2
Trabalhos Relacionados

Este capítulo lista as ferramentas de autoria de documentos hipermídia e/ou multimídia em que este trabalho se baseou, destacando as suas limitações, soluções interessantes e possíveis mudanças para cada uma delas.

2.1
Ferramentas de autoria

Podemos encontrar diversas ferramentas de autoria de documentos hipermídia e multimídia na literatura, como a *M-Tool* (Rousseau et al., 2005), a *IMAT* (Kim & An, 1999) e a *2-D MPEG-4 Auhoring Tool* (Viljoen et al., 2003). Reduzindo o foco para somente a TV Digital, verificamos que também existe uma quantidade considerável, alguns exemplos são encontrados em (Chiao et al., 2006) e (Oliveira et al., 2008). Nesta seção, apresentamos e analisamos as que influenciaram fortemente a direção seguida por este trabalho.

Inicialmente, descrevemos o *Composer* (Guimaraes & Soares, 2007), ferramenta voltada para o padrão brasileiro de TV Digital. A seguir tratamos da *Icareus iTV Suite Author* (Icareus Author, 2007) criada para o padrão europeu de TV Digital, o qual utiliza o *middleware* MHP (*Multimedia Home Platform* - ES 201 812 (ETSI, 2003)). Por último, apresentamos as ferramentas *Adobe Director* (Adobe Director, 2009) e *Adobe Flash* (Adobe Flash, 2008). Mesmo essas últimas não sendo direcionadas à TV Digital, foram escolhidas devido ao amplo uso por não programadores que produzem aplicações hipermídias interativas.

2.1.1
Composer

O *Composer* é um ambiente de autoria para a criação de programas interativos para TV Digital do *middleware* Ginga-NCL. Foi desenvolvido pelo laboratório TeleMídia da PUC-Rio. O modelo de autoria utilizado é o NCM (*Nested Context Model*) (Soares et al., 2003), no qual a NCL se baseia. No modelo NCM, um documento hipermídia pode ser representado por um nó de

contexto, que contém outros nós de contexto ou de mídia, e elos conectando esses nós.

Os arquivos de entrada e saída existentes no Composer são: projeto (.hyp) e aplicação NCL (.ncl), sendo que o projeto é a estrutura raiz da ferramenta que contém todas as informações necessárias para continuar a edição em execuções futuras. O projeto possui também uma ou várias aplicações NCL. Com isso, a ferramenta oferece ao autor a possibilidade de editar diversas aplicações ao mesmo tempo.

As diferentes visões disponibilizadas auxiliam na criação e alteração de uma aplicação. Na Figura 2.1 podemos visualizar todas elas: a estrutural (região 1), a temporal (região 2), a de leiaute (região 3) e a textual (região 4)

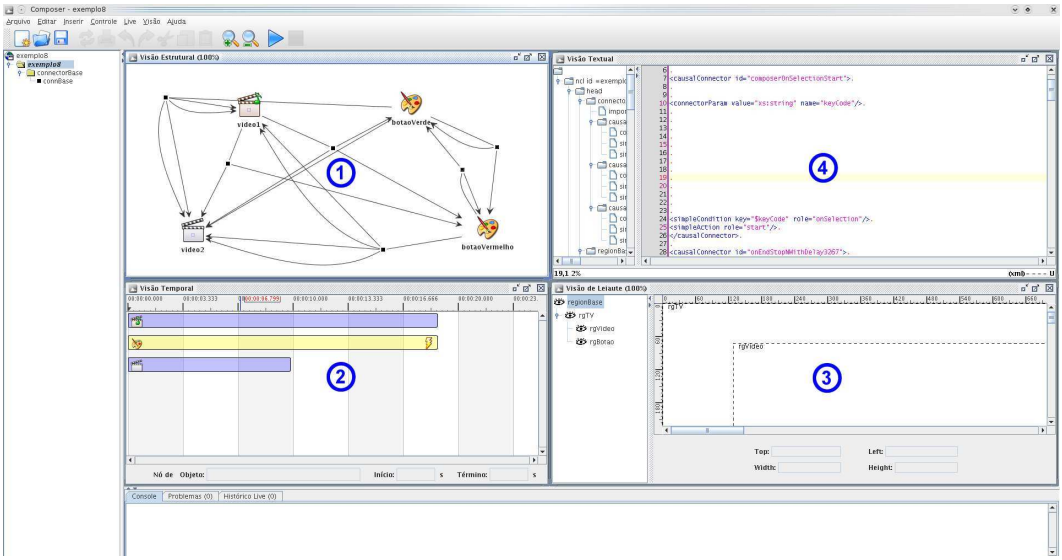

Figura 2.1: Visões do Composer

A visão estrutural é responsável por apresentar e editar nós e elos de uma aplicação NCL. É possível inserir um novo nó de mídia ou de contexto diretamente pela visão. Também podemos criar um elo entre dois ou mais desses nós. Vale lembrar que, na NCL, os elos que são responsáveis por definir o sincronismo espaço-temporal e a interatividade entre os objetos de uma aplicação. Dessa forma, vemos que o autor precisa saber NCL para criar sincronismo e interatividade na sua aplicação.

Essa visão é muito útil por apresentar a estrutura completa da aplicação NCL. Entretanto, aplicações com muitos elos em um mesmo contexto fazem com que o autor fique perdido com os elos se sobrepondo. A opção que habilita a exibição do nome de cada elo só provoca uma confusão ainda maior, pois os nomes se sobrepõem e os elos ficam difíceis de enxergar com tanta informação na tela. Se o autor souber utilizar corretamente os nós de

contexto, é possível minimizar consideravelmente o problema. Vale destacar que a filtragem olho-de-peixe também ajudaria a diminuir esse problema, entretanto não a encontramos no Composer, somente em seu antecessor, o HyperProp (Coelho & Soares, 2004).

