2
Trabalhos Relacionados

Este capítulo tem como principal objetivo discorrer sobre cada uma das ferramentas de autoria existentes atualmente, fazendo uma análise crítica das vantagens e desvantagens por elas apresentadas, levando em conta a linguagem à qual dá suporte e o seu público-alvo. As ferramentas aqui analisadas não se restringem a linguagens voltadas para TVDi. A sua maioria foi construída tendo a Web como nicho tecnológico a ser suportado. As desvantagens levantadas nesse capítulo servirão de base para o levantamento dos requisitos não-funcionais apresentados no Capítulo 3 e das melhoras para a arquitetura que será apresentada no Capítulo 4.

2.1. Ferramentas para linguagens imperativas

Para criar aplicações hipermídia utilizando uma linguagem imperativa, o autor precisa conhecer as bibliotecas específicas e estar familiarizado com o paradigma imposto por esse tipo de linguagem (por exemplo, orientação a objeto no caso de Java). Ferramentas de autoria podem facilitar bastante as tarefas de criação e edição, especialmente para não programadores. Adicionalmente, as linguagens de programação mais utilizadas trazem um ambiente de autoria integrado (IDE), como os pares Java e Eclipse ou Flash e Adobe Flash IDE.

JAME Author, *iTV Suite Author* e *AltiComposer* são ferramentas de autoria para desenvolvimento de aplicações voltadas para TVD. Essas ferramentas têm como linguagem base o Java. Embora essas ferramentas tenham como foco a produção de conteúdo de maneira imperativa, elas seguem um arquétipo declarativo de autoria – escondendo a parte imperativa do autor. Isso demonstra a importância do paradigma declarativo no desenvolvimento de aplicações hipermídia, discutido na próxima seção. Já a ferramenta *Adobe Flash* é amplamente utilizada na criação de conteúdo dinâmico (animações, filmes e jogos) para páginas Web.

*2.1.1.*JAME Author

JAME Author (Fraunhofer Institute for Media Communication IMK, 2004) pretende facilitar o processo de autoria provendo uma abstração (uma visão) composta de "páginas", similares a páginas na Web. Uma página é composta de objetos de mídia e é especificada utilizando uma linguagem chamada de JAME PDL (*JAME Page Description Language*), que é baseada em XML (W3C, 2008).

Existem dois tipos de visões no ambiente do *JAME Author*. Uma das visões permite a adição de componentes pré-definidos em uma página, a especificação e controle do movimento do foco através dos componentes listados na página, e a especificação de elos de navegação entre as páginas. A outra visão presente permite a edição de propriedades de um determinado componente selecionado em uma página. A ferramenta também oferece um emulador capaz de demonstrar ao autor uma prévia da aplicação que está sendo construída.

O processo de autoria é muito similar à autoria de páginas descritas em HTML, o que facilita sua adoção e uso para autores não programadores. Porém, o conjunto de funcionalidades que o *JAME Author* provê é muito limitado (limita em muito as facilidades providas pelas bibliotecas Java). Por exemplo, a ferramenta não dá suporte à definição de relacionamentos espaços-temporais entre os componentes de mídia presentes na página, ou um suporte para definição programática de funções ou métodos escritos em Java.

2.1.2.iTV Suite Author

O *Cardinal Studio* foi por muitos anos a principal ferramenta de desenvolvimento de aplicação hipermídia voltada para o middleware MHP (Multimedia Home Plataform) – middleware Java do sistema europeu de TVD. No final de 2008, a *Cardinal* foi adquirida pela *Icareus Technology* e o *Cardinal Studio* foi substituído por uma ferramenta similar, chamada de *iTV Suite Author* (Icareus Technology, 2010).

iTV Suite Author é uma ferramenta de autoria intuitiva que tem uma abstração denominada de cenas. Os paradigmas e funcionalidades exercidos pelo *iTV Suite Author* e suas cenas, são bastante similares ao *JAME Author* e suas páginas. As cenas definem componentes que podem ou não receber foco, e é

possível marcar componentes a serem reaproveitados entre as cenas, por exemplo um plano de fundo.

A Figura 2, apresenta a interface gráfica do *iTV Suite Author*. Sua interface é composta por um navegador de cenas, onde é possível navegar entre as diferentes cenas construídas; um visualizador de uma cena selecionada; e uma lista de recursos, que são representados pelos objetos de mídia. Além disso, o *iTV Suite Author* provê um pré-visualizador integrado, que dá um retorno em tempo de desenvolvimento das mudanças feitas pelo autor e possibilita a simulação da interação do usuário com a aplicação.

