainda em apenas um nodo

da hierarquia taxonômica.

Advoga-se aqui que essas maneiras de busca podem trazer diversos benefícios ao aprendiz e também ao professor. Um dos prováveis efeitos positivos seria a diminuição do tempo para encontrar determinados erros. Um outro tem a ver com a maior precisão na na especificação da busca, além de um melhor entendimento do erro em si.

CAPÍTULO 6

CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

6.1 Reafirmação da Contribuição

Neste projeto de pesquisa e desenvolvimento foram apresentadas tanto a necessidade como a proposta de um controlador de interatividade genérico que evidencia os aspectos metacognitivos envolvidos em tarefas típicas de aprendizagem. Além disso, foi evidenciada como uma das principais carências dos OA, a ausência de uma forma de utilizar o erro do aprendiz como uma maneira eficiente de mecanismo reparador e de aquisição de conhecimento. Ressaltou-se também, neste documento, a ausência de pesquisas no que diz respeito a este nicho específico de tratamento de contextos de erro por parte do próprio aprendiz humano.

Destacou-se na resenha literária os aspectos concernentes a Ambientes Exploratórios de Aprendizagem assim como às diretrizes fundamentais para o desenvolvimento de interfaces educacionais. Coube também a esta parte elucidar os aspectos que dizem respeito às linguagens e ferramentas de autoria. Finalmente, ainda na resenha enfocou-se as principais limitações dos objetos de aprendizagem para ciências físicas e matemáticas.

Para consolidar os fundamentos de aprendizagem adotados no presente projeto de pesquisa e desenvolvimento apresentou-se, no capítulo 4 uma abordagem arquitetural caracterizada por um núcleo comum para o desenvolvimento de diversos software educacionais. O arcabouço é chamado de Controlador de Acesso Reflexivo e Retroativo Indexado por Erros (CARRIE).

Depois disso, descreveu-se cada módulo pertencente a esta arquitetura, a qual dá suporte à abordagem deste trabalho. Permeando a elucidação destes módulos, encontram-se detalhes do funcionamento deles no protótipo, o qual instancia e valida tal arquitetura.

Por fim, destacou-se as perspectivas sobre a retroação a erros do CARRIE, apresentando os problemas no acesso de longo prazo a erros, juntamente com as limitações do

mecanismo atual de busca textual por erros. Além disso, foram expostas as limitações da representação atual dos erros. Nessa mesma sequência de idéias, foram oferecidas possíveis soluções para contornar tais limitações visando o aperfeiçoamento desse mecanismo retroativo.

Crê-se que os objetivos cumpridos efetivam uma contribuição pertinente ao domínio tratado. Apesar das limitações expostas, acredita-se que o presente trabalho tem grande potencial para enriquecer as pesquisa sobre o assunto em foco além de aperfeiçoar o ensino para as ciências matemáticas e físicas. Finalmente, espera-se que este trabalho de pesquisa e desenvolvimento possa inspirar trabalhos prósperos para a área abordada.

6.2 Trabalhos Futuros

De acordo com resultados obtidos por este projeto de pesquisa e desenvolvimento, as perspectivas futuras apontam naturalmente para um maior aprofundamento das formas de retroação às situações onde o aprendiz cometeu erros. A ideia de registrar e restaurar os quadros de erro ainda é feita de maneira quase linear no arcabouço atual do controlador CARRIE. Adicionalmente, a iniciativa mais próxima de pesquisas do passado que ofereceram ao aprendiz uma visão de ambientes de aprendizagem com recursos para os usuários inspecionarem o que o software assumiu sobre eles foi chamada de modelos abertos de aprendizes (*open student models*) [33]. Nessa categoria de sistemas tutores, o conceito de *skillometer* [10] como marcador de valor em uma escala de habilidade foi uma das formas com que os modelos abertos de aprendizes mais se projetaram no mundo de pesquisa.

