2
Autoria em Nested Context Language

Este capítulo apresenta uma breve introdução aos conceitos de NCL na Seção 2.1. O estudo empírico realizado com aprendizes de NCL é apresentado na Seção 2.2 e o estudo analítico sobre reúso em NCL é apresentado na Seção 2.3. Ambos os estudos focam no emprego da NCL para a concepção de aplicações para TV digital interativa, um caso particular de uso da NCL. Apesar desse foco mais específico, ambos os estudos são abrangentes de forma a poderem ser generalizados para a autoria de documentos hipermídia. Isso fica claro no encerramento do capítulo, onde é feito um levantamento dos requisitos apontados por esses estudos para uma linguagem de especificação de famílias de documentos, na Seção 2.4.

2.1.
Introdução à NCL

NCL é uma linguagem declarativa de domínio específico (DSL) (Deursen *et al.*, 2000) voltada para a especificação de documentos hipermídia. NCL é a linguagem declarativa padrão do Sistema Brasileiro de TV Digital (SBTVD-T) (ABNT, 2007), suportada pelo *middleware* Ginga (Soares, 2008). NCL e Ginga são parte do padrão ISDB-T_B (*International Standard for Digital Broadcasting*) para TV digital terrestre e ainda Recomendação ITU-T para serviços IPTV (ITU-T, 2009).

NCL expressa o modelo NCM (*Nested Context Model*) (Soares, 1995). NCM, por sua vez, é um modelo conceitual de dados centrado no reúso e expressividade por meio do emprego adequado de composições de objetos. Pelo modelo, o sincronismo entre objetos de mídia (vídeo, áudio, imagem, texto, etc.) é expresso no paradigma causalidade-restrição. Relações de causalidade são aquelas baseadas em causa e efeito, ou seja, dado que certas condições são satisfeitas, um conjunto de ações deve ser efetuado. Relações de restrição são aquelas que asseguram certas propriedades aos objetos de mídia como, por exemplo, que dois

objetos específicos sempre iniciem e terminem sua apresentação juntos. O perfil NCL para a TV digital interativa, entretanto, faz uso apenas de relações causais do modelo NCM.

Apresentando brevemente os principais elementos de NCL, a estrutura geral de um documento é dividida em <head> e <body>, como usual em padrões W3C. O <head> contém todas as bases de informação (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> e <connectorBase>) cujos elementos filhos são alvo de reúso pelos elementos filhos do <body>.

O elemento <body> contém elementos filhos <media>, <context>, <switch> e <link>. Um elemento <media> define um objeto de mídia especificando seu tipo e sua localização de conteúdo. O elemento <context> define um objeto de contexto. Um objeto de contexto é uma composição que contém um conjunto de objetos (mídia, contexto ou de alternativa) e um conjunto de elos (definindo relacionamentos entre objetos). O elemento <switch> permite a definição de objetos alternativos (representados por elementos filhos <media>, <context> e <switch>) a serem escolhidos em tempo de exibição do documento. Regras de teste usadas na escolha dessa alternativa são definidas pelos elementos <rule> ou <compositeRule>, presentes na parte <head> do documento.

Interfaces de objetos são usadas em relacionamentos com outras interfaces de objetos. O elemento <area> permite a definição de uma âncora de conteúdo representando um segmento espacial, temporal ou espaço-temporal do conteúdo de um objeto de mídia, e ainda um trecho de código em objetos de mídia imperativos ou com código declarativo. O elemento <property> é usado para a definição de uma propriedade de um objeto (uma variável local) ou um grupo de propriedades de objetos como uma das interfaces do objeto. O elemento <property> também é usado para a definição de variáveis globais, quando é declarado como filho de um objeto de mídia de tipo específico: o tipo “application/x-ncl-settings”. O elemento <port> de um <context> permite externar uma interface de qualquer um de seus objetos filhos internos.

A informação temporal e espacial necessária para iniciar as propriedades necessárias na apresentação de cada objeto de mídia pode ser definida por elementos <descriptor>. O elemento pode referenciar um elemento <region> para definir a posição inicial de um elemento <media> em algum dispositivo de saída. Um elemento <descriptor> pode ainda referenciar um elemento <transition> para

definir efeitos de transição no início ou término da apresentação de um objeto de mídia. A parte <head> de um documento inclui a definição dos elementos <descriptor>, <region> e <transition>.

Um elemento <causalConnector> representa uma relação que pode ser usada criando elementos <link>. Em uma relação causal, um papel de condição deve ser satisfeito a fim de disparar um papel de ação. Um elemento <link> associa as interfaces dos objetos a papéis de conectores (condições ou ações), definindo um relacionamento espaço-temporal entre objetos.

2.2. Estudo empírico com NCL

O objetivo deste estudo foi obter indicadores da usabilidade da NCL no contexto da criação de documentos hipermídia, em especial levando em consideração o desenvolvimento de aplicações interativas para TV digital (um caso particular de uso da linguagem). Através de coleta de dados realizada por meio de questionários preenchidos por alunos de três turmas de cursos de treinamento sobre o Middleware Ginga, e de posterior análise por um misto de métodos qualitativos e quantitativos (Cresswell, 2003), descobriram-se pontos em que o uso da NCL pode e deve ser aprimorado para facilitar as atividades de especificação de documentos.

Considerando ainda o emprego de NCL no ambiente de desenvolvimento de aplicações interativas para TV digital, é conveniente ressaltar que há três grupos de usuários a quem a NCL se destina: i) programadores de software, que tendem a desenvolver aplicações mais complexas, até mesmo residentes nos conversores de TV digital, fazendo uso de todas as funcionalidades da linguagem, incluindo sua linguagem de script Lua (Lua.org, 2010); ii) produtores de conteúdo de TV, para os quais o foco é o desenvolvimento de aplicações visualmente ricas, geralmente transmitidas pela emissora de TV por radiodifusão; e iii) os próprios telespectadores, que constroem aplicações simples em seu próprio receptor, com base em facilidades gráficas e simplificações no uso da linguagem. O foco deste estudo empírico reside no perfil do produtor de conteúdo, que é atraído para a linguagem em função da possibilidade de seu uso, mesmo sem uma forte experiência técnica de desenvolvimento de aplicações para computador. Contudo,

como é mencionado muitas vezes, tal produtor pode ter também um razoável conhecimento de programação.

Resumidamente, para o caso da TV digital, o foco declarativo de NCL deve dar suporte a três características importantes: o sincronismo de mídias (e não apenas a interatividade, que é tratada como um caso particular de sincronismo temporal); a adaptabilidade; e o suporte a múltiplos dispositivos. NCL atua como uma linguagem de cola para os diversos conteúdos de mídia que compõem uma aplicação, provendo as três características essenciais citadas.

A Figura 1 exemplifica os requisitos. No quadro (a), uma propaganda da pilha aparece no momento em que o personagem do filme a exibe, um exemplo de sincronismo temporal sem a interação do usuário. Note que a propaganda poderia ser de acordo com cada usuário ou local onde o usuário se encontra, por exemplo, fazendo a propaganda do local de compra o mais perto da localização do telespectador, sendo, assim, também um exemplo de adaptação de conteúdo. No quadro (b), um ícone aparece sobre os óculos da personagem. Quando selecionado pelo usuário (botão amarelo do controle remoto), uma propaganda dos óculos e um formulário para compra são apresentados, como visto no quadro (c). Esse é um exemplo de sincronismo temporal com a interação do usuário (ocorrência no momento da seleção). A apresentação da propaganda e do fórmulário poderia ser realizada em outro dispositivo, por exemplo, no celular de quem fez a seleção, de forma a não perturbar outros telespectadores localizados no mesmo ambiente, como exemplo de uso de múltiplos dispositivos de exibição, como visto no quadro (d).

(a) Sincronismo de mídias

(b) Ícone de interação

(c) Interatividade do usuário na TV, ou (d) no celular.

Figura 1. Aplicativo de TV com sincronismo de mídias e interatividade do usuário.

2.2.1. Especificação de Conteúdos Digitais com NCL: Perfil de Usuários e Tarefas

O profissional responsável pelo desenvolvimento de aplicações para TV digital interativa deve ter alguma familiaridade com áreas diversas, como design de aplicações, criação, arte e programação. Aplicações para TV têm claras peculiaridades, quando comparadas a aplicações para computador. Como exemplo, a distância entre o telespectador e a tela de TV tende a ser bem maior, gerando a necessidade de aplicações com pouco texto para ser lido e mais centradas no conteúdo audiovisual.

As tarefas de um programador de aplicações interativas para TV digital (ou apenas aplicações DTV) começam com base em artefatos gerados pelo roteirista, como o *storyboard* ou o roteiro não-linear, e transpassa várias fases de produção de conteúdo. Na prática, o roteiro não é necessariamente mais uma linha única no tempo e sim uma cadeia principal de eventos que pode ter várias cadeias secundárias. Como exemplo simplista, imagine um programa com um vídeo inicial em que o telespectador tem que tomar uma decisão. Dependendo de sua escolha, o telespectador assiste a um de dois finais disponíveis. O roteiro de um programa desta forma deve levar em conta as várias linhas de tempo possíveis.

Um ponto-chave para este estudo empírico foi avaliar de que forma esse profissional descreve naturalmente uma dessas aplicações de TV. A tendência é que quanto mais próxima a NCL estiver dessa descrição, mais fácil e mais livre de erros será sua utilização.

Devido às dimensões descritivas privilegiadas pela NCL, uma premissa deste estudo empírico foi a de que aplicações interativas são descritas por frases

construídas a partir de associações entre causa e efeito. A aplicação da Figura 1, por exemplo, poderia ser descrita por frases como: “quando o ator mostra uma pilha, aparece uma propaganda no canto inferior direito”; e “se o telespectador apertar a tecla amarela do controle remoto, então o vídeo principal é redimensionado para o canto esquerdo e um vídeo de merchandising é exibido abaixo dele com um formulário à direita para compra de óculos”. Essa é uma premissa para este estudo, mas sua validade na modelagem do mundo real deve ser verificada.

O paradigma de causalidade/restrição de NCL é muito próximo da descrição em linguagem natural de relacionamentos entre objetos, como as exemplificadas no parágrafo anterior. Conforme já mencionado, o sincronismo de mídias e interatividade do usuário são descritos em NCL por meio de elos que, por sua vez, são descritos por sentenças com uma condição (simples ou composta) e uma ação resultante (simples ou composta). Ainda se referindo à aplicação da Figura 1, um elo poderia descrever a sentença “quando no vídeo os óculos escuros do ator principal estiverem em foco, um ícone representando o botão amarelo do controle remoto deve aparecer”. Usando a terminologia NCL, essa aplicação é composta por um objeto de mídia representando o vídeo, que possui uma âncora delimitando o intervalo de tempo em que é dado foco nos óculos do ator, e ainda possui um objeto de mídia ligado à imagem do ícone amarelo. Um elo associa a condição de início da âncora do vídeo com a ação resultante de exibição da imagem amarela.

No estudo realizado, foram coletados dados de alunos de três cursos introdutórios sobre o desenvolvimento de aplicações DTV. Os cursos foram direcionados a produtores de conteúdo para TV, cobrindo diferentes perfis de alunos. A coleta foi feita por meio do preenchimento de formulários e complementada com a gravação em vídeo das aulas dos dois primeiros cursos. Os três cursos são identificados neste estudo com os números de 1 a 3, de acordo com a ordem cronológica em que foram aplicados.

Como já mencionado, o objetivo do estudo foi traçar indicadores da facilidade de uso da NCL em tarefas típicas de autoria de programas DTV. Adicionalmente, o estudo avaliou as dificuldades e facilidades no aprendizado de NCL e as primeiras impressões de públicos diferentes. O estudo também avaliou os pré-requisitos técnicos para a aprendizagem de NCL, em especial entre produtores de conteúdo para TV.

2.2.2.
Metodologia do Estudo Empírico

Em todos os cursos, os alunos preencheram dois formulários principais: o de "Perfil dos Alunos" e o de "Avaliação do Curso e da NCL". O primeiro sempre foi aplicado no início do segundo dia, tendo como objetivo traçar o perfil profissional e de formação dos participantes, como é apresentado na próxima subseção. O segundo foi sempre aplicado no último dia, com o objetivo de extrair a opinião dos alunos sobre o aprendizado e as ferramentas apresentadas.

As perguntas dos alunos foram transcritas, em sua maioria, e forneceram uma base adicional de informações sobre as principais dúvidas de aprendizagem e de conceituação impostas pela linguagem. Por dificuldades técnicas, não foi possível anotar as dúvidas dos alunos do Curso 3, já que foi inviável fazê-lo em todas as cidades em que o curso foi realizado simultaneamente.

Os cursos diferiram quanto à carga horária e perfil dos alunos e instrutores, mas tiveram o mesmo formato de apresentação do conteúdo. As primeiras 8 horas serviram de introdução para os principais conceitos de TV digital, de aplicações interativas e apresentaram o modelo conceitual NCM. No restante do curso, esse conteúdo foi aplicado na prática, por meio da autoria de pequenos programas de exemplo com cunho notadamente didático. Nessa parte, os cursos focaram os aspectos mais relevantes para o público de alunos presente. Em todos os casos, porém, os exemplos foram criados e alterados pelos alunos com um editor de textos convencional, sem uso de nenhuma ferramenta de autoria específica de NCL. Apenas no último dia de curso, uma ferramenta de autoria gráfica e facilitadora foi apresentada, em no máximo quatro horas de uso. A coleta de dados foi feita por um pesquisador em sala de aula, com consentimento dos alunos e dos instrutores por escrito.

2.2.3.
Caracterização do perfil das turmas de alunos

Com base no formulário de "Perfil dos Alunos", pode-se dizer que cada curso tem particularidades que enriqueceram o estudo empírico.

O Curso 1 foi formado por uma turma inicialmente de 28 alunos, dos quais 23 tiveram seus dados analisados, pois participaram de todas as atividades e preencheram ambos os formulários de coleta de dados. Com relação aos pré-requisitos sugeridos do curso, 22 alunos apontaram experiência com linguagens de marcação, 14 disseram ter experiência com produção de conteúdo e 15 com edição de conteúdo digital. A turma é marcadamente de nível superior de formação, com 21 alunos graduados em alguma área afim de Informática. Os alunos foram perguntados sobre seu perfil profissional, indicando sua área de trabalho em vários tópicos. Reforçando a caracterização da turma como composta em sua maioria por profissionais com experiência em programação, 21 alunos apontaram trabalhar com desenvolvimento de aplicações, 12 com tecnologias *web*, 5 com desenvolvimento de software básico, 3 com produção de conteúdo multimídia, 1 com ensino à distância e nenhum com produção de conteúdo para TV. Podem-se caracterizar os alunos do Curso 1 como profissionais com formação qualificada e especializada, e com boa experiência no desenvolvimento de aplicações para computador, embora sem experiência, salvo raras exceções, no uso de linguagens declarativas baseadas no tempo (como SMIL, NCL, SVG etc.). Como se avalia adiante, houve uma acentuada curva ascendente de aprendizagem para esse perfil. A carga horária total foi de 24 horas, sendo 8 por dia.

O Curso 2 foi composto por 23 voluntários de todas as regiões do país, ligados à produção de conteúdo audiovisual em comunidades de baixa renda, ou a projetos não-governamentais de alfabetização digital e de software livre. Tais alunos funcionaram como multiplicadores lecionando no Curso 3. A diferença clara de perfil formou uma turma mista com dois grupos característicos: por volta de metade dos alunos com experiência no desenvolvimento de aplicações, mas inexperientes na produção de conteúdo para TV; e outra metade de alunos com conhecimento mais ligado à produção audiovisual, mas usuários de computador não-programadores. Essa dualidade de perfis em uma mesma turma foi importante para avaliar como se consolida o conhecimento e aprendizagem de NCL entre não-programadores quando comparado a programadores. A carga horária do Curso 2 foi de 40 horas, divididas igualmente em cinco dias de aulas.

