Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

THACYLA DE SOUSA LIMA

SINCRONISMO TEMPORAL DE TEMPLATES DE LEIAUTE

São Luís
2016

THACYLA DE SOUSA LIMA

SINCRONISMO TEMPORAL DE TEMPLATES DE LEIAUTE

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Carlos de Salles Soares Neto

São Luís

2016

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Lima, Thacyla de Sousa.
Sincronismo Temporal de Templates de Leiaute / Thacyla de Sousa Lima. – 2016.
53 f.

Orientador(a): Carlos de Salles Soares Neto.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2016.

1. Autoria Multimídia. 2. Leiautes. 3. Reúso. 4. Sincronismo. 5. Template. I. Soares Neto, Carlos de Salles. II. Título.

THACYLA DE SOUSA LIMA

SINCRONISMO TEMPORAL DE TEMPLATES DE LEIAUTE

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 01 de Setembro de 2016

BANCA EXAMINADORA

Prof. Dr. Carlos de Salles Soares Neto
(Orientador)
Universidade Federal do Maranhão

Prof. Dr. Mário Antonio Meireles Teixeira
Universidade Federal do Maranhão

Profa. Dra. Simara Vieira da Rocha
Universidade Federal do Maranhão

MSc. Antonio José Grandson Busson
MediaBox Technologies

São Luís
2016

Aos meus pais, Alfredo e Iracema, que sempre me apoiaram.

Agradecimentos

Agradeço primeiramente à Deus por todas as suas bênçãos.

Aos meus pais, Alfredo e Iracema, por todo apoio e dedicação. São meus maiores exemplos de amor, dedicação, e honestidade.

Às minhas irmãs, Thamyla e Thayse, por toda ajuda e suporte.

À todos os amigos e colegas de curso que se fizeram importantes. Aos meus amigos da MediaBox, principalmente André e Busson, pelo aprendizado e contribuição neste trabalho. Aos meus companheiros e amigos do Laws.

Ao meu orientador, Carlos de Salles, por todas as oportunidades e apoio dado.

Aos demais professores do curso, sempre dispostos a ensinar e proporcionar uma educação de qualidade.

À todos que contribuíram direta ou indiretamente na minha formação acadêmica.

“Dificuldades que te surpreendam são os testes aconselháveis em que te cabe encontrar aproveitamento.” - Emmanuel

RESUMO

Templates de leiaute são modelos de documentos reutilizáveis que fornecem definições espaciais para conjuntos de mídias. Suas duas características principais são o aumento do reúso na especificação do sincronismo espacial de mídias e ainda a redução da complexidade do código de aplicações multimídia. No entanto, templates de leiaute não são tipicamente usados como entidades de primeira categoria na especificação do sincronismo temporal de uma aplicação. Para essa finalidade, habitualmente documentos multimídia definem composições que podem ter uma semântica implícita (como nos containers temporais de SMIL) ou explícita (como nos elos em NCL). Neste trabalho, os templates de leiaute são usados como entidades de primeira categoria para especificarem de forma alternativa a semântica espaço-temporal de aplicações multimídia. Na presente proposta, templates são sincronizados entre si utilizando como base as relações temporais definidas pelos operadores de Allen, o que é concretizado e formalizado pelo perfil XML TemplateSync, contribuição deste trabalho. Para demostrar a expressividade da abordagem, é apresentada uma representação gráfica para cada um dos treze operadores. Por último, de forma a ilustrar o emprego da proposta, um estudo de caso foi conduzido para apresentação de relacionamentos temporais entre templates utilizando a ferramenta de autoria multimídia Cacuriá.

Palavras-chaves: Leiautes. Template. Reúso. Sincronismo. Autoria Multimídia. Hipermídia.

ABSTRACT

Layout templates are reusable document models that provide spatial settings for sets of media. The increase of reuse in the specification of spatial synchronization media and code complexity reduction of multimedia applications are its two main features. However, layout templates are often not used as first class entities in the temporal specification of an application. Thereforth, multimedia documents usually define compositions that may have an implicit (e.g., SMIL timing containers) or explicit (e.g., links in NCL) semantics. In this paper, the layout templates are used as first class entities in order to specify the spatial and temporal semantics of multimedia applications. In this proposal, templates are synchronized with each other using as a basis the temporal relations defined by Allen operators, which is formalized by TemplateSync XML profile, a further contribution presented in this paper. Furthermore, it is presented a graphical representation for each of the thirteen operators in order to show the expressiveness of the approach. Finally, a case study was conducted to illustrate the applicability of our proposal and temporal relationships between templates using the multimedia authoring tool Cacuriá.

Keywords: Layouts. Template. Reuse. Synchronism. Multimedia Authoring. Hypermedia.

Lista de ilustrações

Lista de listagens

Lista de tabelas

Lista de abreviaturas e siglas

ADL	Architecture Description Languages
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
NCL	Nested Context Language
NCM	Nested Context Model
NEXT	NCL Editor supporting XTemplate
OA	Objeto de Aprendizagem
SMIL	Synchronized Multimedia Integration Language
TAL	Template Authoring Language
TRIC	Templated Recursive Image Composition
URI	Uniform Resource Identifier
XML	Extensible Markup Language
W3C	World-Wide Web Consortium
WYSIWYG	What You See Is What You Get

Sumário

1 Introdução

Modelos conceituais para autoria hipermídia costumeiramente oferecem como entidade de primeira classe as composições. No NCM (Nested Context Model) (SOARES; RODRIGUES; MUCHALUAT-SAADE, 2003), as composições são chamadas de contextos e são tão importantes que dão nome também à linguagem NCL (Nested Context Language). É por meio deles que se organiza a estrutura e a semântica de apresentação. Contextos aninham nós (de mídia ou de contexto) e elos que carregam a lógica temporal da aplicação. Em SMIL (Synchronized Multimedia Integration Language) (VUORIMAA; BULTERMAN; CESAR, 2008), as composições são a base da lógica de apresentação e são representadas pelos *containers* temporais, como o par e seq. Composições potencializam o reúso ao organizar partes de uma aplicação que podem ser referenciadas diversas vezes no documento.

Templates de composição (SOARES NETO; SOARES; SOUZA, 2010) oferecem um nível ainda maior de reúso, permitindo que se definam composições multimídia em aberto, com lacunas que podem ser preenchidas desde que determinadas restrições sejam satisfeitas. Dito de outra forma, templates de composição são como composições parametrizáveis onde não é necessário especificar todos os elementos que a compõem. Um *slideshow* de fotos, por exemplo, pode ser modelado como um template de composição, onde o que muda entre eles são apenas as mídias de foto utilizadas.

Para oferecer o reúso da especificação do leiaute de aplicações multimídia, geralmente se especifica tais informações de leiaute em separado. Isso se nota, por exemplo, na abordagem típica do par formado pelos documentos HTML (W3C, 2014) e CSS (W3C, 2015), onde o foco é tornar separada a especificação da forma em relação ao conteúdo.

Quando se especifica o leiaute em separado do conteúdo, um maior reúso pode ser obtido para a definição espacial. Essa abordagem também facilita o estabelecimento de uma mesma identidade visual em diversas aplicações multimídia, já que o leiaute em si é representado em separado e pode, assim, ser facilmente aplicado nas diferentes aplicações. Há diversos desafios para a especificação do leiaute que já foram sanadas para a especificação da estrutura de composições. Atuando exatamente para sanar tais desafios, o conceito de templates de leiaute (AMORIM; SANTOS; MUCHALUAT-SAADE, 2015) visa estabelecer modelos reusáveis de documentos que oferecem a descrição espacial para conjuntos de mídias. O desenvolvimento baseado em templates de leiaute é motivado por suas duas principais características: o aumento do reúso na especificação do sincronismo espacial de mídias e a redução da complexidade do

código de aplicações multimídia.

