7
Conclusões

Este último capítulo resume as principais comparações do trabalho desenvolvido nesta tese com trabalhos relacionados, em seguida, realça as contribuições mais importantes e, finalmente, propõe alguns trabalhos futuros.

7.1
Comparação com Trabalhos Relacionados

Algumas comparações com trabalhos relacionados já foram feitas no decorrer do texto, principalmente no Capítulo 2, onde as principais características de linguagens e modelos hipermídia (Furuta, 2002; Na, 2001; W3C, 2001a; W3C, 2001d; W3C, 2001c; Díaz, 2001; Ossenbruggen, 2000; Soares, 2000; Antonacci, 2000c; Antonacci, 2000a; Lowe, 1999; Hardman, 1998; Jourdan, 1998; ISO, 1997; Willrich, 1996; Carr, 1995; Berners-Lee, 1994; Halasz, 1994; van-Rossum, 1993; Hardman, 1993b; Buchanan, 1993; Vazirgiannis, 1993; Yankelovich, 1985), dado o enfoque deste trabalho, foram discutidas e comparadas com características de linguagens de descrição de arquitetura (ADL) (Mehta, 2000; Medvidovic, 2000; Monroe, 1999; Paula, 1999; Allen, 1998; Allen, 1997b; Allen, 1997a; Garlan, 1997; Moriconi, 1997; Shaw, 1996a; Taylor, 1996; Medvidovic, 1996; Magee, 1996; Garlan, 1995a; Magee, 1995; Luckham, 1995b; Luckham, 1995a; Shaw, 1995). Entretanto, alguns aspectos merecem um destaque maior. Para facilitar o entendimento do leitor, cada tópico será discutido separadamente nas próximas subseções.

7.1.1
Tratamento de Relações em Linguagens/Modelos Hipermídia

Apesar da necessidade de representar diversos tipos de relação, tais como relações de referência, de sincronização, de estruturação, de derivação etc., o enfoque dado pela maioria dos modelos/linguagens hipermídia encontrados na

literatura é normalmente sobre como representar os relacionamentos (e não relações) entre componentes de um documento, na maioria das vezes se limitando a relacionamentos de sincronização temporal e de referência. Esse é um dos pontos principais que destaca o trabalho desenvolvido nesta tese dos demais, pois com a introdução do conceito de conector em modelos hipermídia, torna-se possível modelar qualquer tipo de *relação* de forma independente de como ela será usada para indicar um *relacionamento* específico. O conector encapsula a definição da relação e a isola do restante do documento, o que permite aumentar a expressividade da linguagem de autoria, tornando possível definir qualquer tipo de relação sem que o usuário final precise conhecer sua definição em detalhes.

Através de bibliotecas de relações, pode-se agrupar conectores e deixá-los disponíveis para que sejam utilizados na criação de elos. Para usar um conector, o autor só precisa conhecer seus pontos de interface (papéis do conector) e como se associa cada um deles aos nós desejados, para criar os relacionamentos em um documento, o que depende da linguagem de autoria em questão. O reuso de definições é bastante incrementado pelo fato de se poder reutilizar um conector para criar relacionamentos distintos, mas que possuem a mesma semântica, seja ela qual for.

A maioria dos modelos/linguagens hipermídia, tais como Dexter, AHM, NCM (versão anterior), Labyrinth, Microcosm, XHTML, SMIL e XLink, não considera a relação hipermídia como entidade do modelo. Existem algumas exceções, como por exemplo, Madeus (Jourdan, 1998), que oferece relações de alto nível, e os modelos baseados em redes de Petri, tais como HTSPN (Willrich, 1996) e caT (Furuta, 2002; Na, 2001), onde os vários tipos de transição representam tipos distintos de relação. Entretanto, mesmo nesses modelos só são contempladas alguns tipos de relação, normalmente relações específicas de sincronização e referência, não sendo fornecido a possibilidade do usuário definir outros tipos que sejam necessários. Além disso, esses modelos não têm o conceito de ponto de interface da relação, o que diminui a expressividade de cada relação particular, pelo fato de limitar possíveis combinações entre pontos de interface distintos. O conceito de conector para representar uma relação é bem mais geral, além de oferecer diferentes pontos de interface, permite qualquer combinação entre eles para determinar a semântica da relação, através do *glue*.

7.1.1.1
Relações Compostas por Nós e Elos

Outra facilidade decorrente do uso de conectores é a possibilidade de se ter conectores hipermídia compostos, que é uma facilidade não encontrada em nenhuma outra linguagem de autoria hipermídia. Entretanto, para definir conectores compostos, é necessário conhecer quais são os elementos de linguagem usados na definição de componentes e elos, o que torna o módulo de definição de conectores compostos dependente do perfil de linguagem em questão. Por essa razão a linguagem XConnector não contempla a definição de conectores compostos, e sim o módulo *CompositeConnector* de NCL 2.0, que depende de outros módulos da linguagem.