O Curso 3 foi uma iniciativa de democratização do acesso ao conhecimento ligado ao Ginga para comunidades de baixa renda produtoras de conteúdo audiovisual (Pontos de Cultura, Telecentros, Casas Brasil etc.). Diferente dos

outros dois cursos, o Curso 3 foi formado por várias turmas em diferentes cidades de mais de dez estados do Brasil. Outra diferença foi a carga horária total variada, com um mínimo de 30 horas, e formatos também diversos de carga horária diária. De um total de 174 alunos, 112 tinham a escolaridade máxima de ensino médio ou curso técnico. Em alguns estados, a maioria dos alunos do Curso 3 foram encaminhados a partir de projetos de alfabetização digital. Em muitos casos, os alunos possuíam apenas uma base prática de uso do computador, sem conhecimento algum de programação de computadores.

Levando em conta o nível e tipo de dúvidas dos alunos e a opinião dos instrutores, o Curso 1, conforme esperado, dada a maior escolaridade, pareceu ser aquele com maior aproveitamento. Os alunos chegaram a sugerir contribuições interessantes ao uso da linguagem. Quando foram perguntados sobre o grau de dificuldade no entendimento da linguagem, o resultado foi: nenhum aluno apontou muita dificuldade, dez indicaram média dificuldade e outros doze pouca dificuldade.

O Curso 2 teve um desempenho equiparável ao Curso 1, que foi medido quase um mês depois, quando os alunos multiplicaram o conhecimento. É importante ressaltar, no entanto, que os alunos do Curso 2 precisaram de bem mais tempo para consolidar o conhecimento adquirido, o que pode ser constatado pela sua presença constante no fórum da comunidade de software livre do Ginga no mês seguinte.

Não é possível avaliar com precisão o desempenho dos alunos do Curso 3, dado o caráter distribuído das aulas e a quantidade de instrutores, mas a impressão que se tem, com base exclusivamente nas respostas dos formulários, é a de um desempenho geral inferior às demais turmas, o que não parece ser surpresa, dado o perfil de alunos e instrutores.

2.2.4. Análise e Interpretação de Dados

A avaliação dos dados coletados foi dividida em alguns tópicos, que são apresentados nesta subseção. Apesar de parecer fragmentar a discussão, a separação em itens autocontidos ajuda a conduzir a interpretações mais diretas e relacionar mais facilmente aos dados.

2.2.4.1.
Similaridade de NCL com conceitos prévios

Foi feita a seguinte pergunta, para resposta livre, no formulário de avaliação: “Você poderia comparar a NCL e alguma outra linguagem de seu conhecimento? Qual? Que comparação você faria?”.

Com relação aos alunos do Curso 1, sete deles informaram desconhecer uma linguagem parecida. Seis lembraram HTML/XHTML (W3C, 2002), quatro alunos associaram a XML (W3C, 2004), dois a SMIL e, outros dois lembraram, cada um, XMT-O e Pascal Estruturado.

A ligação que os alunos fizeram a HTML parece se justificar pela sua grande popularidade. No entanto, HTML não se destina ao sincronismo de mídias, mas sim para a publicação de hipertexto na web. Na verdade, SMIL, que foi bem menos lembrada, é efetivamente o padrão W3C para sincronismo de mídias. A associação com XML, que é uma recomendação W3C para linguagens de marcação, também é natural, já que NCL é uma aplicação XML. É importante ressaltar o desconhecimento quase total da existência de linguagens declarativas com o foco no sincronismo de mídia, mesmo em um curso onde o conhecimento de programação era grande. Isso sugere uma dificuldade de aprendizado de NCL por um processo de associação com outras linguagens.

HTML não é adequada para a estruturação de documentos multimídia e sim para refletir o modelo de apresentação de hipertextos. É importante salientar que o processo de autoria em HTML difere fundamentalmente do NCL, porque o primeiro está ligado mais diretamente ao conteúdo, que é enriquecido com formatação textual e com mídias externas. Uma especificação NCL, por sua vez, se mantém separada de todo conteúdo e, assim, foca mais a estrutura de composição, os relacionamentos e a apresentação desse conteúdo. As implicações disso para o autor são discutidas em uma subseção própria adiante.

Enfim, o conhecimento de SMIL parece ser realmente o mais valioso entre os citados. SMIL é também uma linguagem de sincronismo de mídias. Em sua versão 3.0, NCL chega a fazer uso de módulos SMIL, o que demonstra a proximidade das duas linguagens. Uma diferença crucial na forma de expressar documentos multimídia em SMIL e em NCL é que na primeira o sincronismo é

dado primordialmente pela composição dos objetos de mídia, enquanto na segunda ele é dado pelo uso de elos.

Os dados do Curso 1 ofereceram mais riqueza de informações para a avaliação de similaridade. Nos outros cursos, o retorno dos alunos, em sua maioria, foi restrito a comparações com HTML e eventualmente com XML.

Com a falta de conhecimento correlato, que é sustentada nos dois parágrafos seguintes, é justificável que os alunos dos Cursos 2 e 3 não tenham apresentado muitas associações, salvo principalmente com HTML, que é a mais conhecida e, com base nos dados, o principal primeiro contato com linguagens de marcação desses alunos.

No Curso 2, pelo menos a metade dos alunos apontou nenhuma experiência com linguagens de programação como ASP, C, C++, Delphi/Pascal, PERL, PHP ou Visual Basic. Com relação à Lua, vinte alunos apontaram nenhuma experiência e outros três se enquadraram em pouca ou média. Todos os alunos desconheceram uma das principais ferramentas de autoria do SMIL. Por outro lado, no quesito HTML, apenas um aluno apontou nenhuma experiência, sete apontaram pouca, outros sete média e mais oito se disseram com muita experiência ou especialistas. Com relação a XML, há a seguinte distribuição de experiência: sete alunos com nenhuma; quatro com pouca; quatro com média; sete com muita; e um se diz especialista.

O mesmo questionário no Curso 3 caracteriza indicadores gerais de ainda menos experiência correlata. Entre 167 alunos que o responderam, pelo menos 118 se disseram sem nenhuma experiência com as linguagens de programação citadas no parágrafo anterior. Nenhum aluno se disse especialista em uma delas e no máximo nove alunos disseram ter muita experiência em alguma. Lua foi desconhecida por 166 alunos e um único disse ter pouca experiência. Uma das principais ferramentas de autoria do SMIL era desconhecida por 158 alunos. HTML tem os seguintes indicadores: 54 alunos com nenhuma experiência; 50 com pouca; 49 com média; 12 com muita; e 2 se dizem especialistas. Os indicadores sobre XML, por parte dos alunos, são de 115 sem nenhuma experiência; 33 com pouca; 16 com média; 2 com muita; e 1 especialista.

Ainda reforçando a dificuldade dos alunos do Curso 3 em imprimirem associações, três alunos mostram inclusive uma expectativa equivocada, quando perguntados no início do curso sobre o que esperavam aprender: “Saber a base

sobre HTML e alguns programas da Web."; "Programação em código, inclusive html, visto que preciso fazer sites"; e "Quero concluir meus conhecimentos em Web designer, HTML e outros tipos de desenvolvimento de mídia".

Em suma, para os alunos deste estudo, há algumas linguagens que remetem à NCL. No entanto, parece que a similaridade identificada está mais relacionada com o fato das linguagens apontadas terem a mesma base sintática (marcação) que NCL do que uma comparação mais profunda com a semântica das construções das linguagens. Há pelo menos o efeito colateral benéfico de XHTML (inclua-se HTML) e SMIL fornecerem a proficiência com linguagens de marcação primária para a edição de documentos em NCL.

2.2.4.2. Pré-requisitos para aprendizagem NCL

No Curso 1, com um público conhecedor de linguagens de marcação, apenas um aluno citou a necessidade de conhecimento prévio de XML ao responder se teve algum tipo de dificuldade: "Não, mas conhecimento de XML é bom (necessário)". No Curso 2, houve uma clara preocupação por parte dos alunos com relação à aprendizagem prévia de linguagens de marcação tanto entre eles próprios quanto futuramente para o público-alvo de média escolaridade do Curso 3. Alguns comentários reforçam isso:

> "O curso está bem estruturado considerando-se um público com conhecimentos básicos em informática e em linguagem de marcação. No entanto, para um público completamente externo à Computação, o curso deve prever uma iniciação em linguagens de marcação (de preferência com exemplificação através de HTML, já que todos sabem o que é uma página Web)";
>
> "Acredito que quando o curso for dado para um público que não tiver nenhum conhecimento prévio sobre linguagens de marcação, o mesmo deva incluir uma breve introdução ao conceito usando HTML como estudo de caso";
>
> "(...) Para quem não conhece conceitos de linguagem de marcação fica difícil visualizar as ligações entre os diversos elementos utilizados na NCL (...)".

Dada a explícita preocupação com o assunto, o Curso 3 incorporou um reforço em termos de nivelamento em linguagens de marcação imediatamente antes do início da prática com NCL. A experiência de escrever um conteúdo que seja facilmente interpretado pelo computador é, ainda, um grande desafio não apenas para autores NCL, o que torna explicitamente necessária a inclusão desse pré-requisito seja como parte do curso, seja como exigência para a matrícula no

mesmo. É no mínimo necessário que um aluno entenda naturalmente a forma como os elementos e atributos são escritos e agrupados para então compreender a sintaxe NCL e o modelo por trás da linguagem (sua semântica). A visita dos dados coletados em cada curso, com suas peculiaridades de público, expõe ainda mais essa observação.

No Curso 2, os alunos ligados à produção de conteúdo audiovisual, demonstraram, inicialmente, mais preocupação com a falta de base em programação:

> "Estou saindo com o conhecimento básico para explorar melhor o NCL (que em primeira vista, parecia uma ferramenta apenas para especialistas) e que ao longo da semana perdeu boa parte de seus segredos.";

> "A primeiro modo achava meio difícil mas aos poucos minha opinião foi mudando e ainda vejo uma certa dificuldade. Para um usuário sem conhecimento de programação ficaria meio complicado fazer uma coisa muito avançada(...)".

É curioso observar que a falta de base de programação não se mostrou um desconforto nos comentários dados pelos alunos do Curso 3. Mesmo aqueles alunos do Curso 3 provenientes de cursos técnicos de informática, com base formal de programação de computadores, também não afirmaram consistentemente que essa é uma vantagem significativa no aprendizado de NCL. Há alguns comentários indiretamente relacionados com o assunto, como o abaixo, que aponta a necessidade por uma ferramenta de validação e as dificuldades percebidas nos detalhes da sintaxe de NCL:

> "(...) Para mim o curso foi muito bem aproveitado. Sugiro que vocês arrumem alguma forma de fazer com que o programa localize os erros que a gente comete quando vai fazer as fórmulas no bloco de notas. Pois quando a gente errava alguma vírgula algum traço esquecia de um pequeno detalhe...a gente tinha que procurar linha por linha o que exigia tempo e atenção nisso consistia o tempo que se passava."

O fato da turma do Curso 1 ter tido o melhor desempenho e ser também aquela com mais experiência de programação não é decisiva para concluir que essa é uma proficiência indispensável para um bom aproveitamento. Isso porque os alunos do Curso 1 são também aqueles com maior escolaridade (quase todos pelo menos graduados) e também com anos de experiência profissional. Esses também poderiam ser os fatores de sua melhor resposta.

A experiência com a turma mista do Curso 2 indica que há uma maior demora por parte de quem nunca teve contato com linguagens de marcação para assimilar os conceitos da linguagem quando comparado a experientes no assunto.

O aproveitamento dos alunos do Curso 3 atesta que NCL pode ser diretamente assimilada por usuários finais[4], mas também indica que isso demanda mais tempo para assimilar os conceitos e consolidar o conhecimento com mais prática. O fator tempo é uma tônica recorrente nos comentários de tais alunos, mesmo em perguntas não-relacionadas com o assunto, em que sugerem que a carga horária do curso deveria ser maior. Quando avaliaram os instrutores, 44 alunos fizeram comentários, dos quais 8 pedem pelo aumento da carga horária. Em comentando os conceitos apresentados, 10 alunos fizeram o mesmo de um total de 24. De um total de 44 que comentaram sobre os exercícios propostos, 10 indicaram ser o tempo suficiente, 10 apontaram pouco tempo e 4 novamente sugeriram um aumento do tempo do curso. Num ponto importante do formulário, quando questionados sobre o aprendizado, 37 alunos teceram comentários, dos quais 9 afirmaram a necessidade por ampliar o tempo de curso, 2 acharam que o tempo dos exercícios foi curto e 6 sugeriram que o tempo é suficiente. No desfecho do formulário de avaliação, quando fazem comentários gerais sobre o curso, 52 alunos emitiram seu parecer, com 17 deles focando na preocupação em aumentar a carga horária do curso. Tudo indica que é necessário um tempo maior de maturação no uso da linguagem.

Os dados levantados parecem apontar que NCL não é simples o bastante para poder ser usada diretamente (sem um razoável treinamento) por pessoas que nunca tiveram contato com linguagens de marcação ou programação. No entanto, dados os relatos de satisfação e superação expressos por alguns alunos, em especial os do Curso 3, é razoável supor que a NCL possa vir a ser utilizada por não-programadores. Resta saber se há como torná-la mais fácil ou intuitiva para este público.

[4] O termo "desenvolvimento pelo usuário final" é definido em () como um conjunto de métodos, técnicas e ferramentas que permitem usuários de sistemas de software, que estão atuando como desenvolvedores não-profissionais de software em algum ponto, criarem, modificarem ou estenderem um artefato de *software*.

2.2.4.3.
O modelo NCM e a aprendizagem NCL

No Curso 1, a quantidade de perguntas no primeiro dia apenas sobre NCM (base semântica da NCL) foi praticamente a mesma dos dois outros dias. As perguntas sobre o modelo no Curso 2 também foram numerosas, principalmente por parte dos alunos ligados à comunidade de software livre. Conforme dito anteriormente, não foi viável anotar as perguntas dos alunos no Curso 3.

A Tabela 1 sumariza e destaca nove questões feitas tanto para os alunos do Curso 2 (sempre a primeira linha de respostas, identificadas pelo texto "C2:") quanto aqueles do Curso 3 (a segunda linha, marcada com "C3:"). As questões 1 e 5, apesar de aplicadas em pontos diversos do questionário, têm o mesmo significado, o que é proposital para tentar apontar contradições de opinião de um mesmo aluno, o que não ocorreu.

Em geral, com base na avaliação dos dados da Tabela 1, os alunos concordam ao menos parcialmente que NCM é fundamental para o uso da NCL. Também ao menos parcialmente, eles concordam que assimilaram bem os conceitos de NCM. Nesse aspecto de entendimento do modelo, há uma importante comparação entre os alunos do Curso 2 e do Curso 3: mais ou menos a metade da turma do Curso 2 *concorda totalmente* que o assimilou bem, enquanto no Curso 3 a metade concorda apenas *parcialmente*; além disso, apenas 27.1% dos alunos do Curso 3 *concordam totalmente* que assimilaram bem o modelo, ao passo que são 39.1% dos alunos do Curso 2 que concordam ao menos *parcialmente* que o absorveram.

Ainda em geral, as respostas da última pergunta da Tabela 1 apontam que aproximadamente um terço dos alunos avaliados dos Cursos 2 e 3 *discordam totalmente* que aprender sozinho é melhor do que com instrutores. Outro um terço *concorda parcialmente* com a afirmação. No entanto, seguramente, o primeiro contato com NCL é bem mais comum por meio do uso de material de apoio para estudo próprio, como (Barbosa *et al.,* 2007), do que em cursos formais. A avaliação de aprendizagem desse outro público pode ser um estudo futuro interessante.

Tabela 1. Dados coletados sobre NCM nos Cursos 2 e 3.