A visão temporal exibe uma linha do tempo com os nós de mídia e os elos entre eles. O início e a duração de cada mídia ficam bem claros, mas os elos, mais uma vez, ficam difíceis de visualizar e de entender as suas ações.

O Composer suporta, pela visão temporal, a inclusão de novos relacionamentos temporais e eventos interativos. Esses nada mais são que elos entre as mídias e a mudança de nomenclatura pode fazer com que o autor se confunda. A Figura 2.2 apresenta a visão estrutural e a temporal antes e depois da criação de um relacionamento temporal. Podemos perceber que o relacionamento temporal é apresentado como um elo (item 1 da figura) na visão estrutural.

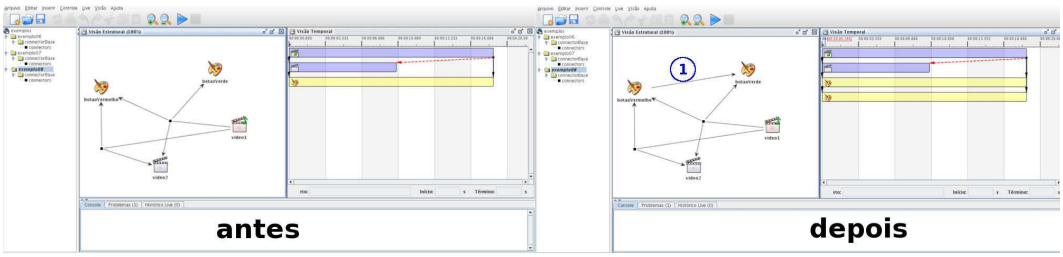

Figura 2.2: Criação de um relacionamento temporal

A visão temporal também é responsável pela simulação da interatividade da aplicação NCL. A mídia que possui interatividade é apresentada com um ícone de raio (item 1 da Figura 2.3) e para simular a interação basta que o usuário clique com o botão direito do *mouse* na mídia e escolha a tecla do controle remoto que deseja acionar. Esse acionamento executa a interação e a visão temporal passa a exibir todas as mídias relacionadas.

Entretanto, essa simulação acaba prejudicando o entendimento da visão temporal. Isso acontece porque alguns elementos presentes na visão estrutural parecem não existir na temporal, deixando o usuário confuso sobre as características temporais dos elementos que não são exibidos. A Figura 2.3 nos fornece um exemplo desse problema através da exibição das visões estrutural e temporal de uma aplicação com interatividade. Nesse exemplo temos a impressão de que a mídia "video2" (item 2) não existe na visão temporal.

A seguir, temos a visão de leiaute onde o autor pode visualizar e editar as regiões onde as mídias aparecem durante a execução da aplicação. A área à esquerda na visão de leiaute apresenta a hierarquia das regiões e nela selecionamos quais regiões estão ou não sendo exibidas na tela.

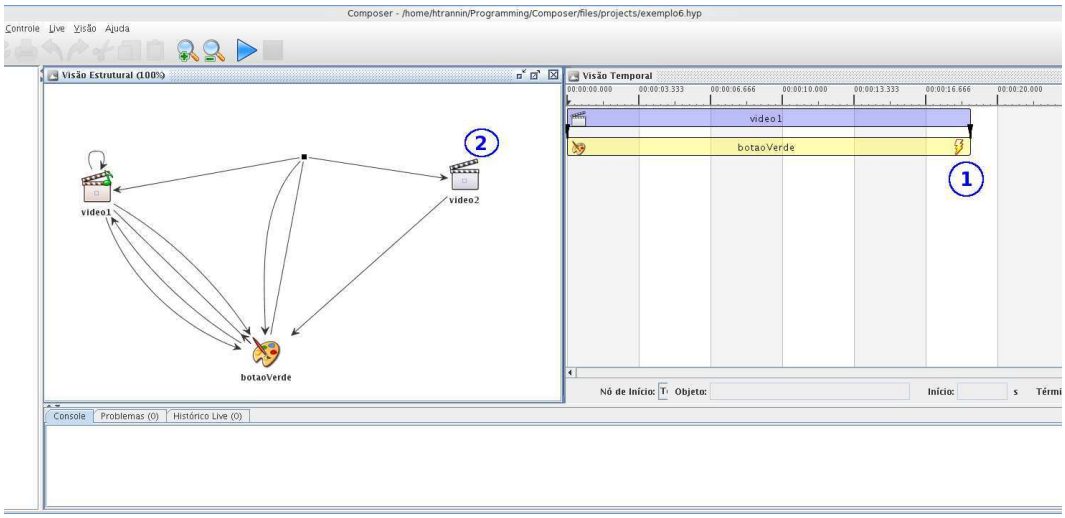

Figura 2.3: Problema na simulação da interatividade

Um problema é a falta de uma visão espacial. Com ela seria possível ter uma sincronização com a visão temporal para que fossem apresentadas somente as mídias visíveis em um determinado momento no tempo.

Na visão textual temos total liberdade para editar diretamente a aplicação NCL. Entretanto, alguns problemas são encontrados: a edição pelas demais visões faz com que seja inserido um espaçamento muito grande entre os elementos, não possui auxílio para completar automaticamente um elemento sendo digitado e a detecção de edição errada não apresenta, em diversos momentos, conteúdo suficiente para que seja possível verificar e corrigir o erro. Vale a pena destacar a existência de uma outra ferramenta, chamada NCL Eclipse (NCLEclipse, 2008), que disponibiliza um ambiente de programação bem completo. Ela é voltada para o programador, completa automaticamente os elementos digitados e ainda realiza validações na aplicação desenvolvida.

