UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Astrha – Um Ambiente Gráfico, Dinâmico e Interativo para Internet Baseado em Hiper-Animações e na Teoria dos Autômatos

por

ROGES HORACIO GRANDI

Dissertação submetida à avaliação como requisito parcial para
a obtenção do grau de Mestre em Ciência da Computação.

Prof. Dr. Paulo Fernando Blauth Menezes
Orientador

Porto Alegre, outubro de 2003.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Grandi, Roges Horacio
Astrha – Um Estudo da Aplicação da Teoria dos Autômatos para Estruturar Hiper-Animações em um Ambiente Gráfico, Dinâmico e Interativo para Internet / por Roges Horacio Grandi. – Porto Alegre: PPGC da UFRGS, 2003.
252 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Menezes, Paulo Fernando Blauth Menezes.

1. teoria dos autômatos 2. hiper-animação 3. linguagem de quarta geração 4. educação a distância 5. software livre. I. Menezes, Paulo F. Blauth. II. Título.

Agradecimentos

Quando se tem amigos, a vida é mais plena. Inicio agradecendo a todos os meus amigos que me amam, me querem bem, e que desejam que eu progrida nos vários aspectos humanos: físicos, mentais, emocionais, sociais, espirituais. A todos, meu agradecimento e desejo.

Não tem como deixar de agradecer ao Instituto de Informática, com sua estrutura digna, de respeito ao conhecimento e à comunidade. Aqui agradeço a todos, desde o diretor, professor Navaux, até o mais humilde e importante funcionário.

Vejo o professor Blauth, meu orientador, como um pai. É a imagem que me traz, porque nos trata, seus orientados, como árvores que devem crescer com sua própria luz e humus, mas que coloca as devidas cercas protetoras no momento apropriado. Quando não precisa mais, tira as cercas e deixa crescer com liberdade. Nosso grupo de pesquisa? Foi quando mais precisei, ao ter que abrir mão de minha bolsa CNPq, ao qual também agradeço, que mais apoio recebi. Obrigado Accorsi, Campani, Carlos, César, Cláudia, Claudio, Fábio, Guilherme, Júlio, Karina, Link, Marnes, Renata Zanella, Renata Wotter.

O professor Tiaraju deu-me um grande incentivo para formar-me mestre. A professora Nedel foi colaboradora decisiva no momento de definir o rumo da pesquisa. A professora Rosa continuou me apoiando e ofereceu o Laboratório de Computação e Música (LC&M) como ambiente de experimentação para estudos de caso, de onde surgiram estudos interdisciplinares e um artigo em co-autoria. Agradecimentos aos colegas musicais Eloy, Evandro, Lauro, Leandro, Luciano, Tales e Vinícius.

Na Procergs, os gerentes Cevallos, Claudete, Emília e Márcia Mattos flexibilizaram meus horários de consultoria permitindo a conclusão desta pesquisa. Os colegas Alexandre, Andres, Dênis, Flávio, Gabriel, Giovani, Maurício, Nivaldo e Píferro ficaram torcendo para que tudo desse certo. Na FARGS, os professores Carlos, Gustavo e demais colegas deram todo o apoio, sabedores da importância pessoal e acadêmica da pesquisa realizada.

A tia Gláucia sempre apóia os sobrinhos. O Antonio Carlos, meu cunhado, já quer que eu inicie os estudos em nível de doutorado. Quando quebrei as pernas financeiramente, devido à prioridade aos estudos, a tia Zilá socorreu, foi providencial. Minha mãe, Vania, sempre me ensina a perceber o valor de pequenas e grandes realizações, lembro sempre disso. Meu pai, Oswaldo, me mostra através do seu exemplo que para se realizar na vida, tem que se esforçar. Acho que é herança da minha avó Ermelinda. O meu irmão Cesar me mostrou o valor do hábito da leitura. Letícia e Larissa, minhas irmãs, agradeço por acreditarem em mim.

Acreditar em mim? Acho que é isso que a Margarete, minha mulher, sempre faz. Revisora oficial desta dissertação, sempre acredita em mim. Acho que essa é uma de suas maiores expressões de amor. Meus filhos? Bem, ainda não vieram. Mas quando lerem estas linhas, saberão que muito desta pesquisa fiz pensando em vocês.

Paramahansa Yogananda, meu mestre,
minha luz, minha alegria

*"Se soubesse que o mundo se desintegraria amanhã,
ainda assim plantaria a minha macieira"* - Martin Luther King

Sumário

Lista de Abreviaturas

ABNT	Associação Brasileira de Normas Técnicas
AIF	Formato de arquivo de som para Macintosh de propriedade da Apple Computer.
AIFF	Formato de arquivo de som para Macintosh de propriedade da Apple Computer.
AOL	America On Line
API	Application Program Interface
ARPA	Advanced Research Projects Agency
AU	Formato de arquivo de som de propriedade da Sun Microsystems
AWT	Abstract Window Toolkit
BNF	Backus-Naur Form
CASE	Computer-Aided Software Engineering
CERN	European Organization for Nuclear Research, Suíça
CGI	Common Gateway Interface
CMU	Carnegie-Mellon University, Estados Unidos
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CSS	Cascading Style Sheet
DARPA	Defense Advanced Research Projects Agency, Estados Unidos
DCA	Defense Communication Agency, Estados Unidos
E-R	Diagrama Entidades-Relacionamentos
DISA	Defense Information Systems Agency, Estados Unidos
DoD	Department of Defense, Estados Unidos
DTD	Document Type Declaration
EAD	Educação a distância
FRESS	File Retrieval and Editing System
FTP	File Transfer Protocol
Funcion.	Funcionalidade
GIF	Graphics Interchange Format
HES	Hypertext Editing System
H-LAM/T	Human using Language, Artifacts, and Methodology
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
IBM	International Business Machines
IEC	International Electrotechnical Commission
IGD	Interactive Graphical Documents
IHC	Interação Humano-Computador
IP	Internet Protocol
ISO	International Organization for Standardization
JAXP	Java API for XML Processing
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JSP	Java Server Pages
L4G	Linguagem de Quarta Geração
LISP	LISt Processing language

LZW	Lempel-Ziv-Welch, algoritmo de compressão
MCT	Ministério da Ciência e da Tecnologia
MIME	Multipurpose Internet Mail Extensions
MIDI	Musical Instrument Digital Interface, formato de arquivo de som
MIT	Massachusetts Institute of Technology, Estados Unidos
NASA	National Aeronautics & Space Administration, Estados Unidos
NLS	oN Line System
NSF	National Science Foundation, Estados Unidos
OMG	Object Management Group
OO	Orientação a objetos
PARC	Xerox Palo Alto Center, Estados Unidos
Portab.	Portabilidade
Priorid.	Prioridade.
RGB	Red, Green and Blue
RTF	Rich Text Format
SDE	Symbolic Document Examiner
SGBD	Sistema Gerenciador de Banco de Dados
SGML	Standard Generalized Markup Language
SDE	Symbolic Document Examiner
SMIL	Synchronized Multimedia Integration Language
SRI	Stanford Research Institute, Estados Unidos
TCP	Transmission Control Program
TCP/IP	Transmission Control Program over Internet Protocol
UCLA	University of California at Los Angeles, Estados Unidos
UCSB	University of California, Santa Barbara, Estados Unidos
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Usab.	Usabilidade
W3C	World Wide Web Consortium
WAIS	Wide Area Information Servers
WAV	Formato de arquivo de som para Windows, de propriedade da Microsoft.
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XLink	XML Linking Language
XML	Extensible Markup Language

Lista de Figuras

Lista de Tabelas

Resumo

Esta pesquisa, batizada Astrha *(Automata Structured Hyper-Animation)*, tem suas raízes no projeto "*Hyper Seed - Framework*, Ferramentas e Métodos para Sistemas Hipermídia voltados para EAD via WWW" que possui, entre seus objetivos e metas: (a) o desenvolvimento de uma fundamentação matemática para a unificação, de maneira coerente e matematicamente rigorosa, de especificações de sistemas hipermídia e animações baseadas na Teoria dos Autômatos; (b) a construção e validação de um protótipo de sistema com suporte à criação de conteúdo multimídia e hipermídia com ênfase em educação assistida por computador; (c) a definição e aplicação de estudos de caso.

Atender às demandas acadêmicas e construtoras supra citadas, no que se refere à unificação de especificações de sistemas hipermídia e animações baseadas na Teoria dos Autômatos, em nível conceitual, é o objetivo principal do Astrha. Mais especificamente, unificar conceitos das especificações *Hyper-Automaton*; *Hyper-Automaton*: Avaliações Interativas; *eXtensible Hyper-Automaton* (XHA) e Animação Bidimensional para *World Wide Web* (AGA). Para resolvê-las, propõe uma solução em cinco fases. A primeira constitui-se numa investigação conceitual sobre unificação de ambientes hipermídia com animações por computador, da qual conclui-se que as hiper-animações são uma resposta adequada ao contexto. Em seguida, um autômato finito não-determinístico, reflexivo, com saídas associadas às transições, denominado Astrha/M, é especializado para modelar, formalmente, estruturas hiper-animadas. Na terceira fase, uma linguagem de quarta geração denominada Astrha/L é proposta com a finalidade de proporcionar semântica à ambientes hiper-animados. Construída a partir da metalinguagem XML, é composta de quatro dialetos: (1) *Mealy*, que traduz o modelo Astrha/M; (2) *Environment,* que oferece opções de configuração e documentação; (3) *Hyper,* linguagem hipermídia, de sintaxe simples, que oferece hiperligações estendidas; (4) *Style,* especificação de estilos em cascata e de caracteres especiais. A quarta fase é a modelagem e construção do protótipo, denominado Astrha/E, através das linguagens UML e Java, respectivamente, com uso de tecnologias de software livre, resultando em um applet interativo, dinâmico, multimídia, que oferece características e propriedades de uma hiper-animação, traduzindo não-determinismos em escolhas pseudo-aleatórias e reflexividades em inoperabilidades aparentes. Por fim, a quinta fase trata de estudos de caso aplicados em educação a distância, em diversas áreas, de onde se conclui sua validade como conceito, modelo e ferramenta para programas educacionais que utilizam a Internet como meio de auxílio ao aprendizado.

Palavras-chaves: teoria dos autômatos, hiper-animação, linguagem de quarta geração, educação a distância, software livre.

TITLE: "ASTRHA - AN INTERNET INTERACTIVE DYNAMIC GRAPHIC ENVIRONMENT BASED ON HYPER-ANIMATIONS AND AUTOMATA THEORY"

Abstract

This research, named Astrha (Automata Structured Hyper-Animation), is a branch of the project "Hyper Seed - Framework, Tools and Methods for Internet based Educational Hypermedia Systems". It has the following objectives and goals: (a) to develop a mathematical foundation for unifying - under consistent restrictions - hypermedia systems and computer animations based on Automata Theory; (b) to build and validate an e-learning hypermedia system's prototype; (c) to define and apply educational cases.

Astrha intends to attend the above mentioned academics and practical demands for unifying, conceptually, hypermedia systems and animations based on Automata Theory. More specifically, to unify specification concepts from Hyper-Automaton; Hyper-Automaton: Interactive Evaluation; eXtensible Hyper-Automaton (XHA) e Two-Dimensional World Wide Web Animation (AGA).

In order to achieve these goals, a five steps solution was proposed. The first one consists of a conceptual inquiry on hypermedia environment unification with computer animations, from which we concluded that hyper-animations have suitable properties for representing this coupling. Next, a non-deterministic reflexive finite automata with outputs associated to the transitions, called Astrha/M, was customized for hyper-animations modeling; The following step was to propose an fourth generation language called Astrha/L to provide semantics for hyper-animated environments. Constructed from XML metalanguage, it is composed of four dialects: (1) *Mealy*, an Astrha/M translation; (2) *Environment*, which offers configuration and documentation options; (3) *Hyper,* which is a simple syntax hypermedia language offering extended links; (4) Style, a cascading style and special characters specification. The fourth step was to model and construct the prototype, called Astrha/E, through UML and Java languages, respectively, using open source technology and resulting in an interactive dynamic multimedia applet that offers hyper-animated properties and behavior that translates non-determinism into pseudo-random selection and reflexivity into apparent no operation (NOP). The fifth and last step deals with applied studies on distance learning in diverse areas, allowing us to have some conclusions about its validity as a concept, model and tool for educational programs where Internet is used as a learning media aid.

Keywords: automata theory, hyper-animation, fourth generation language, distance learning, open source software.

1 Introdução

Esta dissertação faz parte de um projeto de pesquisa denominado "Hyper Seed – Framework, Ferramentas e Métodos para Sistemas Hipermídia voltados para EAD via WWW", proposto pelo Instituto de Informática da UFRGS em 12/11/2002, aprovado e apoiado pelo CNPq, que possui como objetivos e metas:

"**a) Objetivos**

Os objetivos deste projeto são:

1. Baseado nos modelos formais desenvolvidos e nos resultados atingidos em três projetos de pesquisa desenvolvidos pelo Grupo

(a) Hyper-Automaton, um sistema adaptativo de disponibilização de conteúdo hipertexto;

(b) AGA, Animação 2D Baseada em Autômatos para a Web;

(c) Nautilus, linguagem de especificação Orientada a Objetos, com facilidades não-tradicionais inspiradas em Teoria das Categorias;

desenvolver uma fundamentação matemática para a unificação, de maneira coerente e (matematicamente) rigorosa, de especificações de textos e animações, baseada na Teoria dos Autômatos e das Categorias.

2. Baseando-se em tais fundamentos, desenvolver um protótipo de sistema com suporte ao desenvolvimento de conteúdo multimídia e hipermídia com ênfase em aplicações de e-learning.

3. Validar este protótipo em dois contextos:

(a) Num contexto típico de instituição de ensino, junto ao Instituto de Informática da UFRGS;

(b) Num contexto corporativo, junto à empresa PLANCTA (www.plancta.com.br), em experiências de massa, estimado em pelo menos 15.000 usuários.

b) Metas

Ano 1: Desenvolvimento do framework matemático e especificação do protótipo.

Ano 2, semestre 1: Implementação do protótipo da aplicação-chave e definição dos estudos de caso.

Ano 2, semestre 2: Instanciação e validação dos estudos de caso.

Ano 3, semestre 1: Formalização e validação final do modelo e aplicação-chave desenvolvidos, e disseminação da aplicação-chave para outras instituições.

[MEN 2002, p. 5]"

1.1 Motivação

Podemos observar vários desafios nesse projeto *(Hyper Seed)*, entre eles:

a) A unificação de modelos formais desenvolvidos em projetos de pesquisa do grupo (Hyper-Automaton, AGA, Nautilus);

b) Desenvolver um protótipo de sistema com suporte ao desenvolvimento de conteúdo multimídia e hipermídia com ênfase em aplicações de aprendizado assistido por computador através da Internet *(e-learning)*;

c) Validar este protótipo num contexto típico de instituição de ensino, junto ao Instituto de Informática da UFRGS;

d) Definição de estudos de caso.

É nesse contexto de desafios que este projeto de pesquisa busca soluções. Podemos observar nos projetos Hyper-Automaton (e seus derivados *Hyper-Automaton: Interactive Evaluations*, XHA) e AGA, uma aproximação de abordagem no que se refere à utilização de autômatos finitos com saída (AFS) como modelo estrutural [MAC 2000, MAC 2002, MOR 2002, ACC 2002], conforme mostra a FIGURA 1.1. Já o projeto Nautilus[1], apresenta sua abordagem focalizada na Teoria das Categorias [MEN2002a, STA2002] e na Orientação a Objetos (OO) [SEB 2001].

[1] Linguagem acadêmica de especificação orientada a objetos, concorrente e antecipatória, inspirada em Teoria das Categorias [MEN 2002b, MEN 2002, CAR 99].

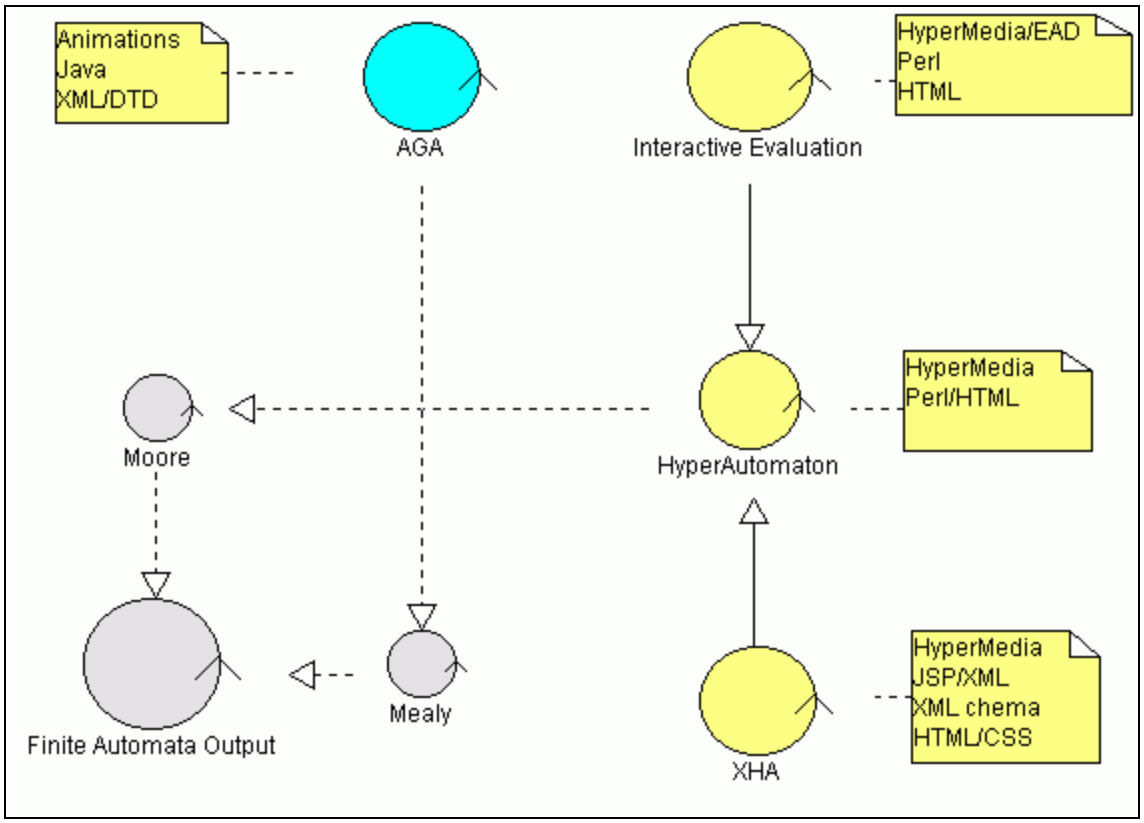


FIGURA 1.1 - Pesquisas baseadas em modelos de autômatos finitos com saída

Sendo um dos objetivos do *Hyper Seed* o desenvolvimento de uma fundamentação matemática para a unificação de especificações de textos e unificações, baseada nas Teoria dos Autômatos e das Categorias, surgem algumas perguntas relativas à unificação dos projetos baseados em AFS - AGA e Hyper-Automaton (incluindo as Avaliações Interativas e XHA) - conforme mostra a FIGURA 1.2:

a) **Conceitos *(Concepts)***: Quais conceitos são envolvidos e gerados a partir da unificação de animações bidimensionais e ambiente hipermídia?

b) **Estrutura *(Structure)***: Uma vez obtido os conceitos necessários, que modelo estrutural de AFS proporcionará sintaxe e, em um segundo passo, semântica simples e objetivas para expressar essa unificação?

c) **Tecnologia *(Technology)***: Sendo objetivos do projeto desenvolver e validar um protótipo, que tecnologia deverá ser empregada? Applets, no caso do AGA, *Hypertext Markup Language* (HTML) dinâmico, na família de projetos Hyper-Automaton, ou ambos?

d) **Produto *(Product)***: O produto será uma mídia multimídia, escolhendo-se tecnologia das applets, ou um ambiente hipermídia, escolhendo-se HTML dinâmico? Ou ambos?

e) **Aplicabilidade *(Applicability)***: Por fim, pode-se validar o produto, na forma de protótipo, em um ambiente típico de instituição de ensino, no caso, o Instituto de Informática da UFRGS.

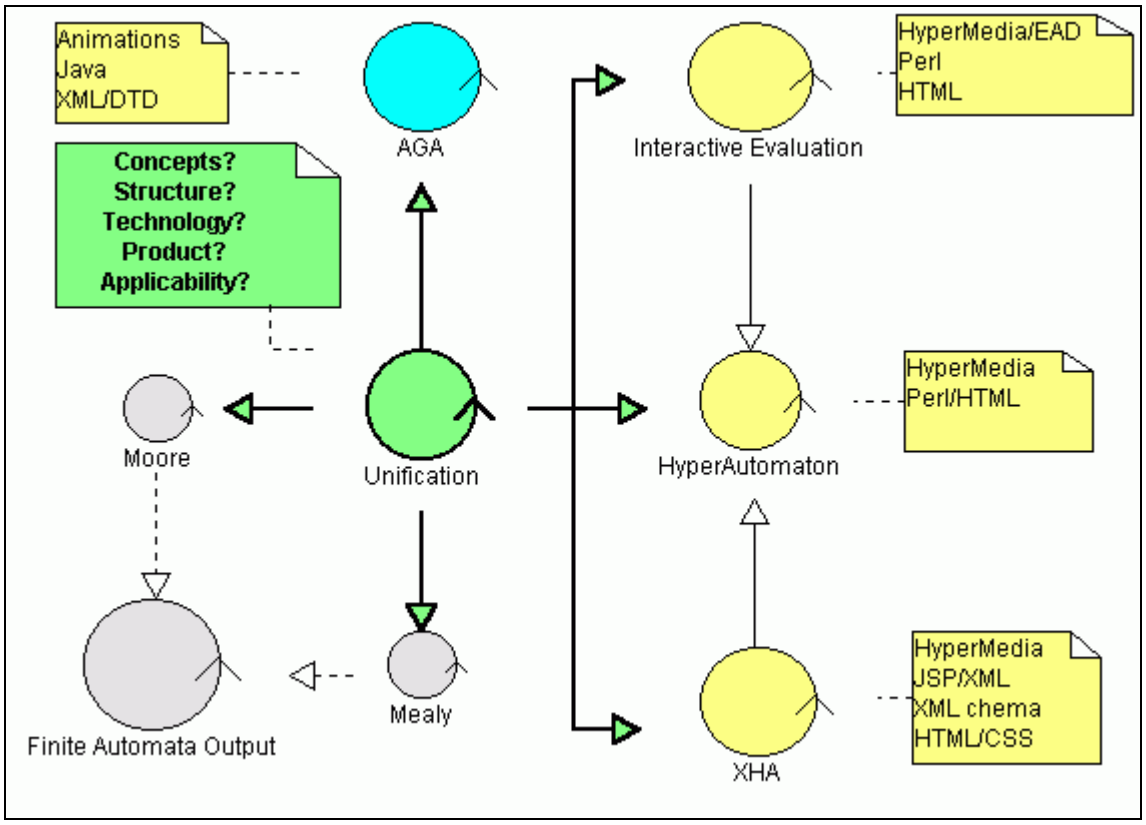


FIGURA 1.2 - Unificação de modelos estruturados por autômatos finitos com saída

1.2 Objetivos

O objetivo geral dessa dissertação é, dentro do contexto do projeto *Hyper Seed*:

1. Investigar uma abstração que expresse animações e ambiente hipermídia em conjunto;
2. Criar um modelo especializado de AFS para estruturar o conceito encontrado;
3. Definir um caminho tecnológico, preferencialmente através de software livre, para a construção de um protótipo funcional baseado no modelo de AFS supra citado;
4. Explorar estudos de caso em um ambiente típico de instituição de ensino - no caso, o Instituto de Informática da UFRGS – que valide todos os passos da pesquisa: conceitos, modelo, tecnologia e aplicabilidade, em especial para educação a distância (EAD).

Note-se que, dentro dos objetivos e metas do *Hyper Seed*, está prevista também a unificação com a linguagem de especificação Nautilus, que por suas características estruturais (Teoria das Categorias mais Orientação a Objetos), diferente do enfoque dos AFS, fica remetida a trabalhos futuros. Cabe, porém, definir a tecnologia para a construção do protótipo neste trabalho preferencialmente utilizando OO, já prevendo a futura unificação com Nautilus. No que se refere ao modelo estrutural, a abordagem

matemática de Nautilus baseada na Teoria das Categorias deverá favorecer à unificação, uma vez que "uma das principais aplicações da Teoria das Categorias é a de *Unificação de Estruturas Matemáticas* [MEN 2002a]".

1.3 Principais Contribuições Científicas

As principais contribuições científica deste trabalho são:

- ***Hyper Seed***: no que se refere a esse projeto de pesquisa, contribui efetivamente para seus objetivos e metas de:
  - unificação de conceitos e de modelos formais de AFS para ambiente hipermídia e animações;
  - construção e validação de um protótipo em um contexto típico de ensino, o Instituto de Informática da UFRGS;
  - instanciação e validação de estudos de caso, em especial aplicados em EAD.
- **Astrha/M**: um modelo de autômato finito com saídas associadas às transições (máquina de Mealy) não-determinística, reflexiva, com conjunto finito de palavras de saída, especializada para estruturar hiper-animações, permitindo semânticas de pseudo-aleatoriedade e de inoperabilidade aparente.
- **Astrha/L**: uma linguagem de quarta geração (vide glossário), composta por quatro dialetos, centrada no modelo Astrha/M, com a finalidade de possibilitar escrita e manutenção simples de ambientes hiper-animados, até mesmo por usuários leigos:
  - **dialeto Mealy** traduz Astrha/M em XML. Para esse dialeto, é apresentado um estudo semântico complementar sobre a possibilidade de utilização do asterisco como caracter curinga (simbolizando a totalidade dos elementos de um conjunto) em substituição a estados, a símbolos de entrada ou palavras de saída em transições;
  - **dialeto Environment** oferece marcas simples e diretas para configuração de um ambiente hiper-animado, documentar dados de produção, listar as máquinas de Mealy que serão utilizadas no ambiente e hiper-animações que serão instanciadas automaticamente ao se carregar o ambiente;
  - **dialeto Style** contribui como especificação de linguagem de estilos em cascata em XML, inspirada na linguagem CSS do W3C;
  - **dialeto Hyper** também contribui cientificamente oferecendo uma linguagem hipermídia de sintaxe simples, com uma pequena quantidade de marcações, mas que oferece hiperligações estendidas e semântica suficiente para descrever textos formatados, hipertextos e mapas clicáveis.
- **Astrha/G**: um protótipo desenvolvido em Java applet que interpreta código Astrha/L, oferecendo ambientes hiper-animados, com tecnologia de software livre.

- Sendo os textos desenhados a partir de bibliotecas gráficas, em Astrha/E operações do tipo copiar e colar, ou então do tipo salvar em arquivos texto, podem ser desabilitadas se desejado, sendo útil para dificultar pirataria eletrônica e preservar direitos autorais dos conteúdos que estão sendo publicados;

Resumidamente, a principal contribuição científica desta pesquisa é a validação dos AFS como modelo estrutural para hiper-animações, em um processo de análise consciente, iniciado em um modelo especializado e formalmente descrito, Astrha/M, passando por uma tradução para a linguagem Astrha/L e terminando por ser construído com tecnologia de software livre em ambientes Astrha/E, através dos quais estudos de caso aplicados em EAD foram validados.

1.4 Estrutura da Dissertação

- **Capítulo 1 Introdução:** apresenta os objetivos, o contexto e as principais contribuições da pesquisa.
- **Capítulo 2 Projeto *Hyper Seed*:** comenta aspectos do Projeto *Hyper Seed* no contexto Astrha.
- **Capítulo 3 Hipertecnologia, Animações e Hiper-Animações:** revisão bibliográfica sobre hipertecnologia, apresentando ambientes gráficos e estruturas que embutem esses métodos de interação.
- **Capítulo 4 Modelo do Autômato (Astrha/M)**: formaliza uma Máquina de Mealy não-determinística e reflexiva especializada para definir hiper-animações.
- **Capítulo 5 Linguagem Astrha/L**: apresenta a linguagem de marcação e seus quatro dialetos destinada a especificar hiper-animações, ambientes, estilos e documentos hipermídia.
- **Capítulo 6 Ambiente Astrha/E**: em um processo incremental e interativo, mostra as características e funcionalidade do protótipo que implementa o modelo de autômato e linguagem propostos.
- **Capítulo 7 Aplicabilidade**: apresenta estudos de casos implementados, em desenvolvimento e potenciais, tendo como foco principal exemplos em educação a distância.
- **Capítulo 8 Conclusão**: reflete sobre os resultados obtidos e propõe trabalhos futuros.
- **Anexo I**: Projeto de pesquisa "*Hyper Seed – Framework*, Ferramentas e Métodos para Sistemas Hipermídia voltados para EAD via WWW", proposto pelo Instituto de Informática da UFRGS em 12/11/2002, aprovado e apoiado pelo CNPq, do qual esta dissertação faz parte [MEN 2002].
- **Anexo II**: Revisão bibliográfica sobre aspectos históricos relacionados à hipertecnologia, uma das bases desta pesquisa.
- **Anexo III**: Artigo "*A Database Structure for Didactic Exercises on Distance*

Learning Programs Adapted to ISO/IEC 9126 Standard". Anais da *International Conference on Engineering and Computer Education* ICECE, realizado na cidade de São Paulo, São Paulo, Brasil, em agosto de 2000 [GRA 2000].

- **Anexo IV**: Artigo "Utilização do XML no Sistema Hyper-Automaton". Anais do *International Symposium on Knowledge Management/Document Management* ISKM/DM, realizado na cidade de Curitiba, Paraná, Brasil, em novembro de 2000 [MAC 2000a].
- **Anexo V**: Artigo "Hiper-Animações - Teoria Hipermídia Aplicada em Animações". Anais da *International Conference on Engineering and Computer Education* ICECE, realizado na cidade de Santos, São Paulo, Brasil, em março de 2003 [GRA 2003].
- **Anexo VI**: Artigo "Astrha/E - Ambiente Java/XML que Implementa Hiper-Animações estruturadas por Máquinas de Mealy". Anais do IV Workshop sobre Software Livre - WSL2003. Porto Alegre, Rio Grande do Sul, Brasil, junho de 2003 [GRA 2003a].
- **Anexo VII**: Artigo "Utilização do Ambiente Astrha para Implementar um Dicionário de Acordes Baseado em Autômatos Finitos". Aprovado pelo Simpósio Brasileiro em Computação Musical SBCM, que será realizado na cidade de Campinas, São Paulo, Brasil, em agosto de 2003 [GRA 2003b].
- **Glossário:** Contém termos técnicos utilizados.
- **Bibliografia**

2 Projeto *Hyper Seed*

2.1 Resumo

Este capítulo sumariza uma revisão bibliográfica sobre o projeto *Hyper Seed*, focalizando aspectos relacionados ao Astrha. No intuito de validar os AFS como modelo estrutural de ambientes hipermídia e animações, as seguintes pesquisas têm sido realizadas no Instituto de Informática da UFRGS: *Hyper-Automaton*; *Hyper-Automaton*: Avaliações Interativas; *eXtensible Hyper-Automaton* (XHA) e Animação Bidimensional para *World Wide Web* (AGA). De uma forma geral, essas pesquisas têm dado preferência à estruturas de máquinas de Mealy[2] em relação às máquinas de Moore[3] visto à tendência das máquinas de Mealy resultarem em máquinas com menor quantidades de estados do que equivalentes máquinas de Moore, devido à semântica associada às transições ou aos estados, respectivamente.

O anexo I apresenta, na íntegra, o projeto *Hyper Seed*. Para maiores informações, sugere-se uma visita à página do grupo em http://teia.inf.ufrgs.br [MEN 2003].

2.2 Hyper-Automaton

Júlio Machado procurou validar o uso de AFS para estruturar sistemas hipertextos, em especial para cursos na Web, apresentando o *Hyper-Automaton*: Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída. Conforme seu autor, essa pesquisa é centrada:

> "...no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na *WWW*, com especial enfoque no desenvolvimento de sistemas hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades descritas no Modelo Dexter como a composição de estruturas hierárquicas, especificação de vários conjuntos de *links* sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação [MAC 2000]."

[2] Autômato finito modificado de forma a gerar uma palavra de saída a cada transição [MEN 2001].

[3] Autômato finito com saída que gera uma palavra de saída (que pode ser vazia) para cada estado da máquina [MEN 2001].

O funcionamento do Hyper-Automaton, conforme a página do grupo de pesquisa do *Hyper Seed*, é como segue:

> "Cada autômato define um curso e consiste em um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web.
>
> A interface do ambiente de um navegador Web fornece uma interpretação tangível para a estrutura de autômatos em hiperdocumentos. Os símbolos do alfabeto de saída são anotados com unidades de informação (páginas HTML) e, neste caso, o resultado das funções de saída (Máquina de Moore) ou de transição (Máquina de Mealy) é a apresentação de um hiperdocumento na janela do navegador, de acordo com os fragmentos indicados na palavra de saída.
>
> O alfabeto de entrada, que nomeia as transições do autômato, são apresentados ao usuário como âncoras de links de navegação.
>
> O sistema Hyper-Automaton também fornece funcionalidade para a construção de avaliações adaptativas via Internet. Neste caso, entende-se como adaptativa, uma prova cuja seqüência de apresentação de questões é dependente da performance do aluno ao responder questões prévias.
>
> Nos exercícios e provas disponibilizados no sistema, as questões foram construídas segundo três níveis de dificuldades e o aluno reponde primeiramente uma questão de nível médio, caso acerte a resposta o sistema providencia uma questão de nível mais alto, caso contrário terá uma chance de recuperar o conceito errôneo através de uma questão formulada especialmente para a recuperação.
>
> O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens:
>
> 1. Facilidade de reuso de páginas em diversos cursos, com eliminação da redundância
>
> 2. Independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa
>
> 3. Permite que qualquer usuário crie links de e para qualquer documento
>
> 4. Facilidade de implementação e manutenção; interface gráfica simples e direta
>
> 5. Elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado
>
> 6. Operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [MEN 2003]."

2.2.1 *Hyper-Automaton*: Avaliação Interativa

Uma versão especializada do *Hyper-Automaton* destinada a automatizar a aplicação de avaliações de conhecimentos, o *Hyper-Automaton:* Avaliação Interativa de Alunos em Cursos na Web Baseado em Autômatos Finitos, foi apresentado por Carlos Morais. Também nas palavras de seu autor, trata-se de:

> "...uma técnica da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em destacar especial, Avaliação e Exercício. Esse objetivo é motivado pela organização e agilização do processo de avaliação proporcionado ao professor e ao aluno [MOR 2002]."

O funcionamento das provas adaptativas é como segue:

> "Cada prova é composta de seis questões objetivas, de cinco opções, sendo que apenas uma está correta. Responda as questões na ordem em que desejar. Ao final, clique em enviar; preencha o campo de nome e clique para enviar outra vez.
>
> Todas as questões devem ser respondidas antes de enviar os dados, caso contrário, será pedido que as responda. O resultado da aplicação da prova é imediato, mostrado ao aluno pouco depois do envio.
>
> Nestes testes eletrônicos é perfeitamente aceitável o uso de material de consulta, seja através de livros, seja pela internet [MEN 2003]."

2.2.2 *eXtensible Hyper-Automaton* (XHA)

Outra pesquisa da mesma linha, *eXtensible Hyper-Automaton* (XHA), apresentada por César Machado e Gustavo Federizzi, objetiva:

> "...estudar as possibilidades de flexibilização da função de saída do Sistema *Hyper-Automaton* além das rígidas possibilidades utilizadas atualmente com a utilização direta do HTML, objetivando eliminar as limitações como execução de aplicações proprietárias, caracteres incompatíveis entre browsers, excesso de tráfego na rede, padronizar aplicações , incrementar recursos didáticos, melhorar o suporte a aplicações multimídia atuais e futuras, facilitar a manutenção, implementação e reuso, alterar o *layout* de saída no *browser* de maneira dinâmica, explorar outros recursos de *links*, estabelecer padrões de organização do material instrucional criado pelo professor e muitas outras. Tal sistema anteriormente desenvolvido e funcionando adequadamente, é baseado no formalismo de Autômatos Finitos com Saída como modelo estrutural para organização de hiperdocumentos instrucionais, em especial em cursos na *Web*, tornando o material hipermídia independente do controle da aplicação. O Sistema *Hyper-Automaton* tornou-se, portanto, um sistema semi-automatizado para suporte a cursos na *Web* [MAC 2002]."

Nas conclusões do autor, é realizada uma comparação do XHA com a implementação anterior do Hyper-Automaton, listando os sequintes itens:

> - **Organização de arquivos**: Na versão anterior cada curso tinha seu próprio diretório onde ficavam os seus três arquivos de definição: curso.fl, alfabeto.fl e estados.fl. Além disso, todos os fragmentos HTML disponíveis eram armazenados em apenas um diretório do servidor;
> - **Manutenção facilitada**: Enquanto que, na versão codificada em Perl, o acesso às máquinas de Moore e de Mealy eram feitos através de dois sistemas diferentes, na nova versão em JSP a funcionalidade das duas máquinas foram incorporadas em um só sistema. Assim, a manutenção se torna muito mais fácil e o acesso ao sistema se torna único;
> - **Reuso inteligente**: O primeiro sistema Hyper-Automaton já utilizava o conceito de reuso através da disponibilização de material instrucional na forma de fragmentos de HTML. Através do uso de XML e XSL, o conteúdo do material de ensino foi separado de sua forma de apresentação. Isso significa que um mesmo objeto de ensino é apresentado conforme a diagramação do curso que o utiliza [MAC 2002, p. 136-137]."

2.2.3 Animação Bidimensional para *World Wide Web* (AGA)

Do mesmo grupo de estudos, surgiu outra pesquisa que utilizou AFS como elemento estrutural em outra área da ciência da computação, a da Computação Gráfica. Fernando Accorsi e Guilherme Magalhães visualizaram a aplicação dos AFS como *framework* para a definição de animações por computador e apresentaram o AGA: Animação Bidimensional para *World Wide Web* Baseada em Autômatos Finitos. Accorsi definiu os objetivos do AGA no seguinte parágrafo:

> "Este trabalho aplica a Teoria de Autômatos na proposição de uma nova alternativa para prover animações 2D na World Wide Web, verificando as contribuições alcançadas para as questões relacionadas ao espaço de armazenamento, reutilização e manutenção do conteúdo e suporte à recuperação de informação. Para este objetivo, é proposto o modelo AGA (Animação Gráfica baseada em Autômatos Finitos), o qual especifica a animação a partir de uma estrutura baseada em autômatos finitos com saída. Esse modelo é definido de tal forma que os mesmos autômatos utilizados na especificação, ao serem simulados, realizam o controle da animação durante a apresentação. O modelo AGA apresenta características que favorecem a redução do espaço de armazenamento da animação, provêem suporte à recuperação de informação, colaboram com a reutilização e manutenção do conteúdo das animações [ACC 2002]."

Entre os vários conceitos e características do projeto de pesquisa AGA, podemos destacar [MEN 2003]:

- **Atores**: Uma animação em AGA é uma lista de atores sincronizados, modelados através de máquinas de Mealy, cada ator contendo sua própria fita de entrada, que irá determinar seu comportamento.
- **Máquina de Mealy**: Como dito no item acima, o comportamento de cada ator é definido através de uma Máquina de Mealy determinística onde o alfabeto de saída é formado por objetos gráficos e mídias sonoras, do qual serão formadas palavras de saída que irão descrever a animação do ator.
- **Controle de Tempo na Fita de Entrada**: A fita de entrada de cada autômato (Máquina de Mealy) deve definir quanto tempo cada imagem de ator deve ser apresentada, sendo assim um parâmetro de sincronização do ator com a animação.
- **Transformação de imagens**: A fita de entrada também pode definir transformações de imagens do tipo rotação, translação, zoom e visibilidade.

- **Reutilização**: Através do AGA, é possível mudar as ações de uma ator apenas mudando sua fita de entrada. Essa característica mais as transformações de imagens permitem a geração de animações longas e variadas com um conjunto pequeno de objetos gráficos e sonoros. As características de reutilização de AGA permitiram, em estudos de caso realizados em laboratório, aprimoramentos substanciais em relação a outras tecnologias. Um exemplo desses estudos é mostrado na FIGURA 2.1.

	Animation 1	Animation 2	Animation 3
AGA			
Images Automaton Total	25,750 6,745 32,495	26,190 7,353 33,543	44,648 9,820 54,468
GIF			
Full Frames Partial Frames	11,305 64,558	176,020 93,508	655,151 278,513
Improvement GIF/AGA			
Full Frames Partial Frames	242.5% 98.7%	424.8% 178.8%	1,102.8% 411.3&

FIGURA 2.1 – Comparação de espaços de armazenamentos, tecnologias AGA e GIF.

2.3 Conclusões

Independente da abordagem realizada por cada uma das pesquisas, pudemos perceber que uma das principais vantagens estruturais de se pensar em autômatos finitos com saída para dar semântica a uma determinada especificação é o reuso de símbolos de saída em várias palavras, proporcionando desejável minimização de recursos computacionais para expressar as saídas desejadas.

Também flexível, e simples, é a utilização de fitas de entrada como estímulo para o comportamento de uma animação ou ambiente hipermídia. Podemos estabelecer uma quantidade virtualmente infinita de animações, ambientes hipermídia, hiper-animações. Quantidade essa limitada, apenas, pela variedade de fitas capaz de ser manipulada pela tecnologia na qual estiver inserida.

Outra propriedade de valor, no que se refere ao uso de AFS como estrutura de uma aplicação é sua contribuição para a depuração e manutenibilidade de sistemas. Ao se construir (programar) um autômato, cada estado, transição ou saída pode receber

informações referentes a sua funcionalidade. Essa característica permite que o desenvolvedor ou usuário consiga perceber com mais clareza se a semântica que está sendo reproduzida corresponde à desejada através da análise das informações fornecidas pelo estado atual da máquina e, também, por sua trilha de execução.

Não menos importante, o fato de um modelo formal de AFS ser matematicamente bem definido, auxilia pesquisadores e desenvolvedores interessados em métodos formais a produzir provas relacionadas à funcionalidade, complexidade e semântica de modelos e aplicações.

3 Hipertecnologia, Animações e Hiper-Animações

3.1 Resumo

Este capítulo apresenta um estudo sobre ambientes hipermídia, animações por computador e hiper-animações, incluindo conceituação, estruturas e taxionomias.

O anexo II complementa esta pesquisa focalizando aspectos científicos e históricos sobre hipertecnologia.

3.2 Hipertecnologia

O primeiro visionário que previu a necessidade de sistemas que organizem, acessem e disponibilizem informações de forma não seqüencial foi Vannevar Bush (1890-1974). Esse importante engenheiro e cientista, foi um pioneiro no desenvolvimento de computadores[4] [MEY 2000, RED 97]. Numa época em que os computadores ainda funcionavam à válvula, seu clássico artigo *"As We May Think"* apropriadamente previu que enciclopédias iriam ser miniaturizadas diminuindo, drasticamente, seus custos de produção e de distribuição. Observou que a miniaturização, a diminuição de custos e a conseqüente popularização da informação iriam demandar outras formas de consulta, além das tradicionais seqüencial e indexada. A essas formas de acesso, faltam propriedades que permitam uma pesquisa de larga abrangência com tempos de resposta aceitável, uma vez que limitavam o acesso direto a um registro de uma determinada base de dados e, também, acesso direto entre bases de dados. Como alternativa ao problema, propôs um mecanismo denominado Memex, cujas características funcionais fundamentaram a teoria do hipertexto.

Nunca construído, o Memex permitiria que o usuário organizasse e consultasse todos os tipos de materiais escritos da forma que desejasse, podendo, inclusive, adicionar comentários e interligando documentos a outros documentos como assim o desejasse. Nas palavras de Bush, "um Memex é um dispositivo no qual um indivíduo armazena todos os seus livros, gravações e comunicações, que possui mecanismos que permitem consultas rápidas e flexíveis." Essa característica do Memex de interligar diretamente documentos através de índices associativos, hoje denominados ligações hipertexto *(hypertext links)*, é considerada a essência do Memex, que também é a essência dos hipertextos. Tais ligações poderiam ser combinadas em formulários de percurso *(form trails)*, hoje disponibilizado através de documentos hipermídia, combinados com navegadores *(browsers)*.

---

[4] V. Bush foi nomeado em 1940 presidente do Comitê de Pesquisa de Defesa Nacional dos Estados Unidos. Com o suporte do *Massachusetts Institute of Technology* (MIT), ele coordenou um projeto que retomou o projeto e os estudos do matemático inglês Charle Babagge (1792-1871) sobre máquinas diferenciais *(Difference Engine)*, que serviram como base a arquitetura de computadores atual, com mecanismos de entrada, processamento, saída e armazenamento, e inventou o Analisador Diferencial *(Differential Analizer)*, dispositivo de cálculo considerado o primeiro computador analógico funcional.

O aparelho teria suas informações armazenadas em microfilmes, podendo, assim, tê-las a sua disposição em sua mesa de escritório, que teria vários projetores de microfilmes, habilitando o usuário a ver vários documentos simultaneamente, conforme mostra a ilustração da FIGURA 3.1. Essa idéia de paralelismo é simulada, hoje, por alguns mecanismos em sistemas hipermídia, como os sistemas de janelas e molduras *(frames)*, dentre vários outros possíveis [BUS 45, DEB 2002].

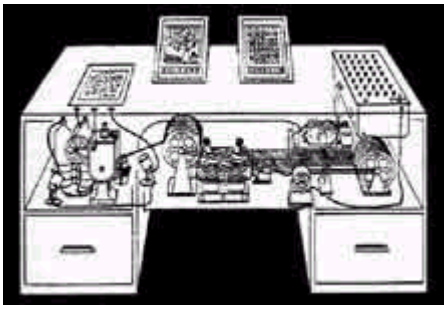


FIGURA 3.1 - Ilustração de uma mesa Memex por Alfred D. Crimi

3.2.1 Sistemas Hipertexto e Hipermídia

Dois dos mais antigos termos relacionados à hipertecnologia são os de hipertexto e hipermídia, ambos cunhados por Theodor H. Nelson em 1967. Dentre várias definições existentes, citamos algumas delas:

Theodor H. Nelson:

> "hipertexto é uma combinação de texto em linguagem natural com capacidades computacionais para gerar ramificações interativas ou apresentação dinâmica de textos de forma não linear, os quais não podem ser impressos em páginas convencionais [NEL 67]."
>
> "hipertexto, ou escrita não seqüencial com liberdade do usuário para se movimentar livremente através de ligações, é uma idéia simples e óbvia. Trata-se, simplesmente, da eletronificação das conexões literárias que já conhecemos [NEL 87]."

Ben Shneiderman:

> "Hipertexto é um banco de dados que possui referências cruzadas ativas e que permite que o leitor pule para outras partes do banco de dados conforme seu desejo [SCH 89]."

Usha Rao:

"É conveniente entender hipertexto como um método para expressar os relacionamentos entre objetos em um banco de dados. O hipertexto deveria ser tratado como uma ferramenta de propósito geral com abordagem direcionada à manipulação de nodos, ligações e acesso, que se ajuste ao contexto de qualquer aplicação e oferece variadas semânticas aos usuários [RAO 90]."

Guilherme M. Doege:

> "Hipertexto é uma rede, constituída pelos nodos e as ligações entre esses nodos. Os textos constituem-se nos nós ou nodos da rede, enquanto que os "links" são os ponteiros que dão ao hipertexto sua característica de não-linearidade [DOE 91]."

Ana C. Salgado:

> "Enquanto um texto possui apenas uma dimensão (linear), um hipertexto possui mais de uma dimensão, que engloba as relações entre os textos. Este livro é um exemplo de um hipertexto onde os capítulos, citações e referências cruzadas correspondem a estas dimensões. Outro exemplo de hipertexto são os livros de referência, como enciclopédias ou dicionários.
>
> Tais livros podem ser vistos como um grafo onde cada nó contém um verbete, e os arcos representam ligações, ou referências, entre os verbetes. Quando se lê um verbete armazenado em um nó, referências explícitas para verbetes relacionados indicam onde se pode obter maiores informações sobre estes itens.
>
> ...
>
> Um sistema hipermídia pode ser definido como aquele que manipula um conjunto de informações, pertencendo a vários tipos de mídia (texto, som, imagem e outros), podendo estas informações serem lidas de forma não-linear através dos diversos caminhos de acesso disponíveis. Em outras palavras, a hipermídia é simplesmente uma extensão do hipertexto que incorpora outro tipo além do texto [SAL 92]."

Aurélio B. de H. Ferreira:

> "hipertexto: informática: conjunto de páginas de informação interligadas ativamente, de forma a possibilitar consultas imediatas em ordem ditada pelo leitor [FER 96]."

Como pudemos observar, existem variadas definições para hipertexto e hipermídia. Uma propriedade, porém, é comum em todas elas: a não linearidade. Pode-se observar, também, que a diferença entre um sistema hipertexto para um sistema hipermídia é simples: o primeiro trabalha somente sobre um conjunto de textos enquanto que o segundo trabalha com pelo menos um tipo de mídia que não seja texto: sons, imagens estáticas, animações, vídeos, etc. Quanto maior o conjunto de tipos de mídia manipulado por um sistema hipermídia, maior será seu poder de expressividade.

No que se refere à diferença entre hipertexto e hipermídia, observamos que o conceito de hipermídia é uma simples extensão de um hipertexto, onde os documentos são enriquecidos por vários tipos de mídia audiovisuais. A palavra mídia vem do inglês, *media*, que por sua vez origina-se do latim, plural de *medium*. Literalmente, significa "aquilo que está entre dois pontos [TOR 42]." No que se refere à comunicação, assume o significado de um veículo que conduz a informação de um ponto para outro. Na área da educação assistida por computador, significa um recurso computacional capaz de transmitir ao ser humano informações. Em interação humano-computador, esse significado pode ser acrescido de meios de navegação em interfaces [HEL 2001].

3.2.2 Hiperligação

Um dos conceitos mais fortes relacionados com a hipertecnologia é o de hiperligação *(hyperlink)*. Genericamente, uma hiperligação é uma referência textual ou gráfica inserida em algum ponto de um documento hipermídia (documento origem) para algum ponto de outro(s) documento(s) hipermídia (documento(s) destino) [W3C 2003a , HOW 2002].

Dentro da vasta gama de propostas de conceitos e tecnologias hipermídia, existem várias propostas e taxionomias para hiperligações, formando um leque de opções bastante amplo.

Conforme a nomenclatura do W3C, se uma hiperligação associar exatamente dois recursos (o documento origem e o documento destino), essa hiperligação é denominada ligação simples *(simple link)*. Uma ligação estendida associa uma quantidade arbitrária de recursos remotos e locais [W3C 2003a].

Davis classifica as hiperligações e âncoras[5] em embutidas e não embutidas. Hiperligações embutidas referenciam as âncoras destino diretamente em seus hiperdocumentos. As não embutidas realizam essas referências de forma indireta, enviando códigos a um servidor que irá fornecer as respectivas âncoras destino. Essa abordagem evita embutir diretamente o endereço das âncoras destino nos hiperdocumentos, permitindo assim a adição de controles via servidor e melhor manutenibilidade [DAV 95].

Uma hiperligação pode ser denominada de duas vias, conforme a nomenclatura do projeto Xanadu, se "qualquer pessoa puder publicar comentários relacionados a qualquer página [XAN 2001]." Outro interessante conceito de hiperligação associado ao projeto Xanadu é o de ligação não rompível *(unbreakable link)*, que uma vez estabelecida em um sistema não pode, simplesmente, ser removida. Tal publicação somente pode deixar de existir se forem ajustados os documentos que o referenciam, garantindo consistência nas ligações. Essa característica é importante no contexto operacional do Xanadu, um sistema macroliterário na classificação de Conklin, cuja finalidade original era formar um sistema de gerenciamento eletrônico de publicações [XAN 2001, XAN 99].

---

[5] Área dentro de um documento hipermídia que serve como fonte ou destino de uma ligação [W3C 2003a, HOW 2002].

3.2.3 Classificações de Sistemas Hipermídia

Jeff Conklin classificou os sistema hipermídia conforme sua aplicabilidade em quatro diferentes áreas [SAL 92 apud CON 87]:

1. **Sistemas macro-literários**: Neste contexto, mimetizam uma biblioteca na qual ligações entre os documentos são suportadas por máquinas. Os leitores podem, conforme a implementação, formar um ambiente colaborativo através de permissões de adição de comentários e críticas às obras arquivadas. A idéia original dos sistemas hipertexto era voltada para sistemas desta categoria, tanto o Memex, de V. Bush, como o Xanadu, de T. H. Nelson.
2. **Ferramentas para exploração de problemas**: São ferramentas que possuem como propósito principal auxiliar a organização de pensamentos para a solução de problemas complexos. Estes sistemas tendem a ser bastante interativos. O NoteCards possui essa característica.
3. **Navegadores *(browsers)***: O primeiro propósito de um navegador é permitir a leitura de várias mídias. Diferencia-se dos sistemas macro-literários pela simplicidade, não havendo a ambição básica de gerenciar as várias funcionalidades relacionadas à manutenção, incremento e melhorias em geral que o funcionamento de uma biblioteca demanda. Os navegadores hipermídia Web possuem essa funcionalidade, que é, na verdade, básica em qualquer sistema hipermídia.
4. **Uso Geral**: Sistemas hipertexto de propósito geral são projetados para resolver uma grande quantidade de aplicações podendo, inclusive, implementar verdadeiros sistemas comerciais que interagem com bancos de dados complexos. Podem interagir com outros serviços de comunicação como o FTP, correios eletrônicos *(e-mails)*, grupos de discussão *(newsgroups)*. Podem formar ambientes colaborativos, enfim, servir para os mais variados propósitos. Os navegadores Web mais sofisticados, como o Internet Explorer e o Mozilla (Netscape Navigator) são considerados sistemas hipermídia dessa categoria.

Roy Rada propôs outra classificação quanto às dimensões, a interatividade e à formação do conhecimento [SAL 92 apud RAD 91]:

1. **Hipertexto pequeno ou microtexto**: sistema que trata-se todas as suas ligações dentro de um único hiperdocumento. O NLS, o ZOG e o Guide são exemplos de sistema deste tipo.
2. **Hipertexto grande ou macrotexto**: sistemas deste tipo sugerem a distribuição das ligações entre vários hiperdocumentos ao invés de centralizá-los em apenas um. O Memex e os navegadores Web em geral apresentam essa proposta.

3. **Hipertexto colaborativo**: tratam-se de sistemas que focalizam solucionar problemas que apresentam uma grande interação entre seus usuários. Os grupos de discussão, os macro-literários de Jeff Conklin, sistemas EAD são normalmente considerados colaborativos.
4. **Hipertexto inteligente**: sistema hipertexto cuja base de conhecimento é formada por mecanismos de inferência, ou outra técnica de inteligência artificial (IA), e esse conhecimento reflete dinamicamente na formação das ligações, que têm o poder de disparar processos. Sistemas hipertextos inteligentes podem ser classificados como pequenos, grandes ou colaborativos.

3.3 Estruturas Formais Hipermídia

Ao pensarmos na estrutura de um hipertexto, precisamos nos reportar as suas definições e, também, às implementações já realizadas, para não fugir da realidade. Em um enfoque comparativo, a seguir apresentamos algumas propostas para estruturas formais de hipertexto.

Em nível de definição, como já comentado, uma propriedade estrutural básica de um sistema hipermídia é a não linearidade no sentido de não existência de uma seqüência linear obrigatória para a apresentação de um conjunto de mídias. Outro quesito fundamental é a recuperação de informações direta a um documento a partir de uma referência em outro, evitando demandas de buscas seqüenciais ou indexadas.

3.3.1 Banco de Dados

Algumas definições com visão mais tecnológica consideram os sistemas hipermídia como uma organização de banco de dados. De Bra aposta nessa definição e visão estrutural, afirmando, inclusive que os sistemas hipermídia são bancos de dados [DEB 2002]. Conforme De Bra, as informações em um sistema hipermídia não são, simplesmente, "baldes cheio de bytes"; elas são estruturadas, à semelhança da informações armazenadas na maioria dos bancos de dados. Ele afirma que, apesar da estrutura das informações ser diferente das ferramentas administrativas mais comuns para bancos de dados, os SGBD atuais são capazes de armazenar as informações utilizadas em um sistema hipermídia.

De Bra considera que a ação típica de um usuário hipermídia é um salto *(jump)*, ou um vai para *(goto)*, de uma para outra parte de um banco de dados. Essa forma de recuperação das informações é diferente da usualmente empregada tradicionalmente nos SGBD, que realiza consultas *(queries)* em diferentes partes do banco de dados, colhendo as informações desejadas, disponibilizando-as e manipulando-as conforme a aplicação. Outra consideração desse autor é que os bancos de dados contêm conexões entre informações que se relacionam, capacitando os sistemas que o gerenciam a auxiliar o usuário quando desejar pular de uma mover-se de forma não linear de uma informação para outra.

Sugere que um SGBD com recursos suficientes pode ser utilizado para armazenar e acessar informações de um sistema hipermídia se as interfaces típicas com o usuário forem substituídas por novas interfaces que suportem interação hipermídia

Nessa abstração oferecida por De Bra, fragmentos de informação, ou porções do banco de dados, são denominados nodos. Interligados, eles formam um hiperdocumento. Os nodos e as ligações podem ser visualizados na forma de um grafo [DEB 2002].

Ana C. Salgado também percebe uma forte relação entre os sistemas hipermídia e os bancos de dados e sugere características que tanto o sistema hipermídia como o banco de dados devem ter de forma a estabelecerem entre si relações otimizadas.

Conforme Salgado [SAL 92, p. 29], a utilização do conceito de hipertexto no contexto de banco de dados é simples: as janelas sobre a tela são associadas a objetos do banco de dados e existem ligações entre esses objetos e um sistema de hipertexto ideal para a manipulação de um banco de dados deve apresentar as seguintes características:

- o banco é uma rede de nós textuais (ou gráficos) que podem ser vistos comu um *hiperdocumento*;
- cada janela tem um nome e corresponde a um nó no banco. Só um certo número de nós estão "abertos" na tela ao mesmo tempo;
- podemos efetuar operações sobre as janelas (fechar, redimensionar, etc.);
- as janelas podem ter qualquer número de ligações representando apontadores para outros nós do banco;
- o usuário pode facilmente criar outros nós e outras ligações para os novos nós já existentes;
- o acesso ao banco pode ser efetuado de três maneiras:
    1. percorrendo as ligações e abrindo sucessivamente as janelas para visualização de seu conteúdo,
    2. procurando na estrutura uma palavra-chave, um valor de atributo, etc.
    3. navegando na estrutura com o auxílio de um "browser";
- ter um enfoque subjacente orientado a objetos.

Nesse enfoque, um sistema de hipertexto constitui, na realidade, na união de três aplicações: um método de acesso ao um banco de dados, um esquema de representações do tipo rede semântica e uma interface interativa orientada a objetos.

3.3.2 Grafos

Tanto De Bra como Salgado[6] referem-se à arquitetura de um sistema hipermídia como um banco de dados no ponto de vista da tecnologia e matematicamente como um grafo, que pode ser definido como $G = <N, A, f_o, f_d>$ onde [MEN 2002a, p. 71]:

1. N é um conjunto de nodos ou vértices;
2. A é um conjunto de arcos
3. $f_o, f_d : N \rightarrow A$ são funções denominadas origem e destino, respectivamente.

Note-se que é bastante natural visualizar um sistema hipermídia através de grafos dirigidos. O próprio vocabulário empregado em hipermídias adota o termo nodo para designar um fragmento de informação dentro de um hiperdocumento [DEB 2002]. As interligações entre esses fragmentos pode ser representadas através dos arcos, das funções origem e destino do grafo. Para representar os eventos que disparam a transição de um nodo para outro, seus arcos podem ser rotulados, como mostra a FIGURA 3.2.

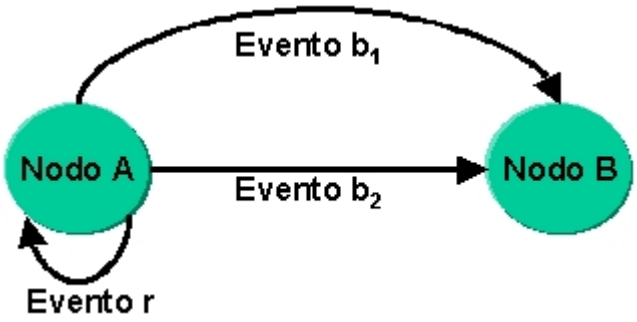


FIGURA 3.2 - Um grafo rotulado representando uma estrutura hipermídia

3.3.3 Máquina Abstrata de Hipertexto (HAM)

Campbell e Goodman propuseram em 1987 um modelo teórico de sistemas hipertexto que denominaram Hypertext Abstract Machine (HAM), que pode ser traduzida para o português como Máquina Abstrata de Hipertexto. A definição da HAM consiste na descrição de objetos HAM e de operações aplicáveis sobre esses objetos. O modelo de armazenamento do HAM é baseado em cinco tipos de objetos: grafos, contextos, nodos, ligações e atributos [CAM 88].

Os grafos são os objetos HAM de maior nível. Contém, geralmente, todas as informações sobre o tópico geral. Um grafo contém um ou mais contexto.

---

[6] Vide a definição de hipertexto por A. C. Salgado na seção Definições.

Os contextos particionam a semântica do hipertexto. Formam uma estrutura em árvore enária, na qual cada contexto pai pode ter qualquer quantidade de contextos filhos. Quando um grafo é criado, um contexto raiz inicia a árvore. Podem ser utilizados para suportar configurações, espaços privados e árvores de histórico de versões. A abstração de contextos para particionar dados foi introduzida por Shwartz e Delisle [SCH 87].

Os nodos contêm dados multimídia arbitrários. Podem ser classificado em arquivado, não arquivado ou anexado. Quando um nodo arquivado é atualizado, uma nova versão é criada, que utilizará os novos conteúdos.

Versões anteriores de um nodo arquivado podem ser recuperadas. Quando um nodo não arquivado é atualizado, os conteúdos anteriores são substituídos pelos novos, não havendo possibilidade de recuperação de versões anteriores. Quando um nodo anexado é atualizado, os novos conteúdos são anexados aos anteriores. Este último tipo é mais útil para tarefas de registro de atividades *(logging)* ou computações semelhantes. Como pode-se notar, o HAM possui um mecanismo automático de gerenciamento de histórico de versões, que é acionado cada vez que um objeto arquivado é modificado. Esse mecanismo provê operações para destruição de versões indesejadas. Os conteúdos de um nodo podem ser consultados através de expressões regulares definidas pelo usuário. O mecanismo de busca permite que todas as versões de um nodo sejam verificadas. É importante observar que esse mecanismo de busca é definido em nível de nodo e não em nível de hiperdocumento.

As ligações estabelecem relações entre os nodos. Uma ligação define um relacionamento do tipo nodo origem e nodo destino e permite navegação bidirecional. Uma ligação de cruzamento de contextos relaciona dois nodos em contextos diferentes, sendo útil para compartilhamento de informações entre dois contextos.

Atributos podem ser anexados a contextos, nodos ou ligações. Podem ser de qualquer tipo: strings, números, tipos definidos pelo usuário. Pares de atributo/valor adicionam semântica a esses objetos. Por exemplo, o conceito de âncora dos hipertextos, uma regra de transição anexada ao nodo origem, assim como os tipos de ligações, conceitos usuais na maioria dos sistemas hipertextos, que focalizam sua funcionalidade básica nas ligações, não existem como primitiva em HAM. Tais conceitos podem, entretanto, serem simulados através de atributos de ligações. Podem, também, servir como um descritor do objeto, representar propriedades específicas do objeto ou da aplicação, ou servir para outros propósitos de classificação ou documentação.

Outra aplicação usual para os atributos é possibilitar a implementação de filtros. Os mecanismos de filtragem permitem que subconjuntos de objetos HAM sejam extraídos dos grafos. Permitem que o usuário especifique predicados de visibilidade, que são expressões relacionadas às versões e aos atributos e seus valores. Podem ser aplicados para buscar contextos em um grafo, nodos e/ou ligações em um contexto.

A especificação da HAM abrange também um mecanismo denominado *Access Control List (*ACL), cuja funcionalidade é designada para prover segurança de acesso aos dados de um grafo. Adicionar um ACL a um objeto é opcional. Um ACL consiste de um usuário ou grupo de usuários e um conjunto de permissões. As permissões possíveis são acessar, anotar, atualizar e destruir. A permissão para anotar, menos comum aos sistemas de controle de acesso, possui a semântica de permitir ao usuário autorizado anexar ligações a um nodo. Possui, portanto, uma semântica de atualização restrita [CAM 88].

Como a maioria dos modelos de referência, HAM não descreve o sistema hipertexto completamente. O mecanismo situa-se entre o sistema de arquivos e a interface do usuário, conforme mostra a FIGURA 3.3.

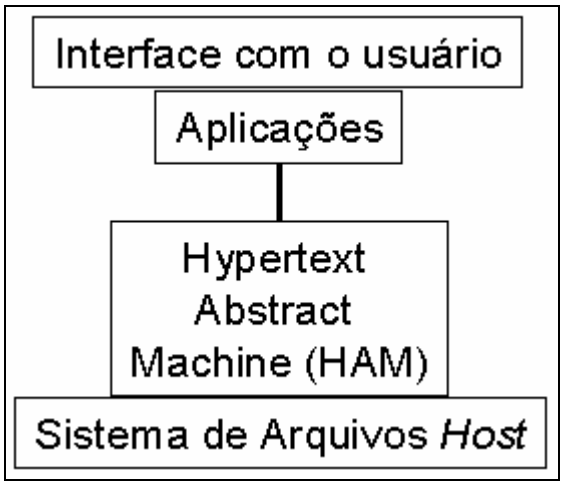


FIGURA 3.3 - Contexto operacional do HAM

3.3.4 Modelo Dexter

F. Halasz e M. Schwartz apresentaram, em 1990, uma especificação escrita em linguagem Z do Modelo Dexter[7], uma arquitetura multicamada em cinco níveis, cuja proposta pode ser compreendida como um padrão de projeto *(design pattern)* para sistemas hipertexto [DEB 2002 apud HAL 90]:

1. **Camada de Execução *(Runtime Layer)***: Nessa camada está a apresentação do hipertexto, propriamente dita. É onde o sistema é executado e ocorrem as interações com o sistema. Para exibir um componente, esta camada utiliza uma *função de instanciação* que se utiliza, também, da *especificação da apresentação*, que contém informações de como o componente será apresentado pelo sistema.

[7] dexter: adjetivo que pode significar propício, favorável, feliz, destro, hábil [TOR 42].

2. **Especificação da Apresentação *(Presentation Specificacions)***: Conforme De Bra, o mecanismo de identificação através de âncoras pode ser combinado o mecanismo de especificação dos componentes para a fim de prover uma especificação do objeto terminal de uma ligação. No modelo Dexter, explica, essa combinação é capturada através de uma entidade denominada *especificador*, que consiste em um identificador de âncora e dois campos: um *endereçamento* e uma *especificação da apresentação*. Um *especificador* define um componente e um local para ancoramento dentro de um componente que serve como destino de uma ligação. O *endereçamento* informa se o terminal se trata de uma origem, um destino, origem e destino de uma ligação ou nem origem, nem destino. Os códigos de endereçamento para essas quatro possibilidades no modelo Dexter são FROM, TO, BIDIRECT e NONE, respectivamente. A especificação da apresentação faz a comunicação entre as camadas de execução e de armazenamento.