No entanto, tais iniciativas não contemplaram a criação e o uso de linguagens de descrição da configuração com que os erros do aprendiz ocorreram para que tais erros pudessem ser revisados em outras condições no futuro. Essa fronteira de pesquisa continua totalmente inexplorada e se constitui em um campo relevante de interesse teórico e prático. Seus conteúdos cobrem desde aspectos epistemológicos da representação do conhecimento humano sobre os estados do mundo até detalhes de implementação que referenciam o chamado cálculo de situações no clássico mundo de blocos [25]. Por si só, a abordagem integradora de tais conteúdos sob uma linguagem única representa um grande desafio de

pesquisa cuja complexidade certamente contribuirá com conhecimentos originais para a Informática na Educação.

Além dos trabalhos futuros supracitados e dos citados no capítulo anterior podemos destacar as seguintes atividades de pesquisa que seguem o atual estágio deste trabalho:

- Melhorar a forma de configuração necessária, por parte do autor do conteúdo, para que o CARRIE possa salvar o estado da aplicação quando o aprendiz cometer um erro. Para isso, pode-se desenvolver uma maneira na qual é exigida menos ou nenhuma configuração por parte do autor do conteúdo;

- Ampliar o alcance do protótipo para que possa ser usado não apenas para as ciências físicas e matemáticas, mas também para outras áreas do conhecimento humano. Para isso, poderiam ser construídos módulos responsáveis por partes mais específicas de outros domínios;

- Abastecer o ambiente com uma galeria de exemplos comentados da utilização de cada módulo do CARRIE. Isto pode estimular ainda mais o autor do conteúdo na criação de seu próprio OA, além de criar uma linha de suporte para que o autor possa conduzir seu desenvolvimento;

- Introduzir um módulo que permita ao autor do conteúdo construir tanto a introdução ao conteúdo quanto os enunciados dos exercícios através de técnicas de programação visual WYSIWYG.

BIBLIOGRAFIA

[1] HORNES A., GRACHINSKI L, Sani de Carvalho Rutz da Silva, e André KOSCIANSKI. Os jogos computacionais no ensino de física. *VII Encontro Nacional de Pesquisadores em Educação em Ciências.*, 2009.

[2] S. Ainsworth. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16(3):183–198, 2006.

[3] G. Alves. Um estudo sobre o desenvolvimento da visualização geométrica com o uso do computador. *SBIE2007 - Simpósio Brasileiro de Informática na Educação*, páginas 3–12, 2007.

[4] John R. Anderson. *The Architecture of Cognition.* Harvard University Press, Cambridge, MA, USA, 1983.

[5] John R. Anderson. *The architecture of cognition / John R. Anderson.* Harvard University Press, Cambridge, Mass. :, 1983.

[6] John R. Anderson. *Rules of Mind.* Hillsdale: Erlbaum, 1993.

[7] Carol Barner. A typology for educational interfaces. EMC 503 Arizona State University Fall, 1997, 1997.

[8] L. Batista, A. Direne, J. Trindade, I. Gimenes, e O. Taha. Autoria de diretrizes pedagógicas destinadas ao treinamento das múltipla capacidades da perícia em conceitos visuais. *SBIE2003 - Simpósio Brasileiro de Informática na Educação*, páginas 573–582, 2003.

[9] BIOE/MEC. Ministério da educacão, banco internacional de objetos educacionais. http://objetoseducacionais2.mec.gov.br, 2009.

[10] S. B. Blessing. A programming by demonstration authoring tool for model-tracing tutors. *International Journal of Artificial Intelligence in Education*, 8:233–261, 1997.

[11] S. B Blessing. A programming by demonstration authoring tool for model-tracing tutors. T. Murray, S. Blessing, e S. Ainsworth, editores, *Authoring tools for advanced technology educational software: toward cost-effective production of adaptive, interactive and intelligent educational software*, páginas 93–120. Kluwer Academic Publishers, Dordrecht, 2003.

[12] J. S. Brown e K. VanLehn. Repair theory: A generative theory of bugs in procedural skills. A. Collins e E. E. Smith, editores, *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*, páginas 338–361. Kaufmann, San Mateo, CA, 1988.

[13] A.I. Direne. Designing intelligent systems for teaching visual concepts. *International Journal of Artificial Intelligence in Education*, 8(8):44–70, 1997.