Finalmente, vale destacar que a edição das propriedades de mídias e elos é realizada através de janelas modais que mudam de acordo com a visão utilizada no momento. Essa abordagem é pouco eficiente uma vez que é preciso diversas interações do usuário para cada elemento editado e/ou visualizado.

A análise do Composer deixa claro que essa ferramenta de autoria é um importante instrumento para auxiliar na criação de aplicações NCL. Entretanto, ele não consegue abstrair a necessidade de conhecimento da NCL, uma vez que a interface de usuário reflete diretamente o modelo NCM. Poderíamos ir além e afirmar que o Composer é a maneira gráfica de se programar em NCL. Além disso, uma visão espacial faz muita falta. O autor precisa executar a aplicação NCL toda vez que deseja visualizar os resultados obtidos pela edição, se a visão espacial existisse diminuiria esse problema.

Dentre as características encontradas no Composer, podemos destacar

as visões temporal e textual. Estss são importantes componentes para uma ferramenta de autoria para TV Digital voltada para programadores. Também destacamos a visão estrutural, pois ao utilizar o NCM como modelo, este paradigma mostrou-se muito útil para o autor visualizar a organização da aplicação NCL.

Finalmente, algumas possíveis melhorias no Composer seriam a substituição da visão de leiaute pela visão espacial e a retirada da simulação da interatividade da visão temporal.

2.1.2 Icareus iTV Suite Author

O *Icareus iTV Suite Author* é uma ferramenta de autoria para desenvolvimento de conteúdo para TV Digital do *middleware* MHP. Foi inicialmente desenvolvida pela *Cardinal Systems* e era chamada de *Cardinal Studio*. Atualmente é mantida pela equipe de desenvolvimento da *Icareus*.

O modelo de autoria foi baseado em algumas outras ferramentas populares, como o *Adobe Flash* (maiores detalhes na Seção 2.1.5). Basicamente, tudo gira em torno de cenas (*Scene*). Cada uma possui um conteúdo próprio (imagens, textos, botões, etc.) que será apresentado no momento de sua execução e é possível ir de uma cena a outra através da inclusão de interatividade.

Existe um único arquivo de entrada e saída que é o projeto (.nkr). Assim como no *Composer* da Seção 2.1.1, o projeto é a estrutura raiz da ferramenta e contém todas as informações necessárias para continuar a edição em execuções futuras, além disso também possui todas as mídias utilizadas.

A Figura 2.4 nos fornece uma visão da interface gráfica do *Icareus iTV*. A região 1 é o componente *View* que implementa a visão espacial; a região 2 nos mostra o *Scene Browser* que é o visualizador de todas as cenas existentes; a região 3 é o componente *Tools* onde se encontram os componente da interface gráfica do usuário, denominados *widgets*, que podem ser usadas na cena, como botões, imagens e formulários; vemos ainda a região 4 apresentando o componente *Properties* que exibe as propriedades específicas do elemento (cena, imagem, botão, etc.) selecionado e, por fim, temos a região 5 com o *Resource List* contendo todas as imagens usadas no projeto.

É importante salientar que o *Icareus* tem como foco os autores não programadores. Por isso, nenhum componente para edição direta do código fonte do documento hipermídia é disponibilizado. Porém, é possível inserir componentes externos para adicionar funcionalidades não oferecidas pelo pacote básico. Esses componentes são códigos Java importados para o *Icareus*. Do ponto de vista do autor, os componentes externos são idênticos aos existentes

Figura 2.4: Interface do Icareus iTV Suite Author

inicialmente na ferramenta.

Além disso, também devemos destacar que o foco da autoria se dá na interatividade e, por isso, não é encontrada a interface de linha do tempo para sincronização temporal. Inclusive, o único vídeo considerado é o que será transmitido pelo canal de televisão durante a transmissão. A aplicação criada pelo *Icareus iTV Suite Author* é somente uma interface gráfica interativa que fica sobre o vídeo sendo transmitido. A Figura 2.5 nos fornece um bom exemplo de uma aplicação gerada pelo *Icareus*.

Figura 2.5: Aplicação exemplo do Icareus iTV Suite Author

A visão espacial utiliza técnicas de WYSIWYG[1]. Podemos arrastar os

[1]http://www.askoxford.com/concise_oed/wysiwyg?view=uk+

elementos do componente *Tools* e soltá-los no *View* para criar novas imagens, botões, etc. Ainda é possível redimensioná-los e posicioná-los nos lugares mais adequados.

Conforme já comentado, o foco do *Icareus* é a interatividade e, neste quesito, a ferramenta realmente realiza um bom trabalho. Elementos que podem ser acionados, como botões, formulários e *menus*, têm duas importantes propriedades: ação (*Action*) e tecla (*Key*). A ação descreve o que será executado quando o elemento for acionado e a tecla é a tecla do controle remoto que deve ser pressionada para executar a ação associada. Cada ação escolhida necessita que o autor preencha alguns parâmetros específicos, um exemplo é a ação de mudar de cena que precisa inserir a próxima cena. Quanto à tecla, são disponibilizadas todas as teclas de um controle remoto padrão para escolha.

Alguns componentes são baseados em ferramentas populares, como é o caso do *Tools* e do *Properties*, e eles se mostram muito úteis no auxílio à edição de um projeto no *Icareus*. O *Tools* agiliza a inclusão de novos elementos gráficos e o *Properties* é apresentado de acordo com o elemento selecionado e permite uma rápida alteração das suas propriedades. Também temos o componente *Resource List*, que serve como uma biblioteca das imagens usadas no projeto.