3. **Camada de Armazenamento *(Storage Layer)***: De Bra afirma também que o foco do modelo Dexter é esta camada, que modela a estrutura básica da rede de nodos e ligações. A camada de armazenamento descreve um "banco de dados" estruturado de modo hierárquico, composto de *componentes* que contêm dados, os nodos, conectados diretamente entre si através de ligações.

4. **Ancoramento *(Anchoring)***: O modelo Dexter provê um identificador único (UID) para cada componente. Ao se descrever os endereçamentos da camada de apresentação é possível identificar, também, subestruturas dentro dos componente. No modelo Dexter, os componentes são estruturas que, apesar de atômicas, que podem possuir endereçamentos para suas subestruturas internas através de entidades denominadas *âncoras*. Uma âncora possui duas partes: um identificador *(anchor id)* e um valor *(anchor value)*. O identificador de uma âncora identifica-a exclusivamente dentro do escopo do seu componente. As âncoras podem, portanto, serem identificadas exclusivamente em um ambiente hipertexto através do identificador componente e de seu próprio identificador. A especialização do identificador de uma âncora em duas partes atribui às âncoras facilidades em nível de manutenção, permitindo que alterações nas estruturas ou subestruturas de um componente mantenham a semântica anterior das ligações, se assim desejado. O valor da âncora é um valor qualquer, arbitrário, que especifica alguma localização, região, item ou subestrutura dentro de um componente.

5. **Camada Interna dos Componentes *(Within Component Layer)***: Esta camada se preocupa com a estrutura interna de um componente (ou nodo). Dada a complexidade crescente dos sistemas hipertexto transformando-se em sistemas multimídia complexos, o modelo Dexter se preocupou especialmente com cada nodo, permitindo que os nodos tenham sua própria estrutura independente de um sistema maior que o gerencie. Tal estrutura facilita modularizações e aninhamentos. Facilita, também, a percepção de cada componente como uma caixa preta, sugerindo a adoção de tecnologias que valorizem a reusabilidade e a manutenibilidade.

3.3.5 Modelo em Treliça (Trellis)

O Modelo em Treliça[8] (Trellis), cujo nome foi inspirado na engenharia civil, é proposto por P. David Stotts e Richard Furuta. O projeto começou em 1988 e continua em vigor [STO 2002]. O primeiro artigo sobre o Trellis, apresentado em 1989, mostra a preocupação dos autores quanto à semântica concorrente possível de se obter através de um sistema hipertexto baseado em redes de Petri[9] [STO 89]. Nesse artigo, os autores definem o Trellis como um modelo de hipertexto que utiliza estruturas e semântica de execução das redes de Petri para especificar tanto o seu modo de ligação como a sua semântica de navegação.

Um sistema hipertexto assim modelado é definido como uma sêxtupla $H=<N,C,W,B,P_l,P_d>$ onde:

a) $N=<S,T,F>$ é uma estrutura de rede Petri

b) C é um conjunto de conteúdos de documentos

c) W é um conjunto de janelas

d) B é um conjunto de botões

e) P_l é uma projeção lógica da rede de Petri para o documento

f) P_d é uma projeção da rede de Petri para exibição do documento

Um hipertexto consiste em uma rede de Petri que representa a estrutura interligada de documentos, conjuntos de componentes IHC (conteúdos, janelas e botões) e dois conjuntos de mapeamentos (projeções), uma entre a rede de Petri e os componentes IHC e outra entre a rede de Petri e o mecanismo de exibição. Stotts e Furuta consideram as redes de Petri como um grafo dirigido e bipartido, aproximando sua definição de outras já apresentadas, com a diferença que as redes de Petri focalizam a semântica concorrente. Uma janela do conjunto W é um local logicamente distinto de informação, que não necessariamente é visível. Uma determinada janela pode requerer recursos para manipular conteúdos multimídia. Um botão do conjunto B é um evento que tem o poder de provocar no mecanismo corrente de exibição alguma mudança. Um conteúdo do conjunto C é uma mídia qualquer: um texto, uma imagem, um vídeo, um programa, um outro sistema hipertexto, etc. A FIGURA 3.4 mostra um exemplo de estrutura de rede de Petri para sistemas hipermídia.

---

[8] Treliça (Do fr. treillis) Sistema de vigas cruzadas empregado no travejamento das pontes [FER96a].

[9] Rede de Petri é um modelo computacional concorrente bastante utilizado na Ciência da Computação e nas engenhariais [MEN 2002[a], pp 273-275].

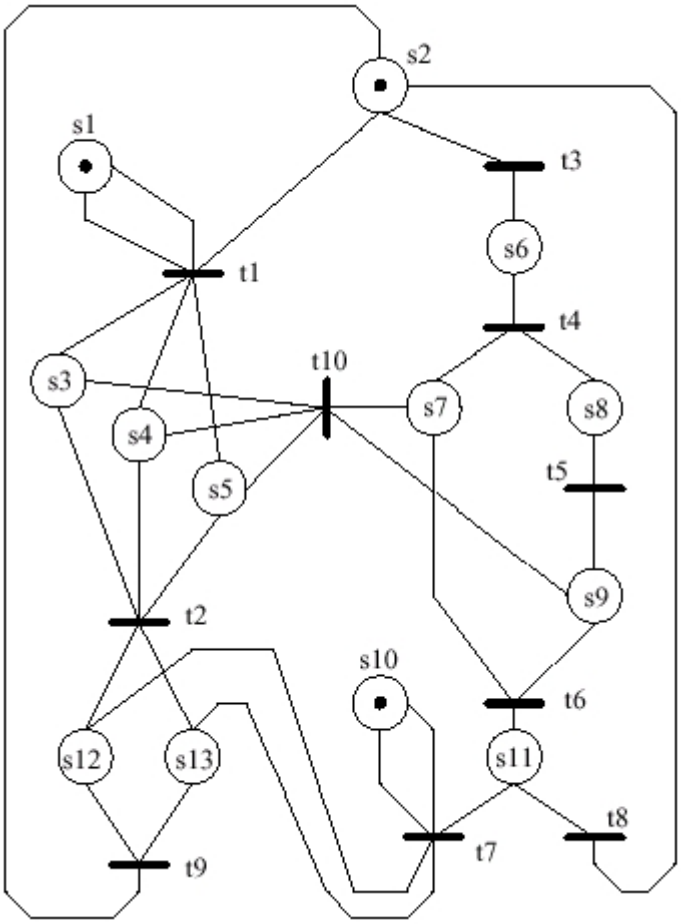


FIGURA 3.4 - Uma estrutura de rede de Petri para sistemas hipermídia

3.3.6 Modelo de Referência Hipertexto Trellis (Modelo-r)

Furuta e Stotts apresentaram em 1990 um metamodelo para descrição de sistemas baseados no modelo Trellis denominado Modelo de Referência Hipertexto Trellis *(Trellis Hypertext Reference Model)*, que resolveram abreviá-lo, denominando-o simplesmente de modelo-r *(r-model)* [FUR 90].

Esse metamodelo também foi definido em cinco níveis lógicos, conforme mostra a FIGURA 3.5:

1. abstrato de componente *(abstract component level)*
2. abstrato de hipertexto *(abstract hypertext level)*
3. concreto de hipertexto *(concrete hypertext level)*
4. concreto contextual *(concrete context level)*
5. visível de hipertexto *(visible hypertext level)*

É interessante notar que essa estrutura apresenta-se hierárquica e que, em um nível mais primitivo, é abstrata, formando, por herança, estruturas concretas, em níveis mais especializados. A cultura de desenvolvimento de software atual sugere tal estruturação através do paradigma da orientação a objetos, pois a mesma possui propriedades estruturais abstratas e concretas, bem como um sofisticado mecanismo de herança. O modelo é agrupado em três categorias, abstrata, concreta e visível conforme recomenda Shaw para documentos impressos [DEB 02 apud SHA 80].

De Bra, ao analisar o Modelo-r, percebeu a ausência de tratamento para os seguintes aspectos:

- **semântica de navegação:** conforme a aplicação, pode-se desejar uma determinada semântica de navegação. Por isso, não é considerado em nível de metamodelagem, deixando-se essa característica para ser definida durante a modelagem do sistema que será construído.

- **comportamento dinâmico:** o modelo-r assume que a navegação através das ligações requer eventos disparados pelo usuário. Alguns sistemas hipermídia podem, entretanto, requerer a possibilidade de transitar de uma estrutura para outra sem a necessidade de haver uma intervenção do usuário. Em especial, sistemas evolutivos ou adaptativos de inteligência artificial podem requerer esse modo de transição de estado. De Bra ressalta que, nesta análise, focaliza-se o comportamento dinâmico do sistema em si e não dos seus conteúdos disponibilizados.

- **características de conteúdo:** alguns sistemas simplesmente restringem o tamanho dos conteúdos nos nodos, alguns facultam o uso de janelas rolantes e outros incrementam a rolagem automaticamente ao se perceber que um determinado conteúdo tem tamanho maior que sua janela.

- **descrições em nível físico:** especificação de formatos de armazenamento e de comunicação entre diferentes sistemas hipermídia.

- **interfaces com o usuário:** quais serão utilizadas para interagir com os usuários do sistema.

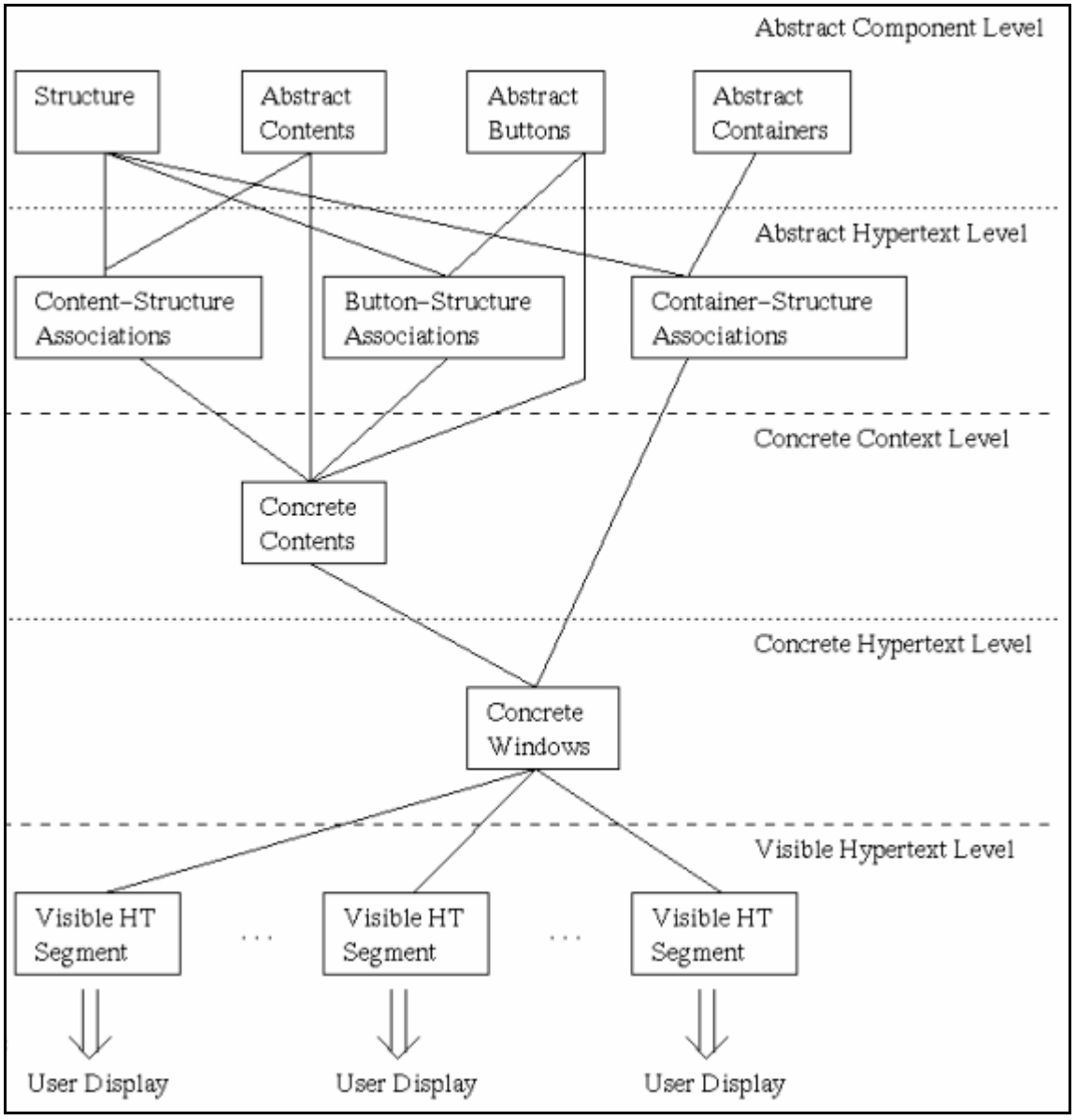


FIGURA 3.5 - Os cinco níveis lógicos do modelo-r para Trellis

3.4 Animações por Computador

Conforme Thalmann, uma animação por computador consiste em "modificar uma cena no tempo [THA 91]." Cenas tridimensionais, por exemplo, podem ser compostas por três tipos de entidade: objetos, câmaras e luzes. Uma animação também pode ser definida como um processo no qual cada quadro de vídeo[10] *(frame)* em um filme é produzido individualmente, causando-nos uma sensação de movimento. Para que nossos olhos e cérebros percebam animações como movimentos contínuos e ininterruptos necessita-se que as seqüências de quadros seja atualizada a uma taxa de, pelo menos, 30 quadros por segundo. Abaixo desta taxa, a maioria das pessoas podem detectar *flickers*[11] que depreciam o realismo. Em um limite superior, taxas acima de 70 quadros por segundo, o ser humano não percebe melhorias devido às características funcionais dos olhos e do cérebro. A razão pela qual acima desta taxa *flickers* não são percebidos deve-se à persistência da visão *(persistence of vision)*. De tempo em tempo, o cérebro armazena as imagens recebidas através dos olhos por uma fração de segundo e, automaticamente, atenua sobressaltos menores [WIK 2002].

Existem várias técnicas para se obter uma animação, com ou sem o uso de computadores. Nos casos de uso de computadores, são objeto de estudo especializado especialmente na área da computação gráfica. O processo de construção de uma animação computadorizada pode incluir as seguintes etapas [ACC 2002 apud PUE 88, THA 91, THA 85, FOL 90]:

- **Pré-processamento**: Trata do planejamento da animação através de esboços seqüenciais *(storyboards)*, que objetiva abstrair os elementos principais da animação, sem porém entrar em detalhes exaustivos. Esta fase é necessária em projetos complexos, podendo ser descartada em animações mais simples.
- **Edição de cenas e modelagem de objetos**: Nesta etapa, os objetos são modelados em detalhes e combinados entre si para comporem as cenas.
- **Animação**: Variações dos objetos são determinadas e calculadas para gerar a seqüência desejada. Os movimentos dos objetos na animação podem ser descritos através de diversos métodos de controle de movimento e em níveis de abstração diferentes. Esses métodos vão desde o controle explícito, no qual o animador é responsável pela descrição dos atributos posicionais dos objetos, até controles altamente automatizados através de sistemas baseados em conhecimento.
- **Renderização *(rendering)* das imagens**: As imagens finais de cada quadro são obtidas através da computação das propriedades visuais dos objetos na cena. Nesta fase são levados em conta diversos parâmetros como iluminação, posicionamento de câmaras, computação de sombras, transparências, faces ocultas, texturas, etc.
- **Pós-processamento**: A seqüência de imagens geradas é sincronizada com sons e gravada em outro tipo de mídia, caso necessário. Nas animações

[10] Um quadro *(frame)*, no contexto das animações, corresponde a uma imagem completa.

[11] Em processamento de imagens, efeitos de tremulação, vibração ou perda de características que ocorrem em um dispositivo de saída visual devido à baixa taxa de renovação da imagem ou corrupção de sinal.

destinadas a filmes, a seqüência é combinada com outras cenas e gravada em película.

- **Análise de resultados**: As características visuais reveladas pela animação são exploradas com o objetivo de extrair informações sobre os modelos utilizados. Essa fase é bastante útil em animações destinadas à simulação de processos naturais e industriais.

Interessante notar que a montagem de uma animação por computador possui etapas semelhantes a um projeto convencional de software. O pré-processamento pode ser comparado à análise preliminar. A edição de cenas e a modelagem de objetos nas animações eqüivalem à etapa de projeto do software. A animação e a renderização correspondem à fase de construção do software. O pós-processamento tem semelhanças com o processo de integração de módulos e a análise de resultados com a busca de informações elaboradas.

3.4.1 Graphics Interchange Format Animados (GIF 89a)

Um dos protocolos mais utilizados para animações curtas na Internet, formadas por seqüências de quadros, é o GIF Animado, também denominado GIF versão 89a por seu desenvolvedor, a empresa Compuserve.

Uma questão levantada sobre esse protocolo, tanto na academia como na indústria, é a ação judicial iniciada pela empresa Unisys em 1994 contra a Compuserve. A Unisys reclama quebra de patente sobre a tecnologia de compressão Lempel-Ziv-Welch (LZW), utilizada pelo protocolo GIF, podendo refletir em outras questões jurídicas aos desenvolvedores que utilizam esse protocolo em caso de ganho de causa a favor da Unisys [MIT 2002]. Conforme informações contidas no dicionário Foldoc, o vice-presidente da Compuserve possui o compromisso de "manter a especificação GIF 89a aberta, com suporte integral e não proprietária para toda a comunidade *on-line*, incluindo a *Wold-Wide Web* [HOW 2002]."

Accorsi ilustra a estrutura de um GIF 89a conforme mostra a FIGURA 3.6 e escreve:

> "Todo arquivo GIF possui um cabeçalho de identificação, um descritor de tela com os parâmetros necessários para definir os recursos de visualização, e é finalizado com um bloco termindor. As tabelas de cor utilizadas como referência pelas imagens, tanto global quanto local, são opcionais. Em geral, para manter o arquivo menor, as imagens contidas na animação fazem referência apenas à tabela de cor global, eliminando assim, as tabelas locais. Entre a tabela de cor global e o bloco finalizador podem existir vários blocos gráficos ou de propósito geral. Os blocos gráficos são utilizados para armazenar as informações relativas aos quadros da animação ou textos. A extensão de controle gráfico possui informações quanto ao tempo de permanência do próximo elemento gráfico e também indica a maneira como este deve ser tratado após ter sido mostrado... Os programas que geram para o formato GIF procuram codificar somente a área de imagem que foi alterada de um quadro para outro, utilizando o descritor de imagem para armazenar as dimensões e coordenadas da área selecionada [ACC 2002, p. 21-22]."

Uma informação interessante relativa à estrutura de um arquivo GIF 89a, no que se refere aos objetivos e metas do *Hyper Seed*, é sua possibilidade de unir textos às imagens nas animações. GIFs animados, porém, não suportam interatividade.

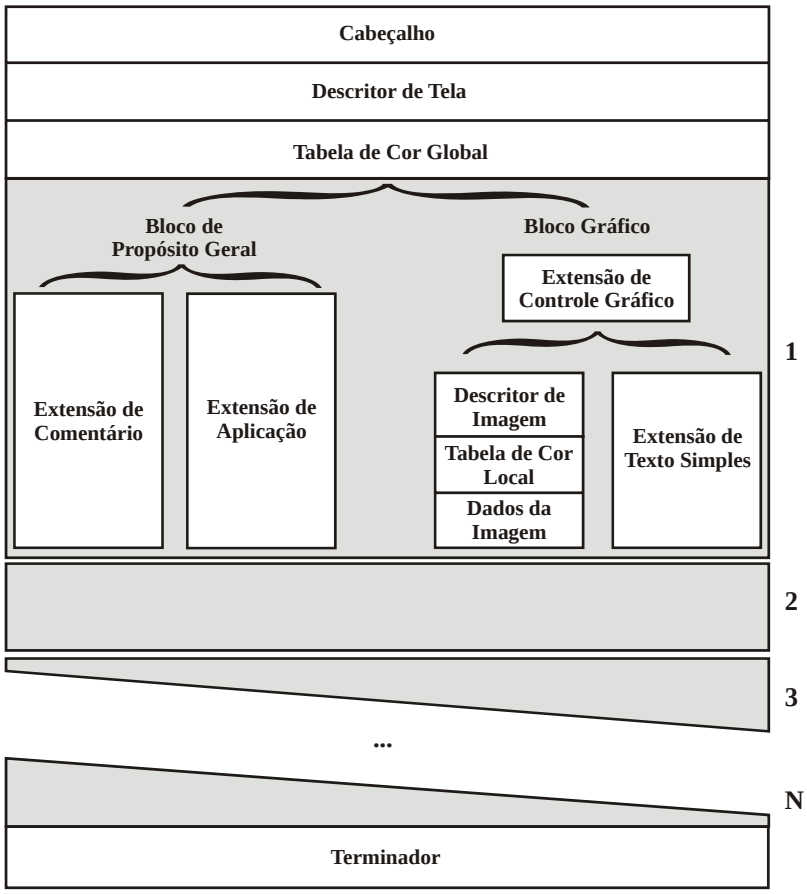


FIGURA 3.6 - Estrutura do protocolo GIF 89a.

3.4.2 Flash

Flash é uma linguagem desenvolvida pela empresa *Macromedia* que utiliza vetores gráficos para definição de objetos, sendo o padrão de fato para a Internet utilizando-se esta abordagem computacional. Suporta interatividade, textos, imagens, movimentos na tela. A interatividade com o usuário é realizada através de códigos escritos em *ActionScript*, linguagem *script*[12] desenvolvida especialmente para comunicação com a linguagem *Flash*. Uma técnica de computação gráfica bastante utilizada por desenvolvedores *Flash* é a *tweening*[13].

Entre os vários efeitos visuais possíveis de se construir em *Flash* através da técnica de *tweening*, podemos citar:

- **Tweenings de movimento *(Motion Guide Tweenings)***: Tipo de tweenings que simula um movimento de um objeto de um ponto gráfico até outro, conforme uma trajetória retilínea ou customizada.
- **Tweenings de coloração *(Tint Tweenings)***: Executa uma mudança gradual na coloração do objeto.
- **Tweenings de forma *(Shape Tweenings)***: Mudança gradual na forma de um objeto. Por exemplo, trocar gradualmente o número 0 para o número 1, ocasionando um efeito morfológico para representar um incremento.

Suas características computacionais, que utilizam vetores gráficos e técnicas de interpolação para a criação de quadros, permitem que animações *Flash* gerem uma grande quantidade de quadros com código pequeno e reuso de mídias. Como conseqüência, tende a demandar pouco tráfego de bits da máquina servidora para a cliente, uma vantagem desejável em ambientes de rede como a Internet, pois torna sua execução mais rápida e menos custosa.

Para se construir programas em *Flash*, é necessário licenciar seu ambiente de desenvolvimento, atualmente denominado *Macromedia Flash* MX. Sua distribuição é gratuita e requer, para execução do código, *plug-in* também gratuito instalado no navegador que os executam.

---

[12] O termo script é aplicado a linguagens fracamente tipadas, geralmente interpretadas, que possuem estruturas de dados simples que servem, tipicamente, como meio de interação *(interface)* com outras linguagens [IMP 2003, MAC 2002a, p. 53].

[13] Técnica de interpolação na qual um programa de animação gera automaticamente quadros intermediários (denominados *in between*) entre quadros chaves *(key frames)*. Esta técnica foi introduzida por Burtnyk e Wein em 1971 [THA 89, IMP 2003].

3.4.3 SMIL Animation

Os grupos de trabalho do *World Wide Web Consortium* (W3C) responsáveis pelas especificações das linguagens *Scalable Vector Graphics* (SVG) e *Synchronized Multimedia Integration Language* (SMIL), em colaboração criaram a recomendação SMIL Animation em setembro de 2001 com o objetivo de especificar:

> "... funcionarlidade de animações para documentos XML. A recomendação descreve uma estrutura *(framework)* para se construir animações assim como um conjunto de elementos básicos em XML adequados para a integração com documentos XML. É baseada no modelo de temporização de SMIL 1.0, com algumas extensões, sendo, na verdade, um subconjunto de SMIL 2.0. Provê um passo intermediário importante em termos de complexidade de implementação para aplicações que desejam ter animações compatíveis com SMIL mas que não necessitam de contêineres *(containers)* de tempo [W3C 2003c]".

No contexto *Hyper Seed*, SMIL Animation possui características de especificação importantes, sendo baseada em XML, permitindo a contrução de animações, textos e imagens dinamicamente, como veremos a seguir no estudo de conceitos e tecnologias relacionadas às hiper-animações.

3.5 Hiper-Animações

Em 1991, F. Kappe propôs uma tese de doutorado na qual apresentou o conceito de hiper-animação[14]:

> "... um novo conceito para a criação de animações interativas de tempo real. Trata-se, essencialmente, da combinação das tecnologias da Computação Gráfica e Hipermídia, a qual denominamos *Hiper-Animação*... Uma hiper-animação não se limita às animações tradicionais: combina imagens, vídeos digitais, animações gráficas e sons com dispositivos de entrada não usuais a fim de realizar um alto nível de interatividade... ligações de uma seqüência de animação para todos os tipos de informação, inclusive outras seqüências de animação, podem ser definidas... O conceito de Hiper-Animação possui uma abordagem genérica, permitindo a criação de seqüências animadas complexas (em termos de complexidade e quantidade de objetos, complexidade de movimentos) e - o mais importante – que podem ser editadas convenientemente [KAP 91, p. 101]."

[14] Conforme o autor, as linhas gerais do conceito de hiper-animações foram apresentadas em [KAP 90].

Kappe denomina seqüências simples de animação como “grupo de documentos” *(document cluster)*, e a visita a um grupo de documentos é definida como um roteiro *(tour)*. Seu texto apresenta o exemplo de hiper-animação mostrado na FIGURA 3.7, onde um grupo de documentos é utilizado para construir um roteiro de material instrucional sobre circuitos elétricos. O exemplo da figura poderia ser associado a um texto com orientações do tipo “Pressione o botão A para ver o que acontece quando a chave S_1 é fechada”, “Pressione o botão B para ver o que acontece quando a chave S_1 é aberta” e “Pressione o botão C para concluir a demonstração”. O grupo de documentos que constitui a hiper-animação associada a essas orientações é formado de 3 documentos: um do tipo “desenho” *(HA_drawing)* e dois do tipo “animação” *(HA_animation)*. Esses documentos são interligados através de ligações de grupo *(cluster links)* definidas em [KAP 91, p. 19-21]. Na ativação, apenas o desenho é disponibilizado. Quando o usuário pressiona o botão A ou B, a animação correspondente é apresentada. O botão C ativa uma ligação para o próximo grupo de documentos. O autor acrescenta que sons podem ser sincronizados com as animações.

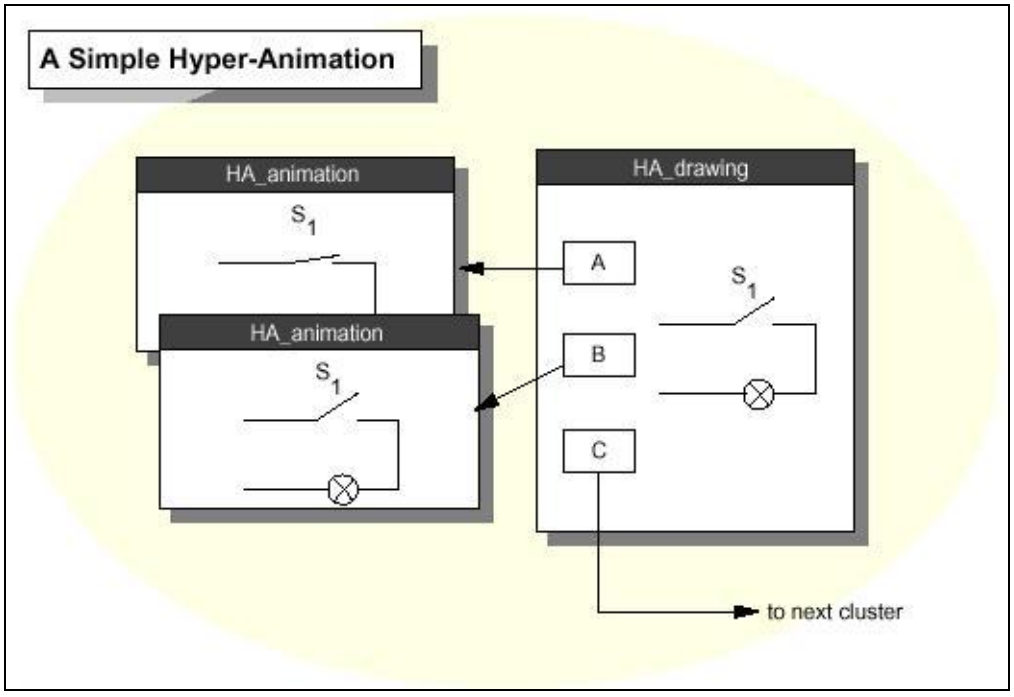


FIGURA 3.7 - Exemplo simples de hiper-animação

3.5.1 Hyper-G

Kappe apresenta hiper-animações associadas ao ambiente Hyper-G: “Uma característica chave de uma hiper-animação é que ela pode ser integrada ao ambiente Hyper-G [KAP 91, p. 101].” Tal abordagem se assemelha à apresentada por esta pesquisa, onde hiper-animações são construídas através de applets, podendo ser integradas a um ambiente maior, a *World Wide Web* (WWW).

Como mídia, uma característica importante das hiper-animações é sua alta capacidade interativa. Heller [HEL 2001, p. 15-16] classifica a interatividade das mídias em quatro categorias: passivas, reativas, pró-ativas e diretivas. No que se refere a mídias

que expressam movimento *(motion media)*, como filmes e animações, mídias passivas são aquelas que não habilitam nenhum controle para o usuário; todo o controle está embutido na aplicação e sua apresentação é seqüencial. As reativas permitem interações predefinidas e limitadas ao usuário, geralmente algumas adequações de valores, como, por exemplo, um ajuste de volume em uma mídia sonora. As pró-ativas permitem que o usuário tenha um controle mais efetivo sobre a mídia, permitindo que desempenhe um papel mais destacado no projeto e na construção de situações. Em animações, o usuário poderá determinar ações do tipo iniciar, parar, pausar, avançar, voltar, etc. As mídias diretivas , mais abrangentes, permitem que o usuário personalize aspectos mais fundamentais. No caso específico das mídias com movimento, Heller considera importante que uma mídia com poderes diretivos de interatividade permita que o usuário crie seqüências animadas.

Devido às alternativas de apresentação de conteúdo oferecidas por suas hiperligações, as hiper-animações pode ser consideradas mídias com nível de interatividade diretivo, uma vez que diferentes seqüências de animações podem ser definidas pelo usuário ao - literalmente - navegar sobre uma hiper-animação.

3.5.2 Hiper-Animações Dinâmicas via W3C

É possível definirmos hiper-animações em um ambiente dinâmico, para Internet, através da tecnologia oferecida, atualmente, pelo W3C.

3.5.2.1 SVG

Em SVG, o usuário pode disparar hiperligações através do elemento "a" *(anchor)*, análogo ao elemento "a" da linguagem HTML. SVG utiliza a linguagem de ligação XML (XLink) para todas as definições de ligações no elemento "a". A especificação atual da SVG requer suporte, apenas, para ligações simples XLink *(XLink simple links)*. Essas ligações associam exatamente dois recursos, um local e outro remoto, com um arco saindo do local e chegando no remoto, conforme mostra a FIGURA 3.8 [W3C 2003a].

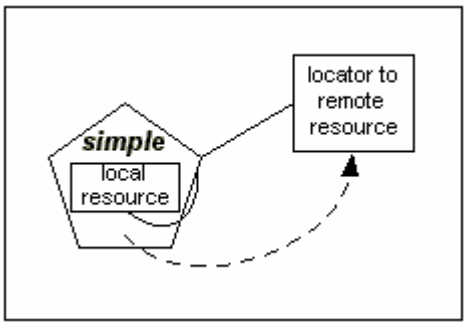


FIGURA 3.8 - Exemplo de ligação simples XLink

Um exemplo simples (não animado) em SVG com hiperligação é mostrado na FIGURA 3.9, onde um arquivo SVG associa uma hiperligação a uma elipse. Se esse arquivo for renderizado por um software que suporta SVG e HTML, ao se clicar na elipse, a janela *(window)* ou quadro *(frame)* corrente será substituída pela página inicial *(home page)* do W3C.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="5cm" height="3cm" viewBox="0 0 5 3"
 version="1.1"
 xmlns="http://www.w3.org/2000/svg"
 xmlns:XLink="http://www.w3.org/1999/XLink">
  <desc>Example link01 - a link on an ellipse</desc>
  <rect x=".01" y=".01" width="4.98" height="2.98"
   fill="none" stroke="blue"  stroke-width=".03"/>
  <a XLink:href="http://www.w3.org">
    <ellipse cx="2.5" cy="1.5" rx="2" ry="1" fill="red"/>
  </a>
</svg>
```

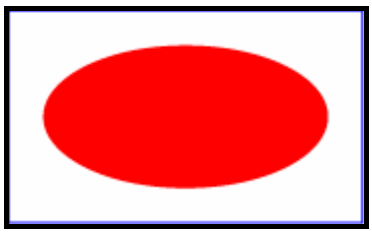


FIGURA 3.9 - Elipse com hiperligação definida em SVG.

3.5.2.2 SMIL

Em SMIL, já podemos notar a combinação entre animações e hipertecnologia. No trecho de código mostrado na FIGURA 3.10, as animações “A” e “B” estão configuradas para iniciar, respectivamente, em 10 e 15 segundos. “C” e “D” não iniciam imediatamente ao se carregar o documento. Suponha que “C” é clicado aos 25 segundos de apresentação. Tal ação fará com que “D” inicie após 5 segundos.

```
<animate id="A" begin="10s" .../>
<animate id="B" begin="A.begin+5s" .../>
<animate id="C" begin="click" .../>
<animate id="D" begin="C.begin+5s" .../>
...
<a href="#D">Start the last animation</a>
```

FIGURA 3.10 - Código em SMIL Animation com hiperligação.

3.5.3 SMIL Animation

No contexto da linguagem SMIL Animation, que combina as tecnologias SMIL, SVG e XLink, a W3C define animação como:

> "Uma animação é uma manipulação temporal de um elemento destino *(target element)*. Mais especificamente, a manipulação de atributo destino *(target attribute)* de um elemento destino. A animação define o mapeamento de tempos para o atributo destino. Esse mapeamento considera todos os aspectos temporais e, também, da semântiva relativa à animação.
>
> As animações especificam um início e uma duração simples *(simple duration)* que pode ser repetida. Cada animação define uma função de animação *(animation function)* que produz um valor para o atributo destino, para qualquer tempo definido no intervalo de duração simples. O autor pode especificar o tempo de duração e quantas vezes uma função de animação pode ser repetida. A duração simples combinada com um comportamento de repetição qualquer define o tempo ativo de duração *(active duration)* [W3C 2003c]."

3.5.4 Hiper-Animações via *Flash*

Ao tempo que SVG, SMIL, SMIL *Animation*, XLink, XML, XHTML, CSS e outras tecnologias formam, em conjunto, o padrão aberto W3C para Internet, o produto *Macromedia Flash* é o padrão *ad hoc* para animações nessa mesma plataforma. Sua licença de uso é proprietária e de código fechado. No que se refere à interatividade através de hiperligações, tema central de nossa pesquisa, o *Flash* oferece um conjunto limitado de opções ao programador. Mais especificamente, apenas um método da linguagem *ActionScript* que descreve os objetos em *Flash* especifica hiperligações, o método actionGetURL(*url*, *target*), onde *url* é o endereço URL a ser requisitado e *target* é a ação a ser executada sobre o conteúdo da URL indicada. Se *target* for "_level0", indica que o conteúdo da URL, geralmente outra animação *Flash*, deve substituir a animação em execução; se for "_level1", indica que o conteúdo da URL deve ser disposto sobre a animação em execução.

3.6 Conclusões

A principal conclusão deste capítulo é que as hiper-animações, propostas por Kappe, demonstraram ser adequadas, conceitualmente, à necessidade de unificar funcionalidades de ambiente hipermídia e animações.

Durante o desenvolvimento desta pesquisa, em diversos momentos surgiram dúvidas se hiper-animações são ambientes operacionais ou mídias, uma vez que, no que se refere à interatividade na recuperação de informações, trata-se de um ambiente operacional e, no que se refere ao modelo computacional gráfico, trata-se de uma mídia temporizada e sincronizada que atualiza quadros de tempos em tempos conforme uma determinada taxa de apresentação *(frame rate)*. Consideramos ambas as percepções válidas, sendo as hiper-animações ao mesmo tempo ambiente operacional e mídia.

Suas características de ambiente operacional mais fortes são: a interatividade através de hiperligações e a justaposição de variados meios visuais, acrescidos de sons, formando um contexto típico de operação de um sistema hipermídia. Percebidas como mídias, tratam-se de recursos computacionais poderosos, capazes de transmitir ao ser humano informações audio-visuais das mais variadas formas (textos, imagens, sons e animações), acrescido de meios de navegação e recuperação de informações através de hiperligações. No que se refere à percepção da mídia pelo usuário, Helleer afirma: "Mídias não-audíveis possuem baixa detectabilidade, sendo os textos classificados com os índices mais baixos e imagens em nível médio. A detectabilidade dos sons nivela-se de médio para alto, e os movimentos o maior nível [HEL 95, p. 40]." São, por essas razões, especialmente úteis em programas de educação a distância.

4 Astrha/M (*Model* – Modelo Estrutural)

4.1 Resumo

Na revisão bibliográfica do capítulo anterior, pudemos perceber um conceito que agrega características de ambiente hipermídia e também de animações: as hiper-animações, propostas por Kappe. Tal conceito é conveniente para as necessidades do *Hyper Seed*, que propõe-se a unificar conceitos de animações e hipertecnologia em torno de estruturas de autômatos finitos com saída (AFS).

Este capítulo desenvolve um modelo de Máquina de Mealy que expresse, sintaticamente, uma hiper-animação com algumas características consideradas convenientes em termos semânticos: reflexividade, não-determinismo (de estado e de palavra de saída) e conjunto finito de palavras de saída .

4.2 Modelo de Máquina de Mealy para Hiper-Animações

Um AFS pode ser, classicamente, uma Máquina de Mealy ou uma Máquina de Moore. Utilizamos nesta pesquisa a Máquina de Mealy reflexiva e não-determinístisca como modelo estrutural e remetemos a trabalhos futuros sua extensão para máquinas de Moore. A opção por Mealy deve-se, principalmente, aos experimentos já realizados no PPGC, nos quais os exemplos criados em Mealy exigiram menos estados do que em Moore, tendendo, assim, a uma menor complexidade descritiva.

Sobre a aplicabilidade em si de um AFS para a descrição de soluções computacionais, conforme Menezes:

> "Uma aplicação comum e freqüentemente recomendada para os autômatos com saída é o projeto de diálogo entre um programa (de computador) e seu usuário. Basicamente, um diálogo pode ser de dois tipos:
>
> - comandado pelo programa;
> - comandado pelo usuário.
>
> Em qualquer caso, uma das principais dificuldades do projetista é a visualização do conjunto de eventos e ações que definam um diálogo adequado para as diversas situações possíveis. [MEN 2001, p. 80]."

No caso do ambiente proposto, onde os enlaces entre diversas mídias, em diversos espaços, tempos e seqüências são permitidos, sua definição como AFS fornece mecanismos de controle sobre os estados da máquina e sobre os fluxos de execução *(traces)* a serem programados. E, sendo o modelo teórico básico da qual um ambiente poderá ser construído, tais características serão absorvidas desde a fase de análise, passando por todas as etapas de um processo de desenvolvimento.

Formalmente, uma Máquina de Mealy não-determinística e reflexiva [MEN 2001, p. 46-51], especializada com a finalidade de expressar a semântica de uma hiper-animação, pode ser representada como uma 6-upla:

$M = (\Sigma_\tau, Q, q0, \Delta, \Omega, \lambda)$ onde:

a) Σ_τ é um conjunto apontado finito de símbolos de entrada, onde é τ é o elemento distinguido.

b) Q é um conjunto finito de estados.

c) q0 é estado inicial, o qual pertence a Q.

d) Δ é um conjunto finito de símbolos de saída , denotando fontes de mídias.

e) Ω é um conjunto finito de palavras de saída, tal que $\Omega \subset \Delta^* \wedge \varepsilon \in \Omega$.

f) λ é uma função parcial de transição $\lambda : Q \times \Sigma_\tau \rightarrow 2^{(Q \times \Omega)}$, tal que $\forall q \in Q$ $\lambda(q, \tau) = \{(q, \varepsilon)\}$

FIGURA 4.1 - Máquina de Mealy especializada para estruturar hiper-animações

Note-se que, em Astrha/M, as palavras de saída são predefinidas, reduzindo seu conjunto a uma quantidade de elementos finitos, com o objetivo de definir com clareza quais são as palavras de saída desejadas para o ambiente.

É necessário que os símbolos de entrada formem um conjunto apontado [MEN 2002a, p. 109] para definir o elemento distinguido τ que levará uma transição a ser reflexiva, cujo próximo estado é o estado atual e cuja palavra de saída (ε) é vazia. As transições reflexivas têm a semântica de inoperabilidade (*no operation*, ou *nop*), onde as computações realizadas durante a transição não são percebidas externamente.

4.3 Não-Determinismo

O modelo apresentado pode apresentar não determinismo tanto em nível de estado como em nível de palavra de saída. Para exemplificar, seja μ uma Máquina de Mealy da FIGURA 4.1 onde:

a) $Q = \{q1, q2\}$

b) $\Sigma_\tau = \{\sigma1, \sigma2\}$

c) $\lambda = \{\lambda1, \lambda2\}$

d) $\Omega = \{\omega1, \omega2\}$

Suponhamos $\lambda1$ e $\lambda2$ com a seguinte funcionalidade:

$\lambda1(q1, \sigma1) = \{(q1, \omega1)\}$

$\lambda2(q1, \sigma1) = \{(q2, \omega1)\}$

Podemos observar, aqui, um não-determinismo em nível de estado.

Suponhamos agora $\lambda1$ e $\lambda2$ como segue:

$\lambda1(q1, \sigma1) = \{(q2, \varpi1)\}$

$\lambda2(q1, \sigma1) = \{(q2, \omega2)\}$

Neste caso, um não-determinismo em nível de palavra de saída.

Podemos ter, também, $\lambda1$ e $\lambda2$ definindo um nível maior de não-determinismo:

$\lambda1(q1, \sigma1) = \{(q1, \varpi1)\}$

$\lambda2(q1, \sigma1) = \{(q2, \omega2)\}$

Temos, neste último caso, não-determinismo tanto em nível de estado como em nível de palavra de saída.

4.4 Conclusões

O modelo de autômato Astrha/M foi definido especialmente para definir hiper-animações, o que facilitará a construção de uma linguagem de quarta geração projetada para esta pesquisa. Seus símbolos de saída denotam mídias e as palavras de saída são especificadas em um conjunto finito para melhor controle da semântica desejada.

Uma de suas características funcionais é o não-determinismo de estado e de palavra de saída com semântica de pseudo-aleatoriedade, o que permite modelar escolhas pseudo-aleatórias de hiper-animações de forma simples.

Outra característica do modelo é a presença de um elemento distinguido no conjunto finito de símbolos de entrada serve para expressar reflexividade em transições, expressando não operabilidade aparente.

Tais características proporcionam propriedades que, traduzidas em código, possuem semântica bem definida e especialmente útil para estruturar hiper-animações em um ambiente gráfico, multimídia e dinâmico.

5 Astrha/L (*Language* – Linguagem)

5.1 Resumo

O trabalho de pesquisa proposto passa pela especificação de uma nova linguagem, de quarta geração[15] (direcionada à expressar hiper-animações) denominada Astrha/L, definida a partir da metalinguagem XML [W3C 2002a] e composta por quatro dialetos: *Environment* (Ambiente), *Hyper* (Hiper), *Mealy* (Mealy) e *Style* (Estilo).

A definição do escopo da linguagem, paradigma e tecnologia foram norteados por objetivos de usabilidade, portabilidade e manutenibilidade, características de qualidade de software que a norma ISO/IEC 9126 [ISO 91] considera de primeira grandeza.

5.2 Características Gerais

A linguagem Astrha/L almeja contribuir, cientificamente, através de suas características listadas a seguir:

- o uso da Máquina de Mealy da FIGURA 4.1 para definir marcas do dialeto Mealy;
- sua vocação para a construção dinâmica de, potencialmente, qualquer tipo de mídia visual ou auditiva, dependendo apenas da tecnologia para suportá-la;
- a possibilidade oferecer alternativas a rígidas seqüências de apresentação através da hipertecnologia;
- a utilização do conceito de identificadores único para a definição de entidades, conforme recomendam a arquitetura Dexter [HAL 90] e a escola da orientação a objetos;
- a incorporação de estilos baseados em CSS em dialeto XML, removendo o problema de serem linguagens com estruturas de marcações diferentes.

[15] "As linguagens de 3ª geração foram projetadas para profissionais de processamento de dados e não para usuários finais. A depuração de programas escritos nessas linguagens consome tempo, e a modificação de sistemas complexos é relativamente difícil. As linguagens de 4ª geração foram projetadas em resposta a esses problemas. Os principais objetivos das linguagens de 4ª geração são: (1) facilitar a programação de computadores de maneira tal que usuários finais possam resolver seus problemas; (2) apressar o processo de desenvolvimento de aplicações; (3) facilitar e reduzir o custo de manutenção de aplicações; (4) minimizar problemas de depuração; e (5) gerar código sem erros a partir de requisitos de alto nível [PRI 2001, p. 3]."

Tais características tornam a linguagem Astrha/L detentora de características únicas, cujos potenciais são introduzidos, mas não exauridos, nesta pesquisa. A partir dos conceitos acima listados, foi projetado um conjunto de DTDs[16] que definem uma linguagem com marcações XML, conforme mostra a FIGURA 5.11.

O dialeto hipermídia *(hyper)* define uma linguagem de marcação simples, capaz de exibir hiperligações tanto na forma de hipertexto como na forma de mapas clicáveis.

Mealy define um dialeto para escrever uma determinada Máquina de Mealy, cujas instâncias irão se transformar em hiper-animações de um ambiente.

O dialeto de definição de ambiente *(environment)* possui informações de produção, tais como título, sinopse, data, *copyright*, autores, créditos, participações; sobre posicionamento, dimensões, hiper-animações (instâncias de Mealy) e estilos.

5.3 Identificador Único (Atributo Id)

Em Astrha/L, sempre que se quer referenciar unicamente um elemento, utiliza-se um atributo obrigatório de nome ***id***, cujo valor é um número inteiro que, dentro de uma lista de elementos de mesmo nome, deve ser único. Recomenda-se que seja um número natural, estritamente positivo. Tal preferência por esse conjunto numérico se dá visando um possível armazenamento em bancos de dados, que lidam com identificadores únicos, denominados chaves, nesse espaço intervalar.

5.4 Cores

Toda e qualquer cor definida em Astrha/L segue o padrão *Red, Green and Blue* (RGB), podendo ser uma cor predefinida ou, então, definida através de uma seqüência de 7 caracteres da seguinte forma:

- primeiro caractere: “#”.
- segundo e terceiro caracteres: dois algarismos hexadecimais, representando a intensidade da cor vermelha.
- quarto e quinto caracteres: dois algarismos hexadecimais, representando a intensidade da cor verde.
- sexto e sétimo caracteres: dois algarismos hexadecimais, representando a intensidade da cor azul.

[16] Uma Definição de Tipo de Documento (*Document Type Definition*, DTD) é um arquivo (ou conjunto de arquivos) escrito em Sintaxe Declarativa XML (XML*’s Declaration Syntax*) que contém uma descrição formal de um tipo de documento em particular [FLY 2003]. Para maiores detalhes, vide glossário.

As cores predefinidas em Astrha/L são as predefinidas pela W3C para CCS em XHTML, conforme mostra a FIGURA 5.1, mais três cores predefinidas pela classe Color da linguagem Java:

- darkGray = “#404040”
- orange = “#FFC800”
- pink = “#FFAFAF”

A classe Color de Java e as cores predefinidas em HTML tem algumas diferenças semânticas e de nomenclatura. Em Java, a cor predefinida *Color.green* possui código RGB = “#00FF00”, significando a cor verde maximizada. Já em HTML, esse mesmo código é denominado verde-lima (lime), sendo o verde codificado com “#008000”, tendo uma conotação menos voltada a questões de código, aproximando-se mais da nomenclatura das cores adotadas no dia-a-dia. Nesses conflitos semânticos, Astrha/L optou pela nomenclatura W3C, por ser mais próxima da realidade e, portanto, de nível maior de apreensibilidade *(learnability)* [ISO 91].

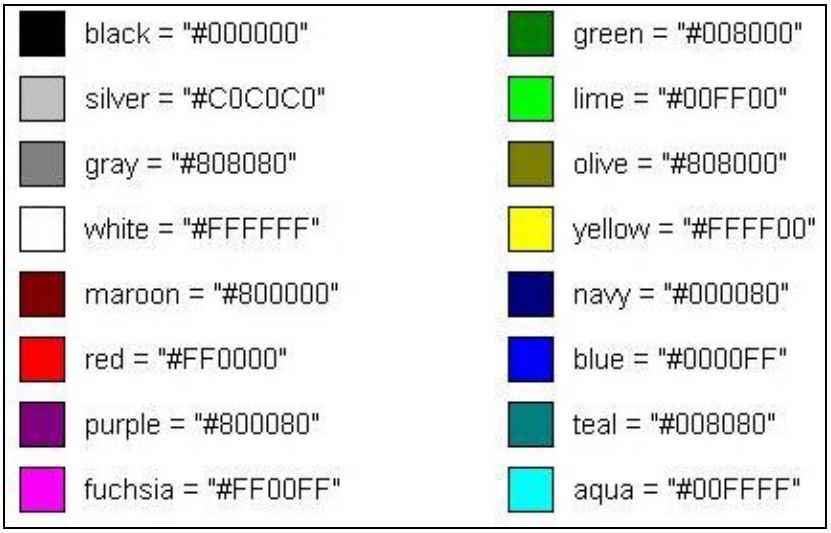


FIGURA 5.1 - Cores predefinidas W3C.

Resumidamente, a estratégia de Astrha/L para cores foi, portanto, adotar as cores predefinidas W3C, mais algumas cores predefinidas em Java, resultando nas 19 cores predefinidas da **Erro! Fonte de referência não encontrada.**. Permite, adicionalmente, exibir qualquer cor RGB onde cada cor básica (vermelho, verde e azul) tem sua tonalidade definida em um intervalo de 256 intensidades de luz. Tanto as cores predefinidas como as codificadas numericamente, podem ser escritas com caracteres maiúsculos ou minúsculos, invariavelmente.

TABELA 5.1 – Cores predefinidas Astrha/L

NOMENCLATURA PREDEFINIDA			
ASTRHA/L	W3C	JAVA	**Cor**
aqua	aqua	cyan	azul piscina
black	black	black	preto
blue	blue	blue	azul
darkGray		darkGray	cinza escuro
fuchsia	fuchsia	magenta	magenta[17]
gray	gray		cinza
green	green		verde
lime	lime	green	verde-limão
maroon	maroon		marrom
navy	navy		azul-marinho
olive	olive		verde-oliva
orange		orange	laranja
pink		pink	rosa
purple	purple		púrpura
red	red	red	vermelho
silver	silver	lightGray	prata
teal	teal		azul-esverdeado
white	white	white	branco
yellow	yellow	yellow	amarelo

[17] Sinônimo de carmim, matéria corante, de um vermelho muito vivo, ligeiramente arroxeado, extraída, originariamente, da cochonilha-do-carmim [FER 96]

5.5 Conjunto de Caracteres Válidos (Character Set)

Os caracteres válidos para se programar em Astrha/L são aqueles definidos pela W3C para a XML 1.0, segunda edição, uma vez que Astrha/L é um dialeto dessa metalinguagem. Conforme a recomendação atual [W3C 2002a], um caracter XML é uma unidade atômica de texto conforme especificado pela norma ISO/IEC 10646 [ISO 93]. Ainda conforme a recomendação citada, caracteres válidos são a tabulação *(tab)*, retorno do carro *(carriage return)*, avanço de linha *(line feed)*, os caracteres válidos do conjunto Unicode e da norma ISO/IEC 10646.

Entretanto, recomenda-se evitar a utilização de todo e qualquer caracter Unicode, restringindo-se portanto:

- aos caracteres de tabulação, retorno de carro, avanço de linha
- aos caracteres da língua inglesa
- aos caracteres da língua latina definidos na norma ISO 8859, Parte 1 [ISO 87], que define o conjunto de caracteres latinos para a metalinguagem SGML, da qual XML é derivada.

Essa restrição é recomendada para fins de compatibilidade com os analisadores *(parsers)* XML disponíveis atualmente.

5.6 Dialeto *Style* (Estilo)

Este subconjunto da linguagem Astrha/L é definido pela DTD *style.dtd* mostrada na FIGURA 5.2 define o estilo de apresentação dos textos e do fundo de tela *(background)* para a apresentação dos conteúdos de:

- um ambiente do dialeto *Environment;*
- uma instância de Máquina de Mealy, que é uma hiper-animação *(hyper-animation)* dentro de um ambiente;
- uma Máquina de Mealy do dialeto *Mealy* ou
- um documento hipermídia, do dialeto *Hyper*.

Este dialeto foi inspirado nas tecnologias *Cascading Style Sheet* (CSS) e HTML, com a vantagem de possuir a sintaxe XML. Dentro de uma árvore de mídias, um estilo definido em um nível mais alto reflete em seus demais nodos. Se uma folha, porém, redefinir o seu estilo, então este será o apresentado na folha [W3C 2002a]. Exemplificando, se definirmos um estilo em um ambiente, e não redefinirmos em seus nodos (hiper-animações), nem em uma Máquina de Mealy ou uma instância hipermídia, todo o ambiente refletirá o estilo de apresentação definido para o ambiente. Se, entretanto, uma hiper-animação redefinir o estilo, todas as mídias filhas refletirão esse novo estilo, e assim por diante.

5.6.1 Elemento *style* (estilo)

O elemento raiz deste dialeto é *styles*, que não possui atributos. Seus nodos filhos são uma lista elementos *style*, que possui os seguintes atributos:

- **Identificador único *(id)***: identificação único de um determinado estilo, dentro de uma lista.
- **Nome *(name)***: Atributo opcional. Serve para informar o nome do estilo que foi criado. Preferencial mas não obrigatoriamente deve ser único dentro de uma lista.

Além de seus atributos, um elemento style pode ter os seguintes elementos filhos, todos opcionais e inspirados na linguagem HTML e no formato CSS:

- ***text***.: definição da fonte do texto.
- ***link***. definição da fonte de uma hiperligação.
- ***aLink***: definição da fonte de uma hiperligação ao ser acionado o dispositivo apontador *(mouse)* dentro de sua área.
- ***vLink***: definição da fonte de uma hiperligação após ser visitada.
- ***oLink***: definição da fonte de uma hiperligação quando o dispositivo apontador *(mouse)* estiver sobre sua área.

Cada elemento filho do elemento style pode ter os seguintes atributos, todos opcionais:

- ***face***: define a família da fonte.
- ***size***: define o tamanho da fonte.
- ***color***: define a cor da fonte.
- ***format***: define o estilo da fonte. Se seu valor contiver a letra:
  - **b**: definirá a fonte em negrito *(bold)*
  - **i**: definirá a fonte em itálico *(italic)*
  - **u**: definirá a fonte sublinhada *(underline)*
- ***bgColor***: define a cor de fundo de tela.
- ***bgFile***: define a URL do arquivo de fundo de tela.
- ***bgShow***: define o alinhamento do arquivo de fundo de tela, quando informado. Pode conter os seguintes valores válidos, inspirados na possibilidades de exibição da família de sistemas operacionais *Windows*:
  - ***center***: a imagem é exibida ao centro. Este é o valor padrão a ser aplicado a *file*, na ausência da especificação deste atributo.
  - ***repeat***: a imagem repete-se ao fundo.
  - ***full***: a imagem sofre uma transformação de escala, ficando ampliada ou reduzida até a largura e altura do objeto que está sendo definido.

5.6.2 Entidades

As entidades predefinidas na segunda edição da especificação XML 1.0 podem ser utilizadas livremente e não necessitam sua definição em DTDs auxiliares [W3C 2002a]. As entidades predefinidas em SGML através de um código numérico também não necessitam ser definidas em uma DTD auxiliar, uma vez que XML é derivada da SGML. As demais entidades da FIGURA 5.2 compõem entidades da linguagem Astrha/L como opções para escrita de caracteres especiais.

Dessa maneira, Astrha/L define entidades para todos os símbolos predefinidos em XML e HTML - mais os símbolos de multiplicação (×), entidade `mult &#215;` e divisão (÷), entidade `div "&#247;` - todos em conformidade com a ISO 8879 da SGML [ISO 86].

DTD do dialeto *style*

```
<!ELEMENT styles (style*)>
<!ELEMENT style (text?, link?, aLink?, vLink?, oLink?)>
<!ATTLIST style
   id CDATA #REQUIRED
   name CDATA #IMPLIED>
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
   face CDATA #IMPLIED
   size CDATA #IMPLIED
   color CDATA #IMPLIED
   format CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   bgFile CDATA #IMPLIED
   bgShow CDATA #IMPLIED>
<!ELEMENT link (#PCDATA)>
<!ATTLIST link
   face CDATA #IMPLIED
   size CDATA #IMPLIED
   color CDATA #IMPLIED
   format CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   bgURL CDATA #IMPLIED
   bgShow CDATA #IMPLIED>
<!ELEMENT aLink (#PCDATA)>
<!ATTLIST aLink
   face CDATA #IMPLIED
   size CDATA #IMPLIED
   color CDATA #IMPLIED
   format CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   bgURL CDATA #IMPLIED
   bgShow CDATA #IMPLIED>
<!ELEMENT vLink (#PCDATA)>
<!ATTLIST vLink
   face CDATA #IMPLIED
   size CDATA #IMPLIED
   color CDATA #IMPLIED
   format CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   bgURL CDATA #IMPLIED
   bgShow CDATA #IMPLIED>
<!ELEMENT oLink (#PCDATA)>
<!ATTLIST oLink
   face CDATA #IMPLIED
   size CDATA #IMPLIED
   color CDATA #IMPLIED
   format CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   bgURL CDATA #IMPLIED
   bgShow CDATA #IMPLIED>
<!ENTITY nbsp    " ">
<!ENTITY iexcl   "&#161;">
<!ENTITY cent    "&#162;">
<!ENTITY pound   "&#163;">
<!ENTITY curren  "&#164;">
<!ENTITY yen     "&#165;">
<!ENTITY brvbar  "&#166;">
<!ENTITY sect    "&#167;">
<!ENTITY uml     "&#168;">
<!ENTITY copy    "&#169;">
```

DTD do dialeto *style*

```
<!ENTITY ordf    "&#170;">
<!ENTITY laquo   "&#171;">
<!ENTITY not     "&#172;">
<!ENTITY reg     "&#174;">
<!ENTITY macron  "&#175;">
<!ENTITY deg     "&#176;">
<!ENTITY plusmn  "&#177;">
<!ENTITY sup2    "&#178;">
<!ENTITY sup3    "&#179;">
<!ENTITY acute   "&#180;">
<!ENTITY micro   "&#181;">
<!ENTITY para    "&#182;">
<!ENTITY middot  "&#183;">
<!ENTITY cedil   "&#184;">
<!ENTITY sup1    "&#185;">
<!ENTITY ordm    "&#186;">
<!ENTITY raquo   "&#187;">
<!ENTITY frac14  "&#188;">
<!ENTITY frac12  "&#189;">
<!ENTITY frac34  "&#190;">
<!ENTITY iquest  "&#191;">
<!ENTITY Agrave  "&#192;">
<!ENTITY Aacute  "&#193;">
<!ENTITY Acirc   "&#194;">
<!ENTITY Atilde  "&#195;">
<!ENTITY Aring   "&#196;">
<!ENTITY Ccedil  "&#199;">
<!ENTITY Egrave  "&#200;">
<!ENTITY Eacute  "&#201;">
<!ENTITY Ecirc   "&#202;">
<!ENTITY Euml    "&#203;">
<!ENTITY Igrave  "&#204;">
<!ENTITY Iacute  "&#205;">
<!ENTITY Icirc   "&#206;">
<!ENTITY Iuml    "&#207;">
<!ENTITY ETH     "&#208;">
<!ENTITY Ntilde  "&#209;">
<!ENTITY Ograve  "&#210;">
<!ENTITY Oacute  "&#211;">
<!ENTITY Ocirc   "&#212;">
<!ENTITY Otilde  "&#213;">
<!ENTITY Ouml    "&#214;">
<!ENTITY mult    "&#215;">
<!ENTITY Oslash  "&#216;">
<!ENTITY Ugrave  "&#217;">
<!ENTITY Uacute  "&#218;">
<!ENTITY Ucirc   "&#219;">
<!ENTITY Uuml    "&#220;">
<!ENTITY Yacute  "&#221;">
<!ENTITY THORN   "&#222;">
<!ENTITY szlig   "&#223;">
<!ENTITY agrave  "&#224;">
<!ENTITY aacute  "&#225;">
<!ENTITY acirc   "&#226;">
<!ENTITY atilde  "&#227;">
<!ENTITY auml    "&#228;">
<!ENTITY aring   "&#229;">
<!ENTITY aelig   "&#230;">
<!ENTITY ccedil  "&#231;">
<!ENTITY egrave  "&#232;">
```

DTD do dialeto *style*

```
<!ENTITY eacute  "&#233;">
<!ENTITY ecirc   "&#234;">
<!ENTITY euml    "&#235;">
<!ENTITY igrave  "&#236;">
<!ENTITY iacute  "&#237;">
<!ENTITY icirc   "&#238;">
<!ENTITY iuml    "&#239;">
<!ENTITY eth     "&#240;">
<!ENTITY ntilde  "&#241;">
<!ENTITY ograve  "&#242;">
<!ENTITY oacute  "&#243;">
<!ENTITY ocirc   "&#244;">
<!ENTITY otilde  "&#245;">
<!ENTITY ouml    "&#246;">
<!ENTITY div     "&#247;">
<!ENTITY oslash  "&#248;">
<!ENTITY ugrave  "&#249;">
<!ENTITY uacute  "&#250;">
<!ENTITY ucirc   "&#251;">
<!ENTITY uuml    "&#252;">
<!ENTITY yacute  "&#253;">
<!ENTITY thorn   "&#254;">
<!ENTITY yuml    "&#255;">
```

FIGURA 5.2 - DTD do dialeto *style*

5.7 Dialeto *Mealy* (Mealy)

Este subconjunto da linguagem Astrha/L é definido pela DTD *mealy.dtd* mostrada na FIGURA 5.7. É a que mais se aproxima do modelo estrutural teórico (Astrha/M), definido na FIGURA 4.1. No contexto de Astrha/L, uma Máquina de Mealy contém:

Os atributos:

- **Nome *(name)***: Nome atribuído a esta Máquina de Mealy, preferencialmente algum que reflita seu funcionamento básico.
- **Identificador do estilo *(styleId)***: Identificador único do estilo de apresentação padrão da máquina. Se houver um estilo com esse identificador em sua lista interna de estilos, este será o estilo configurado, senão deverá ser um estilo definido no documento do dialeto environment que o incorpora. Tal funcionamento reflete estilos em cascata.

Os elementos:

- **Lista de estilos *(styles)***: Estilos de texto e de fundo de tela a serem utilizados nesta máquina em suas palavras de saída, definidos conforme *style.dtd*.
- **Alfabeto de entrada *(inputAlphabet)***: Este elemento define o alfabeto de entrada da máquina de Mealy, podendo possuir zero, um ou vários símbolos de entrada. Um símbolo de entrada (elemento ***input***) possui um atributo identificador ***(id)*** que deve ser único no conjunto de símbolos que compõem o o alfabeto de entrada. O elemento *input* possui também um atributo texto ***(text)***, formado por uma seqüência de caracteres XML válidos. O texto entrada "" (*string* vazio) é reservado, significando transição reflexiva. Possui também, opcionalmente, um nome para documentação. Outro atributo opcional é o tipo de símbolo de entrada ***(type)***. Atualmente, existe funcionalidade associada ao tipo marca de fim de fita ***(endMark)***, que possui a semântica de último símbolo de uma fita. Ao se informar uma fita contendo esse símbolo, os demais símbolos que viriam após sua leitura são ignorados, uma vez que o tamanho da fita fica limitado à primeira ocorrência desse símbolo. Se um símbolo do tipo fim de fita for definido no alfabeto de entrada, ao se informar uma fita através do painel de controle para uma instância dessa máquina, o ambiente interpretador (Astrha/E) deverá automaticamente acrescentar uma marca de fim de fita no final da seqüência de símbolos informada.
- **Símbolos de saída *(outputSymbols)***: Um símbolo de saída é composto de nenhuma, uma ou várias fontes (elementos ***source***). Se não tiver nenhuma fonte, será então uma palavra vazia. Uma vez que as imagens são compostas em camadas, a seqüência de símbolos da palavra de saída é importante. Uma fonte também possui um atributo identificador ***(id),*** que deve ser único nessa lista de símbolos de saída. O atributo modelo ***(model)*** indica em qual nodo da subárvore de fontes devem ser encontradas as mídias de entrada. O atributo de tipo ***(type)*** identifica, por sua vez, o tipo de mídia referenciado. As mídias também são agrupadas na árvore de diretórios por tipo, de modo que a

informação referente ao tipo indica, também, em qual nodo da subárvore de tipos de mídia deve encontrar sua fonte. O atributo arquivo ***(file)*** completa o caminho de busca pela mídia desejada, informado o nome do arquivo a ser referenciado. Caso sejam mídias visuais ou audiovisuais, as fontes podem ser posicionadas e dimensionadas em uma palavra de saída através dos atributos ***x, y, width*** e ***height***. Se não for informado ***x***, que denota a posição inicial da mídia no eixo das coordenadas, então a posição inicial padrão é zero. O mesmo ocorre com o eixo das abcissas, posicionado por ***y***. Os atributos de largura ***(width)*** e de altura ***(height)***, se definidos, podem solicitar que o objeto sofra uma transformação de escala. Os posicionamentos aqui propostos devem seguir um padrão computacional de coordenadas, com o eixo das coordenadas crescendo da esquerda para a direita, e o eixo das abcissas de cima para baixo, em um dispositivo de saída.

- **Palavras de saída *(outputWords)***: Conforme já mencionado no modelo teórico, as palavras de saída em Astrha/M são predefinidas a fim de formar um conjunto finito, evitando que palavras sem um significado desejado sejam formadas. Assim como os demais elementos da máquina, uma palavra de saída também deve ser unicamente identificada através do atributo identificador ***(id)***. A seqüência de símbolos de saída é determinada no atributo identificador das fontes ***(sourceIds)***, que deve listar os identificadores únicos dos símbolos de saída desejados, separados por vírgulas. Uma vez que a semântica de um ambiente Astrha/E é determinada por suas palavras de saída, o atributo significado ***(meaning)*** é fornecido para fins de documentação, permitindo que, se desejado, seja informado o significado de uma determinada palavra de saída. Da mesma forma que os símbolos de saída, existem os atributos de posicionamento e redimensionamento ***x***, ***y***, ***width*** e ***height***, com os mesmos significados. O posicionamento, neste elemento, é relativo à uma instância da máquina em *config.xml*.
- **Conjunto de estados *(statesSet)***: Cada estado listado nesse conjunto deve ter um identificador único e pode, opcionalmente, possuir um atributo nome ***(name)*** para identificação. Por se tratar de uma Máquina de Mealy, não há semântica associada aos estados, somente às transições listadas a seguir.
- **Função de transição *(transitionFunction)***: Por fim, a função de transição, contém nenhuma, uma ou várias transições definidas pelo elemento transição ***(transition)***. Como todos os outros elementos, uma transição é identificada unicamente pelo atributo identificador ***(id)***. Os atributos ***from***, ***input***, ***to*** e ***output*** fazem referência, respectivamente, a um identificador único do conjunto de estados, do alfabeto de estados, conjunto de estados e palavras de saída. Um atributo nome ***(name)*** também é fornecido, opcionalmente, para fins de documentação e pode ser útil, também, para inserir observações sobre a semântica da transição que será realizada.

Note que alguns atributos de uma máquina estão omitidos aqui, sendo esses informados somente nas instâncias, a fim de proporcionar maior flexibilidade. O estado inicial, por exemplo, é deixado para definir no momento de instanciar a máquina. Dessa maneira, a máquina, potencialmente, pode iniciar em qualquer estado. Outros atributos, como o de posicionamento da máquina, também são colocados nas instâncias pelo mesmo motivo, flexibilização.

5.7.1 Caractere Especial Curinga (∗)

Podemos especificar o símbolo especial curinga, denotado por um asterisco (∗), para ser utilizado nas transições para substituir o identificador único de:

- um estado qualquer de Q, tanto no domínio como no codomínio.
- um símbolo qualquer de entrada de Σ_τ.
- uma palavra qualquer de saída de λ.

Dessa maneira, podemos otimizar as transições, se desejarmos, obtendo as semânticas que seguem. Seja μ uma Máquina de Mealy da FIGURA 4.1 onde:

a) q1, q2 $\in$ Q, identificados unicamente pelos números 1 e 2, respectivamente, na linguagem Astrha/L.
b) $\sigma 1 \in \Sigma_\tau$, identificado unicamente pelos número 1 na linguagem Astrha/L.
c) $\omega 1 \in \Omega$, identificado unicamente pelos número 1 na linguagem Astrha/L.

5.7.1.1 Semântica de Estrutura de Desvio

Se utilizarmos o curinga em lugar do estado do domínio, como na linha de código