Nesta monografia é proposto o sincronismo de templates de leiaute em aplicações multimídia. A ideia central é permitir que não apenas se separe a estrutura de apresentação da aplicação do leiaute da mesma, mas também que se permita expressar a lógica temporal junto à especificação desse leiaute (e não apenas junto às composições). O trabalho garante a expressividade por meio da cobertura dos operadores de Allen (ALLEN, 1983) para a lógica temporal de templates de leiaute. A abordagem proposta é ilustrada na ferramenta de autoria multimídia Cacuriá (DAMASCENO; GALABO; SOARES NETO, 2014) com resultados positivos na aceleração e redução de complexidade na autoria.

1.1 Objetivos

O objetivo geral deste trabalho é realizar o sincronismo temporal de templates de leiaute em aplicações multimídia utilizando como base as relações temporais definidas pelos operadores de Allen. Os objetivos específicos são:

- Elaborar uma proposta abstrata para resolver o sincronismo temporal de templates de leiaute.
- Definir um perfil XML próprio, chamado de TemplateSync, para concretizar a proposta de sincronismo temporal cobrindo a expressividade dos operadores de Allen.
- Aplicar o perfil XML TemplateSync na ferramenta de autoria multimídia Cacuriá, a fim de ilustrar o emprego da proposta.

1.2 Organização do Trabalho

Este trabalho é organizado da seguinte maneira: o Capítulo 2 apresenta uma revisão bibliográfica e aprofunda mais a análise do uso de templates de leiaute. O Capítulo 3 cobre de que forma os treze operadores básicos de Allen (ALLEN, 1983) são tratados para relacionar templates de leiaute no tempo. O Capítulo 4 discute como a proposta é concretizada por um perfil XML chamado de TemplateSync. O capítulo 5 apresenta um estudo de caso em que TemplateSync é aplicado na ferramenta de autoria Cacuriá (DAMASCENO; GALABO; SOARES NETO, 2014). O Capítulo 6 conclui com um desfecho ao trabalho e uma discussão dos trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo apresenta os fundamentos teóricos utilizados no desenvolvimento deste trabalho, principalmente no tocante à autoria baseada em templates e em aspectos de sincronismo temporal.

Segundo (MUCHALUAT-SAADE; SOARES, 2002; MUCHALUAT-SAADE; SOARES, 2003), templates de composição especificam estruturas de documentos multimídia, definindo de forma genérica componentes e seus relacionamentos através de elos e conectores. A autoria por templates é um método utilizado para autoria hipermídia. Nesse método, são identificadas famílias de documentos e essas são instanciadas com objetivo de criar novos documentos (SOARES NETO, 2010).

Pode-se encontrar na literatura trabalhos que permitem a definição de templates de composição e o reúso de especificações estruturais. TAL (Template Authoring Language) (SOARES NETO; SOARES; SOUZA, 2010) é uma linguagem para autoria de templates de documentos hipermídia que permite a especificação de composições hipermídia incompletas independente da linguagem de autoria alvo. Ela tem como objetivo principal possibilitar que templates de documentos sejam especificados por autores mais especializados e sejam usados por outros autores para criarem novos documentos de forma mais simples. A base fundamental para o desenvolvimento de TAL foram os templates de composição propostos em versões anteriores da linguagem XTemplate (MUCHALUAT-SAADE; RODRIGUES; SOARES, 2002).

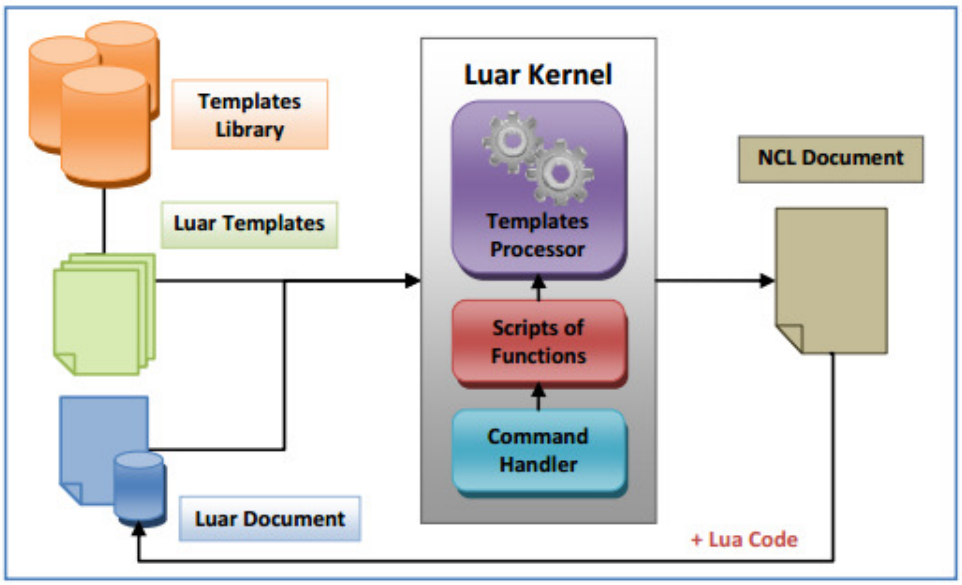
O modelo do XTemplate é baseado na separação entre conceitos de estilo e de configuração introduzidos em Linguagens de Descrição de Arquitetura (ADLs) (CLEMENTS, 1996). A versão do XTemplate 3.0 (SANTOS; MUCHALUAT-SAADE, 2009) foi desenvolvida para uma linguagem específica, NCL 3.0. Apesar do XTemplate ter como foco facilitar a autoria de documentos por especialistas, seus usuários necessitam de conhecimento específico sobre XPath e XSLT.

LimSee3 (DELTOUR; ROISIN, 2006) é uma ferramenta de autoria de documentos multimídia baseada em templates. Sua abordagem foca na estrutura lógica do documento, mantendo a semântica de tecnologias comprovadas, como SMIL. LimSee3 consiste em um modelo genérico para representar documentos multimídia e define uma sintaxe dedicada para representar templates para esses documentos. Entretanto, não é clara a separação entre o template e o documento multimídia, o que reduz a possibilidade de reúso.

Luar (BEZERRA et al., 2012) é uma linguagem de autoria para templates NCL. Tem como objetivo facilitar o desenvolvimento de aplicações interativas descritas em

NCL utilizando reúso. A linguagem Luar tem um processador de templates desenvolvido com a linguagem Lua. Assim, é possível utilizar NCL e Lua juntos para criar templates, generalizando e facilitando seu desenvolvimento. A Figura 1 mostra os elementos básicos e o fluxo do sistema de processamento de templates Luar.

Figura 1 – Elementos básicos e o fluxo do sistema de processamento de templates Luar.



Fonte: (BEZERRA et al., 2012)

Tanto o sincronismo espacial quanto o sincronismo temporal são gerados automaticamente pelo uso do template. Portanto, o usuário do template necessita apenas especificar as mídias a serem utilizadas pelo documento. Entretanto, segundo (AMORIM; SANTOS; MUCHALUAT-SAADE, 2013) linguagens para definição de templates de composição para documentos hipermídia como NCL, não oferecem modelos genéricos de leiautes espaciais. Ou seja, não permite definir conceitualmente um leiaute adaptativo que inclua a criação de elementos para especificar onde e como as mídias serão apresentadas.