[14] Celso Hartmann, Alexandre Direne, Luis Bona, Fabiano Silva, Gabriel dos Santos, André Guedes, Marcos Castilho, e Marcos Sunyé. Linguagem e ferramenta de autoria para promover o desenvolvimento de perícias em xadrez. *SBIE2005 - Simpósio Brasileiro de Informática na Educação*, páginas 656–665, 2005.

[15] Elizabeth E. Katz e Hayden S. Porter. Hypertalk as an overture to cs1. *SIGCSE Bull.*, 23(1):48–54, 1991.

[16] A. L. A. Raabe L. S. Fernandes e F. B. V. Benitti. Interface de software educacional: Desafios de design gráfico. *IV Congresso Brasileiro de Computação - CBComp 2004, Informática na Educação*, 1(3):254–258, 2004.

[17] T. Murray. Authoring knowledge based tutors: tools for content, instructional strategy, student model, and interface design. *Jnl. of the Learning Sciences*, 7(1):5–64, 1998.

[18] Rafael Z. Marchesi Márcia G. de Oliveira, Elias Oliveira. Um qasystem para interação de alunos em avaliações somativas a distância. *XX SBIE - Simpósio Brasileiro de Informática na Educação (SBIE-2009)*, Florianópolis-SC, Brasil, Novembro de 2009. SBC.

[19] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

[20] P.C.P Pavel. Sisautor, um sistema de autoria para a construção de tutores hipermídia em cardiologia. Tese de mestrado, COPPE/ UFRJ. Rio de Janeiro, Rio de Janeiro, RJ, 1995.

[21] Edson Pimentel, Arthur S. Alves1, Bruno W. R. Oliveira, Danilo M. Ikebara, Patrícia A. Bottaro, e Renato Lopes. Avaliações adaptativas baseadas no nível de aquisição de conhecimentos do aprendiz. *XVIII SBIE - Simpósio Brasileiro de Informática na Educação (SBIE-2007)*, São Paulo, Brasil, Novembro de 2007. SBC.

[22] Sadhana Puntambekar e Roland Hübscher. Tools for scaffolding students in complex learning environments: What we have gained and what we have missed? *Educational Psychologist*, 40:1–12, 2005.

[23] Sadhana Puntambekar e Agnes Stylianou. Designing metacognitive support for learning from hypertext: What factors come into play? *AIED*, 2003.

[24] Sadhana Puntambekar, Agnes Stylianou, e Roland Hübscher. Improving navigation and learning in hypertext environments with navigable concept maps. *Hum.-Comput. Interact.*, 18(4):395–428, 2003.

[25] Stuart J. Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[26] N. SANTOS. Design de interfaces de software educacional. *Livros Eletrônicos*, 2004.

[27] W.R. Joolingen van e T. Jong de. Simquest: authoring educational simulations. T. Murray, S. Blessing, e S. Ainsworth, editores, *Authoring tools for advanced technology educational software: toward cost-effective production of adaptive, interactive and intelligent educational software*, páginas 1–31. Kluwer Academic Publishers, Dordrecht, 2003.

[28] K Vanlehn. Learning one subprocedure per lesson. *Artif. Intell.*, 31(1):1–40, 1987.

[29] K Vanlehn. Learning one subprocedure per lesson. *Artif. Intell.*, 31(1):1–40, 1987.

[30] Maria Virvou e Eythimios Alepis. Mobile educational features in authoring tools for personalised tutoring. *Comput. Educ.*, 44(1):53–68, 2005.

[31] Maria Virvou e Maria Moundridou. Adding an instructor modelling component to the architecture of its authoring tools. *International Journal of Artificial Intelligence in Education*, 12:185–211, 2001.

[32] Renata Zanella. Easy : sistema de avaliações via web baseado no hyper-autonomaton. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., 2005.

[33] Diego Zapata-Rivera e Jim E. Greer. Exploring various guidance mechanisms to support interaction with inspectable learner models. *Proceedings of the Conference on Intelligent Tutoring Systems (ITS-2002)*, páginas 442–452, 2002.