O único sistema de autoria que oferece um conceito parecido com o de conectores compostos é o sistema caT (Na, 2001), baseado em um modelo de redes de Petri. No caT, transições podem representar sub-redes (*subnets*) compostas de outras transições e lugares, permitindo o reuso de estruturas em vários documentos distintos, como será comentado na Seção 7.1.5. Entretanto, esse é o único tipo de composição oferecido por caT, que não oferece composição hierárquica nos lugares, como por exemplo o modelo HTSPN (Willrich, 1996). A nova versão do modelo NCM (Soares, 2003) oferece nós de composição e ainda conectores compostos, dando ao usuário a escolha de qual abstração ele deseja utilizar para modelar sua estrutura composta.

7.1.2
Definição de Elos

Mesmo usando o conceito de conector, a definição de um elo continua necessária. A linguagem XConnector não tem como propósito definir relacionamentos, representados por elos específicos, essa responsabilidade continua sendo da linguagem de autoria de documentos. Com isso, questões relativas à definição de elos, como por exemplo, a especificação do contexto de um elo ou a computação automática de origens ou destinos de um elo ficam a critério da linguagem de autoria de elos.

O contexto de um elo (Hardman, 1993a) indica quais partes da apresentação devem permanecer sendo apresentadas e quais devem ser subtituídas quando um

elo é seguido. A definição de relações multiponto é uma solução elegante para esse problema (Soares, 2000), pois diferentes ações de um conector causal podem modelar o comportamento desejado e, na definição do elo, esse comportamento é associado às partes da apresentação correspondentes, não exigindo atributos específicos do elo para isso. Essa abordagem é diferente da utilizada em (Hardman, 1998), que propõe o uso dos chamados *link specifier contexts* para esse propósito.

A computação automática de origens e destinos de um elo, tal como o que acontece com os elos genéricos de Microcosm (Lowe, 1999) e os elos virtuais de NCM (Soares, 2000a) e Labyrinth (Díaz, 2001), também é responsabilidade da linguagem de autoria de elos. A linguagem NCL 2.0, por enquanto, oferece recursos limitados para computação automática de extremidades de um elo. Essa facilidade é decorrente da possibilidade de um nó *switch* ser a extremidade de um elo. Com isso, o formatador do documento, após resolver qual dos componentes do *switch* será apresentado, resolve, então, qual a extremidade real do elo. Recursos mais elaborados para computar origens e destinos de elos em NCL 2.0 foram deixados para trabalho futuro.

7.1.3 Extensões ao padrão XLink

Apesar de ainda poucos os trabalhos encontrados na literatura (Bulcão-Neto, 2002; Ciancarini, 2002; Halsey, 2000) endereçando a linguagem XLink (W3C, 2001c), sem dúvida, padronizar as construções de elos em linguagens XML é muito útil e importante. Nenhum desses trabalhos propõe extensões à XLink, como esta tese realizou. As referências (Bulcão-Neto, 2002; Ciancarini, 2002) apresentam implementações de serviços de elos baseados em XLink e a referência (Halsey, 2000) apresenta o uso de XLink como um formato para exportar elos definidos em um sistema hipermídia aberto.

Um discussão que poderia ser levantada é sobre o motivo de se estender o padrão XLink com facilidades de XConnector para especificar relações de sincronização, já que já existe outra recomendação com essa finalidade que é a linguagem SMIL (W3C, 2001d). Entretanto, vale lembrar que as facilidades oferecidas pelas bases de elos de XLink, que permitem a definição de elos de

forma completamente independente dos recursos relacionados, seguem uma filosofia bem distinta da definição de um documento SMIL. Estender a expressividade de XLink para definir relações de sincronização permite que a especificação do sincronismo entre um conjunto de documentos, incluindo documentos SMIL, possa ser feita em várias bases de elos independentes. Essas bases de elos podem ser agrupadas de acordo com a preferência do autor ou até mesmo do leitor, oferecendo mais flexibilidade que a autoria de um único documento SMIL autocontido. A recomendação XLink sugere uma forma de associar bases de elos a documentos, através de uma regra de navegação (elemento do tipo *arc*), cujo valor do atributo *arcrole* deve ser "http://www.w3.org/1999/xlink/properties/linkbase". Outra opção seria através de um documento NCL 2.0, agrupando as bases de elos desejadas.

Dado que XLink, agora, pode representar relacionamentos complexos entre recursos XML, é necessário permitir o encapsulamento das relações criadas em bibliotecas, tornando-as disponíveis para reuso por vários usuários XLink. Por essa razão, outra facilidade decorrente da extensão de XLink com XConnector é a possibilidade de criação de bases de conectores, além das tradicionais bases de elos, permitindo a criação de tais bibliotecas de relações.

Uma outra vantagem de se estender XLink com o uso de conectores é transformar XLink em um padrão para definir relacionamentos quaisquer entre recursos XML, e não apenas elos hipermídia. Essa é a principal vantagem de usar o conceito de conector aliado à XLink. Talvez a associação desse conceito às facilidades oferecidas pelo padrão RDF – *Resource Description Framework* (W3C, 1999a) resultasse em trabalhos futuros bastante interessantes e que vão de encontro à definição de Web Semântica (Berners-Lee, 2001).

