Leiautes Dinâmicos para Documentos Multimídia Baseados em Templates

Glauco Fiorott Amorim
CEFET-RJ
Rio de Janeiro, RJ
glauco.amorim@cefet-rj.br

Débora Christina Muchaluat-Saade
Lab. Midiacom - UFF
Niterói, RJ
debora@midiacom.uff.br

ABSTRACT

Template-based languages can be used for arranging interface components in a layout model, like a grid. Declarative multimedia authoring languages, such as NCL (Nested Context Language), may use those templates for decreasing the authoring effort while specifying a presentation spatial layout. Although layout models are helpful for specifying presentation characteristics for media items, they usually do not consider the case where changes may happen at runtime. Moreover, presentations may lose tidiness when displayed on a device different then the one it was designed for or due to the viewer context or even due to viewer interaction. This thesis proposes STyLe, a template language for dynamic spatial layout. STyLe is a constraint-based template language for providing dynamic and adaptive spatial layouts for hypermedia documents. It also presents a framework capable of interpreting this language and performing the necessary changes in order to dynamically update NCL media object presentation characteristics at runtime.

KEYWORDS

Dynamic Spatial Layout, Adaptive Layouts, STyLe, NCL, Spatial Constraints, XTemplate, Template Authoring

1 INTRODUÇÃO

Este artigo apresenta um resumo dos resultados obtidos na tese de doutorado intitulada *Leiautes Dinâmicos para Documentos Multimídia Baseados em Templates*, defendida no Programa de Pós-graduação em Computação da Universidade Federal Fluminense (UFF) em 29 de agosto de 2017. O trabalho foi desenvolvido por Glauco Fiorott Amorim, num período de 53 meses, sob a orientação da da professora Débora Christina Muchaluat-Saade. A professora orientadora é membro do Programa de Pós-graduação em Computação da Universidade Federal Fluminense. O trabalho desenvolvido na tese está relacionado às áreas da computação de sistemas multimídia e hipermídia.

O trabalho desenvolvido tem como objetivo criar ferramentas para que autores de aplicações multimídia possam criar leiautes adaptáveis dinamicamente. Essa adaptação pode ocorrer tanto em relação à quantidade de objetos de mídia que são apresentados ao mesmo tempo na tela do exibidor, quanto a modificações que acontecem em tempo real com esses objetos de mídia.

O leiaute de uma aplicação é tão importante quanto seu conteúdo, tanto que as ferramentas desenvolvidas para facilitar a autoria das aplicações possuem elementos específicos para definição dos leiautes. Por exemplo, o *kit* de desenvolvimento *Android Studio* [8] utilizado para criação de aplicações móveis em aparelhos que executam o sistema operacional *Android* possui uma visão separada com diversas facilidades para o desenvolvimento do leiaute da aplicação.

Aplicações interativas multimídia como aplicações hipermídia, *slides* interativos, videoaulas, etc, têm seguido o mesmo caminho e enriqueceram a forma como o conteúdo é apresentado para o usuário. Isso tem sido feito para tentar melhorar cada vez mais a experiência do indivíduo ao interagir com a aplicação.

Como aplicações multimídia interativas podem ser executadas em vários dispositivos com diferentes tipos de telas, por exemplo: TVs, *tablets* e *smartphones*, é interessante que o leiaute dessas aplicações seja adaptável para que a experiência dos usuários ao usar a aplicação possa ser tão boa quanto possível, independentemente do dispositivo usado. Um leiaute adaptativo é, portanto, capaz de se modificar dependendo das características de apresentação do dispositivo ou do contexto onde está sendo apresentado.

A adaptação pode ser realizada tanto em tempo de autoria quanto em tempo de execução. Um exemplo de adaptação em tempo de autoria pode ser visualizado quando o autor não sabe antecipadamente quantos objetos de mídia participarão da aplicação. Isso pode acontecer quando um *template* é utilizado. Nesse caso, há a necessidade de se criar elementos que possam se adaptar para que todos os objetos de mídia possam ser apresentados. Os leiautes com essa características são chamados de adaptativos. Quando a adaptação é realizada em tempo de execução, as alterações ocorrem enquanto a aplicação é apresentada, como por exemplo, alterar a posição de um objeto de mídia porque outro objeto foi inserido na aplicação. Os leiautes com essa características são chamados de dinâmicos.