Perguntas \ Respostas	discordo totalmente	discordo parcialmente	indiferente	concordo parcialmente	concordo totalmente	prefiro não responder	Total
1 - Consegui aprender bem os conceitos do NCM, no qual a NCL se baseia.	C2: 0.0% (0)	C2: 0.0% (0)	C2: 4.3% (1)	C2: 39.1% (9)	C2: 56.5% (13)	C2: 0.0% (0)	Curso 2: 23
	C3: 2.4% (2)	C3: 2.4% (2)	C3: 8.2% (7)	C3: 52.9% (45)	C3: 27.1% (23)	C3: 7.1% (6)	Curso 3: 85
2 - O entendimento dos conceitos do NCM é fundamental para o uso da NCL.	C2: 0.0% (0)	C2: 8.7% (2)	C2: 4.3% (1)	C2: 21.7% (5)	C2: 56.5% (13)	C2: 8.7% (2)	Curso 2: 23
	C3: 4.7% (4)	C3: 0.0% (0)	C3: 11.8% (10)	C3: 31.8% (27)	C3: 40.0% (34)	C3: 11.8% (10)	Curso 3: 85
3 - O curso apresentou os conceitos do NCM numa profundidade adequada.	C2: 0.0% (0)	C2: 4.3% (1)	C2: 8.7% (2)	C2: 26.1% (6)	C2: 56.5% (13)	C2: 4.3% (1)	Curso 2: 23
	C3: 2.4% (2)	C3: 4.9% (4)	C3: 7.3% (6)	C3: 43.9% (36)	C3: 34.1% (28)	C3: 7.3% (6)	Curso 3: 82
4 - O curso apresentou os conceitos da NCL numa profundidade adequada.	C2: 0.0% (0)	C2: 8.7% (2)	C2: 0.0% (0)	C2: 13.0% (3)	C2: 78.3% (18)	C2: 0.0% (0)	Curso 2: 23
	C3: 3.6% (3)	C3: 4.8% (4)	C3: 1.2% (1)	C3: 38.1% (32)	C3: 47.6% (40)	C3: 4.8% (4)	Curso 3: 84
5 - Entendi bem todos os conceitos do NCM.	C2: 0.0% (0)	C2: 0.0% (0)	C2: 4.3% (1)	C2: 47.8% (11)	C2: 47.8% (11)	---	Curso 2: 23
	C3: 4.8% (4)	C3: 3.6% (3)	C3: 10.7% (9)	C3: 61.9% (52)	C3: 19.0% (16)	---	Curso 3: 84
6 - Entendi bem todos os elementos da NCL.	C2: 0.0% (0)	C2: 0.0% (0)	C2: 0.0% (0)	C2: 52.2% (12)	C2: 47.8% (11)	---	Curso 2: 23
	C3: 3.6% (3)	C3: 7.1% (6)	C3: 6.0% (5)	C3: 53.6% (45)	C3: 29.8% (25)	---	Curso 3: 84
7 - Prefiro aprender sozinho, consultando o material didático.	C2: 34.8% (8)	C2: 13.0% (3)	C2: 26.1% (6)	C2: 26.1% (6)	C2: 0.0% (0)	---	Curso 2: 23
	C3: 36.1% (30)	C3: 12.0% (10)	C3: 10.8% (9)	C3: 34.9% (29)	C3: 6.0% (5)	---	Curso 3: 83

Algumas frases retiradas do formulário de avaliação do Curso 1 merecem atenção. Uma delas remete à dificuldade em relacionar os elementos NCL às entidades NCM. Diz o aluno: “Sim (tive dificuldades). A contradição agregada a certos conceitos e atributos dos elementos da linguagem, que atrapalham seu entendimento.” Note-se a força com que o aluno expressa que certos conceitos não são intuitivos para ele: ele usa a palavra ‘contradição’. Outra frase trata da existência de conceitos que conduzem a raciocínios equivocados. Diz o aluno: “Na verdade, tive dificuldades com a definição dos conceitos, pois determinados termos me remetiam a definições que eram equivocadas com as propostas”. De

novo, embora com menos força, aparece a dificuldade de entender conceitos que não são o que parecem (‘definições que eram equivocadas com a proposta’).

Quanto ao segundo comentário, que teoriza que a terminologia da linguagem remete a conceitos equivocados, ele parece expressar uma opinião de um grupo isolado. Por um lado, outro aluno (o mesmo do primeiro comentário) tem opinião parecida nesse aspecto: “Uma linguagem com recursos e aplicações bem interessantes, porém ainda pouco desenvolvidos, com recursos muitas vezes não intuitivos ou mal definidos”. Entretanto, vários outros alunos do Curso 1 emitiram opinião diferente (e inversa), como nesses comentários: “Uma boa linguagem, mas seu principal diferencial é usar conceitos já conhecidos de hipermídia e XML. Outra característica importante é a facilidade.”; “(...) Muito bem estruturada logicamente.”; e “A linguagem é de fácil entendimento e didática”. Na verdade, o modelo NCM é formalmente definido, o que não dá margem a ambigüidades. Os comentários, no entanto, ajudam a avaliar se as primeiras impressões sobre NCM são claras e naturais. Mas, é sintomático que tenham sido alunos do Curso 1, o de participantes mais preparados tecnicamente, os que não tiveram problemas de ‘intuição’ com o modelo.

A metodologia de ensino dos cursos estudados prevê que primeiro o modelo NCM é ensinado em detalhes antes de se introduzir a linguagem. Só depois de um conhecimento sólido em NCM que os alunos são apresentados à sintaxe da linguagem. Entretanto, foge ao escopo deste estudo analisar se tal metodologia é de mais ou menos sucesso.

2.2.4.4. Necessidade de ferramentas de autoria ou geração de código

Escrever o código fonte de programas complexos sem o auxílio de ferramentas computacionais geralmente é uma tarefa difícil (Fischer *et al.*, 2004). Alguns trabalhos apontam a praticidade e ganho ao se utilizar ferramentas de autoria para auxiliar o autor no processo de criação de novos documentos multimídia (Coelho *et al.*, 2004). É notório que a criação de páginas web se tornou muito mais popular depois que se sofisticaram as ferramentas de geração de código (como os modelos para páginas web). Atualmente, percebe-se que o autor, ao saber utilizar uma ferramenta adequada, não precisa ter outros conhecimentos

prévios para criar documentos HTML. Talvez o mesmo processo possa vir a ocorrer no futuro com NCL. Entretanto, toda ferramenta de autoria tem necessariamente algum comprometimento com o modelo semântico do objeto que está sendo criado. Não é por outra razão que apesar de interfaces gráficas sofisticadas, com extensos recursos de manipulação direta, a edição de imagens foto-realistas é difícil. Os usuários não podem dispensar conhecimentos sobre conceitos técnicos tais como '*anti-aliasing*', '*dithering'* e outros que aparecem em vários diálogos de interface. Por isto, mesmo com sofisticadas ferramentas de edição e autoria, certos conceitos fundamentais do modelo semântico da NCL permanecerão no horizonte de decisão dos usuários. A esse respeito, é interessante apontar alguns dados deste estudo.

É importante, porém, mais uma vez ressaltar que, durante a maior parte do curso, os programas foram, propositalmente, realizados sem auxílio de qualquer ferramenta de autoria. Como elas podem esconder ou simplificar uma série de detalhes importantes sobre a NCL e seu modelo semântico, a ferramenta de autoria Composer (Coelho *et al.*, 2004), especificamente desenvolvida para o Ginga-NCL, só foi introduzida no final do curso, depois de todos os conceitos serem ensinados.

Não houve uma pergunta específica nos formulários do Curso 1 sobre a necessidade de ferramentas de autoria, porém nove alunos comentaram espontaneamente sobre isso ao darem sua opinião geral sobre a linguagem. Alguns abordaram a questão de a ferramenta poder tornar a codificação *mais simples*. Disse um: "O problema no momento é que gerar código NCL 'na mão' não é uma tarefa simples (...)". Disse outro: "(Há dificuldade) de ter disposição de programar algo manualmente, por ser bastante ampla exige bastante cautela.". Já outros expressaram um ponto de vista mais fundado na prática: "Tenho dúvidas se a pessoa acostumada a ferramentas de autoria mais complexas, como flash, se interessem em desenvolver diretamente na linguagem sem a ajuda de ferramentas mais elaboradas".

Comentários como esses sugerem que uma ferramenta de autoria capaz de gerar documentos especificados/especificáveis em NCL deve: facilitar a criação de documentos possivelmente por meio de recursos gráficos; e oferecer mecanismos para reduzir o tempo de desenvolvimento. A primeira função deve

auxiliar a compreensão e modelagem NCM, enquanto a segunda deve acelerar a tarefa de geração de aplicações DTV.

Para ilustrar como a necessidade de redução no tempo de desenvolvimento por meio de ferramentas auxiliares é observada, seguem os comentários de dois alunos do Curso 1 sobre esse assunto: “Como é necessário escrever muito para fazer um programa, acho que será imprescindível o uso de uma ferramenta de autoria”; e “Por ser uma linguagem declarativa, o excesso de verbosidade pode aumentar a complexidade do código”.

Da mesma forma, a necessidade de uma ferramenta de autoria gráfica é indicada por alunos do Curso 2:

> “A primeiro modo achava meio difícil mais aos poucos minha opinião foi mudando e ainda vejo uma certa dificuldade. Para um usuário sem conhecimento de programação ficaria meio complicado fazer uma coisa muito avançada. Acho que a salvação de muitas pessoas seria o composer, não sei si haveria a possibilidade de ter uma linguagem mais fácil ao ver de todos ou uma ferramenta facilitadora (b a ba).”

Intuitivamente, os instrutores (e alunos do Curso 2) acreditavam que as turmas do Curso 3 se sentiriam desmotivadas com o uso de um editor de textos convencional para autoria em NCL e o abandonariam tão logo começassem a usar o Composer. Contrariando a opinião dos instrutores, os dados dos formulários indicam diversas opiniões contrárias por parte dos alunos, quando perguntados em que pontos a NCL ajuda ou atrapalha a entender o funcionamento do Composer:

> “Não vejo uma relação direta, é possivelmente capaz de criar uma ferramenta no qual o usuário não precise conhecer NCL. Talvez a evolução do Composer possa chegar a esse ponto. Mas atualmente é necessário saber NCL para entender como funciona o composer, pois hoje não tem como haver transparencia entre a construção gráfica e o código da NCL."

> "Em nenhum, pra mim que faço curso técnico em informatica e tenho contato com linguagens de programação do tipo C, pascal, java, é mais simples entender a NCL e só me ajudou, na verdade a NCL foi mais simples pra mim que o composer, ele é facil, mas a linguagem fixa mais o conteúdo aplicado."

Os dois comentários anteriores são importantes por exemplificarem opiniões diversas dos alunos. O primeiro expressa a necessidade de haver uma ferramenta de autoria gráfica que possa abstrair completamente o uso de NCL. O segundo aponta a necessidade de investir em facilidades para a autoria textual de NCL, que ele considera mais simples do que qualquer abstração gráfica. Ainda com base nos dados coletados do Curso 3, uma importante coincidência pode apontar uma tendência. Os alunos que mostraram preferência pelo uso de NCL sem o

Composer são usuários de linguagens de programação, o que conduz à evidência que NCL já atende, como é, a esse perfil, devendo ser complementada por um ambiente integrado que ofereça uma validação detalhada do código e mais facilidades para autoria textual.

Na verdade, esse é um ponto interessante a explorar. Parece que foi mais sentida a necessidade de uma ferramenta que não escondesse os conceitos da linguagem, mas que facilitasse o preenchimento do código (auto-preenchimento) e a detecção de possíveis erros, do que uma ferramenta que escondesse, por meio de recursos gráficos, os conceitos do modelo NCM.

2.2.4.5. NCL é adequada para a criação de aplicativos de TV interativos?

Não havia uma pergunta especificamente direcionada nos formulários do Curso 1 sobre a adequação de NCL para a TV digital interativa mas, ainda assim, 20 alunos (de 23 no total) opinaram sobre o assunto na pergunta subjetiva "Qual sua opinião geral sobre a NCL?".

Dois alunos emitiram opinião que pode ser classificada como negativa ou pelo menos limitadora. Entretanto dezoito alunos foram explícitos ao considerar a linguagem adequada para a TV digital.

Também no Curso 2, os alunos sublinharam ora a facilidade ora a complexidade da linguagem (e portanto do seu uso para a criação de aplicativos). Vejam-se estes dois depoimentos:

> "(...)Estou saindo com o conhecimento básico para explorar melhor o ncl (que em pimeira vista, parecia uma ferramenta apenas para especialistas) e que ao longo da semana perdeu boa parte de seus segredos. Saio animado com as novas perspectivas e com a cabeça cheia de idéias para trabalhar em minha comunidade e nas escolas onde trabalho. Isso mostra que o NCL não é uma ferramenta muito complicada é possível ensinar e aprender com ela. Agora a dedicação e empenho em realizar desafios particulares no NCL é fundamental para consolidar o conhecimento.";

> "(...)Aprendi a linguagem NCL/Composer - Ginga Brasil. Com um certo grau de dificuldade. Sempre pedia auxilio a vizinhos(colegas ao lado) e aos professores. Acredito que pra quem mexe com edição de Video como eu. Foi um senhor aprendizado.".

Conforme mencionado anteriormente, o próprio perfil dos alunos do Curso 3, com média escolaridade e experiência no uso do computador, mas com desempenho satisfatório na aprendizagem de NCL, a julgar por seu desempenho

no curso, sugere que a linguagem cumpre seus objetivos para uma variedade de usuários finais.

A discussão que sumariza as conclusões e resultados obtidos com o presente estudo empírico é propositalmente deixada para a Seção 2.4. Isso é feito com o objetivo de permitir que essa análise seja feita em conjunto com os resultados obtidos com o estudo analítico apresentado na Seção 2.3, a seguir.

2.3. Estudo Analítico sobre Reúso e Importação em NCL

A principal motivação para a especificação de famílias de documentos é oferecer um maior nível de reúso para autores de documentos hipermídia. A especificação de uma família de documentos pode ser reusada pelo autor, que só precisa informar aquilo que torna seu documento único. Como o reúso ocupa uma importância central na especificação de templates, o suporte existente em linguagens para autoria de documentos hipermídia precisa ser analisado com o propósito de identificar aspectos nos quais elas são satisfatórias ou insuficientes na especificação de famílias de documentos.

Este estudo analítico discute, no contexto de NCL, o *reúso de código estático* (reúso de especificação) e o reúso de código em execução (reúso de execução), de agora em diante chamado *reúso de código ativo*. Novamente, o estudo centra a discussão para o projeto de aplicações interativas para TV digital (aplicações DTV), um caso de uso de NCL para a concepção de documentos hipermídia.

Reúso de código, ou reúso de *software*, tem sido um tópico importante e persistente desde os primórdios das áreas de linguagens de programação e engenharia de software. O principal objetivo tem sido diminuir o tempo de desenvolvimento através do reúso de código bem testado, a utilização de partes de código mais confiáveis, e com menos margem para erros. É importante salientar que é preciso dar atenção não apenas reúso de código estático, mas também ao reúso de código em execução. Tomando como exemplo as linguagens para sistemas paralelos, elas permitem ainda o reúso de código em execução por diferentes fluxos de controle paralelos de uma aplicação, resultando em diferentes tipos de comportamento em cada um dos fluxos.

A área de engenharia de software é bastante robusta com relação a metodologias e técnicas que promovem o reúso de código. Tipicamente e historicamente, tais abordagens sempre estiveram bem mais presentes em linguagens imperativas, apesar de serem igualmente úteis em linguagens declarativas. Linguagens declarativas enfatizam a descrição em alto nível de uma aplicação ao invés de sua decomposição em uma implementação algorítmica. Mais ainda, linguagens declarativas usualmente definem modelos para o projeto de aplicações direcionadas para domínios específicos (uma DSL declarativa) (Deursen *et al.,* 2000) oferecendo um bom equilíbrio entre flexibilidade e simplicidade. Em outras palavras, parte da expressividade é perdida em troca do ganho em simplicidade. O suporte ao reúso em DSLs declarativas é mais difícil de se obter, já que a promoção do reúso deve também ser conciliada com o foco para o qual a linguagem foi projetada.

Neste estudo analítico é detalhado como o projeto da linguagem NCL e de seu modelo conceitual oferecem com sucesso o suporte ao reúso em nível declarativo. Contudo, a fim de mover do suporte à promoção do reúso, uma linguagem de especificação deve oferecer alguns méritos de usabilidade. Se ela não os oferecer, programadores (ou, nesse caso, autores de documentos hipermídia) podem abandoná-la em troca de uma ferramenta de especificação mais usável. Neste estudo, o *framework* de dimensões cognitivas de notações (CDNs) (Blackwell & Green, 2003; Blackwell, 2006) foi usado para a análise de aspectos de usabilidade de NCL como uma linguagem de interface para a criação de documentos hipermídia. Finalmente, o estudo inclui um conjunto de boas práticas de programação para promoverem reúso em processos de autoria baseados em NCL, os quais podem ajudar usuários atuais e futuros da linguagem.