7.1.4 Composições Hipermídia com Semântica Embutida

Oferecer composições com semântica embutida para representar determinados tipos de relacionamentos entre componentes de um documento hipermídia tem a grande vantagem de facilitar bastante o processo de autoria. O exemplo principal desse tipo de abordagem é o antigo formato CMIF (van-Rossum, 1993), que mais tarde deu origem ao padrão SMIL (W3C, 1998c; W3C,

2001d), oferecendo composições com semântica temporal. O uso de composições com semântica facilita bastante o trabalho do autor por evitar que ele tenha que criar vários elos entre um grupo de nós para obter uma semântica idêntica à da composição. A referência (Rodrigues, 1999b) discute a representação de composições paralelas e seqüenciais através de elos de sincronização, ilustrando bem a diferença de complexidade entre os dois tipos de abordagens.

Por outro lado, apesar de introduzir facilidade de autoria, linguagens que usam essa abordagem oferecem um conjunto limitado de tipos de composições. Por exemplo, a versão 1.0 de SMIL (W3C, 1998c) oferecia apenas as composições paralela (*par*) e seqüencial (*seq*). Com isso, cada vez que sente-se a necessidade de um novo tipo de composição, não só a linguagem precisa ser alterada, como todos os formatadores baseados na linguagem precisam ser modificados para contemplar a semântica do novo tipo. Foi o que aconteceu quando a composição exclusiva (*excl*), que indica que apenas um de seus componentes pode ser executado por vez, foi introduzida pela versão 2.0 de SMIL (W3C, 2001d).

Apesar deste trabalho se basear no uso de elos para expressar relacionamentos entre componentes de um documento, isso não quer dizer que o uso de composições com semântica embutida não seja defendido. O que se propõe, no entanto, é flexibilizar os tipos de composição que uma linguagem pode oferecer, e não obrigar que o autor precise montar uma hierarquia de composições de tipos básicos para representar relacionamentos mais complexos entre componentes. Para atingir esse objetivo, este trabalho utilizou o conector hipermídia como unidade básica para definir relacão e propôs a definição de templates de composição para compor essas unidades básicas e especificar determinada semântica para uma composição. Com isso, usuários podem especificar diferentes tipos de templates, cada um deles com uma semântica particular, e a linguagem para especificar documentos que usam templates não precisa ser modificada. Se a expressividade da unidade básica para definir relação, ou seja, do conector, for mantida, nem mesmo os formatadores precisam ser modificados quando novos templates são criados e utilizados. Como a linguagem XConnector para definir conectores só contempla, por enquanto, relações de sincronização e relações de referência, os templates de composição, por

conseqüência, só podem ser especificados atualmente com esse propósito, mas o objetivo da proposta é geral.

7.1.5
Uso de Templates para Autoria Hipermídia

O uso de templates e seus benefícios em termos de reuso para autoria hipermídia já foram discutidos por diversos trabalhos na literatura (Nanard, 2000; Rossi, 2000; Nanard, 1998; Fraïssé, 1996). No entanto, a maioria dos trabalhos que discutem reuso em hipermídia, o fazem no nível de design e não no nível de especificação/implementação, como esta proposta o fez. A referência (Rossi, 2000) discute o uso de *design patterns* e *frameworks* para melhorar o design de aplicações hipermídia baseadas no modelo OOHDM (Schwabe, 1995). A referência (Nanard, 2000) apresenta um formalismo chamado MCF – *Media Construction Formalism* – para a especificação de cenários multimídia baseados em um modelo temporal similar ao do SMIL 2.0, mas, novamente, focando em problemas de design.

A proposta de templates desta tese aborda a autoria de documentos hipermídia no nível de especificação/implementação e seu objetivo principal é facilitar o trabalho do autor, permitindo o reuso de especificações prévias e dando semântica às composições. Nada impede que um modelo de design seja utilizado e que, no processo de conversão do modelo de design para o modelo de especificação/implementação, design patterns possam ser traduzidos para templates de composição. A referência (Nanard, 2000) comenta que os autores tentaram fazer isso de MCF para SMIL e que encontraram dificuldades, pois o modelo de design era mais rico. Isso enfatiza a necessidade de aumentar a expressividade de linguagens de especificação hipermídia.

Reuso no nível de implementação também é provido pelo sistema caT (*context-aware Trellis*) (Na, 2001; Furuta, 2002). Como já mencionado, caT é um sistema hipermídia baseado em redes de Petri, que fornece reuso da estrutura de documentos, além do reuso de conteúdo, tradicionalmente oferecido pela maioria. Na representação do sistema, uma transição em uma rede de Petri de nível mais alto pode ser mapeada em uma sub-rede (*subnet*) separada, e lugares da rede de mais alto nível em lugares da sub-rede, permitindo que a definição de uma

estrutura seja reusada com elementos de conteúdo distintos. Além de *subnets*, caT fornece os chamados *templates*, que são usados para especificar como elementos de conteúdo e elos, modelados como lugares e transições da rede de Petri, são apresentados quando um usuário navega pelo documento na Web. Os arquivos de templates de caT são tratados simplesmente como mais um tipo de conteúdo, que é essencialmente uma página HTML com indicadores de lacunas que sinalizam aonde a informação deverá ser embutida quando as transições e os lugares correspondentes são marcados e habilitados.