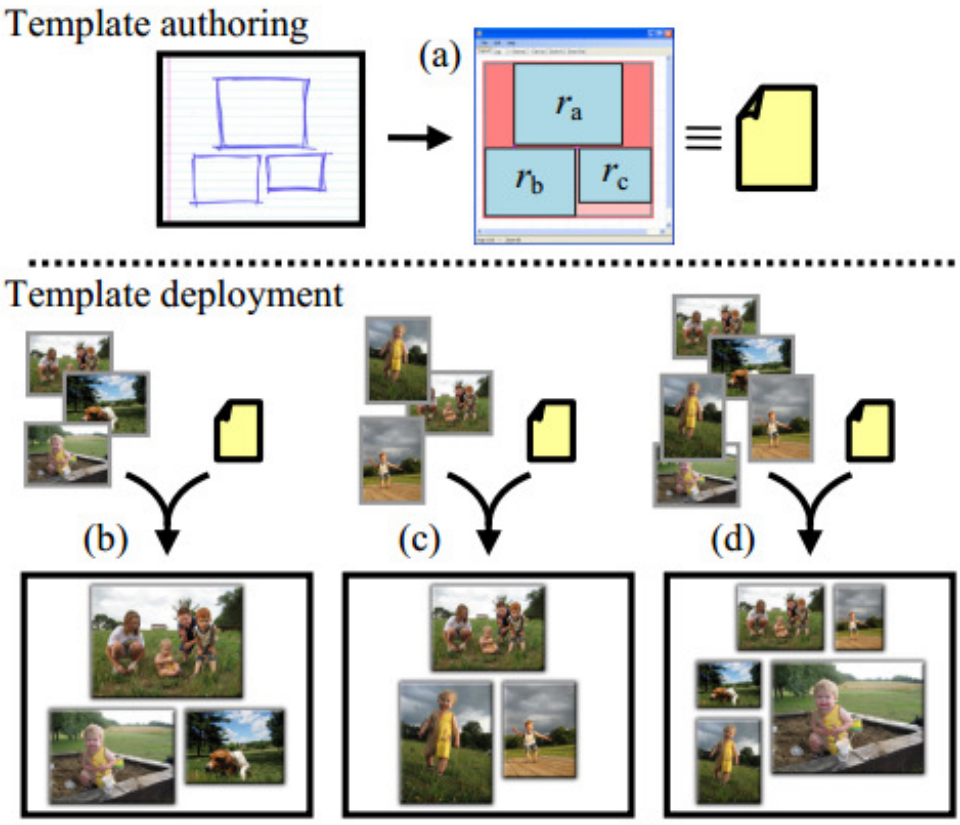
Em (AMORIM; SANTOS; MUCHALUAT-SAADE, 2015) é dito que um documento criado a partir da definição de vários componentes de leiaute é denominado template de leiaute. Por isso, existem na literatura inúmeras soluções para definição de leiautes espaciais na especificação de um template. Nesse caso, o usuário precisa especificar as mídias e o template, os organizando em canvas.

Em (ATKINS et al., 2010) é proposto TRIC (Templated Recursive Image Composition). TRIC organiza praticamente qualquer conjunto de imagens em um canvas contanto que o número de imagens seja pelo menos igual ao número de regiões do

template especificado. Restrições garantem que a imagem não será cortada inadvertidamente, respeitando suas proporções. TRIC tem dois estágios: Autoria do Template e o Processamento do Template, como mostra a Figura 2. Durante a autoria do template, um design original é codificado como uma especificação que captura o posicionamento e a proporção da região. Durante o processamento do template, a especificação organiza o conjunto de imagens de acordo com a quantidade e as proporções.

iSlideshow (CHEN; XIAO, 2010) é um sistema de *slideshow* de fotos inteligente que possui dois modos de navegação: *storytelling* e *person-highlighting*. O sistema analisa automaticamente informações sobre coleções de fotos e utiliza essas informações para gerar composições e transições. Assim como TRIC, o iSlideshow mantém a proporção das dimensões das mídias no canvas.

Figura 2 – Estágios do TRIC.



Fonte: (ATKINS et al., 2010)

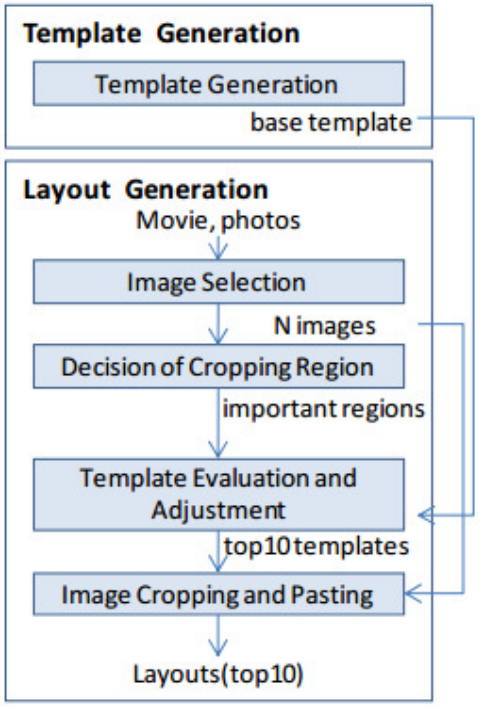
Em (MYODO et al., 2011) é apresentado um sistema automático de criação de leiautes para fotos de eventos, como ilustra a Figura 3. O leiaute gerado apresenta na tela as imagens cortadas e rearranjadas, permitindo que espectadores acompanhem os eventos em ordem sequencial. O sistema consegue obter um leiaute adaptativo gerando todos os templates adequados possíveis e escolhendo o que apresenta menos

cortes. Para evitar cortes sobre áreas importantes das imagens, o método apresentado determina automaticamente as regiões importantes, como faces. A arquitetura do sistema é mostrada na Figura 4.

Figura 3 – Leiaute gerado.

Fonte: (MYODO et al., 2011)

Figura 4 – Arquitetura do sistema.



Fonte: (MYODO et al., 2011)

Em (TIAN; ZHANG; TRETTER, 2011) é proposta uma abordagem para gerar automaticamente composições de fotos, utilizando uma coleção de imagens do usuário. O objetivo é garantir dimensionamentos adequados para as imagens de fundo e de primeiro plano, evitando a sobreposição de conteúdos importantes das fotos. Assim, é proposta uma medida de redundância da imagem para extrair a zona informativa. O método define como zona informativa a menor área retangular que o contém o conteúdo mais único da imagem. As imagens de primeiro plano são posicionadas nas áreas fora da zona informativa. Baseado nas informações sobre as zonas da imagem, um algoritmo é utilizado para posicionar as imagens em um canvas de dimensões arbitrárias, otimizando o dimensionamento do primeiro plano e do plano de fundo. Exemplos de composições geradas por esse método são mostrados na Figura 5.

Figura 5 – Resultado da composição automática de fotos.

Fonte: (TIAN; ZHANG; TRETTER, 2011)

Outros trabalhos (AMORIM; SANTOS; MUCHALUAT-SAADE, 2013; AMORIM; SANTOS; MUCHALUAT-SAADE, 2015) propõem um modelo que fornece ao autor do template a facilidade de criar modelos de leiautes genéricos que se adaptam automaticamente ao número de objetos de mídia em um dado documento.

Em (AMORIM; SANTOS; MUCHALUAT-SAADE, 2015) é apresentada uma linguagem para definição de características de apresentação genérica através de modelos denominados *Leiautes Adaptativos*. A especificação dos leiautes adaptativos proporciona a criação automática de regiões e descritores para documentos NCL. Com isso, na autoria de um documento NCL que utiliza template, faz-se necessário especificar apenas os objetos com conteúdo específico, instanciando os componentes genéricos do template. O leiaute de template é adaptativo quando possibilita a adaptação de sua definição conforme a quantidade de componentes da composição que utilizará o

template. Um exemplo de aplicação que usa leiautes adaptativos é mostrado na Figura 6 e o código utilizado para gerar tal aplicação é mostrado na Figura 7.

Figura 6 – Exemplo de aplicação que utiliza leiautes adaptativos.