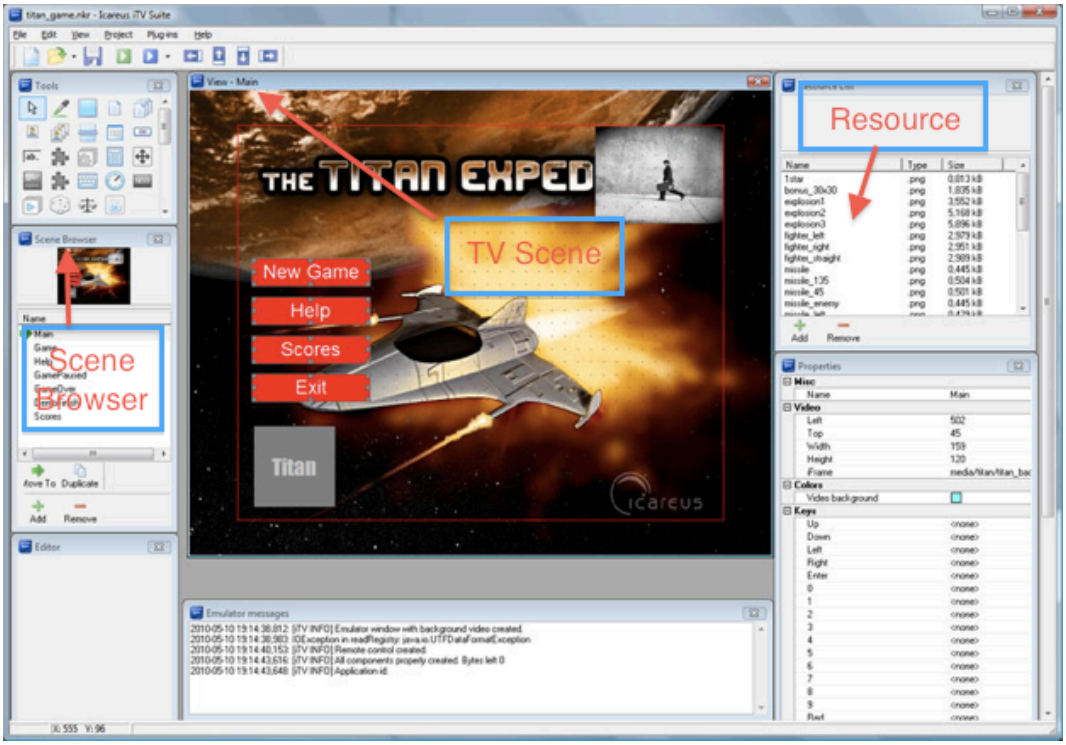

Figura 2 - Interface gráfica do iTV Suite Author

O *iTV Suite Author* é uma das poucas ferramentas onde é possível criar as estruturas de navegação da aplicação, atribuindo ações aos botões do controle remoto; definição de relacionamentos espaços-temporais entre os objetos de mídia presentes em uma cena; e o planejamento do uso do canal de retorno. Porém, essas tarefas exigem um esforço de codificação imperativa por parte do autor da aplicação. Para edição em tempo real o *iTV Suite Author* utiliza fluxo de eventos que podem ser utilizados como uma maneira de disparar a execução de código, também imperativo, definido pelo autor. Assim, a exploração das facilidades um

pouco mais avançadas da linguagem, exige conhecimento usualmente não adquiridos por usuários não-especialistas.

2.1.3.AltiComposer

AltiComposer (Alticast Inc., 2004) utiliza um modelo muito mais elaborado e consistente com a indústria televisiva e cinematográfica, na qual uma cena tem planos, e um plano é composto de tomadas e atores. Assim como no *iTV Suite Author,* é possível a definição do relacionamento espaço-temporal entre os objetos através da codificação por parte do autor. No caso do *AltiComposer* é utilizada uma linguagem de script chamada *AltiComposer Script Language*, um subconjunto de ECMA-262 (Alticast Inc., 2004). O suporte a edição em tempo real também é feito através de gatilhos enviados em fluxos de eventos, assim como no *iTV Suite Author.*

2.1.4.Adobe Flash

Adobe Flash é uma plataforma multimídia usada para combinar animações, vídeos, áudios e interatividade em páginas web. É utilizado principalmente na criação de jogos para web, onde viu seu uso crescer nos últimos anos. A principal crítica está no fato da aplicação feita em *Flash* ser distribuída em um formato proprietário da Abode.

A Figura 3 abaixo apresenta a interface gráfica da ferramenta de edição *Adobe Flash*. Essa ferramenta possuí uma visão espacial, chamada de *stage*, e uma visão temporal. Sobre a visão de *stage* o autor posiciona os objetos de mídia presentes na aplicação. A visão temporal trabalha com pedaços de tempo definidos como *frames*, e a transição entre os *frames* presentes na aplicação que gera a animação final do documento.

Essas visões trabalham juntas dando o posicionamento espacial do objetos ao longo do tempo. Entretanto, a especificação de interação do usuário com a aplicação é feita através da escrita de código utilizando *ActionScript*.