A tese apresenta STyLe (Spatial Template Language) [1], uma linguagem de *templates* baseada em restrições para especificar leiautes espaciais adaptativos e dinâmicos para aplicações multimídia. Chamamos de *leiaute espacial adaptativo* uma abordagem que possibilita que autores possam criar características de apresentação genéricas que adaptam o leiaute espacial especificado ao número de objetos de mídia de um dado documento. Isto diminui o esforço de autoria na criação do leiaute espacial da aplicação.

Um *leiaute espacial adaptativo e dinâmico* é uma extensão de um leiaute adaptativo de tal forma que as características de apresentação dos objetos de mídia possam ser alteradas em tempo de execução e em resposta a ocorrência de eventos na apresentação do documento, tais como interação do usuário ou edição ao vivo de partes do documento. STyLe fornece esta facilidade definindo o leiaute espacial de um documento hipermídia através de restrições

In: I Concurso de Teses e Dissertações (CTD 2019), Rio de Janeiro, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2019.
ISSN 2596-1683

[1] http://www.ic.uff.br/index.php/pt/pos-graduacao/teses-e-dissertacoes

espaciais. Adicionalmente, o uso de restrições no STyLe pode também ser usado para adaptar a aplicação a diferentes tamanhos de tela do dispositivo exibidor.

STyLe define um conjunto de restrições espaciais predefinidas, incluindo as representações dos modelos de leiautes propostos em [1]. Deste modo o leiaute espacial definido no *template* é mantido quando uma mudança ocorre durante a apresentação do documento. Além das restrições predefinidas, STyLe possibilita ao autor personalizar os tipos de restrições espaciais através de conectores de restrição [10].

O uso de um *template* STyLe é processado por um *framework* capaz de interpretar as restrições espaciais contidas no *template* e construir ou atualizar corretamente o leiaute espacial de um documento. Esta tese apresenta uma extensão da linguagem NCL para dar suporte ao uso de STyLe para definição de leiautes espaciais e também uma implementação do framework STyLe para documentos NCL.

A linguagem STyLe fornece *templates* somente para definição do leiaute espacial da aplicação. As relações temporais entre os objetos de mídia devem ser estabelecidas diretamente na linguagem alvo ou geradas a partir de linguagens de *templates* de documentos como XTemplate [2] e TAL [11]. Contudo, é possível criar uma solução completa utilizando STyLe diretamente em um documento NCL ou referenciando *templates* STyLe dentro de um template XTemplate. Para tal, extensões para ambas as linguagens são necessárias e foram apresentadas no decorrer da tese.

O restante deste artigo está estruturado como segue. Na Seção 2 são apresentados os trabalhos relacionados. A Seção 3 discute as contribuições da tese e resultados relevantes obtidos. Na Seção 4, apresentamos os principais produtos da tese, como publicações e softwares desenvolvidos.

2 TRABALHOS RELACIONADOS

Esta tese trata de diferentes temas abordados na literatura, tais como leiautes adaptativos, relações espaciais, linguagem de autoria baseada em *templates* e alteração dinâmica de documentos.

Pesquisas, como [9, 14], apresentam modelos de leiautes dinâmicos e adaptativos. Em [9], os autores propõem um sistema para criar e apresentar documentos em grade para acomodar várias condições de visualização. De acordo com os autores, um estilo de leiaute adaptativo é codificado como um conjunto de *templates* baseados em grade que sabe como se adaptar ao tamanho das páginas e outras condições de visualização. Os *templates* possuem vários tipos de visualização e definem, através de relacionamentos baseados em restrições, como os elementos (texto, figuras, etc) devem ser arranjados usando como base propriedades do conteúdo do elemento, como tamanho ou proporção de uma figura, e propriedades de visualização onde o conteúdo será mostrado, como por exemplo o tamanho da tela do exibidor. Uma extensão desse trabalho é apresentada em [14] e traz algumas novas ideias com relação ao trabalho anterior. Esses dois trabalhos serviram de base para algumas ideias apresentadas nesta tese, como uma linguagem de mais alto nível para definir as restrições espaciais do *template*.