O *Icareus iTV Suite Author* é uma ferramenta muito eficiente se o autor deseja se focar na interatividade. Caso queira alterar a sincronização espaço-temporal de mídias como vídeo, áudio e imagens, é preciso procurar outra ferramenta. Também percebemos que a aplicação básica do *Icareus* apresenta poucos *widgets* no componente *Tools*. A inclusão de componentes externos é a única alternativa para criar aplicações mais complexas.

O grande destaque fica por conta da abstração proporcionada pelos componentes que auxiliam na inclusão de interatividade. O autor não precisa possuir conhecimento de programação para criar uma aplicação interativa. Além disso, o uso de componentes já bem definidos por outras ferramentas proporciona uma interface fácil de ser utilizada.

Finalmente, temos o problema que, durante a edição, o autor não tem conhecimento de como as diferentes cenas se relacionam. Nenhum componente disponibiliza diretamente a informação de como se dá a interatividade das cenas, o autor precisa entrar em cada uma e procurar a relação.

2.1.3 LimSee

O *LimSee2* é uma ferramenta de autoria para criação de documentos *SMIL* (SMIL 3.0, 2008). Foi desenvolvido pelo grupo WAM do instituto francês de pesquisa em ciências da computação (INRIA). O modelo de autoria é a

própria linguagem SMIL.

Os arquivos de entrada e saída do LimSee2 são documentos SMIL. Não são armazenadas informações sobre um projeto separado, todo dado necessário à execução é proveniente do próprio documento *SMIL*. Podemos definir ainda um documento xml para criar rapidamente um *slide show*.

O foco é na sincronização espaço-temporal que é editada através das visões espacial e temporal. A interatividade é obtida através da edição direta do código fonte. Oferece ainda componentes auxiliares para edição de atributos e visualização da estrutura do documento. A Figura 2.6 apresenta a interface gráfica do *LimSee2*. É possível visualizar a visão espacial na região 1, a linha do tempo é apresentada na região 2 e encontramos os componentes auxiliares na região 3 e 4.

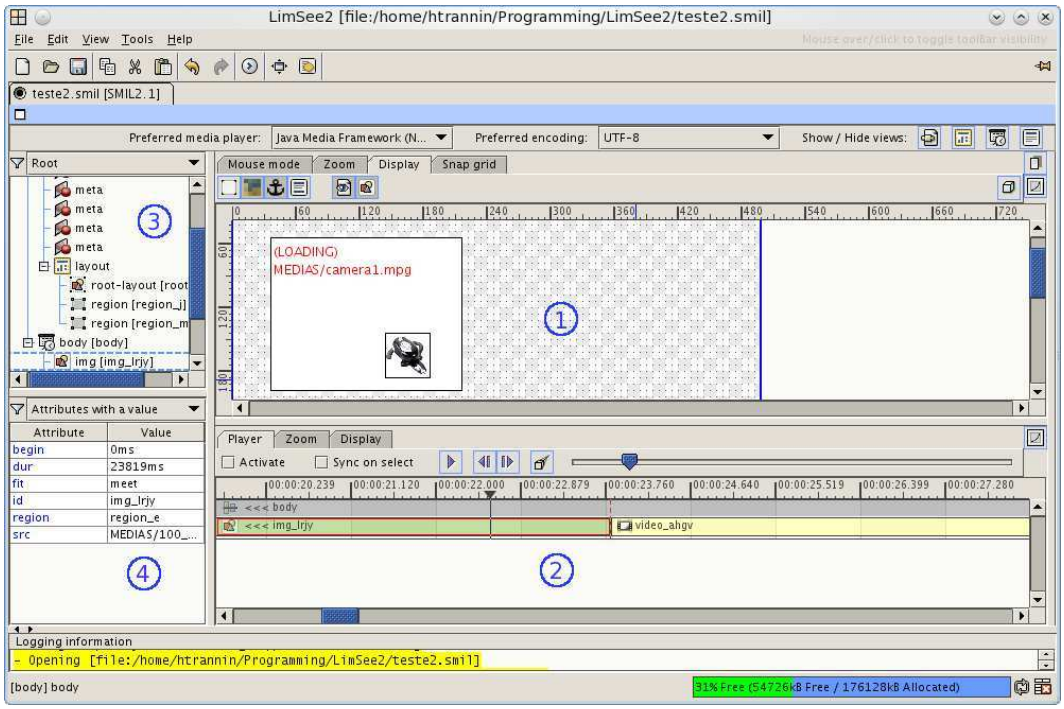

Figura 2.6: Interface do LimSee2

Através da visão espacial (região 1 da Figura 2.6) é possível posicionar as mídias no espaço onde elas serão apresentadas, além de redimensioná-las conforme o desejado. O LimSee2 oferece diversas funcionalidades que aumentam o poder de expressão da visão espacial, como é o caso do *zoom* e da grade que auxilia no posicionamento das mídias.

A visão temporal apresenta todas as mídias sendo visualizadas no tempo. Entretanto, a disposição das mídias no componente pode confundir o autor. Como é possível perceber na Figura 2.6, a visão espacial exibe duas mídias, mas na visão temporal essas mesmas mídias aparecem uma após a outra. O que provoca uma quebra na percepção da sincronização espaço-temporal.

As regiões 3 e 4 da Figura 2.6 apresentam, respectivamente, os componentes Propriedades, que exibe os atributos de uma determinada mídia, e Estrutura lógica, que disponibiliza toda a estrutura do documento *SMIL* sendo editado. Esses componentes são importantes oferecerem uma forma rápida e direta de editar todos os elementos do documento.

Dessa forma, percebemos que, assim como o Composer, o LimSee2 é uma importante ferramenta de auxílio an criação de documentos hipermídias, no caso da linguagem SMIL. Entretanto, ele não consegue abstrair a necessidade de conhecimento da linguagem.