Um trabalho que se aproxima mais da proposta de templates desta tese é discutido em (Fraïssé, 1996; Nanard, 1998). Ele apresenta o conceito de *constructive templates*, que permite a geração automática de instâncias de templates em um documento alvo, fornecendo o reuso de estruturas específicas e bem conhecidas. Os templates propostos por (Fraïssé, 1996) permitem a especificação de relacionamentos causais entre componentes através da cláusula "When", que especifica uma condição e uma ação a serem executadas com um retardo opcional. A linguagem XTemplate oferece mais expressividade para definir relacionamentos, já que usa XConnector, que permite especificar relações multiponto com semântica causal ou de restrição. Além disso, XTemplate também permite a definição de restrições sobre componentes do template, o que não é oferecido por (Nanard, 1998). Um recurso interessante de (Nanard, 1998) é a possibilidade de definir variáveis que podem ser parametrizadas por instâncias do template. Esse recurso é colocado como trabalho futuro desta tese.

7.1.6
Restrições em documentos XML

O uso da linguagem XML Schema (W3C, 2001b) já possibilita a definição de algumas restrições que devem ser satisfeitas por documentos que seguem o esquema com relação a número de ocorrências de elementos e ainda valores de atributos. Por exemplo, na definição de XConnector com XML Schema, disponível no Apêndice A, pode-se observar restrições para exigir que toda referência a papéis do tipo condição, ação ou propriedade, feita por atributos do *glue* de um conector, deve identificar o atributo *id* de um papel daquele mesmo tipo. No entanto, as restrições fornecidas por XML Schema não foram suficientes

para representar os tipos de restrições que um template de composição pode especificar, por exemplo, garantindo que os números de ocorrências de dois tipos de elementos do template sejam idênticos.

Durante o processo de especificação da linguagem XTemplate, foram investigadas algumas linguagens para definição de restrições, entre elas, as linguagens de predicados utilizadas pelas ADLs Wright (Allen, 1997a; Allen, 1998) e Armani (Monroe, 1999) para especificar restrições em estilos arquiteturais, a linguagem OCL – *Object Constraint Language* (OMG, 1997) usada para definição de restrições em especificações UML (Rumbaugh, 1999) e a linguagem XCSL – XML *Constraint Specification Language*, proposta por (Ramalho, 2001; Ramalho, 2000).

XCSL é uma linguagem para restringir o conteúdo de documentos XML. Ela provê construtores que definem ações que devem ser disparadas sempre que uma expressão booleana, sobre o valor do conteúdo de um elemento e de seus atributos, for avaliada como falsa. É fornecido um processador XCSL (XCSL *Processor Generator*), que transforma um documento XCSL em uma folha de estilo XSL, permitindo o uso de um processador XSL tradicional para validar a semântica dos documentos. O diagrama que ilustra o processamento de especificações XCSL é exibido na Figura 49.

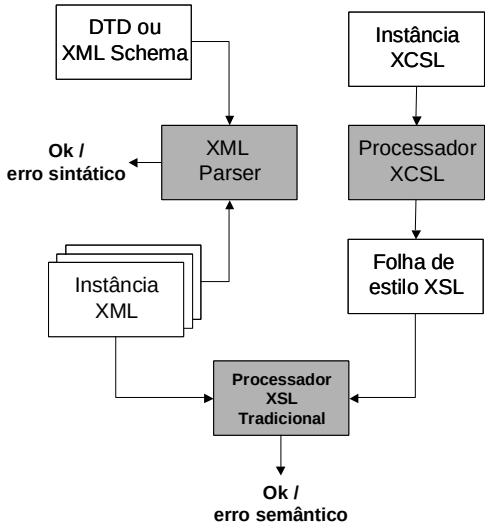

Figura 49. Processamento de especificações XCSL

Um outro trabalho relacionado é (Provost, 2002) que discute o uso de XPath (W3C, 1999b) e XSLT (W3C, 2001f; W3C, 1999c), além de XML Schema, para

especificar restrições sobre o conteúdo de documentos XML, usando também uma abordagem baseada em transformações para resolvê-las.

Inicialmente, pensou-se em definir uma linguagem própria para especificar restrições. No entanto, após o conhecimento sobre esses trabalhos relacionados (Ramalho, 2001; Provost, 2002) e um estudo mais detalhado das linguagens XPath e XSLT, verificou-se a possibilidade de utilizar essas tecnologias, padronizadas pelo próprio W3C, para especificar restrições em um template de composição. Além de utilizar linguagens existentes e bastante poderosas, o que generaliza o tipo de restrição que um template pode especificar, ainda ganha-se a facilidade de poder verificar as restrições através do processamento de uma folha de estilo XSL tradicional, tornando-se critérios decisivos para a especificação da linguagem XTemplate.

Por outro lado, apesar da generalidade que as linguagens XPath e XSLT proporcionam, a especificação de um template de composição torna-se mais complexa. Análogo ao comentado por (Gardner, 2002) para a construção de templates XSLT, a dificuldade na especificação de um template de composição concentra-se em dominar a linguagem XPath. Muitas vezes, os elementos da linguagem XTemplate são especificados corretamente, porém o template não funciona adequadamente. Provavelmente, a razão do problema é a especificação incorreta de expressões XPath, usadas em atributos *select* de alguns elementos do template. Por estarem incorretas, as expressões não selecionam os componentes que deveriam e o documento final gerado após o processamento do template não é o desejado. Esse fato foi constatado durante os testes da implementação do processador de templates de composição.