ANEXO A

EXEMPLO: GLOSSÁRIO DE TERMOS

Neste anexo apresenta-se um exemplo de um arquivo formatado com as características necessárias para que o CARRIE possa realizar a criação do glossário de termos que será apresentado ao aprendiz.

A.1 Glossário utilizado software para domínio de Funções Afim e Linear

```
Função:
  Diz-se que uma variável y é uma função de outra variável x quando a
  cada valor de x corresponde, mediante uma certa lei, um ou mais
  valores de y. A lei que estabelece a correspondência entre os
  valores de x e y é que chamamos de função.

Função do primeiro grau:
  É qualquer função definida por ‘‘y = f(x) = ax + b’’, onde ‘‘a’’ e
  ‘‘b’’ são constantes quaisquer do conjunto dos números Reais e ainda
  ‘‘a’’ é necessariamente diferente de zero.

Alongamento:
  Diferença entre o comprimento que o objeto possui e o comprimento
  que o mesmo possuía antes de um fenômeno ocorrer.

Comprimento:
  Dimensão longitudinal de um objeto, de uma extremidade à outra;
  tamanho de um objeto.

Interdependência:
  Dependência mútua;
  fenômenos (ou objetos) com duas ou mais grandezas a eles associadas,
  cujos valores possuem correspondência entre si.

Coeficiente angular:
  Constante que expressa a tangente trigonométrica do ângulo que uma
  reta forma com o eixo x do plano cartesiano;
  Em uma função do primeiro grau da forma "y = f(x) = ax + b" o
  coeficiente angular é a constante "a";
```

Coeficiente linear:
Constante que possui o valor da ordenada (y) do ponto em que uma reta corta o eixo y do plano cartesiano;
Em uma função do primeiro grau da forma "y = f(x) = ax + b" o coeficiente linear é a constante "b";

Representação gráfica cartesiana (bi-dimensional):
É a forma de apresentar a interdependência entre duas grandezas observadas em um fenômeno por meio de linhas e/ou pontos dispostos no espaço bi-dimensional formado por dois eixos ortogonais;
O eixo tradicionalmente chamado de "x" marca os valores da grandeza (ou variável) independente observada no fenômeno. O eixo tradicionalmente chamado de "y" marca os valores da grandeza dependente.

Representação tabular:
É a forma de apresentar a interdependência entre duas ou mais grandezas observadas em um fenômeno por meio de uma tabela onde cada linha registra um valor de cada grandeza;
A representação tabular é exclusivamente discreta.

Representação analítica:
É a forma de apresentar a interdependência de entre duas ou mais grandezas observadas em um fenômeno por meio de uma equação que contém operandos e operadores aritméticos encadeados;
Em geral, isola-se a grandeza (variável) dependente do lado esquedo da equação e a expressão analítica com a(s) grandeza(s) independentes do lado direito.

Roldana:
Objeto circular que gira entorno de um eixo. A roldana é comumente transpassada por uma corda e serve para facilitar os atos de erguer, abaixar ou equilibrar corpos muito pesados;
Pode ser utilizada de maneira fixa ou móvel em relação ao movimento da corda (cada roldana móvel reduz pela metade a força necessária para equilibrar o peso do copo que está na outra extremidade da corda);
É também conhecida pelo nome de polia.

Polia:
Objeto circular que gira entorno de um eixo. A polia é comumente transpassada por uma corda e serve para facilitar os atos de erguer, abaixar ou equilibrar corpos muito pesados;
Pode ser utilizada de maneira fixa ou móvel em relação ao movimento da corda (cada polia móvel reduz pela metade a força necessária para equilibrar o peso do copo que está na outra extremidade da corda);
É também conhecida pelo nome de roldana.