```
<transition id=”1" from="*" input="1" to="2" output="1"/>
```

Será gerado um conjunto T de transações no qual

$$\forall q \in Q \; \exists \; \lambda(q, \sigma 1) = \{(q2, \omega 1)\}$$

Tal situação denotará que, independente do estado atual, se entrar o símbolo σ, irá para o conjunto das partes formado por q2 com saída ω1. Em Astrha/L, tal agregação de transições denota uma estrutura monolítica de desvio [DIV 2002, p. 9-15], uma vez que, independente do estado em que estivermos, se dermos um determinado estímulo, iremos para um estado desejado sempre com a mesma palavra de saída. Tal recurso pode ser especialmente útil na construção de hiperligações.

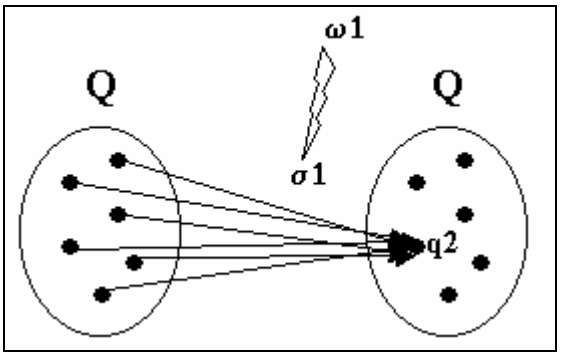


FIGURA 5.3 - Curinga * denotando uma estrutura de desvio

5.7.1.2 Semântica Seqüencial

Se utilizarmos o curinga em lugar do símbolo de entrada, como na linha de código

```
<transition id="1" from="1" input="*" to="2" output="1"/>
```

Será gerado um conjunto T de transações no qual

$\forall \sigma \in \Sigma_\tau \; \exists \; \lambda(q1, \sigma) = \{(q2, \omega1)\}$

Tal situação denotará que, independente do símbolo de entrada, se estiver no estado q1, irá para q2 e gerará a saída ω1. Em Astrha/L, tal agregação de transições força uma seqüencialidade do estado q1 para o estado q2, gerando a palavra de saída ω1.

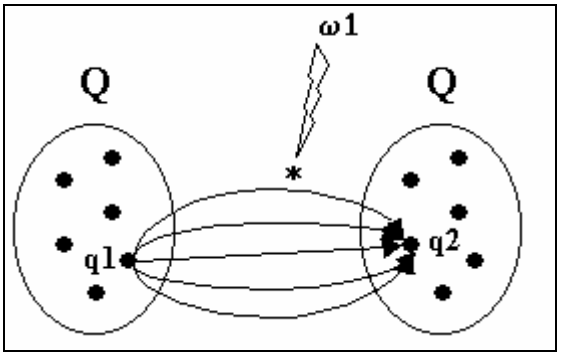


FIGURA 5.4 - Curinga * denotando seqüencialidade

5.7.1.3 Não-Determinismo Total de Estado

Se utilizarmos o curinga em lugar do estado do codomínio, como na linha de código

```
<transition id="1" from="1" input="1" to="*" output="1"/>
```

Será gerado um conjunto T de transações no qual

$\forall q \in Q \exists \lambda(q1, \sigma1) = \{(q, \omega1)\}$

Se a máquina μ tiver mais de um estado em Q, tal situação levará o autômato a um total não-determinismo de estado.

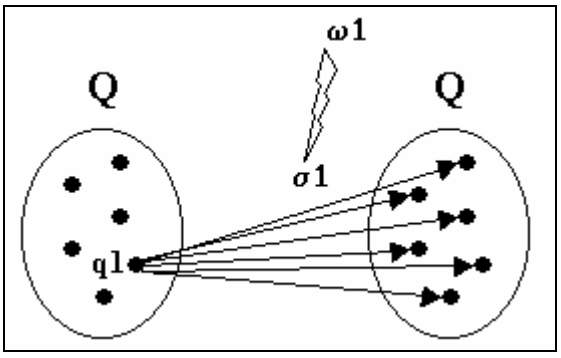


FIGURA 5.5 - Curinga * denotando não-determinismo total de estado

5.7.1.4 Não-Determinismo Total de Palavra de Saída

Se utilizarmos o curinga em lugar da palavra de saída, como na linha de código que segue:

```
<transition id="1" from="1" input="1" to="2" output="*"/>
```

Será gerado um conjunto T de transações no qual:

$\forall \omega \in \Omega \exists \lambda(q1, \sigma1) = \{(q2, \omega)\}$

Se a máquina μ tiver mais de uma palavra de saída em Ω, tal situação levará o autômato a um total não-determinismo de palavra de saída.

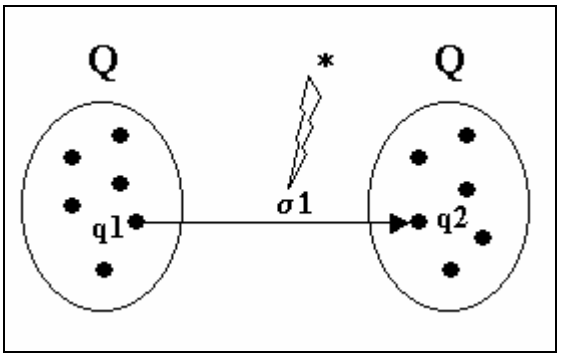


FIGURA 5.6 - Curinga * denotando não-determinismo total de palavra de saída

DTD do dialeto *mealy*

```
<!ELEMENT mealy (styles?, inputAlphabet, outputSymbols, outputWords,
statesSet, transitionFunction)>
<!ATTLIST mealy
   name CDATA #IMPLIED
   styleId CDATA #IMPLIED>
<!ELEMENT inputAlphabet (input*)>
<!ELEMENT input (#PCDATA)>
<!ATTLIST input
   id CDATA #REQUIRED
   text CDATA #REQUIRED
   type CDATA #IMPLIED
   name CDATA #IMPLIED>
<!ELEMENT outputSymbols (source*)>
<!ELEMENT source (#PCDATA)>
<!ATTLIST source
   id CDATA #REQUIRED
   model CDATA #REQUIRED
   type CDATA #REQUIRED
   file CDATA #REQUIRED
   x CDATA #IMPLIED
   y CDATA #IMPLIED
   width CDATA #IMPLIED
   height CDATA #IMPLIED>
<!ELEMENT outputWords (output*)>
<!ELEMENT output (#PCDATA)>
<!ATTLIST output
   id CDATA #REQUIRED
   sourceIds CDATA #REQUIRED
   meaning CDATA #IMPLIED
   x CDATA #IMPLIED
   y CDATA #IMPLIED
   width CDATA #IMPLIED
   height CDATA #IMPLIED>
<!ELEMENT statesSet (state*)>
<!ELEMENT state (#PCDATA)>
<!ATTLIST state
   id CDATA #REQUIRED
   name CDATA #IMPLIED>
<!ELEMENT transitionFunction (transition*)>
<!ELEMENT transition (#PCDATA)>
<!ATTLIST transition
   id CDATA #REQUIRED
   from CDATA #REQUIRED
   input CDATA #REQUIRED
   to CDATA #REQUIRED
   output CDATA #REQUIRED
   name CDATA #IMPLIED>
<!ELEMENT styles (style*)>
<!—-Importa os elementos da DTD style-->
```

FIGURA 5.7 - DTD do dialeto *mealy*

5.8 Dialeto *Hyper* (Hiper)

Este subconjunto da linguagem Astrha/L é definido pela DTD *hyper.dtd* mostrada na FIGURA 5.9. Esse dialeto define uma linguagem de programação hipermídia, contendo, portanto, marcações *(tags)* que definem hiperligações. Uma caractística importante dessa linguagem são suas ligações estendidas [W3C 2003a], que associam uma quantidade arbitrária de hiper-animações a sua âncora.

O elemento raiz do dialeto é ***hyper***, e contém dois atributos:

- **name**: atribui um nome para o documento hipermídia criado.
- **styleId**: identifica o estilo de apresentação padrão para o documento, que deve referenciar a um estilo listado internamente ou herdado em cascata.

Possui como elementos filhos a sua lista interna de estilos ***(styles)***, que é opcional, e uma lista de nenhum a vários parágrafos, declarados através do elemento ***p***, que também pode ter, opcionalmente, um atributo ***styleId*** e os seguintes elementos filhos:

- **referência a estilo *(s)***: Referência abreviada a um estilo através do atributo identificador ***(id)***, que pode ser qualquer um listado internamente ou herdado em cascata. Pode conter elementos texto ***(t)***, âncora ***(a)*** e imagem (***i***). Os elementos internos de uma referência a um estilo assumem o estilo referenciado.
- **texto *(t)***: Uma seqüência válida de caracteres e entidades XML.
- **âncora *(a)***: Uma âncora de hiperligação estendida. Referencia uma lista de hiper-animações, elementos ***(ha)***, através do elemento ***(hal)***. Um elemento ***ha*** possui os seguintes atributos:
  - **identificador *(id)***: atributo obrigatório que identifica esta instância de hiper-animação, unicamente, dentro desta lista de hiper-animações.
  - **nome *(name)***: nome da instância da hiper-animação, opcional.
  - **novo *(new)***: se o valor for ***true***, verdadeiro, define que a instância irá substituir a hiper-animação que a invocou. Na existência de mais de uma instância da lista com esse valor, a instância que substituirá será a primeira da lista. Cada hiper-animação com valor ***false***, falso, invocada irá, na seqüência estabelecida na lista, sobrepor a hiper-animação que a invocou. Este atributo é opcional e seu valor padrão é ***false***.
  - **estado inicial *(initialState)***: Atributo opcional que permite configurar o estado inicial desta instância de hiper-animação. O dialeto *environment* também permite configurar este atributo.
  - **executar *(execute)***: Atributo opcional que permite informar quantas vezes a hiper-animação invocada será executada. O valor padrão é "1", uma vez.

- **espera *(delay)***: Atributo opcional que permite informar a quantidade de ciclos de relógio esta hiper-animação irá esperar para iniciar. O valor padrão é "0", nenhuma vez. Outra maneira de realizar inoperabilidade aparente por instância é através de transições reflexivas informadas na sua fita de entrada.
- **cor de fundo *(bgColor)***: Atributo opcional que permite configurar a cor de fundo da instância.
- **posicionamento *(x, y)***: Atributos opcionais referentes ao posicionamento nos eixos das coordenadas *(x)* e das abcissas *(y)*. Seus valores padrão são *(0,0)*.
- **limites *(width, height)***: Atributos opcionais referentes à limites de largura *(width)* e de altura *(height)* de apresentação das mídias visuais de saída. Na ausência, o valor padrão é sem limites.
- **fita *(tape)***: Atributo opcional que permite informar uma fita de entrada, uma seqüência de elementos de entrada ***(input)***, para a instância da hiper-animação. O dialeto *environment* também permite informar fitas.

- **imagem *(i)***: Trata-se de uma imagem que, se possuir uma lista de âncoras ***(al)***, atributo opcional, torna-se um mapa clicável. Uma lista de âncoras é uma lista de elementos ***a***, conforme descrito no item acima. Como atributos, possui:
  - **arquivo *(file)***: URL do arquivo de imagem que será exibido para realizar o mapa clicável.
  - **limites *(width, height)***: Atributos opcionais referentes à limites de largura *(width)* e de altura *(height)* de apresentação das mídias visuais de saída. Na ausência, o valor padrão é sem limites.

5.8.1 Elemento Referência a Estilo (s)

O elemento de referência a estilo *s* possui como atributo único ***id***, que faz referência a um estilo declarado no documento. Textos, hiperligações e ambientes envoltos por essa referência assumirão o estilo desejado. Uma diferença importante em relação à linguagem HTML que predomina atualmente nos domínios da hipertecnologia, é que, em Astrha/L, um estilo não pode ser aninhado, evitando assim uma explosão não controlada de marcações de estilo. Esse enfoque de declarar previamente estilos antes de utilizá-los tem sido utilizado em outros formatos baseados em marcações como, por exemplo, o também bastante difundido *Rich Text Format (RTF)*, como mostra o trecho de documento RTF da FIGURA 5.8.

No quadro superior da figura, aparece o trecho do código e, abaixo, o texto de saída formatado.

```
...
{\f0\froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New
Roman;}
{\f1\fswiss\fcharset0\fprq2{\*\panose 020b0604020202020204}Arial;}
{\f2\fmodern\fcharset0\fprq1{\*\panose  02070309020205020404}Courier
New;}
...
{\f1\cf2 Fonte arial azul, }
{\cf11 agora times verde, }
{\b\f2 agora courier new em preto, negrito.\par }}
```

Fonte arial azul, agora times verde, **agora courier new em preto, negrito.**

FIGURA 5.8 - Trecho de documento em formato RTF

5.8.2 Elemento Texto (t)

Um elemento texto pode, ou não, estar envolvido em um elemento de referência a estilo; não possui atributos nem elementos filhos, e pode conter uma seqüência qualquer de caracteres válidos.

5.8.3 Elemento Hiperligação (link)

Neste dialeto, uma ligação pode referenciar um ou mais documentos, e ao ser acionada invoca-os paralelamente, possuindo uma semântica de ligação estendida, conforme conceitos da W3C para a recomendação XLink [W3C 2003a].

Um elemento ***link*** possui uma seqüência qualquer de caracteres válidos que formarão sua âncora e os seguintes atributos que formarão suas ligações:

- **Identificadores de hiper-animações *(haIds)***: Referência a hiper-animações descritas em um ambiente, que serão disparadas em paralelo ao se acionar a ligação. Os identificadores devem ser separados entre si através de vírgulas.
- **Indicadores de Nova Janela *(newWindow)***: Para cada identificador de hiper-animação listado em *haIds*, deve-se informar se a hiper-animação referida deve abrir em uma nova janela ou substituir a janela da hiper-animação atual, especificando-se os valores *true* e *false*, respectivamente. A lista de valores *true* ou *false* deve ser separada por vírgulas e pode conter somente um valor *true*.
- **Desvio *(jump)***: Se a hiper-animação que invocou esta ligação continuar ativa, pode-se desviar seu funcionamento para o seu estado inicial atribuindo o valor "*this.home*" ou então sua forma abreviada *"home"*. Se a semântica desejada for o retorno ao estado inicial do ambiente, especifica-se o valor *"super.home"*, causando o retorno do ambiente ao seu estado inicial, ignorando toda e qualquer outra ligação eventualmente solicitada.
- **Identificador de Estado *(stateId)***: Uma outra forma de desvio mais específica, o identificador de estado remete à hiper-animação que está gerenciando o hiperdocumento ao estado identificado.

DTD do dialeto *hyper*

```
<!ELEMENT hyper (styles?, p*)>
<!ATTLIST hyper
   name CDATA #IMPLIED
   styleId CDATA #IMPLIED>
<!ELEMENT p (#PCDATA | t | a | i | s)*>
<!ATTLIST p
   styleId CDATA #IMPLIED>
<!ELEMENT s (#PCDATA | t | a | i)*>
<!ATTLIST s
   id CDATA #REQUIRED>
<!ELEMENT t (#PCDATA)>
<!ELEMENT a (hal)>
<!ATTLIST a
   text CDATA #IMPLIED
   alt CDATA #IMPLIED
   shape CDATA #IMPLIED
   coords CDATA #IMPLIED>
<!ELEMENT hal (ha+)>
<!ELEMENT ha (tape?)>
<!ATTLIST ha
   id CDATA #REQUIRED
   name CDATA #IMPLIED
   new CDATA #IMPLIED
   initialState CDATA #IMPLIED
   execute CDATA #IMPLIED
   delay CDATA #IMPLIED
   bgColor CDATA #IMPLIED
   x CDATA #IMPLIED
   y CDATA #IMPLIED
   width CDATA #IMPLIED
   height CDATA #IMPLIED>
<!ELEMENT tape (input*)>
<!ELEMENT input (#PCDATA)>
<!ATTLIST input
   text CDATA #REQUIRED>
<!ELEMENT i (#PCDATA | al)*>
<!ATTLIST i
   file CDATA #IMPLIED
   width CDATA #IMPLIED
   height CDATA #IMPLIED>
<!ELEMENT al (a*)>
<!ELEMENT styles (style*)>
<!—-Importa os elementos da DTD style-->
```

FIGURA 5.9 - DTD do dialeto *hyper*

5.9 Dialeto *Environment* (Ambiente)

O dialeto de ambientes gráficos interativos, *environment*, é definido pela DTD *environment.dtd* (FIGURA 5.10) e produz, a partir dos outros três dialetos de Astrha/L, especificações a serem lidas por interpretadores capazes de ler os arquivos XML de Astrha/L e transformá-las em uma realidade audiovisual. Seu elemento raiz, *environment*, possui:

Os seguintes atributos:

- **Versão *(version)***: Identifica a versão Astrha/L, atualmente com valor "1.0".
- **Taxa de atualização dos quadros *(frameRate)***: Atributo opcional que informa a taxa de atualização desejada para os quadros apresentados, em milisegundos. Se omitido, o valor padrão é 1000ms.
- **Identificador do estilo *(styleId)***: Identificador único do estilo de apresentação padrão do ambiente, que deve existir em sua lista interna de estilos.
- **Título, sinopse, data e direitos autorais *(title, synopsis, date e copyright)***: Título, sinopse, data e direitos autorais dos conteúdos apresentados, respectivamente, para fins de documentação.

Os seguintes elementos:

- **Fundo de tela *(background)***: define um arquivo para fundo de tela.
- **Lista de estilos *(styles)***: Estilos de texto e fundo de tela a serem utilizados neste ambiente, definidos conforme *style.dtd*.
- **Lista de autores, de créditos e de participações (*authors*, *credits* e *participants*)**: Estes elementos permitem ao produtor do ambiente que cite pessoas ou organizações que contribuíram com a formação dos conteúdos disponibilizados pelo ambiente.
- **Lista de máquinas de Mealy *(mealies)***: Lista de máquinas de Mealy que, potencialmente, serão utilizadas pelo ambiente.
- **Lista de hiper-animações *(hyper-animations)***: Cada hiper-animação apresentada no ambiente é uma instância de uma Máquina de Mealy.

5.9.1 Fundo de tela *(Background)*

Fundos de tela neste dialeto possuem os seguintes atributos:

- **arquivo *(file)*:** Atributo obrigatório que define a URL do arquivo de fundo de tela.
- **posicionamento *(x, y)***: Atributos opcionais referentes ao posicionamento nos eixos das coordenadas *(x)* e das abcissas *(y)*.

5.9.2 Lista de Autores (*Authors*)

Elemento opcional para documentação dos autores da hiper-animação. Cada autor é uma pessoa ***(person)***, que possui um nome ***(name)***, como atributo obrigatório e observações ***(obs)***, como atributo opcional.

5.9.3 Lista de Créditos (*Credits*)

Elemento opcional para documentação, onde se permite informar créditos de produção. Um crédito também é associado a uma pessoas ***(person)***. Opcionalmente, pode ser documentado o seu papel ***(role)*** exercido na produção.

5.9.4 Lista de Participações (*Participants*)

Elemento que complementa os itens opcionais de documentação. Cada participante é uma pessoa ***(person)*** e, opcionalmente, pode-se informar também sua contribuição ***(contribution)***.

5.9.5 Lista de Máquinas de Mealy (*Mealies*)

Um elemento filho desta lista, Mealy *(mealy)*, é uma Máquina de Mealy que, potencialmente, será utilizado por alguma hiper-animação e possui os atributos:

- **Identificador único *(id)*:** Atributo obrigatório, identifica unicamente a máquina dentro da lista.
- **Diretório *(directory)*:** Atributo obrigatório, informa o nome do diretório da Máquina de Mealy a ser referenciada, que é um subdiretório do diretório "*mealies*" do diretório do ambiente, conforme a árvore de diretórios mostrada na FIGURA 6.2.

5.9.6 Lista de Hiper-Animações (*Hyper-Animations*)

Um elemento filho desta lista, hiper-animação *(hyper-animation)*, é uma instância de uma Máquina de Mealy definida pelo dialeto *mealy.dtd* e possui os atributos:

- **Identificador único *(id)***: Atributo obrigatório, identifica unicamente a hiper-animação dentro da lista.
- **Nome *(name)***: Nome da hiper-animação, sendo um atributo opcional para documentação.
- **Iniciar *(begin)***: Atributo obrigatório, pode possuir dois valores: *true* ou *false*. Se o valor for verdadeiro *(true)*, esta hiper-animação é disponibilizada ao ambiente imediatamente ao carregá-lo. Se for falso *(false)*, somente será disponibilizada se invocada através de uma hiperligação ou de algum comando de controle de painel.
- **Identificador da Máquina de Mealy *(mealyId)***: Atributo obrigatório, referencia um identificador único de uma Máquina de Mealy que será instanciada, pertencente à lista de máquinas de Mealy *(mealies)*.
- **Identificador do estado inicial *(initialStateId)***: Atributo obrigatório, identifica o estado inicial desejado para a Máquina de Mealy. Cada instância pode identificar livremente o estado inicial a fim de flexibilizar sua aplicabilidade.
- **Executar *(execute)***: Atributo opcional, permite indicar a quantidade de vezes a executar esta unidade. Seu valor pode ser um número natural estritamente positivo ou *"loop"*. No segundo caso, a hiper-animação repetirá a leitura de sua fita até que um comando externo de parada intervenha ou até a destruição da hiper-animação. Seu valor padrão é "1", uma vez .
- **Atraso *(delay)***: Atributo opcional. Trata-se de um número natural, estritamente positivo, que indica quantos em quantos ciclos esta unidade será atualizada. Se for definido com valor maior, por exemplo 3, a hiper-animação irá ler a fita de entrada de três em três ciclos, sendo o tempo de um ciclo definido pelo ambiente, através do atributo *frameRate*. Seu valor padrão é "1", uma vez.
- **Posicionamento *(x* e *y)***: Atributos opcionais referentes ao posicionamento nos eixos das coordenadas *(x)* e das abcissas *(y)*. Seus valores padrão são *(0,0)*.
- **Dimensões *(width* e *height)***: Atributos opcionais que definem, respectivamente, a altura e a largura máxima de exibição desta hiper-animação. Se uma figura exibida for maior que sua dimensão, então sofrerá uma transformação de escala reduzindo o seu tamanho à dimensão desejada. No caso de textos, os caracteres que ultrapassarem a dimensão não serão exibidos.

Uma hiper-animação possui também, opcionalmente, uma fita ***(tape)***, que possui uma lista de elementos denominados entrada ***(input)***, que por sua vez possuem um atributo texto ***(text)***, cujos valores são símbolos de entrada no modelo teórico. Preferiu-se utilizar o termo texto ***(text)*** por ser um termo mais natural do que símbolo *(symbol)* por usuários menos familiarizados com a teoria dos autômatos.

DTD do dialeto *environment*

```
<!ELEMENT environment (background?, styles?, authors?, credits?,
participants?,
   mealies?, hyperanimations?)>
<!ATTLIST environment
   version CDATA #FIXED "1.0"
   frameRate CDATA #IMPLIED
   styleId CDATA #IMPLIED
   title CDATA #IMPLIED
   synopsis CDATA #IMPLIED
   date CDATA #IMPLIED
   copyright CDATA #IMPLIED>
<!ELEMENT background (#PCDATA)>
<!ATTLIST background
   file CDATA #REQUIRED
   x CDATA #IMPLIED
   y CDATA #IMPLIED>
<!ELEMENT authors (author*)>
<!ELEMENT author (person)>
<!ELEMENT credits (credit*)>
<!ELEMENT credit (person)>
<!ATTLIST credit
   role CDATA #IMPLIED>
<!ELEMENT person (#PCDATA)>
<!ATTLIST person
   name CDATA #REQUIRED
   obs CDATA #IMPLIED>
<!ELEMENT participants (participant*)>
<!ELEMENT participant (person*)>
<!ATTLIST participant
   contribution CDATA #IMPLIED>
<!ELEMENT mealies (mealy*)>
<!ELEMENT mealy (#PCDATA)>
<!ATTLIST mealy
   id CDATA #REQUIRED
   directory CDATA #REQUIRED>
<!ELEMENT hyperanimations (hyperanimation*)>
<!ELEMENT hyperanimation (tape?)>
<!ATTLIST hyperanimation
   id CDATA #REQUIRED
   name CDATA #IMPLIED
   begin CDATA #REQUIRED
   mealyId CDATA #REQUIRED
   initialStateId CDATA #REQUIRED
   execute CDATA #IMPLIED
   delay CDATA #IMPLIED
   x CDATA #IMPLIED
   y CDATA #IMPLIED
   width CDATA #IMPLIED
   height CDATA #IMPLIED>
<!ELEMENT tape (input*)>
<!ELEMENT input (#PCDATA)>
<!ATTLIST input
   text CDATA #REQUIRED>
<!ELEMENT styles (style*)>
<!—-Importa os elementos da DTD style-->
```

FIGURA 5.10 - DTD do dialeto *environment*

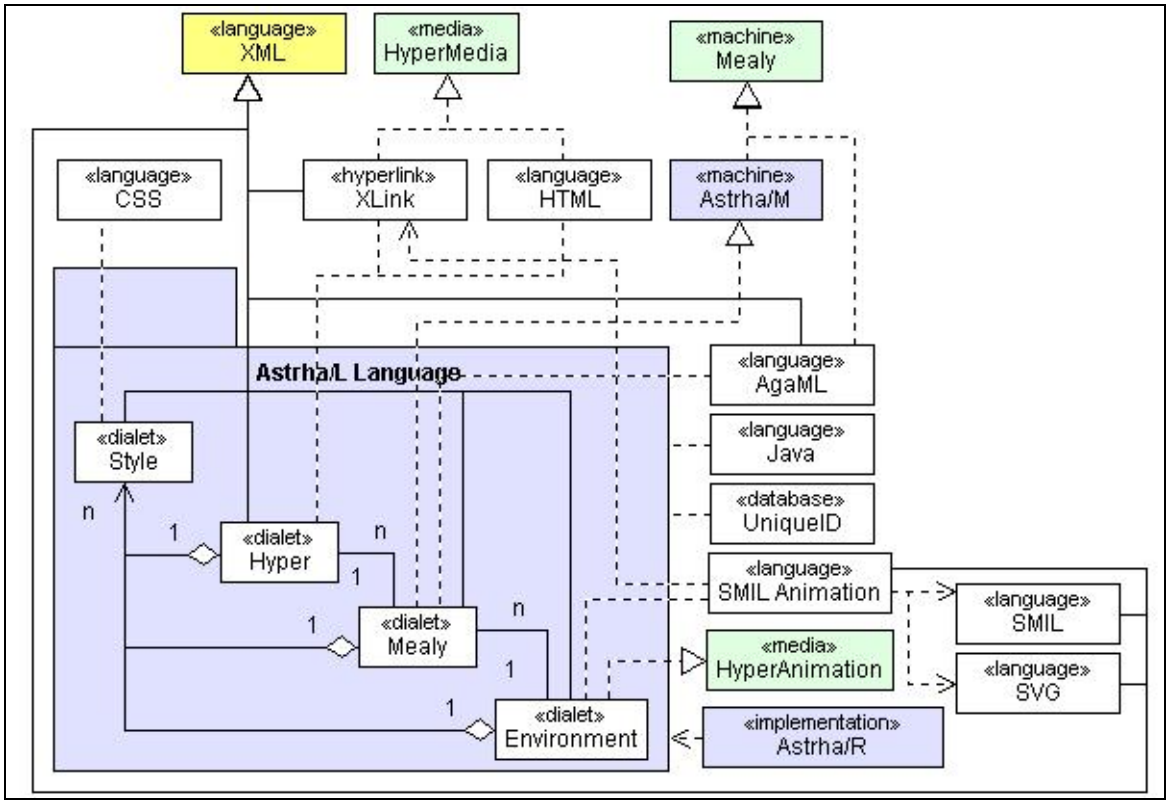


FIGURA 5.11 - Linguagem Astrha/L e suas influências

5.10 Conclusões

A linguagem Astrha/L, composta dos dialetos Mealy, Environment, Hyper e Style proporcionam ao desenvolvedor condições de especificar hiper-animações, configuração inicial do ambiente, hipertextos e estilos de textos em cascata, tudo em XML, evitando que tenha que definir documentos em linguagens como HTML, CSS, XML e JSP.

Apesar da riqueza semântica que o conjunto das linguagens citadas pode proporcionar, a utilização de somente uma linguagem (XML) com um conjunto minimizado de marcas com a finalidade específica de construir hiper-animações a torna uma linguagem de quarta geração, adequada a usuários finais.

5.10.1.1 Dialeto *Mealy* (Mealy)

A definição formal proporcionada pelo modelo teórico Astrha/M tornou-se uma especificação matematicamente segura que permitiu a concepção de um dialeto, denominado *Mealy*, para estruturar e conceber o comportamento desejado de hiper-animações. Abordagens formais evitam os problemas de compreensão que as especificações baseadas somente em linguagens naturais (textos) ou então em diagramas que permitem interpretação ambígua podem proporcionar.

5.10.1.2 Dialeto *Environment* (Ambiente)

O dialeto *Environment* possui elementos que permitem ao desenvolvedor configurar variáveis do ambiente, tais como estilos em cascata, fundo de tela e taxa de atualização dos quadros *(frame rate)*. Permite, também, que informe dados da produção: data, sinopse, direitos autorais, autores, créditos e participantes. Por fim, enumera as máquinas de mealy que serão referenciadas e as hiper-animações que serão instanciadas, automaticamente, ao se carregar o ambiente.

O objetivo desse dialeto é, portanto, permitir ao desenvolvedor que defina o comportamento desejado pelo ambiente e fornecer informações sobre a sua produção. A possibilidade do desenvolvedor definir o comportamento desejado pelo ambiente é uma característica importante que mídias multimídia com nível de interatividade diretivo deve incorporar [HEL 2001, p. 15-16]. As informações sobre a produção, opcionais, são oferecidas como um facilitador de documentação para o registro de dados básicos sobre as aplicações desenvolvidas (no caso, ambientes hiper-animados), característica desejável para linguagens de quarta geração.

5.10.1.3 Dialeto *Hyper* (Hiper)

Símbolos de saída no dialeto *Mealy* referenciam documentos escritos em *Hyper*, permitindo assim funcionalidade hipermídia para as hiper-animações. A opção de se estabelecer em XML um dialeto simples e direto para definir documentos hipermídia com ligações estendidas, é parte da estratégia de se implementar uma linguagem de quarta geração, procurando facilitar ao máximo o desenvolvimento ou personalização de um ambiente hiper-animado, seja por um profissional da informática ou por seu usuário final.

5.10.1.4 Dialeto *Style* (Estilo)

O dialeto *Style* é incorporado nas DTDs dos demais dialetos (*Mealy, Environment* e *Hyper*) proporcionando suporte à definição de estilos em cascata, à semelhança da linguagem CSS, aos caracteres especiais disponíveis em HTML e para os símbolos de multiplicação (×) e divisão (÷). A escolha das entidades HTML para definir caracteres especiais deveu-se à cultura amplamente absorvida pela comunidade de desenvolvedores, tornando-se um padrão de fato em hipermídia. As características do dialeto Style proporcionam aos dialetos que o incorporam propriedades (estilos em cascata e caracteres especiais) desejáveis e transparentes para o desenvolvedor.

6 Astrha/E (*Environment*, Ambiente)

6.1 Resumo

Definida a linguagem Astrha/L, é necessário construir um protótipo, já previsto no *Hyper Seed*, que a interprete. Com o protótipo serão realizados estudos de caso aplicados à educação a distância que validarão os conceitos, linguagem e tecnologia em relação aos objetivos. Este capítulo fala da construção do protótipo, denominado Astrha/E (*Environment*, Ambiente) apresentando escopo, análise de requisitos, paradigma, linguagens de especificação e programação, processo de desenvolvimento adotado, casos de uso e diagramas.

6.2 Interpretação de Astrha/L

A partir das especificações da linguagem Astrha/L, foram projetados diagramas de classes em UML, a partir dos quais foi construído um interpretador para Astrha/L em linguagem Java. Seu processo de desenvolvimento teve como referências iniciais de análise para a definiçã do escopo:

- a Máquina de Mealy especializada para as necessidades do Astrha/M;
- uso da linguagem de programação Astrha/L, definida no escopo desta pesquisa;
- conhecimento e a tecnologia já elaborados em pesquisas conexas realizadas no PPGC da UFRGS;
- sua vocação como ferramenta para construção de ambiente hipermídia, em especial para EAD via WWW, conforme o projeto Hyper Seed, do qual este trabalho faz parte.
- em decorrência de sua vocação, classificamo-lo com software Internet dentre as categorias de software identificadas pelo MCT18 [BRA 2001].
- a utilização das norma ISO/IEC 9126[19] [ISO 91] como diretriz básica para a concepção de um produto de software de elevada qualidade, através da observação das características desejadas de funcionalidade, usabilidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, citadas na norma, durante todo o processo de desenvolvimento do protótipo.
- Dessa maneira, dentre os vários processos, paradigmas e tecnologias para desenvolvimento de software existentes, foram escolhidos:

[18] Embarcado, uso próprio, Internet, pacote e encomenda.

[19] Essa norma possui uma versão brasileira, a NBR 13596 da ABNT.

- Para análise, projeto e construção do software foi escolhido o paradigma da orientação a objetos, devido às facilidades proporcionadas pelo conceito de tipo abstrato de dados e pelos mecanismos de herança e polimorfismo. Além disso, o grupo de pesquisa no PPGC já estava aplicando esses paradigma em seus projetos mais recentes.
- Conseqüentemente, a linguagem escolhida para especificação do software, traduzindo o modelo teórico em termos de orientação a objetos, foi a UML, do *Object Management Group* (OMG), por se tratar da linguagem de especificação padrão de fato para software orientado a objetos.
- Como processo de desenvolvimento[20], foi escolhido o método incremental e iterativo, dirigido por casos de uso, cuja aplicação é facilitada pelas características estruturais da orientação a objetos e pelos diagramas disponibilizados pela UML. Em uma visão geral, o processo foi, em nível macro, dividido em três passos, conforme apresentado em Larman:

> "**1. Planejar e Elaborar** – Planejamento, definição de requisitos, construção de protótipos e assim por diante.
>
> **2. Construir** – A construção do sistema.
>
> **3. Instalar** – A implantação do sistema para uso [LAR 2000, p. 40]."

Em vista de sua utilização em outras pesquisas relacionadas no PPGC da UFRGS; de sua essência estrutural ser orientada a objetos; de sua tecnologia já firmada em produtos Internet; de sua portabilidade; da disponibilidade de uma ferramenta de documentação em seu pacote básico de distribuição, facilitando a manutenibilidade, a linguagem Java, da Sun Microsystems, foi escolhida para sua programação. Mais especificamente, a tecnologia Applet, por se tratar de um ambiente gráfico dinâmico e interativo.

6.3 Planejamento e Elaboração

A fase de planejamento e elaboração pode ser composta de vários artefatos que auxiliam o desenvolvedor a compreender os requisitos do sistema, ou seja: onde se quer chegar. Durante esta fase, podem-se definir:

- **Um rascunho do plano**: trata-se de um texto que tenta explicar, inicialmente, os objetivos a serem atingidos com o software. Trata-se de um texto livre quanto ao formato, mas recomenda-se evitar, nesse instante, evitar definições tecnológicas, porque ainda é muito cedo para se ter uma visão abrangente sobre o problema em si e o caminho a tomar para resolvê-lo. Nesta pesquisa, tal rascunho foi elaborado através de diálogos entre orientador e orientando, com a colaboração de colegas de pós-graduação;

---

[20] Método para organizar as atividades relacionadas com a concepção, criação, entrega e manutenção de sistemas de software.

- **Um relatório de investigação preliminar**: realizado o esboço do que se pretende, é necessário formalizar as anotações em um formato de um relatório. Neste texto, a investigação preliminar de Astrha/E foi exposta no capítulo introdutório. Também compõe a investigação preliminar nesta pesquisa a declaração de sua estrutura básica de dados, formada pela Máquina de Mealy da FIGURA 4.1, pelo conceito de identificador único e árvore de diretórios. Tal estudo permitiu uma compatibilização coerente estre o modelo teórico, a linguagem Astrha/L e a implementação.
- **Requisitos**: Vamos definir os requisitos básicos de Astrha/E adiante, através de uma lista das funções do sistema;
- **Registrar termos do glossário**: atividade permanente, um glossário serve para que uma pessoa que venha a ler um documento de especificação possa entendê-lo com brevidade. A técnica utilizada neste texto para registrar o significado de termos específicos foi a de notas de rodapé ao surgir a primeira ocorrência do termo. Pode vir a ser interessante, em uma fase posterior, colecionar os termos mais relevantes e formalizá-los em um documento próprio;
- **Implementar protótipo**: Foi utilizado o software Microsoft PowerPoint para elaborar o primeiro protótipo da idéia central desta pesquisa, uma vez que o mesmo apresenta características multimídia, oferece hiperligações, permite gerar animações e possui um temporizador. O passo seguinte foi analisar a solução adotada durante a pesquisa do AGA, para observar . Do protótipo em PowerPoint, mais a análise realizada sobre o AGA, foi construído o protótipo de Astrha/E;
- **Definir casos de uso**: Descrições textuais ou gráficas dos processos do domínio, que levam em consideração as interações com o usuário, no processo de desenvolvimento adotado, este artefato oferece visões do cenário do software, e no processo de desenvolvimento escolhido, é um dos pontos chave para sua construção. Os casos de uso de Astrha/E são descritos adiante, textual e graficamente.

6.4 Estruturas de Dados em Astrha/E

Além de Astrha/M, outro conceito estrutural básico em Astrha/L é o identificador único *(id)* numérico, que possui essa característica a fim de facilitar sua manipulação por bancos de dados.

Outra característica básica é a incorporação do conceito de encapsulamento de objetos em sua árvore de diretórios. Dessa maneira, sua linguagem, projeto de sistema de arquivos e UML, construção do protótipo incorporam esses conceitos, facilitando assim projetos de bancos de dados que venham a querer interagir ou armazenar os dados de objetos manipulados pelo Astrha/E.

Em termos de sistemas de arquivos, um ambiente (uma determinada entrada para o Astrha/E), possui em seu próprio diretório o arquivo index.html, que irá invocar a chamada a Astrha/E, e vários arquivos XML, conforme mostra a FIGURA 6.1. O arquivo *authors.xml* contém dados sobre os autores da aplicação gerada; *credits.xml* e *participations.xml* possuem dados sobre créditos de produção, participação e agradecimentos especiais; *hyperanimations.xml* são instâncias das hiper-animações representadas pelas máquinas de Mealy disponíveis no subdiretório *mealies*. Por fim, *styles.xml* apresenta opções de estilos de folha *(style sheets)* a serem utilizadas pelos textos apresentados.

FIGURA 6.1 - Um diretório de ambiente Astrha/E em um servidor *Web*

Uma aplicação, por exemplo acordes, da FIGURA 6.1, pode definir uma ou mais máquinas de Mealy. Na FIGURA 6.2, podemos observar o ambiente `EAD` com duas máquinas de mealy (`Apresentacao` e `Simulador`). A pasta da Máquina de Mealy aberta mostra um arquivo *mealy.xml* que a define, e também o subdiretório *sources* utilizado para construir suas palavras de saída através de arquivos organizados conforme o tipo de mídia:

- ***graphics***: arquivos gráficos importáveis por uma determinada tecnologia. Por exemplo, arquivos formato GIF, JPEG, BMP, etc. Atualmente, o protótipo suporta arquivos nos formatos GIF e JPEG.
- ***hypermedia***: arquivos hipermídia definidos conforme a DTD *hyper.dtd*, que compõem o pacote de definições de linguagem do ambiente Astrha/E. Esses arquivos, à semelhança da linguagem HTML, podem referenciar quaisquer mídias de sua aplicação e, potencialmente, o que seria um bom trabalho futuro, de outros ambientes, inclusive em outros servidores.
- ***sounds***: arquivos de som importáveis por uma determinada tecnologia. Atualmente, o protótipo suporta arquivos nos formato AU, WAV, AIFF (extensões .aif e .aiff) e MIDI (extensões .mid e .rmi).

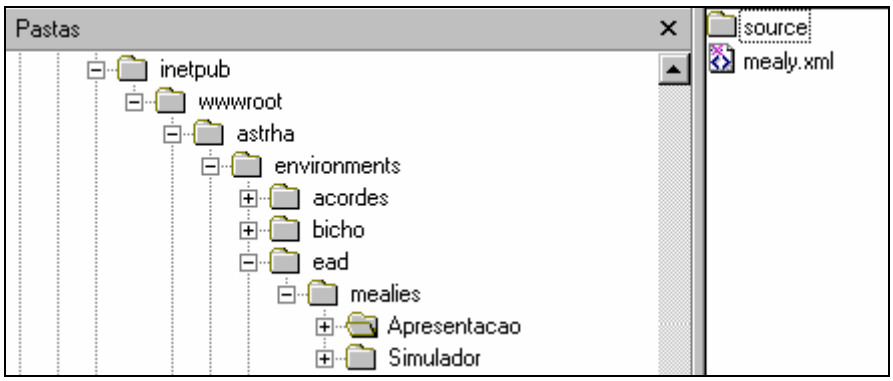


FIGURA 6.2 - Uma estrutura de um ambiente com duas máquinas de Mealy

6.5 Funções do Sistema

Um dos artefatos possíveis de se construir durante a fase de planejamento e elaboração, na visão de Larman, é uma tabela de funções a serem implementadas no ambiente Astrha/E conforme mostra a TABELA 6.1. Nessa tabela, temos a seguinte legenda para prioridades:

- P: primária
- S: secundária
- O: opcional

Outra característica dessa tabela é a observação da característica ISO/IEC 9126 [ISO 91] que mais se aproxima da função. Tal coluna é colocada como referencial para se observar metas de qualidade.

TABELA 6.1 - Funções do Sistema

Id	Função	Priori-dade	ISO 9126
F.1	Proporcionar semântica de hiper-animações que, compostas, formam um ambiente gráfico interativo. Tais hiper-animações devem comportar hipertecnologia tanto do tipo hipertexto, como do tipo mapa clicável.	P	Funcion.
F.2	As hiperligações devem ser capazes de referenciar a mais de um documento, caracterizando Astrha/E como hipertecnologia com multiligações *(multilink)*	P	Funcion.
F.3	O ambiente tem que ser multimídia, permitindo a exibição de mídias visuais e auditivas.	P	Funcion.
F.4	As mídias visuais possíveis de serem exibidas são: textos hipertextos, imagens, mapas clicáveis, animações, hiper-animações.	P	Funcion.
F.5	As hiper-animações devem permitir estruturas pseudo-aleatórias através de declarações não-deterministas em suas respectivas máquinas de Mealy	P	Funcion.
F.6	Uma hiper-animação pode demonstrar um comportamento de inoperabilidade através da símbolos de entrada lidos da tiva que denotem uma transição reflexiva em sua respectiva máquina de Mealy.	P	Funcion.
F.7	Os textos e hipertextos tem que poder ser formatados quanto: à cor de fundo de tela; ao tipo, tamanho e cor da fonte.	D	Usab.

Id	Função	Priori-dade	ISO 9126
F.8	As animações e as hiper-animações tem que ter a capacidade de sincronizar a emissão de sons.	D	Funcion.
F.8	Uma instância de Máquina de Mealy, que representa uma hiper-animação, tem que ter o poder se ser configurável em relação: ao estado inicial à fita de entrada ao momento que será executada, se ao carregar o ambiente, ou ao ser referenciado por uma hiperligação à quantidade de vezes que será executada ao seu posicionamento às suas dimensões	P	Funcion.
F.9	Um ambiente tem que ser configurável quanto: à sua taxa de atualização de quadros (frames) dimensões	P	Funcion.
F.10	Ser executável na maior quantidade possível de plataformas operacionais	P	Portab.
F.11	Ler arquivos especificados na linguagem Astrha/L	P	Funcion.
F.12	Prover, opcionalmente, um painel para o usuário controlar o ambiente, oferecendo os seguintes controles: iniciar, pausar, parar, atualizar fitas, mostrar símbolos de entrada lidos, recarregar a configuração e controlar individualmente cada hiper-animação	S	Usab.
F.13	O tempo de resposta deve permitir a geração de animações em tempo real	D	Eficiência
F.14	A estrutura de dados deve ser organizada de modo a facilitar persistência em banco de dados, evitar redundâncias e ambigüidades.	P	Funcion.
F.15	O fundo de tela da apresentação de conteúdos deve ser configurável	D	Funcion.
F.16	Deve-se poder inserir figuras em fundos de telas	D	Usab.
F.17	As figuras inseridas nos fundos de telas podem ser centralizadas, redimensionadas ou replicadas lada a lado, formando um mosaico.	D	Usab.
F.18	A interface deve oferecer meios do usuário saber informações sobre a produção, como: título, sinopse, data, autores, créditos, participações e direitos autorais.	D	Usab.
F.19	Deve-se poder configurar o estilo de apresentação de	D	Usab.

Id	Função	Priori-dade	ISO 9126
	conteúdos de um modo semelhante à tecnologia CSS.		
F.20	O usuário deve ter o poder de reposicionar as hiper-animações através de comandos de arrastar e soltar através de um dispositivo apontador (mouse).	D	Usab.
F.21	Permitir que o usuário consiga executar, igualmente, qualquer função do sistema através do teclado ou através do dispositivo apontador, melhorando a ergonomia do software.	D	Usab.

6.6 Casos de Uso

Seguindo as recomendações para processo de software de Larman, agora que definimos as principais funções a serem incorporadas ao ambiente, vamos descrever, textualmente, casos de uso expandidos para, depois, transformá-los em diagramas.

6.6.1 Caso de Uso C.1

Id	**C.1**
Atores	Usuário
Priorid.	Primário
Nome	Carregar ambiente sem painel de controle
Finalidade	Carga inicial de um ambiente sem mostrar o painel de controle.
Visão geral	O ator deseja utilizar o Astrha/E. Então, carrega uma página HTML, que irá invocar a exibição de um ou mais ambientes Astrha/E, dentro de um navegador *Web*. Como o painel de controle do ambiente Astrha/E referido não é mostrado, então o temporizador *(timer)* para atualização de quadros é inicializado exibindo as hiper-animações especificadas para iniciar imediatamente.
Ref.	F.10, F.11

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator carrega uma página HTML que contém a applet que executa Astrha/E.	2. A applet é carregada sem painel de controle
	3. O temporizador *(timer)* do ambiente é inicializado.
	4. As hiper-animações especificadas para iniciar imediatamente são exibidas.

Seqüências alternativas:

Item	**Resposta do Sistema**
2.	Applet não inicializada *(Applet not initialized)*. Pode ocorrer por vários motivos: A applet ou alguma classe Java referenciada pela applet não foi

Item	Resposta do Sistema
	encontrada não foi encontrada. Para corrigir, verificar a presença dessas classes e a variável do ambiente Java CLASS PATH. O servidor *Web* está fora do ar. Para corrigir, (re) inicializá-lo. O programa Astrha/L carregado apresenta problemas graves de sintaxe ou semântica. Para corrigir, verificar o programa e recarregar o sistema.
4	Se alguma hiper-animação não for exibida, é provável haver algum problema no programa Astrha/L carregado. Nesse caso, a solução é corrigir o programa.

6.6.2 Caso de Uso C.2

Id	**C.2**
Atores	Usuário
Priorid.	Primário
Nome	Carregar ambiente com painel de controle
Finalidade	Carga inicial de um ambiente mostrando o painel de controle.
Visão geral	O ator deseja utilizar o Astrha/E. Então, carrega uma página HTML que irá invocar a exibição de um ou mais ambientes Astrha/E, dentro de um navegador *Web*. Como o painel de controle do ambiente Astrha/E referido é mostrado, então é mostrada uma tela inicial do sistema, mostrando dados de produção, ficando no aguardo para iniciar a execução do ambiente através do botão START.
Ref.	F.10, F.11

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator carrega uma página HTML que contém a applet que executa Astrha/E.	2. A applet é carregada, oferecendo o ambiente desejado, com o painel de controle.

Seqüências alternativas:

Item	**Resposta do Sistema**
2.	Applet não inicializada *(Applet not initialized)*. Pode ocorrer por vários motivos: A applet ou alguma classe Java referenciada pela applet não foi encontrada não foi encontrada. Para corrigir, verificar a presença dessas classes e a variável do ambiente Java CLASS PATH. O servidor *Web* está fora do ar. Para corrigir, (re) inicializá-lo. O programa Astrha/L carregado apresenta problemas graves de sintaxe ou semântica. Para corrigir, verificar o programa e recarregar o sistema.
2	Dados de produção não exibidos. Tal situação pode ser gerada ou pela simples ausência dessas informações no programa Astrha/L carregado, ou por problemas de sintaxe ou semântica no programa Astrha/L carregado.

6.6.3 Caso de Uso C.3

Id	**C.3**
Atores	Usuário
Priorid.	Primário
Nome	Iniciar execução
Finalidade	Dar início a uma execução de um ambiente Astrha/E através do painel de controle.
Visão geral	Este caso de uso ocorre somente na presença do painel de controle. Nesse caso, o usuário pressiona o botão START.
Ref.	F.21, F.11

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator pressiona o botão *Start*	2. O temporizador *(timer)* do ambiente é inicializado.
	3. As hiper-animações especificadas para iniciar imediatamente são exibidas.

Seqüências alternativas:

Item	**Resposta do Sistema**
3	Se alguma hiper-animação não for exibida, é provável haver algum problema no programa Astrha/L carregado. Nesse caso, a solução é corrigir o programa.

6.6.4 Caso de Uso C.4

Id	**C.4**
Atores	Usuário
Priorid.	Secundário
Nome	Pausar execução
Finalidade	Interromper, momentaneamente, a execução do ambiente.
Visão geral	Este caso de uso ocorre quando o usuário deseja, por algum motivo, pausar a execução do ambiente.
Ref.	F.21

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator pressiona o botão *Pause*	2. O temporizador *(timer)* do ambiente é pausado.

6.6.5 Caso de Uso C.5

Id	**C.5**
Atores	Usuário
Priorid.	Secundário
Nome	Parar execução
Finalidade	Interromper, definitivamente, a execução do ambiente.
Visão geral	Este caso de uso ocorre quando o usuário deseja, por algum motivo, interromper de forma definitiva a execução do ambiente.
Ref.	F.21

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator pressiona o botão *Stop*	2. O temporizador *(timer)* do ambiente é pausado.
	3. Todas as hiper-animações do ambiente voltam ao seu estado inicial.

6.6.6 Caso de Uso C.6

Id	**C.6**
Atores	Usuário
Priorid.	Secundário
Nome	Atualizar todas fitas
Finalidade	Atualização de todas as fitas do ambiente.
Visão geral	O usuário trocou, por algum motivo, uma ou mais fitas do ambiente no programa Astrha/L e deseja que o ambiente leia novamente o programa, atualizando apenas as fitas e voltando todos os cabeçotes de leitura das fitas para seus respectivos estados iniciais.
Ref.	F.21, F.11

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator pressiona o botão *Update Tapes*	2. O temporizador *(timer)* do ambiente é pausado.
	3. O ambiente relê o programa Astrha/L de carga, todas as fitas das hiper-animações e retornando todos os cabeçotes de leitura das fitas para seus respectivos estados iniciais.

Seqüências alternativas:

Item	**Resposta do Sistema**
3	Se alguma hiper-animação não for exibida, é provável haver algum problema no programa Astrha/L carregado. Nesse caso, a solução é corrigir o programa.

6.6.7 Caso de Uso C.7

Id	**C.7**
Atores	Usuário
Priorid.	Secundário
Nome	Atualizar todo ambiente
Finalidade	Atualização de toda a configuração do ambiente.
Visão geral	O usuário atualizou, por algum motivo, algum item do programa Astrha/L e deseja que o ambiente leia novamente o programa, reinicializando-o totalmente.
Ref.	F.21, F.11

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. O ator pressiona o botão *Update Tapes*	2. O temporizador *(timer)* do ambiente é pausado.
	3. O ambiente relê o programa Astrha/L de carga, atualizando-o totalmente, passando a refletir o novo código carregado.

Seqüências alternativas:

Item	Resposta do Sistema
3	Se alguma hiper-animação não for exibida, é provável haver algum problema no programa Astrha/L carregado. Nesse caso, a solução é corrigir o programa.

6.6.8 Caso de Uso C.8

Id	C.8
Atores	Usuário
Priorid.	Secundário
Nome	Selecionar hiper-animação
Finalidade	Seleciona uma hiper-animação
Visão geral	Ao selecionar uma hiper-animação, o usuário passa a ter controle sobre algumas ações específicas a executar, como, por exemplo, exibir os símbolos que estão sendo lidos de sua fita, ou então solicitar a atualização de sua fita fornecendo os dados de entrada através de um campo no painel de controle.
Ref.	F.21

Seqüência típica de eventos:

Ação do Ator	Resposta do Sistema
1. O ator seleciona uma hiper-animação na caixa de listagem *Select Hyper-animation.*	2. Seleciona uma hiper-animação selecionada por seu nome e instância. É necessário acrescentar a instância à identificação porque a mesma hiper-animação pode ser instanciada mais de uma vez, se desejado.

Seqüências alternativas:

Item	Resposta do Sistema
2	Se não houverem hiper-animações a selecionar, então nenhuma foi carregada até o momento.

6.6.9 Caso de Uso C.9

Id	**C.9**
Atores	Usuário
Priorid.	Opcional
Nome	Informar nova fita
Finalidade	Informar uma fita nova para uma hiper-animação.
Visão geral	Se o usuário tiver selecionado previamente uma hiper-animação, ele poderá pausar o sistema e informar uma nova seqüência de símbolos de entrada (uma nova fita) para essa unidade. Ao fazer isso, o cabeçote de leitura da unidade selecionada volta para o início.
Ref.	F.21

Seqüência típica de eventos:

Ação do Ator	**Resposta do Sistema**
1. Estando o ambiente pausado e selecionada uma hiper-animação, o ator preenche em um campo símbolos de entrada separados por espaços em branco e pressiona o botão *Input*. Se um símbolo de entrada tiver caracter branco, então necessariamente terá que ser informado entre aspas, senão as aspas são opcionais.	2. O ambiente atualiza a fita desta hiper-animação.
	3. O cabeçote de leitura volta ao início.

Seqüências alternativas:

Item	**Resposta do Sistema**
2	Se algum símbolo de entrada não pertencer ao seu alfabeto, será ignorado.

6.6.10 Diagrama de Casos de Uso

A partir das descrições de casos de uso acima, concebeu-se o diagrama de casos de uso mostrado na FIGURA 6.3, que nos fornece uma visão geral do cenário de utilização do protótipo. Nessa visão, um usuário pode executar as seguintes ações:

- Carregar ambiente: tal ação pode se dar de duas formas: com ou sem painel de controle. Se for sem painel de controle, a ação inclui iniciar execução.
- Iniciar execução
- Parar execução
- Pausar execução
- Selecionar hiper-animação: para esta ação, é condição prévia pausar a execução
- Atualizar configuração: três tipos de atualização do ambiente são oferecidos:
  - informar uma nova fita de uma instância de hiper-animação;
  - atualizar todas as fitas de todas as instâncias de hiper-animações;
  - atualizar toda a configuração do ambiente.

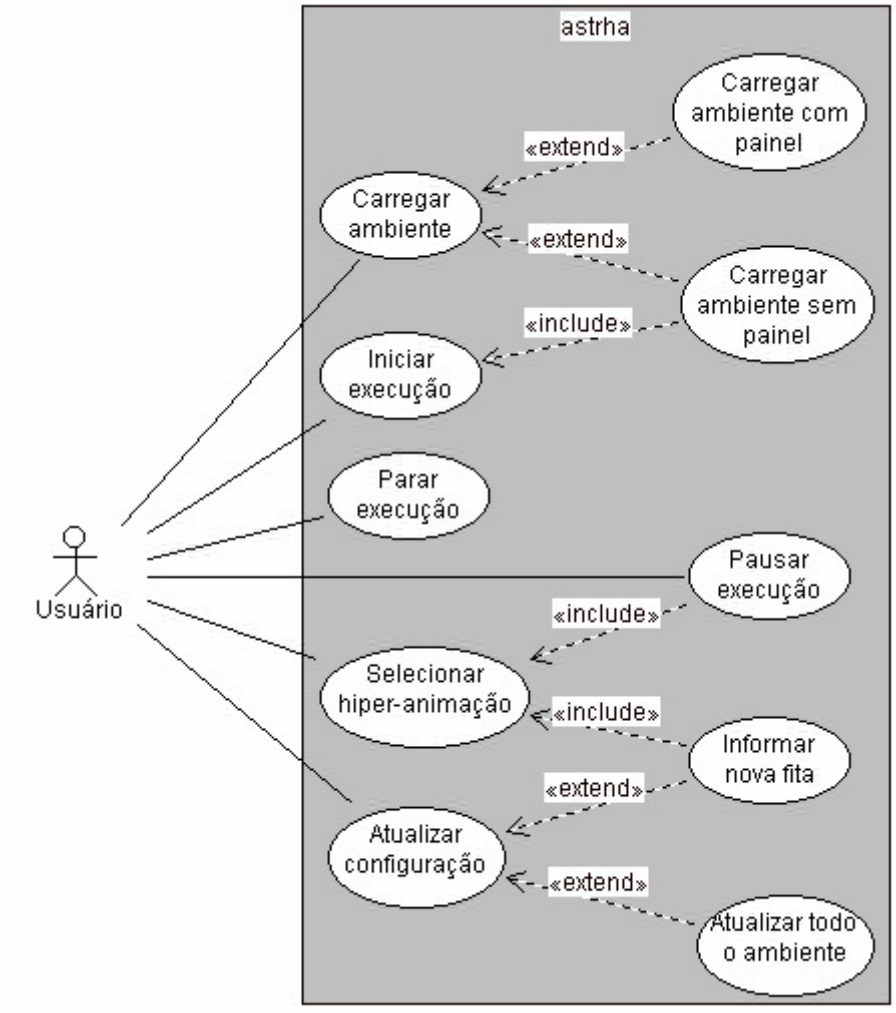


FIGURA 6.3 - Diagrama de Casos de Uso de Astrha/E

6.7 Construção do Astrha/E

Conforme já apresentado, Astrha/E segue o paradigma da orientação a objetos e utiliza a linguagem de programação Java para a construção de seu protótipo, e como tal, procura seguir as recomendações de estilo de programação sugeridas, dentre as quais podemos citar:

- O uso de padrões de projeto recomendados por autores academicamente conhecidos, entre eles Gamma, Larman e Ambler [GAM 2000, LAR 2000, AMB 2000];
- O controle do acesso dos membros de uma classe, definindo:
  - atributos como privados *(private)* e utilizando métodos do tipo obter() *(get())* e definir() *(set())* para seu acesso por outras instâncias;
  - redução da interface pública ao mínimo necessário a fim de reduzir a quantidade de interações necessárias para a definição de um plano de testes [DAV 96, p. 59].

- O uso de pacotes para agrupamento semântico das classes criadas, permitindo uma melhor compreensão do contexto da classe e, também, um compartilhamento melhor de componentes com outros projetos;
- O uso de classes abstratas para generalizar e especializar conceitos relacionados hierarquicamente.
- Documentação interna, utilizando comentários que reportam ao software de documentação da JRE, o javadoc, na declaração de classes, métodos e atributos, sempre que necessário. Como referencial de documentação, utilizaram-se as recomendações de Ambler em [AMB 2000a];

O uso das recomendações acima citadas auxiliaram no desenvolvimento do protótipo, aumentando seu potencial de reutilização, extensibilidade, manutenibilidade e reduzindo tempos de teste.

Continuando com as recomendações de Larman [LAR 2000], vamos passar agora da fase Planejar e Elaborar para o segundo passo, a fase Construir, que é dividida em vários ciclos de desenvolvimento, seguindo o modelo incremental iterativo, passando por repetidas micro fases cíclicas de análise, modelagem, construção e testes.

6.7.1 Primeiro Ciclo – Funções Básicas

No primeiro ciclo, podemos analisar os casos de uso C.1, C.2 e C.3, carregar ambiente sem painel de controle; carregar ambiente com o painel de controle e iniciar execução, dos quais podemos absorver os seguintes conceitos externos ao protótipo, formando seu ambiente de interação:

- página HTML *(HTML page)*
- navegador Web *(Web browser)*
- applet *(applet)*
- biblioteca de classes Java referenciadas *(referred Java library)*

Com o conceito genérico do Astrha/E acrescido de seus conceitos externos básicos, podemos visualizar um esboço genérico do ambiente no qual o protótipo opera, que é mostrado no diagrama de implantação UML da FIGURA 6.4.

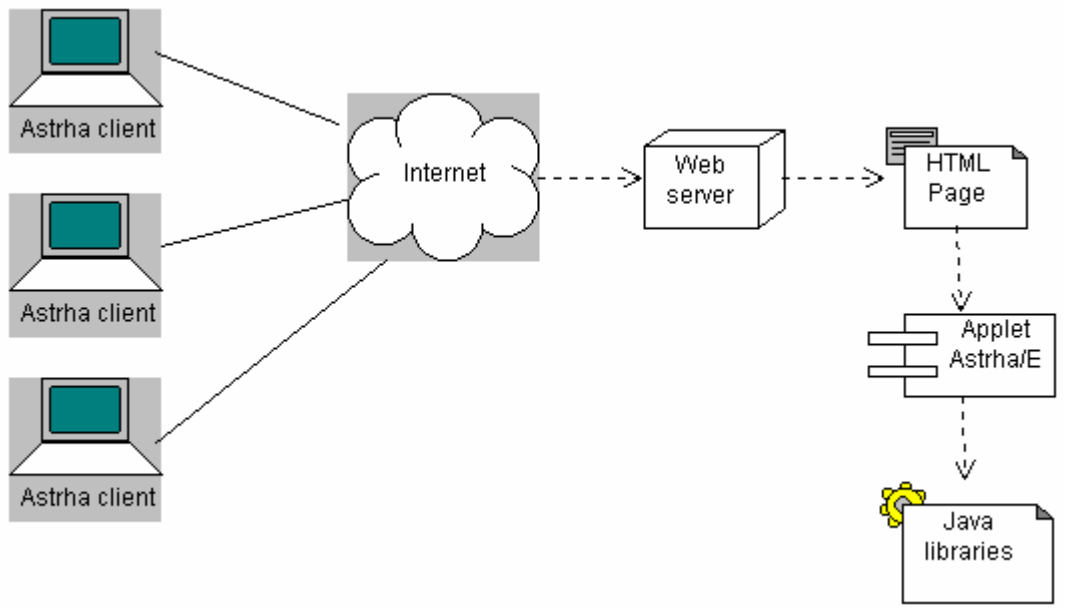


FIGURA 6.4 - Esboço genérico do ambiente operacional de Astrha/E

Na linguagem Astrha/L, foi definido que um ambiente pode conter uma lista de um ou mais estilos. Dessa maneira, devemos representar também essa agregação.

Outra característica importante do projeto da linguagem Astrha/L é o uso de identificadores únicos *(id)*, que em orientação a objetos podem ser gerenciados adequadamente estendendo-se todas as classes de objetos que utilizam esses identificadores a partir de uma classe de objetos persistentes *(persistent objects)*, conforme recomenda Ambler [AMB 2000].

Podemos perceber, também, a existência dos seguintes conceitos internos nesses casos de uso, a partir dos quais começamos a modelar conceitos de objetos, resultando no diagrama de classes UML conceitual da FIGURA 6.5:

- ambiente *(environment)*
- painel de controle *(control panel)*
- temporizador *(timer)*
- quadro *(frame)*
- hiper-animações *(hyper-animations)*, que são instâncias de máquinas de Mealy.
- controle de início de exibição *(run control)*
- carga de programa Astrha/L *(*Astrha/L *program loading)*. Por se tratar de dialeto XML, é conveniente utilizar um analisador *(parser)* XML para carregá-lo. Para essa tarefa, escolheu-se utilizar a biblioteca JDOM, da organização de mesmo nome, que contém o analisador *SAXBuilder*, por se tratar de código aberto, gratuito e amplamente distribuído [JDO 2002]. Além disso, já está sendo utilizada no projeto AGA, dos projetos que esta pesquisa busca unificar.
- botão início (*start*);
- tela inicial do sistema (*initial screen*);
- dados de produção, que conforme a linguagem Astrha/L, são formados de:
  a) um título *(title);*
  b) uma sinopse *(synopsis);*
  c) uma data *(date);*
  d) uma mensagem de direito autoral *(copyright);*
  e) uma lista de autores, que são pessoas *(people as authors);*
  f) uma lista de créditos *(credits)*, que da linguagem Astrha/L percebemos que têm os atributos pessoa *(person)* e papel exercido *(role)*;
  g) uma lista de participações *(participants)*, que da linguagem Astrha/L percebemos que têm os atributos pessoa *(person)* e contribuição *(contribution)*.

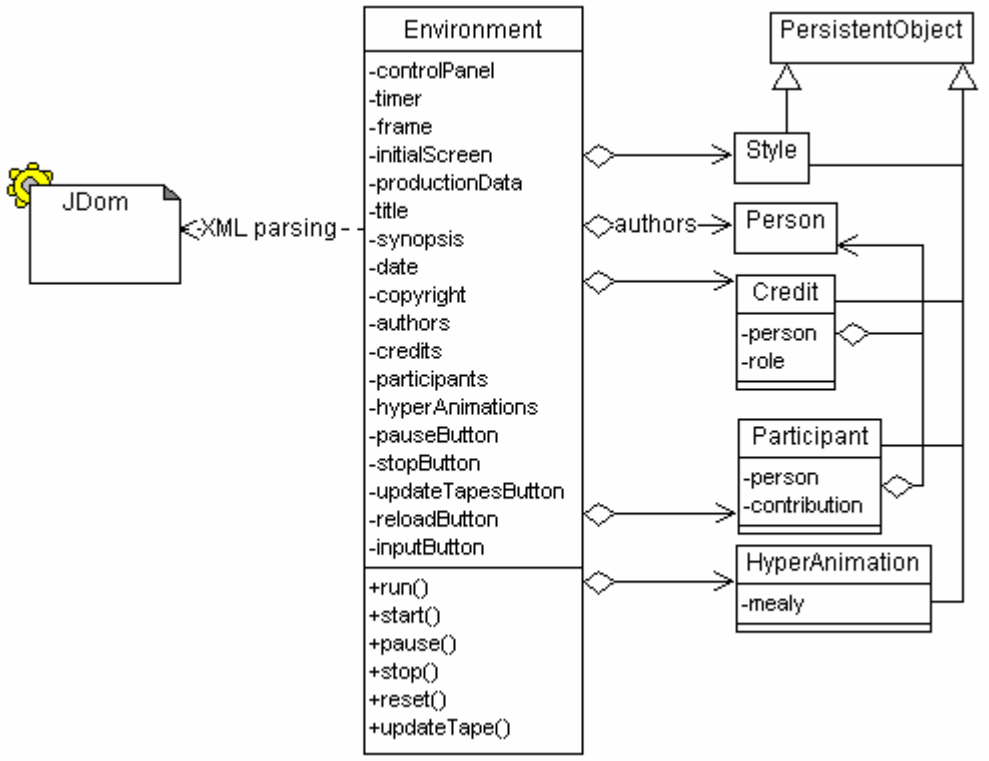


FIGURA 6.5 - Conceitos básicos de um ambiente Astrha/E

Podemos definir, ainda neste ciclo, classes para analisadores *(parsers)* XML no ambiente Astrha/E. Conforme a função do sistema F.11, temos que capacitar a leitura dos arquivos dos quatro dialetos Astrha/L: *environment, mealy, hyper e style*. Uma vez que as marcas *(tags)* do dialeto *style* são internalizadas nos documentos gerados pelos outros três, então foram definidos analisadores para os dialetos environment, mealy e hyper e uma classe Style, para o qual os mesmos fazem referência durante o processo de análise. A modelagem conceitual dos analisadores e da classe Style é mostrada na FIGURA 6.6.

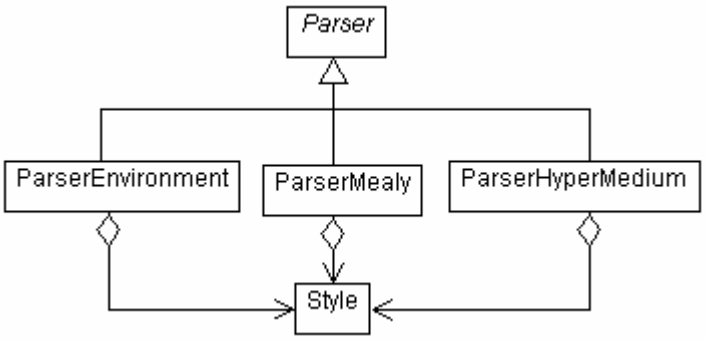


FIGURA 6.6 - Analisadores *(parsers)* dos dialetos de programa Astrha/L

É conveniente, também, neste ciclo definir questões tecnológicas. Ao se escolher pela linguagem Java, Optamos por utilizar applets estendidas da classe JApplet, do pacote *Swing*. Tal opção se deve à necessidade, em Astrha/E, de utilizar componentes do pacote *Swing* e, conforme recomendações do desenvolvedor da linguagem Java, a *Sun Microsystems*, "Toda e qualquer applet que contenha componente *Swing* deve ser implementada através de uma subclasse de JApplet [SUN 2002a]."

Um ambiente além de estender uma JApplet, incorpora a interface *ActionListener*, que permite a recepção de toda e qualquer ação do dispositivo apontador *(mouse)*. O protótipo também incorpora um objeto da classe *MouseHandler*, para tratamento de ações do dispositivo apontador, e um *SliderMouseHandler*, para tratamento de componentes deslizantes *(sliders)*, que permitem que um usuário selecione um valor deslizando um botão sobre uma linha. Incorporando-se os conceitos tecnológicos básicos de um ambiente Astrha/E, obtemos o diagrama conceitual da FIGURA 6.7.

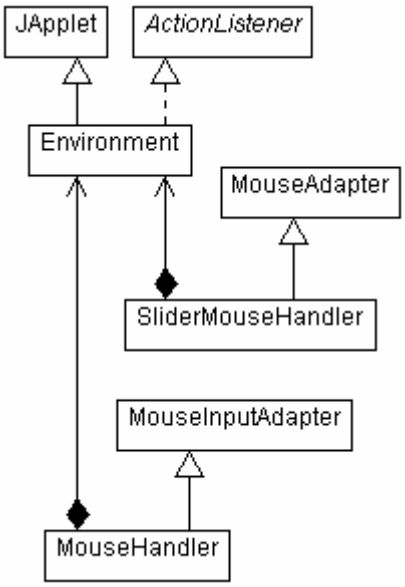


FIGURA 6.7 - Tecnologia básica utilizada em Astrha/E

6.7.2 Segundo Ciclo – Conceito de Máquina de Mealy

O surgimento do conceito de Máquina de Mealy no ciclo anterior, como atributo de uma hiper-animação, nos leva à necessidade de incorporar seus conceitos. A partir do seu modelo teórico proposto na FIGURA 4.1, podemos observar, inicialmente, a presença dos seguintes conceitos:

- Máquina de Mealy *(mealy machine);*
- símbolo de entrada *(input symbol);*
- alfabeto de entrada *(input alphabet)*, que agrega um conjunto de símbolos de entrada*;*

- estado *(state);*
- estado inicial *(initial state)*, que cada instância da máquina *(hyper-animation)* pode defini-lo individualmente;
- símbolo de saída *(output symbol)*, que agrega mídias *(media)* representadas por um conjunto de fontes dessas mídias *(sources)*;
- palavra de saída *(output word),* que agrega símbolos de saída;
- alfabeto de saída *(output alphabet)*
- transição *(transition)*, que agrega um estado atual *(source state)*, um símbolo de entrada, um próximo estado *(target state)* e uma palavra de saída;
- transição reflexiva *(reflexive transition)*, que gera a necessidade de seu tratamento, através de uma operação de aparente inoperabilidade *(nop())*;
- transição não-determinística *(non-deterministic transition)*, que gera a necessidade de seu tratamento, através de uma operação pseudo-aleatória *(choose())*.

Uma vez que o tratamento de uma transação reflexiva ou não-determinística depende da fita de entrada, e que cada instância de Máquina de Mealy possui sua própria fita, as operações *nop()* e *choose()* ficam encapsuladas nas instâncias, que são hiper-animações *(hyper-animations)*. Os novos conceitos absorvidos no segundo ciclo são mostrados na FIGURA 6.8.

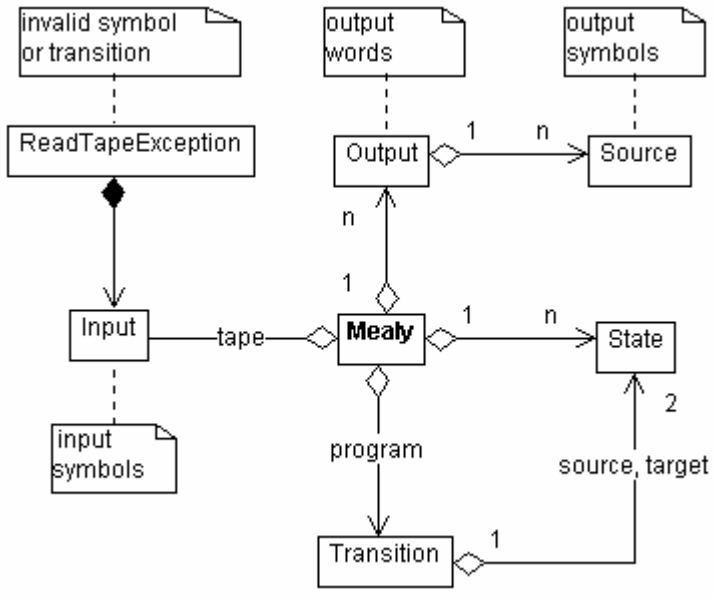


FIGURA 6.8 - Máquinas de Mealy Astrha/M traduzidas para UML

6.7.3 Terceiro Ciclo – Mídias

Como pudemos observar no ciclo anterior, os símbolos de saída são compostos de mídias de diversos tipos. Neste ciclo, vamos especificar que tipos de mídia queremos manipular. Neste caso, não existe um modelo teórico em Mealy e devemos buscar na computação gráfica esses elementos. Existem várias formas de se classificar os diferentes tipo de mídia existentes: quanto ao impacto causado; quanto aos aspectos técnicos de produção; quanto à suas capacidades cognitivas; quanto ao seu valor agregado; quanto à sua complexidade, e outros fatores [HEL 95].

Em nosso caso, como estamos tratando de modelagem em nível estrutural, devemos considerar aspectos técnicos. Inicialmente, podemos classificar mídias simples em dois macro níveis: mídias visuais e mídias sonoras. As mídias sonoras ficam especializadas em uma classe Som *(Sound).* As visuais ficam especializadas em uma classe denominada Representação Visual *(VisualRepresentation)* para evitar conflito de nome com a classe *Image*, predefinida na biblioteca gráfica *Abstract Window Toolkit* (AWT) da linguagem Java. Conforme taxionomias apresentadas por Heller [HEL 2001, HEL 95, p. 37], outros dois tipos de mídia especializadas são textos *(texts)* e animações *(motions)*. Em Astrha/E, os textos são expandidos para hipermídias, que são mídias textuais que agregam parágrafos *(paragraphs)* e hiperligações *(links)* que podem referenciar hiper-animações *(hyper-animations)*. As animações, por sua vez, são representadas pelas hiper-animações compostas das diversas mídias declaradas em suas correspondentes máquinas de Mealy, cuja seqüência de quadros é programada através da fita de entrada.

Uma observação importante, no contexto Astrha/E, é que a possível presença de hiperligações nas hiper-animações têm o poder de quebrar a seqüencialidade tradicional das animações, oferecendo possíveis seqüências alternativas por meio de interações com o usuário.

Quanto aos mapas clicáveis, uma estratégia recomendada é o seu tratamento e especialização como figura. Em [THA 91, p. 46-47], Turner apresenta uma modelagem orientada a objetos para aplicações gráficas bidimensionais baseado em noções de programação gráfica da linguagem Eiffel[21]. A abordagem proposta é conveniente no contexto Astrha/E, uma vez que apresenta figuras compostas, que em Astrha/E são palavras de saída. A proposta é conveniente para a implementação de mapas clicáveis, uma vez que oferece figuras especializadas nas formas de retângulo *(rectangle)*, círculo *(circle)* e polilinha[22] *(polyline)*, que correspondem, respectivamente, aos formatos *rect, circle* e *poly* definidos para mapas clicáveis em HTML, cujas formas básicas estão definidas igualmente para os mapas clicáveis em Astrha/E.

---

21 Linguagem orientada a objetos produzida em 1985 por Bertrand Meyer.

22 Seqüência aberta de pontos coplanares, distintos, não colineares e sem auto-interseção, definida por pelo menos dois vértices [CPD 2002]

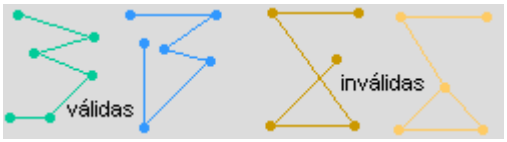


FIGURA 6.9 - Polilinhas válidas e inválidas.

Nesse contexto, um mundo *(world)* é a descrição de uma realidade bidimensional, representada através de um ambiente *(environment)* em Astrha/E. Figuras são componentes do mundo, que em Astrha/E são traduzidas por palavras de saída. Os dispositivos são porções de uma tela de um computador que apresenta seu sistemas de coordenadas, que em Astrha/E é traduzida conforme a semântica de coordenadas das applets. As janelas *(windows)*, servem para estabelecer uma correspondência entre os componentes de um mundo e sua representação gráfica no dispositivo, que em Astrha/E, é a área de apresentação definida nas dimensões de um ambiente.

Conforme mostra a FIGURA 6.10, nesse contexto uma figura simples é especializada em aberta (OPEN_FIG) ou fechada (CLOSED_FIG). Uma figura aberta pode ser especializada em uma polilinha e esta, por sua vez, em um segmento. Uma figura fechada é especializada em texto gráfico (GRAPHIC_TEXT), polígono (POLYGON) e elipse (ELLIPSE). Um polígono é especializado em retângulo (RECTANGLE), triângulo (TRIANGLE) e polígono regular (REG_POLYGON). Uma elipse é especializada em um círculo (CIRCLE). Uma figura complexa é composta das figuras simples descritas, conforme o padrão de projeto *Composite*[23] apresentado por Gamma [GAM 2000].

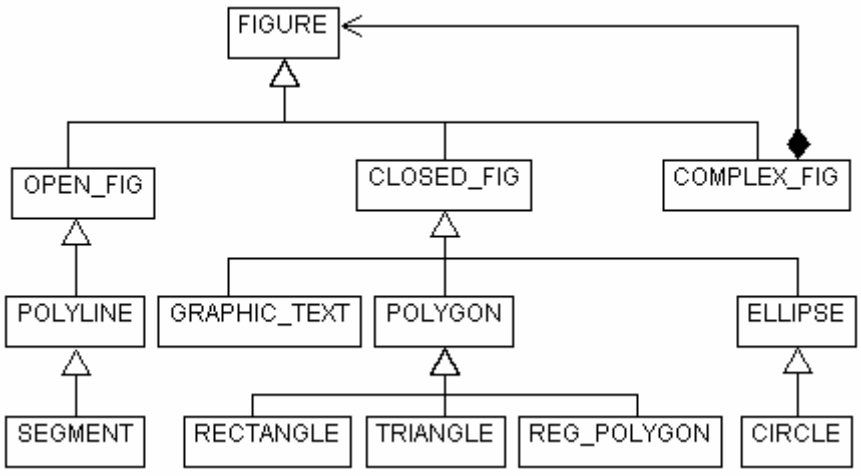


FIGURA 6.10 - Figuras bidimensionais em linguagem Eiffel

23 Padrão de projeto estrutural, orientado a objetos, que compõe objetos em estrutura de árvore parra representar hierarquias do tipo partes-todo.

Como o objetivo atual da modelagem de figuras bidimensionais em Astrha/E é implementar a semântica de mapas clicáveis, oriunda do HTML, a modelagem por Turner ficou reduzida ao mínimo necessário para representar polilinhas *(polylines)*, retângulos *(rectangles)* e círculos *(circles)*, respeitando uma hierarquia lógica que favoreça uma expansão da modelagem de figuras em versões superiores do protótipo.

A organização das mídias em Astrha/E, sumariando as análises realizadas neste ciclo, ficam estruturadas conforme mostra a FIGURA 6.11, na qual uma mídia pode ser um som ou uma representação visual. Uma representação visual pode ser uma figura, um documento hipermídia, ou uma hiper-animação, que é composta de mídias de qualquer tipo, inclusive sonoras. Um documento hipermídia agrega parágrafos e hiperligações, onde cada hiperligação que tem o poder de referenciar uma ou várias hiper-animações. Por fim, um mapa clicável é um tipo de hiperligação que pode referenciar várias figuras do tipo polilinha, retângulo ou círculo.

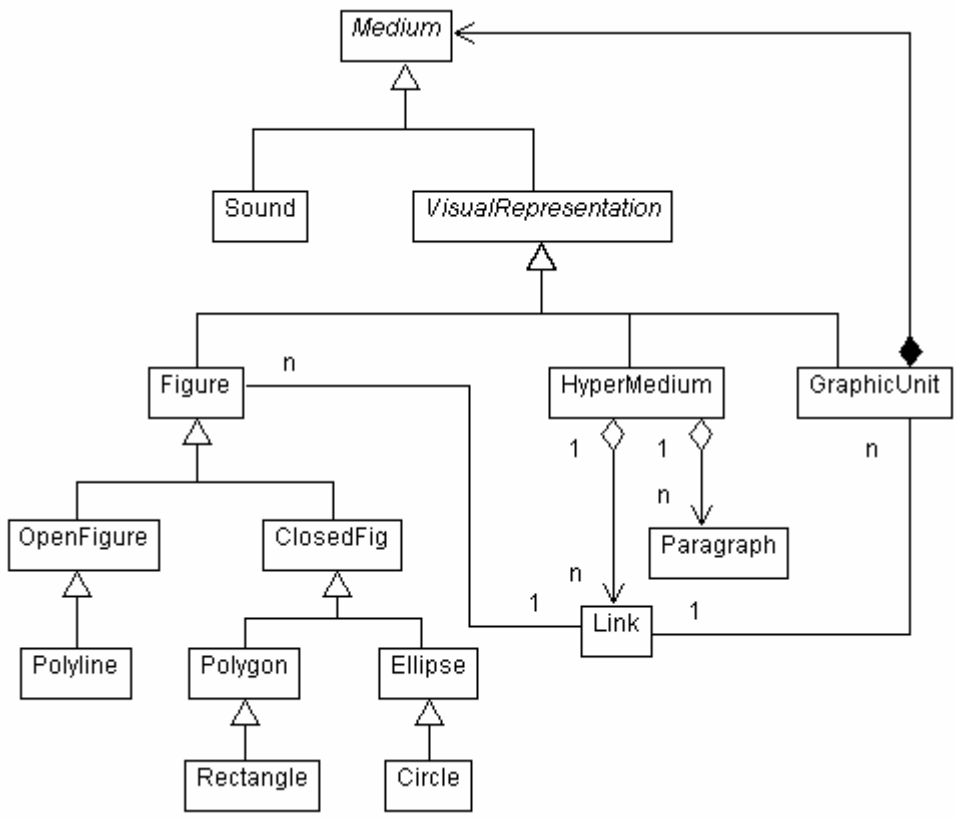


FIGURA 6.11 - Estrutura das mídias em Astrha/E

6.7.4 Quarto Ciclo – Pausar e Parar

Os casos de uso C.4 e C.5 oferecem ao usuário controles de painel. Basicamente, o que se acrescenta, conceitualmente, são atributos e operações no objeto que define ambiente, o *Environment*:

Novos atributos:

- botão Pausar *(pause button)*;
- botão Parar *(stop button)*.

Novas operações:

- Pausar (*Pause*())
- Parar (*Stop*())
- Voltar a configuração inicial *(Reset())*

Com a implementação do quarto ciclo, o objeto Environment ficou acrescido conforme mostra a FIGURA 6.12.

FIGURA 6.12 - Environment acrescido das funções de pausar e parar

6.7.5 Quinto Ciclo – Atualizações de Configuração

O quinto e último ciclo de desenvolvimento para a primeira versão do Astrha/E, são implementados os casos de uso C.6, C.7, C.8 e C.9, que fazem acrescentam na classe *Environment*:

Os atributos:

- botão Atualizar Fitas (*update tapes button*);
- botão Atualizar Ambiente (*reload button*);
- botão Nova Fita *(input button)*;

As operações:

- Selecionar hiper-animação (*selectHyperAnimation*());
- Atualizar fita, que possui como parâmetro, uma hiper-animação *(updateTape(HyperAnimation))*.

Com a implementação do quinto e último ciclo, o objeto Environment ficou acrescido conforme mostra a FIGURA 6.13.

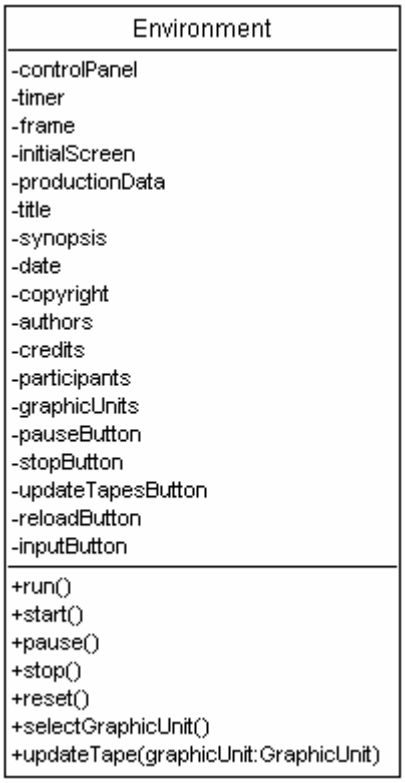


FIGURA 6.13 - Environment acrescido das funções de atualização e seleção

6.8 Visão de Astrha/E para Bancos de Dados Relacionais

O ambiente Astrha/E utiliza, atualmente, o sistemas de arquivos para armazenar suas configurações e mídias. Muitas vezes, entretanto, é desejável seu armazenamento em banco de dados, seja por questões funcionais, de segurança ou outros fatores. Dados armazenados por um SGBD podem ser utilizados, também, por um ambiente Astrha/E, como o nosso protótipo, ou por ferramentas especializadas para produzir, por exemplo, programas em linguagem Astrha/L automaticamente, com potencial de minimizar a escrita de programas com problemas sintáticos ou mesmo semânticos.

Apesar da protótipo do ambiente Astrha/E ter sido analisado e construído no paradigma da orientação a objetos, em vista do modelo relacional ser a tecnologia dominante atualmente para bancos de dados, apresentaremos uma possível estrutura de dados especificada em um diagrama entidades-relacionamentos (E-R), traduzidos a partir da modelagem UML anteriormente apresentada. A estratégia de mapeamento do paradigma da orientação a objetos para o relacional seguiu as orientações de Ambler em [AMB 2000b].

Como podemos observar na FIGURA 6.14, foram evitadas chaves compostas, conforme recomenda Ambler. Nessa modelagem, podemos perceber que um linha da entidade *Environment,* que denota um ambiente, pode se relacionar com várias linhas das entidades *Person, Participant, Credit e HyperAnimation,* formando sua lista de autores, participantes, créditos e hiper-animações, respectivamente. Uma Máquina de Mealy (entidade *Mealy*) pode servir como programa para várias hiper-animações e possui um conjunto de estados, transições, um alfabeto de símbolos de entrada e um alfabeto de palavras de saída representados através de relacionamentos múltiplos com as entidades *State, Transition, Input* e *Output,* respectivamente. Os símbolos de saída de uma Máquina de Mealy, podendo ser de diferentes tipos de mídia, são formados pelo conjunto de sons, imagens e hipermídias, entidades *Sound, Image e HyperMedium*, respectivamente.

Uma transição possui um estado atual *(sourceStateId)*, um símbolo de entrada *(inputId),* um próximo estado *(nextStateId)* e uma palavra de saída *(outputId)*.

Uma entidade hipermídia pode possuir várias hiperligações (entidade *Link*). Se a hiperligação for do tipo mapa clicável, pode conter círculos, retângulos e polilinhas.

Por fim, um mesmo estilo pode se relacionar várias vezes com ambientes, hiper-animações, máquinas de Mealy e documentos hipermídia.

A modelagem E-R apresentada na FIGURA 6.14 está em nível lógico e omite atributos, uma vez que seu objetivo não é apresentar um modelo definitivo, e sim apenas esboçar uma organização que venha a se traduzir, fisicamente, em um banco de dados relacional.

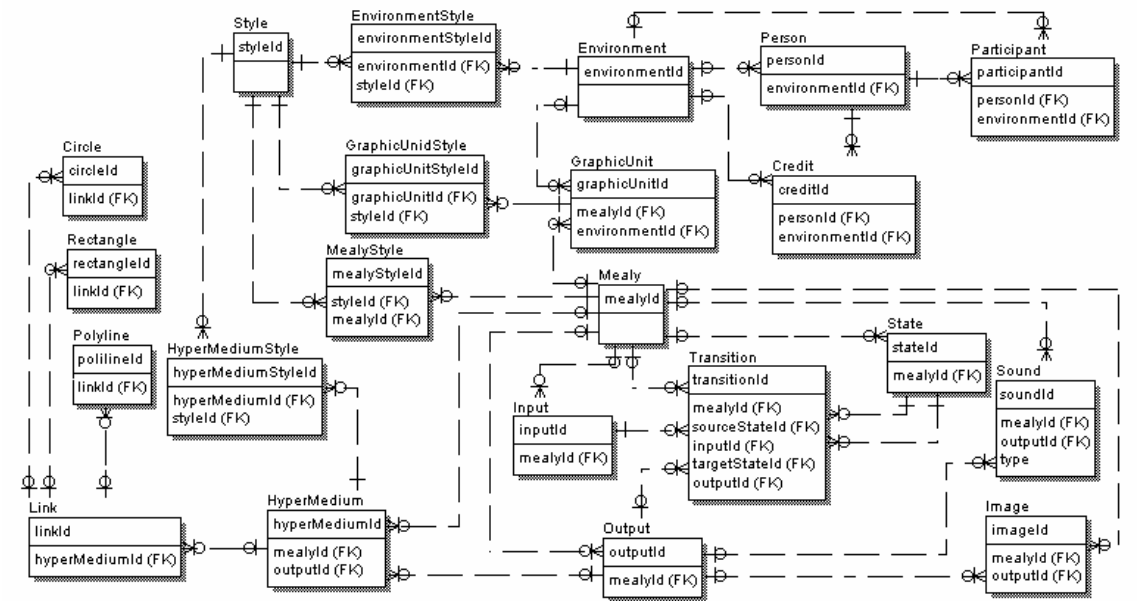


FIGURA 6.14 - Modelagem E-R para o Astrha/E

6.9 Instalação

O protótipo deve ser instalado em ambiente operacional conforme mostra a FIGURA 6.4. Em termos tecnológicos, Astrha/E pode ser instalado em qualquer servidor *Web* que possua instalada uma JRE 1.3.

Em nosso ambiente de testes, o mesmo foi instalado em um servidor Apache HTTP Server 2.0.43, da organização de mesmo nome, em sistema operacional Microsoft Windows 98, que operou normalmente. O mesmo será brevemente portado para sistema operacional *Solaris*, da *Sun Microsystems*, ou alguma distribuição Linux, para operar na página <http://teia.inf.ufrgs.br> do Instituto de Informática da UFRGS.

São três as bibliotecas adicionais necessárias para executar Astrha/E, todas com código aberto e gratuito:

1. **JDOM**: Da *JDOM Organization*, esta API fornece métodos de alto nível destinados ao acesso, manipulação e disponibilização de dados oriundos de arquivos XML [JDO 2002];

2. **JAXP**: Da *Sun Microsystems*, essa biblioteca habilita aplicações a realizar operações de análise *(parse)* e de transformação em documentos XML [SUN 2002b];

3. **Crimson**: Do projeto XML da *Apache Foundation*, Crimson é um analisador *(parser)* para documentos XML da versão 1.0, através de uma das seguintes APIs: JAXP, SAX 2.0, extensões de SAX2 e DOM nível 2 [APA 2002].

Na máquina cliente, caso o navegador que irá carregar Astrha/E não suporte Java versão 1.3 em sua configuração padrão, que é a realidade da maioria dos navegadores disponíveis atualmente, faz-se necessária a instalação de um *plug-in*[24] Java dessa versão ou superior. Se o mesmo não estiver previamente instalado no navegador da máquina cliente, sugere-se que a página HTML na qual um ambiente Astrha/E esteja inserido busque automaticamente o *plug-in*, bem como as bibliotecas adicionais necessárias (JDOM, Crimson e JAXP), baixando esses arquivos e instalando-os automaticamente. Para construir uma página HTML com essas características, pode-se utilizar o utilitário HTML Converter, da *Sun Microsystems*, disponível gratuitamente para baixa *(download)* em <http://java.sun.com/products/plugin/1.2/converter.html>.