As próximas subseções estão organizadas como segue. A Seção 2.3.1 argumenta sobre o reúso em diversos aspectos de uma aplicação para TV digital interativa. A Seção 2.3.2 discute como o reúso de código pode ser feito em uma mesma aplicação NCL. Na Seção 2.3.3, o suporte ao reúso entre diversos documentos NCL é apresentado. A Seção 2.3.4 apresenta a análise das características de reúso de NCL baseado no *framework* CDN.

2.3.1.
Reúso em Aplicações Interativas para TV Digital

Uma aplicação DTV é constituída pelos objetos de mídia (vídeo, áudio, texto e imagem, além de objetos com código imperativo e declarativo) e os relacionamentos de sincronismo espaço temporal entre eles. Como o paralelismo é inerente a aplicações DTV, há diversas situações em que é desejável prover reúso tanto de código ativo (nesse caso, objetos de mídia sendo apresentados) e de código estático (nesse caso, reúso sintático de parte de um documento hipermídia).

O reúso do conteúdo de objetos de mídia é essencial em aplicações DTV. É imprescindível permitir a autores editarem seus arquivos de imagem, áudio e vídeo uma única vez e então reusarem esse conteúdo em diferentes documentos ou diferentes partes de um mesmo documento. O reúso de conteúdo pode estar relacionado tanto com: (a) especificação de conteúdo (reúso estático), permitindo diversos objetos de mídia que referenciam o mesmo conteúdo o apresentar de forma independente por seus exibidores; e (b) apresentação de conteúdo (reúso ativo), permitindo que uma única apresentação de conteúdo seja compartilhada por diversos exibidores de objetos de mídia. Mais ainda, é conveniente dar suporte não somente ao reúso do conteúdo, mas ainda o reúso extensivo de todos os outros atributos da apresentação (toda a especificação do objeto de mídia). Em suma, está sendo discutido o reúso sintático (estático), o que significa ter diversas instâncias independentes do mesmo código (exibições independentes de objetos de mídia), e reúso da mesma apresentação de um objeto de mídia (reúso de código ativo).

O reúso de características de apresentação também é útil. Aplicar padrões de interface tende a aperfeiçoar a usabilidade da aplicação DTV. Quando o mesmo padrão recorrente de interface é usado, famílias de aplicações DTV ganham identidade própria e os telespectadores se acostumam àquele formato. Como exemplo, se diversos objetos de mídia devem ser sempre exibidos na mesma região da tela do mesmo dispositivo, o reúso da especificação da região (estático) força esse padrão. Se não apenas a localização, mas ainda a forma na qual o objeto de mídia é apresentado (por exemplo, o nível de transparência da imagem, cor de

fonte, volume de som, etc.) é declarado em separado, o reúso estático desse trecho de código força um estilo pré-definido.

Movendo o foco do reúso para a autoria estruturada de documentos, muitas aplicações DTV seguem uma organização hierárquica. Por exemplo, uma série de TV pode ser composta de um logotipo do canal de TV (imagem) e vários episódios; cada episódio é composto por cenas que são compostas por tomadas de câmera e possivelmente algumas propagandas (talvez interativas); e assim por diante. Em outras palavras, uma aplicação DTV é uma composição recursiva de objetos (objetos de mídia e outras composições). Geralmente diversas aplicações DTV seguem a mesma estrutura (o mesmo script de TV). Sendo assim, é conveniente suportar o reúso de estrutura de toda a aplicação DTV ou parte dela, para fazer a produção mais fácil e ainda para forçar um formato que dá identidade a programas de TV.

Seguindo para outro aspecto de reúso, uma das grandes vantagens de aplicações DTV é sua capacidade de adaptar conteúdo ou a forma que o conteúdo é apresentando, dependendo dos perfis do telespectador, de sua localização ou da plataforma de exibição. Adaptações são baseadas na resolução de regras, e bases de regras devem poder ser reusadas em diferentes aplicações.

Grande parte da especificação de aplicações DTV envolve a definição dos relacionamentos entre objetos de mídia. Em sendo assim, reduzir ou pelo menos simplificar a definição dos relacionamentos é essencial para diminuir o tempo de desenvolvimento das aplicações e a propensão a erros. Relacionamentos são definidos por um tipo de relação e pelos atores dessas relações (em nosso caso, interfaces de objetos) que fazem papéis definidos pelo tipo de relação. Tipos de relação são difíceis de ser especificados e é uma boa prática defini-los em separado das aplicações e permitir que possam ser compartilhados em vários relacionamentos.

É muito comum encontrar partes de aplicações DTV repetidas em outras aplicações. Esse é, por exemplo, o caso de propagandas inseridas em aplicações. Dessa forma, é importante tratar a especificação de aplicações como uma biblioteca da qual elas podem ser importadas por outras aplicações. De fato, a hierarquia aninhada de composições mencionada previamente e exemplificada no caso do encapsulamento de uma série de TV pode ser pensada como um aninhamento de documentos. No exemplo, cada composição pode representar um

documento de tal forma que cada especificação da cena de um episódio pode ser vista como um documento que pode ser importado e incluído em outro documento representando todo o episódio, o que pode, por sua vez, ser importado por outro documento representando toda a série.

No mais, é desejável tratar bibliotecas de aplicações como uma base de informação onde apenas alguns aspectos de interesse podem ser importados. Como exemplo, poderia ser possível importar apenas o leiaute de uma aplicação em outra aplicação.

Como visto, as características de suporte ao reúso devem transpassar diversas tarefas de autoria e apresentação quando na especificação de aplicações DTV. É conveniente reforçar que este estudo foca o reúso em NCL, mas a mesma abordagem pode ser explorada em outras DSLs baseadas no tempo que tenham como domínio aplicações DTV. Da mesma forma, vários aspectos levantados por este estudo podem ser igualmente generalizados para a autoria de documentos hipermídia e não apenas no contexto de aplicações DTV.

2.3.2. Reúso em uma mesma aplicação NCL

Esta seção discute o suporte dado ao reúso em NCL para permitir que trechos de código possam ser reutilizados no mesmo documento.

2.3.2.1. Reúso de Conteúdo

Em NCL, a especificação de um objeto de mídia define seu conteúdo, suas interfaces (âncoras de conteúdo e propriedades) e como (e onde) o conteúdo deve ser exibido. Esta seção está destinada apenas ao conteúdo.

Um objeto de mídia define seu conteúdo por referência através de seu atributo *src*. O valor do *src* localiza o conteúdo independente de onde ele está (por exemplo, armazenado em um sistema de arquivos local ou remoto, ou em um carrossel de dados transmitido por algum protocolo de envio de dados sem solicitação). Dessa forma, uma aplicação NCL é composta por um documento NCL, o qual descreve a estrutura e semântica da aplicação e os conteúdos de mídia definidos fora desse documento.

Em geral, dois objetos de mídia com o mesmo valor de atributo *src*, para localização de seus conteúdos, terão suas exibições independentes. Isso significa que, se os objetos são instanciados em momentos diferentes, eles terão o mesmo conteúdo apresentado em momentos diferentes. Assim, se, por exemplo, um vídeo de 100 segundos de duração for referenciado por dois objetos de mídia cujas apresentações são iniciadas 50s defasadas uma em relação à outra, ele terá duas apresentações diferentes, cada uma com a duração de 100 segundos e em paralelo durante os 50 segundos iniciais de um objeto e 50 segundos finais do outro.

Entretanto, a NCL também permite que, além de conter a mesma referência a um conteúdo, um objeto de mídia especifique que seu conteúdo é um espelho de outro objeto. Nesse caso, podem-se ter ainda duas apresentações em paralelo, mas idênticas. No caso do exemplo anterior do vídeo, o objeto de mídia que fosse iniciado 50 segundos depois teria a duração de sua exibição apenas de 50s, começando pela metade.

A Figura 2 ilustra o primeiro caso citado de reúso com os objetos “video1” e “video2”, e o segundo caso com os objetos “video2” e “video3”. O atributo *descriptor* é visto na próxima seção.

```
<media id="video1" src="../media/movie.mp4"
       descriptor="descVideo1"/>
<media id="video2" src="../media/movie.mp4"
       descriptor="descVideo2"/>
<media id="video3" src="ncl-mirror://video2"
       descriptor="descVideo3"/>
```

Figura 2. Reúso de Conteúdo.

2.3.2.2. Reúso de Leiaute

Um aspecto importante e que consome um tempo considerável no projeto de documentos multimídia em NCL é a definição espacial do posicionamento inicial dos objetos de mídia. Outra característica relacionada é a forma como um objeto de mídia deve ser inicialmente exibido. Esses dois aspectos de uma apresentação NCL são especificados, cada um deles, como entidades de primeira classe no cabeçalho de um documento NCL, respectivamente através das bases de regiões e de descritores.

Cada base de regiões permite que se divida a tela de um dispositivo de exibição ao qual a base é associada em um conjunto de regiões retangulares. Tais regiões podem estar inclusive sobrepostas umas sobre as outras, caso em que a própria região carrega a informação, por meio do atributo *zIndex*, de qual delas deve ser exibida na frente das demais. Geralmente um documento NCL possui apenas uma base de regiões, mas cada uma delas pode ser associada a um dispositivo de exibição diferente. Esse mecanismo garante o suporte nativo na linguagem ao uso de múltiplos dispositivos de exibição, o que significa que uma mesma apresentação pode sincronizar objetos de mídia sendo apresentados em mais de um dispositivo. É conveniente ressaltar que a forma como se dá a comunicação entre tais dispositivos foge ao escopo declarativo da linguagem e não é preocupação do autor.

A Figura 3 exemplifica um cenário com dois dispositivos de exibição. O dispositivo principal (definido nas linhas 1 a 10) é representado pela região "fullScreenRg" (linhas 2 a 9). Tal região é dividida em duas regiões filho, a primeira delas "centerRg" que corresponde à metade da tela centralizada e a outra "buttonRg", onde um botão vai ser exibido no canto superior direito. As regiões do dispositivo de exibição secundário (identificado na linguagem por "systemScreen(2)") são especificadas nas linhas 11 a 16. Tal dispositivo é representado pela região "remoteControlRg", que possui uma região filho "merchandizingRg" ocupando o quadrante superior esquerdo da tela.

```
1: <regionBase>
2:    <region id="fullScreenRg">
3:       <region id="centerRg"
4:               left="25%" top="25%"
5:               width="50%" height="50%"/>
6:       <region id="buttonRg"
7:               right="2%" top="2%"
8:               width="5%" height="5%"/>
9:    </region>
10:   </regionBase>
11:   <regionBase device="systemScreen(2)">
12:      <region id="remoteControlRg">
13:         <region id="merchandizingRg"
14:                 height="50%" width="50%"/>
15:      </region>
16:   </regionBase>
```

Figura 3. Bases de regiões para múltiplos dispositivos.

O cabeçalho da Figura 3 poderia ilustrar um cenário em que um vídeo é exibido ao centro de uma TV, ocupando a metade da área da tela (região "centerRg"). Em certo momento do vídeo, um ícone interativo apareceria no canto superior direito da TV (região "buttonRg"). Se um botão de interatividade fosse pressionado através de um controle remoto com tela de LCD, uma propaganda seria exibida naquele controle remoto ocupando o quadrante superior esquerdo de sua tela (região "merchandizingRg"). Nesse cenário, o acionamento de uma propaganda interativa não interromperia o programa de TV sendo exibido e seria direcionada apenas ao telespectador que interagiu. Múltiplos controles remotos podem tornar mais individualizada a experiência de assistir à TV.

Cada documento NCL especifica uma base de descritores, os quais definem *como* os objetos de mídia devem ser inicialmente exibidos. O conceito de descritor é importante porque isola do objeto de mídia as informações específicas de sua apresentação. Tem-se então um novo mecanismo de promoção de reúso, já que é muito comum a necessidade de vários objetos de mídia se apoiarem em uma mesma descrição de iniciação.

Um descritor pode ser enriquecido por parâmetros que são tratados especificamente pelo exibidor de cada tipo de mídia, deixando ainda mais versátil o suporte a novos tipos de monomídia. Alguns exemplos disso são quando se quer informar certo grau de transparência à exibição de uma imagem ou determinar o volume do áudio.

A Figura 4 acrescenta ao cenário da Figura 3 uma base de descritores. O descritor "centerDs" aponta para a região "centerRg" com o volume de som inicialmente em 50% (linhas 2 a 5). O descritor "buttonDs" faz uso da região "buttonRg" e especifica uma transparência de 20% (linhas 6 a 9). O descritor "merchandizingDs" (linhas 10 e 11), por sua vez, apenas indica o uso inicial da região "merchandizingRg", que está presente no dispositivo de exibição secundário.

Geralmente NCL promove reúso se referindo a identificadores de partes reusadas, como os valores de localização no atributo *src*, ou os valores do atributo *id*, quando a referência é feita a uma entidade NCL, como na Figura 2 (valores do atributo *descriptor*) e Figura 4 (valores do atributo *region*). Como na maioria das aplicações XML, entidades NCL devem ter um identificador único em um documento (o valor do atributo *id*).

```
1: <descriptorBase>
2:  <descriptor id="centerDs" region="centerRg">
3:     <descriptorParam name="soundLevel"
4:                      value="0.5"/>
5:  </descriptor>
6:  <descriptor id="buttonDs" region="buttonRg">
7:     <descriptorParam name="transparency"
8:                      value="0.2"/>
9:  </descriptor>
10:  <descriptor id="merchandizingDs"
11:              region="merchandizingRg"/>
12:   </descriptorBase>
```

Figura 4. Base de descritores.

É importante reforçar que NCL permite práticas de reúso, mas não as impõe. Referências a elementos <descriptor> e <region> são opcionais. Um elemento <descriptor> poderia definir uma região usando elementos <descriptorParam>, assim sem fazer referência a elementos <region>. Um elemento <media> pode ainda definir todas as propriedades necessárias de apresentação usando elementos <property>, mesmo sem referenciar um elemento <descriptor>. De fato, descritores são usados somente para inicializar as propriedades. A Figura 5 ilustra tanto o elemento <media> cujo conteúdo é a propaganda a ser apresentada (linha 1), conforme cenário ilustrado pelas Figura 3 e Figura 4, quanto uma alternativa para a definição desse elemento <media> sem fazer referência a descritores (linhas 3 a 9).

```
1: <media id="merchandizing" descriptor="merchandizingDs"
          src="index.html"/>
2: <!-- Uma alternativa -->
3: <media id="merchandizing" src="index.html">
4:    <property name="device" value="systemScreen(2)"/>
5:    <property name="left" value="25%"/>
6:    <property name="top" value="25%"/>
7:    <property name="height" value="50%"/>
8:    <property name="width" value="50%"/>
9: </media>
```

Figura 5. Alternativas para a definição do elemento <media>.

2.3.2.3. Reúso de Objeto de Mídia

Como mencionado na Seção 2.3.2.1, ao definir um objeto de mídia é necessário especificar a localização de seu conteúdo, suas interfaces e ainda como ele deve ser apresentado. Há, no entanto, outra forma de fazê-lo. Um elemento

<media> pode ser definido referenciando outro, o que faz ele herdar toda a definição daquele objeto de mídia (conteúdo, propriedades e âncoras).

A Figura 6 estende o cenário de exemplo prévio. Na figura, o vídeo principal da TV (*id*="mainVideo") faz referência ao descritor "centerDs", como previamente explicado (linha 1). Seu atributo *src* especifica que o conteúdo deve ser obtido do fluxo de transporte ou, mais precisamente, do fluxo elementar de transporte com *program_number.component_tag*="0x01.0x05". A linha 2 define o objeto de mídia "button", e a linha 3 o objeto de mídia HTML "merchandizing".

Na linha 4 da Figura 6, o objeto de mídia "buttonRef" referencia o objeto de mídia "button", herdando toda sua definição (nesse caso, propriedades definidas pelo descritor "buttonDs"). O atributo *instance* é explicado como se segue.