ANEXO B

EXEMPLO: MECANISMO DE RETROAÇÃO

B.1 Código com chamada ao mecanismo de retroação a erros

O fragmento de código a seguir foi retirado do primeiro exercício do objeto de aprendizagem para o domínio do funções afim e lineares. A parte que destaca-se aqui é o código abaixo do comentário "Código necessário para salvar o estado da aplicação", que expõem o código necessário para chamada do método que realizará o arquivamento do estado da aplicação.

```
...

if (!correct){
    String idMsg = Integer.toString(tableNumber)+ Integer.toString(rowIndex)
        + Integer.toString(colIndex);
    String msg = checkError.VerifyErrorMsg(idMsg);
    if (msg!= null)
        this.errorBalloon = new TablecellBalloonTip(table,
                msg,
                rowIndex,
                colIndex,
                new EdgedBalloonStyle( new Color(235,0,0),
                new Color(0,0,0)),
                BalloonTip.Orientation.RIGHT_ABOVE,
                BalloonTip.AttachLocation.CENTER, 40, 20, true);

    keyboard.setVisible(false);

    // Código necessário para salvar o estado da aplicação
    applicationState.save("Erro no Exercício "+ tableNumber +"",
            "Você entrou com a resposta "+form+" para a coluna "+ colIndex
            +" e linha "+ rowIndex +", que está incorreta!",
            false);
}
...
```

B.2 Código de configuração de estado do mecanismo de retroação a erros

O código abaixo apresenta a configuração necessária para que o estado da aplicação seja salva. Como o código anterior ele também foi retirado com OA para o domínio do funções afim e lineares.

```
...

/**
 *  Used to configure the state of object to can save state
 *
 */
protected JPanelState configState() {
    ExerciseOne exState = new ExerciseOne();

    /* Parameters to save Spring state*/
    exState.jpSpringA.setCoinsValue(jpSpringA.getCoins());
    exState.jpSpringB.setCoinsValue(jpSpringB.getCoins());
    exState.jpSpringC.setCoinsValue(jpSpringC.getCoins());

    /* Parameters to save Enunciation state*/
    exState.exerciseEnunciantion.setEnuciationVisiable(
            exerciseEnunciantion.enunciationIsVisiable());

    /* Parameters to save splitPanel state*/
    exState.splitPane.setDividerLocation(
            splitPane.getDividerLocation());

    /* Parameters to save table state*/
    table.setValuesToTable(exState.table);

    return exState;
}

...
```

ANEXO C

ARQUIVO README DO CARRIE

Neste anexo é apresentado o arquivo *readme* do CARRIE, o qual contém a descrição de como utilizar as funcionalidades oferecidas por este protótipo, desenvolvido neste projeto de pesquisa e desenvolvimento.

C.1 README

```
------------------------------------------------------------------------
Controlador de Acesso Reflexivo e Retroativo Indexado por Erros (CARRIE)
------------------------------------------------------------------------

Copyright (c) 2008-2010 Centro de Computacao Cientifica e Software Livre
Departamento de Informatica - Universidade Federal do Parana - C3SL/UFPR

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.


---------------------------------------------------------------------
Descrição do CARRIE
---------------------------------------------------------------------

O CARRIE é arcabouço destinado a criação de Objetos de Aprendizagem (OA).
Com ele é possível criar OA sem a preocupação com a codificação das
seguintes características:

(a) Paginação para frente e para trás;
(b) Acesso por índice geral;
(c) Glossário de termos;
(d) Calculadora;
(e) Teclado virtual para a entrada mais natural de operandos de
    uma expressão;
(f) Botões de controle do tamanho do texto;
(g) Bloco de anotações;
(h) Arquivamento do estado da aplicação;
(i) Menu de acesso a erros cometidos anteriormente.
```

O CARRIE foi construído para ser um arcabouço destinado o desenvolvimento de OAs com o uso de Java Applets e Java JFrames com swing.

Primeiramente para usar este arcabouço é necessário adicionar o jar carrie.jar no classpath da sua aplicação.

Após isso, para utilizá-lo com applets é necessário estender a classe AppletCARRIE e para utilizá-lo com JFrames é necessário estender a classe JFrameCARRIE.

Após isso, você pode criar seus paneis isoladamente e então adicionar eles no arcabouço CARRIE.

As configurações que serão apresentadas podem ser usadas tanto para os applets quanto para os JFrames. Mas, por medida de simplificação será indicado como essas configurações se aplicam aos applets.