```
<html>
<head><title>astrha&copy; - Sample 2</title></head>
<body bgcolor="white">
<center>
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = 710 HEIGHT = 376
codebase="http://java.sun.com/products/plugin/1.3/jinstall-13-
win32.cab#Version=1,3,0,0">
<PARAM NAME = CODE
VALUE="br.ufrgs.inf.astrha.environment.Environment.class" >
<PARAM NAME = CODEBASE VALUE="../../." >
<PARAM NAME = ARCHIVE VALUE="astrha.jar, crimson.jar, jdom.jar,
jaxp.jar" >
<PARAM NAME="type" VALUE="application/x-java-applet;version=1.3">
<PARAM NAME="scriptable" VALUE="false">
<PARAM NAME = "server" VALUE="http://localhost">
<PARAM NAME = "directory" VALUE="sample2">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.3"
CODE = "br.ufrgs.inf.astrha.environment.Environment.class"
CODEBASE = "../../." WIDTH = 600 HEIGHT = 400 scriptable="false"
pluginspage="http://java.sun.com/products/plugin/1.3/plugin-install.html">
<NOEMBED>
</COMMENT>
alt="Seu navegador conhece a marca &lt;APPLET&gt;, no entanto a applet
Java não está sendo executada, por alguma razão." Seu navegador não
está reconhecendo a marca &lt;APPLET&gt;.
</NOEMBED>
</EMBED>
</OBJECT>
</center>
</body>
</html>
```

FIGURA 6.15 - Documento convertido pelo HTML Converter para o Astrha/E

[24] Um arquivo que contém dados utilizados para alterar, aprimorar ou aumentar a funcionalidade de um aplicativo-pai. [HOW 2002]

A visão geral da implantação de um ambiente cliente-servidor para Astrha/E é mostrado na FIGURA 6.16. O Cliente, através de um navegador Internet *(Web browser)* que suporta diretamente Java 2 versão 1.3 ou dotado de um *plug-in* para essa finalidade acessa, via Internet, uma página HTML que possui uma ou mais applets Astrha/E. A applet carrega as informações necessárias para construir seus objetos gráficos a partir de arquivos XML, escritos na linguagem Astrha/L. JDOM, JAXP e Crimson auxiliam na leitura e tratamento dos dados XML lidos.

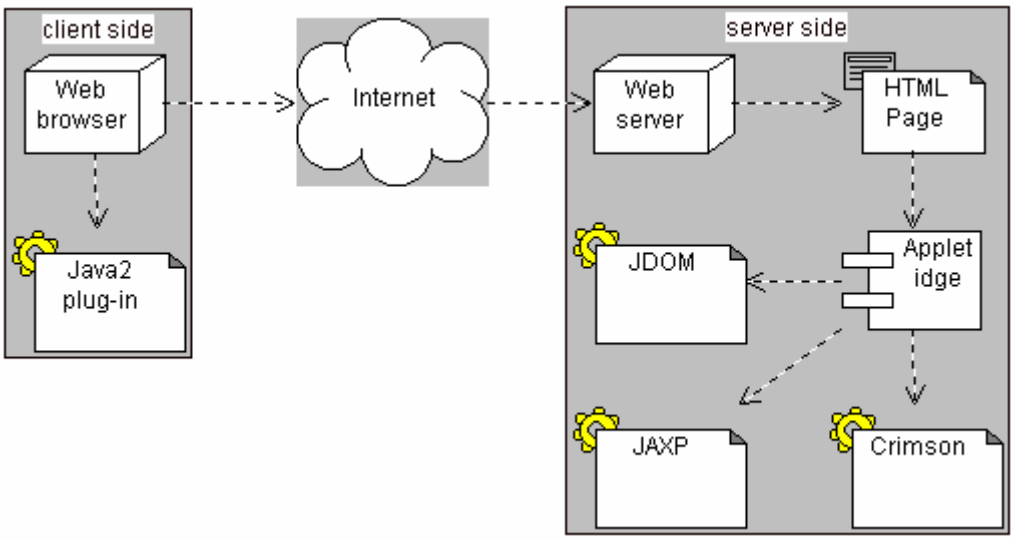


FIGURA 6.16 - Visão geral de um ambiente cliente-servidor para Astrha/E

6.10 Conclusões

O protótipo construído implementa as funcionalidades básicas importantes para a demonstração das idéias centrais da pesquisa, permitindo:

- a manipulação de um ambiente gráfico que implementa hipertecnologia em nível de hipertextos;
- a manipulação de animações que permitem a quebra de sua seqüencialidade através da inserção de hiperligações;
- através das funcionalidades acima listadas, permite a manipulação de características hipermídia e animadas em um único ambiente interativo, passando a compor hiper-animações;
- sincronização de mídias visuais com auditivas;
- definição da cor de fundo de tela;
- a interpretação de textos e hipertextos estilizados, permitindo-se dimensionar fontes, escolher seu tipo, cor de fundo de tela e formato de apresentação (negrito, itálico, sublinhado);
- controle opcional do ambiente pelo usuário através de um painel.
- o tratamento de transições reflexivas com semântica NOP.
- o tratamento de transições não-determinísticas, em níveis de estado e de palavra de saída, com semântica de pseudo-aleatoriedade;

Dessa maneira implementa as principais características propostas para validar: (a) modelo Astrha/M; (b) linguagem Astrha/L; (c) sua implementabilidade na Internet; (d) sua aplicabilidade, em especial para EAD.

7 Aplicabilidade

7.1 Resumo

Neste capítulo, são apresenta casos de uso implementados, em desenvolvimento e potenciais, tendo como foco principal exemplos em educação a distância. Ressalta-se que sua primeira vocação é servir, tal e qual o objetivo inicial do projeto, como um ambiente hipermídia, dinâmico, gráfico, voltado para aplicações Internet, estruturado a partir de autômatos finitos com saída.

7.2 Simulador de autômato

Nas pesquisas que temos realizado com o protótipo do Astrha/E, temos percebido, também, que o ambiente possui uma linguagem de marcação apropriada para produzir simuladores multimídia, em especial para máquinas abstratas finitas, uma vez que AFS possuem, por definição, um controle bem definido sobre estados.

Este estudo de caso, aplicado a linguagens formais, oferece ao aluno um exercício no qual, dada uma representação gráfica de um autômato finito, solicita-se que identifique a linguagem regular associada. Na tela inicial, um texto expõe o problema e mostra-se, ao seu lado, o autômato respectivo. Uma hiperligação é oferecida que, ao ser acionada, irá aceitar ou rejeitar a palavra de entrada que foi codificada no programa Astrha/L que o carregou. O aluno pode alterar essa fita quantas vezes desejar editando o programa em uma ferramenta XML e pressionando o botão Atualizar Ambiente *(Reload Config)*, e a seguir o botão Iniciar *(Start)*. Como neste exemplo exigem-se interações de controle de execução, o programa Astrha/L de carga solicita que o painel de controle é disponibilizado.

Note-se na **FIGURA 7.1** que foi o texto foi estilizado, oferecendo vários recursos como negrito, cor de fonte e itálico.

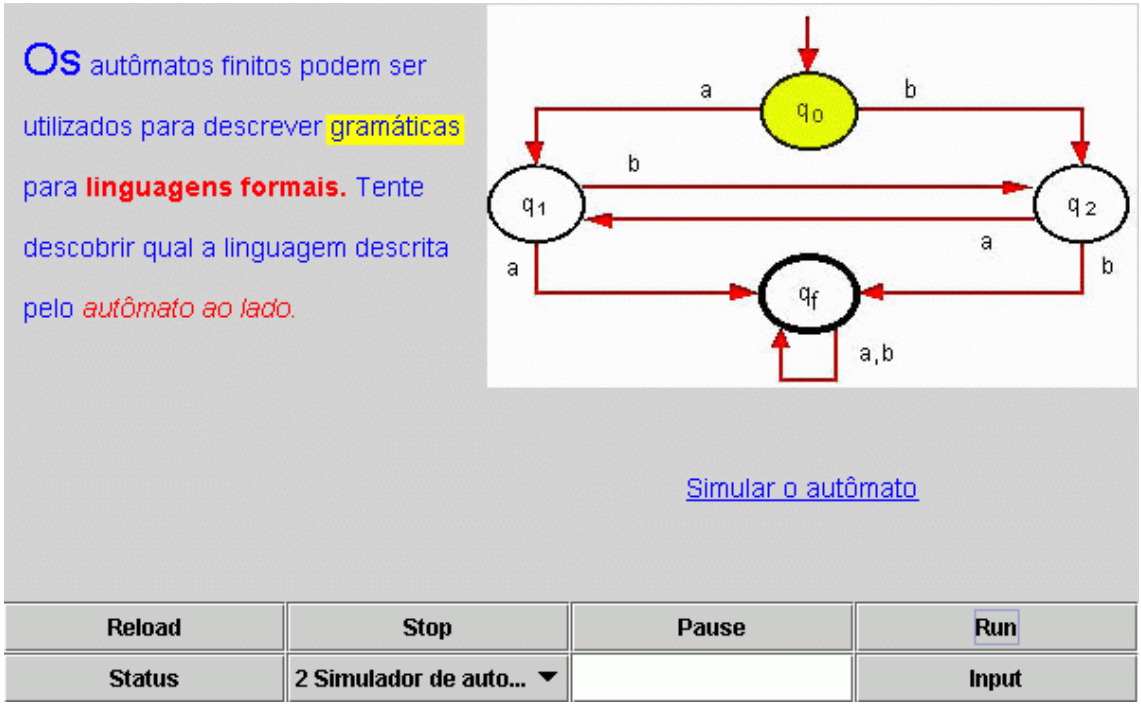


FIGURA 7.1 - Tela inicial de um exercício em linguagens formais

A FIGURA 7.2 mostra o programa escrito em dialeto *hyper*, linguagem Astrha/L, que gerou o texto explicativo da FIGURA 7.2. Pode-se perceber que hyper incorpora o dialeto *style*. Neste programa, são definidos quatro estilos, denominados *default, big, bold* e *italic* que são utilizados, respectivamente, para definir as fontes padrão, grande, negrito e itálica utilizadas. A fonte padrão possui cor azul e tamanho 14. A fonte grande tem tamanho 24 e também é azul. A fonte negrito é vermelha e tem tamanho 14; o negrito é identificado pelo caractere "b" no atributo *format*. Por fim, a fonte itálica é vermelha, possui tamanho 14 e o itálico é identificado pelo caractere "i" no atributo *format*. Diferentemente do HTML, cada troca de estilo é total, não permitindo que um estilo de fonte seja parcialmente trocado através de uma marca *(tag)* específica. A marca `<s/>` identifica o estilo a ser utilizado através do atributo id, que identifica o estilo desejado para seu texto ou hipertexto filho. As marcas `<s/>` não são aninhadas, contribuindo também dessa maneira para o quesito de legibilidade de código.

```
<!DOCTYPE hyper SYSTEM "../../../../../../../language/hyper.dtd">
<hyper styleId="1">
<styles>
  <style id="1" name="default">
    <text size="14" color="blue"/>
  </style>
  <style id="2" name="big">
    <text size="24" color="blue"/>
  </style>
  <style id="3" name="bold">
    <text size="14" color="red" format="b"/>
  </style>
  <style id="4" name="italic">
    <text size="14" color="red" format="i"/>
  </style>
</styles>
<p>
<s id="2"><t>Os</t></s>
<s id="1"><t>automatos finitos podem ser utilizados para descrever
gramaticas para </t></s>
<s id="3"><t>linguagens formais.</t></s>
<s id="1"><t>Tente descobrir qual a linguagem descrita pelo</t></s>
<s id="4"><t>automato ao lado.</t></s>
</p>
</hyper>
```

FIGURA 7.2 - Um programa escrito no dialeto *hyper* da linguagem Astrha/L

Ao se acionar a hiperligação "Simular o autômato", uma animação é dinamicamente montada e executada a partir da leitura de uma fita. Na FIGURA 7.3 é mostrado o resultado de uma simulação onde a palavra de entrada foi "a". O simulador rejeitou a entrada, uma vez que a linguagem regular associada ao autômato é `(a|b)*(aa|bb)* (a|b)*`, ou seja: para uma palavra ser aceita, deve conter pelo menos uma seqüência "aa" ou uma seqüência "bb".

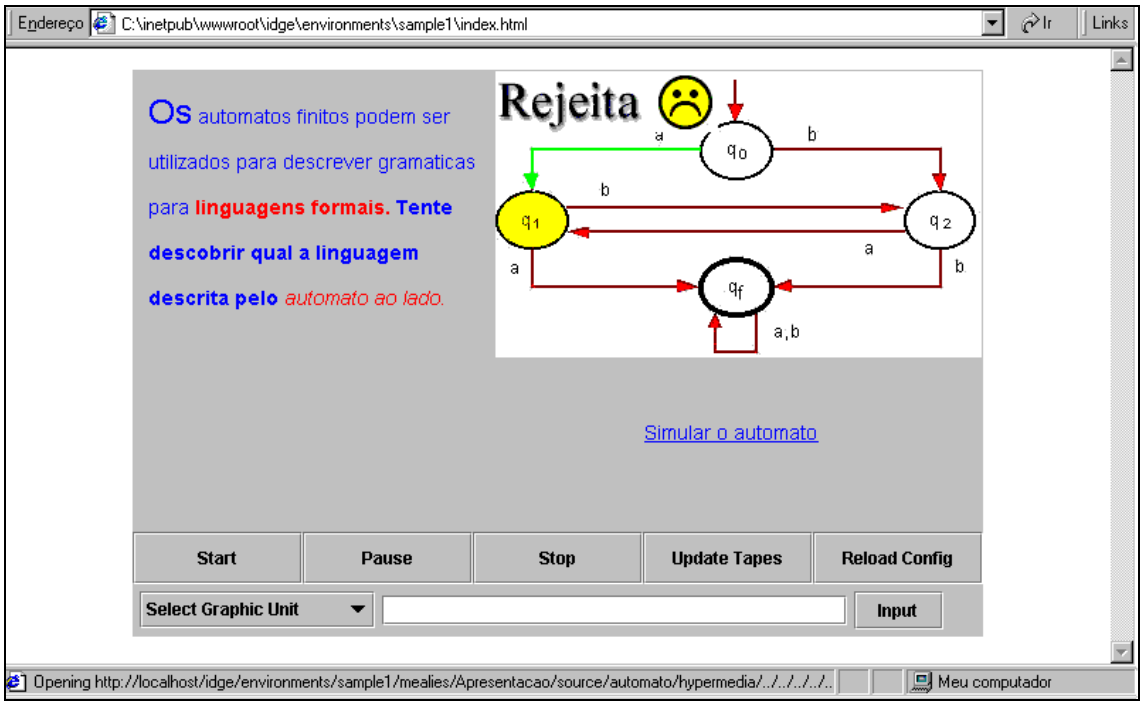


FIGURA 7.3 - Tela mostrando o resultado de uma simulação

7.3 Cobras animadas

O exemplo das cobras animadas é um conjunto de seis de hiper-animações construídas a partir de uma mesma Máquina de Mealy. Foi construído com o objetivo de demonstrar as seguintes características operacionais de Astrha/E:

- **não-determinismo**: o conjunto de transições escrito no dialeto *Mealy* mostrado na FIGURA 7.4 leva todas instâncias das cobras a um modelo não-determinístico, cujas palavras de saída são decididas no protótipo através da implementação de uma função pseudo-aleatória escrita em linguagem Java.

- **reflexividade:** esta característica pode ser implementada ao se definir as fitas das hiper-animações no dialeto *Environment*, uma vez que cada instância de hiper-animação pode definir algumas propriedades individuais, entre elas, a fita de entrada. A FIGURA 7.5 mostra o conjunto de hiper-animações das seis cobras, nas quais podemos perceber, em algumas fitas, a presença de símbolos de entradas `<input word=""/>` que denotam reflexividade, causando sua entradas das instâncias em tempos diferentes.

  O primeiro quadro mostrado pelo ambiente mostra apenas a hiper-animação número 3, uma vez que essa não possui nenhum símbolo de entrada reflexivo. No segundo quadro, aparecem as unidades 2 e 4, que possuem dois símbolos reflexivos. No terceiro, aparece a unidade 1. Uma vez que ambas as unidades 5 e 6 possuem quatro símbolos de entrada reflexivos no início da fita, entrarão apenas no instante 5. A inoperabilidade aparente causada pelo não-determinismo e a entrada gradual das hiper-animações causada pela reflexividade são mostradas na FIGURA 7.6.

```
<transitionFunction>
 <transition id="1" from="1" input="2" to="3" output="3"/>
 <transition id="2" from="1" input="2" to="2" output="2"/>
 <transition id="7" from="1" input="2" to="3" output="4"/>
 <transition id="8" from="1" input="2" to="2" output="3"/>
 <transition id="9" from="1" input="2" to="2" output="1"/>
 <transition id="3" from="2" input="2" to="2" output="2"/>
 <transition id="4" from="2" input="3" to="3" output="3"/>
 <transition id="5" from="3" input="4" to="4" output="4"/>
 <transition id="6" from="4" input="1" to="1" output="1"/>
</transitionFunction>
```

FIGURA 7.4 - Conjunto de transições formando um modelo não-determinístico