Além disso, temos também o *LimSee3* que, apesar do nome, não é uma continuação direta do *LimSee2*. O *LimSee3* abstrai a linguagem através da utilização de *templates* e da possibilidade de editar somente a estrutura lógica do documento multimídia. Ele se apresentou como uma alternativa para o autor sem conhecimento em SMIL.

2.1.4 Adobe Director

O *Adobe Director* é uma ferramenta de autoria de aplicações multimídia criada pela *Macromedia*, agora parte da *Adobe Systems*. A versão utilizada para análise é a 11. Inicialmente, o *Director* era usado para criar animações simples. Após a inclusão da linguagem de *script Lingo* (Wikipedia Lingo, 2009), passou a ser usado para criar aplicações interativas para CD-ROM, DVD e *web*. O modelo utilizado tenta se aproximar à linguagem do cinema e teatro; a intenção é transformar o autor em um diretor da aplicação multimídia.

Existe um arquivo de entrada e saída chamado *Director Movie* (.dir) e outo denominado *Cast* (.cst). O *Director Movie* corresponde ao projeto encontrado no *Composer* (Seção 2.1.1) e no *Icareus* (Seção 2.1.2). Já o *Cast* é um arquivo que armazena as mídias para que possam ser usadas em outros projetos. Fazendo analogia ao cinema, seria como levar o elenco para fazer outros filmes.

A Figura 2.7 apresenta a interface gráfica do *Director*. Percebemos que a metáfora do cinema está amplamente presente. A visão espacial é o palco (*Stage*), a linha do tempo é o roteiro (*Score*) e as mídias são os membros do elenco (*Cast*), representados, respectivamente, pelas regiões 1, 2 e 3 da Figura 2.7. Ao final da edição, salvamos o projeto como um arquivo multimídia denominado filme (*Movie*).

Ainda na Figura 2.7, a região 4 apresenta o componente *Property Inspector* e a região 5 mostra o *Tools*. Eles se assemelham aos componentes *Properties* e *Tools* do *Icareus iTV Suite Author* da Seção 2.1.2 e oferecem os

mesmos benefícios já citados.

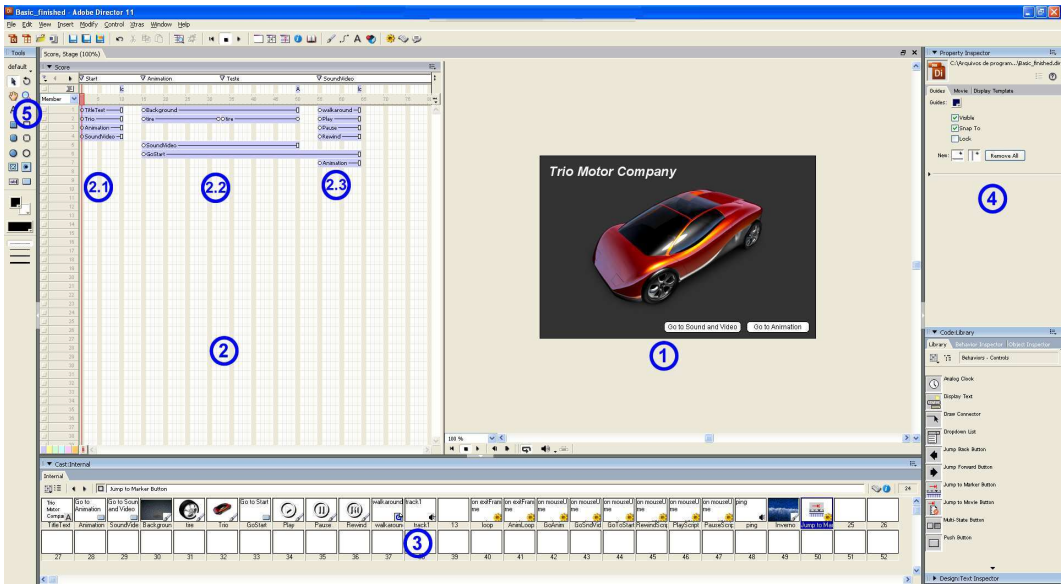

Figura 2.7: Interface do Adobe Director

O *Stage* corresponde à visão espacial. É possível colocar diretamente as mídias na posição do espaço onde elas serão apresentadas, além de redimensioná-las conforme o desejado. A simulação do documento multimídia é feita dentro da própria visão espacial e, por isso, ela retrata fielmente o resultado final obtido.

A linha do tempo do componente *Score* é separada em canais (*channels*) e quadros (*frames*). Os canais agrupam mídias de um mesmo tipo, como por exemplo temos canais para as mídias visuais (imagem, vídeo) e outros para as sonoras (áudio). O quadro é uma unidade do tempo e possuem o mesmo conceito dos quadros de um filme de cinema. Podemos incluir mídias diretamente à linha do tempo e também editar suas características temporais facilmente.

O *Score* é responsável ainda por separar as diferentes cenas do filme, conforme mostrado nas regiões 2.1, 2.2 e 2.3 da Figura 2.7. Quando a interatividade deve ser o único meio de acesso às outras cenas, o autor precisa fazer uso de *scripts* para limitar o tempo de execução de cada uma delas. Isso provoca um esforço a mais que, inicialmente, não seria necessário.

A interatividade é tratada somente pela linguagem de *script Lingo*. Entretanto, o *Director* fornece uma vasta biblioteca com diversos *scripts* pré-fabricados para que o autor não programador possa criar documentos interativos. Vale ressaltar que, na maioria dos casos, os *scripts* da biblioteca satisfazem toda a necessidade do autor. A atribuição de interação é feita simplesmente arrastando e soltando um *script* sobre uma mídia. De acordo com

o *script*, alguns valores devem ser preenchidos para instanciar corretamente a interação.