Adicionando JPanels no Arcabouço CARRIE

Na classe que estendeu o AppletCARRIE deve-se usar o método ‘‘addPanel(name, panel)’’ passando como parâmetro o nome do painel e o objeto painel. O código a seguir representa a adição de painéis no CARRIE.

```
addPanel(‘‘Situação 1’’, new ExerciseOne());
addPanel(‘‘Situação 2’’, new ExerciseTwo());
addPanel(‘‘Situação 3’’, new ExerciseThree());
super.updateNewPanels()
```

O método super.updateNewPanels() deve ser chamado apenas uma vez, e sempre depois de adicionar todos os paneis. Pois ele vai realizar a atualização do CARRIE para que ele possa visualizar os painéis recém adicionados.

OBSERVAÇÃO:

O nome do painel deve ser único, caso contrário o painel com mesmo nome adicionado por último irá sobrescrever o adicionado, de mesmo nome, anteriormente.

Arquivo de configuração do Arcabouço CARRIE

Para as configurações principais do CARRIE deve criar um arquivo com o nome ‘‘appconfigproperties’’ dentro da pasta src. Após isso as seguintes variáveis devem ser setadas:

```
title = nome da applicação
width = 600
height = 500

Se você não fizer isso é provável que sua aplicação não funcione
corretamente. Caso você precise de mais propriedades, você pode
adicioná-las neste arquivo e então recuperar seu valor através do
método:

Util.getProperty(‘‘nome da propriedade’’);

-----------------------------------------------------------------
Bloqueio e Desbloqueio de Painéis no Arcabouço CARRIE
-----------------------------------------------------------------

Através do CARRIE é possível bloquear e desbloquear painéis quando
achar-se necessário. Para isso, é só seguir os seguintes exemplos:

  AppletCARRIE.blockPanels(String[] names)
  Através desse método é possível passar uma array de strings com os
  nomes dos painéis a serem bloqueados.

  AppletCARRIE.unblockAllPanels()
  Através desse método pode-se desbloquear todos painéis.

  AppletCARRIE.unblockPanels(String[] names)
  Através desse método é possível passar uma array de strings com os
  nomes dos painéis que serão desbloqueados.

-----------------------------------------------------------------
Esconder e Mostrar Painéis no Arcabouço CARRIE
-----------------------------------------------------------------

O CARRIE permite que painéis sejam escondidos e mostrados em qualquer
momento. Para isso, três métodos podem ser utilizados:


  AppletCARRIE.EdenyAccessToPanels(String[] names)
  Através desse método é possível passar uma array de strings com os
  nomes dos painéis que deseja esconder.

  AppletCARRIE.allowAccessToAllPanels();
  Através desse método pode-se disponibilizar todos painéis.

  AppletCARRIE.allowAccessToPanels(String[] names);
  Através desse método pode-se passar uma array de strings com os
  nomes dos painéis que deseja disponibilizar.

-----------------------------------------------------------------
Glossário de Termos do Arcabouço CARRIE
```

```
----------------------------------------------------------------------

Para usar o glossário do CARRIE você precisa fazer duas coisas:

1- No seu diretório src crie a estrutura de pacotes
   /br/ufpr/c3sl/condigital/glossary/

2- Dentro do diretório glossary crie um arquivo chamado de
   glossary.glo

Agora para uma melhor compreenção da formatação deste arquivo siga o
seguinte exemplo:

  |--------------------------------------
  |          File glossary.glo
  |--------------------------------------
  |Abaxial:
  |  Aquilo que está fora do corpo ou de uma parte do órgão.
  |
  |ABES:
  |  Associação Brasileira de Engenharia Sanitária.
  |
  |Abiocenose:
  |  Todos os elementos não vivos de um ecossistema. Por exemplo:
  |  as características geológicas e climáticas.
  |
  |--------------------------------------
  |End glossary.glo
  |--------------------------------------

Este arquivo deve ter a seguinte formatação:

Termo      -> Deve estar no início da linha e terminar com ‘‘:’’
Explicação -> Deve ficar logo abaixo do termo e sempre a dois espaços
              do início da linha.

OBS: Linhas em branco serão ignoradas.

----------------------------------------------------------------------
Notificar Mudança de Paginação do CARRIE
----------------------------------------------------------------------

Em muitos caso é necessário realizar alguma tarefa quando uma mudança
de página ocorre.

Para isso, é necessário que implemente a classe Observer na
classe que precisa ser notificada quando existir uma mudança de
paginação. Após isso, você deve adicionar sua classe como um
observador de PaginationNotify. Para mais detalhes siga o exemplo:

public class Example implements Observer {
```

```
public Example(){
    PaginationChangeNotify.getInstance().addObserver(this);
}

public void update(Observable o, Object arg) {
    if(o instanceOf PaginationChangeNotify)
     //Do what you want!!
}
}
```