No modelo conceitual de NCL, a estruturação do documento é feita usando contextos. Um contexto é um objeto especial que agrupa objetos de mídia e outros contextos recursivamente. A única restrição aplicável é que um contexto não pode ser incluído recursivamente nele próprio. Um contexto pode ainda conter relacionamentos entre seus componentes. Em NCL, um contexto é representado pelos elementos <body> e <context>. O elemento <body> é apenas um nome para o ancestral de todos os contextos em um documento NCL.

```
1: <media id="mainVideo"
2:        descriptor="centerDs"
3:        src="sbtvd-ts://0x01.0x05"/>
4: <media id="button"
5:        descriptor="buttonDs"
6:        src="media/redButton.png" />
7: <media id="merchandizing"
8:        descriptor="merchandizingDs"
9:        src="index.html" />
10:   <media id="buttonRef"
11:          refer="button"
12:          instance="new"/>
```

Figura 6. Reúso de objeto de mídia.

O aninhamento de contextos cria o que NCL chama de perspectiva para um objeto, um conceito similar ao de "escopo" em linguagens de programação de propósito geral. A perspectiva de um objeto é a estrutura de aninhamentos do contexto mais externo (representado pelo elemento <body>) para o objeto. Em uma perspectiva, um objeto herda todos os relacionamentos definidos em contextos ancestrais referenciando diretamente ou indiretamente para ele.

O reúso de objeto (seja ele de objeto de mídia ou de contexto) permite que um objeto pertença a mais que uma perspectiva, e assim a mais que um contexto. Essa é uma característica de reúso importante já que um mesmo objeto em diferentes perspectivas pode ter diferentes comportamentos (dependendo dos relacionamentos herdados), promovendo o conceito de reúso de código ativo para objetos de mídia.

Na especificação de um elemento <media> que referencia (e assim reúsa) outro elemento <media>, o atributo *instance* permite a definição se o reúso de código sintático (estático) é desejado (*instance*="new", vide Figura 6), ou se o objetivo é o reúso de código ativo. No último caso, o reúso pode ser tanto instantâneo (*instance*="instSame") ou gradual (instance="gradSame"), como segue.

Suponha que o objeto de mídia A referencia o objeto de mídia B.

1) Se *instance*="new", uma nova instância independente de A será criada quando A for iniciado, herdando todo código definido por B, independente do fato de B estar sendo exibido ou não. O objeto A herda as interfaces do objeto B, mas pode adicionar novas. Mais ainda, todas as interfaces fazem parte de relacionamentos definidos apenas na perspectiva onde o objeto foi criado.

2) No caso de reúso de código ativo (*instance* igual a "instSame" ou "gradSame"), A e B são o mesmo objeto.

2i) Se *instance*="instSame", as interfaces para este mesmo objeto vêm das especificações de A e B. Mais ainda, relacionamentos definidos ambos na perspectiva de A e na perspectiva de B estão habilitados.

2ii) Se *instance*="gradSame", a única diferença é que cada representação de objeto deve receber uma ação explícita para sua ativação. Uma vez ativo o objeto, suas interfaces são incorporadas ao objeto de apresentação comum. Da mesma forma, relacionamentos definidos na perspectiva de um objeto de mídia ativado se tornam habilitados.

O reúso de código de objeto de mídia em execução tem se mostrado muito útil, mais ainda que o reúso de código estático.

2.3.2.4. Reúso de Relação

Ao contrário de outras linguagens XML, NCL separa os conceitos de relação e relacionamento (Muchaluat-Saade, 2003) como é usual em ADLs (Clements, 1996).

Relações são definidas em uma base de relações na parte do cabeçalho do documento (elemento <head>). Relações definem papéis e a cola relacionando papéis. NCL permite definir qualquer tipo de relação, mas palavras reservadas foram definidas para simplificar a definição de relações causais espaço-temporais. Relações causais são definidas por elementos <causalConnector>. Em relações causais, papéis de condição devem ser satisfeitos a fim de disparar papéis de ação.

Relacionamentos (representados por elementos NCL <link>) podem ser definidos referenciando uma relação e definindo quais atores devem efetuar os papéis da relação. O reúso de relação é, assim, natural em NCL. Embora esta seja a única forma de definir relacionamentos em NCL 3.0, um açúcar sintático pode ser adicionado à linguagem para permitir a definição da relação e relacionamento em um único elemento, conforme discutido na Seção 2.4.

Como exemplo de reúso de relação, suponha que, durante uma apresentação multimídia, o começo da exibição de um vídeo deve iniciar uma imagem em paralelo. Suponha também que, em certo momento da apresentação, o início de uma trilha de áudio deve iniciar, também sincronizadamente, uma animação. Esses dois relacionamentos podem compartilhar (reusar) a mesma relação: “ao iniciar o papel X, então inicie o papel Y”. Baseado nessa relação, o primeiro relacionamento pode associar o papel X ao vídeo e o papel Y à imagem. Por outro lado, o segundo relacionamento pode associar o papel X à trilha de áudio e o papel Y à animação.

É conveniente reforçar que relacionamentos podem ser definidos entre qualquer tipo de objeto de mídia: objetos de mídia perceptuais, objetos de mídia com código imperativo (código Lua, por exemplo), ou objetos de mídia com código NCL ou outra linguagem declarativa (HTML, por exemplo).

2.3.2.5. Reúso de Estrutura

O conceito de contexto, definido previamente, é importante não somente para estruturar documentos, mas ainda para encapsular sua semântica de apresentação. Uma vez que um contexto contém não somente objetos, como também os relacionamentos entre eles, um contexto define de fato um documento embutido em outro, com uma semântica temporal e espacial bem definida por meio de seus elos.

Estruturar um documento usando contextos é uma boa prática de programação. Há diversos exemplos (o que pode ser visto em (Clube NCL, 2010)) que demonstram essa utilidade, alguns dos quais já discutidos neste trabalho.

Da mesma forma que objetos de mídia podem ser reusados, contextos também podem ser reusados pela definição de outro contexto. No entanto, somente o reúso de código estático é permitido, o que significa que ambos os contextos são considerados novas instâncias independentes. Não há o equivalente aos valores “instSame” ou “gradSame” no reúso de contexto.

O reúso de estrutura está associado aos níveis de abstração máximo e mínimo expostos pela notação da linguagem, e o quanto de detalhes da notação pode ser encapsulado.

2.3.2.6. Reúso de Regras e Efeitos de Transição

No elemento <head> de um documento NCL são definidas as bases de regras e de transições. Transições são efeitos visuais ou acústicos que podem ser aplicados no início ou término da apresentação de um objeto de mídia. Regras estabelecem um dialeto lógico que pode ser usado para suportar a adaptação de conteúdo ou da apresentação de conteúdo.

A Figura 7 ilustra um exemplo de base de transições (linhas 2 a 4). A linha 3 especifica o efeito de transição “fade” que deve ser aplicado à apresentação de um conteúdo durante 3 segundos. Linhas 12 e 13 mostram como um elemento <descriptor> referencia essa transição para aplicar seus efeitos no início (atributo *transIn*) da apresentação de um objeto de mídia.

```
1:   <head>
2:    <transitionBase>
3:     <transition id="fade3s" type="fade" dur="3s"/>
4:    </transitionBase>
5:    <ruleBase>
6:     <compositeRule id="canPlayPtVideoR" operator="and">
7:      <rule id="ptR" var="system.language" comparator="eq"
              value="pt"/>
8:      <rule id="processR" var="system.CPU" comparator="gt"
              value="0.4"/>
9:     </compositeRule>
10:   </ruleBase>
11:   <descriptorBase>
12:    <descriptor id="merchandizingDs" region="merchandizingRg"
                  transIn="fade3s"/>
13:   </descriptorBase>
14:  </head>
15:  <body>
16:   <switch id="merchandizing">
17:    <bindRule rule="canPlayPtVideoR"
                 constituent="vMerchandizing"/>
18:    <defaultComponent component="fMerchandizing"/>
19:    <media id="vMerchandizing" descriptor="merchandizingDs"
              src="merchandizing.mp4"/>
20:    <media id="fMerchandizing" descriptor="merchandizingDs"
              src="index.html"/>
21:   </switch>
22:  </body>
```

Figura 7. Bases de transições e regras.

Da mesma forma que na separação entre os elementos <descriptor> e <region>, a separação dos efeitos de transição dos descritores permite seu reúso. É muito comum a definição de uns poucos efeitos de transição usados por diversos objetos de mídia durante uma apresentação DTV. No entanto, diferente do que ocorre com regiões, um elemento <descriptor> deve sempre fazer referência a uma transição se o efeito é desejado. Não há uma forma de definir transições como parâmetros de um elemento <descriptor>. Talvez este seja um conceito da linguagem que deva ser revisado em sua próxima versão.

Como previamente citado, regras expressam critérios que permitem as adaptações de conteúdo e de apresentação de conteúdo (Soares *et al.,* 2009). Regras podem ser simples ou compostas. Regras simples comparam uma variável a um valor ou a outra variável. Regras compostas são feitas por expressões lógicas "ou" ou "e" entre regras (simples ou compostas).

Na Figura 7, a regra composta "canPlayPtVideoR" testa se um dispositivo pode tocar um vídeo em português (linhas 6 a 9) avaliando duas variáveis de sistema.

Em NCL, elementos <switch> contêm objetos alternativos que podem ser escolhidos baseados em qual regra é satisfeita. Similar a elementos <context>, elementos <switch> podem ainda ser sintaticamente reusados. Como exemplo de <switch>, a Figura 7 define o <switch id="merchandizing" ...> (linhas 6 a 21). A linha 17 especifica que, no caso do dispositivo de exibição poder tocar o vídeo em português (regra "canPlayPtVideoR"), o objeto de mídia "vMerchandizing" deve ser selecionado. Do contrário, o objeto de mídia *default* "fMerchandizing" é a escolha correta.

Similarmente a elementos <switch>, os elementos <descriptorSwitch> contêm descritores alternativos que podem ser escolhidos baseados em qual regra é satisfeita. Assim como elementos <descriptor>, elementos <descriptorSwitch> podem também ser referenciados por objetos de mídia.

É importante observar que é impossível reusar uma regra a fim de definir outras regras. Mais ainda, regras podem referenciar propriedades globais (propriedades de um tipo especial de objeto de mídia chamado "application/x-ncl-settings") sem que o documento as tenha explicitamente declarado. Por outro lado, qualquer outro uso de uma propriedade requer que a propriedade seja explicitamente declarada. Regras são rara exceção na coerência do reúso em NCL. Talvez esse seja um segundo conceito da linguagem a ser revisado em futuras versões.

2.3.3. Reúso entre aplicações NCL

Em todos os casos mencionados anteriormente na Seção 2.3.2, o reúso foi discutido dentro do escopo de uma mesma aplicação NCL. É possível, contudo, importar bases de informações especificadas em outro documento e reusar seus componentes. Além disso, mesmo parte ou o todo da estrutura de outro documento é passível de ser importada e reusada. Isso permite que se organize ainda mais a estrutura de apresentação em documentos diversos. Com ilustração, imagine essa necessidade em uma propaganda (uma aplicação NCL) que possa ser embutida e exibida em diversas aplicações DTV (definidas por outros documentos NCL).

2.3.3.1. Documentos NCL aninhados

Há duas alternativas para o reúso da especificação de um documento inteiro (isto é, sua estrutura e semântica de apresentação). A primeira delas envolve importar um documento e o reusar como um objeto de contexto em um novo documento. A segunda alternativa envolve tratar o documento como um objeto de mídia com código declarativo NCL no novo documento.

A importação de um documento A como um contexto em um documento B permite que se crie uma cópia lógica do documento A. Esse contexto, que, como sempre em reúso de contextos, é uma cópia, é passível de sofrer ações derivadas de relacionamentos (elos NCL) como qualquer outro objeto. A referência ao documento importado é feita no formato "alias#docID" (indicando um alias arbitrário ao documento importado, que é informado no cabeçalho do documento importador) e o identificador (atributo *id*) do documento importado, indicando que do documento se quer reusar todo o seu corpo. Como é visto na Seção 2.3.3.3, partes de um documento também podem ser reusadas em outro documento.

A Figura 8(a) apresenta um documento NCL, cujo identificador é "A". Seu conteúdo está propositalmente omitido sem prejuízo de entendimento. A Figura 8(b) mostra como é a importação do documento A, definido com *alias* "docA" no escopo do novo documento "B" (linhas 5 e 6). O contexto "referDocA" referencia o documento "docA#A". Isso define tal contexto como uma cópia passível de ser manipulada normalmente por elos e contendo os mesmos pontos de interface do documento importado.

(a) Documento A.ncl

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:  <ncl id="A" (...) >
3:     (...)
4:  </ncl>
```

(b) Importando o Documento A em B.ncl

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:  <ncl id="reuse-NCL" (...) >
3:  <head>
4:     <importedDocumentBase>
5:        <importNCL documentURI="A.ncl" alias="docA"/>
6:     </importedDocumentBase>
7:  </head>
8:  <body>
9:     (...)
10:    <context id="referDocA" refer="docA#A"/>
11: </body>
12: </ncl>
```

Figura 8. Importando e aninhando um documento NCL.

O reúso de um documento pode também ser feito pela utilização de um objeto de mídia contendo código NCL (do tipo “application/x-ncl-NCL”), esse recurso é utilizado quando se quer aninhar a apresentação de um documento NCL em uma região de outro documento, como se a apresentação do documento importado fosse de um tipo de mídia primitivo. A Figura 9(b) descreve o reúso do documento A por meio do objeto de mídia “cDocA” na região apontada pelo descritor “centerDs”. Na prática, quando esse objeto de mídia é exibido, o documento A é apresentado naquela região respeitando as regras normais de sobreposição de regiões.

(a) Documento A.ncl

```
1: <?xml version=”1.0” encoding=”ISO-8859-1”?>
2:  <ncl id=”A” (...) >
3:     (...)
4:  </ncl>
```

(b) Importando o Documento A para uma região em C.ncl

```
1:  <?xml version=”1.0” encoding=”ISO-8859-1”?>
2:  <ncl id=”C” (...) >
3:  <body>
4:     <media type=”application/x-ncl-NCL”
5:                  id=”cDocA” src=”A.ncl”
6:                  descriptor=”centerDs”/> (...)
7:  </body>
8:  </ncl>
```

Figura 9. Importando um documento NCL para ser exibido em uma região definida no documento importador.

2.3.3.2. Importando Bases de Informações

Toda base definida em um elemento <head> de um documento NCL pode ser importada por uma base correspondente em outro documento por meio do elemento <importBase> de NCL. Esse elemento define um nome (*alias*) para a base importada e a referencia por meio de sua URI (URI_da_base#id_da_base).

Bases de região e de descritores podem ser definidas em arquivos separados daqueles que especificam aplicações NCL e então importados por esses documentos como se fossem bibliotecas de leiaute. Isso é bem útil em famílias de aplicações que devem seguir um mesmo padrão de apresentação como identidade visual e ainda para aperfeiçoar a usabilidade. Isso é típico em aplicações DTV, como em novelas ou séries de TV.

Bases de efeitos de transição podem ser importadas e referenciadas nos descritores do documento que as importam. Esse é um procedimento habitual, já que é de praxe utilizar um mesmo conjunto pequeno de efeitos de transição para não criar apresentações cansativas ou repetitivas visualmente.

A importação de bases de conectores é a prática mais comum em aplicações NCL. Autores e organizações tendem a adicionar cada relação que eles definem em uma mesma base comum compartilhada por todas as suas aplicações. Geralmente, elementos <causalConnector> são definidos por autores NCL mais experientes e incluídos nessa base comum de conectores a fim de serem importados em aplicações especificadas por usuários iniciantes. Mais ainda, uma convenção é comumente definida por companhias para identificar seus conectores a fim de facilitar seu reúso. É raro em aplicações DTV convencionais a necessidade de conectores diferentes dos usuais, como os ilustrados em (ABNT, 2007).

A importação e reúso de conectores têm a vantagem adicional de reduzir a curva de aprendizagem da linguagem, já que o conceito de conector é um dos mais difíceis da linguagem. A importação e reúso de conectores permitem a autores produzirem aplicações DTV logo no início da aprendizagem de NCL.