Existem ainda ferramentas que trabalham com a definição de características de leiaute e/ou autoria de documentos XML baseada em *templates* para documentos web, como: CSS Regions Module [3] e AXEL [5]. O Módulo de Regiões CSS permite que o conteúdo de um ou mais elementos ultrapasse de uma para outras áreas denominadas regiões CSS. Além disso, permite que leiautes dinâmicos sejam instanciados. Já AXEL (*Adaptable XML Editing Library*) é uma biblioteca *JavaScript* para autoria de conteúdo XML baseado em *templates* na Web. Baseia-se no uso de *template*, que é utilizada para expressar restrições estruturais no conteúdo editado. A construção do documento é feita sem uma clara divisão entre o *template* e o conteúdo do documento.

Outros trabalhos têm um foco maior na discussão sobre as relações espaciais como em [6, 12, 13, 17]. Em [6], o autor define relações espaciais para documentos hipermídia. Essas relações podem ser estabelecidas entre janelas ou regiões. Embora o conjunto de relações seja interessante, sua utilização é desnecessariamente complexa. STyLe simplifica as relações definidas em [6] e fornece a mesma expressividade. Além disso, a linguagem proposta nesta tese é baseada em *template* e estabelece uma clara separação entre definições espaciais e temporais. Os autores em [17] definem um modelo, baseado em um conjunto de relações espaçais e temporais entre os objetos, para especificar aplicações multimídia.

É importante notar que a expressividade da linguagem STyLe está fortemente ligada a possibilidade de se fornecer todas as relações espaciais determinadas pelos trabalhos anteriores. STyLe utiliza uma extensão dos modelos de leiautes adaptativos introduzido em [2] como uma de suas características para criar as relações espaciais desejadas.

Autores em [18, 19] propõem módulos CSS para criar leiautes de aplicativos mais responsivos. O *CSS Flexible Box Layout* [18] é usado para criar regiões pai, chamadas *flex container* e regiões filhas, chamadas *flex item*. Os módulos manipulam algumas das propriedades espaciais do *flex container* e do *flex item* para torná-los adaptáveis. Já o *CSS Grid Layout* [19] controla o dimensionamento e posicionamento das regiões e seus conteúdos. Ao contrário do *CSS Flexible Box Layout*, ele trabalha considerando a disposição dos itens em ambas as dimensões. Os autores podem adaptar o layout espacial do aplicativo às alterações no dispositivo de apresentação. Embora esses módulos tragam mais poder para a linguagem CSS, ambos não usam relacionamentos espaciais para criar um leiaute. Esse fato pode causar problemas, como sobreposição de itens. A linguagem proposta na tese apresentada fornece contêineres tipados e relações espaciais que podem ser usadas para criar leiautes dinâmicos.

3 CONTRIBUIÇÕES E RESULTADOS

Ao se atingir os dois objetivos principais destacados nesta tese, que são: definição de uma linguagem para criação de leiautes espaciais adaptativos e dinâmicos e construção de uma arquitetura capaz de realizar os ajustes espaciais necessários quando o documento é alterado em tempo de exibição, as contribuições alcançadas foram: Linguagem STyLe para autoria de modelos de leiaute adaptativos e dinâmicos baseados em restrições espaciais para documentos multimídia, extensão da linguagem NCL para uso da linguagem STyLe, implementação do processador da linguagem STyLe para documentos NCL, versão 4.0 da linguagem XTemplate para uso da linguagem STyLe, implementação do processador XTemplate 4.0 para documentos NCL e especificação e implementação de uma arquitetura que possibilite a criação de leiautes adaptativos e dinâmicos a partir

da linguagem STyLe e que responda às alterações das características de apresentação de documentos multimídia NCL em tempo de exibição.

Como mencionado anteriormente, STyLe é uma linguagem baseada em XML, usada para definição de *templates* de leiaute espaciais que serão instanciados em documentos hipermídia. Ela estende a linguagem de leiaute espacial definida em XTemplate 4.0 [1], acrescentando restrições. Além da linguagem, um *framework* foi desenvolvido para permitir mudanças dinâmicas das características de apresentação de objetos de mídia que são exibidos na tela de um dispositivo.

As restrições permitem especificar a definição do leiaute de um documento através de relações espaciais que utilizam atributos de posição, tornando-se uma solução flexível. Esta é uma das principais vantagens da utilização de restrições para definição do leiaute espacial.