A Figura 2.8 destaca os componentes de edição direta do *script* (região 1) e da biblioteca (região 2). O componente de edição possui alguns problemas, como não completar automaticamente o código fonte e a detecção de alguns erros é feita somente durante a execução.

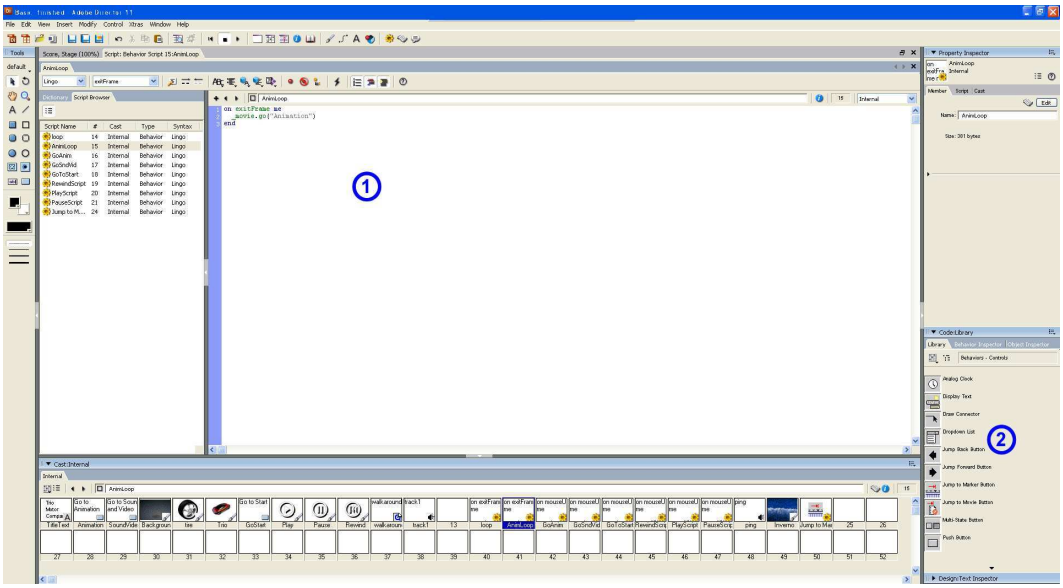

Figura 2.8: Interface de Edição de Scripts do Adobe Director

O *Adobe Director* é uma poderosa ferramenta para criação de documentos multimídia. Entretanto, o modelo com analogia ao cinema provoca uma dificuldade inicial em entender a função de cada componente da interface. Um outro problema, como já relatado, é a utilização da linha do tempo para separar as diferentes cenas do filme, acessíveis apenas através de ações do autor. Na prática, isso provoca maior trabalho para o autor e confusão no entendimento da interatividade, pois esta se mistura com o tempo.

O destaque fica por conta da sincronização espaço-temporal que atende correta e facilmente os desejos dos autores. Além disso, o agrupamento das mídias em um componente, no caso o *Cast*, agiliza consideravelmente a inclusão de mídias no *Stage* e *Score* e também facilita a edição direta das mídias, uma vez que é só realizar um duplo clique com o mouse sobre a mídia para começar a editá-la.

As possíveis alterações seriam retirar as cenas da linha do tempo e também a melhoria do ambiente de programação da linguagem *Lingo*.

2.1.5 Adobe Flash

O *Adobe Flash* é um ambiente de autoria para criação de aplicações multimídia criada pela *Macromedia* e atualmente desenvolvida e distribuída pela *Adobe Systems*. A versão analisada é a última liberada, a CS4 Professional.

O *Flash* é utilizado, principalmente, para criação de animações (ex.: *banners*), jogos e páginas *web* completas ou componentes (ex.: vídeos) para elas. O modelo de autoria foi criado exclusivamente para essa ferramenta, entretanto possui analogias ao cinema, como o *Stage* que corresponde à visão espacial, e também faz referência a conceitos comumente encontrados em interfaces gráficas atuais, como *Panels* e *Tools*.

Existe ainda a separação em cenas, assim como ocorre no *Icareus iTV Suite Author* da Seção 2.1.2. Cada cena possui conteúdo específico, podendo ser espacial, temporal ou interativo. Sem a interatividade, a exibição do vídeo começa pela primeira cena, passa para a segunda e continua assim sucessivamente.

É possível carregar e salvar diversos arquivos no *Flash*, mas vamos citar somente os mais relevantes para este trabalho. Sendo assim, temos os seguintes arquivos do projeto (.flp), o do *flash* ou *template* (.fla) e o do *ActionScript* (ActionScript, 2006) (.as). O arquivo do projeto agrupa os demais arquivos em uma hierarquia de diretórios para facilitar o entendimento de quais arquivos se relacionam. O arquivo do *flash*, chamado FLA, é o primeiro a ser trabalhado e contém as informações das mídias, da linha do tempo e dos *scripts*. O arquivo do *ActionScript*, também conhecido como AS, é utilizado para manter *scripts* fora do arquivo FLA, o que ajuda na organização do código e quando muitas pessoas trabalham no projeto.

Por fim, temos o arquivo do *template* que possui a mesma estrutura do arquivo do *flash*, mas que teoricamente é diferente. Os *Templates* do *Flash* são pré-configurados com conteúdo que pode ser customizado de acordo com o objetivo do autor. Eles oferecem uma maneira rápida e fácil de iniciar um projeto comumente desenvolvido. Os *Templates* podem ser encontrados na própria página de ajuda do *Adobe Flash* e em páginas de comunidades da ferramenta.