2.3.3.3. Importando Objetos de uma Aplicação NCL

A Seção 2.3.3.1 discute como um documento pode ser importado e reusado como um todo. Em adição, partes de um documento também podem ser reusadas. De fato, qualquer objeto NCL (objetos de mídia, contextos e de alternativa) pode ser reusado.

Em todos esses casos, o reúso é similar ao de um documento inteiro. A única diferença é a necessidade de referenciar o identificador do objeto reusado ao invés do identificador do documento.

A Figura 10 mostra diversos exemplos de importação. Duas bases são importadas pelo documento: a base de descritores em “descriptors.ncl” (linha 5); e a base de conectores em “connectors.ncl” (linha 8). Adicionalmente, o documento “A.ncl” é importado (linha 11): o conteúdo de seu corpo e todas as suas bases.

No corpo do documento na Figura 10, o contexto “someContext” importado é reusado (linha 15) para ser apresentado nas mesmas regiões definidas no documento importado “A”. O objeto de mídia “refMediaA” (linha 16) reusa o objeto de mídia “someMedia” definido no documento “A”. O objeto de mídia “image” (linha 17) com o conteúdo definido pelo arquivo “someImage.png” deve ser inicialmente exibido seguindo o descritor importado “centerDs”, especificado em “descriptors.ncl”.

```
1:   <?xml version="1.0" encoding="ISO-8859-1"?>
2:   <ncl id="importing" (...) >
3:    <head>
4:     <descriptorBase>
5:      <importBase alias="desc" documentURI="descriptors.ncl"/>
6:     </descriptorBase>
7:     <connectorBase>
8:      <importBase alias="conn" documentURI="connectors.ncl"/>
9:     </connectorBase>
10:    <importedDocumentBase>
11:     <importNCL alias="docA" documentURI="A.ncl"/>
12:    </importedDocumentBase>
13:   </head>
14:   <body>
15:    <context id="refDoc" refer="docA#someContext"/>
16:    <media id="refMediaA" instance="new"
              refer="docA#someMedia"/>
17:    <media id="image" src="someImage.png"
              descriptor="desc#centerDs"/>
18:    (...)
19:    <link xconnector="conn#onBeginStopStart">
20:      <bind component="image" role="onBegin"/>
21:      <bind component="refMediaA" role="stop"/>
22:      <bind component="refDoc" role="start"/>
23:    </link>
24:   </body>
25:  </ncl>
```

Figura 10. Importando Bases e Documentos.

Ainda na Figura 10, o elemento <link> (linhas 19 a 23) faz referência ao conector importado “onBeginStopStart”. Como pode ser observada, a nomenclatura usada é suficiente para um autor entender o que a relação significa. Cabe ao elemento <link> estabelecer as associações, definindo que, quando o objeto de mídia “image” começar a ser apresentado (linha 20), então o objeto de mídia “refMediaA” deve ser terminado (linha 21), e o contexto “refDoc” (linha 22) deve começar a ser apresentado.

2.3.4.
Analisando a Usabilidade da NCL

Entre diversos métodos possíveis para avaliar o quão usáveis são as características de reúso NCL para seu público alvo, há métodos empíricos e analíticos. Os primeiros envolvem observações empíricas de como as pessoas de fato usam as características providas por NCL em situações reais de realização de tarefas ou em configurações de laboratório realísticas. Os métodos analíticos são derivados de teorias, modelos ou *frameworks*, em variados graus de formalidade. Uma combinação de métodos é claramente a melhor escolha com o propósito de se obter profundidade e entendimento com relação a como e por que NCL suporta o reúso em um contexto prático. É, contudo, necessário decidir como combiná-los: por exemplo, eles deveriam ser aplicados independentes um do outro, ou sequencialmente?

A decisão tomada na avaliação específica do reúso em NCL é iniciar por um método analítico. Entre as vantagens dessa escolha, duas são particularmente relevantes. Primeiramente, métodos analíticos podem ajudar a detectar características específicas de NCL que seus projetistas não estão cientes. Uma vez detectadas, essas características podem ser empiricamente testadas com usuários NCL em diferentes contextos de uso. Segundo, métodos analíticos podem ajudar a estabelecer conexões teóricas ou pré-teóricas entre (classes de) características, ajudando analistas em sondar a natureza e consequências dessas características NCL e criar relações entre elas. Essa possibilidade permite estabelecer conexões entre NCL e outras linguagens de programação ou de especificação, um fator que tem papel central no recrutamento de participantes para testes empíricos. Para mencionar pelo menos um desses fatores, caso se perceba que NCL compartilha certas características com linguagens de programação orientadas a eventos, por exemplo, testes empíricos talvez devam discriminar entre participantes com ou sem conhecimento prévio de tais linguagens.

Uma ferramenta analítica comumente usada para avaliar a usabilidade de linguagens computacionais nos últimos anos é o *Framework* de Dimensões Cognitivas de Notações (CDN) (Blackwell & Green, 2003) (Blackwell, 2006). Embora o CDN originalmente consista de princípios de projeto para analisar a usabilidade de artefatos de informação, ele também toma linguagens de

programação como artefatos de informação de forma a obter considerável influência no entendimento do esforço cognitivo que elas possam ocasionalmente impor a programadores com diferentes perfis e níveis de perícia. O CDN tem sido criticado por não ser "científico de acordo com padrões normalmente aplicados em pesquisa" (Moody, 2009). Por outro lado, para pesquisadores que dão importância à interpretação e a *frameworks* interpretativos na ciência, tais críticas não se aplicam. O CDN pode e (como se observa neste estudo) deve fornecer informações importantes em como interpretar o significado e impacto de características sintáticas e semânticas de linguagens de programação na cognição humana.

Ao se adotar o CDN, é preciso que o analista dê um passo extra quando comparado ao uso de técnicas não-interpretativas da pesquisa científica. É necessário definir como cada dimensão cognitiva deve ser sistematicamente aplicada ao objeto de análise antes de se apreciar os resultados do emprego do método. Neste estudo foram listadas e definidas brevemente as dimensões propostas pelo *framework* CDN, mostrando como cada uma delas foi aplicada na análise de NCL. Essa aplicação do CDN pode ser vista como uma contribuição subsidiária para outros pesquisadores interessados na análise de outras linguagens de especificação para aplicações DTV, e possivelmente também para outros tipos de linguagens de especificação.

CDN é um *framework* geralmente aplicado ao se raciocinar com diagramas (Blackwell *et al.,* 2004), especialmente quando projetando Linguagens de Programação Visuais (VPLs). Ele tem sido aplicado também para avaliar diferentes ferramentas e linguagens de programação. Como exemplos, as especificações formais em Z no ambiente TranZit (Khazaei & Triffitt, 2002) e técnicas de programação para o usuário final no *Interactive Football Playbook* (Neumann *et al.,* 2009) foram analisados com o CDN. De fato, já que a programação para o usuário final e atividades de desenvolvimento destacam a importância da usabilidade e baixa carga cognitiva em ferramentas de desenvolvimento de *software*, o CDN ganhou popularidade como um *framework* de avaliação nesse contexto (Guerra *et al.,* 2009), dos quais *Web mash-ups* (Danh *et al.,* 2009)(Ennals *et al.,* 2007) e codificação de jogos (Kauhanen *et al.,* 2007) são duas instâncias notórias.

2.3.4.1. Método de Análise

O primeiro passo da análise é poder interpretar NCL como um artefato de informação de algum tipo. Olhar uma linguagem de especificação como um artefato é fácil. Artefatos são objetos não-naturais, objetos feitos pelo homem para um fim particular. Esse inegavelmente sendo o caso com NCL, é necessário mostrar somente que ele é um artefato de informação. Um artefato de informação é um que não apenas carrega informação (o que muitos artefatos fazem já que eles carregam significado), mas cujo propósito primário é representar e suportar o processo de informação. Essa definição parece acomodar todo o espectro de artefatos que o CDN tem sido usado para analisar até agora, e confortavelmente localiza NCL dentro dele para propósito da análise que segue.

Como próximo passo, linguagens computacionais são artefatos com uma natureza dupla. Eles representam informação em um sentido referencial e ainda constroem informação em um sentido generativo (Gelernter &). Em outras palavras, elas podem não somente referenciar informação pré-existente (por exemplo, uma construção de uma linguagem de programação como "se A=1 então ..." testa uma condição existente para verificar se a é igual a 1 ou não), mas podem ainda criar representações como consequência da execução computacional (por exemplo, uma construção como "se A=1 então B=falso" causa a atribuição do valor "falso" a variável B, algo que não existia necessariamente antes). Em termos práticos, essa natureza dupla leva a incluir, na análise de linguagens computacionais, não somente as construções linguísticas que elas oferecem para especificar representações e processos computacionais, mas também sua semântica operacional (ou seja, que efeitos as construções da linguagem acarretam quando elas são interpretadas por um computador). Um terceiro aspecto de linguagens computacionais é adicionado na análise: a estrutura de programação que dá suporte a programadores na criação e interpretação das construções da linguagem. Embora esse terceiro aspecto possa ser certamente considerado um fator externo que não é realmente intrínseco à linguagem sendo analisada (já que sempre é possível haver diferentes editores e ferramentas CASE para produzir programas, independente da linguagem que se está trabalhando), editores de programas, por exemplo, destacam e, de forma extensiva, tornam explícitas certas

características de linguagens de programação. Isso é especialmente verdadeiro para NCL e suas abstrações declarativas.

Na Figura 11 é apresentado como se organiza o espaço de análise. Na parte superior da figura aparece o Documento NCL, o qual é produzido com o suporte de Editores da linguagem. Esses editores tipicamente têm como foco os aspectos léxicos e sintáticos de linguagens de programação, ajudando seus usuários a perceber certas características da linguagem, evitar certos erros, estabelecer certas relações e assim por diante. Na parte inferior da figura, pode ser vista a Apresentação NCL, que é resultado da interpretação computacional do código NCL. Essa apresentação é feita em interpretadores computacionais chamados Exibidores NCL, que produzem representações e comportamento que ajudam seus usuários perceberem o resultado das construções da linguagem, seu significado, escopo e os efeitos para os quais ela pode ser usada. Juntos, e apenas juntos, esses dois determinam a usabilidade de NCL como um artefato de informação. Essa é a razão pela qual são incluídos aspectos de editores e interpretadores na presente análise do reúso em NCL com o *framework* CDN. É importante salientar, como mostra a Figura 11, que certas dimensões são aplicadas a ambos o reúso de código estático e ativo, enquanto outras só fazem sentido (ou são marcadamente mais relevantes) em termos de um ou de outro. Na definição das dimensões CDN, como no caso da visibilidade, devem ser consideradas certas características do ambiente computacional, o que se faz necessário para decidir sobre como elementos da linguagem são “acessados” ou “tornados visíveis”.

A decisão de incluir os editores no escopo da análise é justificada também pela necessidade de avaliar quais são suas características que podem vir a influenciar a concepção de uma linguagem para autoria de famílias de documentos. Um ambiente de autoria pode influenciar decisivamente na forma de se conceber famílias de documentos ao fornecer abstrações gráficas mais concisas ou outros tipos de facilidades.

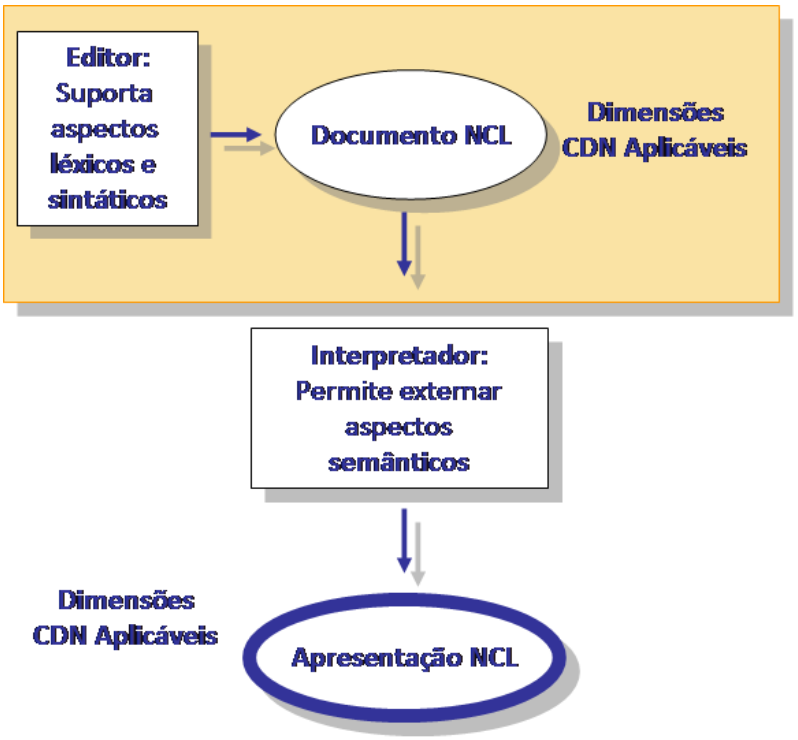

Figura 11. Organizando a aplicação das dimensões CDN no contexto do reúso NCL.

Os editores considerados nessa análise são: um editor de texto XML não-especializado para NCL; o NCL Eclipse (Azevedo *et al.*, 2009); e o Composer (Guimarães *et al.*, 2008). As características avaliadas no primeiro ambiente são intrinsecamente relacionadas à notação NCL, focando em um ambiente quase neutro que não provê funcionalidades específicas de NCL. O NCL Eclipse (Azevedo *et al.*, 2009), no entanto, é uma ferramenta textual de autoria com poucos recursos gráficos que nunca se sobrepõem à exibição do texto. Ele foi proposto para dar suporte à interação do autor diretamente com a notação NCL. O Composer (Guimarães *et al.*, 2008) é uma ferramenta de autoria que faz uso de quatro visões sincronizadas. Usuários do Composer podem focar em uma visão específica dependendo de qual tarefa eles estão realizando. Obviamente, a avaliação de usabilidade de NCL é diferente em cada um desses três ambientes, já que eles constituem diferentes sistemas (notação + ambiente). O interpretador sendo considerado pela análise é o mesmo em todos os três casos e corresponde à implementação de referência do padrão Ginga (ABNT, 2007) (ITU-T, 2009).

As treze dimensões CDN usadas na análise são definidas em (Blackwell & Green, 2003) e listadas abaixo, junto com notas sobre a interpretação específica dada neste estudo para cada uma delas:

• Visibilidade: “habilidade de olhar os componentes facilmente”.

Toda informação necessária é facilmente identificada e acessível quando autores estão editando parte de um documento NCL? Como editores aperfeiçoam a visibilidade?

• Gradientes de Abstração: “tipos e disponibilidade de mecanismos de abstração”.

Quão fácil é dar suporte à composição e encapsulamento em NCL? Como editores se aproveitam da especificação dos níveis de abstração?

• Proximidade de Mapeamento: “proximidade da representação para o domínio”.

Quão próxima está a especificação do documento NCL com relação a sua apresentação? Como editores ajudam com esse mapeamento?

• Consistência: “semântica similar é expressa de forma sintaticamente similar”.

É possível inferir outras construções NCL uma vez que um subconjunto básico da linguagem é aprendido? Há exceções ou casos especiais para serem aprendidos?

• Difusão: “verbosidade da linguagem”.

Quantos elementos NCL de primeira classe devem ser definidos em um documento para produzir resultados visualmente perceptíveis em apresentações NCL?

• Propensão a Erros: “a notação convida a erros e o sistema dá pouca proteção”.

A notação NCL induz a erros de programação ou semânticos? Editores criam novos tipos de indução de erros ou evitam alguns?

• Operações Mentais Difíceis: “alta demanda em recursos cognitivos”.

Quais tarefas de autoria NCL se tornam mais fáceis quando se usam anotações externas? Como editores fazem uso de anotações externas? Que tarefas de autoria NCL podem se tornar mais fáceis por meio de ajudas cognitivas externas? Quais são essas ajudas? Os editores e o exibidor NCL provêem suporte a essas ajudas?

• Dependências Escondidas: “associações importantes entre entidades não são visíveis”.

Como são identificadas as dependências em documentos NCL? Quais são as características de editores projetadas para ajudar a encontrar as dependências?

• Comprometimento Precoce: “restrições na ordem de fazer as coisas”.