```
<hyperanimations>
 <hyperanimation id="1" begin="true" mealyId="1" initialState="3"
  name="Bicho rindo 1" execute="loop" x="10" y="0">
  <tape>
   <input word=""/>
   <input word=""/>
   <input word="4"/>
   <input word="1"/>
   <input word="2"/>
  </tape>
 </hyperanimation>
 <hyperanimation id="2" begin="true" mealyId="1" initialState="3"
  name="Bicho rindo 2" execute="loop" x="100" y="151">
  <tape>
   <input word=""/>
   <input word="4"/>
   <input word="1"/>
   <input word="2"/>
  </tape>
  </hyperanimation>
  <hyperanimation id="3" begin="true" mealyId="1" initialState="3"
   name="Bicho rindo 3" execute="loop" x="211" y="0">
   <tape>
    <input word="4"/>
    <input word="1"/>
    <input word="2"/>
   </tape>
  </hyperanimation>
  <hyperanimation id="4" begin="true" mealyId="1" initialState="3"
   name="Bicho rindo 4" execute="loop" x="301" y="151">
   <tape>
    <input word=""/>
    <input word="4"/>
    <input word="1"/>
    <input word="2"/>
   </tape>
  </hyperanimation>
  <hyperanimation id="5" begin="true" mealyId="1" initialState="3"
   name="Bicho rindo 5" execute="loop" x="412" y="0">
   <tape>
    <input word=""/>
    <input word=""/>
    <input word=""/>
    <input word=""/>
    <input word="4"/>
    <input word="1"/>
    <input word="2"/>
   </tape>
  </hyperanimation>
  <hyperanimation id="6" begin="true" mealyId="1" initialState="3"
   name="Bicho rindo 4"   execute="loop" x="502" y="151">
   <tape>
    <input word=""/>
    <input word=""/>
    <input word=""/>
    <input word=""/>
    <input word="4"/>
    <input word="1"/>
    <input word="2"/>
   </tape>
  </hyperanimation>
```

FIGURA 7.5 - Conjunto de hiper-animações com fitas que contêm símbolos de entrada denotando reflexividade

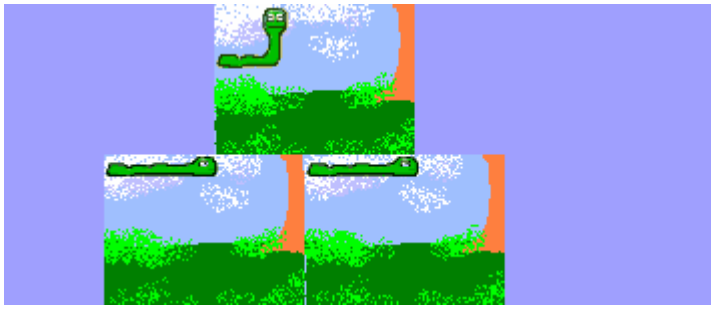
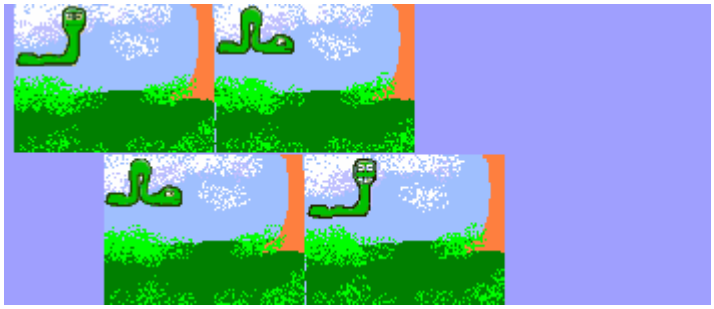
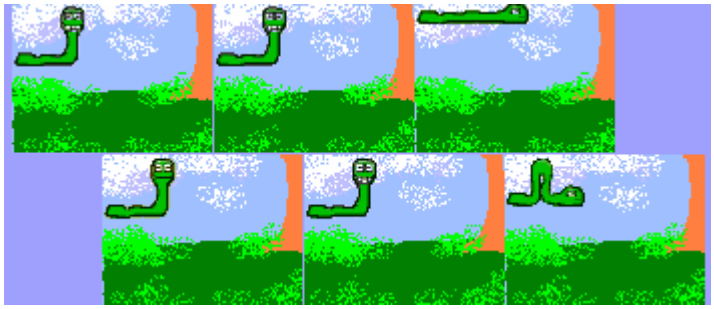
quadro 1, instante 1	
quadro 2, instante 2	
quadro 3, instante 3	
quadro 4, instante 5	

FIGURA 7.6 - Hiper-animações com propriedades não-determinísticas e reflexivas

7.4 Dicionário de Acordes Musicais

Um experimento realizado em conjunto com o Laboratório de Computação e Música (LC&M) do Instituto de Informática da UFRGS, que possui forte vocação EAD, foi a construção de um dicionário de acordes baseado em autômatos no ambiente Astrha, o qual gerou o artigo "Utilização do Ambiente Astrha para Implementar um Dicionário de Acordes Baseado em Autômatos Finitos [GRA 2003b]", que pode ser lido no Anexo VII - Artigo SBCM 2003 - desta dissertação.

Dentre as conclusões que pudemos tirar desse experimento, destacamos:

> "Pudemos perceber, através da implementação realizada, que Astrha é propício para dar a semântica necessária a um dicionário de acordes. Sua manutenção é fácil, uma vez que nenhum ajuste em programa Java é necessário para se aumentar, diminuir ou corrigir o autômato, bastando, para essas atividades, simplesmente editar o código no dialeto Mealy da linguagem Astrha/L. Dessa maneira, um músico que conheça a linguagem XML e possua orientações sobre a estrutura da linguagem Astrha/L, poderá editar os arquivos XML, criando seu próprio dicionário de acordes, personalizado.
>
> Outras vantagens do ambiente Astrha são a sua portabilidade, rodando em qualquer navegador Internet que possua Java *plug-in* versão 1.2 ou superior instalado. Além disso, o projeto de Astrha foi realizado com código aberto e gratuito, permitindo sua livre distribuição [GRA 2003b]."

Existem outros projetos interdisciplinares em potencial para serem explorados em cooperação com o LC&M. Alguns softwares de educação musical nesse laboratório que foram desenvolvidos em linguagem ToolBook, da Asymetrix, que têm tido bastante dificuldades para rodar em ambiente Internet, uma vez que seu *plug-in* para os navegadores disponíveis no mercado limitam muito sua funcionalidade, fazendo com que exercícios desenvolvidos percam sua capacidade de interação. Dentre esses softwares, podemos citar o Sistema de Treinamento Rítmico (STR) e o Sistema de Treinamento de Intervalos (STI) [KRÜ 99, HEN 98, FRI 98, FRI 96]. Ambos esses softwares possuem módulos gráficos, interativos, que apresentam exercícios em uma seqüência aleatória para que o aprendiz tente adivinhar qual o ritmo ou o intervalo musical apresentado de forma visual e/ou sonora. Com as características multimídia e não-determinísticas de Astrha, tais exercícios podem ser facilmente gerados e apresentados em ambiente Internet, potencializando o alcance dos exercícios para um grande número de aprendizes.

Note-se que o protótipo do Astrha/E foi elaborado compondo hierarquia de componentes Java, visando, também, uma potencial troca de componentes com outros sistemas EAD desenvolvidos pelo Instituto de Informática da UFRGS ou por outras instituições de ensino.

7.5 Conclusão

Os experimentos realizados através de estudos de caso com Astrha/E demonstraram que tanto Astrha/M como Astrha/L possuem a capacidade de representar hiper-animações; que a tecnologia empregada, Java applets, permitem expressar suas

principais características na Internet e, também, que são úteis e aplicáveis em programas de EAD. A utilização de código aberto e gratuito em todas as suas camadas facilitam a sua distribuição, favorecendo programas EAD, onde essa característica é desejável..

Mas sua aplicabilidade é ampla. Exemplos construídos com materiais instrucionais sobre teoria dos autômatos demostraram que Astrha/E, pelas suas características herdadas de Astrha/M e Astrha/L, serve-se bem como um ambiente propício para execução de simuladores, em especial aqueles descritos por gramáticas regulares.

Seu ambiente gráfico, com textos desenhados a partir de bibliotecas gráficas, dificulta operações do tipo copiar e colar textos, sendo útil para combater pirataria eletrônica e preservar direitos autorais dos conteúdos que estão sendo publicados.

Pode, simplesmente, expressar hipertextos com ligações estendidas, ou animações puramente seqüenciais. Ou, então, combinar o poder de ambos e formar hiper-animações. Todos com opções de sincronizar textos, sons e imagens, proporcionando estruturas altamente expressivas, no que se refere ao poder de comunicação, e com poder diretivo de interação através das hiperligações.

Permite, opcionalmente, o uso de um painel de controle com as funções de início, pausa e continuação da apresentação de quadros das hiper-animações. Através do painel de controle também é possível reprogramar e recarregar a entrada através da reescrita dos programas escritos em Astrha/L através de um editos XML qualquer.

Trata-se, portanto, de um ambiente gráfico, multimídia, dinâmico e interativo para Internet baseado em hiper-animações e autômatos finitos com saída que cumpre suas finalidades básicas: expressar hiper-animações via Internet e ser aplicável em programas de educação a distância que utilizam a Internet como meio de auxílio ao aprendizado.

8 Conclusões e Trabalhos Futuros

Esta última seção resume as conclusões obtidas em cada capítulo anterior, que podem ser relidas caso se deseje relembrar aspectos conclusivos importantes.

Também são sugeridos trabalhos futuros que poderão ser de interesse para o aprofundamento do assunto e aproveitamento dos conceitos criados.

8.1 Conclusões

A unificação dos conceitos de sistema hipermídia e animações através do conceito de hiper-animação permitiu que as características mais relevantes das especificações *Hyper-Automaton, Hyper-Automaton*: Avaliações Interativas, *eXtensible Hyper-Automaton* (XHA) e Animação Bidimensional para *World Wide Web* (AGA) fossem unificadas em nível conceitual e estruturadas através de uma definição sintática de Máquina de Mealy (Astrha/M), especializada para esse fim.

Especializada, também, a linguagem Astrha/L permite que os desenvolvedores de códigos nessa linguagem especifiquem, de forma simples e direta, hiper-animações, uma vez que a linguagem é de quarta geração.

Tanto Astrha/M como Astrha/L definem máquinas de Mealy reflexivas, com conjunto finito de palavras de saída, não-determinísticas em nível de estado e de palavra de saída. No protótipo Astrha/E a reflexividade possui semântica de inoperabilidade aparente (NOP) e o não-determinismo de escolha pseudo-aleatória, dando poder de programação ao desenvolvedor Astrha/L através da definição de transições convenientes, nas máquinas de Mealy, para a solução de problemas.

Ao se realizar os estudos de caso aplicados em educação a distância, pode-se comprovar a sua aplicabilidade em diversas áreas, desde o desenvolvimento de simuladores – em especial daqueles expressos por gramáticas regulares – passando por livros multimídia eletrônicos, exercícios de educação musical, até aplicações gráficas de relativa complexidade.

Tais definições, especificações, desenvolvimentos e estudos de caso atendem objetivos de unificação conceitual, construção e validação de sistemas hipermídia e de animações para Internet baseadas em autômatos finitos com saída do projeto *Hyper Seed*, do qual o Astrha é componente.

8.2 Trabalhos Futuros

Os trabalhos imediatamente vislumbrados para continuar esta pesquisa referem-se à continuação do projeto *Hyper Seed* [MEN 2002, p. 5]. Mais especificamente:

- Complementar as pesquisas do Astrha para máquinas de Moore, criando um modelo formal equivalente à Astrha/M e um dialeto *Moore*, à semelhança do dialeto *Mealy* de Astrha/L. Tal dialeto se tornará, dessa maneira, uma opção à mais para escrever hiper-animações. O dialeto *Environment* passará a ter como opções máquinas de Mealy ou de Moore, e o protótipo Astrha/E deverá ser capaz de ler dialetos *Moore*.
- Unificar as pesquisas do Astrha e do Nautilus, através de uma fundamentação matemática baseada na Teoria dos Autômatos e na Teoria das Categorias;
- Implementar um protótipo de sistema com características Astrha e Nautilus, com ênfase em aplicações de educação assistida por computador *(e-learning)*, em contextos de ensino (Instituto de Informática da UFRGS) e corporativo (empresa PLANCTA);
- Instanciação e validação de estudos de caso com esse novo modelo;
- Formalização e validação final do modelo e aplicação-chave desenvolvidos, e disseminação da aplicação-chave para outras instituições.

O dialeto Mealy (Moore) poderá ter sua funcionalidade ampliada contemplando todas as transformações gráficas definidas na pesquisa AGA, permitindo dessa maneira maior otimização dos recursos gráficos. AGA também propôs uma extensão de seu modelo denominado Animação Gráfica baseada em Autômatos Temporizados Sincronizados (AGA-S) para prover restrições temporais às especificações de seus atores e, também, para ampliar a funcionalidade de seu modelo quando à interação com o observador da animação, sugerindo para tal a utilização dos modelos de autômatos temporizados propostos por Alur e Dill [ACC 2002, ALU 94]. De forma semelhante, os autômatos definidos em Astrha poderão ser temporizados para aumentar a funcionalidade e adicionar restrições ao comportamento de hiper-animações.

Por fim, os dialetos de Astrha/L poderão receber novas marcações, aumentando sua semântica; o protótipo Astrha/E poderá evoluir passando a absorver toda a funcionalidade prevista nesta pesquisa, estudos de caso podem ser ampliados e validados, especialmente se aplicados à educação assistida por computador.

Anexo 1 – Projeto *Hyper Seed*

Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq

Dados do Projeto e do(a) Proponente

Sigla:	Hyper Seed
Título do Projeto:	Hyper Seed - Framework, Ferramentas e Métodos para Sistemas Hipermídia voltados para EAD via WWW
Referência da Chamada:	**2ª Fase** da Chamada Conjunta MCT/SEPIN – FINEP – CNPq 01/2002, Programa de Apoio à Pesquisa e Desenvolvimento e Inovação em Tecnologia da Informação – PDI-TI, **financiados pelo CNPq**
Linha(s) de atuação predominante(*):	(7) pesquisa e desenvolvimento ( ) projeto de demonstração (5) inovação tecnológica (3) transferência de tecnologia (4) formação e capacitação de talentos humanos (6) aplicações-chave ( ) estudo prospectivo
Coordenador do Projeto: **E-mail:**	Paulo Fernando Blauth Menezes blauth@inf.ufrgs.br
Instituição/Unidade Executora:	Instituto de Informática - Universidade Federal do Rio Grande do Sul
Data:	Porto Alegre, 12 de novembro de 2002

Palavras-chave: www, hipermídia, mídia adaptativa, teoria dos autômatos, teoria das categorias, ensino a distância

(*) Indicar a predominância da temática da proposta, utilizando-se de uma escala de um a sete. Exemplificando: caso uma proposta se apresente concentrada em três itens, estes receberiam pesos 7, 6 e 5, respectivamente, de acordo com sua ordem de domínio decrescente.

1. Identificação e caracterização do problema **(máximo de 1 página)**

Descreva, objetivamente, com o apoio do referencial teórico, a questão ou problema focalizado.

Com o desenvolvimento atual da Internet, começa a haver uma grande demanda por serviços via web. Uma das aplicações que pode tirar mais proveito das melhorias na capacidade de transferência de arquivos da internet e das novas mídias é a de ensino a distância [36]. Na verdade, algumas das críticas que se fazia ao ensino a distância, baseado tradicionalmente em correio convencional, rádio e televisão, em parte desapareceu com o surgimento do ensino a distância baseado em redes de computadores.

A partir da migração dos sistemas hipermídia em direção a Web, alguns autores perceberam que o novo ambiente não oferecia as condições necessárias para a criação de ambientes integrados de ensino. Como apontou [17], é preciso definir-se um sistema mais poderoso de WWW que seja capaz de estruturar o material hipermídia em módulos, que possam ser reutilizados em diversos contextos.

Assim, neste Projeto busca-se uma fundamentação matemática para a unificação, de maneira coerente e rigorosa, de especificações de hiperdocumentos e animações, e, baseado nesta fundamentação, desenvolver um protótipo e um conjunto de ferramentas para suporte ao desenvolvimento de conteúdo hipermídia/multimídia que permita uma integração natural a ambientes voltados para o Ensino a Distância (EAD) via WWW.

O modelo de hipermídia [30] a ser desenvolvido é baseado na representação da estrutura de uma hiperbase através de formalismos da Teoria de Autômatos [12]. Este modelo baseia-se no fato de que autômatos não somente capturam o poder descritivo dos grafos diretos, conhecidos por serem um abstração útil em sistemas de hipertexto [4], como também fornecem uma máquina abstrata para o controle e análise de hipertextos e animações, além de ser um formalismo universalmente aceito e possuir uma história de uso em muitas disciplinas como uma técnica de diagramação formal [6, 21]. Os autômatos podem ser representados por uma estrutura de grafo, a qual pode ser manipulada sobre vários aspectos. Grafos com uma semântica formal podem ser usados para prover interfaces de programação para o controle de material hipermídia. Diversos trabalhos relacionados ao uso de representações gráficas na modelagem de aplicações hipermídia utilizam outras construções, como Mapas Conceituais [9], Workflow [2], modelos Entidade-Relacionamento, Orientação a Objetos [10], e também Redes de Petri (o modelo Trellis [8] é um exemplo).

Adicionalmente, utilizando-se as ferramentas da Teoria das Categorias [15, 20] é possível definir precisamente um conjunto de operações composicionais sobre os componentes de uma hiperbase (hiperdocumento e animações), ou seja, construtores capazes de gerar novos elementos hipermídia sobre outros já existentes. Tal aplicação permitirá uma maior flexibilidade na construção e manipulação de elementos hipermídia na WWW.

2. **Justificativa** (máximo de 1 página)

Faça, de forma sucinta, um relato da situação-problema abordada, citando dados ou informações significativas que possam delimitar seu contexto histórico. Fundamente sua defesa e linha de atuação/tema indicando a relevância de seu projeto para o País, seu caráter inovador e porque seu projeto se enquadra dentro das exigências do Programa.

A educação científica e de qualidade capaz de sustentar o avanço tecnológico é sempre um objetivo a ser buscado por qualquer nação. Neste contexto, o advento do e-learning gerou grandes expectativas da popularização do ensino científico de qualidade. No entanto, o e-learning, na maioria dos casos, é uma simples transposição de conteúdos tradicionais (livros, revistas, etc) para Internet. Ou seja, as reais potencialidades ainda estão por serem exploradas pela média dos professores e empresas, principalmente se comparado aos meios de aprendizado (autodidatas/apoio) tradicionais.

É típico da Computação que qualquer solução gerada para abordar questões como esta será temporária, pois a dinâmica do surgimento de novas tecnologias necessariamente devem ser incorporadas com rapidez. Neste caso, freqüentemente a solução perde grande parte das suas qualidades originais, resultando em sistemas "inchados".

Historicamente, um dos principais objetivos a ser buscado na utilização dos fundamentos matemáticos na computação é o de inspirar sistemas melhores, mais simples, com implementações mais eficientes e robustas [35].

Assim, uma das premissas do Sistema Hyper Seed é desenvolver uma fundamentação matemática para a unificação, de maneira coerente e (matematicamente) rigorosa, de especificações (baseadas na Teoria dos Autômatos e Teoria das Categorias) de hiperdocumentos e animações, e, baseado nesta fundamentação, desenvolver um protótipo para suporte ao desenvolvimento de conteúdo hipermídia/multimídia que permita uma integração natural a ambientes voltados para o EAD via WWW.

O sistema HyperSeed permitirá que o conteúdo disposto em uma base de dados seja apresentado em diversos níveis de amplitude e profundidade. Isto permitirá que disciplinas de diversos cursos possam dispor de uma base de conteúdos avançados para o desenvolvimento de disciplinas interativas, dirigidas pela demanda dos alunos e baseadas na construção do conhecimento.

Como resultado, é esperado que tais fundamentos e aplicação-chave contribuam para o avanço do estado-da-arte em desenvolvimento e formalização de sistemas hipermídia e induzam novas ferramentas e métodos que, quando usados de forma apropriada, reduzirão o custo e esforço de manutenção e desenvolvimento de bases de dados voltadas ao ensino a distância, bem como a disponibilização de material hipermídia e multimídia via WWW.

3. Plano de crescimento do Grupo de Pesquisa no contexto do Plano Estratégico da Instituição (máximo de 2 páginas)

Descrever as ações de participação efetiva do Grupo de Pesquisa no ambiente institucional, visando os desenvolvimentos acadêmico, científico e tecnológico da instituição, destacando relacionamentos internos e externos (oportunidades e ameaças entre parcerias, fornecedores, clientes etc.)

A formação de recursos humanos foi um dos pontos mais cuidados pelos Projetos que antecederam a este e foi um dos principais fatores que contribuíram positivamente para os resultados atingidos. Portanto, a idéia é dar continuidade e aprofundar este trabalho.

De fato, é realizado um trabalho de base, iniciando com alunos de graduação, já nos primeiros semestres do Curso, passando por alunos de Mestrado e de Doutorado, dentro de uma política de formação de recursos humanos a longo prazo. Nos últimos anos, o número de alunos de mestrado diretamente envolvidos nas atividades correlatas a este Projeto se tornou expressivo (oito), bem como o de bolsistas de iniciação científica, o qual tem variado de três a cinco. A nível de doutorado, correntemente estão envolvidos dois alunos.

Ao longo deste Projeto a ênfase é no crescimento da equipe de alunos principalmente no nível de doutorado, além do crescimento natural nos demais níveis, conforme estratégia do Instituto de Informática da UFRGS.

Observe-se que as publicações realizadas possuem, na sua maioria participação de alunos do pós-graduação e de IC, inclusive, como principal autor. Adicionalmente, a maioria das apresentações tanto em eventos nacionais como internacionais dos artigos publicados tem sido realizada pelos alunos. Ao longo deste Projeto, será dada ênfase maior às publicações em periódicos, sem comprometer o bom nível das publicações em eventos, conforme estratégia do Programa de Pós-Graduação do Instituto de Informática da UFRGS.

Independentemente do crescimento e amadurecimento do Grupo de Pesquisa em si, cada vez mais tem sido dada ênfase em trabalhos cooperativos com outros Grupos e Instituições, enriquecendo o trabalho como um todo. Observe-se que o Grupo de Pesquisa deste Projeto agrega pesquisadores de três institutos, de duas instituições, sendo uma delas do interior do estado.

Adicionalmente, o Grupo possui trabalhos direta ou indiretamente relacionados com este Projeto, envolvendo outros Grupos de outras instuituições, tais como:

a) Grupo do Prof. E. Hermann Haeusler da PUC-Rio;

b) Grupo do Prof. Ugo Montanari, Univ. Piza, Italia;

c) Grupo do Prof. Amílcar Sernadas do Instituto Superior Técnico em Lisboa, Portugal;

d) Grupo da Prof. Simone Costa da UNISINOS (RS);

e) Grupo dos Professores Rocha Costa e Graçaliz Dimuro da UCPel.

Neste Projeto, além de reforçar as cooperações já existentes, estão sendo tratadas outras cooperações internacionais em particular com pesquisadores da França e dos EUA. Neste contexto, será especialmente incentivado o doutorado sanduíche.

Portanto, como pode ser observado, além do crescimento natural do Grupo e consolidação dos procedimentos que, comprovadamente, vem dando bons resultados, ao longo deste Projeto será dada uma atenção especial à formação de recursos humanos a nível de doutorado, ao enriquecimento e melhoria da qualidade do trabalho via cooperação internacional. Adicionalmente, trata-se de um Projeto com fortes aplicações em Ensino a Distância, de acordo com o Plano Estratégico da UFRGS, conforme pode ser comprovado na página oficial da UFRGS (http://www.ufrgs.br/ufrgs/), sendo que o Grupo é explicitamente referenciado (Projeto TEIA, o qual engloba o Hyper-Automaton e o AGA).

4. **Objetivos e metas** (máximo de 2 páginas)

Indique os objetivos e metas plurianuais a serem alcançados pelo projeto, num horizonte mínimo de dois anos.

a) Objetivos

Os objetivos deste projeto são:

1. Baseado nos modelos formais desenvolvidos e nos resultados atingidos em três projetos de pesquisa desenvolvidos pelo Grupo
   (a) Hyper-Automaton, um sistema adaptativo de disponibilização de conteúdo hipertexto;
   (b) AGA, Animação 2D Baseada em Autômatos para a Web;
   (c) Nautilus, linguagem de especificação Orientada a Objetos, com facilidades não-tradicionais inspiradas em Teoria das Categorias;

   desenvolver uma fundamentação matemática para a unificação, de maneira coerente e (matematicamente) rigorosa, de especificações de textos e animações, baseada na Teoria dos Autômatos e das Categorias.
2. Baseando-se em tais fundamentos, desenvolver um protótipo de sistema com suporte ao desenvolvimento de conteúdo multimídia e hipermídia com ênfase em aplicações de e-learning.
3. Validar este protótipo em dois contextos:
   (a) Num contexto típico de instituição de ensino, junto ao Instituto de Informática da UFRGS;
   (b) Num contexto coorporativo, junto à empresa PLANCTA (www.plancta.com.br), em experiências de massa, estimado em pelo menos 15.000 usuários.

b) Metas

Ano 1: Desenvolvimento do framework matemático e especificação do protótipo.

Ano 2, semestre 1: Implementação do protótipo da aplicação-chave e definição dos estudos de caso.

Ano 2, semestre 2: Instanciação e validação dos estudos de caso.

Ano 3, semestre 1: Formalização e validação final do modelo e aplicação-chave desenvolvidos, e disseminação da aplicação-chave para outras instituições.

5. **Metodologia e estratégia de ação** (máximo de 2 páginas)

Descreva a metodologia a ser empregada na execução do projeto e de que maneira a estratégia adotada contribuirá para alcançar os objetivos propostos.

Devemos abordar a metodologia/estratégia de ação em duas partes. A primeira relaciona-se com o desenvolvimento da estrutura teórica e a segunda com a implementação das ferramentas e do protótipo.

A estrutura teórica que será trabalhada neste projeto está bem fundamentada na teoria dos autômatos e em teoria das categorias. O uso de teoria dos autômatos para definir hipermídia fornece uma estrutura matematicamente bem definida e flexível. Teoria das categorias possui um conjunto de conceitos poderosos para a especificação de transformações sobre elementos hipermídia.

A metodologia de trabalho com estes referenciais teóricos deve considerar a literatura corrente na área. Devemos lembrar que a equipe possui muita experiência com trabalhos neste área, o que pode ser constatado nas publicações do grupo.

Com relação à implementação das ferramentas e do protótipo, ambos devem seguir a metodologia de versões sucessivas, já consagrada neste tipo de aplicação.

A avaliação dos resultados e do protótipo deve considerar aspectos qualitativos previamente definidos pela equipe, baseados em sua experiência prática na implementação de programas e páginas WWW, com ênfase na facilidade de uso e flexibilidade das ferramentas na aplicação proposta. Deverão ser consideradas na avaliação melhorias no processo de desenvolvimento de conteúdo hipermídia e na qualidade do resultado final da aplicação destas ferramentas.

O grupo será coordenado pelo Prof. Dr. Paulo Blauth Menezes, do Instituto de Informática da UFRGS, pesquisador do CNPq, e coordenador dos três projetos antecessores que constituem a base desta proposta, estando portanto integrado e capacitado para coordenar o projeto proposto. Pesquisadores da UFRGS, UFPel e da empresa privada PLANCTA participarão do projeto como líderes e colaboradores do Grupo. Os principais pesquisadores envolvidos são os seguintes:

- Prof. Dr. Paulo Blauth Menezes, do Instituto de Informática da UFRGS, é Doutor pelo Instituto Superior Técnico (Portugal), pesquisador do CNPq e coordenador dos três projetos antecessores que constituem a base desta proposta. Atua em Semântica Formal, Modelos para Concorrência, Informática na Educação, Linguagens Formais, Teoria da Computação e Teoria das Categorias.
- Prof. Dr. Tiaraju Asmuz Diverio, do Instituto de Informática da UFRGS, é Doutor pelo PPGC/UFRGS, pesquisador do CNPq e colaborador em diversos projetos. Atua em Processamento de Alto desempenho, Matemática Intervalar, Computação Científica, Ambientes de Alto Desempenho, Teoria da Computação, Complexidade de Algoritmos, Informática na Educação e Educação a Distância.
- Profa. Dra. Carla Maria Dal Sasso Freitas, do Instituto de Informática da UFRGS, é Doutora pelo PPGC/UFRGS, com Pós-Doutorado no International Computer Science Institute- EUA. Atua nas áreas de computação gráfica, visualização científica, visualização de informações e realidade virtual.
- Profa. Dra. Luciana Porcher Nedel, do Instituto de Informática da UFRGS, é Doutora pelo Swiss Federal Institute of Technology em Lausanne, Suíça. É

pesquisadora do Grupo de Computação Gráfica da UFRGS, e desenvolve trabalhos na área de animação por computador.

- Prof. Dr. Julio Alberto Nitzke, do Instituto de Ciência e Tecnologia dos Alimentos da UFRGS, é Doutor em Informática na Educação pela UFRGS. Atua nas áreas de desenvolvimento de ambientes de aprendizagem colaborativa apoiada por computador, educação a distância e tecnologia de alimentos.
- Carlos Campani, da UFPel, é Mestre e Doutorando em Ciência da Computação pelo PPGC/UFRGS. Atua em Algoritmos, Complexidade de Algoritmos, Linguagens Formais e Teoria da Computação.
- Júlio Pereira Machado, do Instituto de Informática da UFRGS, é Mestre e Doutorando pelo PPGC/UFRGS. Atua em Semântica Formal, Modelos para Concorrência, Informática na Educação, Linguagens Formais e Teoria das Categorias.

Além dos pesquisadores acima descritos, o Grupo será constituído de bolsistas nos níveis de graduação, pós-graduação e DTI.

6. Cronograma físico-financeiro (máximo de 1 página)

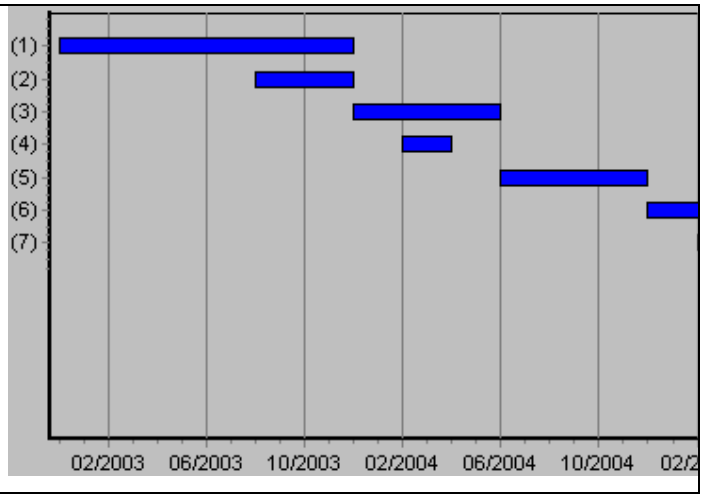
Apresente, graficamente, em seqüência cronológica, as etapas físicas do projeto, indicando seus prazos prováveis de execução e as respectivas previsões de despesa para cada uma delas. Relatórios, testes, solicitação de recursos (pessoal, material e financeiro), requisições (passagens, hospedagens e outras necessidades de pessoal), realização de eventos etc, se considerados relevantes para o andamento do processo, deverão ser informados neste cronograma.

	Valor Solicitado R$ 211.421,50		
	ANO 1	ANO 2	ANO 3 (primeiro semestre)
Custeio	Material de Consumo R$ 2.000,00 Viagens (N) R$ 8.200,00 Viagens (I) R$16.000,00	Material de Consumo R$ 2.000,00 Viagens (N) R$ 8.200,00 Viagens (I) R$16.000,00	Material de Consumo R$ 1.000,00
Capital	2 Servidores http R$ 20.000,00 1 Computador R$ 5.000,00 PeriféricosR$ 1000,00 Equipamentos de rede R$ 4.000,00 Softwares R$ 10.000,00 Bibliografia R$ 7.000,00	Atualização de Equip. R$ 2.000,00 Atualização dos Softwares R$ 2.000,00 Equipamentos de rede R$ 1.000,00 Bibliografia R$ 2.000,00	Atualização de Equip. R$ 1.000,00 Atualização dos Softwares R$ 1.000,00 Bibliografia R$ 1.000,00
Bolsas	2 DTI (H) R$ 20.833,92 1 DTI (G) R$ 12.550,68 1 SPE R$ 17.600,00 (4 meses)	2 DTI (H) R$ 20.833,92 1 DTI (G) R$ 12.550,68	2 DTI (H) R$ 10.416,96 (6 meses) 1 DTI (G) R$ 6.235,34 (6 meses)
Total	124.184,60	66.584,60	20.652,30

Os itens de material de consumo, software e hardware são para viabilizar a atualização dos equipamentos existentes e, principalmente, acomodar a equipe que efetivamente trabalha no Projeto a qual cresceu significativamente nesse período graças ao sucesso dos resultados obtidos. Os itens de periféricos (scanner, impressora) são necessidades gerais usuais e necessárias nesse tipo de Projeto. O item bibliografia inclui uma atualização bibliográfica para os estudos teóricos e manuais para os softwares a serem utilizados.

A possibilidade de compartilhamento da maioria dos recursos, é quase total (ver descrição dos recursos acima), não só para o Instituto de Informática como para toda a UFRGS e a comunidade acadêmica em geral. Os demais recursos (os não compartilháveis diretamente), como todos os recursos existentes no Instituto, sempre que estiverem ociosos, serão liberados para outros Projetos, cursos, etc.

Considerando o tipo dos recursos solicitados e o estágio atual em que se encontram as atividades em desenvolvimento, é importante que todo ou sua significativa maioria seja liberado no início do Projeto. Ou seja, idealmente, haveria somente uma etapa de grande liberação de recursos para o custeio e capital e os demais recursos de custeio e capital seriam liberados em mais duas etapas no início do segundo e terceiro ano do Projeto. Quanto ao valor de bolsas, este estaria diluído mensalmente nos 30 meses de vigência do Projeto.

(1) Desenvolvimento do Framework (2) Especificação do Protótipo (3) Implementação do Protótipo (4) Definição de Estudos de Caso (5) Instanciação e Validação (6) Formalização e Validação Final (7) Disseminação da Aplicação Chave	

7. Relevância dos resultados e os impactos esperados (máximo de 2 páginas)

Descreva os resultados e/ou produtos esperados. Avalie a repercussão e/ou impactos (sócio-econômicos, técnico-científicos e ambientais) desses resultados na solução do problema focalizado.

a) Indicadores de resultados (ao final do projeto)

Pretende-se ao final do projeto ter produzido e publicado um bom número de artigos científicos que apontem para a corroboração dos resultados teóricos na comunidade científica internaci-onal e nacional. Artigos descrevendo a aplicabilidade do protótipo de ferramenta a ser construído e os diversos estudos de caso validarão a sua adequação como proposta para seu posterior desenvolvimento comercial. Estes últimos descreverão a forma com que cada estudo de caso foi conduzido e seu uso pela ferramenta. Um relatório final fechará o projeto no que diz respeito a produção acadêmica.

O desenvolvimento deste projeto, principalmente na área teórico-formal, será fortemente centrado na formação de doutores, mestres e graduados em Ciência da Computação com forte apoio de bolsistas de DTI e IC. Como trata-se de um projeto que em diversos momentos é multidisciplinar, será dada continuidade à política correntemente utilizada pelo Grupo de colaboração e/ou co-orientação com outros grupos complementares, tanto da UFRGS quanto de outras instituições.

O produto da aplicação chave constitui-se em um sistema para composição e manipulação de hiperdocumentos na Web, fundamentado em conceitos teórico-formais, com destaque para a Teoria dos Autômatos e a Teoria das Categorias, incluindo um modelo para representação de animações gráficas que proporciona a estruturação do conteúdo multimídia. Para o sistema Hyper Seed será desenvolvido um manual do usuário e de instalação do software. Este manual apresentará ainda a metodologia a ser utilizada para o desenvolvimento de material hipermídia sobre o sistema apresentado.

b) Indicadores de progresso (ao final de cada 12 meses de projeto)

Indicadores de progresso se darão a partir dos workshops internos de avaliação de resultados. Nestes workshops os integrantes do projeto farão relatórios de atividades de condução da pesquisa/desenvolvimento a seu cargo. Após cada workshop um relatório conjunto, com anuência de todos os integrantes do grupo será redigido, não só como uma forma de tornar todos os pontos claros a totalidade dos integrantes da pesquisa, mas também com o objetivo de firmar acordo nas etapas posteriores a serem realizadas. É claro que resultados de condução da pesquisa por cada integrante individualmente ,ou na parcela do grupo, devem e serão submetidos a foro internacional e/ou nacional especializado para corroboração e validação da pesquisa do grupo. Serão realizados um total de 3 workshops:

- O primeiro, ao término do Ano 1, onde será apresentado como resultado da etapa de pesquisa o framework matemático e a especificação do protótpio;
- O segundo, ao término do Ano 2, onde serão apresentados, além dos resultados da implementação do protótipo e validação dos estudos de caso, também metas de adequação do framework e do protótipo para a terceira e última etapa;
- O terceiro, ao término do primeiro semestre do Ano 3, onde será apresentado o relatório final de conclusão do Projeto.

A parte de implementação do protótipo de ferramenta terá a avaliação de progresso medida, além da submissão/publicação de artigos técnico-científicos, por relatório do grupo de usuários que conduzem os estudos de caso. Esperamos contar com a empresa PLANCTA para a condução desta etapa.

c) Repercussão e/ou impactos dos resultados

Conforme já introduzido, é esperado que os fundamentos teóricos e aplicação-chave contribuam para o avanço do estado-da-arte em desenvolvimento e formalização de sistemas hipermídia e induzam novas ferramentas e métodos que, quando usados de forma apropriada, reduzirão o custo e esforço de manutenção e desenvolvimento de bases de dados voltadas ao ensino a distância, bem como a disponibilização de material hipermídia e multimídia, via WWW.

Como principal produto teórico-formal, espera-se obter um framework matemático (baseado em Teoria dos Autômatos e Teoria das Categorias) capaz de unificar os três frameworks desenvolvidos nos projetos antecessores, suficientemente expressivo para tratar todas as questões aqui propostas.

Como principal produto aplicado, espera-se obter um sistema semi-automatizado de hipermídia e multimídia na WWW, que incorpore os resultados do modelo formal desenvolvido através de modernas técnicas de engenharia de software (com ênfase em orientação a objetos e composição de sistemas) a ser disponibilizado à sociedade através da caracterização do mesmo como software livre.

Espera-se que este trabalho contribua para que a comunidade de pesquisadores tenha condições de dominar o conjunto de tecnologias de aquisição, armazenamento, processamento, exibição, animação e distribuição de informação, englobando modernas tecnologias de manipulação de informações (mais notadamente a grande gama de conceitos e aplicações em torno do XML).

Desta forma, a expectativa é de uma forte interação com a indústria no segundo ano do projeto, a partir de estudos de caso a serem utilizados na validação da metodologia definida e a transferência de tecnologia para a empresa parceira.

8. **Riscos e dificuldades** (máximo de 1 página)

Informe as dificuldades e riscos que poderão interferir na execução das ações propostas e comprometer o alcance das metas e objetivos preconizados. Explicite as medidas previstas para contornar ou superar essas dificuldades.

A meta mais crítica no cronograma proposto é a detalhada no Ano 1. Entretanto, observe-se que o seu grau de incerteza é muito baixo, uma vez que os principais resultados teórico-formais parciais já foram obtidos em projetos anteriores nos últimos 7 anos, conforme explicitado no item Objetivos. Ou seja, o desenvolvimento do framework em questão é de fato uma unificação de três frameworks individualmente desenvolvidos e testados.

De forma análoga, cada projeto antecessor originou seu correspondente protótipo. Portanto, não são esperadas maiores incertezas na meta do Ano 2. Também neste momento será de fundamental importância a experiência aplicada e comercial da empresa colaboradora PLANCTA.

Entretanto, o desenvolvimento de grande parte das aplicações e atividades de apoio do Projeto deverá ser suportada por bolsistas. Ou seja, a implementação das bolsas das instituições de fomento é básica para garantir a continuidade plena do Projeto. Objetivando antecipar eventuais problemas, tem sido buscada, sempre que possível, uma diversidade de fontes de bolsas.

9. Experiências e eventuais financiamentos de projetos (máximo de 1 página)

a) Portfólio

Indique as áreas de atuação onde a Instituição proponente tenha demonstrado (nos últimos cinco anos) inequívocas competências ou experiências de sucesso que possam credenciá-la, dentre as demais instituições, a receber o apoio desejado.

Os grupos de pesquisa do Instituto de Informática, cadastrados junto ao CNPq, que desenvolveram projetos financiados por órgão de pesquisa são os seguintes:

- Arquitetura e Projeto de Sistemas Computacionais
- Bioinformática
- Computação Gráfica e Processamento de Imagens
- Concepcão de Circuitos Integrados
- Fundamentos da Computação e Métodos Formais
- Grupo de Tolerância a Falhas
- Informática na Educação
- Inteligência Artificial
- Processamento da Fala e Robótica
- Processamento Paralelo e Distribuído
- Redes e Automação Industrial
- Sistemas de Informação

b) Ações em andamento

Informar projetos de pesquisa dos quais participem membros da equipe proponente, incluindo o título, vigência, dedicação em hora/homem/mês, origem e valor do financiamento.

(Informe se uma proposta idêntica ou equivalente foi submetida a outra agência financiadora).

O líder do grupo participa dos seguintes projetos:

MEFIA - Mathematical and Engeneering Foundations for interoperability via Architecture, Órgão(s) Financiador(es): CNPq/NSF, Natureza do Financiamento: auxílio financeiro, Itens Financiados: R$ 60.000,00, bolsas de curta e longa duração (DTI), verba de custeio e equipamentos. Período: 1999 a 2002 (ver último relatório de progresso em [11]).

HoVer-CAM - Composicionalidade Horizontal e Vertical de Sistemas Concorrentes e Auto-Modificantes, Órgão Financiador: CNPq, O principal objetivo deste Projeto Integrado é atingir, desenvolver, testar, comparar e aplicar a composicionalidade tri-diagonal para sistemas concorrentes e comunicantes relativamente à reificação e auto-modificação.

QaP-For Aplicação de Métodos Formais para a Melhoria da Qualidade e Produtividade em Software, Órgão Financiador: FAPERGS, Natureza do Financiamento: consolidação de grupo de pesquisa, Itens Financiados: R$ 108.000,00.

Outros projetos nos quais os integrantes do grupo de pesquisa estão engajados:

ARCA - Ambiente de Realidade Virtual Cooperativo de Aprendizagem, do Pós-Graduação em Informática na Educação da UFRGS, que visa criar um ambiente cooperativo de realidade virtual para aprendizagem de alimentos. Período: 1999 a 2002.

10. Atendimento aos critérios da chamada (máximo de ½ página)

Destaque os aspectos relevantes de sua proposta que justifique seu enquadramento nos critérios da Chamada.

Este projeto claramente se enquadra nas vertentes Aplicações Chaves e Fundamentos Científicos. Nesta linha de desenvolvimento, a equipe responsável pelo Projeto tem, reconhecidamente, experiência e conhecimento.

A principal aplicação do Projeto, em ensino a distância, é de grande relevância econômico-social para o nosso país, e está na ordem do dia das questões mais importantes para o desenvolvimento social do Brasil.

Dada a característica da aplicação pretendida e o valor solicitado, este Projeto se classifica como de Porte Pequeno (Faixa C). No entanto, ele possui muitos eixos de atuação, que incluem desde a pesquisa de fundamentação teórica até a implementação do protótipo.

11. Considerações finais (máximo de ½ página)

Informe, caso julgue necessário, outros critérios que possam ser considerados na avaliação de sua proposta (além dos constantes da Chamada) e, sucintamente, alguma informação adicional que, a seu juízo, seja relevante para a elucidação, compreensão ou apreciação de seu projeto.

O grupo de pesquisa proponente possui experiência na formalização de modelos e desenvolvimento de sistemas para a WWW como demonstram algumas das publicações relevantes a este projeto obtidas por seus integrantes e listadas no item Referências Bibliográficas (a lista completa das publicações pode ser obtida na Plataforma Lattes do CNPq). Observe-se que diversas publicações são em conjunto com outros grupos de pesquisa de instituições nacionais e internacionais.

O coordenador do projeto possui também forte atuação na formação de recursos humanos nesta área de pesquisa, incluindo a conclusão, no último ano, da orientação de três projetos de graduação e quatro trabalhos de mestrado, além de outros em andamento.

O principal produto já desenvolvido pelo grupo encontra-se na página http://teia.inf.ufrgs.br e se constitui em um sistema semi-automatizado para a organização de hiperdocumentos instrucionais e um *player* para animações baseadas em autômatos. Este sistema está atualmente sendo utilizado como forma de suporte ao ensino de disciplinas no curso de Ciência da Computação da UFRGS e pode ser encontrado no link http://teia.inf.ufrgs.br.

Adicionalmente, com a parceria proposta com uma empresa privada, este Projeto garante que grande parte da tecnologia desenvolvida possa ser testada e utilizada em contextos comerciais normais.

Finalmente, devemos lembrar o desafio que representa o trabalho proposto neste projeto. A área envolve uma grande quantidade de conhecimentos de áreas distintas, evidenciando-se assim um caracter interdisciplinar e multidisciplinar.

12. Referências bibliográficas

Relacione as obras do referencial teórico citado, de acordo com as normas da ABNT.

[1] ACCORSI, F.; MENEZES, P. F. Blauth; NEDEL, L. P. Animacão Gráfica Baseada em Autômatos Temporizados Sincronizados. In: WMF'2001: IV Workshop de Métodos Formais, 2001, Rio de Janeiro. Anais do IV Workshop de Métodos Formais. Rio de Janeiro: SBC, 2001. v. 1, p. 75-80.

[2] ADELSBERGER, H.; KÖRNER, F. X.; PAWLOWSKI, J. M. A Conceptual Model for an Integrated Design of Computer Supported Learning Environments and Workflow Management Systems. In: XV IFIP World Computer Congress, 1998, Vienna. Proceedings of the XV IFIP World Computer Congress - Teleteaching 98. Vienna: IFIP, 1998. p.55-64.

[3] CARNEIRO, C. R. J. B.; REIS, R. Q.; MENEZES, P. F. Blauth. Especificação Formal de uma Ferramenta de Trabalho Colaborativo através da Composição de Objetos Nautilus. In: SBES'99: XIII SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 1999, Florianópolis. Anais do XIII Simpósio Brasileiro de Engenharia de Software. Florianópolis: SBC, 1999. p.95-110.

[4] CONKLIN, J. Hypertext: An Introduction and Survey. IEEE Computer, v.20, n.9, p.17-41, Set. 1987.

[5] DIVERIO, T. A., MITO, I. V. Foundations for Virtual Enviroments to Support Theory of Computation Teaching In: Advances Educational Technologies: multimedia, www and distance education ed.New York : John Wiley & Sons, 2001

[6] DIVERIO, T. A.; MENEZES, P. F. B. Teoria da Computação: máquinas universais e computabilidade.2 ed. Porto Alegre : Instituto de Informática da UFRGS, Editora Sagra Luzzatto, 2000, v.1. p.224.

[7] FREITAS, C. M. D. S., CHUBACHI, O. M., LUZZARDI, P. R. G., CAVA, R. A. Introdução à Visualização de Informações. Revista de Informática Teórica e Aplicada. Porto Alegre, RS, v.8, n.2, p.143 - 158, 2001.

[8] FURUTA, R.; STOTS, P. D. Programmable Browsing Semantics in Trellis. In: Proceeding of the Hypertext'89, 1989, p.27-42.

[9] GAINES, B. R.; SHAW, M. L. G. Concept Maps as Hypermedia Components. 1996. http://ksi.cpsc.ucalgary.ca/articles/ConceptMaps/

[10] GARZOTTO, F.; SCHWABE, D.; PAOLINI, P. HDM: A Model Based Approach to Hypertext Application Design. ACM Transactions on Information Systems, v.11, n.1, p.1-26, January 1993.

[11] [HAE 2001] HAEUSLER, E. H.; MESEGUER, J. Mathematical and Engineering Foundations for Interoperability through Architecture. In: Proceedings of the Projects Evaluation Workshop CNPq/NSF/Inria, Ed. Nelson Prugner, Paulo Ernesto Castilho Lima e Celso Deusdeti Costa., p. 175-208, 2001.

[12] HOPCROFT, J. E.; ULLMAN, J. D., Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 1979.

[13] MACHADO, C. C.; FEDERIZZI, G. L.; MENEZES, P. F. B. Flexibility and Adequacy of the Output's Layout of the Content Displayed in The Hyper-Automaton System. In: IC'2001: International Conference on Internet Computing, 2001, Las Vegas. 2nd International Conference on Internet Computing. Las Vegas: CSREA Press, 2001. v. 1, p. 424-430.

[14] MACHADO, J. H. A. P.; MORAIS, C. T. Q. de; MENEZES, P. F. Blauth; REIS, R. A. da Luz. Structuring Web Course Pages as Automata: revising concepts. In: RIAO'2000: Rechérche

d'Informations Assistée par Ordinateur, 2000, Paris. Content-Based Multimedia Information Access: conference proceedings. Paris: C.I.D., C.A.S.I.S., 2000. v.1, p. 150-159.

[15] MAC LANE, S. Categories for the Working Mathematician. Springer-Verlag, 1971.

[16] MANSSOUR, I. H., FURUIE, S. S., OLABARRIAGA, S. D., FREITAS, C. M. D. S. Visualizing Inner Structures in Multimodal Volume Data In: Brazilian Symposium on Computer Graphics and Image Processing, 2002, Fortaleza. SIBGRAPi 2002. Los Alamitos: IEEE Computer Society, 2002. v.1. p.51-58.

[17] MAURER, H. Necessary Ingredients of Integrated Network Based Learning Environments. In: ED-MEDIA/ED-TELECOM Educational Multimedia/Hypermedia and Telecommunications, Charlottesville, v.1, 1997, p.619-624.

[18] MENEZES, P. F. B.; COSTA, S. A. da; MACHADO, J. H. A. P.; RAMOS, J. Nautilus: a concurrent anticipatory programming language. In: CASYS'2001: 5th International Cconference on Computing Antecipatory Systems, 2001, Liège. AIP Conference Proceedings. Melville: American Institute of Physics, 2002. v. 627, p. 553-564.

[19] MENEZES, P. F. B.; MACHADO, J. H. A. P. Adaptive Web Courses: a Categorial Framework International Journal of Computing Anticipatory Systems, Liege, v.9, p. 318-336, 2001.

[20] MENEZES, P. F. B.; HAEUSLER, E. H. Teoria das Categorias para Ciência da Computação.1 ed. Porto Alegre : Instituto de Informática da UFRGS, Editora Sagra Luzzatto, 2001, v.1. p.324.

[21] MENEZES, P. F. B. Linguagens Formais e Autômatos.4 ed. Porto Alegre : Instituto de Informática da UFRGS, Editora Sagra Luzzatto, 2001, v.1. p.192.

[22] MENEZES, P. F. B.; MACHADO, J. H. A. P. Hyper-Automaton: hypertext framework with categorical operations. In: SBLP'2000: IV Simpósio Brasileiro de Linguagens de Programação, 2000, Recife. Anais do IV Simpósio Brasileiro de Linguagens de Programação. Recife: Centro de Informática da UFPE, 2000. p. 29-47.

[23] MENEZES, P. F. B.; MACHADO, J. H. A. P. Web Courses are Automata: a categorial framework. In: WMF'99: II Workshop de Métodos Formais, 1999, Florianópolis. II Workshop on Formal Methods. Florianópolis: UFSC, 1999. p. 79-88.

[24] MENEZES P. F. B. Compositional Reification of Concurrent, Interacting Systems. In: PDPTA'98: International Cconference on Parallel and Distributed Processing Techniques and Applications, 1998, Las Vegas. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas: CSREA Press, 1998, p. 1754-1761.

[25] MENEZES, P. F. B., SERNADAS, A., COSTA, J.F., Nonsequential Automata Semantics for Concurrent, Object-Based Language. In First US-BRAZIL Workshop on Formal Foundations of Software Systems, 1998, Rio de Janeiro Eletronic Notes in Theoretical Computer Science, Amsterdan Elsevier, 1998, v. 14.

[26] MENEZES, P. F. B.; COSTA, J. F. Synchronization in Petri Nets, Fundamenta Informaticae, Amsterdan, v. 26, n.1, p. 11-22, 1996.

[27] MORAIS, C. T. Q. de; MACHADO, J. H. A. P.; MENEZES, P. F. B.; REIS, R. A. da Luz. A Web Teaching System Based on Formal Methods. In: WCC'2000: 16th IFIP World Computer Congress, 2000, Beijing. Proceedings ICEUT, IFIP World Computer Congress. Beijing: PHEI, 2000. v. 1, p. 221-224.

[28] NEDEL, L. P.; THALMANN, D. Anatomical modeling of deformable human bodies. The Visual Computer, Heidelber, Germany, v.16, n.6, p.306-321, 2000.

[29] NEDEL, L. P.; FREITAS, C. M. D. S.; WAGNER, F. R. Simulação de objetos deformáveis baseada na análise dinâmica. Revista de Informatica Teórica e Aplicada, Porto Alegre, v.5, n.2, p.23-33, 1999.

[30] [NIELSEN 90] NIELSEN, J. Hypertext and Hypermedia. San Diego: Academic Press, 1990.

[31] NITZKE, J. A., CARNEIRO, M. L. F., FRANCO, S. R. K. Ambientes de Aprendizagem Cooperativa Apoiados pelo Computador e sua epistemologia. Informática na Educação Teoria e Prática, Porto Alegre, v.5, n.1, p.13-23, 2002.

[32] NITZKE, J. A., POLONIA, E., SLOCZINSKI, H., ZEVE, C., LIMA, J. V. A CD- ROM environment for collaborative learning integrated to the Internet In: Computers and Advanced TEchnology in Education, 2000, Cancun. Proceedings of the IASTED International Conference. Calgary: IASTED/ ACTA PRESS, 2000. p.297-302.

[33] NITZKE, J. A., MANFROI, V., SILVEIRA, A. E. Give us paper notes - Problems and solutions for a new strategy to food engineers based on computer supported collaborative learning In: nternational Conference on Engineering and Computer Education, 2000, SÃO PAULO. IEEEE - ICECE2000 - Proceedings. São Paulo: IEEE/ SENAC, 2000.

[34] PASQUALOTTI, A., FREITAS, C. M. D. S. Experimentação de Ambiente Virtual para Melhoria do ensino-aprendizagem de Matemática. Boletim de Educação Matemática. UNESP - Rio Claro, São Paulo: , v.14, n.16, p.79 - 101, 2001.

[35] [STO 77] STOY, J. E. Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory, MIT Press, 1977.

[36] [WEB 98] WEBER, T. et al. Uma Experiência com Hiperdocumentos e Internet no Suporte a Disciplinas de Computação, VI WEI - Workshop sobre Ensino em Informática, vol. 1, p. 532-545, Belo Horizonte, 1998.

Anexo 2 – Aspectos Históricos da Hipertecnologia

II.I Resumo

Este anexo apresenta uma revisão bibliográfica sobre aspectos históricos relacionados à hipertecnologia, um dos fundamentos deste trabalho, complementando o capítulo 2. Numa linha do tempo que inicia no Século XX, identifica sistemas gráficos que utilizam hipertecnologia como o *Aspen Movie Map, o NoteCards, o Intermedia, o HyperCard*, etc., até o surgimento da Internet. Mostra os primeiros ambientes que utilizaram SGML, a metalinguagem de marcação das linguagens HTML e XML. Percorre, complementarmente, algumas arquiteturas propostas para sistemas hipermídia, uma vez que a estruturação de hiper-animações - que implicitamente utilizam hipertecnologia – também compõe-se como uma das bases desta pesquisa.

II.II Projeto Xanadu™ e a Criação dos Termos Hipertexto e Hipermídia

Theodor H. Nelson, um filósofo nascido em 1937, visualizou um ambiente que denominou Xanadu[25] [XAN 99]. Trata-se de um projeto ambicioso e aprimorado. Foi apresentado à comunidade acadêmica em 1965 através do também clássico artigo *"A File Structure for the Complex, the Changing and the Indeterminate"* [NEL 65] .

A idéia central do projeto de Nelson era criar um ambiente global para literatura, um repositório permanente para todo e qualquer material escrito que se desejasse publicar [DEB 2002]. Permanente porque, uma vez publicado, o material jamais será removido. Xanadu guarda toda e qualquer versão de um documento publicado. Dessa maneira, se alguém em algum lugar do planeta criou uma ligação para aquele documento, ele pode ficar tranqüilo que sua referência será mantida. Por se tratar de um ambiente direcionado a publicações literárias, sua proposta propôs, também, uma novo modelo de tratamento aos direitos autorais, que denominou *transcopyright*.

Dentre as características funcionais mais importantes de seu projeto, pode-se citar:

- ligações não rompíveis: característica altamente desejável, sendo um dos maiores problemas dos sistemas hipertexto atuais;
- ligações bidirecionais: as ligações do sistema hipertexto padrão vigente, a World Wide Web (WWW), são unidirecionais, dificultando a navegação de ida e volta entre contextos. As ligações naturalmente bidirecionais resolvem bem esse problema;
- publicação incremental e controle de versões: por causa da permanência dos documentos *ad eternum* idealizada por Nelson, o sistema exige um sofisticado mecanismo de publicação incremental e de controle de versões.

---

[25] Em alusão ao poema "Kubla Khan", de Samuel T. Coleridge, onde Xanadu é um "lugar mágico da memória da literatura e da liberdade, no qual nada seria esquecido"

- ligações entre documentos baseada em grafos dirigidos: esta característica é substancialmente diferente da arquitetura hierárquica baseada em árvores do WWW. Cada abordagem tem suas as vantagens e desvantagens naturais de suas representações matemáticas.

- disponibilização de ferramentas para os mais diversos tipos de comparações entre de documentos: conteúdos, versões, comentários, etc.

Nesse artigo, Nelson também cunhou o termo hipertexto *(hypertext)* definindo-o como:

> "... um contexto composto de textos ou imagens interconectados de um modo complexo, de tal maneira que não poderia ser convenientemente representado em papel. Pode conter resumos ou mapas de seus conteúdos e suas inter-relações; pode conter anotações, adições e notas de rodapé de estudiosos que o examinarem [NEL 65]."

O termo hipermídia também foi apresentado como extensão do hipertexto, onde elementos multimídia (gráficos estáticos e animados, bi e tridimensionais, vídeos, sons), são adicionados ao hipertexto.

O Xanadu foi, inicialmente, desenvolvido na Brown University. Depois, o projeto foi adquirido pela empresa Autodesk[26]. Em vista do seu fracasso comercial em relação ao projeto WWW, a Autodesk perdeu o interesse no seu desenvolvimento, sendo readquirido por Theodor H. Nelson. Dada a sua complexidade, que exige um volumoso aporte de capital e de recursos humanos para o desenvolvimento de um produto estável, e considerando que o WWW, com suas virtudes e deficiências, domina o mercado atual, o Xanadu continua sendo um sonho a se concretizar, uma vez que ainda não se tornou uma realidade comercial.

[26] Empresa voltada a produtos de software de apoio à engenharia e proprietária da ferramenta Autocad™.

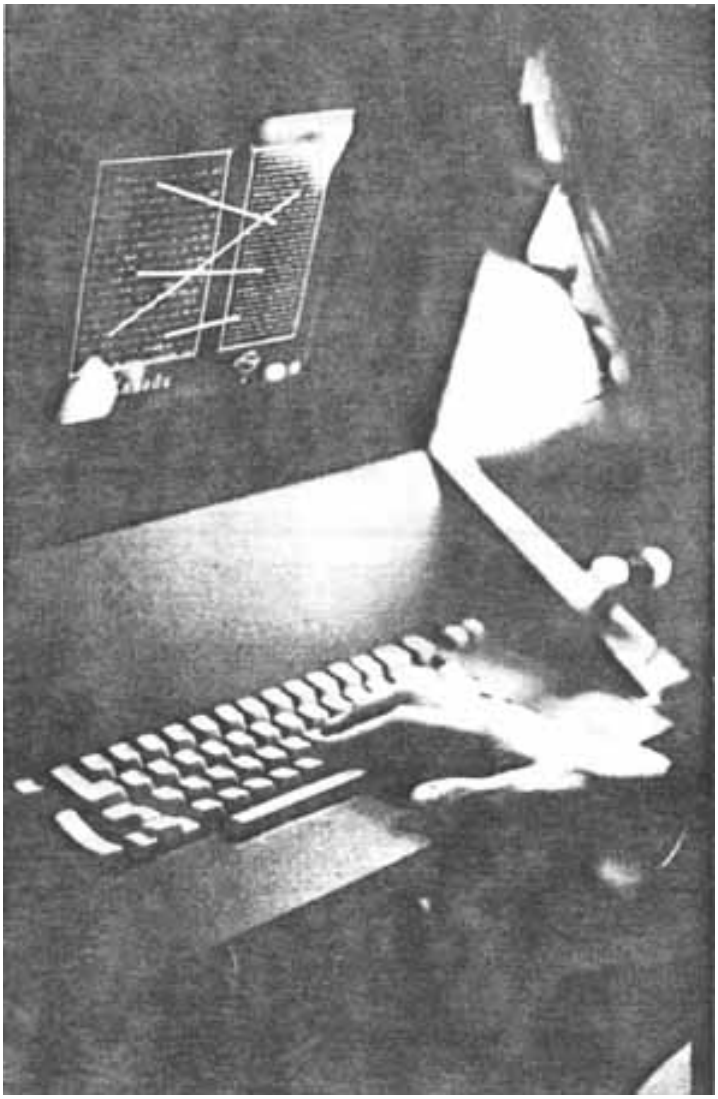


FIGURA 8.1 - Modelo de janelas comunicantes. Projeto Xanadu, 1972

II.III HES, FRESS e IGD

O *Hypertext Editing System* (HES) foi projetado na *Brown University*, em parceria com a *International Business Machines* (IBM), por uma equipe liderada por Andy van Dam e Theodor H. Nelson. A arquitetura de suas ligações era hierárquica e bidirecional. Utilizava um computador IBM 2250 e foi exibido em 1968. O sistema foi licenciado para o *Houston Manned Spacecraft Center*, da *National Aeronautics & Space Administration* (NASA), como ferramenta de documentação para o famoso programa espacial *Apollo*, que através da nave *Apollo 11* levou os primeiros seres humanos à Lua em 20 de julho de 1969.

O *File Retrieval and Editing System* (FRESS) é uma evolução do HES, também desenvolvido na *Brown University* em parceria com a IBM e com a coordenação de van Dam. Os aperfeiçoamentos foram, basicamente, operacionais, permitindo o uso simultâneo de vários terminais através de compartilhamento de fatias de tempo *(timesharing)*. Tornou-se disponível em 1969; na década de setenta, foi reintroduzido no mercado pela empresa Philips.

Um aprimoramento tecnológico do FRESS fez surgir uma terceira geração de sistemas hipertexto da *Brown University*, o *Electronic Document*, também conhecido como *Interactive Graphical Documents* (IGD) que, basicamente, passou a suportar melhores recursos hipermídia, permitindo a inclusão de imagens coloridas, e melhorou o sistema de navegação. Esses aprimoramentos foram coordenados por Steven Feiner, Sandor Nagy e Andries van Dam [DOE 91, DEB 2002, NAS 2002].

II.IV oN Line System (NLS)

Inspirado pelo artigo de Bush, Douglas Engelbart e um grupo de pesquisadores do *Stanford Research Institute* (SRI), iniciaram em 1962 o projeto *Augment*, assim batizado para expressar uma ferramenta que aumenta a produtividade e as capacidades humanas. Engelbart sempre considerou a usabilidade um dos itens mais importantes de um projeto, tornando-se um pioneiro da área da Interação Humano-Computador (IHC). Em 1963, escreveu o artigo "*A Conceptual Framework for the Augmentation of Man's Intellect*", propondo um sistema que denominou *Human using Language, Artifacts, and Methodology* (H-LAM/T), projetado no qual o usuário possui papel central.

Em dezembro de 1968, durante a *"Fall Joint Computer Conference",* nos Estados Unidos, o grupo realizou uma apresentação funcional e ao vivo do *oN Line System* (NLS), uma ferramenta experimental *Computer-Aided Software Engineering* (CASE) direcionada ao armazenamento e a recuperação dos documentos de especificação de um sistema computacional: planos, projetos, programas, relatórios, memorandos, bibliografia, notas de referência, etc.. Assim como o HES e o FRESS, a arquitetura de suas ligações era hierárquica e bidirecional.

Os terminais de operação eram bastante sofisticados, incluindo projetores de vídeo, teclados especiais e introduzindo o dispositivo apontador *(mouse)*, mostrado na FIGURA 8.2, uma das invenções mais celebradas de Engelbart. De seu *modus operandi* surgiram conceitos básicos para os navegadores *(browsers)* hipertextos de hoje. [DEB 2002, DOE 91].

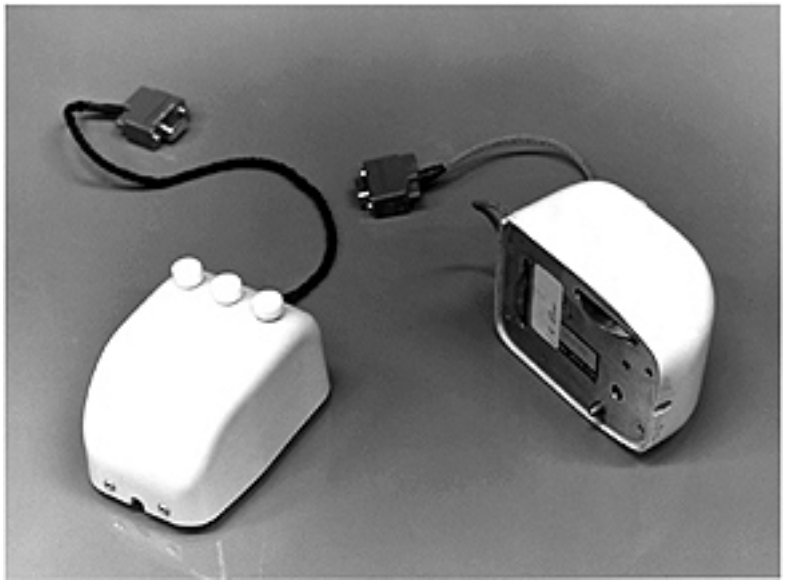


FIGURA 8.2 - Dispositivo apontador *(mouse)* inventado por Engelbart

II.V ZOG e KMS

A *Carnegie-Mellon University* (CMU) iniciou seu primeiro projeto hipertexto, ZOG, em 1972, tornando-se funcional três anos mais tarde, em 1975. A palavra ZOG não tem significado, tendo sido escolhida por sua sonoridade e facilidade de memorização. As pesquisas desse projeto foram lideradas por Donald McCracken, Robert Akscyn, George Robertson, Allen Newell e Kamesh Ramakrishna.

A preocupação principal do projeto ZOG era permitir o acesso simultâneo a uma grande comunidade de usuários, o que levou seus idealizadores a optar por um sistema multiusuário por *time sharing*. Suas ligações possuíam arquitetura basicamente hierárquica, uma vez que permitia algumas ligações de referência cruzada. Foi originalmente concebido para trabalhar em mainframes, tendo sido portado posteriormente para estações de trabalho PERQ. Era um sistema centrado em bancos de dados divididos em contextos denominados quadror *(frames)*, que consistiam em um título, uma descrição, uma linha com os comandos padrão ZOG e um conjunto de itens de menu denominados seleções *(selections)*, que conduziam o operador a outros *frames*.

Em 1982, o ZOG foi instalado no porta-aviões nuclear norte-americano USS Carl Vinson em 28 estações de trabalho PERQ, como campo de testes para manutenção de manuais e recuperação de informações.

Dois dos principais líderes do projeto, McCracken e Akscyn, fundarem em 1981 uma empresa denominada *Knowledge Systems* e criaram o *Knowledge Management System* (KMS), um sistema hipertexto baseado no ZOG, para estações de trabalho *Sun* e *Apollo* [DEB 2002, DOE 91].

II.VI Aspen Movie Map

O *Aspen Movie Map*, desenvolvido pelo grupo de Andrew Lippman no MIT em 1978, é especialmente importante em termos históricos porque foi o primeiro sistema hipertexto concebido para suportar características multimídia, tornando-se, assim, o primeiro sistema verdadeiramente hipermídia conhecido.

O sistema utilizava um conjunto de videodiscos contendo fotografias de todas as ruas da cidade de Aspen, no estado do Colorado, nos Estados Unidos. A gravação foi realizada através de quatro câmaras, focalizadas cada uma para um sentido diferente, colocadas sobre um caminhão, que registravam fotografias em distâncias de três em três metros. Para exibir as imagens, utilizava-se de dois monitores, um em posição vertical, para exibir o vídeo das ruas da cidade e outro em posição horizontal, que mostrava o mapa das ruas de Aspen.

O *Aspen Movie Map* foi um sistema de finalidades acadêmicas antes de ser comercial, sendo ainda hoje um dos mais sofisticados sistemas hipermídia existentes [DEB 2002].

II.VII Symbolic Document Examiner (SDE)

Os sistema hipertexto que foram construídos até a década de setenta foram direcionados ou a aplicações internas, empresariais, ou a aplicações militares, ou então para fins acadêmicos. Em 1982, Janet Walker liderou um para a Symbolics Technology Inc., denominado *Symbolic Document Examiner* (SDE), com o objetivo de criar um sistema hipertexto que pudesse ser utilizado pelo grande público através das estações de trabalho produzidas pela empresa, o que foi realizado em 1985.

Por esse motivo, este foi outro projeto que atribuiu grande importância à usabilidade. A fim de evitar a complexidade da interface do usuário leitor, foi criada uma interface especialmente produzida para a escrita de documentos hipertexto denominada Concordia tornando-se, provavelmente, a primeira ferramenta especialmente produzida para a autoria para sistemas hipertexto *(hypertext authoring tool)*. O ambiente *Concordia* utilizava uma linguagem de marcação genérica, à semelhança da Standard *Generalized Markup Language* (SGML) e implementava ligações bidirecionais, exigindo que o autor especificasse as ligações de entrada e de saída.

Foi dada, também, grande importância para adaptar o manual do usuário às suas necessidades, sendo composto de aproximadamente 8.000 páginas contendo em torno de 10.000 nodos e 23.000 ligações hipertexto. Outro aspecto inovador do projeto foi a disponibilização de listas de endereços favoritos *(bookmarks)*.

No intuito de diminuir o esforço para a compreensão do ambiente, o projeto utilizou para construir os hiperdocumentos a metáfora dos livros, atribuindo-lhes sua arquitetura hierárquica tradicional com título, capítulos e sessões [WAL 87, DEB 2002].

II.VIII NoteCards

Em 1983, Frank Halasz, Randy Trigg e Tom Moran lideraram no *Xerox Palo Alto Center* (PARC) outro projeto denominado *NoteCards*. Como seu nome sugere, esse sistema hipermídia procurou mimetizar cartões de anotações, como os utilizados em escritórios para transmitir recados, formando um sistema que os organizasse, transformando coleções caóticas de pensamentos em interpretações integradas e ordenadas de idéias e suas interconexões. Inicialmente projetado para servir como um ambiente de pesquisa, foi posteriormente oferecido como um produto comercial. Os desenvolvedores do *NoteCards* observaram que durante o processo de desenvolvimento de um produto tecnológico, analistas formulam conceitos em suas mentes e os formalizam através de descrições textuais e modelos gráficos. Para fins didáticos, dividiram os processos de desenvolvimento em três etapas genéricas, que corresponderiam ao processo de montagem de um sistema hipermídia:

- leitura de recursos (relatórios, artigos, etc.), etapa que consiste na análise propriamente dita do problema que se pretende solucionar;

- seleção, escrita e arquivamento de trechos, fragmentos e sínteses, dos quais se originam os textos e gráficos a interligar formando a base do conhecimento para a solução do problema.;

- escrita de relatórios analíticos, referente à estruturação dos trechos selecionados e escritos em formato de hiperdocumentos.

O sistema foi desenvolvido em estações de trabalho LISP da Xerox, utilizando-se do ambiente de programação *InterLisp*, o qual a aplicação se torna parte do ambiente, permitindo aos usuários a implementação de primitivas especializadas a partir de seus dois construtores primitivos, os cartões de anotações *(notecards)* e as ligações *(links)*, formando uma arquitetura aberta *(open source)*.

No que se refere à arquitetura, os hiperdocumentos eram formalizados através dos crtões, formando um sistema hierárquico, em árvore, com ligações classificadas por tipos. Possuía mais de 50 leiautes especializados de cartões, para aplicações que necessitavam de estruturas de dados diferenciadas. Dois tipos especiais de cartões, os navegadores *(browsers)* e os fichários *(fileboxes)* permitiam ao usuário gerir árvores extensas de cartões e de ligações.

Alguns dos motivos do sucesso atribuído ao *NoteCards*, em sua época, são a elevada capacidade de processamento das estações LISP nas quais foi implementado e sua interface que exibia janelas, ligações e ícones em alta resolução [DOE 91, DEB 2002].

II.IX Intermedia

Depois do HES, do FRESS e do IGD, a *Brown University* empenhou-se em um novo projeto hipertexto, ainda mais avançado, denominado Intermedia. Liderado por Nicole Yankelovich, Bernard J. Haan, Norman K. Meyrowitz e Steven M. Druker, esse sistema mostrou-se realmente maduro, tendo sido prejudicado em suas pretensões comerciais pela plataforma de execução escolhida, sistema operacional A/UX, da família UNIX, em microcomputadores Macintosh, uma vez que, apesar de sua confiabilidade e boa integração, não havia na época uma forte base instalada.

Uma das preocupações centrais do Intermedia foi o suporte e gerenciamento de recursos multimídia, tornando-se um sistema hipermídia em nível inicial de projeto. Outro requisito de projeto que teve um tratamento especial foi o de interatividade com outras aplicações, habilitado através de um protocolo de ligação. Suas janelas possuíam rolagem vertical. As ligações eram bidirecionais e conectavam âncoras ao invés de nodos. Desse modo, ao se percorrer uma ligação até o seu nodo destino, a janela sofria um rolamento até a âncora destino se tornar visível.

Outra característica de projeto que veio a contribuir com o desenvolvimento das interfaces foi a exibição de ligações dependentes de contexto, através de uma construção que denominaram teia *(web)*. Cada ligação pertence a uma ou mais teias e se torna visível somente quando uma dessas teias estiver ativa, resultando em telas menos poluídas. Permitia, também, a criação de mapas navegáveis de ligações que proporcionavam visões gerais *(overviews)* de uma teia, de uma coleção de nodos ou de um hiperdocumento.

Inicialmente, a *Brown University* utilizou esse sistema como ferramenta para professores organizarem e apresentarem seus materiais didáticos, aos quais os alunos podiam consultar e adicionar comentários próprios, formando um ambiente hipermídia colaborativo EAD. [DOE 91, UOV 2002, DEB 2002].

II.X Guide

Em 1982, Peter J. Brown e uma equipe de pesquisadores da *University of Kent at Canterbury*, Inglaterra, deram início ao projeto *Guide*, um sistema hipertexto que tinha, como premissas, a simplicidade, a interação, a flexibilidade e a generalização no tratamento de documentos. Um hiperdocumento em Guide era definido como uma combinação de textos e botões *(buttons)*. Os botões de substituição *(replace-buttons)* possuíam a mesma fonte de caracteres do menu principal e causavam a substituição completa do texto da janela corrente pelo texto apontado pela ligação. Os botões de glossário *(glossary-buttons)* eram palavras ou frases diferenciadas do restante do texto por uma sublinha, que permitiam apresentar o texto de destino em uma nova janela, dividindo a tela. Além de serem utilizados para facilitar a criação de glossários, esses botões também foram utilizados para apresentar citações, notas e outros detalhes de um texto.

O sistema foi rapidamente desenvolvido e em 1983 já estava disponível para UNIX em estações de trabalho PERQ. Em 1984, a *Office Workstations Limited* (OWL), uma empresa de software do Reino Unido, foi criada para desenvolver de uma versão do *Guide* para Macintosh em 1984, que foi disponibilizada em 1986, tornando-se o primeiro sistema hipertexto comercial, de propósito geral disponibilizado para o grande público. No ano seguinte, foi disponibilizada uma versão para *Microsoft Windows*, tornando-se também o primeiro sistema hipertexto disponível tanto para *Macintosh* como para IBM-PC compatíveis. Após a versão 2.0, a OWL foi comprada pela *Matsushita Electric Industrial Corporation*, que perdeu o interesse no projeto. Em 1992, o desenvolvimento do Guide migrou para a empresa norte-americana *InfoAccess* [BRO 2002, UOV 2002, DEB 2002, DOE 91].

II.XI HyperCard

Em 1987, a *Apple Computer Inc*. disponibilizou, sem custos, o *HyperCard*, para seus usuários *Macintosh*, tornando-se rapidamente bastante popular. Projetado pelo engenheiro William Atkinson[27], o *HyperCard* não é um sistema hipermídia tradicional, apesar de integrar algumas de suas características, como a mudança não linear de contextos. Os únicos mecanismos de auxílio à navegação oferecidos são as funções de busca e de histórico. No entanto, o *HyperCard* é suportado pela *HyperTalk*[28], uma linguagem script direcionada à prototipação e construção de interfaces gráficas, que permite que desenvolvedores possam adicionar facilidades a um determinado sistema.

Uma influência importante para o *HyperCard* foram os trabalhos de Jakob Nielsen, especialista em interfaces, que direcionou o projeto centrando-o na usabilidade e não no desejo de proporcionar um sistema de uso geral. Algumas características de interface proporcionadas pela linguagem *HyperTalk* é a possibilidade de se mostrar e esconder campos dos cartões, conforme o contexto desejado. Outra característica importante de navegação proporcionada pela linguagem é que as ligações podem ser construídas dinamicamente.

A primitiva básica para o desenvolvimento de sistemas em *HyperCard* são os cartões, que representam nodos. Uma coleção de cartões é denominada pilha *(stack)*. Tipicamente, os cartões em *HyperCard* possuem botões para o próximo cartão e para o anterior, à semelhança dos sistemas eletrônicos de apresentação tipo *PowerPoint*, da Microsoft, com a finalidade de proporcionar uma paginação linear tradicional. Outros botões, porém, permitem a leitura não seqüencial dos cartões, dando a semântica hipermídia desejada [DEB 2002, UOV 2002, DOE 91, NIE 90].

---

[27] Atkinson criou o MacPaint, sistema gráfico de desenhos para Macintosh e foi inspirado em seus trabalhos pelas inovadoras tecnologias de interface desenvolvidas pelo PARC.

[28] A linguagem *HyperTalk* sofreu influência da linguagem *SmallTalk*, daí porque o nome *HyperTalk*.

II.XII Internet e World Wide Web (WWW)

Por fim, surge a *World Wide Web* (WWW), ou simplesmente *Web*, como se tornou conhecida popularmente, um sistema multimídia atrelado à Internet, a primeira rede de computadores a atingir escala global, interagindo, de forma crescente, com milhões de pessoas no mundo inteiro.

Sobre a Internet, essa rede começou a ser estudada ainda na década de 1960, talvez na anterior, nos laboratórios da *Advanced Research Projects Agency* (ARPA)[29], que coordenada pelo *Department of Defense* (DoD) norte-americano, desenvolvem vários projetos de ciência e tecnologia aplicáveis à área militar.

Em 1969, a ARPANET, uma rede de computadores criada pela ARPA em colaboração com universidades norte-americanas, entrou em operação interligando, gradualmente, quatro importantes centros acadêmicos[30]. Um fato importante a ser notado é que cada universidade interligada apresentava uma plataforma operacional diferente, tanto em nível de hardware como em nível de sistema operacional. Dessa forma, a ARPANET já nasceu multiplataforma [MEY 2000, CER 97, WIK 2002].

Em 1974, Vinton G. Cerf e Robert E. Kahn propuseram o *Transmission Control Program* (TCP), um protocolo para empacotamento de informações para a transmissão de dados via rede de computadores [CER 74]. Em 1982, a *Defense Communication Agency* (DCA)[31] e a *Defense Advanced Research Projects Agency* (DARPA) adicionaram o *Internet Protocol* ao IP ao TCP criando um novo protocolo denominado *Transmission Control Program over Internet Protocol* (TCP/IP), com capacidades de empacotamento e endereçamento e o implementaram no ARPANET, atribuindo ao ARPANET capacidades de interligar várias redes heterogêneas de computadores através de um protocolo de comunicação projetado com essa finalidade [ZAK 2002]. Nas palavras de Meyer:

---

[29] Em 1972, a ARPA foi renomeada para *Defense Advanced Research Projects Agency* (DARPA). Em 1993, o presidente norte-americano Bill Clinton fez a agência voltar ao se nome original. Em 1996, foi mais uma vez renomeada DARPA por um ato publicado pelo departamento de defesa.

[30] *University of California at Los Angeles* (UCLA); *University of California, Santa Barbara* (UCSB); *Stanford Research Institute* (SRI); *University of Utah*.

[31] Em 1991, a DCA foi renomeada para *Defense Information Systems Agency* (DISA)

> "Embora o acesso da ARPANET inicialmente fosse restrito às universidade ou aos centros de pesquisa com contratos com o Departamento de Defesa dos Estados Unidos, a rede cresceu rapidamente. Tornou-se uma rede internacional em 1973, com a adição de computadores em *sites* relacionados com defesa na Inglaterra e na Noruega. Em 1981, conectava 213 computadores. Em 1984, já eram 1.000 computadores e, em 1987, esse número cresceu para 10 mil. As universidades que não tinham acesso à ARPANET estavam implorando para obtê-lo... A medida que a tecnologia Internet tomava forma, a ARPANET constantemente se distanciava de suas origens militares. A ARPANET estava provando ser de grande valor para as comunidades de pesquisa universitárias; ao mesmo tempo, estava amplamente disponível para ser utilizada com segurança para a pesquisa relacionada com assuntos militares confidenciais. Por essa razão, em 1982 a rede civil (ARPANET) foi separada da parte militar (MILNET). A supervisão da ARPANET passou para a U.S. National Science Foundation (NSF), que subsidiou a ARPANET para ajudar pesquisadores universitários. a NSF financiou a construção de uma nova rede de transmissão de dados de longa distância, chamada NSFnet. O *backbone* antigo da ARPANET foi aposentado em 1990, tendo desempenhado sua função de pesquisa com sucesso espetacular. Coletivamente, o *backbone* da NSFnet e as várias redes regionais conectadas a ele tornaram-se conhecidas como Internet [MEY 2000, p. 282-283]."

Em sua origem, os serviços mais utilizados eram o correio eletrônico *(e-mail)* e a transferência eletrônica de arquivos via *File Transfer Protocol* (FTP). Rapidamente, porém, os sistemas hipertextos se tornaram extremamente populares após a inclusão de serviços *Web* na Internet. Desde 1995, a *Web* passou a ser o serviço de maior tráfego na rede [LIV 2002].

A base tecnológica da *Web* originou-se da *European Organization for Nuclear Research* (CERN), sob a coordenação de Tim Berners-Lee, como uma ferramenta de comunicação e colaboração entre a comunidade de físicos nucleares. A primeira proposta foi apresentada em 1989, sendo refinada em 1990 por Berners-Lee e Robert Cailliau. Nesse mesmo ano, um protótipo do sistema foi apresentado com o nome de Enquire Within Upon Everything, executado em arquitetura NeXT. Desse protótipo, o CERN construiu o primeiro navegador *(browser) Web*, que foi disponibilizado em 1991. Vários outros navegadores Web foram lançados após o do CERN, entre eles, o *Mosaic*, o *Mozilla* e o *Internet Explorer*.

II.XIII Mosaic

O *Mosaic* foi o primeiro navegador *Web* a atingir popularidade e auxiliou os demais que vieram a se desenvolver. Disponibilizado pelo *National Center for National Center for Supercomputing Applications* (NCSA), da *University of Illinois*, em 1993. Coordenado por Marc Andreessen e Eric Bina, foi o primeiro projeto de navegador Web a suportar recursos multimídia para imagens, sons e vídeo. Foram adicionados formulários *(forms)* ao navegador, habilitando o sistema para um vasto sortimento de aplicações. Outros aprimoramentos em relação ao navegador do CERN foram listas de endereços favoritos *(bookmarks)*, que já existiam no SDE, e o histórico de arquivos *(history files)*.

Desenvolvido originalmente para UNIX com X-Windows, no mesmo ano, versões para Macintosh e Microsoft *Windows* foram disponibilizadas, tornando o *Mosaic* o primeiro navegador Web multiplataforma. Em 1994, a Spyglass Inc. foi licenciada pelo NCSA para comercializar o Mosaic. Em 1997, o NCSA interrompeu o desenvolvimento deste navegador, uma vez que as empresas Netscape e Microsoft contrataram grandes equipes para o desenvolvimento de seus navegadores [LIV 2002].

II.XIV Mozilla

Em 1994, Jim Clark[32], Marc Andreessen e outros empreendedores formaram a empresa Netscape[33] com o objetivo de desenvolver o primeiro navegador *Web* comercial, o *Mozilla*, também conhecido comercialmente como *Navigator*, que foi lançado em dezembro de 1994.

Com muito mais recursos do que o NCSA, a Netscape rapidamente adicionou vários recursos ao navegador, integrando, inclusive, serviços de *e-mail* e de grupos de discussão *(newsgroup)*. Outro fator estratégico da empresa foi oferecer seu navegador para as maiores plataformas instaladas: *Windows*, Macintosh, UNIX e Linux.

Estrategicamente, a empresa disponibilizou o software gratuitamente para pessoas físicas e entidades filantrópicas, que foi um fator importante para o seu uso crescer a passos largos. Para empresas comerciais, a Netscape solicitava pagamento pelo uso do software. Sob a pressão da concorrência do *Internet Explorer*, disponibilizado gratuitamente pela Microsoft, a Netscape parou também de solicitar quaisquer pagamentos. Também por causa dessa concorrência, a *Netscape* perdeu forças e foi adquirida pela *America On Line* (AOL) em 1998. No mesmo ano, a AOL abriu o código do *Mozilla* e uma organização homônima foi criada para administrar o projeto [LIV 2002, MOZ 2002].

II.XV Internet Explorer

A *Microsoft* licenciou a tecnologia *Mosaic* em contrato com a *Spyglass* para produzir seu próprio navegador. Seu primeiro objetivo foi adquirir a funcionalidade do *Netscape Navigator*. Sua primeira versão foi lançada em 1995 embutida no sistema operacional *Windows* 95. Devido aos altos investimentos realizados pela *Microsoft* e sua ampla fatia de mercado de sistemas operacionais, a utilização do *Explorer* cresceu com facilidade e se transformou no navegador *Web* mais utilizado no mundo desde o ano de 1999. Disponível hoje para Mac OS 8 e 9, Mac OS-X, *Microsoft Windows*, *Solaris* e HP-UX.

---

[32] Jim Clark é um dos fundadores da *Silicon Graphic, Inc.*

[33] Originalmente, a empresa se denominou *Mosaic Communications Corporation*. Sob a ameaça de processo de parte da *University of Illinois*, que detém os direitos autorais sobre o navegador *Web Mosaic*, a empresa trocou seu nome para Netscape.

II.XVI HTML e XML

Os hiperdocumentos suportados por navegadores *Web* são escritos, atualmente, em uma linguagem denominada *Hypertext Markup Language* (HTML). Desde janeiro de 2000, a W3C está recomendando uma substituição da HTML pela *Extensible HyperText Markup Language* (XHTML):

> "XHTML 1.0 é uma reformulação da HTML 4.01 sendo uma aplicação XML 1.0, e inclui três DTD correspondentes às três definidas para HTML 4.01. A semântica dos elementos e de seus atributos são definidas na Recomendação W3C para HTML 4.01 [W3C 2001]".

Todas essas linguagens, HTML, XML e XHTML derivam de uma metalinguagem de marcação, a SGML, e seus documentos formam ligações hierárquicas unidirecionais em árvore, escritas através de um padrão para especificação da localização de objetos denominado *Uniform Resource Locator* (URL).

II.XVI.I Uniform Resource Locator (URL)

Um endereço URL é um *string* que pode iniciar com a especificação do protocolo de comunicação a utilizar. Essa informação, se fornecida, vai do início do *string* até o primeiro sinal de dois pontos. Pode ser o HTTP, o FTP, gopher, *Wide Area Information Servers* (WAIS), news, telnet, mailto, entre outros. Se não informado, o navegador geralmente assume que se trata do protocolo HTTP.

Depois dos dois pontos, o endereço é interpretado conforme o protocolo de comunicação utilizado. Geralmente, duas barras (//) após os dois pontos introduzem o nome ou número de um servidor conectado à rede *(host)* que se deseja acessar. Pode-se, também, informar a porta de acesso ao servidor, complementando-o com dois pontos seguindo o número da porta. Se não for informado o número da porta, assume-se a porta padrão do protocolo utilizado. Para o HTTP, por exemplo, a porta padrão é 80. Em FTP, pode-se também, opcionalmente, informar o usuário ou a senha de acesso após o endereço do servidor.

Em HTTP e FTP, a próxima parte identifica o nodo *(nodename)*, através um caminho que pode, ou não, especificar um determinado nome de arquivo. Se for especificado um arquivo, este tentará ser interpretado pelo navegador, que fará com que o sistema se comporte de uma determinada maneira conforme o seu tipo, características e correção. Poderá ser um arquivo HTML, uma imagem em formato Graphics Interchange Format (GIF) ou Joint Photographic Experts Group (JPEG), etc.

O tipo do arquivo a ser interpretado pode ser informado ao navegador através de um tipo *Multipurpose Internet Mail Extensions* (MIME), como por exemplo "text/html" ou "image/gif", retornado pelo cabeçalho do protocolo HTTP, e/ou por sua extensão.

Pode-se seguir, opcionalmente, um *string* de consulta precedido por um caracter "?" ou um identificador de fragmento precedido pelo caracter "#", que indicará uma posição (âncora) dentro do hiperdocumento destino, podendo provocar no navegador uma rolagem vertical até o fragmento desejado se tornar visível, à semelhança das ligações do sistema Hipermedia.

Pode-se classificar os URL, também, quanto à forma de endereçamento. Serão de endereçamento absoluto, se a URL indicar o servidor antes do caminho do diretório ou arquivo desejado, e serão de endereçamento relativo, se a URL indicar somente o caminho.

Um URL, portanto, pode formar ligações de vários tipos, conforme o protocolo, aplicação e a estratégia de endereçamento desejados. Algumas regras de produção da gramática *Backus-Naur Form* (BNF) para URL endereçando protocolo HTTP são mostradas na TABELA A1.1.

TABELA A1.1 - Algumas regras de produção BNF para URL endereçando HTTP

1ª Componente	**2ª Componente**
url	httpaddress | outrosProtocolos
httpaddress	http : / / hostport [/ path] [? search]
hostport	host [: port]
host	hostname | hostnumber
hostname	ialpha [. hostname]
hostnumber	digits . digits . digits . digits
path	void | segment [/ path]
segment	xpalphas
search	xalphas [+ search]
ialpha	alpha [xalphas]
xpalphas	xpalpha [xpalphas]
xpalpha	xalpha | +
xalpha	alpha | digit | safe | extra | escape
alpha	a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
digits	digit [digits]
digit	0 |1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
safe	$ | - | _ | @ | . | & | + | -
extra	! | * | " | ' | (|) | ,
escape	% hex hex
hex	digit | a | b | c | d | e | f | A | B | C | D | E | F
reserved	= | ; | / | # | ? | : | space
void	

Fonte: *World Wide Web Consortium* [W3C 2002]

Anexo 3 – Artigo ICECE 2000

A Database Structure for Didactic Exercises on Distance Learning Programs Adapted to ISO/IEC 9126 Standard

Roges Horacio Grandi, Rosa Maria Viccari and Paulo Fernando Blauth Menezes

***Abstract* — This work proposes the next generation for Conhecer Project [1], an Internet authorship tool for setting up didactic exercises pages. It extends its original specification to become part of a major project called Jade [2], an Internet client-server environment proposed to be an intelligent tutor on Distance Learning Programs (DLP). We propose a database structure for didactic exercises on DLP declaring formally the ISO/IEC-9126 aims: functionality, usability, reliability, efficiency, maintainability and portability. We also designed it for international use. A key feature for this project is to design its database structure as flexible as possible for easy adaptability and maintenance.**

***Index Terms* — Distance learning, Didactic Exercises, Database, Software product evaluation, ISO/IEC 9126**

I. INTRODUCTION

As the most major universities all around the world, we at Federal University of Rio Grande do Sul (UFRGS) are also developing distance learning projects. We will explain here the main features of one of them, the Conhecer Project. Its first generation was an Internet authorship tool for setting up didactic exercises pages, codified upon Java applets and applications over an Oracle 8.0.4 database. It also cares about international usage because through Internet we don't have frontiers anymore and every student is a possible participant. So we can export the system itself as well as its services. We can see Conhecer's Automatic Exercises List Generator screen in Fig. 1. It automatically generates Hypertext Markup Language (HTML) exercises pages for another project of this University, the Internet Multiagent Environment for Teaching and Learning Project (AME-A) [3].

II. JADE'S EXERCISE MANAGER MODULE

Now we propose to extend Conhecer's database design and functionality to be used inside the *Exercise Manager* module (Fig. 3) of an Internet client-server environment called Jade [2], also in development at this University, proposed to be an intelligent tutor composed of eight subject units consisting of *Lessons*, *Examples* and *Exercises*. A pilot distance learning project for Jade, called Eletrotutor [2], is a site for teaching Electricity (Ohm's Law and its applications).

Roges Horacio Grandi, UFRGS – Instituto de Informática, Av. Bento Gonçalves, 9500 Bloco IV, Caixa Postal 15064 CEP 91501-570
[roges, rosa, blauth]@inf.ufrgs.br

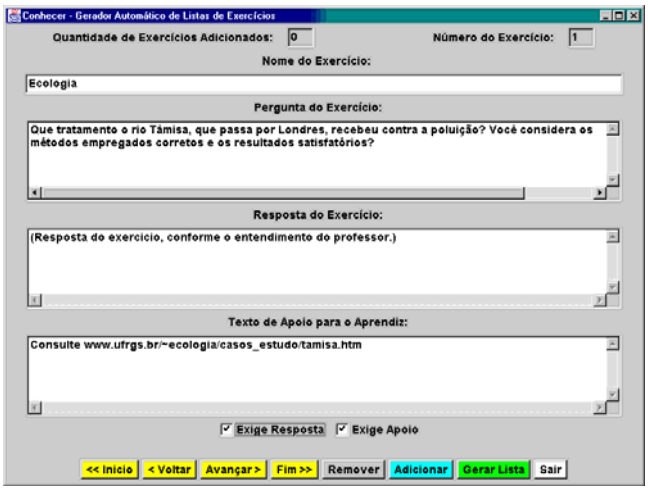


Fig. 1. Conhecer's Automatic Exercises List Generator

The first Eletrotutor's framework proposed to be included into Conhecer's database is a Java applet structure, once Eletrotutor's present version implements exercises as applets. The first step to create an exercise using Conhecer is to create an empty exercise list from an Eletrotutor framework and store it in the *List* table (Fig. 2). Then the user, probably a teacher, chooses the exercise or exercises to compose the list storing this information in the *Exerc_List* table. A web database-driven page server (such as Perl, PHP, Microsoft ASP, Sun JSP, Allaire ColdFusion, etc.) searches this information set and generate one or more web pages containing the Java applet source code. The user receives the generated applet code using the Internet browser I/O services. Once Jade is made upon Java applets the last step to be done is to place the generated applet at the correct directory. Maybe it will also be necessary to update some content on Eletrotutor because of this new information.

III. OUR EXPERIENCE ON ISO/IEC 9126

Why to be worried about quality when thinking on distance learning programs? And more, why to extend this concern when designing a database? One or more of these reasons may justify it [1]:

- Quality self-analysis, according to a given quality standard.
- Adaptation to a given quality standard. This is stronger than only self-analyze the project.
- Preparation to a future certification, in order to have the quality efforts recognized by the public.
- To improve the quality of the software and services we are offering.
- To reduce the cost and the time of the software development.

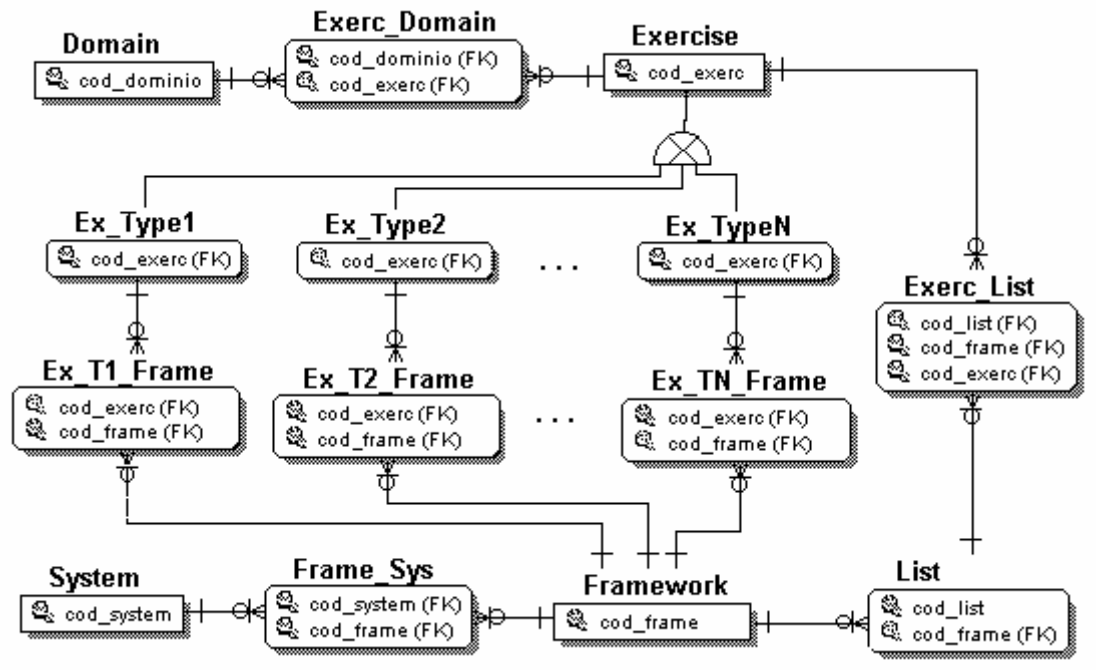


Fig. 2. Conhecer's Current Entity-Relationship (E-R) Diagram for frameworks and exercises

These are general reasons to care about quality. Specifically on distance learning, we can observe a worldwide spreading trend toward this technology. Not of simple applications. But of huge ones. In most of the cases, these big applications are built over smaller ones that by the time become integrate. If the applications were designed in terms of quality it will be easier, faster and less expensive to join them. So it is very concerning to implement quality management on them. Out of experience, the sooner we do it the sooner the positive results come. Usually, the database becomes the core of an information system. This reason alone justify to care about quality during the designing process of a database.

Some of the internationally recognized standards for software development are:

- ISO 9000 Series (especially ISO 9000-3 and ISO 9001), ISO/IEC 9126 and ISO/IEC 12207, from the International Organization for Standardization (ISO) and the International Electrotechanical Commission (IEC) [4];
- Capability Maturity Model (CMM), from CarnegieMellon University (CMU)/Software Engineer Institute (SEI) [5].

In our opinion, which standard we use as a guide for the software development is not the most important. What really matter is to follow one that really improves the process and the succeeding software. In general, we find consonance among good standards. According to Mark Paulk [6], from CMU/SEI, when a project reaches the minimum demands of a Total Quality Management (TQM) program through ISO 9001 certification it is the same as to reach the level number two among the five levels in CMM.

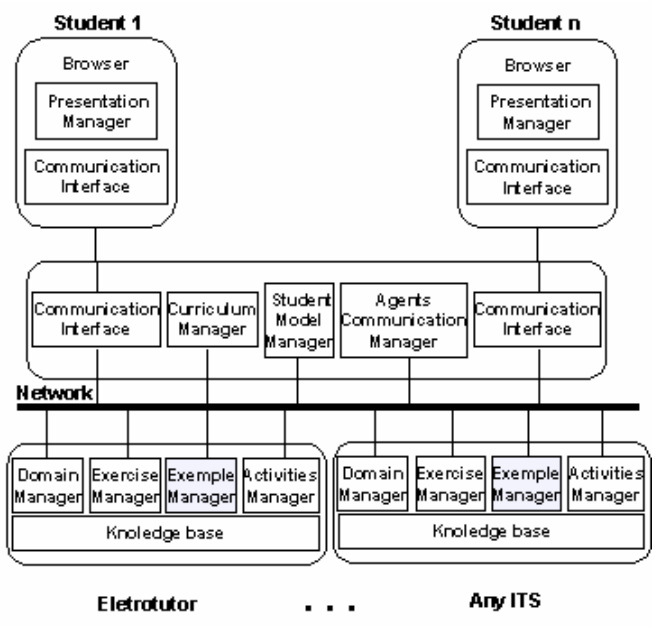


Fig. 3. Jade's Organizational Model

When designing our database for didactic exercises we followed the ISO/IEC 9126 Standard[7] recommendations. It enumerates six desirable characteristics for a software to have: functionality, usability, reliability, efficiency, maintainability and portability, and give guidelines for internal use. A draft for information includes twenty-one sub characteristics. In spite of this Standard provides guidelines to be followed, it does not forces the metrics to be used. It is a developer's task to find indicators that fits well into his own project. All the indicators must be described in a way that they can be comparable. Then, for each quality aim should be established indicators, metrics and minimum percents for its acceptance or release. The quality plan must declare how the product will attain the specified quality aims. This collection of documents is called Software Release Criteria (SRC) [8]. In a short, the structure of a SRC is:

- Characteristic
- Sub characteristic (optional)
  - Quality aim
    - Indicator
    - Aim

At least one SRC should be defined for each one of the six basic characteristics. Following we describe the SRC we established for this project.

- *Characteristic: Functionality*
- *Sub characteristic: Suitability*
  - Quality aim: To have more options for building exercises then the existent softwares.
    - Indicator: Number of options.
    - Aim: Offer at least 10% more options.
- Implementation strategy: If we wish to offer a good sort of options to build exercises we need to give the database a flexible structure (Fig. 2). Our solution allows one system to have many didactic exercise frameworks. An entity called *Exercise* store exercises, and it is specialized according to its own format. This way, any kind of computable exercise may be defined and stored. A table called *Domain* was *created* to classify the exercises according to its knowledge area. Finally, it was created the tables representing many-to-many relationships. As a result, we had a powerful entity-relationship scheme for building exercises, given the project the potential to reach the goal. To turn this aim a reality we have to validate the relation n >= 1.1 m, where n is the amount of exercise types offered by Conhecer and m is the amount of exercise types offered by the more functional existent software.

- *Characteristic: Functionality*
- *Sub characteristic: Security*
  - Quality aim: To prevent unauthorized access, whether accidental or deliberate, to programs and data.
    - Indicator: Amount of network servers' attributes to store user's information.
    - Aim: Offer at lest the same attributes for security the main network server does.
- Implementation strategy: We created an entity called *Participant* with similar attributes to those existent at network servers to control login access, such as: login id, password, last setup update operator's code. It was given the possibility to change password by period, fixed date or amount of logins done. All of these attributes are available to each participant. We still have to measure this quality aim to conclude if we had reached or not the quality aim comparing our specifications with those written for the major network server.

- *Characteristic: Reliability*
- Sub characteristic: Fault Tolerance
  - Quality aim: To have a fault tolerant architecture.
    - Indicator: Number of devices to tolerate faults.
    - Aim: To have at least 1 basic device.
- Implementation strategy: To have at least Redundant Arrays of Independent Disks (RAID) Level 1 (mirroring), which provides redundancy by duplicating all data from one drive on another drive.

- *Characteristic: Reliability*
- *Sub characteristic: Recoverability*
  - Quality aim: To be able recover the data directly affected in case of a failure.
    - Indicator: Quality of the Database Management System (DBMS).
    - Aim: To use a DBMS that implements safe rollback, secure transactions concept and log files for those transactions.
- Implementation strategy: We decided to use Oracle 8.0.4 once it implements 100% of the features aimed [9].

- *Characteristic: Reliability*
- *Sub characteristics: Maturity, Fault Tolerance and Recoverability*
  - Quality aim: To be able to recover the data directly affected in case of a failure, on line, 24 hours a day.
    - Indicator: To have a prepared certified staff to operate the system.
    - Aim: To have 100% of the staff working on the system 24 hours a day, everyone certified by the DBMS company or equivalent course.
- Implementation strategy: It means that we must invest on peopleware, training the staff and keep the system full time supervised.

- *Characteristic: Usability*
- *Sub characteristics: Understandability, Learnability and Operability.*
  - Quality aim: To make the database definitions understandable to the most of the technical community.
    - Indicator: Use of well-known methodology.
    - Aim: To use 100% well-known methodology.
- Implementation strategy: We chose the Platinum Erwin 3.5.2 as Computer Aided Software Engineering (CASE) tool to design the database. It allows automatic edition and conversion among the IDEF1X, IE and DM methodologies. For this project, we used the IE methodology because it is cognitive and well-known at our community.

- *Characteristic: Efficiency*
- *Sub characteristics: Time behavior*
  - Quality aim: To design the database in order to allow good resource and time management.
    - Indicator: Number of relationships, logical level design.
    - Aim: To reduce at much as possible the amount of relationships between entities.
- Implementation strategy: We analyzed carefully each relationship reducing them at its minimum without lose the project semantics. It is important to reduce the processing time when executing a query.

- *Characteristic: Efficiency*
- *Sub characteristics: Time and Resource behavior*
  - Quality aim: To design the database in order to allow good resource and time management.
    - Indicator: Attribute's size and type, physical level design.
    - Aim: To size its attributes as short as possible and to use types that allows the best resource and time management.
- Implementation strategy: We decided to use Oracle varchar2 attribute type for texts bigger that two characters, once it is allows good data compression reducing the input/output time. Oracle 8.0.4 is an Object-Relational Database Management System (O-RDMS), so decided to specialize complex data - sounds, images, videos, etc. - into proper attributes as much as possible. Both of those strategies should not to endanger seriously its portability.

- *Characteristic: Maintainability*
- *Sub characteristics: Analyzability, Changeability, Stability and Testability.*
  - Quality aim:
    - Indicator: Totality of documentation.
    - Aim: To document logical E-R with all of its attributes, all key-attributes (primary or not) and the most important physical information for each attributes (name, type, null allowed and usage).
- Implementation strategy: To write a technical document containing all this information.

- *Characteristic: Portability*
- *Sub characteristics: Adaptability, Conformance, Installability and Replaceability*
  - Quality aim: To project the database adaptable to different environments without applying other actions or means than those provided by its design.
    - Indicator: Number of environments 100% adaptable for the project.
    - Aim: To be adaptable to at least 10 major environments.
- Implementation strategy: To follow the American National Standards Institute (ANSI) Structured Query Language (SQL-2) specification for the logical design. This way the basic project is portable to the top ten and any other SQL Database Management System (DBMS). If we will need in future a reengineering process, the chose CASE tool for database designing - Platinum Erwin 3.5.2 - is able to convert the physical E-R from one DBMS to another.

CONCLUSIONS

Because of its flexible structure, this project has been analyzed for use also in the *Storage Manager* module of Hyper-Automata Project (Fig. 4). Initially designed for teaching Computer Science theory through the Internet, Hyper-Automata [10,11,12] is a powerful environment for assembling hyperdocuments. Based on Automata and Categorial theories, it promises high level of context reusability. Conhecer's Current Entity-Relationship (E-R) Diagram for frameworks and exercises (Fig. 2) fits well to Hyper-Automata present needs. The System (Hyper-Automata) may have one or many frameworks (hypertext, hypermidia, Adobe Portable Document Format (PDF), etc.). The contents may be stored as exercise types and classified by domain. Hyper-Automata may retrieve data directly from any table and assemble the page by its own functionality, using or not frameworks, depending on how the Automata Control System is defined.

The mentioned co-operating projects - AME-A, Eletrotutor and Hyper-Automata - are different views of distance learning programs with their own structure and beliefs [13]. These experiences exchanged among them reinforce our convictions that this work is a flexible structure to store/assemble information when we have multiple frameworks with multiple data format.

Once the exercises are stored individually and with its own structure, we can easily reuse them, creating new lists as our creativity comes. We may also think about interdisciplinary applications. If we have many systems using Conhecer as an exercise lists generator, one of them may see the exercises or contents of the other. It may be done directly, if the exercise format is visible for both systems, or by conversion.

Our concerning on quality is already given its fruits. Nowadays we are transferring the Conhecer Project from Microsoft Windows NT platform to Linux. Maybe we will change the DBMS from Oracle to PostgreSQL or MySQL. New functionality has been added. Fortunately, we conceived the project to be usable, portable and maintainable.

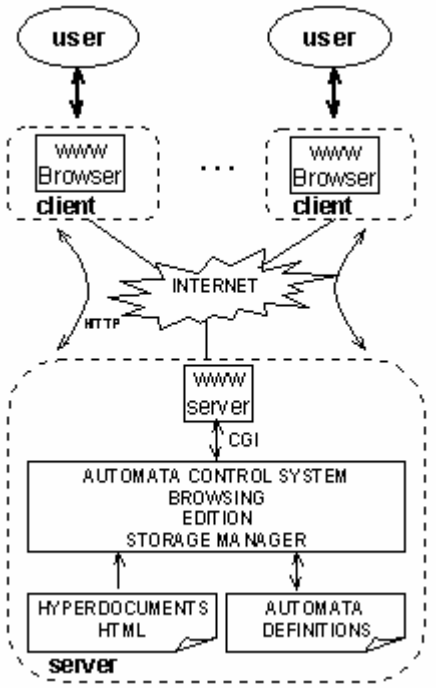


Fig. 4. Hyper-Automata's Organizational Model

ACKNOWLEDGMENTS

We say thank you to CNPq, Ricardo A. Silveira, Carmen B. Damico, Julio P. Machado and Carlos T. Q. de Morais for the help given to this project.

REFERENCES

[1] Roges H. Grandi and Rosa M. Viccari, "Conhecer – Internet Authorship Tool for Setting Up Didactic Exercises Pages": Diplomation Project. Porto Alegre: UFRGS. 77p.

[2] Ricardo A. Silveira and Rosa M. Viccari, "Eletrotutor Project: Development and Avaliation of Intelligent Learning-Teaching Environments", CLEI-PANEL'97 in *XXIII Latin American Conference on Informatics*. Valparaiso: CLEI, 1997.

[3] Carmen B. Damico, Rosa M. Viccari and Luis O. Alvares, "A Framework for Teaching and Learning Environments" in *VIII Brazilian Symposium on Informatics and Education*. 1997. São Paulo: SBC.

[4] International Organization for Standardization, "International Standards", available by http in *www.iso.ch/cate/cat.html*, Apr. 2000.

[5] CarnegieMellon University/Software Engineer Institute, "Capability Maturity Model® (SW-CMM®) for Software", available by http in *www.sei.cmu.edu/cmm/cmm.html*, Apr. 2000.

[6] Kival Chaves Weber et al. "Melhoria da Qualidade dos Processos de Software: Uso do Método SEI/CMM em Complemento à ISO 9000-3", in *IX Workshop on Software Quality, Brazilian Symposium on Software Engineering*. Recife: SBC, 1995, p. 37-41.

[7] ISO/IEC 9126 Definition of Software Quality Characteristics, apud ISO/IEC 9126 Information Technology Software Product Evaluation Quality Characteristics and Guidelines for Internal Use, available by http in *193.60.61.231/other_documentation/SoftwareQualityISO.htm*, Apr. 2000.

[8] José A. Antonioni and Newton Braga Rosa, "Software Quality (*Qualidade em Software, in Portuguese)*", São Paulo: Makron Books, 1995. 108p.

[9] Analytic Functions for Oracle8i - Technical White Paper, available by http in www.oracle.com/database/documents/ analytic_functions_

o8i_twp.pdf, Apr. 2000.

[10] Júlio P. Machado et al. Finite Automata: "A Formalism for Web Courses" ("Autômatos Finitos: Um Formalismo para Cursos na Web", in Portuguese), in *XIII Brazilian Symposium on Software Engineering*. Florianópolis: SBC, 1999, p. 213-223.

[11] Júlio P. Machado et al. "Structuring Web Course Pages as Automata: Revising Concepts", in *Recherce d'Informations Assistee par Ordinateur, VI Conference on Content-Based Multimedia Information Access*. Paris: C.I.D., C.A.S.I.S., 2000.

[12] Paulo B. Menezes, Júlio P. Machado. "Hyper-Automaton: Hypertext Framework with Categorial Operations" in *IV Brazilian Symposium on Programming Languages*. Electronic Notes on Theoretical Computer Science, in press.

[13] Lucia M. M. Giraffa, Rosa M. Viccari and John Self, "Improving Tutoring Activities Using a Multi-Agents System Architecture", in *Theoretical and Applied Computer Science Magazine (Revista de Informática Teórica e Aplicada, in Portuguese), V. 5, n. 2.* Dec. 1998. p. 87-90. Il.

Anexo 4 – Artigo ISKM/DM 2000

Utilização do XML no Sistema Hyper-Automaton

César Costa Machado[1], Júlio P. Machado[2], Roges Grandi[2], P. Blauth Menezes[2]
cmachado@atlas.ucpel.tche.br
{jhapm, roges, blauth}@inf.ufrgs.br

1 - Universidade Católica de Pelotas / CEFET-RS
Félix da Cunha, 412. CEP 96010-000.
Pelotas - RS

2 - Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Bloco IV.
Caixa Postal: 15064 CEP 91501-970.
Porto Alegre - RS

Resumo

Este trabalho dá prosseguimento a estudos anteriores buscando oferecer uma forma de estruturação e visualização do conteúdo do material utilizado em cursos disponibilizados na WWW utilizando-se construções formais conhecidas como Autômatos Finitos com Saída. O estudo baseado nas evoluções tecnológicas da Internet sugere que o sistema atual incorpore os benefícios da estruturação moderna de documentos, consoantes com as necessidades e benefícios tecnológicos, bem como, incorpore importantes recursos derivados das linguagens de marcação referentes às possibilidades de formatação do conteúdo disponibilizado visualmente. O objetivo deste trabalho é esquematizar e descrever, de acordo com uma visão prática, o que é necessário dentro das inúmeras possibilidades do XML a fim de apontar caminhos para a reestruturação e inclusão do XML e XSL no sistema Hyper-Automaton.

Palavras-chave: WWW, XML, Autômatos finitos, Gerenciamento de hiperdocumentos, Cursos na web.

1 Introdução

Este artigo dá prosseguimento ao trabalho Modelagem de Cursos na Web Utilizando Sistemas Formais [18, 1, 2] e tem como objetivo propor alternativas com respeito à modelagem de hiperdocumentos e formatação da interface com o usuário através da utilização do XML e de suas aplicações.

Cursos na Web baseados no modelo Hyper-Automaton [1,2,18] utilizam construções formais conhecidas como Autômatos Finitos Determinísticos [3]. Introduzimos o conceito de "cursos são autômatos" como uma estrutura que permite fácil implementação, criação de material hipermídia independente do autômato, ao mesmo tempo que encoraja o reuso de páginas Web em vários cursos, os quais podem ser construídos com enfoques diferentes, diminuindo a redundância na criação de páginas.

Este projeto já é realidade e constitui-se de um ambiente semi-automatizado para o suporte do ensino de Informática Teórica (http://teia.inf.ufrgs.br) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos e Teoria das Categorias, juntamente à tecnologia de Hiperdocumentos, reunindo os benefícios de ambas [1].

O objetivo geral do modelo Hyper-Automaton centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) [3] como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web. O modelo é inspirado por pesquisas clássicas na área de hiperdocumentos e recentes iniciativas na WWW, com especial enfoque no desenvolvimento de sistemas de hipertexto onde a base de hiperdocumentos é projetada de forma independente da estrutura de controle da aplicação hipermídia, e suporta algumas facilidades descritas no Modelo Dexter [5] como a composição de estruturas hierárquicas, especificação de vários conjuntos de links sobre um mesmo corpo de hiperdocumentos e objetos separados da estrutura de navegação.

Cada autômato define um curso e consiste em um conjunto de hiperdocumentos independentes, os quais podem pertencer a outros cursos. A função de transição funciona como ligação lógica entre os hiperdocumentos e a função de saída compõe as páginas. O resultado final é a estrutura básica de páginas e links de hipertexto em um site na Web. O modelo leva a um alto grau de modularização do material instrucional, apresentando as seguintes vantagens: facilidade de reuso de páginas em diversos cursos, com eliminação da redundância; independência dos hiperdocumentos da estrutura do autômato, cuja alteração não influi nas páginas e vice-versa; permite que qualquer usuário crie links de e para qualquer documento; facilidade de implementação e manutenção; interface gráfica simples e direta; elaboração de seqüências instrucionais com enfoques específicos e capaz de oferecer estudo individualizado; operações categoriais fornecem um esquema de composição de cursos que permite a construção de novos cursos sobre cursos já existentes através de procedimentos de alto nível [1,2,18].

Este trabalho está centrado nas várias possibilidades de formatação do conteúdo disponibilizado, culminando em uma adequada interface com o usuário, onde, a flexibilização da formatação do conteúdo obtido na função de saída do autômato será uma aplicação que remonte o documento nas partes desejadas, com características mais adequadas às necessidades do usuário e do curso em questão. O estudo proposto trará uma maior flexibilidade ao sistema já existente pois, no estado atual, o que se apresenta é uma única possibilidade composta de fragmentos em HTML (concatenação de páginas).

2 Visão do Sistema

O diagrama em blocos, que ilustra as partes do sistema (fig.1), apresenta em destaque a extensão ao sistema Hyper-Automaton discutida neste trabalho. A novidade está em estruturar os conteúdos (hiperdocumentos) em XML obedecendo às definições preestabelecidas para os documentos (DTD) [6,7] e utilização de templates XSL e/ou folhas de estilos em cascata (CSS) [8,9] a fim de visualização e layout do referido curso on-line. Estas alterações objetivarão principalmente a atualização tecnológica do sistema de acordo com os benefícios relacionados ao XML e descritos na seções seguintes.

A partir deste momento, este trabalho descreverá outras importantes características do XML [9] intimamente ligadas aos nossos objetivos, explicitando sua utilização conforme descrito na fig.1. Características estas que serão amplamente utilizadas na nova proposta de implementação sobre o sistema Hyper-Automaton.

Observando a fig.1 (direita), procura-se detalhar de forma gráfica o que deverá ser implementado. As características de cada bloco serão detalhadas na seqüência.

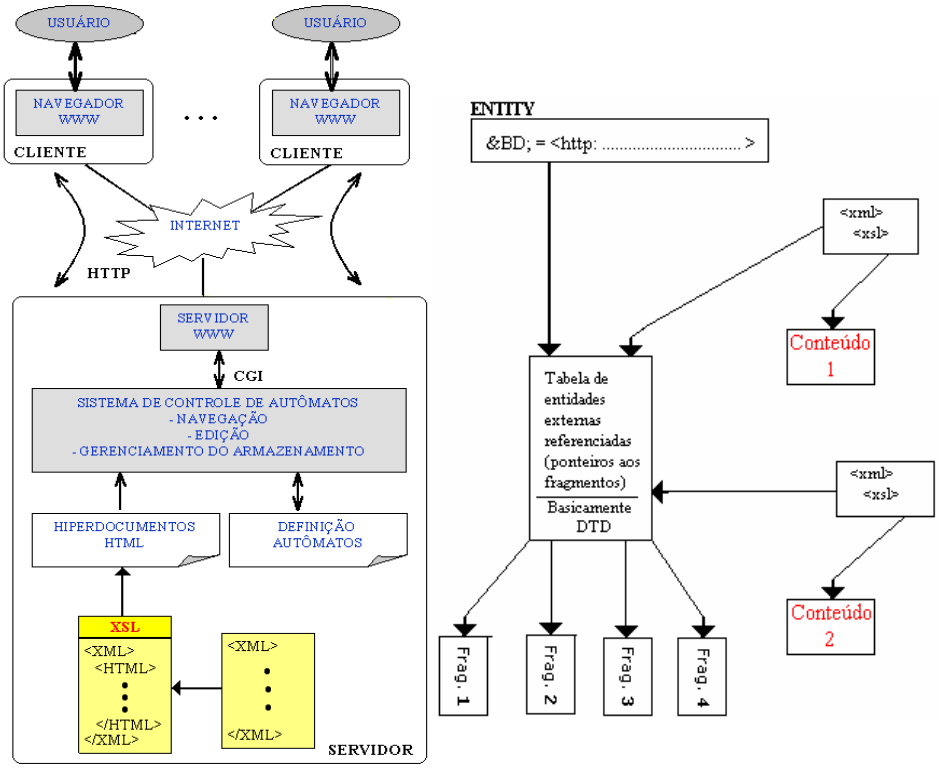


Figura 1 - Estrutura em blocos do sistema

Conforme o modelo de autômatos com saída utilizado, no nível mais baixo do sistema têm-se fragmentos de informação. Estes são considerados unidades atômicas se levarmos em consideração o conceito de alfabeto de saída. Um fragmento pode vir a ser um parágrafo de um texto, uma imagem, um vídeo, etc. A unidade de apresentação é

uma página *Web* (documento *HTML*) chamada de palavra de saída no universo dos autômatos (na terminologia de hipermídia o conceito é equivalente ao termo nodo). Uma página é, então, construída sobre esses fragmentos e quais construtores são possíveis depende da implementação do sistema. Na versão atual do Hyper-Automaton, cada página é uma seqüência linear de fragmentos estáticos. Novos construtores podem ser definidos a partir da extensão da função de saída dos autômatos com saída. Um benefício imediato da utilização de hiperdocumentos marcados com XML, é a criação de funções de saída que constróem automaticamente um novo hiperdocumento (como um índice, ou resumo de um livro) a partir de consultas XSL. Como a função de saída também é responsável pela finalização das páginas *Web* a serem visualizadas por um navegador, um uso imediato do XSL é permitir que a função determine aspectos de *layout* das páginas para os conteúdos (fontes, alinhamento, cores, etc.) utilizando-se de folhas de estilo. Além disso, através de um modelo de usuário mantido em banco de dados, é possível permitir aos usuários armazenarem opções pessoais para a visualização dos conteúdos.

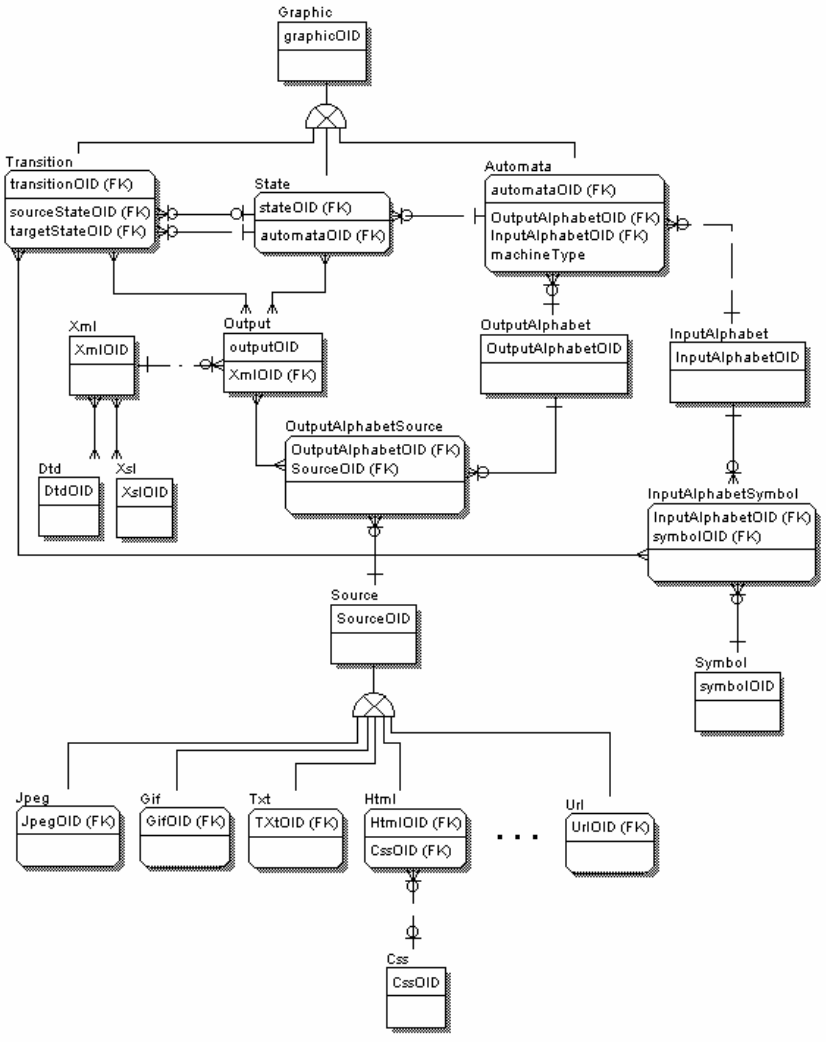


Figura 2 - Diagrama entidade-relacionamento do sistema

3 Gerenciamento do Armazenamento

Uma última abordagem a considerar sobre os formatos para os cursos realizados através do Hyper-Automaton é seu armazenamento persistente em banco de dados. Foi elaborado um diagrama entidades-relacionamentos (DER) a partir do diagrama de classes do Hyper-Automaton, que foi gerado através de mapeamento do paradigma da orientação a objetos para o paradigma relacional [15, 16, 17] . Como podemos observar (fig.2), existem várias tabelas relacionadas à formatação dos documentos apresentados via Web. Em relação aos conceitos do Hyper-Automaton, os documentos disponibilizados aos usuários são funções de saída sobre um alfabeto.

As saídas, (tabela *Output*), especializadas em documentos GIF, JPEG, etc., associam-se, portanto, a alfabetos de saída (tabela *OutputAlphabetSource*), que as vinculam aos hyper-automata, e um formato que as define, conforme os conceitos de XML (tabela *Xml*). Cada documento XML armazenado pode se relacionar, por sua vez, a declarações de tipo de documento (tabela *Dtd*) e a códigos XSL (tabela *Xsl*). Ainda relacionado ao tema formatação, podemos relacionar estilos diretamente aos documentos HTML armazenados através da tabela CSS.

4 XML (eXtensible Markup Language)

XML é um texto baseado em linguagem de marcação que esta se tornando rapidamente um padrão de troca de dados na Web. Em parte é similar à HTML, pois pode-se identificar dados utilizando marcações (identificadores inclusos).

As marcações XML identificam (explicitam) o que o dado significa de forma melhor que o HTML, o qual preocupa-se somente como os dados serão dispostos visualmente quando acessados pelos usuários. Uma marcação XML age como um campo de nome, permitindo identificar com um rótulo um determinado dado (por exemplo, "<capítulo>...</capítulo>"), tornando o documento muito mais claro ao desenvolvedor e aos programas de manipulação (denominados de parsers).

Da mesma maneira que é definido os nomes dos rótulos dos dados, é também definida a estrutura do documento. Existe uma liberdade para que seja utilizado qualquer marcação XML que faça sentido para uma certa aplicação e, desta maneira, para múltiplas aplicações, pode-se utilizar os mesmos dados XML. Além disso, a característica de uma marcação conter outras marcações fornece a habilidade ao XML de representar hierarquicamente a estrutura de dados de qualquer documento.

Um outro aspecto fundamental dos documentos XML é que estes podem ser compostos por vários módulos. É permitido assim que se possa acessar uma parte de um documento sem que seja necessário ao cliente carregá-lo totalmente. Assim sendo, documentos podem ser constituídos por peças separadas, as quais podem estar localizadas em qualquer repositório na rede, facilitando a atualização e reutilização de hiperdocumentos [6].

4.1 A DTD (Data Type Definition)

A DTD significa, em português, declaração de tipo de documento. Esta tem por função indicar as regras que o documento XML está seguindo. Como se pode observar, é o centro do esquema da fig.1 (direita).

As regras contidas na DTD podem estar inclusas em um documento XML, chamado de subgrupo interno, ou em outra unidade, referenciadas pelo URF, chamado de subgrupo externo. É oportuno ressaltar que nada impede que tenhamos em um mesmo documento XML os dois tipos de DTD's.

A importância da utilização de documentos XML em acordo com uma DTD é fundamental quando é necessário construir documentos obedecendo a padrões corretos de elaboração física e/ou lógica. Desta maneira os documentos elaborados podem ser:

- Documentos inválidos: é quando um documento XML não é validado contra uma DTD associada ao documento, não existindo elementos e atributos declarados. Assim sendo, nenhuma verificação é feita para assegurar que cada elemento contenha os subelementos que a DTD informa que deveriam conter.
- Documentos válidos: o documento XML válido deve obedecer a todas as regras. Este só não deve ter uma DTD, mas cada elemento deve estar em conformidade com as regras que a DTD contém. Este é o modelo no qual os documentos XML geralmente serão criados e atualizados. As regras de validade do XML são uma camada adicional às regras para serem bem formados. Todo documento XML válido deve ser também bem formado.

Os documentos XML podem fazer referência a recursos não-XML que estão fora do documento, como por exemplo: arquivos de imagem, clipes de vídeo, clipes de áudio, arquivos de processador de texto e applets Java [6]. Para que se possa vincular estes arquivos externos em um documento é necessário que estejam vinculados a uma entidade externa, convenientemente declarada na DTD, por exemplo:

```
<!ENTITY fig.estrela SYTEM "/Images/estrela.gif" NDATA GIF>
```

Desta maneira, entidades terão função fundamental em nosso sistema, pois é nesta parte da DTD que todos os arquivos pertencentes a uma página do curso on-line serão "apontados". Para enfatizar melhor o uso de entidades, veja a lista de possibilidades [6,7]:

- Representar caracteres que não são padrão no seu documento XML;
- Funcionar como abreviação para frases freqüentemente utilizadas;
- Manter partes de marcação que podem aparecer em mais de um documento XML;
- Manter seções ou capítulos de um amplo documento XML;
- Representar recursos que não são XML.

4.2 Folhas de Estilo (Style Sheets)

A noção de folhas de estilo é complementar a estrutura de documentos. Documentos contêm conteúdos e estrutura, e as folhas de estilo descrevem como os documentos devem ser apresentados. Esta apresentação é uma necessidade para tornar independente do dispositivo o conteúdo do documento, isto é, todas as características necessárias para o conteúdo ser apresentado de maneira correta em um dispositivo específico é informada na folha de estilo, pois isto simplifica o gerenciamento do documento, uma vez que um único documento pode esta associado a muitas folhas de estilo [11].

Por exemplo, se um documento XML utiliza elementos como: autor, nome e e-mail (fig.3), não haveria modo de dizer como este conteúdo deveria ser apresentado.

```
Markup:
        <autor>
        <nome>Irwing Cosow</nome>
        <email>cosow@w3.org</email>
        <autor>

Style sheet:
        autor {font: 12pt Times}
        nome {font-weight: bold}
        email {font-style: italic}
```

Figura 3 - CSS em XML

CSS foi desenvolvido em 1994 pela CERN com a meta de criar uma linguagem de folha de estilo para a Web que fornecesse ao autor um maior controle de estilos sobre documentos HTML. Em 1996, CSS1 (o primeiro nível de CSS) tornou-se uma recomendação da W3C [11]. Em 1997, CSS1 passou a ser suportada pelos principais navegadores, incluindo o Netscape Navigator e o Microsoft Internet Explorer, bem como várias ferramentas de autoria [10].

CSS utiliza regras declarativas para anexar estilos aos documentos. No exemplo abaixo, uma simples regra informa que todos os elementos "P", da classe "definição" são exibidos com o texto em vermelho com um fundo branco:

```
P.Definição {
                color: red;
                background: white;
              }
```

CSS1 suporta screen-based formating, incluindo fontes, cores, e layout [8,10]. Antes das folhas de estilo existirem, autores de conteúdo para a Web criavam imagens de texto para construir fontes (no caso de notação matemática) e cores convenientes. Isto resultou que a maior parte da largura de banda e usado não para texto e sim para elementos gráficos. Além do mais, folhas de estilo tem o potencial de aumentar significativamente a performance da rede de comunicação dados, como concluído por um recente estudo de como a tecnologia afeta a performance da rede [10,12].

Utilizando-se folhas de estilo ao invés de imagens, permite-se acessibilidade, característica de grande valia em sistemas de ensino a distância. Como exemplos: um sintetizador de voz pode ler um texto codificado em HTML para um usuário cego; o texto pode também ser apresentado em algum dispositivo que permita a leitura em braille; notação matemática com fontes não-padrão podem ser visualizadas em diversos navegadores.

O próximo nível da CSS, a CSS2, surgiu em 1998 [9] e, de acordo com a W3C, fortalece a acessibilidade a Web através da adição de conceitos de multimídia em folhas de estilo. Por exemplo, uma folha de estilo pode aplicar no documento, caso o dispositivo suportar, recursos sonoros.

4.3 Folhas de Estilo Avançadas

A *Extensible Stylesheet Language* (XSL), que está sendo definida pela W3C, oferece um passo adiante neste conceito, pelo fato desta estar apta a transformar a estrutura do documento. Por exemplo, uma folha de estilo XSL pode automaticamente gerar uma tabela de conteúdos (ou índice automático) através da extração dos títulos dos capítulos de um documento. O uso de XSL para transformar os dados estruturados em XML, traz grandes benefícios para a área de publicação de documentos nos mais variados formatos.

Muitas páginas Web misturam dados declarativos (como HTML, XML e CSS) com programas executáveis (como scripts e applets), pois desenvolvedores de conteúdo são motivados a usá-los para permitir efeitos especiais de apresentação (como menus especiais pop-up). O mais importante é o fato de que até agora não foi levado em consideração pelos desenvolvedores os custos e benefícios antes confiados aos scripts e applets para mostrar a informação. Os custos incluem [10] :

- Acessibilidade: conteúdo misturado com programa, torna este escondido aos sites de busca (search engines), e torna difícil se não impossível, converter este conteúdo em outro formato.
- Manteneabilidade: dados declarativos são mais fáceis de manter sua longevidade comparados aos programas.
- Independência do equipamento: muitos scripts são incompatíveis entre navegadores.

Com o desenvolvimento das folhas de estilo, espera-se que os efeitos mais comuns de apresentação estejam fazendo parte de regras declaradas no estilo. Por exemplo, a CSS2 inclui a funcionalidade de alteração na cor de um elemento ao passar do cursor do mouse, tal efeito somente era possível através da utilização de scripts.

O mais importante é que o padrão XML é independente do estilo, o que permite que para cada documento seja aplicado os mais variados formatos de visualização, podendo ser aplicado diferentes folhas de estilo para produzir documentos de saída diversos [13].

4.4 Comparação entre CSS e XSL

XSL é freqüentemente comparada a CSS como uma maneira de aplicar diferentes formatos para as marcações XML. Entretanto esta comparação é um pequeno engano. CSS lê cada elemento XML como se este fosse "escaneado" (fig.4) no documento e aplica estilos em ordem. Em outras palavras, CSS não altera a estrutura do documento XML, ela somente muda a aparência visual para cada elemento. Se for colocado um nome no topo de um documento XML, CSS irá colocar este nome no mesmo local, ao menos que seja explicitado outra posição absoluta. Além do mais, CSS irá tratar cada marcação de um dado tipo exatamente da mesma maneira.

Do ponto de vista dos desenvolvedores, pode-se alcançar uma maior flexibilidade usando uma combinação das três tecnologias: XML contém os dados, CSS (na forma interna ou externa) provê o estilo ao documento (manipulação da apresentação), enquanto o XSL é usado para modificar a estrutura do documento. Através da separação destes pedaços, o controle sobre alteração dos dados começa a ser total e os benefícios são significativos [13].

Para concluir, CSS não pode filtrar, reordenar dados, adicionar texto ou subordinar a estruturas HTML. Alguns destes problemas, podem ser resolvidos através da utilização de ambientes DHTML (Dynamic HTML), mas esta tecnologia é caracterizada por implementações proprietárias.

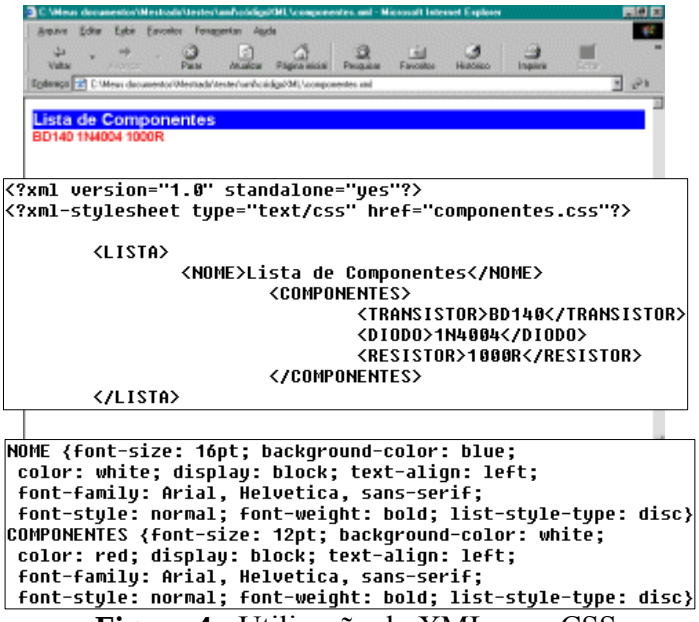


Figura 4 - Utilização do XML com CSS

XSL permite solução complementar para a formatação de XML. Diferente de CSS, o qual aplica informação de estilo para cada nodo XML que for encontrado na seqüência, XSL efetivamente repõe uma informação com a outra, independentemente da ordem que a informação encontra-se estruturada. Outras características de grande importância é a transformação que o XSL pode fazer em um documento XML tornado-o em outro documento de estrutura diferente como XML, HTML, texto, SQL [13,14]. Diferentemente de outra linguagem de transformação, XSL possui o benefício de ser escrita em XML, o que significa que o mesmo parser que pode manipular dados XML pode também manipular XSL.

XSL consiste em uma série de templates que podem ser usados para encontrar informações em um documento. Estas templates aplicam padrões ao fluxo de informações de entrada que as transformam em um outro fluxo de saída, o qual pode conter código HTML.

XSL possui um número de ferramentas para fazer comparações condicionais, ordenamento e executar operações em grupo. Assim sendo, fica justificado que a saída do processamento XSL, está de longe, desvinculada da seqüência em que as marcações aparecem no documento XML. Por exemplo, consideraremos o seguinte código XSL que manipula a criação de um "nome":

```
<xsl: template match=”nome”>
        <h1><xsl:value-of/></h1>
</xsl:template>
```

Esta simples template XSL irá chamar em qualquer momento o processador XSL a encontrar a marcação "<nome>" (neste caso somente será um). Quando houver a ocorrência, o parser XSL irá, de acordo com o texto da marcação "<xsl:value-of/>", colocá-lo entre duas marcações "<h1>" na saída da informação.

O exemplo final (fig.5, 6 e 7), um pouco mais completo, ilustra com listagens dos códigos XML/XSL, exemplificando como ficaria a saída para o documento, através da aplicação da folha de estilo em questão. Pode-se ainda combinar XML/XSL com JavaScript para criar complexas aplicações [13,14].

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="dados.xsl"?>
<ender ano="2000">
        <nome>Cesar C. Machado</nome>
        <residencial>
                <rua>Vila Real, 1160</rua>
                <cep>96083-370</cep>
                <tel>278.8704</tel>
                <estado>RS</estado>
        </residencial>
        <email>
          <email1>
            machado@atlas.ucpel.tche.br</email1>
          <email2>
            machado@etfpel.tche.br
          </email2>
        </email>
        <url>atlas.ucpel.tche.br/~cmachado</url>
</ender>
```