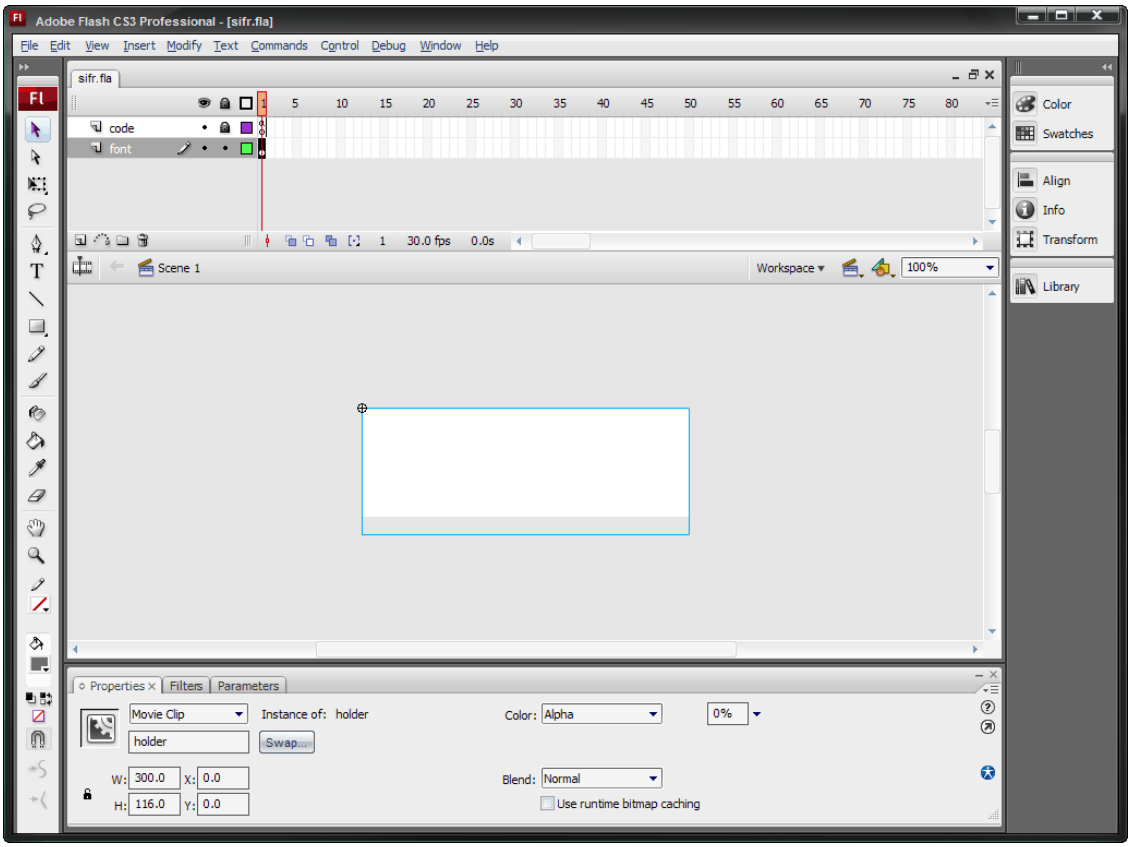

Figura 3 - Interface gráfica do Adobe Flash

2.2.Ferramentas para linguagens declarativas

Linguagens declarativas para desenvolvimento de aplicações hipermídia são usualmente de alto nível de abstração e de domínio específico. *GRiNS, LimSee2* e *SmilBuilder* são voltadas para aplicações para a Web, ao passo que NCLEclipse e Composer abraçam também o domínio de TVDi.

2.2.1.GRiNS e LimSee2

GRiNS (Bulterman D. C., Hardman, Jansen, Mullender, & Rutledg, 1998) e *LimSee2* (INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 2008), são ferramentas de autoria voltadas para a linguagem SMIL, recomendação W3C para aplicações Web com sincronismo temporal, que possuem diversas visões (textual, espacial, temporal etc.) integradas (sincronizadas). Mudanças realizadas pelo autor do documento SMIL em qualquer uma das visões são automaticamente refletidas nas outras visões. No *GRiNS* a visão temporal é a mais importante e é utilizada para compor e manipular a

apresentação do documento SMIL sobre uma linha do tempo horizontal. Já no *LimSee2*, as visões temporal e espacial trabalham juntas provendo um mecanismo poderoso na criação de aplicações hipermídia. Por ser mais completo, toma-se, a seguir, o LimSee2 como comparação.