Em que ordem os elementos NCL devem ser criados durante a autoria de documentos? Os editores oferecem ordens pré-definidas ou eles permitem a autores criarem os elementos na ordem que eles desejarem?

• Avaliação Progressiva: “o trabalho parcial pode ser checado em qualquer tempo”.

É possível obter retorno sobre apresentações de documentos NCL parcialmente definidas? É possível exibir documentos incompletos?

• Expressividade de Papéis: “o propósito de uma entidade é prontamente inferido”.

Quão óbvio é o papel de cada elemento NCL na solução como um todo?

• Notação Secundária: “informação extra em outra forma que não a sintaxe formal”.

Quais são os tipos de comentários e documentação que podem ser encontrados em documentos NCL? Como editores tratam comentários e metadados?

• Viscosidade: “resistência à mudança”.

Quanto esforço é requerido para fazer uma mudança em um documento NCL? Que tipos de mudanças são mais difíceis de aplicar?

2.3.4.2. Questões de Usabilidade Associadas ao Reúso de NCL

Esta seção descreve algumas necessidades típicas de autores quando criando aplicações hipermídia em NCL. A seção estende a análise feita nas Seções 2.3.2 e 2.3.3 com a avaliação da usabilidade de NCL no contexto de seu uso em três ambientes de autoria. Em vez de avaliar exclusivamente a notação NCL propriamente dita, o estudo avaliou como a notação comunica (aos usuários) seus princípios de projeto e a intenção de seus projetistas com o suporte de diferentes ambientes computacionais, os quais fornecem a infraestrutura necessária para programação em NCL.

A ordem na qual as dimensões cognitivas são discutidas a seguir foi escolhida por fluência apenas. O leitor pode se referir à Seção 2.3.4.1 para revisitar a interpretação dada para cada dimensão cognitiva no contexto desse estudo.

2.3.4.2.1. Visibilidade

O reúso de conteúdo em NCL tem o potencial de introduzir um problema de visibilidade porque um autor não pode ver qualquer conteúdo de mídia (imagens, vídeos, etc.) apenas ao ler o documento NCL. Contudo, esse é um problema inevitável, encontrado em todas as linguagens textuais baseadas na estrutura (*structure-based*) projetadas para a autoria de documentos hipermídia.

O NCL Eclipse oferece suporte à pré-visualização de conteúdo, de tal forma que imagens, áudio e vídeos possam ser facilmente vistos em tempo de autoria. Isso ajuda autores a relacionar o conteúdo mentalmente com o respectivo objeto de mídia que o referencia. Embora esse último aspecto esteja relacionado à dimensão da visibilidade, é claro que ele também alivia a carga de operações cognitivas (outra dimensão CDN) à medida que suporta o reconhecimento do conteúdo em vez de exigir a lembrança dos objetos pelos seus nomes.

Como NCL é uma linguagem textual para a especificação de documentos hipermídia de tamanhos arbitrários, ela sempre vai expor algum problema cognitivo quando um documento maior estiver sendo editado. Isso permanece sendo verdade mesmo se abstrações gráficas mais concisas forem empregadas.

Como mencionado na Seção 2.3.2.3, o reúso de código de objetos de mídia em execução tem se mostrado mais útil, mais ainda que reúso de código estático. Nesse caso, no entanto, o reúso pode diminuir o problema de visibilidade já que um mesmo objeto pode ser repetido em diferentes partes do código, somente com suas interfaces de interesse.

Já que relacionamentos em NCL são sempre definidos fazendo referência a uma relação previamente definida, isso também pode causar problemas cognitivos de visibilidade. Contudo, um açúcar sintático poderia ser adicionado à linguagem para diminuir esse problema, como sugerido na Seção 2.4.

Como discutido na Seção 2.3.2.6, não há forma de definir transições como parâmetros de um elemento <descriptor>. Isso significa que transições são mais suscetíveis a problemas cognitivos de visibilidade que regiões. Embora isso não tenha sido considerado relevante no perfil EDTV de NCL 3.0 (ABNT, 2007), esse é um conceito da linguagem que deve ser revisado na próxima versão.

Em quase a totalidade dos editores XML de propósito geral, é possível colapsar e expandir a estrutura da árvore XML de tal forma que trechos de código possam ser escondidos ou mostrados, conforme desejado. A mesma característica é oferecida tanto nas ferramentas NCL Eclipse quanto no Composer. Isso provê, por si só, algum controle de visibilidade.

NCL Eclipse, em adição, suporta a navegação hipertextual, alternando diretamente de uma posição de edição para outra. Atalhos (hiper-elos) são definidos pelo NCL Eclipse quando um elemento NCL referencia outro elemento. Com essa funcionalidade, um autor NCL pode, por exemplo, rapidamente trocar do corpo do documento para o cabeçalho, a fim de alcançar trechos não-visíveis de código. O ambiente ainda provê um atalho para retornar à posição de edição prévia.

2.3.4.2.2. Gradientes de Abstração

O reúso de estrutura está intimamente relacionado com a dimensão cognitiva de gradientes de abstração. Ao ter composições hipermídia (contextos) como o conceito de abstração central de NCL, é destacada a importância dessa dimensão no projeto da linguagem e a importância dessa dimensão por si só.

A notação NCL é projetada para ajudar autores em diferentes níveis de abstração. Editores XML de propósito geral não oferecem qualquer tipo de suporte ao encapsulamento de fragmentos de código.

A visão estrutural do Composer, contudo, representa o documento como um grafo composto, onde os nós do grafo representam objetos de mídia ou contextos, e as arestas do grafo representam elos. Essa visão revela os diversos níveis de abstração especificados em um documento NCL e torna mais fácil o reúso de estruturas. Como exemplo, autores podem facilmente identificar similaridades

entre estruturas em um nível particular de abstração e escolher reusar o código correspondente às abstrações que eles observam.

2.3.4.2.3. Proximidade de Mapeamento

O suporte NCL à importação está relacionado à dimensão CDN de proximidade de mapeamento, já que esse conceito pode ser considerado um "truque de programação". Importar um documento é fazer seu cabeçalho e corpo "visíveis" ao documento importador. Uma vez que a importação seja feita, o reúso pode ser obtido pela referência a bases de documentos ou partes do corpo do documento importado.

Como previamente mencionado, o NCL Eclipse trata uma especificação NCL como um hipertexto, representando referências a elementos reusados como hiper-elos. Dessa forma, o próprio ambiente hipermídia é uma metáfora das aplicações para as quais ele foi projetado para criar. É importante salientar que mapear o código NCL em apresentações NCL nesse editor é ainda uma operação complexa. Em especial, quando oportunidades de reúso dependem do reconhecimento de funcionalidades de apresentação difíceis de serem mapeadas diretamente em se olhando o código NCL, o ambiente não favorece o reúso.

O Composer, por sua vez, faz uso de múltiplas visões gráficas sincronizadas, cada qual relacionada a aspectos particulares das tarefas de edição ou ainda a um perfil de usuário particular. Essa é a forma como a ferramenta atende a uma ampla gama de necessidades dos usuários, é a mais apta a dar suporte ao reúso de código e a que mais facilita o mapeamento das abstrações quando comparada a outros editores. Na visão estrutural, um documento é visto como um grafo composto, o que pode não ser natural para autores sem conhecimento prévio de programação. A visão temporal, contudo, interpreta o documento como uma abstração que é modificada no decorrer do tempo, o que pode ser mais familiar a produtores de conteúdo audiovisual com pouca proficiência de programação e mais acostumados a edições na linha do tempo. Essas duas representações podem ajudar autores a detectar oportunidades de reúso enquanto estão lidando com a especificação de código NCL, já que essas

oportunidades são fortemente relacionadas a características de apresentação de um documento.

2.3.4.2.4. Difusão

O recurso de autocompletar do NCL Eclipse sugere valores válidos quando se está editando atributos NCL. Quando um autor ativa o autocompletar enquanto está editando o atributo *region* de um elemento <descriptor>, por exemplo, somente identificadores de elementos <region> são oferecidos. Esse mecanismo de lembrete rápido ajuda a reduzir o problema de difusão através da redução da distância de codificação entre elementos relacionados, o que significa que o editor agrega elementos dispersos no código NCL.

Na visão espacial do Composer, autores podem ver e manipular todas as mídias aplicáveis em um dado instante de tempo da apresentação do documento. A visão espacial ajuda a diminuir a difusão por agregar as abstrações do documento em uma única representação concreta na tela, a qual está em correspondência um-para-um com a exibição final do documento.

A difusão de código NCL pode criar problemas quando o reúso de um elemento introduz cadeias de dependência (ou seja, um elemento reúsa outro, o que reúsa outro, e assim por diante). Algumas cadeias comuns de elementos são:

- media -> descriptor -> region
- media -> descriptor -> transition
- switch -> rule -> bindRule

Essa característica NCL em criar cadeias está também relacionada à dimensão de dependências escondidas, discutida a seguir. Com respeito à difusão, é importante observar que a "verbosidade" NCL de alguma forma contribui para agravar o problema de dependências escondidas, e que a infraestrutura existente para edição e exibição de documentos NCL ainda não provê soluções robustas para ela.

2.3.4.2.5. Propensão a Erros

Dois tipos de erros podem ser observados. O primeiro é simplesmente o que gera um código NCL defeituoso e torna a apresentação de um documento NCL impossível. Tais erros podem ser detectados em tempo de autoria. O NCL Eclipse provê o suporte extensivo de mecanismos de validação de erro, o que permite a autores facilmente encontrar e remover erros de programação. O segundo tipo de erros são os semânticos, os quais são encontrados em documentos sintaticamente corretos, mas que não atendem às expectativas ou intenção do autor.

Documentos NCL tipicamente possuem diversos elementos que referenciam outros elementos. Esse é um recurso bem-vindo para o reúso, mas pode criar a necessidade de checar código não-visível. Ao especificar um elo, por exemplo, o autor pode precisar checar a cardinalidade de papéis na especificação do conector referenciado. Essa informação não está disponível imediatamente e pode levar a erros. O mecanismo de referência pode levar a vários outros tipos de erros. Evitar problemas com esse mecanismo é uma questão de projeto de linguagem e uma necessidade em ferramentas de autoria, o que também está fortemente relacionado à discussão prévia da dimensão da visibilidade.

2.3.4.2.6. Operações Mentais Difíceis

A codificação e reúso NCL ainda oferecem problemas consideráveis relacionados a operações mentais difíceis. Apesar de facilitar a reutilização em muitos outros aspectos, há ainda alguns casos em que é claramente necessário melhorar o suporte ao reúso. Um exemplo disso são as regiões NCL. A região é tipicamente definida por atributos como *left*, *top*, *width* e *height*, que podem ter valores em pixels ou porcentagem (da região pai). Não é tão simples de visualizar mentalmente a área da região durante a leitura dos valores de atributos de região. A região também possui a informação de sobreposição (atributo *zIndex*), indicando qual região deve ser apresentada quando duas ou mais delas ocupam a mesma área da tela. Os autores também geralmente precisam ver o conteúdo desejado exibindo na respectiva região, a fim de encontrar a sua posição e dimensão adequadas. Além disso, a área de exibição do objeto pode mudar

durante sua apresentação. Portanto, a definição e controle da geometria de regiões são consideravelmente difíceis, e normalmente requerem ferramentas associadas. O problema é pior em situações de reúso, onde as informações relacionadas são espalhadas em vários elementos.

A pré-visualização de conteúdo em sua respectiva área da tela pode ajudar autores a realizar tal operação mental difícil. Como anteriormente mencionado, a visão de leiaute do Composer suporta a especificação de regiões por manipulações gráficas. Basicamente, o Composer exibe um retângulo para cada área da tela. A posição e dimensão podem ser mudadas usando operações de arrastar e soltar (*drag and drop*). A visão de leiaute também permite a edição de áreas da tela relacionadas a instantes específicos na linha do tempo da apresentação, ajudando autores a fazer mudanças a parâmetros previamente definidos.

2.3.4.2.7. Dependências Escondidas

Quando autores editam um elemento NCL, outros elementos que fazem referência a ele irão mudar como efeito colateral. Isoladamente um elemento referenciado não possui informação sobre que outros elementos fazem referência a ele. Dependências não são visíveis no elemento referenciado.

O NCL Eclipse oferece um mecanismo de hiper-elo inverso, previamente mencionado, com o qual usuários podem ver elementos que referenciam um elemento sendo editado. Dessa forma, autores podem checar dependências em ambas as direções, possivelmente identificando qual o impacto da mudança.

As visões sincronizadas do Composer oferecem retorno imediato sobre mudanças aos autores, ajudando a reduzir os efeitos de dependências escondidas.

Há diversos exemplos de problemas de dependências escondidas especificamente relacionados ao reúso. Como exemplo, quando um atributo de uma região muda, todos os descritores associados colateralmente recebem o efeito dessa mudança. De forma similar, mudanças em descritores têm impacto em objetos de mídia que fazem referência a eles. É claro, esse efeito de dependência escondido pode ser evitado se autores especificarem todas as propriedades de apresentação usando elementos <property>, o que traz como inconveniente uma redução no reúso, como discutido anteriormente.

2.3.4.2.8. Comprometimento Precoce

O NCL Eclipse força o comprometimento precoce porque ele valida o documento a cada nova edição. Isso dá aos autores um retorno instantâneo em erros de codificação e desencoraja a criação de documentos que são propositalmente incompletos ou inválidos. Por outro lado, forçar uma ordem pré-definida de especificação de elementos pode ser de pouco interesse dos autores, que geralmente escolhem trabalhar usando suas próprias estratégias. Assim, no NCL Eclipse, autores podem desligar as mensagens de aviso geradas pelo recurso de validação de código e construir especificações em uma ordem diferente que aquela que produz código correto. O problema é que, como é enfatizado na dimensão CDN de avaliação progressiva, os autores não ficam aptos a ver o resultado (a apresentação) de seu código parcial. O Exibidor NCL requer construções sintaticamente corretas para tocar um documento.

O Composer ainda tem um mecanismo muito primitivo para a validação de código quando comparado ao NCL Eclipse, e não oferece recursos adicionais para a exibição de código NCL parcialmente especificado.

2.3.4.2.9. Avaliação Progressiva

Nem o NCL Eclipse ou o Composer suportam qualquer tipo de avaliação progressiva, ou provêem retorno parcial para programas incompletos. Ambos os ambientes são integrados a um Exibidor NCL externo que exibe somente documentos válidos e sintaticamente completos. Esse é um aspecto de usabilidade importante, não somente com relação ao reúso, mas em geral. Em termos amplos, ele força uma estratégia particular *top-down* por parte dos autores, a qual pode ir contra a expectativa e preferência de autores individuais. Esse é, portanto, um item importante de requisitos para o aperfeiçoamento da usabilidade em ferramentas de autoria NCL.

2.3.4.2.10. Expressividade de Papéis

A dimensão de expressividade de papéis ajuda a explicar por que NCL é facilmente aprendida por não-programadores e como características de reúso ajudam nessa tarefa. Há elementos <media> para representar objetos de mídia (conteúdo). Sentenças causais são expressas por elementos <link> de primeira classe, os quais reusam especificações de relações temporais definidas em conectores, os quais são também elementos de primeira classe (<causalConnector>). O mesmo padrão é repetido para todos os elementos necessários para especificar um documento. Cada elemento tem um papel claramente definido no modelo conceitual de NCL.

O Composer se apóia fortemente na expressividade de papéis do modelo conceitual de NCL ao prover diversas visões de autoria. O recurso de autocompletar do NCL Eclipse também explora a expressividade de papéis extensivamente. Sugestões de código são oferecidas dependendo da posição do documento em que autores estão editando. Ao oferecer sugestões dentro de um elemento <ncl>, por exemplo, o NCL Eclipse mostra apenas dois possíveis elementos filho: <head> ou <body>. O mesmo ocorre quando a ferramenta sugere valores de atributos, em particular valores de referência a identificadores de outros elementos sendo reusados.

2.3.4.2.11. Notação Secundária

Comentários XML provêem uma notação secundária para NCL. NCL permite ainda a definição de metadados, através de seus elementos <meta> e <metadata>. Isso pode aperfeiçoar o código fonte com informação semântica adicional. Em especial, notações secundárias podem ajudar a fazer a documentação sobre o emprego de reúso.