Figura 5 - Arquivo fonte XML

```
<xsl:template match="/">
   <HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
   <HEAD>
   <TITLE>
   <xsl:for-each select="ender">
   <xsl:value-of select="@ano"/>
   </xsl:for-each>
   - Dados pessoais atualizados
   </TITLE>
   </HEAD>
   <BODY>
      <xsl:for-each select="ender">
      <H1>
      <xsl:value-of select="@ano"/>
      Dados pessoais atualizados
      </H1>
      <H2>
      <xsl:value-of select="nome"/>
      </H2>
      <xsl:for-each select="residencial">
      <H2>
      telefone -
      <xsl:value-of select="tel"/>
      </H2>
      </xsl:for-each>
      </xsl:for-each>
      <HR></HR>
      <A HREF="http://atlas.ucpel.tche.br/~cmachado">
      Cesar C. Machado
      </A><BR />
      <A HREF="mailto:cmachado@atlas.ucpel.tche.br">
      e-mail
      </A>
  </BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

Figura 6 - Arquivo fonte XSL

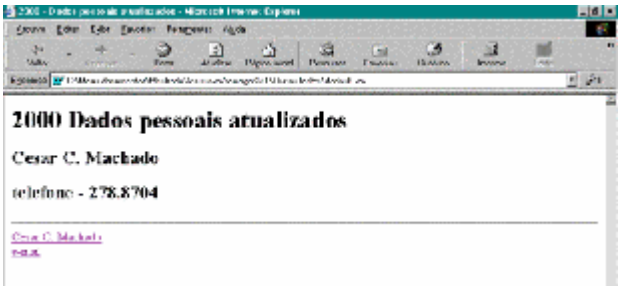


Figura 7 - Resultado da visualização em navegador

5 Conclusões

Este artigo apresentou as principais características da linguagem de marcação de documentos estruturados na Web, a eXtensible Markup Language (XML), a qual oferece significativos avanços em termos de disponibilização e visualização de documentos e a sua utilização como extensão ao sistema Hyper-Automaton.

Todos as características tratadas neste trabalho, fazem parte da implementação para a inclusão da tecnologia XML, pois como se pode observar há o envolvimento de vários módulos com diferentes implementações, isto é, estrutura do documentos em marcações apropriadas, elaboração correta da DTD e folhas de estilos XSL.

O núcleo central deste trabalho compreende uma DTD que funcionará basicamente com ponteiros a fragmentos de documentos existentes na hiperbase do curso na Web, os quais são alvo de folhas de estilo XSL, que de posse de uma DTD interna referenciada a este núcleo, terá então a capacidade de formatação visual dos aspectos necessários para a visualização dos conteúdos no navegador do usuário. O projeto neste estágio, encontra-se em fase de estudos para a construção adequada desta DTD.

As características básicas do XML e XSL identificadas neste artigo, como a estrutura clara e de fácil depuração e processamento, a capacidade de identificação de dados, o poder de estilização de conteúdos, a reusabilidade de definições e a manipulação de links, trazem benefícios adicionais ao sistema Hyper-Automaton, entre os quais: possibilidade de implementação de ferramentas de busca sobre hiperdocumentos; maior estruturação das informações disponibilizadas em hiperdocumentos e facilidades para a manipulação e gerenciamento dos dados; flexibilização da função de saída; múltiplas formatações de hiperdocumentos para um mesmo conteúdo; capacidade do usuário em alterar a formatação de saída dos hiperdocumentos de acordo com preferências pessoais; flexibilização do processamento de hiperdocumentos em operações de composição; capacidade de formatação de caracteres matemáticos.

Bibliografia

[1] Machado, Júlio P. Hyper-Automaton: Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída. Porto Alegre: PPGC da UFRGS, 2000. (Dissertação de mestrado).

[2] Menezes, P. Blauth; Machado, Júlio P. Web Courses Are Automata: a categorial framework. Anais do II Workshop de Métodos Formais, 1999, Florianópolis. Florianópolis: UFSC, Instituto de Informática da UFRGS, 1999. p.79-88.

[3] Menezes, P. Blauth. Linguagens Formais e Autômatos. Terceira Edição. Editora Sagra-Luzzatto, 2000.

[4] Machado, Júlio P.; P. Blauth Menezes. Sistemas de Gerenciamento para o Ensino a Distância. Anais do V Congresso Internacional de Educação a Distância. São Paulo: ABED, 1998. http://www.abed.org.br

[5] Halasz, F.; Schwartz, M. The Dexter HyperText Reference Model. Communications of the ACM, 1994, v.37, n.2, p.30-39.

[6] Light, Richard. Iniciando em XML. Makron Books do Brasil, 1999.

[7] Harold, Elliotte R. XML Bible. IDG Books Worldwide, 1999.

[8] Lie, Håkon W.; Bos, B. Cascading Style Sheets, Level1 W3C Recommendation. 17 dez. 1996, revisão 11 jan. 1999. http://www.w3.org/TR/REC-CSS1

[9] Bos, B; Lie, Håkon W.; Liley, C; Jacobs, I. Cascading Style Sheets, Level 2 Specification. Maio 1998. http://www.w3.org/TR/REC-CSS2

[10] Lie, Håkon W.; Saarela, Janne. Multipurpose Web Plublishing Using HTML, XML and CSS. Comunications of the ACM, 1999,v.42, n.10.

[11] World Wide Web Consortium (W3C). Setembro 1999. http://www.w3.org

[12]Nielsen, H.; Gettys, J.; Baird-Smith, A.; Prud´Hommeaux, E.; Lie, H.; Lilley, C. Network Performance Effects of HTTP/1.1, CSS1, and PNG. Proceedings of ACM SIGCOMM´97, 1997.

[13] Cagle, Kurt. Transform Your Data With XSL. 1999. http://www.xmlmag.com/upload/free/features/xml/1999/01/01win99/kc2win99/kcswin99.asp

[14] Moultis, Natanya P.; Kirk, Cheryl. XML Black Book. 1999.

[15] Ambler, Scott W. Mapping Objects to Relational Databases. Julho 2000. http://www.AmbySoft.com/mappingObjects.pdf

[16] Lermen, Alessandra de Lucena. A Mapping Framework from ODMG to Object/Relational DBMS. Porto Alegre: PPGC da UFRGS, 1999.

[17] Silberschatz, Abraham et al. Database System Concepts. 3th Ed. McGraw-Hill, 1997.

[18] Machado, Júlio P.; Morais, Carlos T.Q. de; Menezes, P. Blauth; Reis, Ricardo A.L. Structuring Web Course Pages as Automata: revising concepts. Proceedings of RIAO'2000 Recherce d'Informations Assistee par Ordinateur, 2000, Paris. Paris: Centre de Hautes Etudes Internationales d'Informatique Documentaires, Center for the Advanced Study of Information Systems, 2000, v.1, p.150-159.

Anexo 5 – Artigo ICECE 2003

Hiper-Animações - Teoria Hipermídia Aplicada em Animações

Roges Horacio Grandi[34] *e Paulo Fernando Blauth Menezes*[35]

Resumo — *As animações são seqüências de imagens individualmente concebidas, acompanhadas ou não de sons, que objetivam simular um movimento. A teoria hipermídia é baseada na liberdade oferecida ao usuário de um determinado sistema computacional de escolher a ordem da apresentação de conteúdos diversos. Tradicionalmente, animações são construídas através de uma seqüência de quadros estáticos ou, então, de uma seqüência de transformações gráficas. Se oferecermos ao usuário a possibilidade de seguir, interativamente, não somente uma seqüência predeterminada de quadros ou transformações, oferecendo seqüências alternativas, estaremos criando hiper-animações, cujas potencialidades tecnológicas na informática, educação e engenharia estão para serem exploradas. Este artigo propõe a utilização de autômatos finitos com saída para uma forma de criar e controlar hiper-animações e demonstra sua aplicabilidade em educação a distância através de um exemplo de um simulador animado de um autômato que expressa uma determinada gramática de uma linguagem formal como exercício de teoria da computação.*

Palavras-chave — animações, hipermídia, teoria dos autômatos, educação a distância, padrões de projeto, UML.

INTRODUÇÃO

Antes do advento da eletrônica, o conhecimento da humanidade era registrado, basicamente, em livros. Uma sinfonia de Mozart, uma novela de Dostoievski, um fato histórico, um princípio filosófico, uma conclusão científica, que melhor maneira de armazenar e passar para gerações seguintes um conhecimento além de um bom livro?

A eletrônica, porém, permitiu expressivas diminuições dos custos e do espaço necessário para o armazenamento do conhecimento, conforme previra Vannevar Bush na década de 1940. Além disso, surgiram novos meios de comunicação que tornaram a troca de informações muito mais rápida e eficaz.

A busca seqüencial e indexada, características dos livros tradicionais tornaram-se insuficientes para a busca e associação de informações correlatas, exigindo a criação de alternativas mais diretas de acesso e relacionamento. O próprio Vannevar Bush, ao discutir essa questão, já propôs um mecanismo de associação e busca direta de informações que denominou Memex, um aparelho que funcionaria tal e qual uma extensão da memória humana. Esse aparato tornou-se a base de inspiração para a hipertecnologia, uma área de pesquisa da informática que se preocupa em propor formas alternativas de associação e busca entre informações além da seqüencial e da indexada [1].

---

[34] Roges Horacio Grandi, Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática. Av. Bento Gonçalves, 9500 Campus do Vale - Bloco IV, 91501-970, Porto Alegre, RS, Brazil, roges@inf.ufrgs.br

[35] Paulo Fernando Blauth Menezes, Universidade Federal do Rio Grande do Sul (UFRGS), Instituto de Informática. Av. Bento Gonçalves, 9500 Campus do Vale - Bloco IV, Porto Alegre, RS, Brazil, blauth@inf.ufrgs.br

Hoje, devido ao *boom* de informações gerado, a hipertecnologia é uma necessidade crescente e está se tornando uma característica funcional padrão dos sistemas de informação.

As primeiras implementações da hipertecnologia visaram permitir leitura não seqüencial de textos e foram denominadas hipertexto por Theodor Nelson em 1965. Outras formas de mídia (imagens, sons, animações, etc.) podem ser igualmente desejados em uma leitura não seqüencial. Por não se tratarem tais mídias de um texto puro Nelson também cunhou o termo hipermídia para definir a hipertecnologia multimídia [2].

ANIMAÇÕES GRÁFICAS E SISTEMAS HIPERMÍDIA BASEADOS EM AFS

Sendo, essencialmente, a hipertecnologia uma rede de informações, buscou na teoria dos grafos e em outros fundamentos matemáticos formas e propriedades para apresentar suas características, modelagens e implementações. Os autômatos finitos com saída (AFS) baseiam sua forma nos grafos. O Laboratório de Fundamentos da Computação (LFC) da Universidade Federal do Rio Grande do Sul (UFRGS), do qual os autores participam, tem realizado pesquisas propondo os AFS como base matemática e de modelagem de sistemas hipertexto, que tem gerado implementações bem sucedidas e de qualidade e podem ser vistas no sítio (site) do laboratório [3]. Uma dessas pesquisas, denominada Animação Gráfica baseada em Autômatos Finitos (AGA), propõe a montagem e gerenciamento de animações bidimensionais de tempo real modelando-as como AFS especializados, contendo alfabetos de entrada e de saída, fitas, estados e funções, onde os atores – objetos que participam da animação - são definidos através dos autômatos e as fitas definem seus comportamentos.

"Independente da dimensão, os sistemas de animação por computador também podem ser classificados, segundo o critério de modo de produção, em sistemas de tempo real ou quadro a quadro [4,5].

Os sistemas de animação em tempo real geram a imagem final para visualização no momento de sua apresentação. Esta abordagem favorece principalmente as animações interativas, onde a imagem visualizada deve corresponder às ações instantâneas tomadas pelo usuário [4 apud 6]". Sendo uma das características básicas da hipertecnologia - a interatividade - e sendo que a abordagem de tempo real a favorece, mais uma vez podemos perceber um elevado potencial de integração entre as duas tecnologias.

Uma vez que a base matemática dos AFS, utilizados no AGA, é a teoria dos grafos, a mesma que baseia a hipertecnologia, percebemos uma interseção natural de propriedades permitindo-nos supor que a inserção de características hipermídia no AGA, do ponto de vista matemático, é um tanto facilitada e natural. O LFC já implementou alguns sistemas hipermídia baseados em AFS: o Hyper-Automaton, para apresentar cursos na Web [7] e as Provas Adaptativas, um modelo hipermídia de construção de avaliações eletrônicas [8]. Dessa maneira, já estão definidos sistemas hipermídia e, também, um modelo de animação todos baseados em AFS.

INSERINDO HIPERTECNOLOGIA EM ANIMAÇÕES

Podemos classificar os diversos tipos de mídia de forma independente em textos, sons, imagens, filmes e multimídia, sendo o último uma composição dos anteriores [9]. Outra classificação possível é, inicialmente, dividir diversas mídias em sons, que estimulam o sentido da audição, e em imagens, que estimulam o sentido da visão. As imagens, por sua vez, podem ser especializadas em gráficos e textos, conforme a presença ou não de elementos gramaticais. Filmes e animações são composições seqüenciais de mídias visuais sendo as suas formas de aquisição, respectivamente, por fotografias ou concepção individual. Podem incluir, opcionalmente, seqüências de elementos sonoros. Um exemplo de animação contendo seqüências de imagens e textos combinados são as animações legendadas. A sincronização ou não de sons com imagens permite-nos mais uma vez classificar filmes/animações em "falados" ou "mudos". Animações podem, também, ser inseridas em filmes, à exemplo das produções cinematográficas que contém efeitos especiais produzidos por animações [10]. Classificando-se desta maneira, filmes e animações podem ser considerados como produtos multimídia, uma vez que possuem a capacidade de absorver mais de um tipo de mídia. Se formos mais específicos, podemos ainda é subdividir as imagens em bi e tridimensionais. O

escopo deste trabalho são as imagens bidimensionais, sendo que o LFC pretende dar continuidade a esta pesquisa contemplando a tridimensionalidade.

Por fim, mídias podem contemplar hipertecnologia, se desejado. Qualquer mídia, seja auditiva ou sonora, podem servir de âncora hipermídia. Com a tecnologia atual, entretanto, somente imagens podem servir como elementos clicáveis servindo como opção interativa para determinar seqüências de execução através de hiperligações. Textos podem ter, por exemplo, características funcionais semelhantes à uma referência hipertexto em HTML[36]. Da mesma forma, tendo como base a WWW/Internet, sistema hipertexto mais popular atualmente, uma imagem pode ser definida como um mapa clicável.

Um diagrama de classes escrito em UML[37], produto desta forma de classificar, pode ser visto na Fig. 1. Observa-se no diagrama a aplicação do padrão de projeto orientado a objetos Composite [11].

Animações em tempo real podem permitir que várias mídias sejam transformadas simultânea e concorrentemente, disponibilizando textos, sons e imagens em uma mesma composição. Se tais características multimídia tiverem, adicionalmente, características hipermídia, cremos que a semântica operacional poderá ser fortemente enriquecida. É nesse intuito que se propõe o estudo de animações em tempo real com características hipermídia, formando hiper-animações.

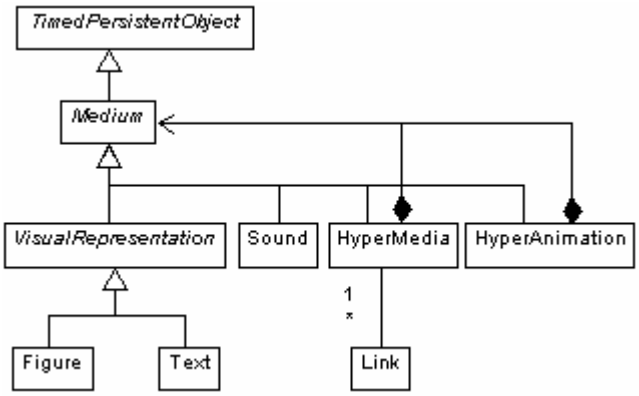


FIGURA 1 - DIAGRAMA DE CLASSES EM UML ESPECIFICANDO UMA TAXIONOMIA DE MÍDIAS CONTEMPLANDO HIPERTECNOLOGIA.

HIPER-ANIMAÇÕES BASEADAS EM AFS

Um padrão de projetos comportamental bastante útil para se modelar interações homem-máquina, principalmente quando a mudança de estados na máquina é bem controlada, é o *State*, uma vez que elimina a necessidade de formação de grandes estruturas condicionais, explicita as transições e os estados e permite que os objetos referenciados sejam reaproveitados interna e externamente, através da interação com outros sistemas [11].

Tal abordagem, conforme já comentado, é adotada pelo LFC através da implementação de AFS em vários projetos e apresenta-se igualmente útil para modelar hiper-animações. A fim de integrar a modelagem das hiper-animações (Fig. 1) com a modelagem de um AFS, podemos montar uma máquina de Mealy[38] [12] especializada. A estrutura básica de uma máquina de Mealy, para este caso, inclui os conceitos de estado, transição, alfabetos de entrada e de saída e função parcial programa. Dentre algumas especializações, as palavras de

---

[36] Hypertext Markup Language. Linguagem de marcação utilizada pela World-Wide Web na Internet.

[37] Unified Modeling Language. Linguagem de especificação padrão de fato em sistemas computacionais orientados a objetos.

[38] Autômato finito que gera uma palavra de saída para cada transição.

saída são mídias hiper-animadas que, para tecnologia Internet, podem ser referenciadas por URL[39].

Por objetivar a construção de uma aplicação persistente, pode-se aplicar também o padrão de projeto *PersistentObject* [13], a partir do qual todos os objetos persistidos possuem um identificador e, por conveniência de publicações, podem ter também um nome e um autor. O resultado é o diagrama de classes UML da Fig. 2.

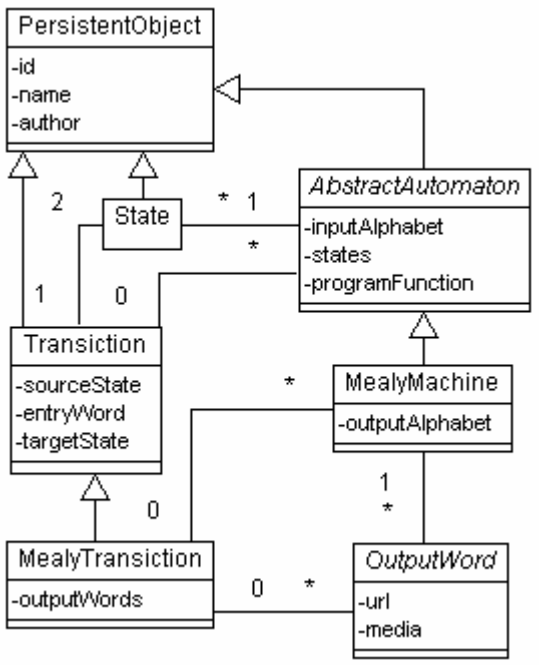


FIGURA 2 - DIAGRAMA DE CLASSES EM UML ESPECIFICANDO UMA TAXIONOMIA DE MÍDIAS CONTEMPLANDO HIPERTECNOLOGIA.

Um autômato abstrato reúne os conceitos básicos de coleção de estados, alfabeto de entrada e função parcial programa. Concretizando-se o autômato abstrato em uma máquina de Mealy, sua função programa é especializada passando a gerar palavras de saída a casa transição, no nosso caso palavras vazias, mídias simples ou compostas.

SIMULADOR ANIMADO

Hiper-animações podem ser aplicadas em diversas áreas, dentre elas a da educação a distância (EAD). Um simulador animado de um autômato que expressa uma determinada gramática de uma linguagem formal é um exemplo de sua aplicação em cursos eletrônicos sobre teoria da computação, disponibilizados via Internet. A Fig. 3 mostra uma instância desse tipo de autômato equivalente à expressão regular `(a|b)*(aa|bb)(a|b)*`.

[39] Uniform Resource Locator

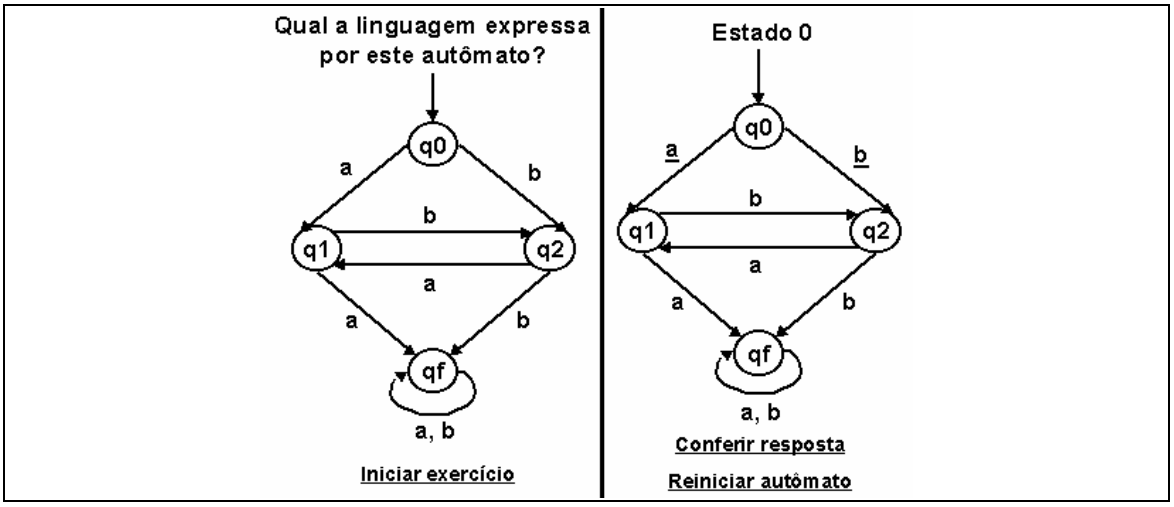


FIGURA 3 - EXEMPLO DE HIPER-ANIMAÇÃO APLICADA EM ENSINO DE TEORIA DA COMPUTAÇÃO

Neste exemplo, o aluno pode optar como controlar a simulação. Após ser introduzido ao problema proposto, quadros seguintes da animação, acionadas pela hiperligação "Iniciar exercício" permitem que o aluno, a partir do estado zero (representado por q0) interaja com o autômato experimentando-o com as hiperligações a e b, ou, opcionalmente, com as hiperligações "Conferir resposta" ou "Reiniciar autômato". Note-se que a animação não segue, necessariamente, uma seqüência específica de quadros. Esta é determinada pela vontade do aprendiz que pode experimentar vários quadros seguintes (estados 1, 2 e f), conferir diretamente a resposta, caso já a tenha abstraído, ou reiniciar o autômato (reiniciar a animação). Esta liberdade de escolha de seqüência de apresentação de mídias é desejada pela hipertecnologia e sua programação, bem como sua documentação, são facilitadas pela presença das hiperligações e, também, por sua modelagem utilizando-se os padrões de projeto supra citados.

CONCLUSÕES

Conforme demonstrado no exemplo aplicado em educação a distância, as hiper-animações possuem um forte potencial de simplificação de programação, aprimoramento de documentação e, também, de reaproveitamento tanto de análise como dos produtos gerados, se utilizados padrões de projeto corretos.

AGRADECIMENTOS

Este trabalho é parcialmente suportado pelo CNPq (projetos HoVer-CAM, Hyper Seed, GRAPHIT), CNPq/NSF (Projeto MEFIA) e FAPERGS (Projeto QaP-For). Agradecimentos extensivos a todos os colegas de pós-graduação e bolsistas envolvidos nesses projetos pelo apoio e colaboração oferecidos.

REFERÊNCIAS

- Bush, Vannevar. “As We May Think”. *Atlantic Monthly*, 1945. p.101-108
- Nelson, Theodor Holm. "A File Structure for the Complex, the Changing and the indeterminate.", *Proceedings of the ACM National Conference*, 1965.
- UFRGS Federal University of Rio Grande do Sul. Laboratório de Fundamentos da Computação. Online WWW address http://teia.inf.ufrgs.br.
- Accorsi, Fernando. “Animação Bidimensional para World Wide Web Baseada em Autômatos Finitos”. Porto Alegre: Instituto de Informática da UFRGS. 2002.
- Magalhães, Guilherme de C.. “AGA Player: Animação 2D Baseada em Autômatos para a Web”. Porto Alegre: Instituto de Informática da UFRGS. 2002. 29p.
- Thalmann, Nadia M.; Thalmann, Daniel. " Computer Animation: Theory and Practice.". Tokyo: Springer-Verlag, 1985. 239p.
- Machado, Julio H. A. P. et al. “Structuring Web Course Pages as Automata: revising concepts.” *In: RIAO’2000: Rechérche d’Informations Assistée par Ordinateur. Content-Based Multimedia Information Access: conference proceedings. Paris: C.I.D., C.A.S.I.S., 2000 V.*1 P.150-159.
- Morais, C. T. Q. de et al. “A Web Teaching System Based on Formal Methods.” *In: WCC’2000: 16th. Proceedings ICEUT, IFIP World Computer Congress. Beijing: PHEI 2000 v.1, p. 221-224.*
- Heller, Rachelle S. et al. “Using a Theoretical Multimedia Taxonomy Framework.” *In: ACM Journal of Educational Resources in Computing, Vol. 1, Nº 1, Spring 2001, Article #4, 22 pages.*
- Wikipedia. “The Free Encyclopedia.” *In www.wikipedia.com. July 2002.*
- Gamma, E. et al. “Design Patterns: Elements of Reusable Object-Oriented Software”. *Addison Wesley. 1995.*
- Menezes, Paulo F. B. “Formal Languages and Automata”. *4th Ed. Porto Alegre: Sagra Luzzato - UFRGS. 2001.*
- Ambler, Scott W. “The Design of a Robust Persistence Layer for Relational Databases”. *In www.ambysoft.com/persistenceLayer.pdf. 2000. 36p.*

Anexo 6 – Artigo WSL 2003

Astrha/E – Ambiente Java/XML que Implementa Hiper-Animações estruturadas por Máquinas de Mealy

Roges Horacio Grandi, Paulo Fernando Blauth Menezes

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91501-970 – Porto Alegre – RS – Brazil

{roges, blauth}@inf.ufrgs.br

Abstract. *Hyper-Animations are media that where hypertechnology, multimedia and computer animation techniques are built in together. Astrha is a research project where customized Mealy machines are used as hyper-animation frameworks. It is divided in three parts: Astrha/M, theoretical framework model; Astrha/L, a set of four XML dialects (mealy, style, hyper and environment) used to, respectively, to describe Astrha/M machines, text styles, hypermedia files and environments; Astrha/E is a Java environment where Astrha/L codes are interpreted in order to offer the hyper-animated desired semantics. All the implementation is under open source code and is specially recommended to build up simulators and educational media.*

Resumo. *Hiper-Animações são mídias que reúnem teoria hipermídia, multimídia, com técnicas de animação por computador. Astrha é um projeto de pesquisa no qual máquinas de Mealy especializadas são utilizadas para estruturar hiper-animações. Divide-se em três partes: Astrha/M, modelo teórico estrutural; Astrha/L, conjunto de quatro dialetos XML (mealy, style, hyper e environment) utilizados, respectivamente, para descrever máquinas Astrha/M, estilos de texto, arquivos hipermídia e ambientes; Astrha/E, ambiente Java que lê códigos Astrha/L, oferecendo a semântica desejada. Toda implementação é realizada com código aberto, sendo especialmente adequada para a construção de simuladores e programas educacionais.*

1. Introdução

Hiper-animação é um conceito que foi apresentado por (Kappe 1990). Conforme o autor, uma hiper-animação é um conceito para criação de animações interativas, combinando, essencialmente, tecnologias da Computação Gráfica e Hipermídia. Kappe denomina seqüências simples de animação "grupo de documentos" *(document cluster)*, e a visita a um grupo de documentos é definida como um roteiro *(tour)*.

Máquinas de Mealy são autômatos finitos modificados de forma a gerar uma palavra de saída a cada transição (Menezes 2001, p. 79). O projeto acadêmico Astrha propõe-se a validar a utilização de máquinas de Mealy especializadas para estruturar hiper-animações. Para tal, sua concepção foi dividida em três passos incrementais: modelo teórico estrutural (Astrha/M), linguagem (Astrha/L) e ambiente (Astrha/E).

2. Astrha/M (*Model* - Modelo Estrutural)

Hiper-animações são mídias com nível diretivo de interatividade[40]. Dessa maneira, autômatos finitos com saída são especialmente adequados para estruturar, depurar e manter códigos de diálogos humano-computador. Cada estado ou transição ou estado pode ser documentado, tratado ou depurado, se assim especificado.

Astrha/M é um modelo especializado de Máquina de Mealy que possibilita a construção de estruturas para unidades gráficas que, em um meio gráfico e dinâmico representem instâncias de hiper-animações. Trata-se de um modelo não-determinístico e reflexivo, podendo ser representada como uma 6-upla, conforme mostra a Figura 1.

$M = (\Sigma_\tau, Q, q0, \Delta, \Omega, \lambda)$ onde:

g) Σ_τ é um conjunto apontado finito de símbolos de entrada, onde é τ é o elemento distinguido.

h) Q é um conjunto finito de estados.

i) q0 é estado inicial, o qual pertence a Q.

j) Δ é um conjunto finito de símbolos de saída , denotando mídias.

k) Ω é um conjunto de palavras de saída, tal que $\Omega \subset \Delta^* \wedge \varepsilon \in \Omega$.

l) λ é uma função parcial de transição $\lambda : Q \times \Sigma_\tau \rightarrow 2^{(Q \times \Omega)}$, tal que $\forall q \in Q \; \lambda(q, \tau) = \{(q, \varepsilon)\}$

Figura 1 – Máquina de Mealy especializada para estruturar hiper-animações

Note-se que as palavras de saída são predefinidas, reduzindo seu conjunto a uma quantidade de elementos finitos, com o objetivo de definir com clareza quais são as palavras de saída desejadas para o ambiente. É necessário que os símbolos de entrada formem um conjunto apontado (Menezes 2002, p. 109) para definir o elemento distinguido τ que levará uma transição a ser reflexiva, cujo próximo estado é o estado atual e cuja palavra de saída (ε) é vazia. As transições reflexivas têm a semântica de inoperabilidade (*no operation*, ou *nop*), onde as computações realizadas durante a transição não são percebidas externamente.

3. Astrha/L (*Language*- Linguagem)

A forma escolhida para traduzir o modelo teórico estrutural de Astrha/M para Astrha/E foi descrever uma linguagem de marcação XML, composta de quatro dialetos, denominada Astrha/L.

O dialeto *mealy* é utilizado para traduzir Astrha/M através de uma linguagem de programação. Sendo as hiper-animações mídias compostas, uma palavra de saída descrita no dialeto *mealy* representa uma seqüência animada complexa na definição de (Kappe 1990). Os demais dialetos de Astrha/L, *hyper*, *style* e *environment* descrevem, respectivamente, hiperligações, estilos e ambiente. Esses três dialetos têm influência direta da tecnologia do *World Wide Web Consortium* (W3C). *Hyper* foi influenciado

[40] "Multimídia com nível diretivo de interatividade permite que o usuário inicie e responda a ações internas da aplicação, assim como adaptar aspectos do ambiente como escolha de cor, tipo de retorno, etc. (Heller 2001)."

especialmente por XLink, no que se refere aos conceitos de ligações estendidas. Texto sublinhado, mapas clicáveis e em cor diferenciada como padrão de escrita para âncoras de um hipertexto é oriundo do HTML. *Style* é uma tradução em XML de um subconjunto de atributos HTML/CSS para textos e hiperligações. O dialeto *environment* teve como base tecnológica de inspiração SMIL Animation (W3C 2001). A semântica de uma animação com suas hiperligações em SMIL Animation corresponde, parcialmente, a de uma hiper-animação em Astrha/L. Podemos perceber uma forte aproximação e influência da linguagem Astrha/L com a tecnologia e conceitos do W3C.

4. Astrha/E (*Environment* – Ambiente)

Astrha/E implementa um ambiente dinâmico, gráfico e interativo para Internet, que lê e interpreta definições de hiper-animações baseadas em máquinas de Mealy escritas em Astrha/L. Ao ser executado, Astrha/E permite que interações com o usuário através de hiperligações definam, dinamicamente, a seqüência de mídias a serem apresentadas.

Modelado em UML, foi construído em linguagem Java e utiliza software livre e gratuito em todos os seus componentes, facilitando sua distribuição e favorecendo a produção de mídias para educação a distância. A Figura 2 mostra o pacote de mídias de Astrha/E em nível conceitual. Nesse pacote, podemos perceber que uma hiper-animação é composta por hiperligações *(links)* e variados tipos de mídia: sons *(sounds)*, figuras *(figures)*, textos *(hypermedium* e *paragraph)*. Permite, inclusive, composição com outras hiper-animações.

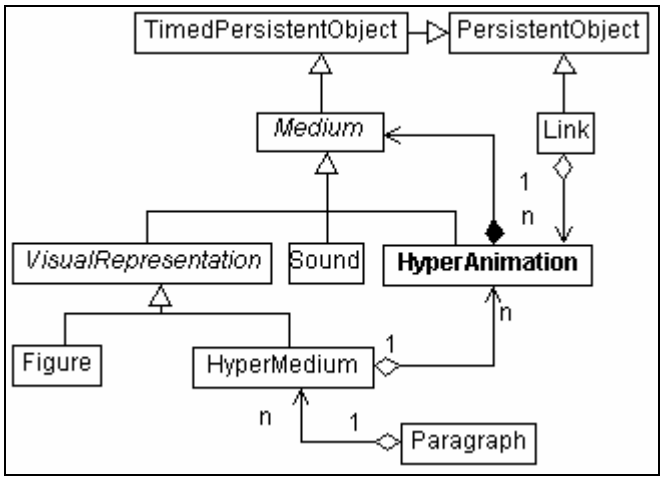


Figura 2 – Pacote de mídias de Astrha/E em nível conceitual

O diagrama de implantação da Figura 3 mostra os componentes utilizados, todos de tecnologia aberta. Java Applets[41] e JAXP[42], da Sun Microsystems; Crimson[43], da Apache e JDOM[44] da organização do mesmo nome.

[41] Programas escritos em linguagem Java que podem ser incluídos em uma página HTML.
[42] Biblioteca Java para processamento de arquivos XML.
[43] Analisador XML suportado pela biblioteca JAXP.
[44] Biblioteca Java que facilita a leitura, manipulação e escrita de arquivos XML.

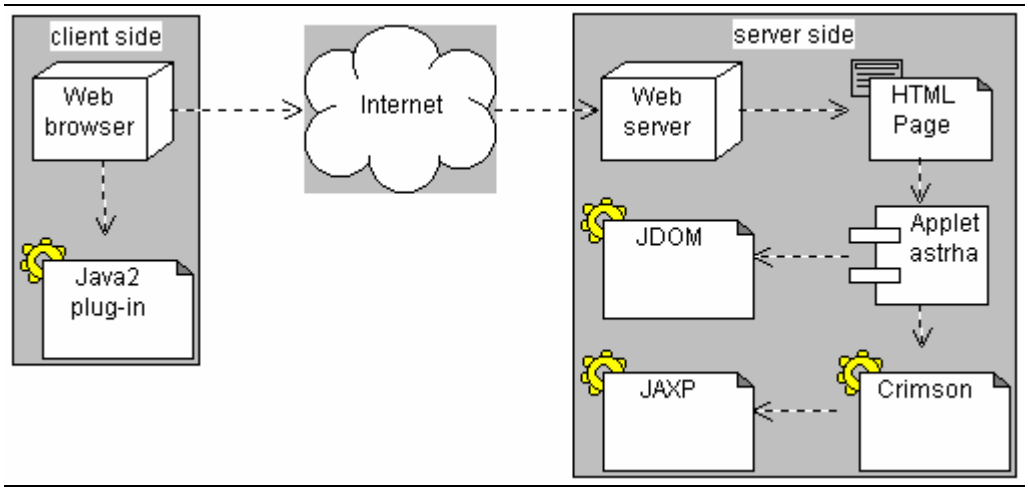


Figura 3 – Visão geral de um ambiente cliente-servidor para Astrha/E

5. Conclusões

Em experimentos realizados em laboratório, pudemos comprovar que tecnologia aberta tem a capacidade de implementar, totalmente e com vantagens, hiper-animações estruturadas por máquinas de Mealy especializadas, que é o objetivo do projeto de pesquisa Astrha, sendo a combinação das tecnologias Java/XML uma escolha bem sucedida para garantir portabilidade dentro do paradigmas da orientação a objeto para sistemas Internet. Para conhecer mais sobre o projeto Astrha: http://teia.inf.ufrgs.br.

6. Agradecimentos

Este trabalho é parcialmente suportado pelo CNPq (projetos Hyper Seed, HoVer-CAM, GRAPHIT), CNPq/NSF (Projeto MEFIA) e FAPERGS (Projeto QaP-For). Agradecimentos extensivos a todos os colegas de pesquisa.

Bibliografia

Kappe, F. M.; Maurer, H. “Animation in Hyper-G – An Outline” In Haase V. and Zinterhof P. (editors), *Proc. Future Trends in Information Technology ’90, Salzburg, Austria*, pages 235-248, Austrian Computer Society. Vienna, Munich, Sep. 1990.

Menezes, P. F. B. “Linguagens Formais e Autômatos”**.** 4.ed. Porto Alegre: Sagra Luzzato, 2001.

Menezes, P. F. B.; Hausler, Edward Hermann. Teoria das Categorias para Ciência da Computação. 1. Ed. Porto Alegre: Sagra Luzzato – UFRGS, 2002. 324 p.

Heller, R. S. et al. “Using a Theoretical Multimedia Taxonomy Framework.” ACM Journal of Education Resources in Computing, v.1, n.1, 2001.

W3C World Wide Web Consortium. “SMIL Animation - W3C Recommendation 04-September-2001”, http://www.w3.org/TR/2001/REC-smil-animation-20010904 Jan. 2003.

Anexo 7 – Artigo SBCM 2003

Utilização do Ambiente Astrha para Implementar um Dicionário de Acordes Baseado em Autômatos Finitos

Roges H. Grandi, Leandro L. Costalonga, Paulo F. B. Menezes, Rosa M. Viccari

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{roges, llcostalonga, blauth, rosa}@inf.ufrgs.br

Abstract. *This article proposes, essentially, to validate Astrha, an Internet multimedia interactive dynamic graphic environment based on hyper-animations and automata theory as media for implementing an guitar accords dictionary. Initially describe through an non deterministic finite automata, the dictionary is transformed to a equivalent deterministic automata. Then outputs are associated to transitions. Having the dictionary model been adapted to Astrha theoretical model, the next step was translate it to Astrha/L XML language. Finally, the Astrha/L dictionary code was interpreted by an Astrha/E graphic environment implemented in Java language as an applet.*

Resumo. *Este artigo propõe-se, essencialmente, a validar o Astrha, um ambiente multimídia, gráfico, dinâmico, interativo, para Internet baseado em hiper-animações e na teoria dos autômatos, através de uma implementação de um dicionário de acordes. Descrito, inicialmente, através de um autômato finito não-determinístico, o dicionário é transformado em um autômato determinístico equivalente. Em seguida, saídas multimídia são associadas às transições. Tendo o modelo teórico do dicionário sido adaptado ao modelo teórico de Astrha, o passo seguinte foi traduzi-lo para Astrha/L, linguagem XML que, por fim, é interpretada pelo ambiente gráfico Astrha/E, implementado em linguagem Java através de uma applet.*

1. Introdução

O estudo apresentado neste artigo, interdisciplinar, envolve teoria dos autômatos, sistemas multimídia, computação gráfica e linguagens de programação no que se refere à ciência da computação. Na área da música, aborda-se a teoria musical empregada na formação de acordes: intervalos, escalas, harmonização e notação musical. Trata-se de um estudo de caso que valida o Astrha (acrônimo de *Automata Structured Hyper-Animation*) - um ambiente multimídia, gráfico, dinâmico, interativo, para Internet baseado em hiper-animações e na teoria dos autômatos, como tecnologia para se implementar um dicionário de acordes[45] definido na forma de um autômato finito.

[45] Dicionário de acordes são representações populares e detalhadas de como um acorde é gerado e executado no instrumento em questão.

Acordes são uma combinação de sons simultâneos ou sucessivos quando arpeados. Cifras são símbolos criados para representar acordes, sendo compostos de letras, números e sinais (Chediak 1984). Popularmente, os acordes podem ser notados por cifras, que indicam a nota fundamental, intervalos e inversão.

O cálculo de um acorde consiste, basicamente, em transformar uma cifra em um conjunto de notas. É uma aplicação comum encontrada em muitos softwares de edição de partitura e de performances musicais. Devido à generalidade de sua aplicação em sistemas legados, o cálculo de acorde foi escolhido para ser caso de uso deste trabalho.

As cifras ainda não estão mundialmente padronizadas (Chediak 1984). Entretanto, são largamente utilizadas nas notações musicais mais simples e populares. Visando um público crescente que demanda simplicidade, é usual haver uma separação dos diversos elementos musicais (West, Howel & Cross 1991). Em música popular, é comum a utilização de músicas cifradas (Sher 1991), que se concentram no componente harmônico da música, supondo o conhecimento da melodia e do ritmo por parte do músico.

Recursos multimídia, no entanto, podem ser usados para obter-se uma notação mais rica sem com isso perder muito em precisão (Roads 1996). Neste aspecto, os recursos multimídia do ambiente Astrha mostram-se especialmente úteis.

Para construção do autômato, uma gramática foi definida com os símbolos mais comuns encontrados nas notações de cifragem mais populares no Brasil (brasileira e americana), sendo que os símbolos escolhidos para compor linguagem não podem contradizer seu sentido em nenhuma situação em particular.

2. O Ambiente Astrha

Astrha é uma proposta de unificação de modelos formais baseados na teoria dos autômatos (Machado 2000, Accorsi 2002) que, através de seus recursos multimídia e gráficos, gera hiper-animações[46] na Internet, imprimindo características de dinamismo e interatividade. O ambiente é composto de três partes: **Astrha/M** (modelo teórico): define uma Máquina de Mealy[47] reflexiva e não-determinística especializada a fim de possibilitar a construção de estruturas de hiper-animações. **Astrha/L** (linguagem): definida a partir da metalinguagem *Extensible Markup Language* (XML), contém quatro dialetos: *style*, para definir estilos de apresentação; *hyper*, linguagem de programação hipermídia com ligações que referenciam múltiplos documentos; *mealy*, que define máquinas de Mealy reflexivas e não-determinísticas especializadas para criar hiper-animações interativas e *environment*, que utiliza os três outros dialetos para a construção de ambientes gráficos interativos. **Astrha/E** (ambiente): protótipo funcional da solução, utiliza programas escritos em Astrha/L como entrada de dados, foi descrito em *Unified Modeling Language* (UML) e traduzido para a linguagem Java, que utiliza applets para exposição dos ambientes criados.

---

[46] Uma hiper-animação pode ser definida como um conceito para a criação de animações interativas de tempo real. Trata-se, essencialmente, da combinação das tecnologias da Computação Gráfica e Hipermídia, a qual denominamos *Hiper-Animação* [Kappe 1991, p. 101]

[47] Autômato finito modificado de forma a gerar uma palavra de saída a cada transição [Menezes 2001].

3. O Autômato do Dicionário de Acordes

O autômato do dicionário de acordes foi formalizado, inicialmente, sendo não-determinístico e possuindo 69 estados, com a capacidade de validar várias notações, como por exemplo a cifra C7(add11). Neste estudo de caso apresentaremos, porém, apenas sete estados desse autômato, uma vez que os mesmos são suficientes para validar a sua implementação no ambiente Astrha.

Dessa maneira, é representado pela 5-upla D1 = (Σ, Q, δ, q0, F) onde:

a) Σ é o alfabeto de símbolos de entrada, Σ = nota ∪ alt ∪ sus ∪ var

   nota = {A, B, C, D, E, F, G}, alt = {#, b}, var = {º, m, 5}, susN = {sus2, sus4, sus9, sus11}

b) Q é conjunto de estados possíveis, Q = {S1, S2, S3, S4, S5, S6, S7}

c) δ é a função parcial de transição $\delta : Q \times \Sigma \rightarrow Q$

d) q0 é o estado inicial, q0 = S1

e) F é o conjunto de estados finais, F = {S2, S3, S4, S5, S6, S7}

Na Figura 1, observa-se o autômato inicial capaz de validar acordes maiores, menores, diminutos, força (duplicação da quinta justa) e suspensos. Os estados finais S2, S3, S4 e S5, no autômato original, levam a outros estados que validação cifra mais complexas.

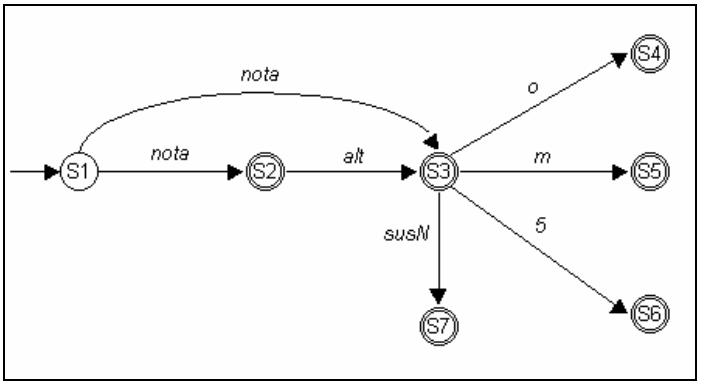


Figura 1. Autômato inicial do dicionário de acordes com sete estados

O primeiro passo no sentido de traduzir o autômato para o ambiente Astrha, como um reconhecedor de cifras, é transformá-lo de não-determinístico para determinístico, uma vez que, em Astrha, o não-determinismo possui a semântica de pseudo-aleatoriedade. Tal transformação é relativamente simples e direta, uma vez que a classe dos Autômatos Finitos Determinísticos (AFD) é equivalente à classe dos Autômatos Finitos Não-Determinísticos (AFN). Além disso, existe um algoritmo de baixa complexidade computacional que realiza a construção de um AFD a partir de um AFN (Menezes 2001, p. 49-51). Podemos notar não determinismo no autômato D1 no estado S1, onde uma nota tem a capacidade de realizar uma transição para S2 ou S3. Aplicando

o algoritmo citado por Menezes, o AFN desaparece a transição de S1 para S3, sendo acrescentadas transições diretas de S2 para S4, S5, S6 e S7, conforme mostra o autômato N1, representado pela Figura 2.

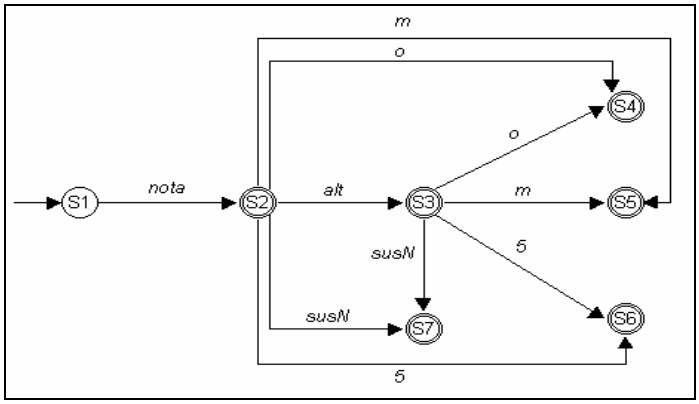


Figura 2. Autômato determinístico do dicionário de acordes com sete estados

4. Tradução do Dicionário para Astrha/L

O dialeto *Mealy*, da linguagem Astrha/L, é utilizado com a finalidade de se descrever, em XML, máquinas de Mealy com semântica multimídia, especializadas para representar hiper-animações. O poder das hiper-animações permite interações com o usuário, apresentação de animações, sons e textos, todos em uma única mídia, possuindo semântica suficiente para validar cifras, demonstrando seus respectivos acordes, executar arquivos de som para ouvi-los ou, até mesmo, apresentar uma seqüência de quadros animados que demonstram uma execução em um instrumento.

Uma vez obtido um AFD bem formalizado, e tendo a consciência da semântica desejada para cada transição do autômato, para se escrever um código no dialeto Mealy é simples. Cada símbolo de entrada válido é listado em uma marca *(tag) input* do alfabeto de símbolos de entrada *(input alphabet)*. Os autômatos D1 e N1 possuem, ambos, 16 símbolos de entrada válidos. Símbolos de saída *(output symbols)* são listados em um conjunto de marcas de fontes de mídia *(source)*. Um conjunto finito de zero ou mais símbolos de saída formam, em série, palavras de saída *(output words)*. O conjunto de estados é listado em marcas *state*. Por fim, as transições são listadas em marcas *transition*, que associam um estado origem *(from)*, um símbolo de entrada *(input)*, um estado destino *(to)*, e uma palavra de saída *(output)*. Observa-se que Astrha/L especifica o estado inicial da Máquina de Mealy (neste estudo de caso, S1) no dialeto *Environment*, a fim de proporcionar uma maior flexibilidade para as aplicações genéricas a que se destina. Um trecho da tradução do autômato N1 no dialeto *Mealy* da linguagem Astrha/L é mostrado na Figura 3.

Input id é um identificador único para uma entrada.

```
<!DOCTYPE mealy SYSTEM "../../../../language/mealy.dtd">
<mealy name="Valida cifras">
<inputAlphabet>
 <input id="1" text="A"/>
 ...
 <input id="4" text ="D"/>
 ...
</inputAlphabet>
<outputSymbols>
 <source  id="1" model="cifras" type="hypermedia" file="A.xml"/>
 ...
 <source  id="4" model="cifras" type="hypermedia" file="D.xml"/>
 ...
 <source  id="23" model="cifras" type="graphic" file="D.gif" x=200/>
 ...
 <source  id="56" model="cifras" type="sound" file="D.mid"/>
 ...
</outputSymbols>
<outputWords>
 <output id="1" sourceIds="1,21,54" meaning="A"/>
 ...
 <output id="4" sourceIds="4,23,56" meaning="D"/>
 ...
 <output id="7" sourceIds="7" meaning="G"/>
 ...
</outputWords>
<statesSet>
 <state id="1" name="inicial"/>
 ...
 <state id="7" name="susN"/>
</statesSet>
<transitionFunction>
 <transition id="1" from="1" input="1" to="2" output="1"/>
 ...
 <transition id="3" from="1" input="4" to="2" output="4"/>
 ...
</transitionFunction>
</mealy>
```

Figura 3. Trecho de código no dialeto Mealy de Astrha/L representado o autômato N1 com saídas associadas às transições.

5. Dicionário de Acordes implementado em Astrha/E

A última etapa para se expressar o autômato do dicionário de acordes no ambiente Astrha, é criar uma estrutura de diretórios em um servidor Internet *(Web server)*, criando um ambiente *(environment)* específico para o dicionário, através de uma estrutura de diretórios cuja raiz é astrha[48]. Nessa estrutura de dicionários, são colocados uma página HTML que irá chamar a applet de um ambiente Astrha (Astrha/E). Ao ser iniciada, a applet irá carregar o código escrito em Astrha/L que proporcionará a semântica do dicionário de acordes desejado.

[48] Para se executar Astrha, é necessário Java *plug-in* versão 1.2 ou superior e bibliotecas JDOM, JAXP e Crimson compatíveis.

Um exemplo bastante simples do ambiente Astrha/E reconhecendo uma cifra “D” como um acorde de ré maior é mostrado na Figura 4 O usuário fornece, como entrada, a cifra “D”, correspondente a um acorde de dó maior. O ambiente Astrha/E (uma applet) interpreta o código Astrha/L escrito conforme a Figura 3. Estando no estado inicial S1, recebendo como entrada o símbolo “D”, aciona a transição identificada pelo número 4 `(transition id="4")` gerando a palavra de saída número 4 (`output id="4"`), formada pelos símbolos de saída números 4, 23 e 56, correspondendo, respectivamente, aos arquivos D.xml, D.gif e D.mid. Essas três mídias são disponibilizadas simultaneamente, uma vez que o ambiente Astrha/E possui mecanismos de sincronização de mídias através de um temporizador de carga de arquivos. Ao final da carga dessas três mídias, a applet dinamicamente gera o conteúdo mostrado na Figura 4 e executa o arquivo de áudio D.mid, expressando uma semântica bastante completa, reunindo texto, som e imagem.

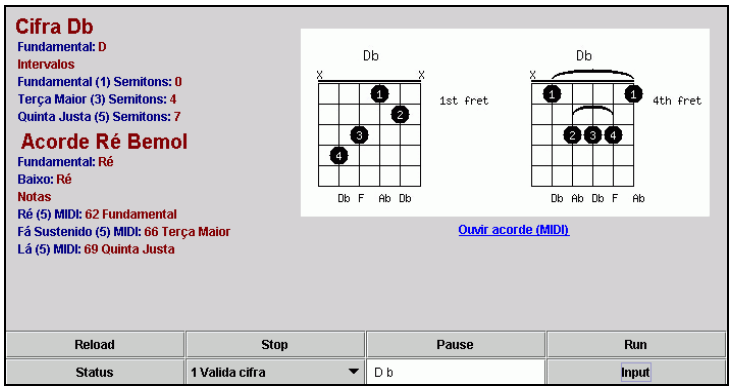


Figura 4. Exemplo de execução do dicionário de acordes em um ambiente Astrha

6. Conclusões

Pudemos perceber, através da implementação realizada, que Astrha é propício para dar a semântica necessária a um dicionário de acordes. Sua manutenção é fácil, uma vez que nenhum ajuste em programa Java é necessário para se aumentar, diminuir ou corrigir o autômato, bastando, para essas atividades, simplesmente editar o código no dialeto Mealy da linguagem Astrha/L. Dessa maneira, um músico que conheça a linguagem XML e possua orientações sobre a estrutura da linguagem Astrha/L, poderá editar os arquivos XML, criando seu próprio dicionário de acordes, personalizado.

Outras vantagens do ambiente Astrha são a sua portabilidade, rodando em qualquer navegador Internet que possua Java *plug-in* versão 1.2 ou superior instalado. Além disso, o projeto de Astrha foi realizado com código aberto e gratuito, permitindo sua livre distribuição. Para conhecer mais sobre o projeto Astrha: http://teia.inf.ufrgs.br.

7. Bibliografia

Menezes, Paulo B. “Linguagens Formais e Autômatos”. 4.ed. Porto Alegre: Sagra Luzzato, 2001.

Accorsi, F. et al. “Animação Bidimentional para World Wide Web Baseada em Autômatos Finitos**.”** Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2002. 113p.

Machado, J. H. A. P.; Menezes, P. F. B. “Hyper-Automaton: Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída”. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre, 2000. 149p.

Kappe, F. M. Aspects of a Modern Multi-Media Information System. PhD. Thesis – Institute for Foundations of Information Processing and Computer Supported New Media (IICM), Graz University of Technology. Graz, 1991. 155p.

Chediak, A. “Dicionário de Acordes Cifrados. Harmonia aplicada música popular." Ed. Irmãos Vitale. São Paulo, 1984.

Roads, C. “The Computer Music Tutorial.” Massachusetts: MIT Press, 1996.

Sher, C. “The New Real Book”. Vols. 1 and 2. Berkeley: Sher Music, 1991.

West, R., Howell, P.; Cross, I. “Musical Structure and Knowledge Representation.” In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (p. 1-30). London: Academic Press, 1991

Anexo 8 – Modelagem UML

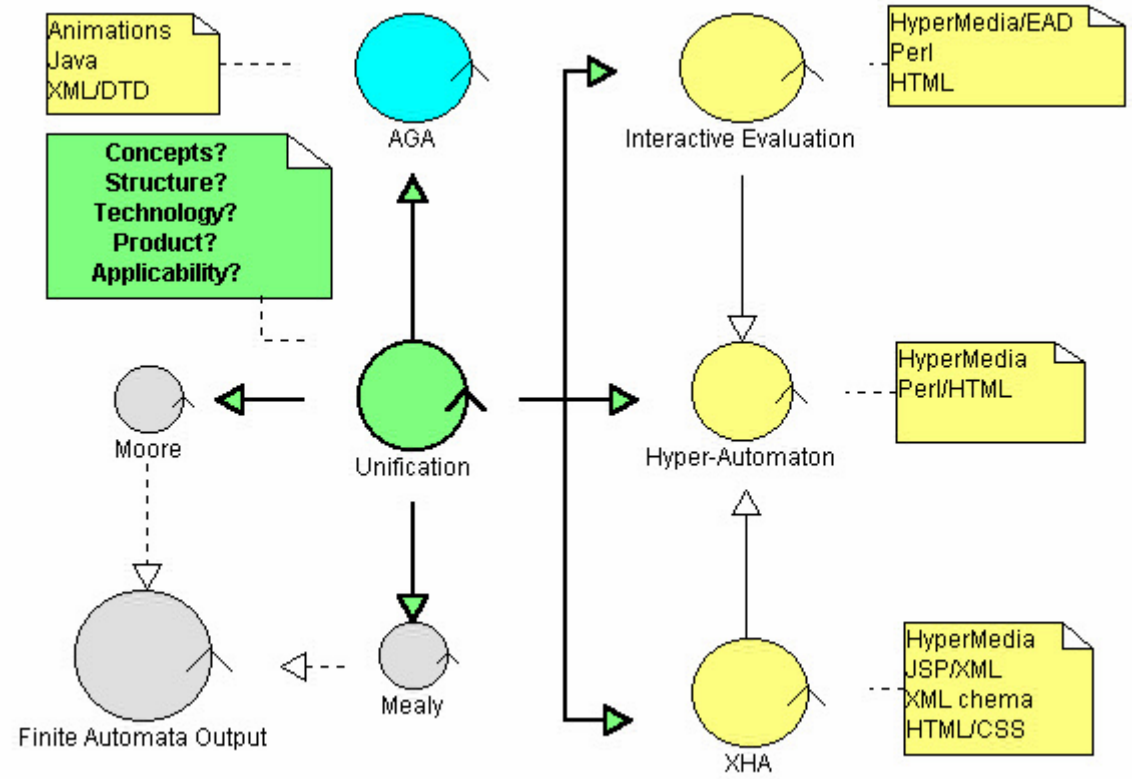
VIII.I Resumo

Este anexo apresenta a modelagem UML do Astrha, que é uma expressão do modelo teórico, dos requisitos funcionais e dos casos de uso analisados. Esta modelagem foi sincronizada através de engenharia reversa através da ferramenta Pragsoft UMLStudio 6.2.

Astrha Models

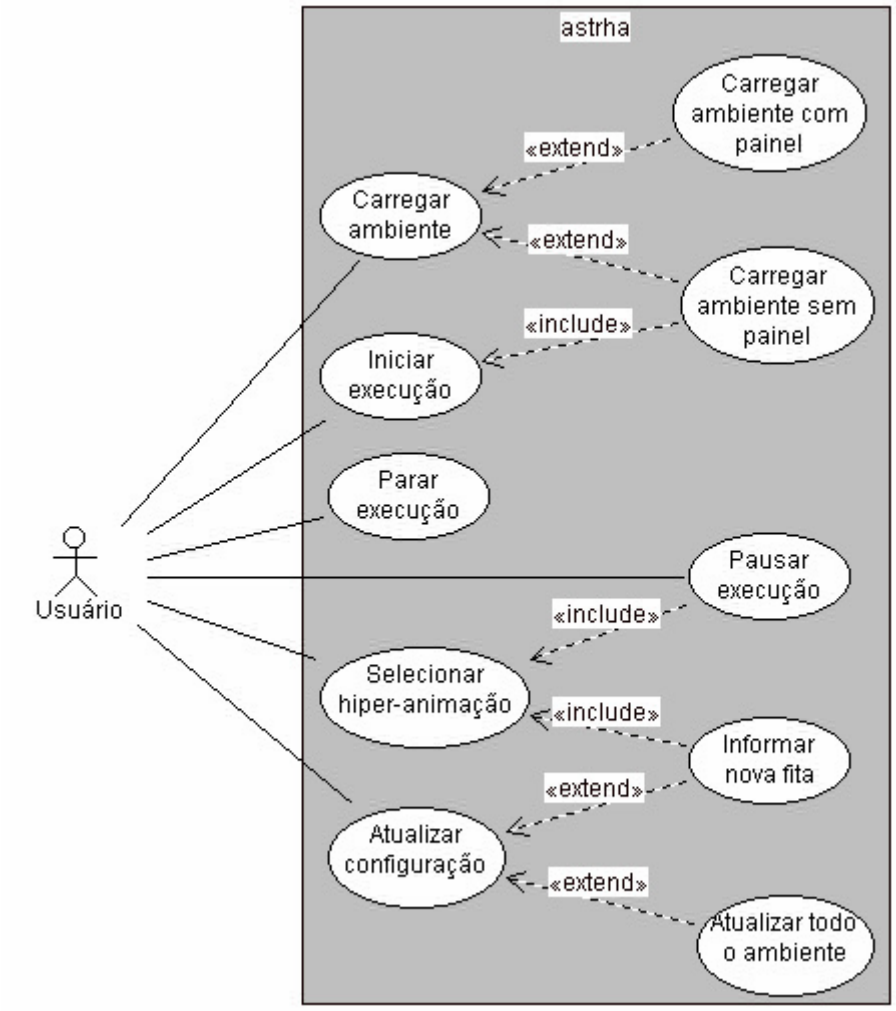
preliminary (public class model)

Contains: **Finite Automata Output, Mealy, Moore, AGA, XHA, Hyper-Automaton, Interactive Evaluation, Unification, Concepts? Structure? Technology? Product? Applicability? , HyperMedia Perl/HTML, HyperMedia/EAD Perl HTML, HyperMedia JSP/XML XMLSchema HTML/CSS, Animations Java XML/DTD.**



use case (public use case model)

Contains: Carregar ambiente, Carregar ambiente com painel, Carregar ambiente sem painel, Iniciar execução, Parar execução, Pausar execução, Selecionar hiper-animação, Informar nova fita, Atualizar configuração, Atualizar todo o ambiente, astrha, Usuário.



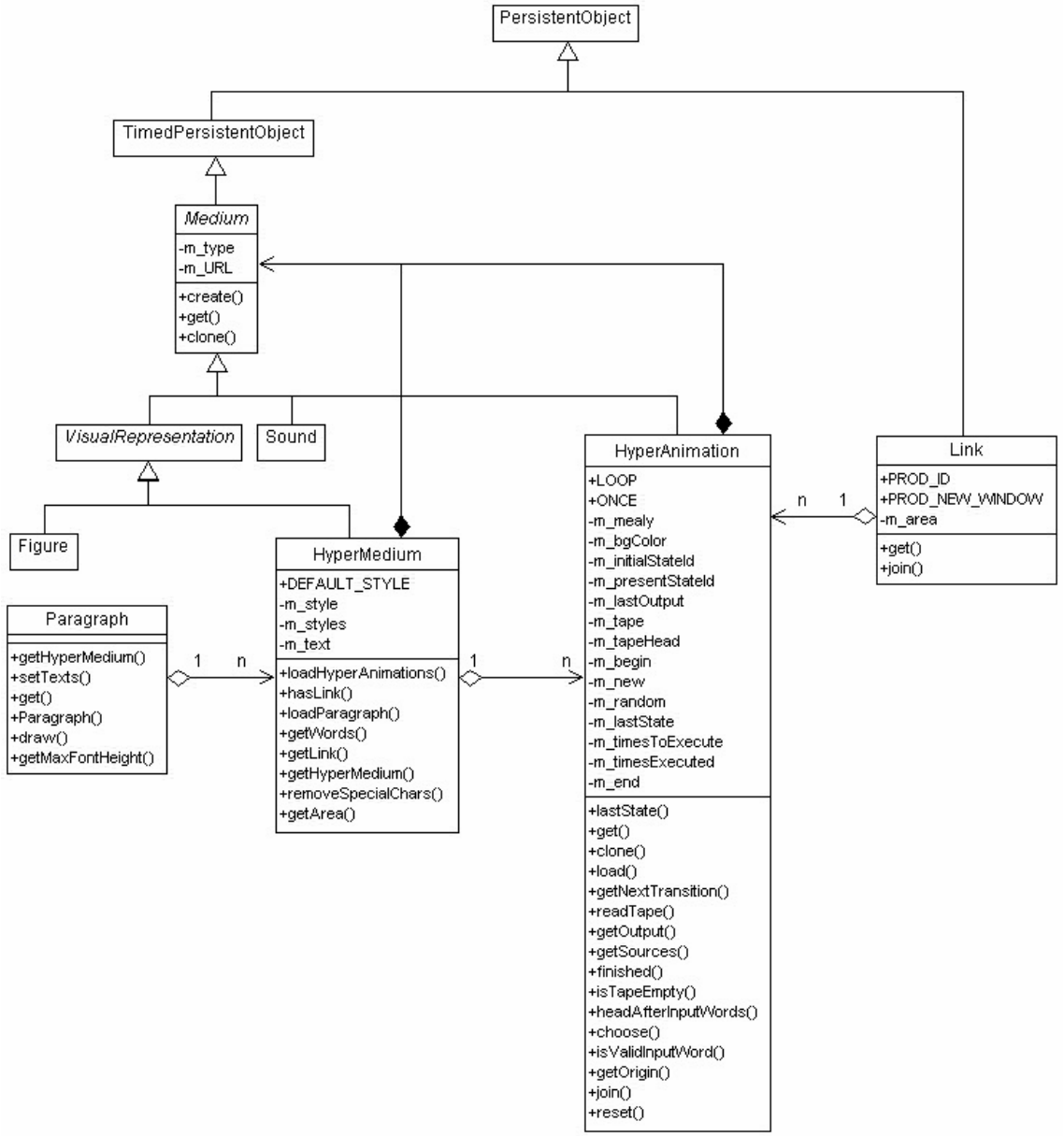
br.ufrgs.inf.astrha (public class model)

Contains: media, language, environment, persistence, mealy, individual, production, JApplet.

media (public class model)

Parent: **media**

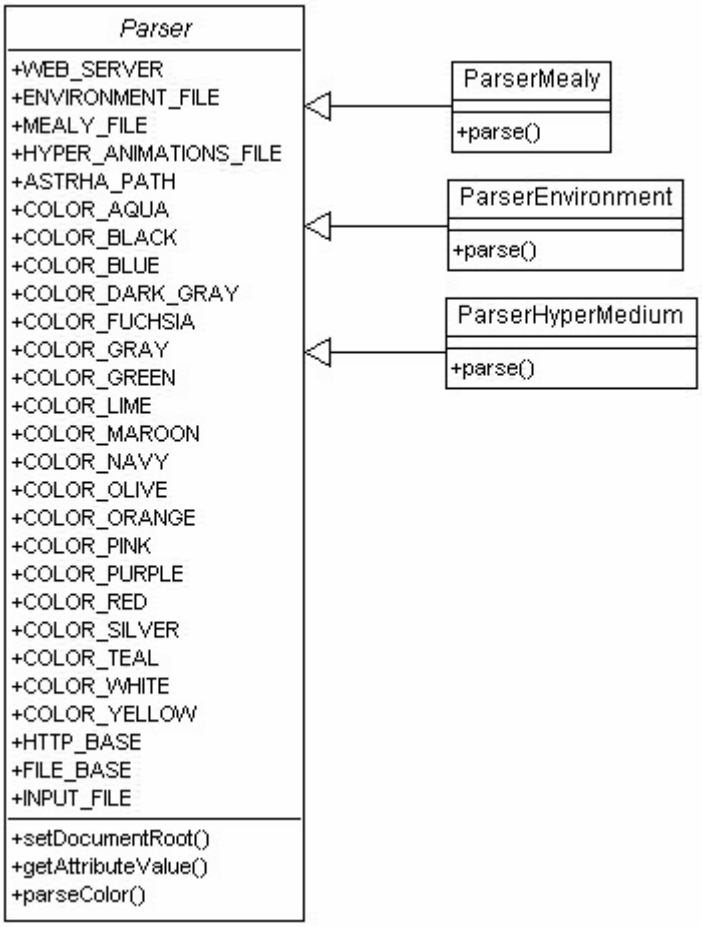
Contains: **Medium, Figure, VisualRepresentation, PersistentObject, TimedPersistentObject, HyperAnimation, HyperMedium, Sound, Link, Paragraph.**



language (public class model)

Parent: **language**

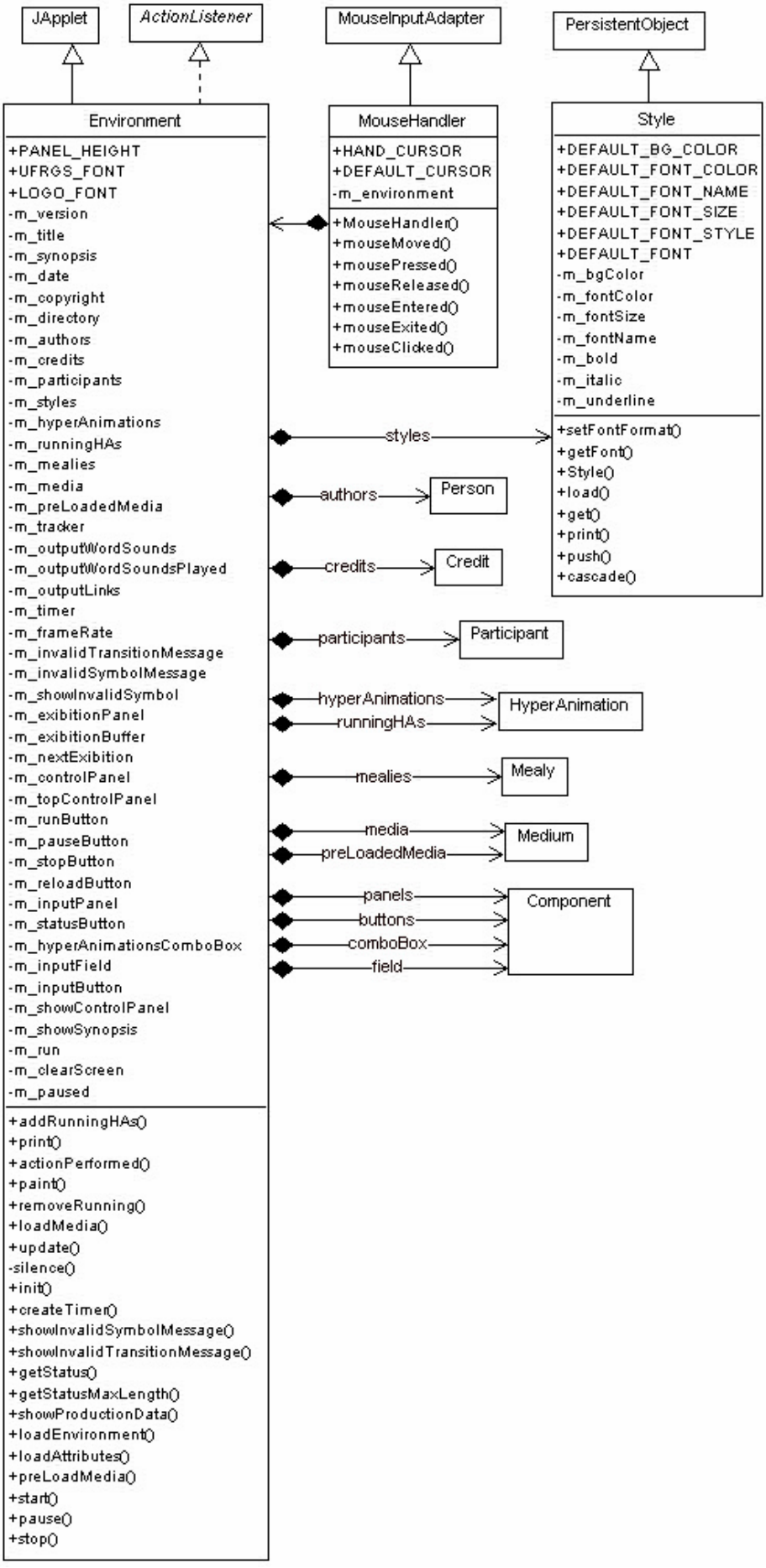
Contains: **ParserEnvironment, Parser, ParserMealy, ParserHyperMedium.**



environment (public class model)

Parent: **environment**

Contains: **MouseHandler, PersistentObject, ActionListener, Style, Environment, MouseInputAdapter, Person, Credit, Participant, HyperAnimation, Medium, Mealy, Component.**


JApplet
ActionListener
MouseInputAdapter
PersistentObject
Environment
+PANEL_HEIGHT
+UFRGS_FONT
+LOGO_FONT
-m_version
-m_title
-m_synopsis
-m_date
-m_copyright
-m_directory
-m_authors
-m_credits
-m_participants
-m_styles
-m_hyperAnimations
-m_runningHAs
-m_mealies
-m_media
-m_preLoadedMedia
-m_tracker
-m_outputWordSounds
-m_outputWordSoundsPlayed
-m_outputLinks
-m_timer
-m_frameRate
-m_invalidTransitionMessage
-m_invalidSymbolMessage
-m_showInvalidSymbol
-m_exibitionPanel
-m_exibitionBuffer
-m_nextExibition
-m_controlPanel
-m_topControlPanel
-m_runButton
-m_pauseButton
-m_stopButton
-m_reloadButton
-m_inputPanel
-m_statusButton
-m_hyperAnimationsComboBox
-m_inputField
-m_inputButton
-m_showControlPanel
-m_showSynopsis
-m_run
-m_clearScreen
-m_paused
+addRunningHAs()
+print()
+actionPerformed()
+paint()
+removeRunning()
+loadMedia()
+update()
-silence()
+init()
+createTimer()
+showInvalidSymbolMessage()
+showInvalidTransitionMessage()
+getStatus()
+getStatusMaxLength()
+showProductionData()
+loadEnvironment()
+loadAttributes()
+preLoadMedia()
+start()
+pause()
+stop()
MouseHandler
+HAND_CURSOR
+DEFAULT_CURSOR
-m_environment
+MouseHandler()
+mouseMoved()
+mousePressed()
+mouseReleased()
+mouseEntered()
+mouseExited()
+mouseClicked()
Style
+DEFAULT_BG_COLOR
+DEFAULT_FONT_COLOR
+DEFAULT_FONT_NAME
+DEFAULT_FONT_SIZE
+DEFAULT_FONT_STYLE
+DEFAULT_FONT
-m_bgColor
-m_fontColor
-m_fontSize
-m_fontName
-m_bold
-m_italic
-m_underline
+setFontFormat()
+getFont()
+Style()
+load()
+get()
+print()
+push()
+cascade()
styles
authors
Person
credits
Credit
participants
Participant
hyperAnimations
runningHAs
HyperAnimation
mealies
Mealy
media
preLoadedMedia
Medium
panels
buttons
comboBox
field
Component

persistence (public class model)

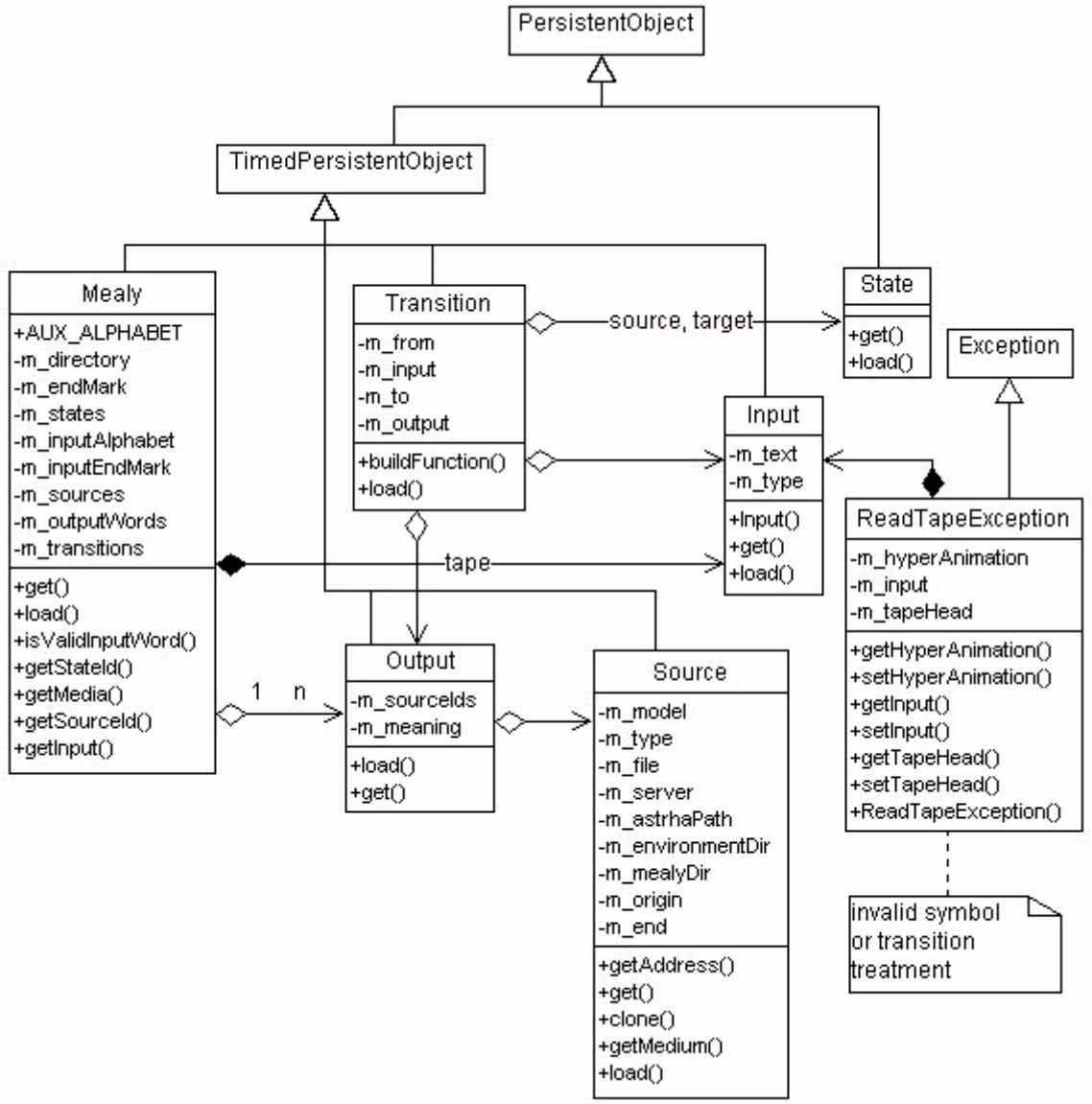
Parent: **persistence**

Contains: **PersistentObject, TimedPersistentObject, Cloneable.**

mealy (public class model)

Parent: **mealy**

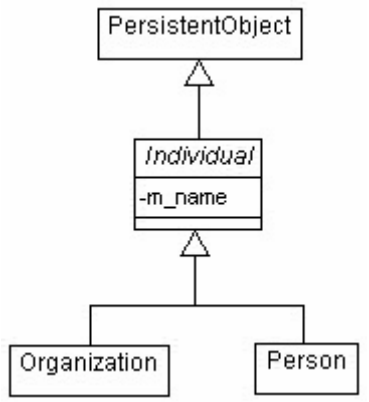
Contains: **Transition, Exception, Source, ReadTapeException, Mealy, TimedPersistentObject, State, Input, Output, invalid symbol or transition treatment.**



individual (public class model)

Parent: **individual**

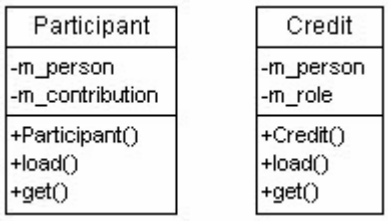
Contains: **Organization, Person, Individual.**



production (public class model)

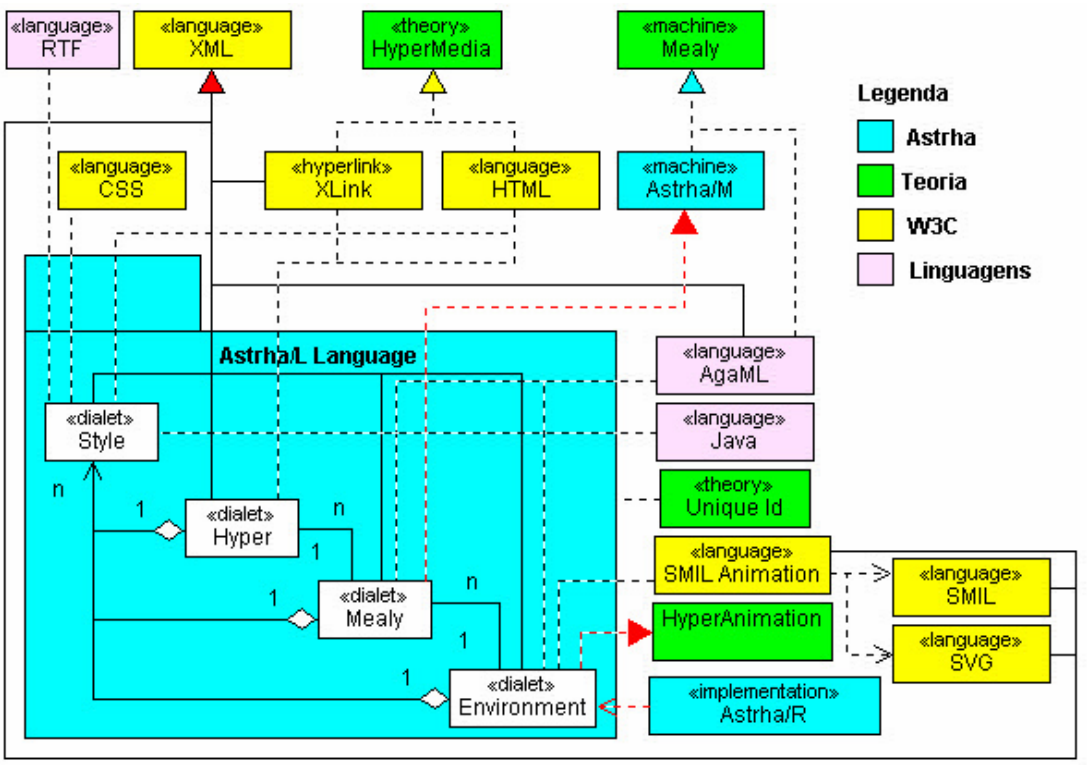
Parent: **production**

Contains: **Participant, Credit.**



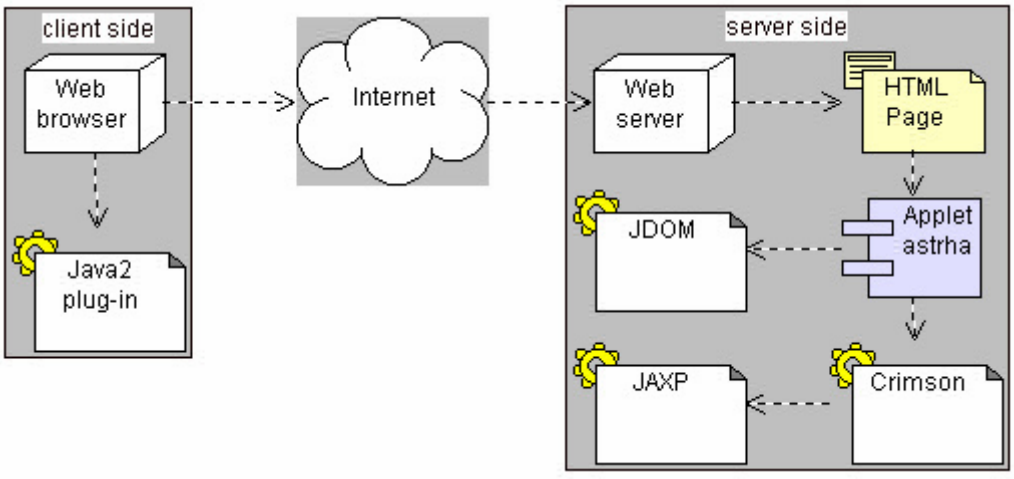
astrhaL (public class model)

Contains: **Astrha/L Language, XML, AgaML, Unique Id, Mealy, Style, Environment, Hyper, Mealy, XLink, CSS, SMIL Animation, Astrha/M, Astrha/R, SMIL, SVG, HyperMedia, HTML, Java, , RTF.**



implementation (public implementation model)

Contains: **Crimson, Applet astrha, Web server, HTML Page, JAXP, JDOM, server side, Web browser, Java2 plug-in, client side, Internet.**



Finite Automata Output (**public** Control Class)

Mealy (**public** Control Class)

Moore (**public** Control Class)

XHA (**public** Control Class)

Bases: **public Hyper-Automaton.**

Interactive Evaluation (**public** Control Class)

Bases: **public Hyper-Automaton.**

AGA (**public** Control Class)

HyperMedia Perl/HTML (**public** Note)

Relations:

Unification (**public** Control Class)

Bases: **public Interactive Evaluation, public XHA, public Hyper-Automaton, public AGA.**

Concepts? Structure? Technology? Product? Applicability? (**public** Note)

Relations:

HyperMedia JSP/XML XML chema HTML/CSS (**public** Note)

Relations:

Animations Java XML/DTD (**public** Note)

Relations:

HyperMedia/EAD Perl HTML (**public** Note)

Relations:

Hyper-Automaton (**public** Control Class)

HTML Page (**public** Client Page)

Relations:

Internet (**public** Arbitrary)

Relations:

Web server (**public** Node)

Relations:

Crimson (**public** Server Page)

Relations:

JAXP (**public** Server Page)

JDOM (**public** Server Page)

Web browser (**public** Node)

Relations:

Java2 plug-in (**public** Server Page)

client side (**public** Arbitrary)

server side (**public** Arbitrary)

Applet astrha (**public** Component)

Relations:

Astrha/L Language (**public** Package/Task)

«dialet» Mealy (**public** Class/Interface)

Bases: **public XML.**

Relations: **public #n -- #1 <Environment>***
public #1 -- #n <Style>*

«dialet» Hyper (**public** Class/Interface)

Bases: **public XML.**

Relations:
public #n -- #1 <Mealy>*
public #1 -- #n <Style>*

«dialet» Environment (public Class/Interface)

Bases: **public XML.**

Relations:

public #1 -- #n <Style>*

«dialet» Style (public Class/Interface)

Bases: **public XML.**

Relations:

«language» XML (public Class/Interface)

«language» AgaML (public Class/Interface)

Bases: **public XML.**

Relations:

«machine» Mealy (public Class/Interface)

«hyperlink» XLink (public Class/Interface)

Bases: **public XML.**

Relations:

«language» CSS (public Class/Interface)

HyperAnimation (public Class/Interface)

«language» SMIL Animation (public Class/Interface)

Bases: **public XML.**

Relations:

«machine» Astrha/M (public Class/Interface)

«implementation» Astrha/R (public Class/Interface)

Relations:

«language» SMIL (public Class/Interface)

Bases: **public XML.**

«language» SVG (public Class/Interface)

Bases: **public XML.**

«theory» HyperMedia (public Class/Interface)

«language» HTML (public Class/Interface)

Relations:

«language» Java (public Class/Interface)

Relations:

(public Class/Interface)

«language» RTF (public Class/Interface)

Relations:

«theory» Unique Id (public Class/Interface)

Relations:

Credit (public Class/Interface)

Methods:

```
public getPerson () : Person
public setPerson (val: Person)
public getRole () : String
public setRole (val: String)
public Credit ()
@since 1.0
public static load (root: Element) : Credit[]
@since 1.0
public static get (vec: Vector) : Credit[]
@since 1.0
public static print (credits: Credit[])
Para testes e log
@since 1.0
```

Get/Set:

Attributes:

```
m_person: Person = null
m_role: String = null
```

Participant (**public** Class/Interface)

Comment: *Participantes da produção da hiper-animação.*
@since 1.0

Methods: **public** getPerson () : Person
public setPerson (val: Person)
public getContribution () : String
public setContribution (val: String)
public Participant ()
@since 1.0
public static load (root: Element) : Participant[]
@since 1.0
public static get (vec: Vector) : Participant[]
@since 1.0
public static print (participants: Participant[])
Para testes e log
@since 1.0

Get/Set:

Attributes: m_person: Person = null
m_contribution: String = null

Individual (**public** Class/Interface)

Bases: **public** PersistentObject.

Methods: **public** getName () : String
public setName (val: String)

Get/Set:

Attributes: **private** m_name: String = null

Person (**public** Class/Interface)

Bases: **public** Individual.

Comment: *Indivíduos tipo pessoa.*
@since 1.0

Methods: **public static** get (vec: Vector) : Person[]
@since 1.0
public static load (root: Element) : Person[]
@since 1.0
public static print (people: Person[])
Para testes e log
@since 1.0

Organization (**public** Class/Interface)

Bases: **public** Individual.

JApplet (package Class/Interface)

Output (public Class/Interface)

Bases: **public TimedPersistentObject.**

Comment: ***@since 1.0***

Methods: **public getSourceIds () : Long[]**
get & set methods
public setSourceIds (val: Long[])
public setSourceIds (ids: String)
public getMeaning () : String
public setMeaning (val: String)
public static load (root: Element) : Output[]
@since 1.0
public static get (vec: Vector) : Output[]
@since 1.0

Get/Set:

Attributes: **private m_sourceIds: Long[] = null**
attributes
private m_meaning: String = null

Relations: **public -- <Source>***

Input (public Class/Interface)

Bases: **public TimedPersistentObject.**

Comment: ***@author Roges Grandi***
@since 1.0

Methods: **public getText () : String**
public setText (val: String)
public getType () : String
public setType (val: String)
public Input ()
@since 1.0
public Input (text: String)
@since 1.0
public static get (vec: Vector, mealy: Mealy) : Input[]
@since 1.0
public static load (root: Element) : HashSet
@since 1.0

Get/Set:

Attributes: **private m_text: String = null**
private m_type: String = null

State (public Class/Interface)

Bases: **public PersistentObject.**

Comment: ***Estados do autômato.***
@since 1.0

Methods: **public static get (vec: Vector) : State[] raises(Exception)**

@since 1.0
public static load (root: Element) : HashSet
@since 1.0

Mealy (public Class/Interface)

Bases: **public TimedPersistentObject.**

Comment: ***Tipo de mídia composto de imagens e, opcionalmente, sons, havendo flexibilidade na ordem de exposição dessas mídias, sendo possíveis mudanças nessa ordem através de solicitações realizadas pelo usuário.***
@since 1.0

Methods: **public getDirectory () : String**
get & set methods
public setDirectory (val: String)
public getEndMark () : String
public setEndMark (val: String)
public getStates () : HashSet
public setStates (val: HashSet) public getTransitions () : HashSet
public setTransitions (val: HashSet)
public getInputAlphabet () : HashSet
public setInputAlphabet (val: HashSet)
public getInputEndMark () : Input
public setInputEndMark (val: Input)
public getSources () : HashSet
public setSources (val: HashSet)
public getOutputs () : Output[]
public setOutputs (val: Output[])
public static get (vec: Vector) : Mealy[] raises(Exception)
@since 1.0
public static load (environment: Environment, root: Element) : HashSet
Carrega as mealies de um ambiente, sem, porém, pré-carregar suas mídias, que serão carregadas na memória somente no momento que forem necessárias, a fim de deixar a aplicação mais leve e otimizada.
@since 1.0
public isValidInputWord (input: Input) : boolean
@since 1.0
public getStateId (id: Long) : State

@since 1.0
public **getMedia () : HashSet**
@since 1.0
public **getSourceId (id: Long) : Source**
@since 1.0
public **getInput (text: String) : Input**
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "Mealy."**
public static AUX_ALPHABET: HashSet = null
private m_directory: String = null
attributes
private m_endMark: String = null

private m_states: HashSet = null
private m_inputAlphabet: HashSet = new HashSet()
private m_inputEndMark: Input = null
private m_sources: HashSet = new HashSet()
private m_outputWords: Output[] = new Output[0]
private m_transitions: HashSet = null

Has: **private tape: -- <Input>**

Relations: **private #1 -- #n <Output>***

ReadTapeException (public Class/Interface)

Bases: **public Exception.**

Comment: *@author Roges Grandi, Guilherme Magalhães*
@since 1.0

Methods: **public getHyperAnimation () : HyperAnimation**
get & set methods
public setHyperAnimation (val: HyperAnimation)
public getInput () : Input
public setInput (val: Input)
public getTapeHead () : int
public setTapeHead (val: int)
public ReadTapeException (hyperAnimation: HyperAnimation, input: Input, tapeHead: int)
@since 1.0

Get/Set:

Attributes: **private m_hyperAnimation: HyperAnimation = null**
attributes
private m_input: Input = null
private m_tapeHead: int = 0

Has: **private -- <Input>**

Source (public Class/Interface)

Bases: **public TimedPersistentObject.**

Comment: ***@since 1.0***

Methods: **public getModel () : String**
get & set methods
public setModel (val: String)
public getType () : String
public setType (val: String)
public getFile () : String
public setFile (val: String)
public getServer () : String
public setServer (val: String)
public getAstrhaPath () : String
public setAstrhaPath (val: String)
public getEnvironmentDir () : String
public setEnvironmentDir (val: String)
public getMealyDir () : String
public setMealyDir (val: String)
public getOrigin () : Point
public setOrigin (val: Point)
public setOrigin (x: String, y: String)

public getEnd () : Point
public setEnd (val: Point)
public setEnd (x: String, y: String)
public getAddress () : String
public static get (vec: Vector) : Source[]
@since 1.0
public static clone (outputs: Source[], outputId: Long) : Source raises(Exception)

Dado um id, busca sua mídia dentro de um vetor.
Se não houver mídia com esse id no vetor, retorna null.
@since 1.0
public getMedium () : Medium
@since 1.0
public static load (root: Element, environment: Environment, mealy: Mealy) : HashSet
@since 1.0

Get/Set:

Attributes: **private m_model: String = null**
attributes
private m_type: String = null
private m_file: String = null
private m_server: String = null
private m_astrhaPath: String = null
private m_environmentDir: String = null
private m_mealyDir: String = null
private m_origin: Point = null

private m_end: Point = null

Exception (**package** Class/Interface)

Transition (**public** Class/Interface)

Bases: **public** TimedPersistentObject.

Comment: *@author Roges Grandi, Guilherme Magalhães*
@since 1.0

Methods: **public** getFrom () : Long
get & set methods
public setFrom (val: String)
public getInput () : Long
public setInput (val: String)
public getTo () : Long
public setTo (val: String)
public getOutput () : Long
public setOutput (val: String)
public static buildFunction (vec: Vector) : HashSet raises(Exception)

@since 1.0
public static load (root: Element) : HashSet
@since 1.0

Get/Set:

Attributes: **private** m_from: Long = null
attributes
private m_input: Long = null
private m_to: Long = null
private m_output: Long = null

Relations: **public** -- <Input>*
public -- <Output>*
public source, target: -- <State>*

Cloneable (**package** Class/Interface)

TimedPersistentObject (**public** Class/Interface)

Bases: **public PersistentObject.**

Comment: ***@author Roges Grandi, Guilherme Magalhães***
@since 1.0

Methods: **public getTime () : int**
public setTime (val: String)
@since 1.0

Get/Set:

Attributes: **private m_time: int = -1**
Se time 0, executa tempo igual a seu valor.
Se time = 0, não executa.
Se time

PersistentObject (**public** Class/Interface)

Comment: ***@author Roges Grandi, Guilherme Magalhães***
@since 1.0

Methods: **public getId () : Long**
get & set methods
public setId (val: String)
@since 1.0
public getName () : String
public setName (val: String)
public getAuthor () : Individual
public setAuthor (val: Individual)
public getX () : Integer
public setX (val: String)
public getY () : Integer
public setY (val: String)
public getWidth () : Integer
public setWidth (val: String)
public getHeight () : Integer
public setHeight (val: String)
public getIdName () : String
other operations

Get/Set:

Attributes: **private m_id: Long = null**
attributes
private m_name: String = null
private m_author: Individual = null
private m_x: Integer = null
private m_y: Integer = null
private m_width: Integer = null
private m_height: Integer = null

MouseInputAdapter (**package** Class/Interface)

Environment (**public** Class/Interface)

Bases: **public JApplet.**

Comment: ***@author Roges Grandi, Guilherme Magalhães***
@since 1.0

Methods: **public getInvalidSymbolMessage () : String**
get & set methods
public setInvalidSymbolMessage (val: String)
public getShowInvalidSymbol () : boolean
public setShowInvalidSymbol (val: String)
public getVersion () : String
public setVersion (val: String)
public getDirectory () : String
public setDirectory (val: String)
public getDate () : String
public setDate (val: String)
public getCopyright () : String
public setCopyright (val: String)
public getAuthors () : Person[]
public setAuthors (val: Person[])
public getCredits () : Credit[]
public setCredits (val: Credit[])
public getParticipants () : Participant[]
public setParticipants (val: Participant[])
public getFrameRate () : int
public setFrameRate (val: String)
@since 1.0
public getTitle () : String
public setTitle (val: String)
public getSynopsis () : String
public setSynopsis (val: String)
public getMedia () : HashSet
public setMedia (val: HashSet)
public getMealies () : HashSet
public setMealies (val: HashSet)
public getPreLoadedMedia () : HashSet
public setPreLoadedMedia (val: HashSet)
public getTracker () : MediaTracker
public setTracker (val: MediaTracker)
public getHyperAnimations () : HyperAnimation[]
public setHyperAnimations (val: HyperAnimation[])
public getHyperAnimation (id: Long) : HyperAnimation
public getMealy (id: Long) : Mealy
@since 1.0
public getRunningHAs () : HyperAnimation[]
public setRunningHAs (val: HyperAnimation[])
public addRunningHAs (added: HyperAnimation[])
@since 1.0
public getStyles () : HashSet
public setStyles (val: HashSet)

public getStyle () : Style
public setStyle (val: String)
@since 1.0
public getRun () : boolean
public setRun (val: boolean)
public getShowControlPanel () : boolean
public setShowControlPanel (val: boolean)
public getShowSynopsis () : boolean
public setShowSynopsis (val: boolean)
public getClearScreen () : boolean
public setClearScreen (val: boolean)
public getPaused () : boolean
public setPaused (val: boolean)
public getNextExibition () : Graphics
public setNextExibition (val: Graphics)
public getExibitionPanel () : JPanel
public setExibitionPanel (val: JPanel)
public getControlPanel () : JPanel
public setControlPanel (val: JPanel)
public getTopControlPanel () : JPanel
public setTopControlPanel (val: JPanel)
public getRunButton () : JButton
public setRunButton (val: JButton)
public getPauseButton () : JButton
public setPauseButton (val: JButton)
public getStopButton () : JButton
public setStopButton (val: JButton)
public getReloadButton () : JButton
public setReloadButton (val: JButton)
public getInputButton () : JButton
public setInputButton (val: JButton)
public getStatusButton () : JButton
public setStatusButton (val: JButton)
public getInputPanel () : JPanel
public setInputPanel (val: JPanel)
public getHyperAnimationsComboBox () : JComboBox
public setHyperAnimationsComboBox (val: JComboBox)
public getInputField () : JTextField
public setInputField (val: JTextField)
public getExibitionBuffer () : BufferedImage
public setExibitionBuffer (val: BufferedImage)
public getTimer () : Timer
public setTimer (val: Timer)
public getOutputSounds () : Vector
public setOutputSounds (val: Vector)
public getOutputSoundsPlayed () : Vector
public setOutputSoundsPlayed (val: Vector)
public getOutputLinks () : Link[]
public setOutputLinks (val: Link[])
public static print (environment: Environment)

Para testes e log
@since 1.0
public **actionPerformed (event: ActionEvent)**
Este método pertence à interface ActionListener.
Sua função é receber e manipular eventos recebidos.
Uma vez que o ActionListener está sendo controlado por
um timer, eventos podem ser disparados de três fontes:
1) do timer, que dispara um evento de tempos em tempos,
conforme o retardo (delay) programado
2) de ações do mouse sobre o painel de controle ou
3) de ações do mouse sobre hiperligações.

Aqui, além de aplicada transições para cada autômato,
afim de configurar a próxima cena,
também é configurada transformações em imagens.
public **paint (g: Graphics)**
Desenha um state.
public **removeRunning (pos: int)**
@since 1.0
public **loadMedia (g: Graphics)**
Carrega no buffer as mídias para serem exibidas
nesta cena.
Aqui, as transformações nas imagens são aplicadas.
public **getWidth (hyperAnimation: HyperAnimation, output: Output, source: Source) : int**
@since 1.0
public **getHeight (hyperAnimation: HyperAnimation, output: Output, source: Source) : int**
@since 1.0
public synchronized final **update (g: Graphics)**
Exibe um state da animação.
Desencadeado pelo repaint().
Cada state é desenhado em um buffer offscreen
no método paint()
Quando o buffer estiver pronto, é exibido.
@see #paint(Graphics)
private **silence ()**
Pára todos os sons que estiverem tocando na animação.
public **init ()**
Inicializa e dispara a animação.
public **createTimer ()**
@since 1.0
public **showInvalidSymbolMessage (ha: HyperAnimation, inputText: String, tapeHead: int)**
@since 1.0
public **showInvalidTransitionMessage (rte: ReadTapeException)**
@since 1.0
public **getStatus () : String**
@since 1.0
public **getStatusMaxLength () : int**

@since 1.0
public showProductionData (g: Graphics)
@since 1.0
public loadEnvironment ()
@since 1.0
public loadAttributes (root: Element)
@since 1.0
public preLoadMedia (mealy: Mealy)
@since 1.0
public synchronized start ()
Ativa o timer.
@see #timer
public synchronized pause ()
Pausa a animação desativando o timer e
mantendo os estados do autômato de cada ator.
public synchronized stop ()
Pára a animação e reinicia a fita da animação.

Get/Set:

Attributes:
public final static CLASS: String = "Environment."
public final static PANEL_HEIGHT: int = 81
public final static UFRGS_FONT: Font = new
Font("Helvetica",Font.BOLD,20)
public final static LOGO_FONT: Font = new
Font("Helvetica",Font.BOLD,16)
private m_version: String = null
Versão do ambiente astrha para o qual os arquivos em Astrha/L foram codificados.
private m_title: String = null
Título da produção.
private m_synopsis: String = null
Sinopse da produção.
private m_date: String = null
Data de criação da produção.
private m_copyright: String = null
Mensagem de direitos autorais sobre a produção.
private m_directory: String = null
Diretório que contém a estrutura e as mídias do ambiente.
private m_authors: Person[] = null
Nomes dos autores da produção.
private m_credits: Credit[] = null
Lista de créditos da produção.
private m_participants: Participant[] = null
Lista de participantes que auxiliaram a produção.
private m_styles: HashSet = null
Conjunto de estilos que podem ser utilizados neste ambiente.
private m_hyperAnimations: HyperAnimation[] = null
Conjunto de hiper-animações disponíveis para o ambiente.
private m_runningHAs: HyperAnimation[] = null
Lista de hiper-animações que estão em execução no ambiente.

```
private m_mealies: HashSet = new HashSet()
Conjunto de máquinas de Mealy disponíveis para estruturar as
hiper-animações.
private m_media: HashSet = new HashSet()
Conjunto de mídias disponíveis para as hiper-animações.
private m_preLoadedMedia: HashSet = new HashSet()
Conjunto de mídias disponíveis para as hiper-animações que já
foram pré-carregadas.
private m_tracker: MediaTracker = null
Utilizado para sincronizar o carregamento de imagens
antes de exibi-las em um determinado estado.
private m_outputWordSounds: Vector = new Vector()
Lista de mídias auditivas que compõem a palavra de saída atual.
private m_outputWordSoundsPlayed: Vector = new Vector()
Lista de mídias auditivas que compõem a palavra de saída atual que
já foram executadas.
private m_outputLinks: Link[] = new Link[0]
Lista de hiperligações que compõem a palavra de saída atual.
private m_timer: Timer = null
Temporizador que controla o tempo de transição de estados de cada
hiper-animação em execução. A taxa de atualização de quadros é
determinada pela propriedade m_frameRate).
@see #m_frameRate
private m_frameRate: int = 1000
Taxa de atualização para o temporizador, em milisegundos.
O tempo padrão é 1000ms.
@see #m_timer
private m_invalidTransitionMessage: String = null
USER INTERFACE ATTRIBUTES
MESSAGES
private m_invalidSymbolMessage: String = null
private m_showInvalidSymbol: boolean = true
private m_exibitionPanel: JPanel = null
Objeto usado para carregar um state na animação e ser passado
para o layout da applet.
private m_exibitionBuffer: BufferedImage = null
Buffer auxiliar para desenhar offscreen, evitando flicker.
Utilizado pelo m_exibitionPanel
@see #m_exibitionPanel.
private m_nextExibition: Graphics = null
Utilizado para montar elementos gráficos para a próxima cena
no exibition panel.
private m_controlPanel: JPanel = null
Painel de controle.
private m_topControlPanel: JPanel = null
Painel para dispor objetos para a parte superior do painel de
controle.
private m_runButton: JButton = null
Botão que dispara o temporizador da animação.
private m_pauseButton: JButton = null
```

Botão que pára o temporizador sem reiniciar a animação.
private m_stopButton: JButton = null
Botão que pára o temporizador, reiniciando a animação.
private m_reloadButton: JButton = null
Botão que pára o temporizador, relê todo o programa de entrada e reinicializa a animação.
private m_inputPanel: JPanel = null
Painel para dispor objetos para a parte superior do painel de controle.
private m_statusButton: JButton = null
Botão que dispara a exibição de uma tela pop up contendo dados sobre o estado atual da hiper-animação em termos de Máquina de Mealy.
private m_hyperAnimationsComboBox: JComboBox = null
ComboBox que seleciona qual hiper-animação receberá uma fita de entrada através do botão m_inputButton.
@see m_inputButton.
private m_inputField: JTextField = null
Campo de texto para receber uma fita de entrada.
private m_inputButton: JButton = null
Campo para atualizar a fita de entrada recebida no cambo de texto m_inputField, relativo à hiper-animação selecionada pela m_hyperAnimationsComboBox.
private m_showControlPanel: boolean = true
private m_showSynopsis: boolean = true
private m_run: boolean = false
private m_clearScreen: boolean = false
private m_paused: boolean = false

Has: **public field: -- <Component>**
public comboBox: -- <Component>
public buttons: -- <Component>
public panels: -- <Component>
public mealies: -- <Mealy>
public preLoadedMedia: -- <Medium>
public media: -- <Medium>
public runningHAs: -- <HyperAnimation>
public hyperAnimations: -- <HyperAnimation>
public participants: -- <Participant>
public credits: -- <Credit>
public authors: -- <Person>
private styles: -- <Style>

Style (public Class/Interface)

Bases: **public PersistentObject.**

Comment: *@since 1.0*

Methods: **public getBgColor () : Color**
get & set methods
public setBgColor (val: String) raises(Exception)

@since 1.0
public getFontColor () : Color
public setFontColor (val: String) raises(Exception)

@since 1.0
public getFontSize () : int
public setFontSize (val: String)
@since 1.0
public getFontName () : String
public setFontName (val: String)
@since 1.0
public getBold () : boolean
public setBold (val: boolean)
public getItalic () : boolean
public setItalic (val: boolean)
public getUnderline () : boolean
public setUnderline (val: boolean)
public setFontFormat (style: String)
@since 1.0
public getFont () : Font
@since 1.0
public Style ()
@since 1.0
public Style (root: Element)
@since 1.0
public static load (root: Element) : HashSet
@since 1.0
public static get (styles: HashSet, id: Long) : Style
@since 1.0
public static print (styles: HashSet)
Para testes e log
@since 1.0
public static push (stack: Stack, styles: HashSet, styleId: String)
@since 1.0
public static cascade (env: Environment, hm: HyperMedium) : HashSet
@since 1.0

Get/Set:

Attributes: **public static DEFAULT_BG_COLOR: Color = new Color(0xD3,0xD3,0xD3)**
public static DEFAULT_FONT_COLOR: Color = Color.black
public static DEFAULT_FONT_NAME: String = "Helvetica"
public static DEFAULT_FONT_SIZE: int = 12
public static DEFAULT_FONT_STYLE: int = Font.PLAIN
public final static DEFAULT_FONT: Font = new

Font(DEFAULT_FONT_NAME,DEFAULT_FONT_STYLE,DEFAULT_FONT_SIZE)
private m_bgColor: Color = DEFAULT_BG_COLOR
attributes
private m_fontColor: Color = DEFAULT_FONT_COLOR
private m_fontSize: int = DEFAULT_FONT_SIZE
private m_fontName: String = DEFAULT_FONT_NAME
private m_bold: boolean = false
referentes à fontStyle
private m_italic: boolean = false
private m_underline: boolean = false

ActionListener (package Class/Interface)

MouseHandler (public Class/Interface)

Bases: **public MouseInputAdapter.**

Comment: ***@since 1.0***

Methods: **public getEnvironment () : Environment**
public setEnvironment (val: Environment)
public MouseHandler (environment: Environment)
@since 1.0
public mouseMoved (e: MouseEvent)
Concretização de métdo abstrato de MouseMotionListener(MouseEvent).
public mousePressed (ev: MouseEvent)
public mouseReleased (e: MouseEvent)
public mouseEntered (e: MouseEvent)
public mouseExited (e: MouseEvent)
public mouseClicked (e: MouseEvent)

Get/Set:

Attributes: **public final static HAND_CURSOR: Cursor = new Cursor(Cursor.HAND_CURSOR)**
public final static DEFAULT_CURSOR: Cursor = new Cursor(Cursor.DEFAULT_CURSOR)
private m_environment: Environment = null
Aponta para o objeto Environment.

Has: **private -- <Environment>**

ParserHyperMedium (**public** Class/Interface)

Bases: **public Parser.**

Comment: ***Uma classe para ler uma XML que contém uma fita de entrada para uma Mealy.***
@author Roges Grandi
@version 1.0

Methods: **public static parse (url: URL, environment: Environment) : Paragraph[] raises(Exception)**

@since 1.0
public static parse (file: File, environment: Environment) : Paragraph[] raises(Exception)

@since 1.0
public static parse (doc: Document, environment: Environment) : Paragraph[] raises(Exception)

@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "ParserHyperMedium."**

ParserMealy (**public** Class/Interface)

Bases: **public Parser.**

Comment: ***Uma classe para ler o XML que gera a animação e carregar as informações.***
@author Roges Grandi, Guilherme de C. Magalhães
@version 1.0

Methods: **public static parse (environment: Environment, mealy: Mealy)**
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "ParserMealy."**

Parser (public Class/Interface)

Comment: *Uma classe para ler o XML que gera a animação e carregar as informações.*
@author Roges Grandi, Guilherme de C. Magalhães
@version 1.0

Methods: **public static setDocumentRoot (docRoot: String)**
@since 1.0
public static getAttributeValue (el: Element, attribute: String) : String
@since 1.0
public static parseColor (val: String) : Color
As cores aqui pré-definidas seguem, preferencialmente, o padrão de cores HTML 3.2. São acrescentadas algumas cores pré-definidas Java. Havendo conflito de especificação de tonalidades entre HTML e Java, deu-se preferência ao padrão HTML.
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "Parser."**
public static WEB_SERVER: String = "http://localhost"
Reconfigurável
public static ENVIRONMENT_FILE: String = "/environment.xml"
public static MEALY_FILE: String = "/mealy.xml"
public static HYPER_ANIMATIONS_FILE: String = "/hyperanimations.xml"
public static ASTRHA_PATH: String = "/astrha/environments"
public final static COLOR_AQUA: Color = new Color(0x00,0xFF,0xFF)
HTML.aqua = Color.cyan.
public final static COLOR_BLACK: Color = new Color(0x00,0x00,0x00)
HTML.black = Color.black.
public final static COLOR_BLUE: Color = new Color(0x00,0x00,0xFF)
HTML.blue = Color.blue.
public final static COLOR_DARK_GRAY: Color = new Color(0x40,0x40,0x40)
HTML não tem. Color.darkGray,
public final static COLOR_FUCHSIA: Color = new Color(0xFF,0x00,0xFF)
HTML.fuchsia = Color.magenta.
public final static COLOR_GRAY: Color = new Color(0x80,0x80,0x80)
HTML.gray. Color não tem,
public final static COLOR_GREEN: Color = new Color(0x00,0x80,0x00)
HTML.green. Color tem valor diferente (00FF00)
public final static COLOR_LIME: Color = new Color(0x00,0xFF,0x00)
HTML.lime = Color.green.
public final static COLOR_MAROON: Color = new Color(0x80,0x00,0x00)
HTML.maroon. Color não tem,
public final static COLOR_NAVY: Color = new Color(0x00,0x00,0x80)
HTML.navy. Color não tem.
public final static COLOR_OLIVE: Color = new Color(0x80,0x80,0x00)

```
HTML.olive. Color não tem.
public final static COLOR_ORANGE: Color = new
Color(0xFF,0xC8,0x00)
HTML não tem. Color.orange,
public final static COLOR_PINK: Color = new Color(0xFF,0xAF,0xAF)
HTML não tem. Color.pink.
public final static COLOR_PURPLE: Color = new
Color(0x80,0x00,0x80)
HTML.purple. Color não tem.
public final static COLOR_RED: Color = new Color(0xFF,0x00,0x00)
HTML red = Color.red.
public final static COLOR_SILVER: Color = new
Color(0xC0,0xC0,0xC0)
HTML.silver = lightGray.
public final static COLOR_TEAL: Color = new Color(0x00,0x80,0x80)
HTML.teal. Color não tem.
public final static COLOR_WHITE: Color = new
Color(0xFF,0xFF,0xFF)
HTML.white = Color.white
public final static COLOR_YELLOW: Color = new
Color(0xFF,0xFF,0x00)
HTML.yellow = Color.yellow
public static HTTP_BASE: String = WEB_SERVER+ASTRHA_PATH
Para execuções reais via Internet usando protocolo HTTP.
Opcionalmente, serve também para testes.
Reconfigurável através de WEB_SERVER;
public static FILE_BASE: String =
"/inetpub/wwwroot"+ASTRHA_PATH
Para testes de execução sem HTTP.
Opcionalmente, pode servir para execução fora da Internet.
public static INPUT_FILE: String = "/input.xml"
```

ParserEnvironment (public Class/Interface)

Bases: **public Parser.**

Comment: ***Uma classe para ler uma XML que contém uma fita de entrada para uma HyperAnimation.***
@author Roges Grandi
@version 1.0

Methods: **public static parse (environment: Environment, directory: String)**
Constrói um novo objeto Parser. Este construtor é o default para esta classe. Ele lê o XML que gera a animação e o carrega em memória.
@param url Endereço internet do XML a ser lido.
public static parse (environment: Environment, directory: String, basePath: String)
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "ParserEnvironment."**

media (public Package/Task)

Submodel: **media**

language (public Package/Task)

Submodel: **language**

environment (public Package/Task)

Submodel: **environment**

persistence (public Package/Task)

Submodel: **persistence**

mealy (public Package/Task)

Submodel: **mealy**

individual (public Package/Task)

Submodel: **individual**

production (public Package/Task)

Submodel: **production**

PersistentObject (package Class/Interface)

TimedPersistentObject (package Class/Interface)

Bases: **public PersistentObject.**

Medium (**public** Class/Interface)

Bases: **public TimedPersistentObject.**

Methods: **public getType () : String**
get & set methods
public setType (val: String)
public getURL () : URL
public setURL (val: URL)
public static create (type: String, urlFile: String) : Medium
@since 1.0
public static get (vec: Vector) : Medium[] raises(Exception)

@since 1.0
public static clone (media: Medium[], mediumId: Long) : Medium raises(Exception)

Dado um id, busca sua mídia dentro de um vetor.
Se não houver mídia com esse id no vetor, retorna null.
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "Medium." private m_type: String = null**
attributes
private m_URL: URL = null

Figure (**public** Class/Interface)

Bases: **public VisualRepresentation.**

VisualRepresentation (**public** Class/Interface)

Bases: **public Medium.**

PersistentObject (**package** Class/Interface)

TimedPersistentObject (**package** Class/Interface)

Bases: **public PersistentObject.**

HyperAnimation (**public** Class/Interface)

Bases: **public Medium.**

Comment: ***Tipo de mídia composto de imagens e, opcionalmente, sons, havendo flexibilidade na ordem de exposição dessas mídias, sendo possíveis mudanças nessa ordem através de solicitações realizadas pelo usuário.***
@since 1.0

Methods: **public getMealy () : Mealy**
get & set methods
public setMealy (val: Mealy)
public getBgColor () : Color

public setBgColor (val: String) raises(Exception)

@since 1.0
public getTimesToExecute () : int
public setTimesToExecute (val: int)
public getTimesExecuted () : int
public setTimesExecuted (val: int)
public lastState () : boolean
public setLastState (val: boolean)
public getInitialStateId () : Long
public setInitialStateId (val: Long)
public getPresentStateId () : Long
public setPresentStateId (val: Long)
public getPresentState () : State
@since 1.0
public getRandom () : Random
public setRandom (val: Random)
public getLastOutput () : Output
public setLastOutput (val: Output)
public getTape () : Input[]
public setTape (val: Input[])
public getTapeHead () : int
public setTapeHead (val: int)
public getBegin () : boolean
public setBegin (val: String)
@since 1.0
public getNew () : boolean
public setNew (val: String)
@since 1.0
public static get (vec: Vector) : Medium[]
@since 1.0
public static clone (hyperanimations: HyperAnimation[],
hyperanimationId: Long) : HyperAnimation raises(Exception)

Dado um id, busca sua mídia dentro de um vetor.
Se não houver mídia com esse id no vetor, retorna null.
@since 1.0
public static load (environment: Environment, root: Element)
@since 1.0
public getNextTransition (currentState: State, input: Input) : Transition
Dada a cena atual e a palavra de entrada,
procura nas possíveis transições de cenas
a próxima cena para este estado, com esta palavra
de entrada.
@since 1.0
public readTape () : Output raises(Exception)

Cabeça de leitura da fita.
@since 1.0
public getOutput (transition: Transition) : Output

@since 1.0
public getSources (output: Output) : Source[]
@since 1.0
public finished () : boolean
@since 1.0
public isTapeEmpty () : boolean
@since 1.0
public headAfterInputWords () : boolean
@since 1.0
public choose (set: HashSet) : Transition
@since 1.0
public isValidInputWord (input: Input) : boolean
@since 1.0
public getOrigin (output: Output, source: Source) : Point
@since 1.0
public static join (first: HyperAnimation[], second: HyperAnimation[]) : HyperAnimation[]
@since 1.0
public reset ()
@since 1.0
public static reset (has: HyperAnimation[])
@since 1.0

Get/Set:

Attributes:
public final static CLASS: String = "HyperAnimation."
public final static LOOP: int = -1
public final static ONCE: int = 0
private m_mealy: Mealy = null
attributes
private m_bgColor: Color = null
private m_initialStateId: Long = null
private m_presentStateId: Long = null
private m_lastOutput: Output = null
private m_tape: Input[] = new Input[0]
private m_tapeHead: int = 0
private m_begin: boolean = true
private m_new: boolean = true
private m_random: Random = new Random(System.currentTimeMillis())
Utilizado para realizar não determinismo aparente.
private m_lastState: boolean = false
private m_timesToExecute: int = 0
Se m_timesToExecute 0, repete quantidade de vezes igual a seu valor.
Se m_timesToExecute = 0, não repete.
Se m_timesToExecute
private m_timesExecuted: int = 0
private m_end: Point = null

Has:
public -- <Medium>

HyperMedium (public Class/Interface)

Bases: **public VisualRepresentation.**

Methods: **public getText () : String**
get & set methods
public setText (val: String)
public getStyle () : Style
public setStyle (val: Style)
public getStyles () : HashSet
public setStyles (val: HashSet)
public getHyperAnimations () : HyperAnimation[]
public setHyperAnimations (val: HyperAnimation[])
public loadHyperAnimations (root: Element, environment: Environment)
Carrega as hiper-animações que serão exibidas por este documento hipermídia.
@since 1.0
public hasLink () : boolean
Indica se este documento hiper-animado possui links. Sabemos disso através do vetor de hiper-animações.
public static loadParagraph (parent: Element, el: Element, environment: Environment, styles: HashSet, stylesStack: Stack, loadedHyperMedia: Vector) raises(Exception)

Lê um elemento do tipo parágrafo (marca) em um documento escrito no dialeto hyper de Astrha/L.
Sendo recursivamente chamada, pode ser também os seguintes elementos:
: um estilo
: um texto
: uma âncora hipermídia estendida
: um mapa-clicável (imagem)
Um mapa clicável podem invocar, internamente:
: uma lista de âncoras
: uma âncora hipermídia estendida dentro de
@since 1.0
public static getWords (text: String) : String[]
@since 1.0
public static getSubstrings (str: String, c: char) : String[]
Obtem uma lista de substrings do string line, separando-os pelo caracter c. Diferentemente da classe StringTokenizer, ele considera substrings em branco.
@since 2.1
public static getLongs (str: String, c: char) : Long[]
Obtem uma lista de substrings do string line, separando-os pelo caracter c. Diferentemente da classe StringTokenizer, ele considera substrings em branco.
@since 2.1
public static getBooleans (str: String, c: char) : boolean[]
Obtem uma lista de substrings do string line, separando-os

pelo caracter c. Diferentemente da classe StringTokenizer, ele considera substrings em branco.
@since 2.1
public getLink () : Link
@since 1.0
public static getHyperMedium (vec: Vector) : HyperMedium[]
raises(Exception)

@since 1.0
public static removeSpecialChars (content: String) : String
raises(Exception)

@since 1.0
public static getArea (hyperMedium: HyperMedium, str: String, x: int, y: int) : Rectangle
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "HyperMedium."**
public final static DEFAULT_STYLE: Style = new Style()
private m_style: Style = DEFAULT_STYLE
attributes
private m_styles: HashSet = new HashSet()
private m_text: String = ""

Has: **public -- <Medium>**

Relations: **private #1 -- #n <HyperAnimation>***

Sound (public Class/Interface)

Bases: **public Medium.**

Link (public Class/Interface)

Bases: **public PersistentObject.**

Comment: *@since 1.0*

Methods: **public getHyperAnimations () : HyperAnimation[]**
public setHyperAnimations (val: HyperAnimation[])
public getArea () : Rectangle
public setArea (val: Rectangle)
public static get (vec: Vector) : Link[]
@since 1.0
public static join (first: Link[], second: Link[]) : Link[]
@since 1.0

Get/Set:

Attributes: **public final static PROD_ID: int = 0**
public final static PROD_NEW_WINDOW: int = 1
private m_area: Rectangle = null

Relations: **private #1 -- #n <HyperAnimation>***

Paragraph (**public** Class/Interface)

Methods: **public getHyperMedium () : HyperMedium[]**
get & set methods
public setTexts (val: HyperMedium[])
public static get (vec: Vector) : Paragraph[] raises(Exception)

@since 1.0
public Paragraph (texts: HyperMedium[])
@since 1.0
public static draw (paragraphs: Paragraph[], g2D: Graphics2D, origin: Point, width: int, height: int) : Link[]
other operations
public getMaxFontHeight () : int
@since 1.0

Get/Set:

Attributes: **public final static CLASS: String = "Paragraph."**

Relations: **private #1 -- #n <HyperMedium>***

invalid symbol or transition treatment (**public** Note)

Relations:

Person (**public** Class/Interface)

Credit (**public** Class/Interface)

Participant (**public** Class/Interface)

Medium (**public** Class/Interface)

Mealy (**public** Class/Interface)

Component (**public** Class/Interface)

Comment: ***Componentes dos pacotes awt e swing para desenho de interface.***

Carregar ambiente (**public** Use Case)

Carregar ambiente com painel (**public** Use Case)

Relations:

Carregar ambiente sem painel (**public** Use Case)

Relations:

Iniciar execução (**public** Use Case)

Usuário (**public** Actor)

Relations: **public** -- <**Atualizar configuração**>*
public -- <**Selecionar hiper-animação**>*
public -- <**Pausar execução**>*
public -- <**Parar execução**>*
public -- <**Iniciar execução**>*
public -- <**Carregar ambiente**>*

Parar execução (**public** Use Case)

Pausar execução (**public** Use Case)

Relations:

Selecionar hiper-animação (**public** Use Case)

Informar nova fita (**public** Use Case)

Relations:

Atualizar configuração (**public** Use Case)

Atualizar todo o ambiente (**public** Use Case)

Relations:

astrha (**public** Arbitrary)

Glossário

ambiente hipermídia	Interface multimídia entre ser humano e computador que permite recuperação de informações através de hiperligações entre documentos [WIK 2002].
animação	Conforme Thalmann, uma animação por computador consiste em "modificar uma cena no tempo. Uma animação pode ser definida, também, como um processo no qual cada quadro de vídeo *(frame)* em um filme é produzido individualmente, causando-nos uma sensação de movimento. Existem várias técnicas para se obter uma animação, com ou sem o uso de computadores. Nos casos de uso de computadores, são objeto de estudo especializado especialmente na área da computação gráfica [THA 91, THA 85, ACC 2002 apud PUE 88, FOL 90].
astrha	Acrônimo criado para abreviar Hiper-Animações Estruturadas por Autômatos *(Automata Structured Hyper-Animations)*, referindo-se à dissertação de mestrado "*Astrha* –Um Ambiente Gráfico, Dinâmico e Interativo para Internet Baseado em Hiper-Animações e na Teoria dos Autômatos". Porto Alegre: Instituto de Informática da UFRGS, 2003.
DTD	Uma Definição de Tipo de Documento (*Document Type Definition*, DTD) é um arquivo (ou conjunto de arquivos) escrito em Sintaxe Declarativa XML (XML*'s Declaration Syntax*) que contém uma descrição formal de um tipo de documento em particular. Uma DTD configura quais nomes pdem ser utilizados por elementos XML, onde eles podem ocorrer e de que maneira são ordenados [FLY 2003].
e-learning	Termo da língua inglesa derivado da composição da abreviação dos termos eletrônico *(electronic)*, utilizado em contexto Internet, e aprendizado *(learning)*, significando aprendizado assistido por computador através da Internet.

hiper-animação	Termo criado por Frank M. Kappe em 1990 derivado das palavras hipermídia *(hypermedia)* e animação *(animation)*. Conforme Kappe, uma hiper-animação é um "conceito para a criação de animações interativas de tempo real. Trata-se, essencialmente, da combinação das tecnologias da Computação Gráfica e Hipermídia, a qual denominamos *Hiper-Animação*... Uma hiper-animação não se limita às animações tradicionais: combina imagens, vídeos digitais, animações gráficas e sons com dispositivos de entrada não usuais a fim de realizar um alto nível de interatividade... ligações de uma seqüência de animação para todos os tipos de informação, inclusive outras seqüências de animação, podem ser definidas... O conceito de Hiper-Animação possui uma abordagem genérica, permitindo a criação de seqüências animadas complexas (em termos de complexidade e quantidade de objetos, complexidade de movimentos) e - o mais importante – que podem ser editadas convenientemente [KAP 91, p. 101]." Pode-se entender, também, como uma combinação de mídia e ambiente hipermídia que apresenta fontes visuais através de sobreposição ou justaposição, conforme uma determinada taxa de atualização, permitindo que seu usuário decida caminhos de apresentação através de hiperligações.
hipermídia	Termo criado por Theodor H. Nelson em 1967 (hypermedia), derivado das palavras hipertexto *(hypertext)* e multimídia *(multimedia)*. Várias áreas da ciência da computação (incluindo outras ciências) envolvem essa teoria, incluindo: Recuperação de Informações *(Information Retrieval)*, Computação Gráfica *(Computer Graphics)*, Educação Assistida por Computador *(Computer Based Learning)*, Interação entre Seres Humanos e Sistemas Computacionais *(Human-Computer Interactions)*, Projeto de Interfaces *(User Interface Design)*, editoração eletrônica *(Electronic Publishing)*, Sistemas Comunicantes *(Communication Systems)* e Psicologia Cognitiva *(Cognitive Psychology)* [NEL 67, KAP 91, p. 1]
hyper-automaton	Termo derivado das palavras hipertexto *(hypertext)* e autômato *(automaton)* , referindo-se à dissertação de mestrado "Hyper-Automaton: Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída", cujo objetivo geral "... centra-se no estudo da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de Cursos na Web [MAC 2000]".
hyper-automaton: interactive evaluation	Dissertação de mestrado com título *Hyper-Automaton*:Avaliação Interativa de Alunos em Cursos na WEB Baseado em Autômatos Finitos, cujo objetivo é "...a elaboração de uma técnica da aplicação do formalismo de Autômatos Finitos com Saída (Máquina de Mealy e Máquina de Moore) como um modelo estrutural para a organização de hiperdocumentos instrucionais, em destacar especial, Avaliação e Exercício. [MOR 2002]".

linguagem de quarta geração	"Linguagem fácil, amigável ao usuário, que não precisa conhecer detalhes tais como mnemônicos, formatos de dados, construções complexas, e que permitam o rápido desenvolvimento de sistemas. As L4G surgiram para atender a estes dois requisitos: (1) possibilitar o desenvolvimento de aplicações pelos usuários não profissionais de informática; e (2) aumentar a produtividade na construção de sistemas [KIP 93, p. 251]". Não possuem, dessa maneira, uma vocação à solução genérica de problemas, característica das linguagens de terceira geração. "Os principais objetivos das linguagens de 4ª geração são: (1) facilitar a programação de computadores de maneira tal que usuários finais possam resolver seus problemas; (2) apressar o processo de desenvolvimento de aplicações; (3) facilitar e reduzir o custo de manutenção de aplicações; (4) minimizar problemas de depuração; e (5) gerar código sem erros a partir de requisitos de alto nível [PRI 2001, p. 3]."
linguagem script	O termo script é aplicado a linguagens fracamente tipadas, geralmente interpretadas, que possuem estruturas de dados simples que servem, tipicamente, como meio de interação *(interface)* com outras linguagens [IMP 2003, MAC 2002a, p. 53]
máquina de mealy	Autômato finito modificado de forma a gerar uma palavra de saída a cada transição [MEN 2001].
máquina de moore	Autômato finito com saída que gera uma palavra de saída (que pode ser vazia) para cada estado da máquina [MEN 2001].
mídia	Termo oriundo do inglês, *media*, que por sua vez origina-se do latim, plural de *medium*. Literalmente, significa "aquilo que está entre dois pontos [TOR 42]." No que se refere à comunicação, assume o significado de um veículo que conduz a informação de um ponto para outro. Na área da educação assistida por computador, significa um recurso computacional capaz de transmitir ao ser humano informações. Em interação humano-computador, esse significado pode ser acrescido de meios de navegação em interfaces [HEL 2001].
multimídia	Termo que possui vários significados, entre eles o de integração de dois ou mais tipos de mídias, que é a semântica utilizada nesta pesquisa [HEL 2001, KAP 91].
nautilus	Linguagem acadêmica de especificação e programação baseada em objetos concorrentes, inspirada em Teoria das Categorias e com características antecipatórias. Foi criada originalmente como um exemplo de linguagem de especificação para seu domínio semântico, dos autômatos não seqüenciais. Sua funcionalidade pode ser percebida como computações de uma Rede de Petri, apresentando propriedades de composicionalidade horizontal e vertical. As construções não usuais de Nautilus, tais como reificação e agregação, foram baseadas nesse formalismo [MEN 2002b, MEN 2002, CAR 99].

orientação a objetos	Técnica de modelagem de sistemas baseada em entidades comunicantes denominadas objetos cujas estruturas de dados e comportamento são definidos e particionados (geralmente em entidades denominadas classes) com o auxílio de mecanismos de encapsulamento, herança e polimorfismo [SEB 2001].
teoria dos autômatos	Teoria que suporta a definição de máquinas computacionais denominada autômatos, que podem reunir vários conceitos, entre eles: fita, dispositivo de entrada e/ou de saída; alfabeto de símbolos de entrada e/ou saída; pilha, dispositivo de memória auxiliar; função parcial de transição, que define as operações de leitura e gravação nas fitas e o estado corrente da máquina; estados iniciais e finais; determinismo e não-determinismo; temporização.
teoria das categorias	Teoria proposta por S. Eilenberg e S. Mac Lane em 1945, trata-se de uma teoria matemática, genérica, sobre estruturas, sendo uma de suas aplicações diretas a unificação de estruturas matemáticas [MEN 2002a, p. 266, STA 2002].
tweening	Técnica de interpolação na qual um programa de animação gera automaticamente quadros intermediários (denominados *in between*) entre quadros chaves *(key frames)*. Esta técnica foi introduzida por Burtnyk e Wein em 1971 [THA 89, IMP 2003].
XHA	Acrônimo criado para abreviar Hiper-Autômato Extensível (eXtensible Hyper-Automaton), dissertação de mestrado, cujo objetivo geral é "...estudar as possibilidades de flexibilização da função de saída do Sistema Hyper-Automaton além das rígidas possibilidades utilizadas atualmente com a utilização direta do HTML [MAC 2002]".

Bibliografia

[ACC 2002] ACCORSI, F. **Animação Bidimentional para World Wide Web Baseada em Autômatos Finitos.** 2002. 113p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[ALU 94] ALUR, R.; DILL, D. L. A Theory of Timed Automata. **Theoretical Computer Science**, Amsterdam, v.126, p. 183-235, 1994.

[AMB 2000] AMBLER, S. **The Design of Robust Persistence Layer For Relational Databases.** Disponível em:

<http://www.ambysoft.com/persistenceLayer.pdf>. Acesso em: jun. 2001.

[AMB 2000a] AMBLER, S. **Writing Robust Java Code.** Disponível em:

<www.AmbySoft.com/javaCodingStandards.pdf>. Acesso em: jun. 2001.

[AMB 2000b] AMBLER, St. **Mapping Objects to Relational Databases.** Disponível em: <http://www.AmbySoft.com/mappingObjects.pdf>. Acesso em: jun. 2001.

[APA 2002] APACHE FOUNDATION. Apache XML Project. **Crimson Home Page**. Disponível em: <http://xml.apache.org/crimson>. Acesso em: dez. 2002.

[BRO 2002] BROWN, P. **Peter Brown's Home Page.** Canterbury: University of Kent. Disponível em: <http://www.cs.ukc.ac.uk/people/staff/pjb>. Acesso em: jul. 2002.

[BUS 45] BUSH, V. As We May Think. **Atlantic Monthly**, [S.l.], p.101-108, 1945.

[CAM 88] CAMPBELL, B.; GOODMAN, J.M. HAM: A General Purpose Hypertext Abstract Machine. **Communications of ACM**, New York, v.31, n.7, p.856-861, July 1988.

[CAR 99] CARNEIRO, C. R. J. B. et al. Especificação Formal de uma Ferramenta de Trabalho Colaborativo através da Composição de Objetos Nautilus. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, SBES, 13., 1999. Florianópolis. **Anais...** Florianópolis: UFSC, 1999. p. 95-110.

[CER 74] CERF, V.; KAHN, R. E. A Protocol for Packet Network Intercommunication. **IEEE Transactions on Comunications**, [S.l.], v. COM-22, n. 5, 1974.

[CER 97] CERN. **An overview of the World-Wide Web.** Disponível em:

<http://public.web.cern.ch/Public/ACHIEVEMENTS/WEB/Welcome.html>. Acesso em: jul. 2002

[CON 87] CONKLIN, J. **A Survey of Hypertext.** Austin, Texas: Microelectronics and Computer Technology Corporation, 1987. (Technical Report STP-356-86).

[CPD 2002] CPDEE Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica. **Polilinha.** Disponível em:

<http://www.cpdee.ufmg.br/~gopac/gsm/DocOnLine/Modeling/Primitiv/Primitive2D/Polilinha.htm>. Acesso em: dez. 2002.

[DAV 95] DAVIS, H. To Embed or Not To Embed**... Communications of ACM,** New York, v.38, n. 8, 1995.

[DAV 96] DAVIS, S. R. **Learn Java Now.** Redmond: Microsoft Press, 1996.

[DEB 2002] DE BRA, P. M. E. **Hypermedia Structures and Systems.** Einhoven: EUT. Disponível em: <http://wwwis.win.tue.nl/2L670/static> Acesso em: jul. 2002.

[DIV 2002] DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação:** Máquinas Universais e Computabilidade**.** Porto Alegre: Sagra Luzzato – UFRGS, 2002. 205p.

[DOE 91] DOEGE, G. M. **Hipertexto:** Um Protótipo Implementado em ZIM. 1991. 148p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[FER 96] FERREIRA, A. B. de H. **Dicionário Eletrônico Aurélio:** v. 2.0. Rio de Janeiro: Nova Fronteira, 1996. 1 CD-ROM.

[FLY 2003] FLYNN, P. **The XML FAQ:** v. 3.0 (2000-01-01). University College Cork. Disponível em: <http://www.ucc.ie/xml/faq_a4.pdf>. Acesso em: jan. 2003.

[FOL 90] FOLEY, J. et al. **Computer Graphics:** Principles and Practice. USA: Addison-Wesley, 1990. 1174 p.

[FRI 96] FRITSCH, E. F. STI - Sistema para Treinamento de Intervalos. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO E MÚSICA, 3., 1996, Recife. **Anais...** Recife: UFPE, 1996. p. 45-55.

[FRI 98] FRITSCH, E. F. et al. Desenvolvimento de Software Educacional para a Música: STR - Sistema de Treinamento Rítmico. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO E MÚSICA, SBCEM, 5., 1998. Belo Horizonte. **Anais...** Belo Horizonte: SBC, 1998. p.209-218.

[FUR 90] FURUTA R.; STOTTS, P. D. The Trellis Hypertext Reference Model. In: NIST HYPERTEXT STANDARDIZATION WORKSHOP, 1990. **Proceedings...** [S.l.: s.n.], 1990, p. 83-93.

[GAM 2000] GAMMA, E. et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000. 364p.

[GRA 2000] GRANDI, R. H.; VICCARI, R. M.; MENEZES, P. F. B. A Database Structure for Didactic Exercises on Distance Learning Programs Adapted to ISO/IEC 9126 Standard. In: INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION DEVELOPMENT, 2000, São Paulo. **Cooperative Network for Engineering and Computer Education Development:** Proceedings. São Paulo: SENAC/SP, 2000.

[GRA 2003] GRANDI, R. H.; MENEZES, P. F. B. Hiper-Animações - Teoria Hipermídia Aplicada em Animações. In: INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION DEVELOPMENT, 2003, Santos. **Engineering Education in the World of no Frontiers:** Proceedings. Santos: COPEC, 2003.

[GRA 2003a] GRANDI, R. H.; MENEZES, P. F. B. Astrha/E - Ambiente Java/XML que Implementa Hiper-Animações Estruturadas por Máquinas de Mealy. In: WORKSHOP SOBRE SOFTWARE LIVRE, WSL, 4., 2003. **Anais...** Porto Alegre: SBC, 2003. p. 83-86.

[GRA 2003b] GRANDI, R. H.; MENEZES, P. F. B. Utilização do Ambiente Astrha para Implementar um Dicionário de Acordes Baseado em Autômatos Finitos. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO E MÚSICA, SBCM, 9., 2003. **Waiting for Proceedings.** Campinas: SBC, 2003.

[HAL 90] HALASZ, F.; SCHWARTZ, M. The Dexter Hypertext Reference Model. In: NIST HYPERTEXT STANDARDIZATION WORKSHOP, 1990. **Proceedings...** [S.l.: s.n.], 1990, p. 95-133.

[HEL 2001] HELLER, R. S. et al. Using a Theoretical Multimedia Taxonomy Framework. **ACM Journal of Education Resources in Computing**, New York, v.1, n.1, 2001.

[HEL 95] HELLER, R. S.; MARTIN, C.D. A Media Taxonomy**. IEEE MultiMedia**, [S.l.], v.2, n.4, p.36-45, Winter 1995.

[HEN 98] HENTSCHKE, L.; VICCARI, R. M.; KRÜGER, S. E.; FRITSCH, E. F.; FLORES, L. V.; GRANDI, R. H.; SANTOS, T. R. A Interdisciplinaridade na Criação de Software de Educação Musical. In: ENCONTRO NACIONAL DA ABEM., 7., 1998. **Anais...** Recife: ABEM, 1998.

[IMP 2002] IMPERIAL COLLEGE OF LONDON. Department of Computing. **FOLDOC Free On-line Dictionary of Computing**. Disponível em: <http://foldoc.doc.ic.ac.uk/foldoc>. Acesso em: jan. 2002.

[ISO 86] ISO (International Organization for Standardization). **ISO 8879:** Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). Geneva, 1986.

[ISO 87] ISO (International Organization for Standardization). **ISO 8859-1:** Information Processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1. Geneva, 1987.

[ISO 91] ISO (International Organization for Standardization). **ISO/IEC 9126:** International Standard: Information Technology – Software Product Evaluation. Geneva, 1991. 13p.

[ISO 93] ISO (International Organization for Standardization). **ISO/IEC 10646:** Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. Geneva, 1993.

[JDO 2002] JDOM ORGANIZATON. **JDOM**. Disponível em: <http://www.jdom.org>. Acesso em: maio 2002.

[KAP 90] KAPPE, F. M.; MAURER, H. Animation in Hyper-G – An Outline. In: FUTURE TRENDS IN INFORMATION TECHNOLOGY, 1990, Salzburg. **Proceedings...** Salzburg: Austrian Computer Society, 1990. p. 235-248.

[KAP 91] KAPPE, F. M. **Aspects of a Modern Multi-Media Information System.** 1991. 155p. PhD. Thesis – Institute for Foundations of Information Processing and Computer Supported New Media (IICM), Graz University of Technology, Graz.

[KIP 93] KIPPER, E. F. et al. **Engenharia de Informações:** Conceitos, Técnicas e Métodos. Porto Alegre: Sagra-Luzzatto, 1993. 332p.

[KRÜ 99] KRÜGER, S. E.; FRITSCH, E. F.; FLORES, L. V.; GRANDI, R. H.; SANTOS, T. R.; HENTSCHKE, L.; VICCARI, R. M. Developing a Software for Music Education: An Interdisciplinary Project. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 9., 1999, Rio de Janeiro. **Anais...** Rio de Janeiro: EntreLugar, 1999. p.251-264.

[LAN 89] LANSDOWN, J. et al. **Computers in Art, Design and Animation.** New York: Springer-Verlag, 1989.

[LAR 2000] LARMAN, C. **Utilizando UML e Padrões:** Uma introdução à Análise e ao Projeto Orientados a Objetos. Porto Alegre: Bookman, 2000.

[LIV 2002] THE LIVING Internet. Disponível em: <http://www.livinginternet.com>. Acesso em: jul. 2002

[MAC 2000] MACHADO, J. H. A. P. **Hyper-Automaton:** Hipertextos e Cursos na Web Usando Autômatos Finitos com Saída. 2000. 149p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MAC 2000a] MACHADO, C. C. et al. Utilização do XML no Sistema Hyper-Automaton. In: INTERNATIONAL SYMPOSIUM ON KNOWLEDGE MANAGEMENT/DOCUMENT MANAGEMENT, ISKM/DM, 2000. **Proceedings...** Curitiba: Ed. Universitária Champagnat, 2000. p. 439-455.

[MAC 2002] MACHADO, C. C. **XHA:** eXtensible Hyper-Automaton. 2002. 142p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MAC 2002a] MACROMEDIA. **Flash MX Tutorials.** São Francisco: Macromedia, 2002. 90 p.

[MAC 2003] MACROMEDIA. **Macromedia Flash Development Center.** Disponível em: <http://www.macromedia.com/devnet/mx/flash>. Acesso em: jan. 2003.

[MAG 2002] MAGALHÃES, G. C. de. **AGA Player:** Animação 2D Baseada em Autômatos para a Web**.** 2002. 43p. Projeto de Diplomação (Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BRA 2001] BRASIL. Ministério da Ciência e da Tecnologia. **Qualidade e Produtividade no Setor de Software.** 2001. Disponível em: <http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm>. Acesso em: nov. 2002.

[MEN 2001] MENEZES, P. F. B. **Linguagens Formais e Autômatos.** 4.ed. Porto Alegre: Sagra Luzzato, 2001.

[MEN 2002] MENEZES, P. F. B. et al. **Hyper Seed – Framework, Ferramentas e Métodos para Sistemas Hipermídia voltados para EAD via WWW.** Projeto de Pesquisa proposto pelo Instituto de Informática da Universidade Federal do Rio Grande do Sul, aprovado e apoiado pelo CNPq. 2002. 16 p.

[MEN 2002a] MENEZES, P. F. B.; HAUSLER, E. H. **Teoria das Categorias para Ciência da Computação.** Porto Alegre: Sagra Luzzato – UFRGS, 2002. 324 p.

[MEN 2002b] MENEZES, P. F. B. et al. Nautilus: A Concurrent Anticipatory Programming Language. In: INTERNATIONAL CONFERENCE ON COMPUTING ANTECIPATORY SYSTEMS, CASYS, 5., 2001. **Proceedings...** Melville: American Institute of Physics, 2002. v. 627, p. 553-564.

[MEN 2003] MENEZES, P. F. B. et al. **Laboratório de Fundamentos da Computação.** Instituto de Informática da Universidade Federal do Rio Grande do Sul. Disponível em: <http://teia.inf.ufrgs.br>. Acesso em: jan. 2003.

[MEY 2000] MEYER, M. et al. **Nosso Futuro e o Computador.** 3. ed. Porto Alegre: Bookman, 2000. 599 p.

[MIT 2002] MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **League for Programming Freedom.** Disponível em: <http://lpf.ai.mit.edu>. Acesso em: nov. 2002.

[MOR 2002] MORAIS, C. T. Q. de. **Hyper-Automaton:** Avaliação Interativa de Alunos em Cursos na WEB Baseado em Autômatos Finitos. 2002. 111p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MOZ 2002] MOZILLA ORGANIZATION. **Mozilla Organization Home Page.** Disponível em: <http://www.mozilla.org>. Acesso em: jul. 2002.

[MSN 2002] MSN Messenger for Windows. Disponível em: <http://messenger.msn.com.br>. Acesso em: dez. 2002.

[NAS 2002] NASA. **The Appollo Program.** Disponível em: <http://www.hq.nasa.gov/office/pao/History/apollo.html>. Acesso em: jul. 2002.

[NEL 65] NELSON, T. H. A File Structure for the Complex, the Changing and the Indeterminate. In: ACM NATIONAL CONFERENCE, 20., 1965, **Proceedings…** New York: ACM, 1965. p.84-100.

[NEL 67] NELSON, T. H. Getting It Out of Our System. In: **Information Retrieval:** A Critical Review. Washington D.C.: Thompson Books , 1967. p. 191-210.

[NEL 87] NELSON, T. H. All or One and One for All. In: HYPERTEXT PAPERS. 1987, Chapel Hill. **Proceedings...** Chapel Hill: The University of North Carolina, 1987. p. v-vii.

[NIE 90] NIELSEN, J. The Art of Navigating through Hypertext. **Communications of the ACM**, New York, p. 296-310, Mar. 1990.

[NYC 91] NYCE, J.; KAHN; P. **From Memex to Hypertext**. Boston: Academic Press, 1991. 380p.

[PRI 2001] PRICE, A. M. de A.; TOSCANI, S. S. **Implementação de Linguagens de Programação:** Compiladores. 2. ed. Porto Alegre: Sagra Luzzato, 2001.

[PUE 88] PUEYO, X.; TOST, D. A Survey of Computer Animation.**Computer Graphics Forum**, Amsterdam, v.7, p. 281-300, 1988.

[RAD 91] RADA, R. **Hypertext:** From Text to Expertext**.** London: McGraw-Hill, 1991.

[RAO 90] RAO, U.; TUROFF, M. Hypertext functionality: A Theoretical Framework. **International Journal of Human-Computer Interaction**, [S.l.], v. 2, n. 4, p. 333-357, 1990.

[RED 97] REDSHAW, K. **Vannevar Bush (1890-1974).** Disponível em: <http://kerryr.net/pioneers/bush.htm>. Acesso em: jul. 2002.

[SAL 92] SALGADO, A. C. et al. **Sistemas Hipermídia:** Hipertexto e Banco de Dados. Porto Alegre: Instituto de Informática da UFRGS, 1992

[SCH 87] SCHWARTZ, M.; DELISLE, N. Contexts - A Partitioning Concept for Hypertext**. ACM Transactions on Office Information Systems**, New York, v.5, n.2, p. 168-186, Apr. 1987.

[SCH 89] SHNEIDERMAN, B., KEARSLEY, G. P. **Hypertext Hands-On!** An Introduction to a New Way of Organizing and Accessing Information. New York: Addison Wesley, 1989.

[SEB 2001] SEBESTA, R. W. **Concepts of Programming Languages.** 5th ed. Boston: Addison-Wesley, 2001.

[SHA 80] SHAW, A. C. **A Model for Document Preparation Systems**. Seattle: University of Washington, Dept. of Computer Science, 1980.

[STA 2002] STANFORD UNIVERSITY. **Stanford Encyclopedia of Philosophy.** Disponível em: <http://plato.stanford.edu/archives/fall1997/entries/category-theory> Acesso em: jul. 2002

[STA 2002a] STANFORD UNIVERSITY. **Mouse Site:** The Demo**.** Disponível em: <http://sloan.stanford.edu/MouseSite/1968Demo.html>. Acesso em: jul. 2002

[STO 2002] STOTTS, P. D. **David Stotts Home Page.** Disponível em: <http://www.cs.unc.edu/~stotts>. Acesso em: jul. 2002.

[STO 89] STOTTS, P. D.; FURUTA, R. Petri-net-based Hypertext: Document Structure with Browsing Semantics. **ACM Transactions on Information Systems**, New York, v. 7, n. 1, p. 3-29, Jan. 1989.

[SUN 2002] SUN MICROSYSTEMS. **Applets.** Disponível em: <http://java.sun.com/applets>. Acesso em: nov. 2002.

[SUN 2002a] SUN MICROSYSTEMS. **How to Make Applets.** Disponível em: <http://java.sun.com/docs/books/tutorial/uiswing/components/applet.html>. Acesso em: nov. 2002.

[SUN 2002b] SUN MICROSYSTEMS. **Java™ API for XML Processing (JAXP).** Disponível em: <http://java.sun.com/xml/jaxp>. Acesso em: dez. 2002.

[THA 85] THALMANN, N. M.; THALMANN, D. **Computer Animation:** Theory and Practice. Tokyo: Springer-Verlag, 1985. 239 p.

[THA 89] THALMAN, D. Motion Control: From Keyframe to Task-Level Animation. In: COMPUTER ANIMATION, 1989, Geneva. **State-of-the-art in Computer Animation**: Proceedings. Tokyo: Springer-Verlag, 1989. p.3-14.

[THA 91] THALMAN, N. M; THALMAN, D. **New Trends in Animation and Visualization.** New York: John Wiley, 1991.

[TOR 42] TORRINHA, F. **Dicionário Latino Português.** 4. ed. Porto: Gráficos Reunidos, 1942. 947 p.

[UOV 2002] UNIVERSITY OF VIRGINIA. **The Electronic Labyrinth.** Disponível em: <http://www.iath.virginia.edu/elab/hfl0267.html>. Acesso em: jul. 2002.

[W3C 2001] W3C. **HTML Working Group Charter.** Disponível em: <http://www.w3c.org/2002/05/html/charter> Acesso em: jan. 2001.

[W3C 2002] W3C. **BNF for specific URL schemes.** Disponível em: <http://www.w3.org/Addressing/URL/5_BNF.html> Acesso em: jul. 2002.

[W3C 2002a] W3C. **Extensible Markup Language (XML) 1.0 (Second Edition).** Oct. 2000. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: dez. 2002.

[W3C 2003] W3C. **Scalable Vector Graphics (SVG) 1.1 Specification**. Jan. 2003. Disponível em: <http://www.w3.org/TR/2003/REC-SVG11-20030114>. Acesso em: jan. 2003.

[W3C 2003a] W3C. **XML Linking Language (XLink) Version 1.0**. Disponível em: <http://www.w3.org/TR/XLink>. Acesso em: jan. 2003.

[W3C 2003b] W3C. **Synchronized Multimedia Integration Language (SMIL) 1.0 Specification.** June 1998. Disponível em: <http://www.w3.org/TR/REC-smil>. Acesso em: jan. 2003.

[W3C 2003c] W3C. **SMIL Animation.** Sept. 2001. Disponível em: <http://www.w3.org/TR/2001/REC-smil-animation-20010904>. Acesso em: jan. 2003.

[W3C 2003d] W3C. **Adding a Touch of Style.** Disponível em: <http://www.w3.org/MarkUp/Guide/Style>. Acesso em: jan. 2003.

[WAL 87] WALKER, J.H. Document Examiner: Delivery interface for hypertext documents. In: ACM HYPERTEXT CONFERENCE ON HYPERTEST, 1., 1987, Chapel Hill. **Proceedings...** Chapel Hill: ACM, 1987. p. 307-323.

[WIK 2002] WIKIPEDIA. **The Free Encyclopedia.** Disponível em: <http://www.wikipedia.com>. Acesso em: jul. 2002

[XAN 2001] XANADU, Project. **Deep Hypertext**: The Xanadu Model. Disponível em: <http://www.xanadu.com/xuTheModel >. Acesso em: jul. 2002.

[XAN 99] XANADU , Project. **Project Xanady History.** Disponível em: <http://www.xanadu.com/xuhistory.html>. Acesso em: jul. 2002.

[ZAK 2002] ZAKON, R. H. **Hobbes' Internet Timeline.** v. 5.6. Disponível em: <http://www.zakon.org/robert/internet/timeline>. Acesso em: jul. 2002.