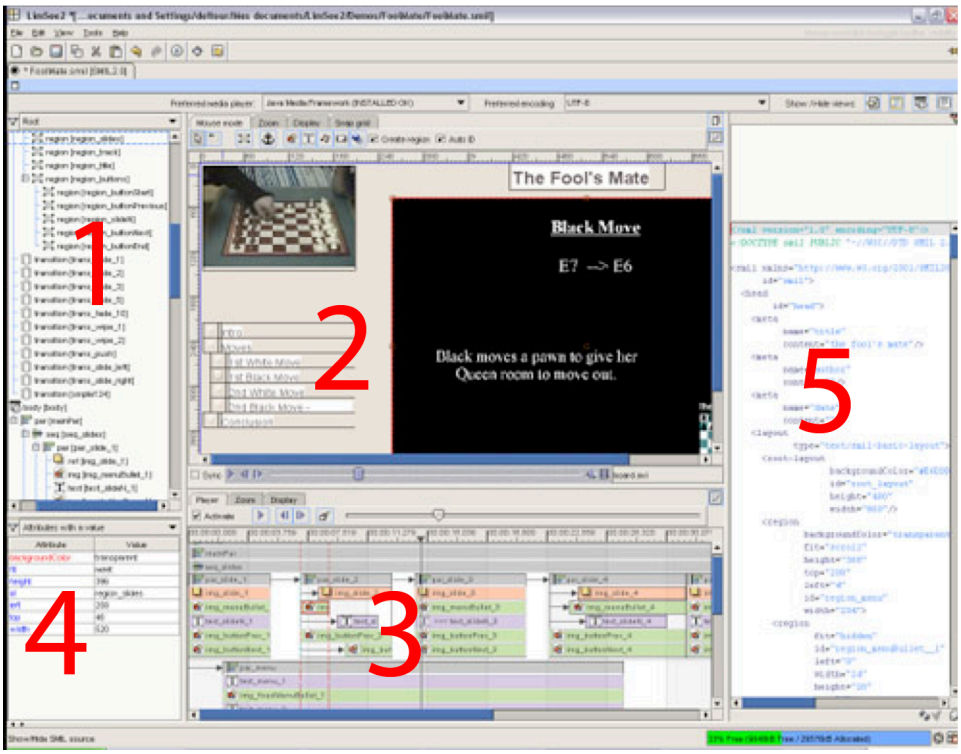

Figura 4 - Interface gráfica do LimSee2

A Figura 4, apresenta a interface gráfica do *LimSee2*. A região marcada como 1 mostra a estrutura[1] do documento SMIL em forma de árvore. A região 2 apresenta a visão espacial, onde o autor define o posicionamento espacial dos objetos de mídia. A visão espacial também permite a pré-visualização do conteúdo que compõe os objetos de mídia, mas apenas os textos e imagens. Essa pré-visualização é de acordo com um momento específico no tempo selecionado na visão temporal. A região marcada como 3 é a visão temporal. Nessa visão os objetos de mídia são representados como retângulos e o comprimento desses retângulos é proporcional ao tempo de exibição daquela mídia. Esses retângulos podem ser adicionados, removidos e manipulados de acordo com a vontade do autor. A região 4 é a visão de propriedades, é através dessa visão que o autor pode modificar propriedades de uma mídia selecionada. Finalmente, a região 5 é a

[1] Em SMIL a estrutura de um documento reflete sua lógica de apresentação (sequencial ou paralela) no tempo.

visão textual. No *LimSee2* essa visão é tida como somente leitura, não é possível a modificação do código-fonte de maneira que essa modificação venha a ser incorporada pelas demais visões. Um ponto negativo em relação ao *GRiNS* é o fato do *LimSee2* não ter uma ferramenta de visualização de aplicativos integrada.

Diferente das ferramentas previamente citadas, tanto o *GRiNS* quanto o *LimSee2* favorecem a especificação do sincronismo temporal entre os objetos de mídia, mas sem a interação do usuário. Entretanto, é mais fácil especificar a interação do usuário nas ferramentas anteriores, voltadas para Java. No *GRiNS* e no *LimSee2*, o autor só consegue especificar a interação do usuário com a aplicação através da visão textual, e essa interação não é graficamente editável, e sua edição, por fora da ferramenta, não é graficamente visível por nenhuma outra visão.

A visão textual dessas ferramentas é muito simples e não agregam funcionalidades que possam facilitar a codificação do documento SMIL. Diferente do *NCLEclipse* que ainda será discutido neste capítulo, o *GRiNS* e *LimSee2* não dão suporte a nenhum tipo de edição em tempo real da aplicação. Essa é uma das desvantagens em relação à ferramentas previamente discutidas.

Atualmente, o *LimSee2* coexiste com o *LimSee3* (Deltour & Roisin, 2006), esse último tem como principal advento o suporte a arquétipos de documentos SMIL. Esse paradigma é baseado em famílias de documentos adaptáveis para diferentes tipos de categoria de autores e aplicações, permitindo a geração rápida de um documento SMIL baseado sobre um arquétipo especifico.

Nesse paradigma, o autor tem como ponto de partida do desenvolvimento a escolha de um arquétipo. Após sua escolha o autor tem seu foco restrito na especificação das lacunas, objetos de mídia, que devem ser preenchidas. Essas lacunas são o que tornam aquela aplicação única.

2.2.2.SmilBuilder

SmilBuilder (Bouyakoub & Belkhir, 2011) é um editor temporal de SMIL com capacidades de verificação incremental do modelo interno. Essa ferramenta permite que o autor construa sua aplicação SMIL passo-a-passo, enquanto garante que em cada passo da autoria o modelo central se encontra em um estado válido.

Esse modelo é baseado no H-SMIL-Net, que é uma extensão temporal de Redes de Petri.

Essa ferramenta é a primeira a trazer um sistema de sincronismo e verificação incremental entre as visões existentes. A Figura 5 apresenta uma visão geral do processo de autoria utilizando o *SmilBuilder*. Cada modificação a ser realizada na aplicação (Documento SMIL) passa pelo teste de consistência antes de sua aceitação: se a modificação levar a um documento inconsistente então é rejeitada e uma mensagem de erro é apresentada para o autor; caso contrário é modificação é aceita e o cenário temporal é atualizado.