7.1.7
Linguagem NCL 2.0

Como mencionado no Capítulo 3, a linguagem de autoria hipermídia NCL 2.0 oferece várias facilidades, satisfazendo os requisitos necessários para a especificação completa de um documento. NCL 2.0 permite a representação de tipos diferentes de objetos de mídia; a especificação de documentos de forma estruturada, através de suas composições; a definição de portas e mapeamentos para nós de composição, satisfazendo a propriedade de composicionalidade dos

documentos; a definição de relações espaço-temporais complexas com semântica causal ou de restrição, através de conectores hipermídia simples ou compostos; o uso de conectores para a autoria de elos; a definição de templates de composição hipermídia, possibilitando a especificação de uma semântica para nós de composição; o uso de templates para a autoria de nós de composição; a definição de bases de elos, bases de conectores e bases de templates de composição; o reuso de nós, elos, bases de elos, conectores e templates de composição em diferentes documentos; a definição das características de apresentação separada da definição dos nós; a especificação de documentos que podem ser adaptados ao leitor e à plataforma em que estiverem sendo exibidos, através da especificação do comportamento temporal de forma flexível, de alternativas de conteúdo e, ainda, de alternativas de apresentação para nós do documento.

Sabe-se que, atualmente, a linguagem SMIL 2.0 é o padrão consagrado pelo W3C para autoria de documentos multimídia interativos. Com isso, um questionamento que pode ser feito é sobre o porquê da especificação de uma nova linguagem, sendo que já existe um padrão com objetivos similares. A resposta a essa pergunta é o seguinte. As contribuições principais desta tese são, sem dúvida, a introdução de conectores e templates de composição como novos conceitos em linguagens de autoria hipermídia. A especificação de NCL 2.0 foi uma conseqüência de utilizarmos a linguagem NCL para abrigar a implementação de tais conceitos. Uma nova questão que pode ser colocada é o porquê de não utilizar SMIL 2.0 como linguagem para implementação, ao invés de NCL. De fato, NCL 2.0 utiliza alguns módulos e conceitos definidos por SMIL 2.0. Entretanto, NCL apresenta algumas diferenças marcantes com relação à filosofia de autoria adotada por SMIL, que acabou justificando a especificação de uma nova linguagem ao invés de um novo perfil SMIL.

A principal diferença é a utilização de composições com semântica embutida para especificar relacionamentos de sincronização, seguida por SMIL, e o uso de elos para atingir o mesmo objetivo, seguido por NCL, desde sua primeira versão (Antonacci, 2000a). Além disso, o fato de se basear no modelo NCM, faz com que NCL permita a definição de composições simplesmente para agrupar ou estruturar partes de um documento, o que não é contemplado por SMIL. Os conceitos de conectores e templates de composição hipermídia, originados dos

conectores e estilos arquiteturais de ADLs, se enquadram melhor junto à abordagem de NCL. Como mencionado, uma abstração diferente da de componente (ou nó) é usada para representar relações (conectores) em um sistema de software (ou documento). Uma terceira entidade, chamada de configuração (ou composição), é responsável por interligar componentes (ou nós) através de relacionamentos específicos (ou elos) para representar a arquitetura de um sistema (ou documento) com um comportamento particular.

Um outro fato que determinou o uso de NCL, foi dispor da implementação do sistema HyperProp, e de seu formatador, para executar documentos baseados no modelo NCM, o que facilitou a implementação realizada e o teste das idéias propostas.

Mais uma vez, um dos objetivos principais do trabalho foi propor contribuições gerais que possam ser aplicadas a outras linguagens, onde a implementação do perfil SMIL+XTemplate é considerada como trabalho futuro.

Em adição às comparações sobre conectores e templates, já realizadas nas seções anteriores, outras facilidades de NCL 2.0 merecem considerações especiais.

Além da possibilidade de especificar alternativas de conteúdo, através do elemento *switch*, idêntico ao que é oferecido por SMIL, NCL 2.0 permite a especificação de alternativas de exibição para um mesmo nó. Alternativas de exibição podem ser especificadas através do elemento *descriptorSwitch*, que pode ser associado a um nó através de um elo, de uma composição ou pelo próprio nó, seguindo o cascateamento determinado na Seção 3.1. Na linguagem SMIL, as características de apresentação de um nó estão especificadas junto ao nó, exceto informações de layout espacial, que são definidas em separado. Para especificar alternativas de exibição para um mesmo nó em um documento SMIL, é necessário definir um elemento *switch* contendo repetições de um elemento com o mesmo conteúdo, associado a diferentes informações de exibição. Em NCL, basta especificar um único nó com o conteúdo desejado e associar um conjunto de descritores alternativos, representado por um elemento *descriptorSwitch*.