Fonte: (AMORIM; SANTOS; MUCHALUAT-SAADE, 2015)

Figura 7 – Código da aplicação que utiliza leiautes adaptativos.

```
...
<layout id="F1" type="flowLayout" >
  <format align="center" hspace="10" vspace="10"
    top="20" left="10" width="780"
    height="440"/>
  <cell id="c1"  width="50" height="20"/>
  <cell id="c2"  width="650" height="420"/>
  <cell id="c3"  width="50" height="20"/>
  <focus focusIndex="1" moveDown="2" moveUp="2"/>
</layout>
<layout id="G2" type="gridLayout" >
  <format top="450" left="80" width="670"
    height="130" columns="5" rows="1"
    hspace="10" vspace="0" />
  <focus focusIndex="2" moveDown="1" moveUp="1"/>
</layout>

<region id="rgfundo" top="0" left="0" width="100%"
  height="100%">
...
```

Fonte: (AMORIM; SANTOS; MUCHALUAT-SAADE, 2015)

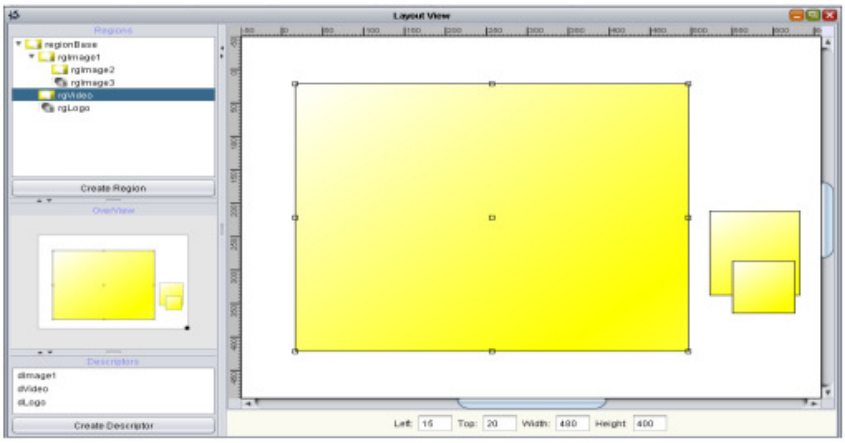
Em (AMORIM; SANTOS; MUCHALUAT-SAADE, 2013) é apresentada uma extensão da linguagem XTemplate 3.0, chamada XTemplate 4.0. A linguagem de autoria XTemplate 4.0 é utilizada para especificar templates de composições hipermídia, podendo incluir leiautes adaptativos. Além da facilidade de permitir a definição de leiautes

adaptativos, XTemplate 4.0 permite a definição de templates aninhados. Esses, possibilitam a associação de componentes de templates a outros templates. XTemplate 4.0 satisfaz a propriedade de composicionalidade para a autoria dos templates, especificando pontos de interface para comunicação entre templates internos e externos.

Para facilitar a criação e a utilização de templates por usuários com pouco ou nenhum conhecimento em linguagem de autoria multimídia, foram desenvolvidas ferramentas usando abordagem gráfico visual (CELENTANO; GAGGI, 2006; SILVA; MUCHALUAT-SAADE, 2012; DAMASCENO; SANTOS; MUCHALUAT-SAADE, 2010; DELTOUR; LAYAIDA; WECK, 2014).

EDITEC (DAMASCENO; SANTOS; MUCHALUAT-SAADE, 2010) é editor gráfico de templates de composição baseado na linguagem XTemplate 3.0 e tem como objetivo facilitar a criação de documentos NCL. Ele permite a criação de diversos templates com as mais variadas semânticas espaço-temporais. A Figura 8 mostra a visão de leiaute do EDITEC.

Figura 8 – Visão de leiaute do EDITEC.

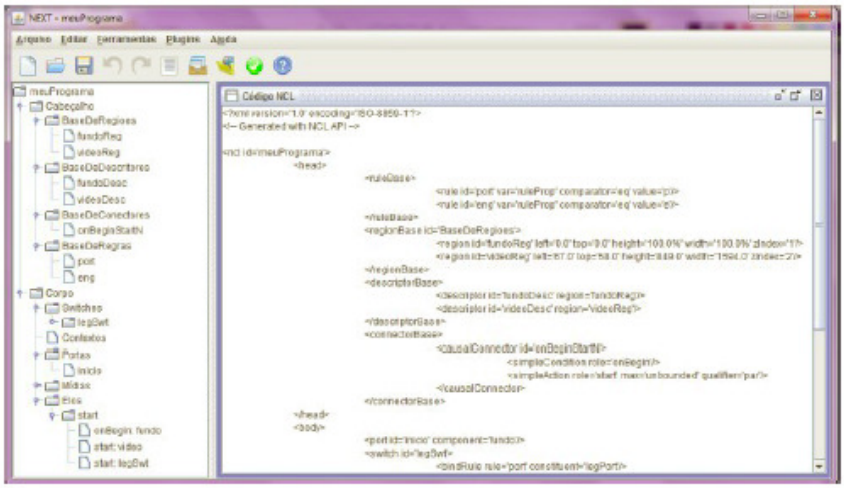


Fonte: (DAMASCENO; SANTOS; MUCHALUAT-SAADE, 2010)

NEXT (NCL Editor supporting XTemplate) (SILVA; MUCHALUAT-SAADE, 2012) é uma ferramenta de autoria gráfica que tem como objetivo facilitar a criação de aplicações para TV digital, utilizando a linguagem NCL. Ela permite o uso de templates de composição, criados com a linguagem XTemplate 3.0. NEXT facilita o trabalho de autores com pouco ou nenhum conhecimento sobre NCL. Na Figura 9, é mostrada a interface gráfica do NEXT.

Entretanto, essas ferramentas que utilizam abordagem gráfico visual não permitem reusar uma mesma mídia em diferentes templates de leiaute. Assim, se o usuário quiser que diferentes templates de leiaute utilizem a mesma mídia, ele deverá instanciar

Figura 9 – Interface gráfica do NEXT.



Fonte: (SILVA; MUCHALUAT-SAADE, 2012)

a mídia para cada template, além de definir os elos de sincronismo que descrevem o comportamento da mídia em cada template durante sua exibição.

Este trabalho visa aumentar o reúso de código, fazendo com que diferentes templates de leiaute compartilhem a mesma mídia, relacionando apenas as regiões de cada template que a mídia será exibida e quando o template deverá ser executado. Ademais, o presente trabalho visa permitir que se sincronize temporalmente os templates de leiaute como entidades de primeira classe, um recurso que não é observado em nenhum dos trabalhos previamente descritos.

3 Sincronismo Espaço-Temporal

Neste capítulo é descrito como se resolve a semântica espaço-temporal na presente proposta. O conceito central é permitir que se expresse não apenas os templates de leiaute como elementos de primeira classe (fornecendo as informações de sincronismo espacial propriamente ditas), mas também que se possa definir relacionamentos de sincronismo temporal entre tais templates. Uma implicação direta da presente proposta é a possibilidade de simplificar a especificação do sincronismo espaço-temporal com considerável expressividade e reúso.

Para garantir tal expressividade, os templates podem ser sincronizados no tempo e entre si utilizando como base as relações temporais definidas por Allen (ALLEN, 1983). Allen especificou treze operadores básicos que representam todos os relacionamentos de sincronismo possíveis entre dois intervalos temporais. Os operadores são: *before*, *after*, *equals*, *meet*, *met by*, *overlaps*, *overlapped by*, *during*, *contains*, *starts*, *started by*, *finishes* e *finished by*, mostrados na Figura 10.

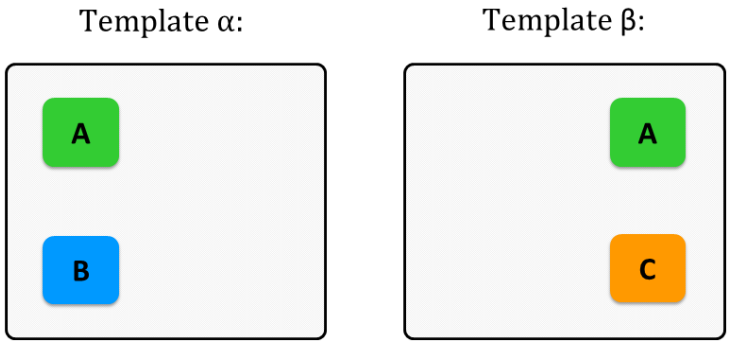
Figura 10 – Operadores de Allen.

Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Será usado um mesmo cenário para ilustrar a semântica temporal proposta fazendo uso de cada um dos 13 operadores de Allen. No cenário, considera-se o relacionamento entre dois templates de leiaute, o template α e o template β. O template α, como pode ser observado na Figura 11 é simples e define duas regiões demarcadas: a região *A* e a região *B*. O template β, representado também na Figura 11, é composto por duas regiões: a região *A* e a região *C*. Depois de especificadas tais regiões, as

mesmas podem ser reusadas ao definir as mídias. Ao invés de uma mídia expressar todos os parâmetros para sua posição espacial, basta apontar para uma das três regiões *A*, *B* ou *C*. A região de mesmo identificador *A* está expressa em ambos os templates de leiaute, em posições diferentes. Quando se sincronizar no tempo α e β, deve haver uma regra não-ambígua para determinar em qual das duas posições a mídia deve aparecer, conforme descrito a seguir. Esse recurso permite que o posicionamento de uma mídia seja modificado no decorrer de uma aplicação através do sincronismo de templates de leiaute e não por meio da mudança de propriedades daquela mídia.

Figura 11 – Template α e β com as regiões *A*, *B* e *C*.



- Relações "α *before* β" e "β *after* α"

  Nas relações "α *before* β" (α < β) e "β *after* α" (β > α), o template β é inicializado após o término do template α ($\alpha_f < \beta_i$). Dessa forma, durante a execução de α, as regiões *A* e *B* são exibidas como foram definidas no templates α. Da mesma maneira, durante a execução de β, as regiões *A* e *C* são exibidas como foram definidas no templates β, o que pode ser visto na Figura 12. No intervalo entre o fim do template α e o início do template β nenhuma região é visível.

- Relação "α *equal* β"

  Na relação "α *equal* β" (α = β), os templates α e β iniciam e terminam juntos ($\alpha_i = \beta_i$ e $\alpha_f = \beta_f$). Nesse caso, ambos os templates definem a região *A*, mas com posicionamento espacial diferente. Como o início e término do template α é determinado em função do template β, prevalece o posicionamento da região *A* definida no template β. As demais regiões, *B* e *C*, são exibidas de acordo como foram definidas nos templates α e β, como mostra a Figura 13.

- Relações "α *meets* β" e "β *met by* α"

  Nas relações "α *meets* β" (α *m* β) e "β *met by* α" (β *mi* α), o início da execução do template β é igual ao término de execução do template α ($\alpha_f = \beta_i$). Dessa forma, as regiões *A* e *B* são exibidas como foram definidas no template α, que

Figura 12 – Relações "α *before* β" e "β *after* α".

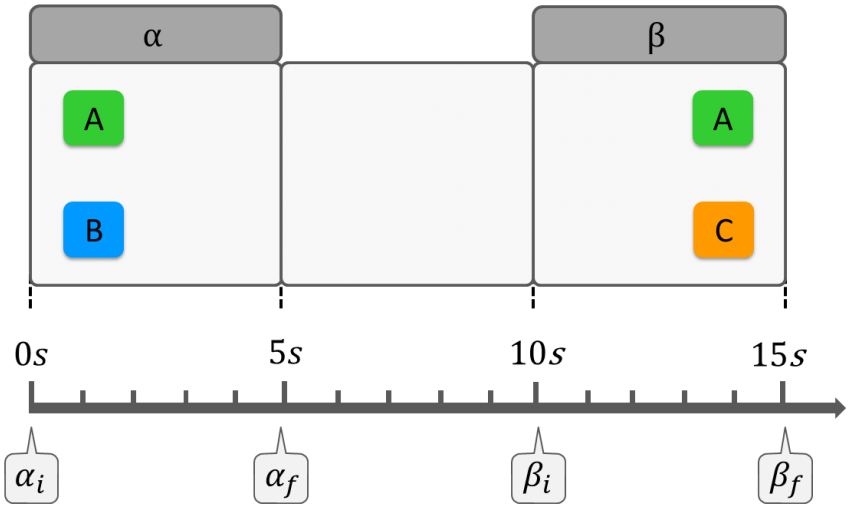
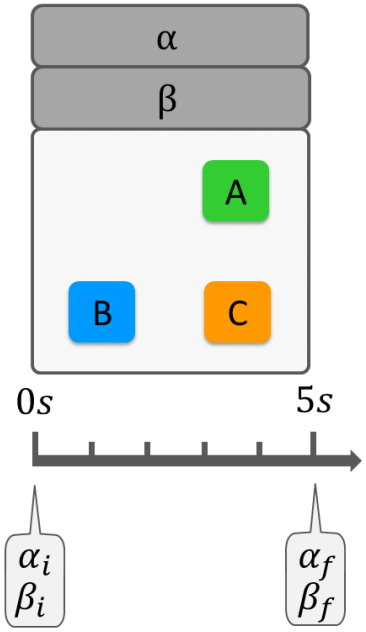


Figura 13 – Relação "α *equal* β".

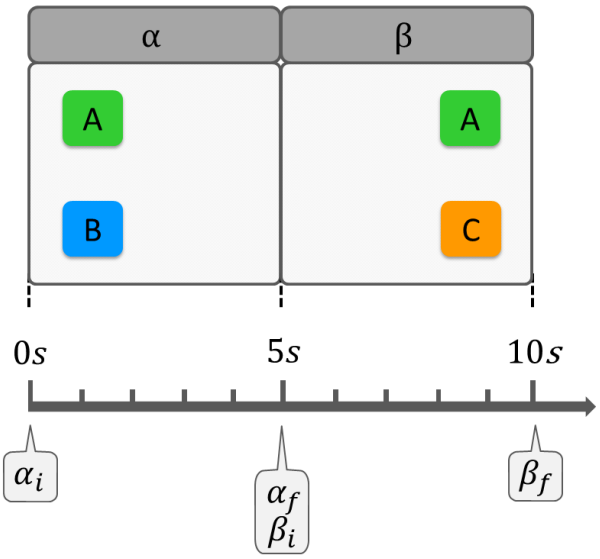


é finalizado ao início da execução do template β. Com o template β iniciado, tornam-se visíveis as regiões que o compõem, o que mostra a Figura 14.

- Relações "α *overlaps* β" e "β *overlapped by* α"

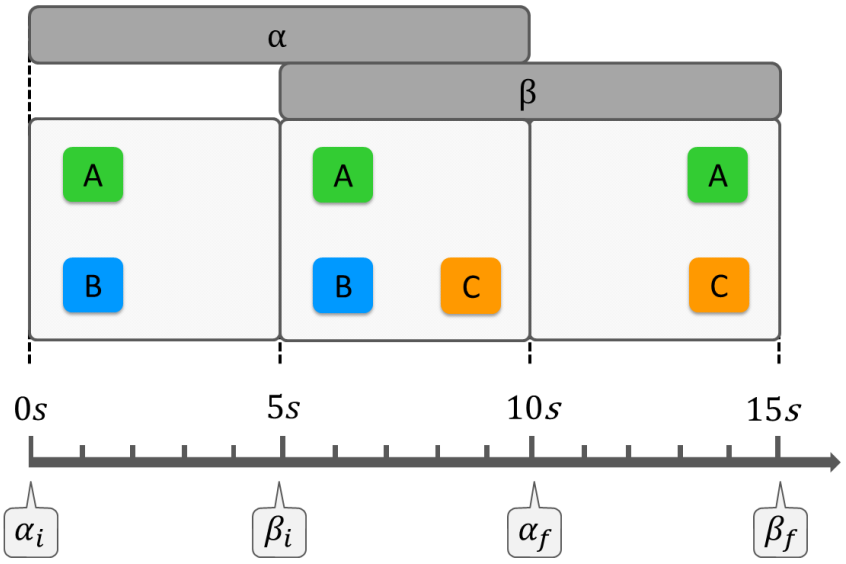
  Nas relações "α *overlaps* β" (α *o* β) e "β *overlapped by* α" (β *oi* α), o template α sobrepõe o template β, onde β inicia antes do término de α e finaliza depois do término de α ($\alpha_i < \beta_i$ e $\alpha_f > \beta_i$ e $\alpha_f < \beta_f$). Dessa forma, no intervalo de tempo entre o início de α e o início de β são visíveis apenas as regiões que compõem o template α. Já no início do template β, como o template α o sobrepõe, prevalece o

Figura 14 – Relações "α *meets* β" e "β *met by* α".



posicionamento definido no template α para a região *A*. Entre o início do template β e o fim do template α, as regiões *B* e *C* são posicionadas como definidas nos respectivos templates (Figura 15). Após o fim do template α, a região *A* volta a ser posicionada como é definida no template β, sendo exibida junto com a outra região que compõe o template β.