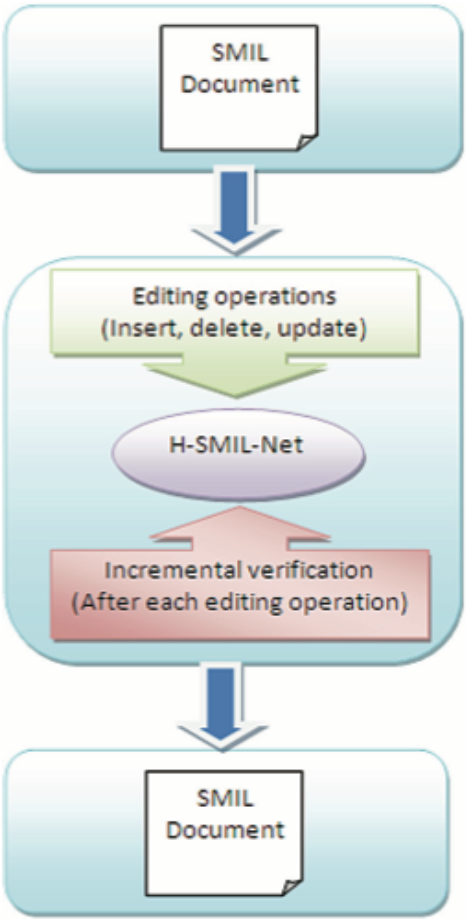

Figura 5 - Visão geral da autoria em SmilBuilder

A arquitetura do *SmilBuilder* é apresentada na Figura 6. O sistema de autoria é dividido em 4 módulos principais que interagem ao longo do processo de autoria descrito acima.

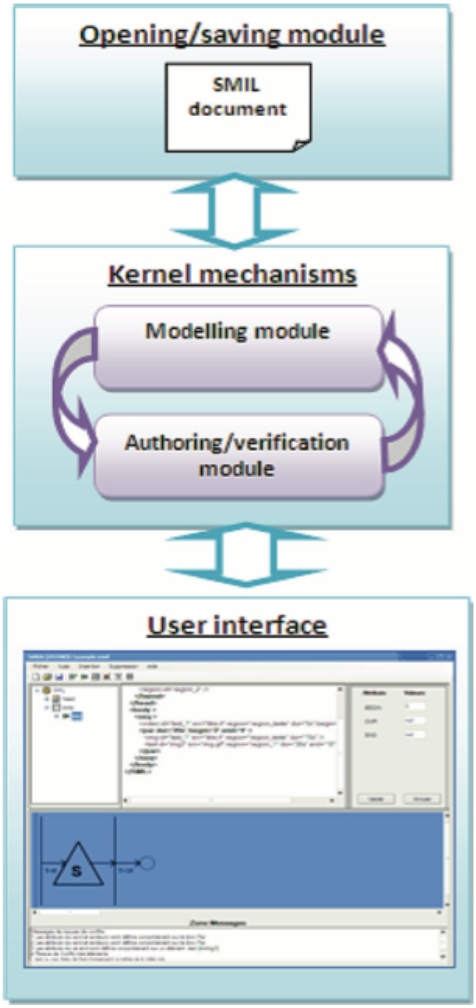

Figura 6 - Arquitetura do SmilBuilder

O formato de entrada da ferramenta pode ser tanto um documento SMIL quanto um documento no formato H-SMIL-Net. O módulo *Open/Saving* é responsável por abrir e salvar tais formatos em disco.

Internamente, a ferramenta trabalha somente com o H-SMIL-Net. Por isso,caso o documento de entrada seja em SMIL, é necessária a tradução desse documento no formato H-SMIL-Net. Esse processo de tradução é executado pelo módulo *Modeling*.

Ao final da construção da aplicação, o modelo interno pode ser salvo tanto como um documento SMIL quanto um documento H-SMIL-Net. Esse último formato é utilizado pela ferramenta para salvar o progresso das alterações feitas até o momento pelo autor. É importante ressaltar que a tradução do H-SMIL-Net para SMIL ou vice-versa não é um processo 1 para 1, resultando em diferentes documentos SMIL com semântica igual.

O módulo *Authoring/Verification* provê funções para criar, modificar o cenário temporal do modelo H-SMIL-Net que está sendo construído. Para manter a coerência sobre a especificação SMIL, o sistema não permite que o documento entre em um estado inconsistente.

Finalmente, o módulo de *User Interface*. Nesse módulo encontra-se a interface gráfica da ferramenta com o autor, apresentada na Figura 7. É composto de 4 visões: hierárquica, textual, atributos e temporal. A visão **textual** é uma visão apenas de **leitura** que apresenta o código-fonte do documento SMIL. A visão **hierárquica** dispõe os elementos do documento em uma estrutura de árvore. A visão **temporal** tem uma representação gráfica do modelo H-SMIL-Net. Através dessa visão, o autor pode visualizar os relacionamentos de sincronismo entre os objetos de mídia do documento. A visão de **atributos** permite ao autor visualizar e modificar os atributos dos elementos selecionados na visão hierárquica.