NCL oferece elos multiponto, representando relações com semântica causal ou de restrição, de acordo com o conector usado pelo elo. Além disso, uma mesma relação causal, por exemplo, pode relacionar eventos de diversos tipos,

além dos tradicionais eventos de seleção e apresentação, normalmente contemplados. SMIL 2.0 só oferece elos ponto-a-ponto, que podem ser disparados por eventos temporais. A possibilidade de usar eventos na especificação temporal de um documento SMIL 2.0 deu bastante flexibilidade à linguagem, comparada com sua versão anterior. Entretanto, para especificar relacionamentos complexos, o autor deve mesclar o uso dos contêineres temporais *par*, *seq* e *excl*, com o uso de eventos e ainda com o uso de elos, se for o caso, para especificar relacionamentos que envolvem a ocorrência de vários tipos de evento. Em NCL 2.0, uma relação, por mais complexa que seja, pode ser abstraída por um único conector e utilizada da mesma forma que uma relação bem simples.

Outra questão que pode ser discutida se refere à definição dos atributos de teste para escolha de um objeto contido em um *switch* ou em um *descriptorSwitch*. Em SMIL, esses atributos são declarados diretamente nos objetos contidos em um elemento *switch*. Em NCL, optou-se pela similaridade à SMIL, adotando o mesmo procedimento. Entretanto, em linguagens de programação, normalmente existe um outro elemento de linguagem para declarar o teste a ser feito para decidir que parte do programa será executada. Talvez essa fosse a forma mais adequada de definir atributos de teste para cada alternativa de um *switch* ou de um *descriptorSwitch*.

Outro comentário que merece destaque é sobre a especificação do comportamento temporal de objetos de forma flexível. Desde a versão 1.0 de NCL, que as durações de objetos na especificação de um documento podem ser definidas com valores ótimo, mínimo e máximo. Essa facilidade só foi incluída na linguagem SMIL em sua segunda versão. Além disso, NCL prevê a especificação de funções de custo (Rodrigues, 2003) para ajudar o formatador a realizar possíveis ajustes temporais durante a apresentação do documento, visando minimizar a perda de qualidade.

7.2 Contribuições da Tese

A principal contribuição desta tese foi a introdução de mais uma entidade nos modelos hipermídia, para a representação de relações. Além dos tradicionais, nós, elos e composições, os conectores hipermídia (Muchaluat-Saade, 2001a;

Muchaluat-Saade, 2001b) desempenham um papel importante na modelagem de um documento, encapsulando a definição de uma relação que pode ser usada para a criação de vários elos distintos em documentos quaisquer.

O conceito de conector foi trazido do domínio de arquitetura de software para o domínio hipermídia, sendo que o primeiro passo para a introdução desse novo conceito foi realizar uma comparação entre linguagens de autoria hipermídia e linguagens de descrição de arquitetura, identificando conceitos presentes em ADLs que poderiam ser aplicados à área de sistemas hipermídia (Muchaluat-Saade, 2001c). Assim sendo, a própria comparação entre estruturas geradas pelos dois tipos de linguagens pode ser destacada como outra contribuição do trabalho.

Como conseqüência da comparação, foi proposto um metamodelo estrutural genérico (Muchaluat-Saade, 2001c) capaz de representar tanto estruturas de documentos hipermídia como estruturas de arquitetura de software. Baseado nesse metamodelo, foi feito um refinamento do modelo hipermídia NCM, incorporando o conceito de conector, que pode, inclusive, representar uma estrutura composta por nós e elos. Com isso, o modelo NCM oferece agora composição não só em seus nós, como antes, mas também composição em suas relações expressas por elos. Outro refinamento importante no modelo NCM, conseqüência da aplicação do metamodelo estrutural, foi tornar o modelo composicional. A introdução do conceito de portas e mapeamentos em nós de composição impede que um elo atravesse os limites de uma composição e continua mantendo o mesmo poder de expressão anterior do NCM. Dessa forma, pode-se considerar como contribuições da tese a definição do metamodelo estrutural e o refinamento do modelo NCM com a introdução de conectores, conectores compostos, portas e mapeamentos.

Com a introdução do conceito de conector, tornou-se necessária a definição de uma linguagem para autoria de conectores. Visto a complexidade de relações de sincronização comparada à complexidade dos outros tipos de relações que uma linguagem hipermídia deve dar suporte (derivação, tarefas etc.), o enfoque do trabalho foi direcionado à definição de relações espaço-temporais. Daí surgiu a linguagem declarativa XConnector (Muchaluat-Saade, 2002a) para autoria de conectores representando relações de sincronização multiponto complexas com semântica causal e de restrição. A expressividade das relações causais de XConnector é similar a dos elos NCM, em sua versão anterior (Soares, 2000a),

sendo que a definição de relações de restrição foi bastante refinada. A especificação de XConnector se refletiu em novos refinamentos na definição de conectores na nova versão do modelo NCM. Sendo assim, pode-se salientar como contribuições a especificação da linguagem XConnector e o refinamento da definição de relações espaço-temporais no modelo NCM.

Após o desenvolvimento da linguagem XConnector, usada para especificar relações através de conectores hipermídia, usados, por sua vez, na criação de elos, surgiu a idéia de propor extensões ao padrão recente XLink do W3C, recomendado para a criação de elos. A extensão proposta a XLink (Muchaluat-Saade, 2002a) incorpora as facilidades de XConnector, capacitando a linguagem para a definição de relações de sincronização entre recursos na Web. A proposta de extensão de XLink com definições de XConnector é outra contribuição da tese.