Figura 15 – Relações "α *overlaps* β" e "β *overlapped by* α".

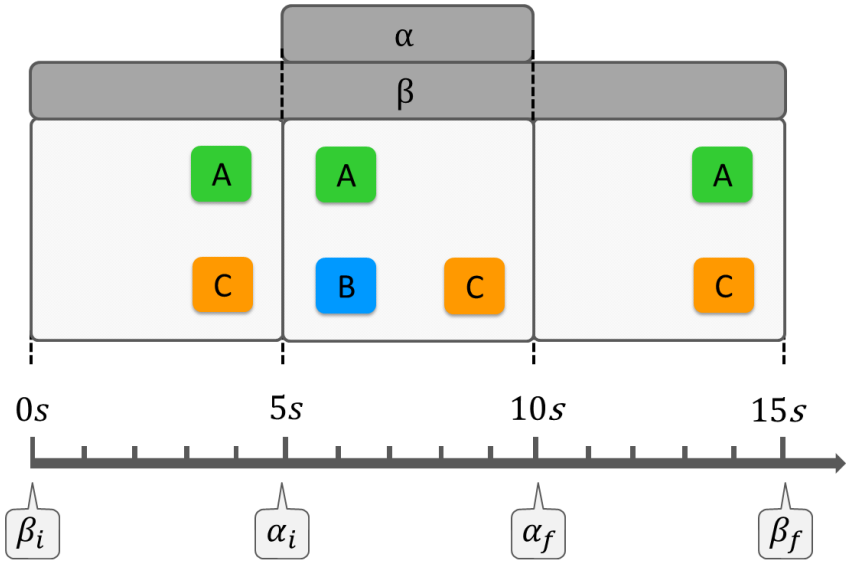


- Relações "α *during* β" e "β *contains* α"

  Nas relações "α *during* β" (α *d* β) e "β *contains* α" (β *di* α), o template α é exibido durante a execução do template β, ou seja, α inicia após o início de β e termina

antes do fim de β ($\alpha_i > \beta_i$ e $\alpha_f < \beta_f$). Dessa forma, nos intervalos entre o início do template β e o início do template α e o fim do template α e o fim do template β, são exibidas as regiões que compõem o template β. E, como o template α é exibido durante o template β, no intervalo em que os templates α e β são visíveis simultaneamente, a região *A* é disposta conforme definido em α, como mostra a Figura 16.

Figura 16 – Relações "α *during* β" e "β *contains* α".



- Relações "α *starts* β" e "β *started by* α"

  Nas relações "α *starts* β" (α *s* β) e "β *started by* α" (β *si* α), o template α inicia o template β. Portanto, o início de β é igual ao início de α ($\alpha_i = \beta_i$). Dessa forma, no intervalo de tempo em que os dois templates são exibidos simultaneamente, a região *A*, que está contida em ambos, é disposta conforme definido no template β, como mostra a Figura 17.

- Relações "α *finishes* β" e "β *finished by* α"

  Nas relações "α *finishes* β" (α *f* β) e "β *finished by* α" (β *fi* α), o template α finaliza o template β. Portanto, o instante de término de β é igual ao instante de término de α ($\alpha_f = \beta_f$). Dessa forma, no intervalo de tempo em que os dois templates são exibidos simultaneamente, a região *A*, que está contida em ambos, é disposta conforme foi definido no template β, como mostra a Figura 18.

Figura 17 – Relações "α *starts* β" e "β *started by* α".

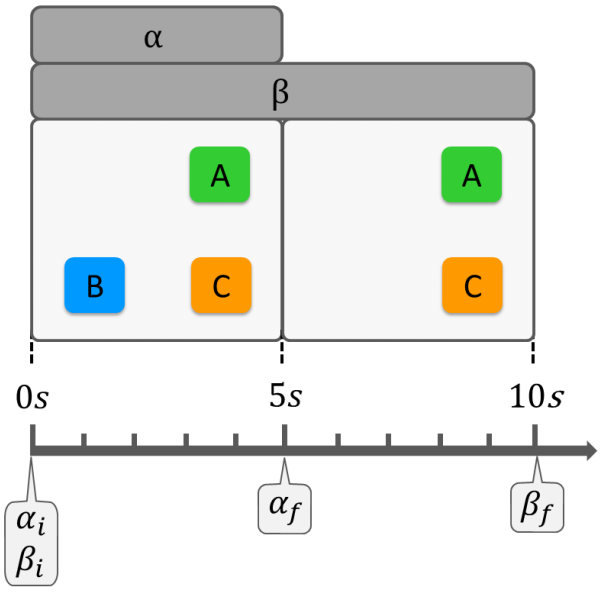
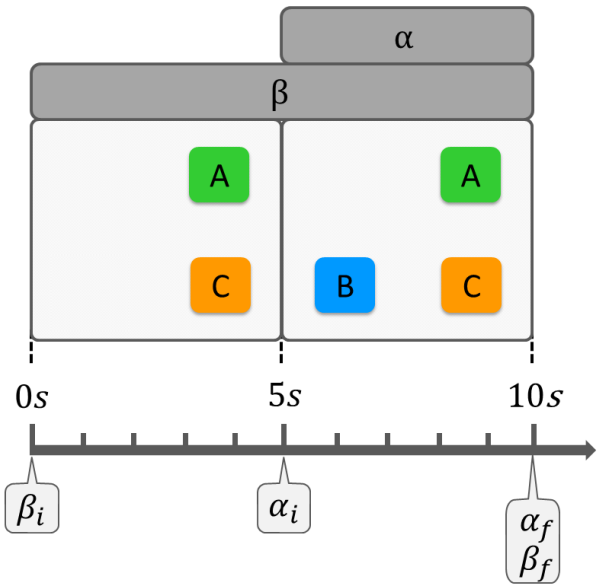


Figura 18 – Relações "α *finishes* β" e "β *finished by* α".



4 Perfil XML Template Sync

A proposta de sincronismo temporal cobrindo a expressividade dos operadores de Allen, conforme descrito no capítulo anterior, precisa ser concretizada e formalizada. Neste capítulo é apresentado o perfil XML TemplateSync, que foi especificado utilizando a abordagem modular do XML Schema, seguindo o padrão definido em (W3C et al., 2003). O perfil TemplateSync pode ser reusado em aplicações XML para tratar especificamente do aspecto de sincronismo de templates. A Tabela 1 lista os elementos e atributos do perfil.

Tabela 1 – Elementos do Perfil TemplateSync

Elemento	Atributos	Elementos Filhos
templatebase	id	(template)$*$
template	id, src	sync$*$
sync	id, time	(start, stop)$*$
start	id, target, delay	-
stop	id, target, delay	-

A estrutura básica do perfil TemplateSync é formada pelo elemento <templatebase>. O elemento <templatebase> é a raíz do perfil e consiste de uma composição de objetos de template (<template>). O elemento <templatebase> possui o atributo de identificação id, que pode ser definido por qualquer cadeia de caracteres, desde que seu valor seja único no documento.

Objetos de template são representados pelo elemento <template>. Além do identificador, também possuem o atributo *src*, que especifica a *URI* do conteúdo do template. Dessa forma, a especificação do posicionamento espacial é expressa em outro documento externo, o que é bem-vindo, já que assim TemplateSync não trata de parâmetros específicos de posicionamento (como *left*, *top*, *zIndex*, etc) e sim os herda ou reaproveita. Objetos de template também possuem uma lista de objetos sincronismo. Em TemplateSync, cada objeto de template possui um cronômetro de execução interno que é usado como referencial temporal pelos objetos de sincronismo (elementos <sync>) filhos.