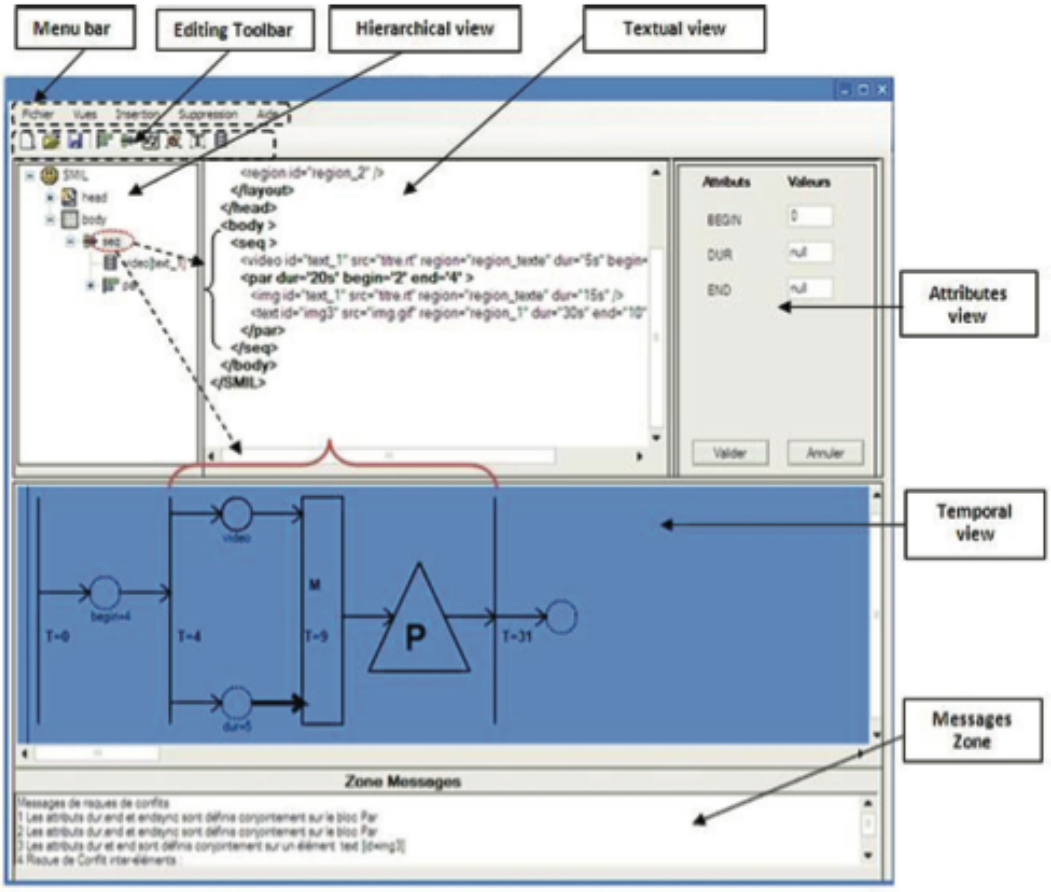

Figura 7 - Interface gráfica SmilBuilder

2.2.3.NCLEclipse

NCLEclipse (Azevedo, Neto, & Texeira, 2009) é um plug-in para o ambiente de desenvolvimento Eclipse que provê suporte para autoria textual de aplicações hipermídia escritas em NCL. Na literatura existem diversos trabalhos (Azevedo, Lima, Neto, & Texeira, 2009) (Santos, Gomes, Azevedo, Neto, & Texeira, 2010) que detalham as funcionalidades do *NCLEclipse*, todas as funcionalidades agregadas têm como foco principal facilitar a codificação de um documento NCL por programadores.

Dentre as principais funcionalidades destacam-se: sugestão automática e contextual de elementos e atributos de NCL; validação em tempo de edição do documento NCL, indicando erros e advertências; coloração dos elementos XML e palavras reservadas da linguagem; navegação sobre o código através de elos de referências; e em sua última versão a refatoração automática de partes do código NCL. Além da visão textual, o *NCLEclipse* tem uma visão de *outline* gráfica que dispõe os elementos presentes no documento em forma de árvore. Esse *outline* dá uma visão geral da estrutura[2] do documento e uma navegação rápida sobre os elementos do documento. Em sua outra visão gráfica, é apresentado a disposição espacial inicial dos objetos de mídia declarados no documento.

[2] Diferente de SMIL, a estrutura lógica de um documento NCL não se confunde com a estrutura temporal de apresentação. A lógica de estruturação utilizada pode até ser a temporal, mas não é necessariamente sempre a temporal, como em SMIL.