Quando um leiaute espacial é definido utilizando restrições, ele pode se adaptar ao tamanho da tela, por exemplo, e pode ser alterado, em tempo de execução, quando acontecem eventos durante a apresentação do documento. Isto ocorre por que as restrições deverão ser sempre satisfeitas independente das mudanças que ocorrerem. A ideia é manter o leiaute definido no *template* STyLe durante toda a execução do documento.

O elemento básico em um *template* STyLe é o *item*. Ele define uma área retangular no leiaute espacial do documento. A escolha de uma área retangular deve-se ao fato de que a maioria das linguagens de autoria para aplicações multimídia interativas, tais como NCL e SMIL, utilizam regiões retangulares para definição espacial de objetos de mídia. Um *item* representa características de apresentação que serão atribuídas a, possivelmente, muitos objetos de mídia.

Em STyLe, *itens* podem ser agrupados dentro de um elemento espacial, denominado *container*. Um *container* pode especificar um tipo que define como os itens declarados dentro desse *container* serão organizados no espaço. Os seguintes tipos estão disponíveis: *gridLayout*, *flowLayout*, *stackLayout* e *carouselLayout*. Quando o tipo do *container* não é especificado, a organização dos itens deve ser estabelecida pelas restrições que são declaradas dentro do próprio *container*. O *containers* tipados facilitam a especificação do layout, pois o autor não precisa especificar as relações espaciais entre componentes (*itens* ou *containers*) contidos no *container*. Já os *containers* sem um tipo predefinido dão bastante flexibilidade ao modelo de layout, pois o autor pode especificar diferentes tipos de restrições espaciais entre seus *itens*. STyLe permite o aninhamento de *containers*.

Itens e *containers* definem pontos de interface que representam coordenadas específicas em sua área retangular. Seus valores possíveis são: *width*, *height*, *top*, *middle*, *bottom*, *left*, *center* e *right*. Esses pontos de interface são atributos utilizados nas restrições.

STyLe propõe três tipos de restrições predefinidas para especificar um leiaute espacial, que são: *align*, *distribute* e *size*. Ainda é possível para o autor do *template*, definir seu próprio tipo de restrição. Uma restrição definida pelo autor é determinada utilizando-se um conector de restrição (*constraint connector*). De acordo com [16], "um *constraint connector* é uma relação multiponto com semântica de restrição, que define uma expressão assertiva. Esta expressão deve ser satisfeita durante a execução do documento".

Para mostrar a utilização de STyLe com uma linguagem de autoria multimídia, estendemos NCL. Para executar o documento NCL estendido, é necessário processar o *template* STyLe junto com o documento NCL estendido para transformá-lo em um documento NCL padrão, como mostrado na Figura 1.

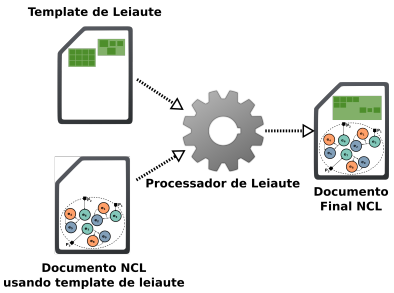

Figura 1: Processamento do documento NCL com leiautes STyLe

Um *template* STyLe define *itens* e/ou *containers* para representar regiões espaciais onde grupos de objetos de mídia serão posicionados. O documento que usa um dado *template* espacial associa seus objetos de mídia aos *itens* e/ou *containers* do *template* através de rótulos (atributo *layout*). Relações de restrição que relacionam componentes espaciais são declaradas no *template* STyLe. Em tempo de processamento, esses componentes dão origem a regiões NCL que serão ocupadas por objetos de mídia declarados no documento que utiliza o *template* espacial.

Para que mudanças no leiaute espacial da apresentação sejam possíveis em tempo de execução, desenvolveu-se um *framework* que utiliza como base a arquitetura descrita em [15] e mostrada na Figura 2.