Controle da Fonte de um Container Externo - JFrame

É possível que o controle do tamanho da fonte de janelas externas seja feito pelo CARRIE. Para isso, você deve chamar o método abaixo, passando como parâmetro o container que terá o tamanho da fonte controlada.

```
AppletCARRIE.controlFontForExternalContainer(externalContainer);
```

Por exemplo, para controlar a fonte do glossário é necessário fazer o seguinte:

```
AppletCARRIE.controlFontForExternalContainer(GlossaryGUI.getInstance());
```

Para remover o controle da fonte o seguinte método deve ser utilizado:

```
AppletCRI.removeControlFontForExternalContainer(GlossaryGUI.getInstance());
```

Salvamento do estado da aplicação quando um erro ocorre, ou quando for necessário.

Para salvar o estado da aplicação, mais precisamente da painel que foi adicionado no CARRIE, é necessário que seu painel estenda a classe JPanelState e depois disso você deve implementar o método ‘‘configState’’ que retornará um novo objeto com o estado atual do painel adicionado no CARRIE. Para ilustrar isto, visualize o seguinte exemplo:

Classe que adicionada no CARRIE.

```
public class Example extends JPanelState {
    private JTextField jtfText;
    private JCheckBox jcbShowHide;

    public Example(){
        jcbShowHide = new JCheckBox();
```

```
        jtfText = new JTextField();
    }

    ....

    /**
     *  Used to configure the state of object to can save state
     *
     */
    protected JPanelState configState() {
        Example copy = new Example();

        copy.jtfText.setText(jtfText.getText());
        copy.jcbShowHide.setSelected(jcbShowHide.isSelected());

        return copy;
    }
}
```

Após isso você deve marcar os pontos de observação, ou seja, os pontos onde o estado da aplicação será salva.

Para isso, você pode seguir o exemplo abaixo:

```
public class Example {

   private ApplicationState applicationState = new ApplicationState();

   ...

   private void corret() {
       ...
       if (!correct)
           applicationState.save(‘‘Título do erro’’,
                 ‘‘Descrição do erro’’, true);

           /* A última variável é um booleano que quando for
           verdadeiro uma janela popup será mostrada indicando o erro
           e quando for falsa nada será mostrado. */
       ...
   }
   ...
}
```