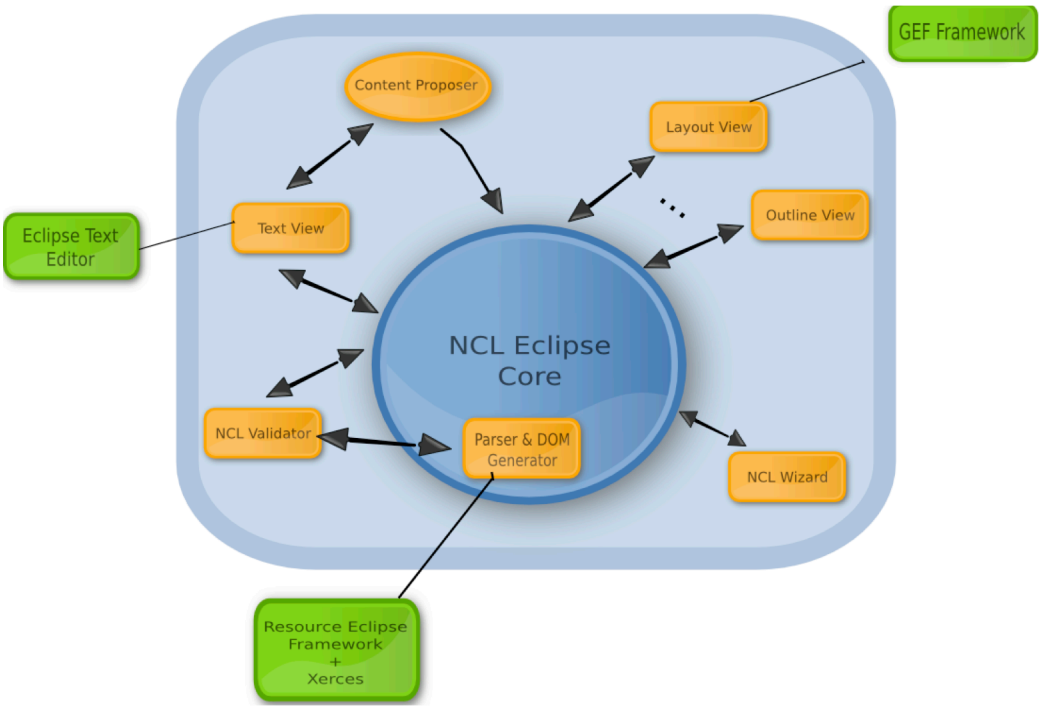

Figura 8 - Arquitetura do NCLEclipse

A Figura 8, apresenta a arquitetura do *NCLEclipse*. Os quadros em verde fazem parte do *framework* de desenvolvimento do Eclipse ou de outros plug-ins do Eclipse que são utilizados pelo *NCLEclipse*, enquanto os quadros em laranja fazem a ponte entre o núcleo do *NCLEclipse* e a plataforma Eclipse. O *Content Proposer* é responsável pela busca e sugestão de código contextualizada, refletindo a estrutura sintática e semântica da linguagem NCL. Essa funcionalidade é a grande vantagem dessa ferramenta, auxiliando os programadores a buscar rapidamente os elementos NCL possíveis em um determinado trecho de código, evitando erros durante a edição.

2.2.4.Composer

A primeira versão do *Composer* (Guimarães, 2007), assim como o *GRiNS e LimSee2*, oferece um alto nível de abstração através de visões gráficas, facilitando o processo de autoria de aplicações NCL em conformidade com o perfil EDTV (Enhanced DTV) da linguagem. A ferramenta foi inicialmente concebida tendo como público-alvo não programadores com pouco ou nenhum conhecimento em NCL.

Cada visão do *Composer*, dá ao autor uma perspectiva diferente e específica sobre o documento. A Figura 9, apresenta a janela principal do *Composer I*, assim

como outras janelas que expõem as visões de autoria existentes: textual, temporal, leiaute e estrutural.

Na Figura 9, o número 1 marca a visão estrutural, que permite ao autor visualizar e editar os objetos de mídia, conforme seus tipos, e a estruturação (agrupamento) desses objetos em conjuntos (contextos que podem conter outros contextos aninhados, incluindo conjuntos contendo objetos alternativos a serem escolhidos em tempo de apresentação) dentro do documento. Relacionamentos espaços-temporais entre objetos de mídia de um conjunto podem ser editados nessa visão. A visão estrutural provê um filtro baseado no algoritmo de olho de peixe dos contextos aninhados presentes na aplicação.

A visão de temporal, marcada como 2 na figura, é semelhante à linha temporal do *LimSee2* e *GRiNS*, dispondo os objetos de mídia sobre um referencial de tempo, dando ao autor a ideia de duração e linearidade da aplicação. No entanto, diferente do *LimSee2* e *GRiNS*, é possível editar relacionamentos com interatividade e até simular sua ocorrência na visão temporal. A visão temporal usa um modelo chamado de HTG (*Hypermedia Temporal Graphs*) (Costa, 2010) para representar o comportamento temporal da aplicação. O HTG preserva os relacionamentos entre os eventos de apresentação de uma aplicação (incluindo a interação do usuário) e permite a especificação desses eventos sobre a linha do tempo, sem perder a habilidade de expressar importantes propriedades presentes em uma aplicação de TVD, por exemplo, objetos de duração indefinida, eventos de interação e adaptação de conteúdo (conteúdos alternativos para um mesmo objeto de mídia). Também pela visão temporal o autor consegue marcar na linha do tempo o ponto de início da visualização de uma aplicação, que será apresentada no *player* NCL integrado à ferramenta.

A visão de leiaute, número 3 na Figura 9, permite ao autor dispor visualmente a posição inicial dos objetos de mídia no espaço. A variação do posicionamento espacial no tempo seria apresentada em uma outra visão (visão espacial) que não foi implementada na versão corrente.

Finalmente, a visão textual, número 4, na Figura 9, dá a liberdade do autor de manipular diretamente o código-fonte do documento NCL, mas sem as facilidades hoje proporcionadas pelo editor NCLEclipse.