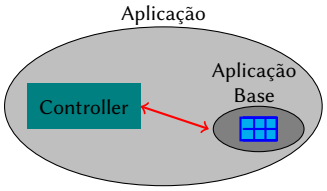

Figura 2: Estrutura do framework STyLe

Como visto na Figura 2, o documento NCL, que terá seu leiaute espacial alterado dinamicamente, é denominado *Aplicação Base*. Esta aplicação será inserida em outra Aplicação NCL que contém o componente *controller*, responsável por capturar ocorrência de eventos e realizar as mudanças necessárias no leiaute espacial da Aplicação Base em tempo de execução.

Os eventos capturados pelo componente *controller* podem ser relacionados a começo/fim da apresentação de objetos de mídia, a seleção desses objetos pelos usuários ou a recepção de comandos de edição ao vivo vindos das emissoras. O componente *controller* foi implementado em Lua, que é usada como linguagem de script auxiliar para aplicações NCL.

Baseado na aplicação base original e no *template* STyLe usado por ela, o *controller* constrói um mapa inicial de apresentação para os

objetos de mídia da aplicação base. Sempre que o *controller* captura uma ocorrência de um evento, ele verifica se tal evento gera uma atualização do mapa de apresentação, uma mudança na estrutura do documento, ou ambos. O início/fim da apresentação de um objeto de mídia é um caso típico de atualização no mapa de apresentação. A recepção de um comando de edição ao vivo é um caso típico de alteração na estrutura do documento e no mapa de apresentação. Após identificar a natureza da modificação, o *controller* irá enviar comandos de edição para mudar o leiaute espacial da apresentação.

O componente *controller* é composto por três componentes, que são: o processador STyLe, um componente denominado *diff* e um *solver*. A arquitetura do *controller* é apresentada na Figura 3.

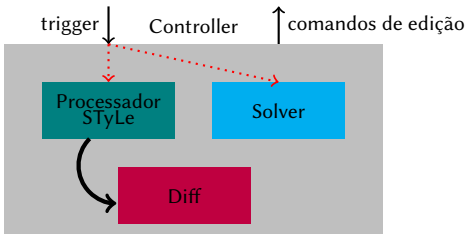

Figura 3: Arquitetura do Controller

Depois de receber o disparo da ocorrência de um evento, o *controller* decide se tal evento precisa ser tratado ou pelo processador STyLe ou pelo componente *solver*. O processador STyLe é escolhido quando acontece uma alteração na estrutura do documento e o componente *solver* é escolhido quando acontece apenas uma atualização no mapa de apresentação.

O processador STyLe é usado para gerar novos relacionamentos de sincronização (links NCL) para novos objetos de mídia incluídos na aplicação base (por comandos de edição ao vivo) ou para atualizar o conjunto de relações existentes quando objetos de mídia são removidos da aplicação base. Isto é feito pelo reprocessamento do documento resultante (considerando a edição) com o *template* STyLe. Ambos, documento original e o novo documento processado, são enviados ao componente *diff* para que se possa estabelecer quais modificações serão necessárias. A partir das diferenças encontradas entre os documentos, comandos de edição são gerados e, de forma incremental, aplicados na aplicação antiga. O novo documento torna-se, então, a aplicação base para futuras modificações.

O componente *solver* é usado para resolver o conjunto de restrições do *template* STyLe. Restrições espaciais são representadas e resolvidas utilizando-se um *solver* SMT (*Satisfiability Modulo Theories*) [4]. Este trabalho utiliza o *solver* Yices 2.4 [7].

Foram feitas avaliações tanto da linguagem como da arquitetura proposta. A avaliação de usabilidade da linguagem mostrou que STyLe é percebida como autoexplicativa e fácil de usar, além de manter a verbosidade baixa. O parâmetro utilizado para medir o desempenho da arquitetura foi o tempo de resposta. Vários cenários foram testados e os resultados mostram que a arquitetura tem potencial para ser usada na prática por aplicações NCL. Maiores detalhes dos testes podem ser encontrados na tese.

4 PRODUTOS DA TESE

Como produtos desta tese podemos citar: XML Schema da Linguagem STyLe e da extensão tanto para NCL quanto para XTemplate, que podem ser encontrados nos apêndices do documento entregue no programa.