Após o desenvolvimento da linguagem para autoria de conectores hipermídia, começou-se a investigar mais detalhadamente o conceito de estilos arquiteturais em ADLs e sua possível aplicação no domínio hipermídia, já identificada na comparação inicial entre ADLs e linguagens hipermídia. A partir daí, iniciou-se o desenvolvimento do conceito de templates de composição hipermídia, surgindo a necessidade de uma linguagem para autoria de templates de composição. Como conseqüência, foi desenvolvida a linguagem XTemplate (Muchaluat-Saade, 2002b). Pode-se salientar como outra contribuição principal da tese a introdução do conceito de template de composição hipermídia, como alternativa para especificação de semântica de uma composição (Muchaluat-Saade, 2002c). Apesar de, atualmente, a semântica, que um template pode definir, estar associada a relações de sincronização, o conceito de template é mais genérico e poder ser aplicado com outra semântica.

Com a introdução do conceito de templates em linguagens de autoria hipermídia, mais refinamentos foram realizados no modelo NCM. As composições com semântica de apresentação paralela e seqüencial, introduzidas na versão anterior do modelo (Rodrigues, 2000), não são mais necessárias, pois qualquer composição NCM pode adquirir uma semântica qualquer com o uso de um template. Com isso, o modelo se tornou mais simples e mais geral, deixando a semântica de uma composição sob responsabilidade da linguagem de autoria

baseada no modelo (Soares, 2003). Esse outro refinamento no modelo NCM é mais uma contribuição da tese.

Usando idéias de XTemplate, uma nova proposta de extensão a XLink foi realizada, incorporando algumas facilidades de templates de composição (Muchaluat-Saade, 2002b), aumentando o reuso em especificações XLink. Essa segunda proposta de extensão pode ser apontada como outra contribuição da tese.

Como conseqüência do estudo sobre a recomendação XLink e o tratamento de elos em outros modelos hipermídia, mais um refinamento foi realizado no modelo NCM, contemplando a definição de bases de elos e do reuso de elos e bases de elos em documentos distintos. Com isso, uma restrição imposta por versões anteriores do modelo NCM (Casanova, 1991; Soares, 1995; Soares, 2000), que obrigava que as extremidades de um elo fossem nós recursivamente contidas na composição que contém o elo, não é mais colocada como restrição do modelo e sim da linguagem de autoria baseada no modelo. É óbvio que se essa restrição não for obedecida, a propriedade de composicionalidade é perdida. Entretanto, essa nova definição torna o NCM mais geral e permite modelar documentos web na íntegra.

Como objetivo de validar as propostas feitas por esta tese, principalmente introduzindo o conceito de conectores e templates de composição em linguagens de autoria hipermídia, uma nova versão da linguagem declarativa NCL, baseada no modelo NCM, foi desenvolvida. A linguagem NCL 2.0 apresenta uma estrutura modular, seguindo recomendações recentes do W3C, e oferece todas as facilidades para definição de conectores e templates de composição, além de refletir todos os refinamentos feitos no modelo NCM (Soares, 2003). Considerando os trabalhos futuros indicados na especificação da primeira versão de NCL (Antonacci, 2000a), NCL 2.0 atingiu todos os objetivos, ou seja, foi especificada de forma modular, o que tornou a linguagem extensível; introduziu maior facilidade de autoria, com o uso de conectores e templates; e aumentou ainda mais o reuso de especificações prontas, inicialmente contemplado apenas para nós hipermídia, e agora permitido para elos, bases de elos, conectores e templates. Assim sendo, uma outra contribuição da tese foi a definição da linguagem de autoria hipermídia NCL 2.0.

7.3
Trabalhos Futuros

A linguagem XConnector, proposta nesta tese, concentrou seus esforços na definição de relações de sincronização e referência, entretanto, o conceito de conector pode ser aplicado para definir qualquer tipo de relação. Por exemplo, pode-se definir um conector de versão para representar a relação ancestral/descendente usada em mecanismos para controle de versões de objetos, ou então um conector canal para representar o paradigma *publish-subscribe* usado em mecanismos de notificação de mensagens (Muniz, 2000). Baseando-se em novos tipos de conectores, a linguagem XTemplate pode ser usada para definir outras semânticas para composições, diferentes da semântica de sincronização temporal e espacial já disponível. Por exemplo, pode-se criar templates de composição para agrupar versões ou variantes (Soares, 1994; Soares, 1999) de um mesmo objeto e seu grafo de derivação. O desenvolvimento de outros módulos de linguagem para criação de outros tipos de relação é um trabalho futuro.

Foram propostas extensões ao padrão XLink incorporando facilidades de XConnector e XTemplate. Trabalhos futuros podem utilizar XConnector e XTemplate para estender funcionalidades em outras linguagens de autoria, por exemplo, implementando os perfis XHTML+XConnector, XHTML+XTemplate e SMIL+XTemplate.

Apesar de XConnector permitir a parametrização de valores de atributos de um conector, essa facilidade ainda não está disponível em XTemplate. Um trabalho futuro é identificar o que poderia ser parametrizado em XTemplate, e refinar a linguagem para dar o suporte necessário à definição de parâmetros. Além disso, é necessário implementar o suporte à definição de parâmetros de conectores no sistema HyperProp.