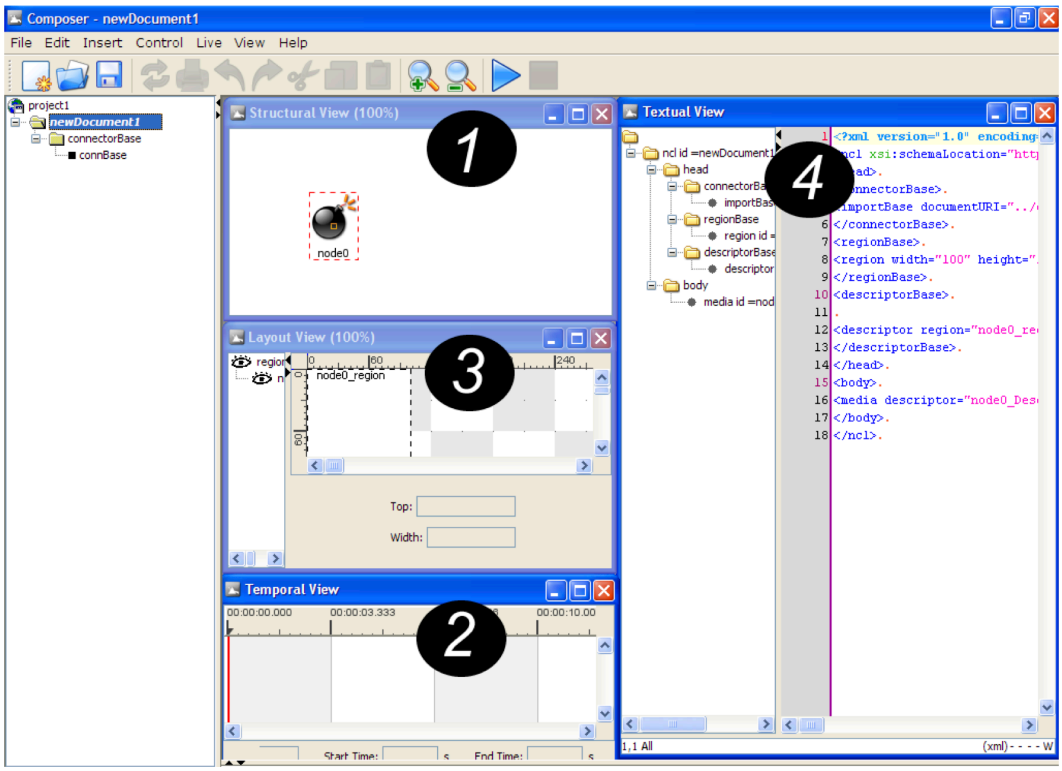

Figura 9 - Visões presentes no Composer I

A aplicação gerada pelo *Composer*, pode ser exportada em um documento NCL. Adicionalmente, o autor pode ativar o modo de edição em tempo real, onde qualquer ação de edição feita pelo autor a partir daquele momento irá gerar um arquivo contendo *NCLEditingCommands* que pode ser utilizado para transmitir as edições a serem executadas em tempo de exibição.

2.3.Comparação das ferramentas

A Tabela 1, faz um comparativo entre as principais características funcionais apresentadas nas ferramentas discutidas neste capítulo. É possível perceber que existem ferramentas com um número maior de visões, mas mesmo essas ferramentas têm um número de visões restrito: não é possível a extensão das visões existentes nem a adição de novas visões. Somente algumas ferramentas possuem suas visões sincronizadas, onde a mudança do autor em uma das visões é refletida nas outras visões de maneira automatizada. Dentre as ferramentas que possuem esse mecanismo de sincronização, somente o *SmilBuilder* o faz de maneira incremental.

Somente o *Composer I* dá suporte total para edição em tempo real dos aplicativos, apesar do *iTV Suite* e *AltiComposer* fornecerem um mecanismo similar em suas funcionalidades. O *iTV Suíte* é a única ferramenta analisada que provê integração com equipamentos de transmissão da aplicação, no caso, o sistema europeu de TVD. Outro critério exposto foi a integração de um exibidor (*player*) integrado à ferramenta, o que é primordial em ferramentas de autoria desse tipo. Através do exibidor o autor tem uma prévia de sua aplicação em tempo de edição. Finalmente, o último critério avaliado foi a possibilidade do autor editar simultaneamente mais de um documento hipermídia. Todas as ferramentas aqui analisadas não permitem essa funcionalidade.

A comparação das características não-funcionais das ferramentas é deixada para o próximo capítulo.

	Número Visões	Sincronismo Visões	Edição em tempo real	Player Integrado	Transmissão do aplicativo	Edição simultânea de documentos
JAME	2	Não	Não	Sim	Não	Não
ITV Suite	2	Não	Parcial	Sim	Sim	Não
AltiComposer	N/A	Não	Parcial	Sim	Não	Não
GRiNS	4	Sim, mas não incremental	Não	Sim	Não	Não
LimSee2	4	Sim, mas não incremental	Não	Não	Não	Não
LimSee3	4	Sim, mas não incremental	Não	Não	Não	Não
SmilBuilder	4	Sim, incremental	Não	Não	Não	Não
Flash	2	Parcial	Não	Não	Não	Não
NCLEclipse	3	Sim, mas não incremental	Não	Não	Não	Não
Composer	4	Sim, mas não incremental	Sim	Sim	Não	Não

Tabela 1 - Comparação funcional entre as ferramentas apresentadas