O NCL Eclipse suporta o destaque (*highlighting*) de código, usando diferentes fontes e cores para nomes de elementos e atributos e valores de atributos. O ambiente faz uso de outros tipos de notação secundária existentes em NCL, tais como comentários XML e metadados. O NCL Eclipse também suporta em sua versão mais recente uma notação própria para documentação.

A visão textual existente atualmente no Composer não suporta o destaque de código. Em outras visões do Composer, poderia haver outras anotações extras, mas elas não estão implementadas na versão atual. Comentários XML não são explorados por qualquer uma das visões do Composer, exceto pela visão textual.

Notações secundárias são ignoradas pelo Exibidor NCL.

2.3.4.2.12. Consistência

O entendimento dos três elementos básicos de NCL (<media>, <link> e <context>) guia o entendimento dos outros elementos, o que está relacionado com a dimensão cognitiva da consistência, a qual analisa o quanto pode ser inferido da notação em se sabendo apenas parte dela. Os elementos <descriptor> e <region>, por exemplo, são opcionalmente usados em conjunto com o elemento básico <media>. Da mesma forma, elementos <causalConnector> podem ser usados juntos com elementos básicos <link>. Elementos de interface (<area>, <property> e <port>) são usados juntos com elementos <media> ou <context>.

É importante mencionar novamente (vide Seção 2.3.2.6) que é impossível reusar uma regra ao se definir outras regras. Essa é uma rara exceção na coerência do reúso em NCL, como mencionado previamente. Ainda com relação à consistência, conforme apresentado na Seção 2.3.2.6, regras podem se referir a propriedades globais sem que elas tenham sido explicitamente declaradas. Como mencionado naquela seção, esse é um conceito da linguagem a ser revisado em futuras versões.

NCL tem diversos méritos em termos de consistência. Como mais extensivamente discutido nas Seções 2.3.2 e 2.3.3, o entendimento de um pequeno conjunto de conceitos centrais e poucos elementos XML é suficiente para autores anteciparem e induzirem o todo. Nesse estudo, não são considerados relevantes para a dimensão CDN da consistência aspectos referentes à infraestrutura de edição ou de exibição do documento NCL, o que tem por objetivo relacionar bem mais a dimensão da consistência estritamente a aspectos linguísticos de NCL (a notação).

2.3.4.2.13. Viscosidade

De certa forma, a viscosidade é uma dimensão cognitiva afetada pela maioria (ou talvez todas) as dimensões previamente discutidas. Essa dimensão avalia o quanto difícil é fazer mudanças em um objeto de interesse manipulado por um artefato de informação. No contexto do reúso em NCL, isso praticamente sugere perguntar o quanto difícil é reusar objetos. As subseções anteriores sugerem que os ambientes de autoria e exibição NCL ainda são viscosos principalmente por causa de alguns aspectos relacionados com difusão, operações mentais difíceis, dependências escondidas e comprometimento precoce. Mecanismos para controlar a visibilidade e níveis de abstrações, para suportar diferentes tipos de mapeamento entre representações e objetos de domínio, para diminuir a propensão a erros, para aumentar a expressividade de papéis, todos juntos com um projeto consistente de linguagem e flexibilidade em incorporar notações secundárias, ajudam usuários a lidar com os aspectos atuais de viscosidade na programação NCL.

2.4. Requisitos para uma Linguagem de Autoria de Famílias de Documentos levantados pelos Estudos de NCL

Algumas conclusões podem ser obtidas como resultado do estudo empírico sobre o uso de NCL na especificação de documentos hipermídia e do estudo analítico sobre o reúso em NCL. Essas conclusões parecem endereçar e até mesmo sugerir alguns requisitos desejados para a especificação de uma linguagem para autoria de famílias de documentos.

Com base nos dados analisados no estudo empírico, a NCL se mostra uma linguagem eficaz (i.e. cumpre a sua função de criação documentos hipermídia), mas apresenta desafios de usabilidade que devem ser vencidos para tornar-se uma linguagem eficiente para a ampla gama de usuários a que visa servir.

Para um público com formação técnica em computação, a NCL se apresenta como uma linguagem ‘verbosa’, mas não difícil. O desafio de usabilidade se apresenta na forma facilitar a escrita de especificações NCL. Há dois caminhos a seguir. Um é o desenvolvimento de ferramentas de edição de especificações em NCL que aceleram esta escrita, como através de mecanismos do tipo

autocompletar ou da criação de *macros* de preenchimento de código. Outro caminho é embutir na NCL mecanismos sintáticos que permitam uma redação mais enxuta das especificações. Essa última alternativa, porém, se é que é possível, talvez seja mais dispendiosa do que desenvolver outra linguagem, com maior redigibilidade e legibilidade do que a NCL, a partir do mesmo modelo semântico que ela usa (NCM).

Já para um público sem formação técnica em computação, o estudo empírico não permite ver claramente se as dificuldades reportadas em alguns dos depoimentos apresentados são dificuldades de elaborar uma descrição de um aplicativo em termos de relações que expressam apenas causalidade/restrição (em vez, por exemplo, de fazê-lo apenas em termos de relações temporais), ou dificuldades de 'programar', simplesmente. Se a primeira hipótese for a correta, uma ferramenta de autoria aderente ao modelo NCM provavelmente não eliminaria todas as dificuldades reportadas, embora pudesse oferecer mais explicações e analogias para o usuário 'aprender' o modelo enquanto usasse a ferramenta. Já se a segunda hipótese for verdadeira, a eficácia da ferramenta de autoria para sanar o problema seria uma alternativa praticamente certeira.

Em suma, há pelo menos três possíveis esforços que podem ser feitos no sentido de tornar NCL mais usável: i) modificar a linguagem; ii) desenvolver uma linguagem de mais alto nível que possa ser traduzida ou interpretada para NCL; iii) usar um editor de aplicativos para DTV, que abstraia alguns aspectos da NCL. O presente estudo já rendeu alterações na NCL (especialmente na especificação do leiaute de aplicações) que foram incorporadas ao padrão ABNT para o SBTVD. Em sendo uma norma, o esforço de modificação da linguagem é custoso e pode deixar legado. Isso conduz a um foco maior nas duas últimas linhas.

Um requisito básico para uma linguagem voltada à autoria de famílias de documentos pode ser apontado como fruto do estudo empírico realizado: ela precisa ter méritos de usabilidade. Complementando isso, como a tarefa de concepção de documentos hipermídia se assemelha à criação de famílias de documentos, é necessário que a usabilidade dessa linguagem não seja muito inferior à já oferecida por linguagens de autoria hipermídia existentes, como NCL.

O estudo empírico ressalta, também, alguns dos primeiros desafios a serem enfrentados na disseminação da tecnologia de DTV – a elaboração de ferramentas (linguagens, ambientes de autoria, simuladores, etc.) que enderecem ao mesmo

tempo as necessidades e expectativas de profissionais de TV, de profissionais de informática e dos telespectadores. Está claro que, do momento que a TV passa a ser ‘interativa’, elementos básicos de há muito conhecidos e dominados por programadores profissionais (e.g. estruturas de controle de desvios de execução, de sincronização, etc.) terão de ser dominados pelos profissionais que atuarão em DTV e até pelos telespectadores. Neste sentido, as iniciativas de capacitação de usuários finais para tornarem-se desenvolvedores de ‘software’ – *end-user development* (Lieberman, 2006) - apresentam-se como uma alternativa promissora a ser perseguida.

Como se nota, há a necessidade, na autoria de documentos hipermídia, de atender a uma ampla gama de perfis profissionais. Essa mesma necessidade, no entanto, não é imediatamente induzida na outra tarefa que é a autoria de famílias de documentos. Com o propósito de promover maior abrangência para a tarefa de criação de documentos, não se deve esperar desses autores conhecimento prévio técnico profundo ou mesmo necessariamente base de conhecimento em programação de computadores. A autoria de famílias de documentos, no entanto, é uma tarefa diferente e pode se esperar que ela seja realizada por um perfil de autor mais especializado. Em suma, não é um requisito para linguagens de autoria de famílias de documento que elas possam ser usadas por autores não-especializados. No máximo, é conveniente fazer uso de conceitos oriundos de modelos para autoria hipermídia (objetos de mídia, composições etc.) com o propósito que autores possam aprender e usar por analogia.

Com relação ao estudo analítico, outros requisitos também podem ser obtidos de suas conclusões. Um primeiro requisito antecede os resultados obtidos com o estudo e é, na verdade, o motivador para sua realização: o nível de reúso oferecido por uma linguagem de autoria para famílias de documentos deve ser equiparável ou maior ao já existente, em uma ampla variedade de aspectos, em linguagens voltadas para a autoria de documentos hipermídia.

A estrutura geral da linguagem NCL é formada por um cabeçalho com bases diversas e um corpo com toda estrutura de organização e de apresentação do documento. Essa estrutura diz muito sobre a preocupação intrínseca com o reúso no projeto da linguagem, em que o corpo do documento recorre constantemente a informações contidas no cabeçalho. Em especial, o estudo analítico demonstra

como o autor de NCL é guiado, desde o primeiro contato com a própria estrutura geral da linguagem, no sentido de criar documentos com alto grau de reúso.

A forma direta em que NCL oferece o reúso endereça outro requisito de uma linguagem para autoria de famílias de documentos: é preciso que o reúso seja promovido naturalmente. Isso significa que a oferta de mecanismos de reúso não pode tornar a autoria mais difícil ou tomar mais tempo por si só.

A separação entre relação e relacionamento hipermídia é uma importante característica de expressividade e reúso de NCL. A definição de tipos de relações é a tarefa mais difícil na autoria de documentos. Contudo, uma vez que relações sejam definidas, elas podem ser reusadas em diversos relacionamentos. Como previamente mencionado, é comum ver produtores de aplicações DTV terem uma base bem definida de relações feita por programadores peritos e compartilhada entre autores iniciantes de documentos. Problemas de visibilidade em usar uma relação pré-definida podem ser resolvidos usando uma boa nomenclatura para identificação de relações. Uma terminologia que seja bem conhecida pelos autores, como exemplificado na Seção 2.3.2.4, é um trunfo em todas as tarefas de autoria.

A prática habitual do reúso como forma de facilitar a construção de aplicações por parte de autores menos especializados é um efeito colateral que não pode ser esquecido no projeto de uma linguagem para autoria de famílias de documentos. É importante permitir que autores menos especializados possam apenas instanciar e não necessariamente criarem novas famílias de documentos. Esse aspecto pode ser resumido como outro requisito: é preciso haver facilidades para autores menos especializados apenas usarem famílias, sem que autores mais especializados percam em expressividade ao criarem ou modificarem famílias.

É raro que uma aplicação DTV requisite conectores outros que os usuais. Diversas palavras reservadas são definidas por NCL a fim de facilitar a criação de relações espaciais e temporais. Contudo, nos raros casos onde a definição de novas relações é necessária, um problema de visibilidade persiste. Relacionamentos são criados distantes no texto de onde as relações são definidas. Como oposto a outras características opcionais de reúso em NCL (como na definição de leiaute) não há opção nesse caso. A fim de superar esse problema, um açúcar sintático já foi proposto para inclusão em novas versões da linguagem (Soares *et al.*, 2010). Ele permite a definição conjunta de um relacionamento e sua

relação em um pseudocódigo, escrito quase em linguagem natural, como conteúdo de um elemento <link>. Um *parser* NCL pode traduzir esse elemento em elementos NCL convencionais <causalConnector> e <link> de versões anteriores da linguagem.

A definição de relacionamentos como entidades de primeira classe, permitindo que eles possam fazer parte de composições hipermídia (contextos) provê semântica de apresentação a essas estruturas. Em NCL, relacionamentos que são externos a um contexto podem atuar sobre eles através de pontos de interface bem definidos. Dessa forma, contextos encapsulam objetos e seus relacionamentos internos. Esse conceito é bem simples, mas bem difícil de controlar (ABNT, 2007). Ele fornece composicionalidade a contextos permitindo a prova formal de propriedades sobre documentos (Felix *et al.,* 2002), em adição a um rico potencial para o reúso.

Outro requisito para uma linguagem de autoria de famílias de documentos pode ser apontado: composições hipermídia devem ser avaliadas como abstração central no processo de especificação de famílias. Isso pode significar, inclusive, fornecer melhorias (ou enriquecer) na especificação de composições de forma a atender melhor às necessidades de definição de famílias.

A importação e aninhamento de documentos NCL e o suporte à importação de bases tornam bem mais escalável o processo de construção de documentos complexos e grandes sem trazer prejuízos para a elaboração de documentos simples e pequenos. Os mecanismos de reúso e importação da linguagem permitem que se racionalize e compreenda mais facilmente a apresentação final, agrupando informações semânticas e acelerando o raciocínio inferencial. Essa é mais uma consequência do encapsulamento oferecido pelos contextos.

O fato de a linguagem NCL separar diversos aspectos do processo de autoria em bases isoladas poderia sofrer a crítica de pulverizar em pontos diversos do documento informações relevantes sob o pretexto de proporcionar o reúso de tais bases. Na prática, não é o que ocorre. Isso porque a informação presente em cada base é autocontida. A referência que se faz a elementos de uma base por outro elemento ajuda a enriquecê-lo com informações que podem ser entendidas como de primeira classe e não como informações parciais, sem significado por si só. Mais ainda, o reúso é uma característica opcional na grande maioria dos casos e pode ser evitado, se necessário. Isso endereça diretamente outro requisito para

uma linguagem voltada à autoria de famílias de documentos: o reúso precisa ser promovido, não forçado.

Com relação à infraestrutura de edição e exibição de NCL como um artefato de informação, a análise CDN sugere que, para o reúso em particular, mas também para a especificação NCL em geral, os ambientes de autoria atualmente disponíveis ainda são viscosos. A viscosidade vem principalmente de aspectos relacionados à difusão, operações mentais difíceis, dependências escondidas, e comprometimento precoce. Contudo, os méritos de NCL em termos de consistência no projeto da linguagem e múltiplos recursos, em termos de visibilidade, abstração, mapeamento conceitual, e expressividade de papéis, entre outros, contribui para aliviar os problemas.

O reúso não apenas de código estático (objetos de dados NCL), mas de códigos em execução (objetos de representação NCL) conferem à linguagem uma facilidade ímpar de reúso não encontrada em outras linguagens declarativas para concepção de documentos hipermídia, tornando uma aplicação muito mais limpa, de fácil compreensão e menos propensa a erros de programação.

O projeto da linguagem NCL e do modelo hipermídia que fornece a base semântica da linguagem foram concebidos desde o início visando conduzir o autor de uma aplicação, naturalmente, à criação de documentos com alto grau de reúso. Durante a autoria, assim que um documento NCL começa a crescer, há a tendência imediata do autor organizar o documento em contextos. Isso é suficiente para promover o reúso e estabelecer que o próprio uso continuado de NCL conduza a boas práticas de programação.

Um último requisito deve ser estabelecido para a concepção de uma linguagem para autoria de famílias de documentos. O projeto dessa linguagem precisa ser pensado tendo como foco principal o reúso e não o tratando como um aspecto marginal. As dimensões CDN podem ser um poderoso instrumento de análise para fundamentar essas decisões de projeto.

Finalmente, a lista a seguir tem como objetivo resumir os requisitos apontados para uma linguagem de autoria de famílias de documentos, os quais são oriundos da análise de NCL na tarefa de criação de documentos hipermídia e de seu suporte ao reúso:

- precisa ter méritos de usabilidade (ser fácil de usar);

- seu emprego deve ser mais indicado que o uso direto de linguagens para autoria de documentos hipermídia sempre que uma estrutura de repetição for identificada (uma família de documentos é caracterizada);
- autores que criam famílias de documentos podem ser mais especializados, mas autores que apenas usam famílias para conceberem documentos devem precisar conhecer menos requisitos técnicos;
- o reúso precisa ser promovido naturalmente, no nível de domínio da linguagem (não deve ser mais difícil criar famílias apenas porque são mais facilmente reusáveis);
- o conceito de composição hipermídia deve ser revisitado para atender às necessidades de famílias de documentos;
- o reúso precisa ser promovido, não forçado;
- o projeto da linguagem precisa ser pensado e refletir o foco no reúso.