O elemento <sync> define relações de sincronismo temporal entre templates. Além do identificador, o elemento <sync> possui o atributo *time*, uma âncora temporal que utiliza o cronômetro de execução do elemento pai como referencial para definir o instante em que as ações são disparadas. O elemento <sync> contém uma lista de objetos de ações.

Em TemplateSync existem dois tipos de ações, representadas pelos elementos <start> e <stop>. O elemento <start> inicia a apresentação de um objeto de template. E o elemento <stop>, termina a apresentação de um objeto de template. Além do identificador, todas as ações também possuem o atributo *target*, que define o objeto de template alvo que sofre a ação, e o atributo *delay*, que define o tempo de atraso da execução da ação. Para ilustrar o emprego e expressividade de TemplateSync, as representações de documento para cada operador temporal de Allen são descritas a seguir.

Para representar as relações "α *before* β" e "β *after* α" faz-se necessário especificar uma composição onde o template β é inicializado após o término do template α. Para isso, como ilustra a Listagem 1, define-se um objeto de sincronismo (<sync>), que dispara aos dez segundos. Este objeto de sincronismo possui um objeto de ação do tipo <stop>, que termina a apresentação do próprio template α, e um objeto de ação do tipo <start>, que inicia a apresentação do template β. Dessa forma, o template β é iniciado após cinco segundos do término do template α.

Listagem 1 – Relações "α *before* β" e "β *after* α".

```
1 <template id="template_a" src="template1.temp" >
2    <sync time="10s">
3       <stop target="template_a" />
4       <start target="template_b" delay="5s" />
5    </sync>
6 </template>
```

Para expressar as relações "α *meets* β "e "β *met by* α "é definido um objeto de sincronismo (<sync>) que dispara aos quinze segundos, como ilustra a Listagem 2. Esse objeto de sincronismo possui um objeto de ação do tipo <stop>, que termina a apresentação do próprio template α, e um objeto de ação do tipo <start>, que inicia a apresentação do template β. Assim, o template β é iniciado ao término do template α.

Listagem 2 – Relações "α *meets* β" e "β *met by* α".

```
1 <template id="template_a" src="template1.temp" >
2    <sync time="15s">
3       <stop target="template_a" />
4       <start target="template_b" />
5    </sync>
6 </template>
```

A relação "α *equal* β" ocorre quando os templates α e β iniciam e terminam no mesmo instante. Para expressar essa relação, como ilustra a Listagem 3, define-se o

primeiro objeto de sincronismo dentro do tamplate α. Dentro do objeto de sincronismo, é declarado um objeto de ação do tipo <start> que inicia a execução do template β. Dessa forma, ambos os templates iniciam a apresentação no mesmo instante. Para especificar o término dos templates, no mesmo instante, define-se um segundo objeto de sincronismo dentro do template α que dispara aos dez segundos, cujo objeto de ação do tipo <stop> finaliza a apresentação de ambos os templates.

Listagem 3 – Relação "α *equal* β"

```
1 <template id="template_a" src="template1.temp">
2    <sync>
3       <start target="template_b" />
4    </sync>
5    <sync time="10s">
6       <stop target="template_a" />
7       <stop target="template_b" />
8    </sync>
9 </template>
```

Para representar as relações "α *starts* β" e "β *started by* α", que ocorrem quando o template α inicia o template β, é utilizado um objeto de sincronismo dentro do template α, como mostra a ilustra a Listagem 4. Dentro do objeto de sincronismo, é declarado um objeto de ação do tipo <start> que inicia a execução do template β. Assim, o template β é iniciado no mesmo instante que o template α.

Listagem 4 – Relações "α *starts* β" e "β *started by* α"

```
1 <template id="template_a" src="template1.temp">
2    <sync>
3       <start target="template_b" />
4    </sync>
5 </template>
```

As relações "α *finishes* β" e "β *finished by* α" ocorrem quando o template α termina o template β. Essas relações são representadas utilizando um objeto de sincronismo, conforme ilustrado na Listagem 5, que dispara aos dez segundos. Esse objeto de sincronismo possui dois objeto de ação do tipo <stop> que finalizam as apresentações de ambos os templates.

Listagem 5 – Relações "α *finishes* β" e "β *finished by* α"

```
1 <template id="template_a" src="template1.temp">
2    <sync time="10s">
3       <stop target="template_a" />
```

```
4       <stop target="template_b" />
5    </sync>
6 </template>
```

Relações como "α *overlaps* β" e "β *overlapped by* α" são expressas conforme ilustrado na Listagem 6. Nesse caso, o template β inicia antes do término do template α e finaliza depois do término do template α. Para isso, define-se o primeiro objeto de sincronismo dentro do template α, que dispara aos cinco segundos da sua execução. Dentro do primeiro objeto de sincronismo é declarado um objeto de ação do tipo <start> que inicia a apresentação do template β. Em seguida, é declarado um segundo objeto de sincronismo dentro do template α, que dispara aos dez segundos. E, dentro do segundo objeto de sincronismo é declarado um objeto de ação do tipo <stop> que finaliza a apresentação do template α.

Listagem 6 – Relações "α *overlaps* β" e "β *overlapped by* α".

```
1 <template id="template_a" src="template1.temp">
2    <sync time="5s">
3       <start target="template_b" />
4    </sync>
5    <sync time="10s">
6       <stop target="template_a" />
7    </sync>
8 </template>
```

As relações "α *during* β" e "β *contains* α", ocorrem quando o template α é exibido durante a execução do template β, ou seja, o template α é iniciado após o início do template β e termina antes do fim do template β. Para representar essa relação em TemplateSync, como ilustra a Listagem 7, define-se o primeiro objeto de sincronismo dentro do template β, que dispara aos cinco segundos da sua execução. Dentro do primeiro objeto de sincronismo é declarado um objeto do elemento <start> que inicia a apresentação do template α. Em seguida, é declarado um segundo objeto de sincronismo dentro do template β, que dispara aos dez segundos. Dentro do segundo objeto de sincronismo é declarado um objeto do elemento <stop> que finaliza a apresentação do template α.

Listagem 7 – Relações "α *during* β" e "β *contains* α".

```
1 <template id="template_b" src="template2.temp">
2    <sync time="5s">
3       <start target="template_a" />
4    </sync>
5    <sync time="10s">
```

```
6       <stop target="template_a" />
7     </sync>
8 </template>
```

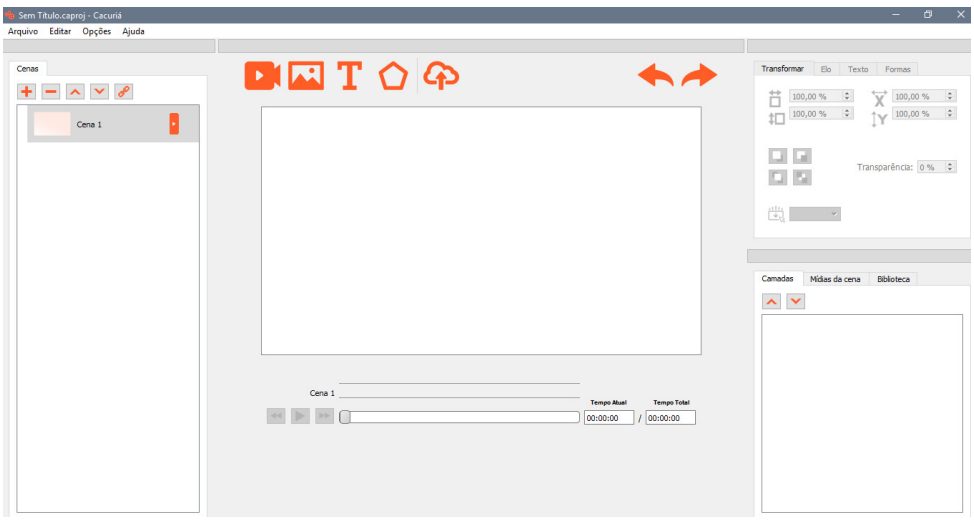
5 Estudo de Caso

Para ilustrar a aplicação da proposta, o perfil TemplateSync foi utilizado na ferramenta de autoria Cacuriá (DAMASCENO; GALABO; SOARES NETO, 2014) para permitir a sincronização dos templates de leiautes que estão disponíveis na ferramenta.