A Figura 2.9 destaca os principais componentes gráficos da interface. A região 1 apresenta a visão espacial ou *Stage*, na região 2 temos a linha do tempo e na região 3 aparecem os *Panels*, os quais podem ser uma biblioteca com todas as mídias, um editor das propriedades da mídia selecionada no *Stage*, dentre outros. Também podemos ver o componente *Tools* na região 4. Vale destacar que o *Tools* e o *Panel* de edição de propriedades tem as mesmas características

dos componentes das seções 2.1.2 e 2.1.4.

É importante esclarecer que o Flash possui uma grande variedade de componentes para auxiliar a edição mais avançada, mas este trabalho considera como melhor alternativa manter o foco nos paradigmas e em alguns componentes genéricos úteis a uma ferramenta de autoria para TV Digital.

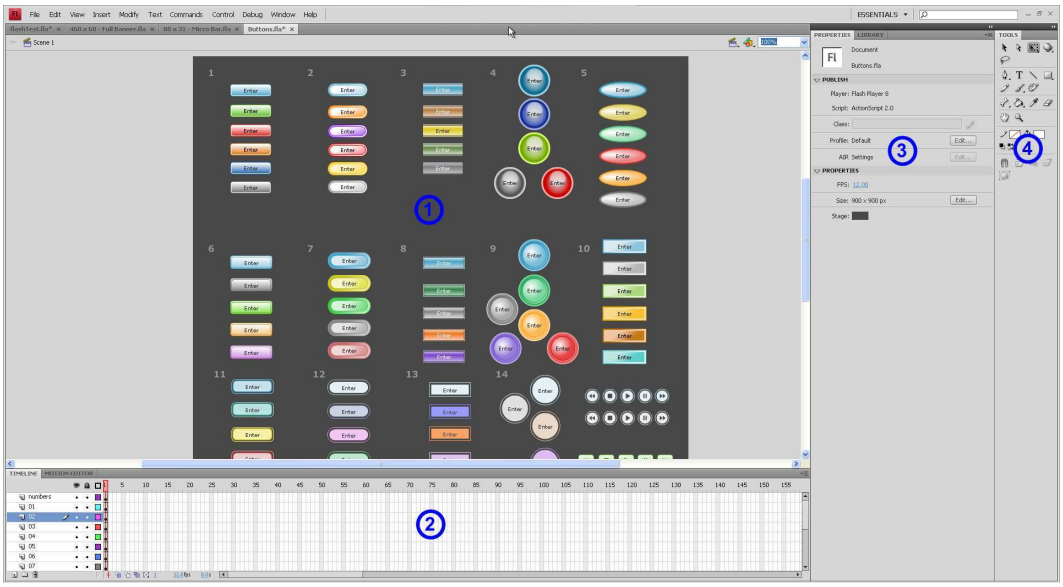

Figura 2.9: Interface do Adobe Flash CS4

Continuando, vemos que o *Stage* possui todas as características desejadas a uma visão espacial. A linha do tempo divide o tempo em quadros, assim como o Director da Seção 2.1.4. A separação de elementos na linha do tempo é feita através de camadas (*Layers*), onde cada uma delas pode conter ações de interatividade, vídeos, áudio, dentre outros. As camadas permitem que o autor escolha as que deseja visualizar ou não, semelhante ao componente da visão de leiaute do Composer da Seção 2.1.1. Devemos dar destaque também à ótima sincronização espaço-temporal encontrada no *Flash*.

Um dos focos principais do *Flash* é o desenvolvimento de animações. Elas são feitas diretamente na linha do tempo e existem componentes auxiliares para os autores que desejam criar animações complexas.

O *Panel* biblioteca armazena todo tipo de mídia utilizado no projeto e, por isso, proporciona uma maior rapidez em encontrar aquela mídia que o autor necessita. Uma característica interessante do *Flash* é que as mídias podem ser transformadas em outros objetos, chamados *Symbols*, para que seja possível utilizá-los nas animações e efeitos 3D, como exemplo podemos gerar um *Graphics* representando uma imagem comum ou um *Movie Clip* para representar uma animação. Quando um símbolo é criado, ele é armazenado na

biblioteca e cada vez que é inserido no *Stage*, uma nova instância é criada. Com isso, pode-se utilizar uma mesma mídia diversas vezes.

Mudando o foco para a interatividade, vemos que assim como o Director da Seção 2.1.4, o *Flash* somente permite a inclusão de interação através da programação de *scripts*, no caso a linguagem utilizada é o *ActionScript* 3.0 (ActionScript, 2006). A Figura 2.10 apresenta na região 1 o componente para edição dos *scripts* e na região 2 componentes pré-fabricados (botão, formulário, etc) para agilizar a programação. Um dos principais problemas dessa última versão do *Flash* é que até para a interatividade mais simples precisa ser criado um *script* consideravelmente complicado para autores não programadores.

Por outro lado, o editor do código em *ActionScript* é bem melhor que os verificados nas seções 2.1.1 e 2.1.4. O código se completa automaticamente a medida que o autor digita e os erros são bem claros.

É importante destacar também que nenhuma representação gráfica é exibida a fim de auxiliar o entendimento das interações presentes no documento. O autor precisa abrir todos os *scripts* para poder compreender a interatividade.