Teclado Virtual do CARRIE

O teclado virtual contém números, operações, variáveis, decimal entre outras operações utilizando uma linguagem natural. Para utilizá-lo você deve seguir os seguintes passos:

```
Criar um teclado virtual
  - Crie uma instância de VirtualKeyBoardMain e uma KeyBoardComunication
  - Adicione um observer no KeyBoardComunication
  - Parâmetros passados para o construtor do  VirtualKeyBoardMain
    Primeiro parâmetro:
      true - para habilitar as operações (+, -, *, /)
    Segundo parâmetro:
      true - para habilitar as variáveis n e l.
  - Setar a comunicação com o teclado

Pegar a formula do teclado
  - Crie um objeto do tipo ElementOfFormula
  - Pegue a formula através do método getFormula().getClone();

Retornar uma formula para o teclado virtual
  - Para retornar uma formula ao teclado utilize o seguinte método
    setFormula(ElementOfFormula object);

Exemplo de uma classe que utiliza o teclado virtual.


public class ExampleKeyBoard extends JFrame implements Observer{

    private KeyBoardComunication kbComunication;
    private VirtualKeyBoardMain virtualKeyboard;

    private JPanel mainPanel;
    private JPanel formula;

    public ExampleKeyBoard() {
        kbComunication = new KeyBoardComunication();
        kbComunication.addObserver(this);

        virtualKeyboard = new VirtualKeyBoardMain(true, true);
        virtualKeyboard.setCommunication(kbComunication);

        mainPanel = new JPanel(new BorderLayout());
        mainPanel.setPreferredSize(new Dimension(200, 100));

        formula = new JPanel();

        createGUI();

        setContentPane(mainPanel);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setLocationRelativeTo(null);
        setVisible(true);
    }
```

```
    private void createGUI(){
        mainPanel.add(new JLabel(‘‘Clique para abrir o teclado’’),
                BorderLayout.NORTH);

        mainPanel.add(formula, BorderLayout.CENTER);

        formula.addMouseListener(new MouseListener() {
                public void mouseReleased(MouseEvent e) {}
                public void mousePressed(MouseEvent e) {}
                public void mouseExited(MouseEvent e) {}

                public void mouseEntered(MouseEvent e) {
                ExampleKeyBoard.this.requestFocusInWindow();
                }

                public void mouseClicked(MouseEvent e) {
                virtualKeyboard.setVisible(true);
                if (formula.getComponentCount() > 0 &&
                    formula.getComponent(0) instanceof ElementOfFormula) {
                virtualKeyboard.setFormula((ElementOfFormula) formula.getComponent(0));
                }
                }
                });
    }

    public void update(Observable o, Object arg) {
        ElementOfFormula form = virtualKeyboard.getFormula().getClone();
        formula.removeAll();
        formula.add(form);
        formula.updateUI();

        virtualKeyboard.setVisible(false);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                new ExampleKeyBoard();
                }
         });
    }
}
{
-----------------------------------------------------------------
Criação de Introdução com o CARRIE
-----------------------------------------------------------------

Com o CARRIE é possível criar a introdução ao conteúdo a ser tratado
de uma maneira clara e simples pois, para isso pode-se criar um
arquivo de texto com as configurações que o HTML permite e a partir
desse arquivo o CARRIE consegue montar a introdução.
```

Para utilizar este recurso você precisar indicar para o CARRIE qual é o caminho do arquivo a ser carregado através da chamada do método addIntroductionFromHtmlFile em sua classe principal. Nele você deve passar como parâmetro uma string contendo o caminho do arquivo.
Por exemplo:

```
addIntroductionFromHtmlFile(‘‘filename’’);
```

Porém para isso você deve tomar alguns cuidados, pois as seguintes tags devem estar no seu arquivo HTML:

```
<legend>Introduction Name, name that will appear in the JComboBox</legend>
<title>title of the introduction</title>
```

Outro ponto importante é que as tags <html></html> são totalmentes dispensadas.

Criação de Enunciado com o CARRIE

O CARRIE oferece um módulo no qual é possível beneficiar-se das configurações de formatação do html para criar enunciados para os exercícios. Para isso você precisa criar um instância da classe Enunciation passando para o construtor o código html e então adicionar está instância no seu painel.

Por exemplo:

```
Enunciation enunciation = new Enunciation(‘‘código em html’’);

yourPanel.add(enunciation);
```

Criar links entre a introdução ou entre enunciado com o Glossário

---

Quando você escreve seu texto em html você pode criar links de palavras com suas definições no glossário de termos. De modo que quando clicadas abram o glossário de termos com a definição em foco.

Para isso você precisa fazer como o exemplo:

```
<a href=‘‘Apple’’> apple </a>
```

Esta linha irá criar um hyperlink para a definição encontrada no glossário de termos da palavra ‘‘Apple’’.

Navegar pelas páginas de dentro do código

Em alguns momentos você pode querer avançar um página a frente ou voltar uma página para trás sem a interação do usuário. Para isso você pode utilizar os seguintes métodos:

moveAhead(); Avança para a página seguinte;

mobeBack(); Retorna para a página anterior;

moveToIndex(int index); Vai para página correspondente ao index.

OBS: Estes métodos são herdados pela classe que estende AppletCARRIE

DIEGO MARCZAL

UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2010