Considerando os refinamentos introduzidos por este trabalho no modelo NCM, onde destaca-se a entidade conector acrescentada ao modelo para representar relações, todo o controle de versões definido pelo modelo NCM precisa ser revisto. Provavelmente, será necessário realizar o controle de versões para conectores, e avaliar o impacto dessas modificações na autoria de elos e propagação de versões. A revisão do mecanismo de versões do NCM é um trabalho futuro importante.

Considerando que XConnector permite a definição de bibliotecas de conectores e que uma biblioteca pode ser utilizada por usuários diferentes daquele que a construiu, tem-se a necessidade de descrever semanticamente um conector, permitindo o seu entendimento por alguém ou algo que não conheça os detalhes da linguagem XConnector, mas que precise utilizá-lo para criar um relacionamento em seu documento. Visto essa necessidade, um trabalho futuro bastante interessante seria a descrição semântica de um conector, utilizando o padrão RDF (W3C, 1999a).

A definição de um estilo arquitetural em ADLs permite a prova de propriedades com base em sua especificação formal. Um trabalho futuro bastante interessante é a análise de consistência de documentos considerando o uso de templates, além da análise de consistência dos próprios templates, que seria uma continuação do trabalho inciado por (Félix, 2002).

Outro trabalho futuro bastante necessário é uma ferramenta para autoria de templates de composição, que ajude o usuário a especificar um template, principalmente no que se refere à definição de expressões XPath, usadas para selecionar componentes de uma composição.

Outro trabalho futuro é o desenvolvimento de ferramentas genéricas para visualização e edição gráfica de estruturas baseadas no metamodelo apresentado na Seção 2.3. O sistema HyperProp possui um editor estrutural gráfico que oferece mecanismos de filtragem de informações para estruturas compostas (Muchaluat-Saade, 1996a; Pinto, 2000). Tornar o editor independente da semântica do modelo em questão permitiria o seu uso para visualizar e editar qualquer estrutura composta, seja ela um documento hipermídia, uma descrição arquitetural de um software, um projeto de um sistema de comunicação etc. Para atingir esse objetivo, é necessário implementar o conceito de conectores compostos no sistema HyperProp, que ainda não está operacional.

Um trabalho futuro que tem que ser realizado é implementar um parser para documentos seguindo o perfil completo da linguagem NCL 2.0. Como salientado na Seção 3.4, a implementação atual só contempla o perfil *Simple NCL Profile*, que não engloba todos os módulos da linguagem. Além dos parsers, o suporte necessário à manipulação de tais documentos no sistema HyperProp também precisa ser desenvolvido.

A linguagem NCL 2.0, apesar de ter acrescentado muitas novas facilidades em relação à versão 1.0 da linguagem, com certeza, precisa ser freqüentemente revista e refinada, se necessário. A especificação de novos módulos de NCL 2.0 para permitir a definição de funções de custo para durações de componentes de um documento, possibilitando a realização de ajustes finos pelo formatador (Rodrigues, 2003), é colocada como trabalho futuro. Além disso, o suporte para computação automática de extremidades de elos, contemplando os antigos elos virtuais do modelo NCM (Soares, 2000a), precisa ser construído.

Um trabalho futuro bastante interessante e útil é contemplar a definição de templates no sistema de autoria gráfico do HyperProp. Na implementação atual, o uso de templates de composição só está disponível no modo declarativo, e mesmo assim com algumas limitações, como comentado no Capítulo 6. Além disso, é necessário refinar a implementação do processamento de templates, mesmo no modo declarativo, para contemplar documentos utilizando vários templates. O suporte completo à definição de templates no modo de autoria gráfico exige o desenvolvimento de uma nova integração entre as visões estrutural e temporal do sistema HyperProp, considerando a apresentação temporal de estruturas compostas com a semântica definida por um template.

Outro trabalho futuro é refinar a linguagem XTemplate para permitir a especificação de templates aninhados e a definição de características de apresentação de objetos gerados automaticamente pelo template.

Um outro trabalho futuro é integrar a autoria declarativa de documentos NCL com o modo gráfico de autoria do sistema HyperProp, acoplando um editor XML ao sistema e permitindo que o autor alterne o modo de autoria dinamicamente durante a edição de partes de um documento. O primeiro passo para integrar os dois modos de autoria é implementar os conversores de objetos NCM em Java para a representação textual em NCL 2.0, considerado também como trabalho futuro.

Um outro trabalho futuro interessante é o estudo de como extrair características comuns de uma coleção de documentos a fim de representá-las como templates. Esse tipo de estudo poderia ajudar na geração automática de documentos hipermídia a partir de uma base de objetos de mídia inter-relacionados.

Outro trabalho futuro é levar algumas contribuições de hipermídia para o domínio de arquitetura de software. Como comentado na Seção 2.6, algumas soluções propostas para o domínio hipermídia talvez pudessem ser aplicadas ao domínio de ADLs com vantagens, seguindo uma abordagem no sentido inverso ao que foi desenvolvido nesta tese. Certamente, os conceitos de transformação de documentos e adaptabilidade de documentos poderiam trazer contribuições para arquitetura de software, no entanto, esse é um trabalho futuro bastante extenso e que deve ser feito de forma bem cuidadosa.