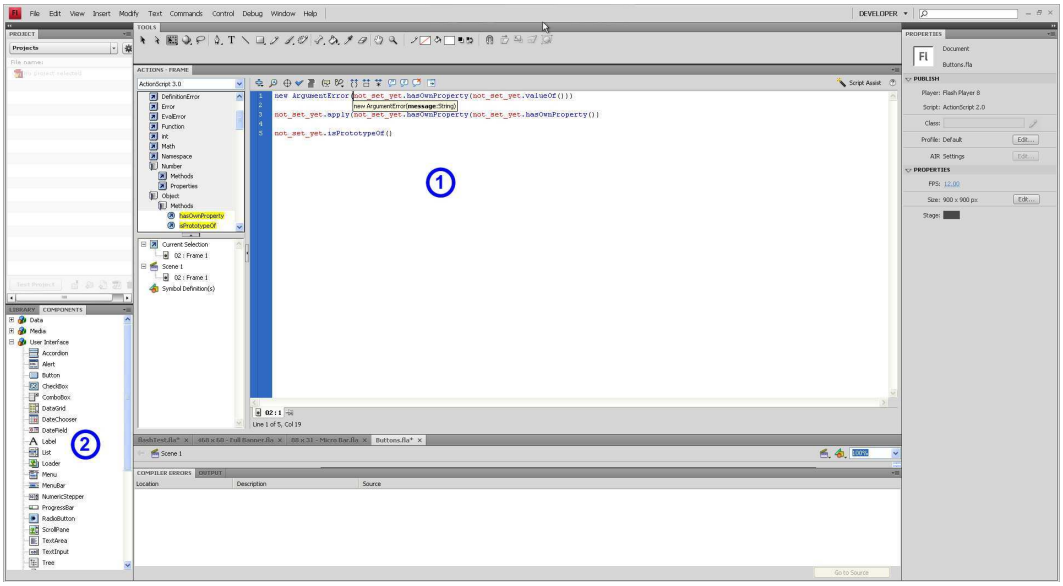

Figura 2.10: Interface de Edição de Scripts do Adobe Flash CS4

O *Adobe Flash* é uma das ferramentas de autoria mais populares do mercado, para não dizer a mais popular. Seu foco sempre foi em autores não programadores e, até as últimas versões, conseguia realizar esse propósito muito bem. Entretanto, o maior problema encontrado atualmente é a limitação de atuação desses autores pelo aumento da complexidade de programação dos *scripts* de interatividade.

A sincronização espaço-temporal é o que mais se destaca no Flash, sendo inclusive melhor que a do Director da Seção 2.1.4. Essa melhora é obtida pela

disponibilidade de diversos componentes que auxiliam na criação de animações mais complexas. A biblioteca também se mostrou muito útil por agilizar a localização e inclusão das mídias no projeto. Além disso, devemos destacar que o editor de código do *ActionScript* é bem mais completo que os encontrados nas ferramentas anteriores.

Por fim, verificamos que a falta de uma representação gráfica para a interatividade faz com que os autores, tanto os programadores quanto os não programadores, fiquem perdidos tentando entender os *scripts*.

2.2 Comparação dos trabalhos relacionados

Terminada a apresentação dos trabalhos relacionados, podemos agora compará-los quanto à plataforma, perfil do autor e aos componentes existentes neles. A Tabela 2.1 apresenta essa comparação. A abreviação N/A representa que a ferramenta não consegue contemplar a característica descrita na linha. Quando a característica é completamente contemplada utilizamos a palavra *Aplicado* e quando é parcialmente contemplada apresentamos o motivo para isso.

Detalhando cada uma das características presentes na Tabela 2.1, temos os dois primeiros, *TV Digital* e *Ginga-NCL*, que apresentam quais ferramentas são voltadas para a TV Digital e para a linguagem NCL, respectivamente.

Os dois seguintes, *Não programador* e *Programador*, nos fornecem a informação de qual ferramenta contempla os dois tipos possíveis de autores.

Encontramos ainda a característica *Interatividade visual*, que informa quais ferramentas utilizam componentes gráficos para auxiliar na edição e visualização da interatividade da aplicação multimídia.

A seguir, temos a *Visão Espacial* e a *Linha do tempo* que se mostraram os componentes mais adequados para tratar as propriedades espaciais e temporais. Temos também os *Modelos estruturais* que são documentos que definem somente a estrutura de uma aplicação e agiliza o processo de autoria por utilizar um modelo baseado em exemplos.

Por fim, verificamos quais ferramentas contêm os componentes *Propriedades*, *Widgets Interativos* e *Biblioteca de mídias*. Eles são secundários se comparados aos anteriormente citados, mas foram incluídos na tabela porque representam um aumento na facilidade de interação do autor com a ferramenta.

		Ferramentas de autoria				
		Composer	Icareus	Director	Flash	LimSee2
Plataforma	TV Digital	Aplicado	Aplicado	N/A	N/A	N/A
	Ginga-NCL	Aplicado	N/A	N/A	N/A	N/A
Perfil do autor	Não programador	Precisa conhecer NCL	Aplicado	Aplicado	Interatividade só por programação	Precisa saber SMIL
	Programador	Aplicado	N/A	Aplicado	Aplicado	Aplicado
Componentes	Interatividade visual	Visão temporal auxilia	Aplicado	Biblioteca com *scripts* pré-fabricados	Interatividade só por programação	Só por programação
	Visão espacial	Somente visão de leiaute	Aplicado	Aplicado	Aplicado	Aplicado
	Linha do tempo	Aplicado	Sem sincronismo temporal	Aplicado	Aplicado	Aplicado
	Editor de código-fonte	Visão textual pobre	Só para não programadores	Editor pobre	Aplicado	Editor pobre
	Modelos estruturais	Usa aplicações NCL e projetos completos	Usa só projeto completo	Usa só projeto completo	Aplicado	Só documento SMIL
	Propriedades	Janelas modais	Aplicado	Aplicado	Aplicado	Aplicado
	Widgets Interativos	Inserção de elementos pelo menu	Aplicado	Aplicado	Aplicado	Componente pobre
	Biblioteca de mídias	N/A	Aplicado	Aplicado	Aplicado	N/A

Tabela 2.1: Comparativo dos trabalhos relacionados