Além disso, os seguintes artigos são produtos diretos desta tese:

- Adaptive layouts for authoring NCL programs. In WebMedia '13.
- Adaptive layouts and nesting templates for hypermedia composite templates. In WebMedia '15.
- Xtemplate 4.0: Providing adaptive layouts and nested templates for hypermedia documents. In MMM '16.
- Style: Extending ncl for providing dynamic layouts. In WebMedia '16.
- Providing Adjustable and Dynamic Spatial Layouts for Multimedia Applications with STyLe. Submitted to MTAP '19.

Todos os artigos têm como autores: Glauco F. Amorim, Joel A. F. dos Santos e Débora C. Muchaluat-Saade.

É importante ressaltar que, na área de multimídia, a conferência WebMedia (Simpósio Brasileiro de Multimídia e Web) é a principal do Brasil, enquanto a conferência MMM (International Conference on Multimedia Modeling) é uma das principais internacionais.

REFERÊNCIAS

[1] G. F. Amorim, J. A. F. dos Santos, and D. C. Muchaluat-Saade. 2016. STyLe: Extending NCL for Providing Dynamic Layouts. In *Webmedia '16*. ACM, 71–78.
[2] G. F. Amorim, J. A. F. dos Santos, and D. C. Muchaluat-Saade. 2016. XTemplate 4.0: Providing Adaptive Layouts and Nested Templates for Hypermedia Documents. In *MMM '16*. 642–653.
[3] R. Atanassov and A. Stearns. 2014. CSS Regions Module Level 1. http://www.w3.org/TR/css-regions-1/. W3C Working Draft.
[4] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. 2009. Satisfiability Modulo Theories. *Handbook of satisfiability* 185 (2009), 825–885.
[5] S. Sire C. Vanoirbeek, V. Quint and C. Roisin. 2014. A lightweight framework for authoring XML multimedia content on the web. *MTAP* 70 (2014), 1229–1250.
[6] M. S. A. de Moura. 2001. *Relações Espaciais em Documentos Hipermídia.* Master's thesis. PUC-Rio (in portuguese).
[7] B. Dutertre. 2015. *Yices Manual Version 2.4.* SRI International.
[8] Google. 2016. Android Studio. https://developer.android.com/studio/index.html. visitado em Março.
[9] C. Jacobs, W. Li, E. Schirier, D. Bargeron, and D. Salesin. 2003. Adaptive grid-based document layout. *ACM Transactions on Graphics* 22, 3 (July 2003).
[10] D. C. Muchaluat-Saade and L. F. G. Soares. 2002. XConnector & XTemplate: Improving the Expressiveness and Reuse in Web Authoring Languages. *The New Review of Hypermedia and Multimedia Journal* 8, 1 (2002), 139–169.
[11] C. S. Neto, H. F. Pinto, and L. F. G. Soares. 2012. TAL Processor of Hypermedia Applications. In *DocEng '12*. ACM, 69–78.
[12] D. Papadias, T. Sellis, Y. Theodoridis, and M. J. Egenhofer. 1995. Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees. In *SIGMOD '95*. ACM, 92–103.
[13] D. A. Randell, Z. Cui, and A. G. Cohn. 1992. A Spatial Logic based on Regions and Connection. In *KR '92*.
[14] E. Schrier, M. Dontcheva, C. Jacobs, G. Wade, and D. Salesin. 2008. Adaptive layout for dynamically aggregated documents. In *IUI '08*. ACM, 99–108.
[15] L. F. G. Soares, C. S. S. Neto, and J. G. S. Junior. 2014. Framework for Automatic Generation of Hypermedia Applications in Runtime. In *WebMedia '14*.
[16] L. F. G. Soares and R. F. Rodrigues. 2005. *Nested Context Model 3.0: Part 1 – NCM Core.* Technical Report ISSN: 0103-9741. PUC-RJ, Rio de Janeiro.
[17] M. Vazirgiannis, Y. Theodoridis, and T. Sellis. 1998. Spatio-temporal composition and indexing for large multimedia applications. *Multimedia Systems* 6, 4 (1998), 284–298.
[18] W3C. 2017. CSS Flexible Box Layout Module Level 1. https://www.w3.org/TR/css-flexbox-1/. W3C Candidate Recommendation. Accessed 15 march 2018.
[19] W3C. 2017. CSS Grid Layout Module Level 1. https://www.w3.org/TR/css-grid/. W3C Candidate Recommendation. Accessed 15 march 2018.