Cacuriá é uma ferramenta que permite a criação e compartilhamento de vídeos enriquecidos com conteúdo multimídia interativo. Ela é desenvolvida em linguagem de programação C++, com a utilização do *framework* Qt (BLANCHETTE; SUMMERFIELD, 2008). Esse *framework* permite o desenvolvimento de aplicativos multiplataforma utilizando a abordagem “escreva uma vez, compile em qualquer lugar”. A abordagem utilizada na ferramenta é WYSIWYG (What You See Is What You Get), em que o conteúdo visto e que está sendo modificado é idêntico à aplicação final gerada pela ferramenta (DAMASCENO, 2015).

A ferramenta Cacuriá utiliza uma abstração chamada de Cena. Cada cena pode ser composta por um vídeo, outros tipos de mídia (imagem, texto e formas) e templates de leiaute predefinidos. O Cacuriá também oferece ao usuário a adição de elos, que possuem o efeito iniciar uma cena em um determinado instante de tempo. O elo pode pode ser acionado quando a cena chega em um certo instante de tempo ou pela interação do usuário através do clique do mouse sob uma mídia ou o apertar de um botão do teclado. A Figura 19 mostra a interface da ferramenta Cacuriá.

Figura 19 – Interface do Cacuriá.



Na ferramenta Cacuriá, os templates de leiaute contêm regiões. As regiões definem a posição e o tamanho da área em que as mídias serão ser apresentadas

no template. Quando um template é inserido na cena, as regiões que o compõem são mostradas na Visão de Leiaute. Os templates de leiaute possuem regiões com diversas cores e essas identificam as regiões. Assim, se em uma cena são inseridos dois templates e ambos possuem uma região de mesma cor, significa que ambos compartilham essa região. As regiões só são exibidas quando a cena está pausada, ou seja, quando é possível editá-la. Dessa forma, ao utilizar um template, o autor da aplicação só precisa utilizar as regiões que o convém. No centro de cada região são dispostos quatro ícones que permitem a adição de mídias (vídeo, imagem, texto e formas). Ao clicar em um dos ícones de uma determinada região, a mídia escolhida passa a ocupar o lugar da mesma, ou seja, assume suas dimensões e posicionamento.

Nas seções seguintes são descritas as construções de um Objeto de Aprendizagem (OA) com a temática de Algoritmos de Ordenação e um programa de telejornal utilizando os templates de leiaute disponíveis na ferramenta Cacuriá.

5.1 Objeto de Aprendizagem sobre Algoritmos

Como mostra a Figura 20, o OA consiste em um vídeo interativo de um professor apresentando uma introdução à algoritmos de ordenação. Em um determinado momento, o professor questiona o aluno sobre qual algoritmo, insertsort ou quicksort, ele quer assistir a explicação. O aluno pode, então, clicar na figura que representa o algoritmo escolhido. Como consequência, o vídeo corrente é redirecionado para o vídeo com a explicação referente ao algoritmo escolhido pelo aluno.

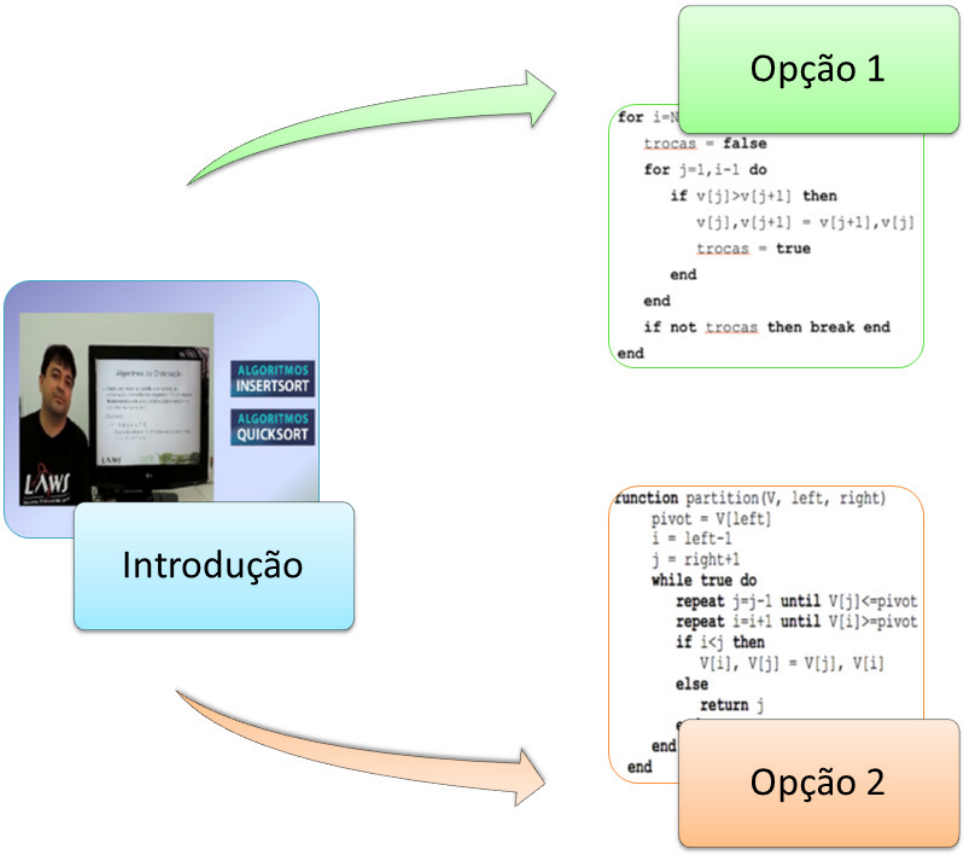
Esse OA pode ser construído utilizando templates de leiaute, fornecidos pela ferramenta Cacuriá, o que oferece mais rapidez e praticidade. O seu desenvolvimento é feito com a elaboração três cenas: a primeira é composta pelo vídeo introdutório e pelas imagens com as opções dos algoritmos; a segunda é composta pelo vídeo sobre o algoritmo insertsort e mídias que ajudarão o professor na explicação; e a terceira contém o vídeo sobre quicksort e algumas mídias de apoio. A seguir, é detalhado o desenvolvimento de cada cena.

- Construção da Cena 1

  É desejável que a primeira cena termine com o fim do vídeo de introdução, chamado “Algoritmos.mp4”. Assim, inicialmente, é definido o tempo total da Cena 1 como 1 minuto e 2 segundos, tempo de duração do vídeo.

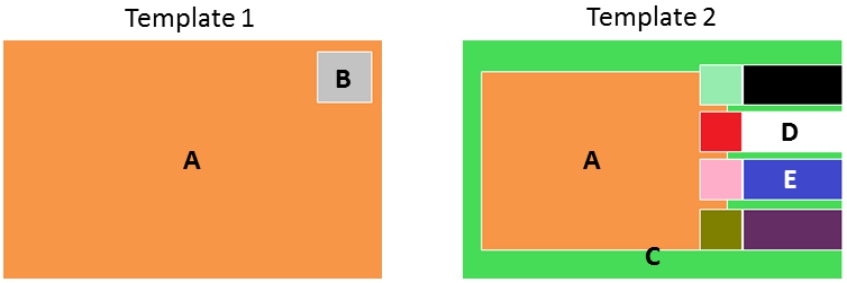
  Na Cena 1 são utilizados dois templates de leiaute, ilustrados na Figura 21. O template 1 é usado nos instantes em que vídeo de introdução preenche toda tela. O template 2 é utilizado quando o professor apresenta as opções ao aluno. Nesse

Figura 20 – OA de Algoritmos de Ordenação.



momento, o vídeo é redimensionado e são exibidas a imagem de fundo e as duas imagens que servem de elo para os outros vídeos explicativos.

Figura 21 – Templates utilizados na Cena 1.

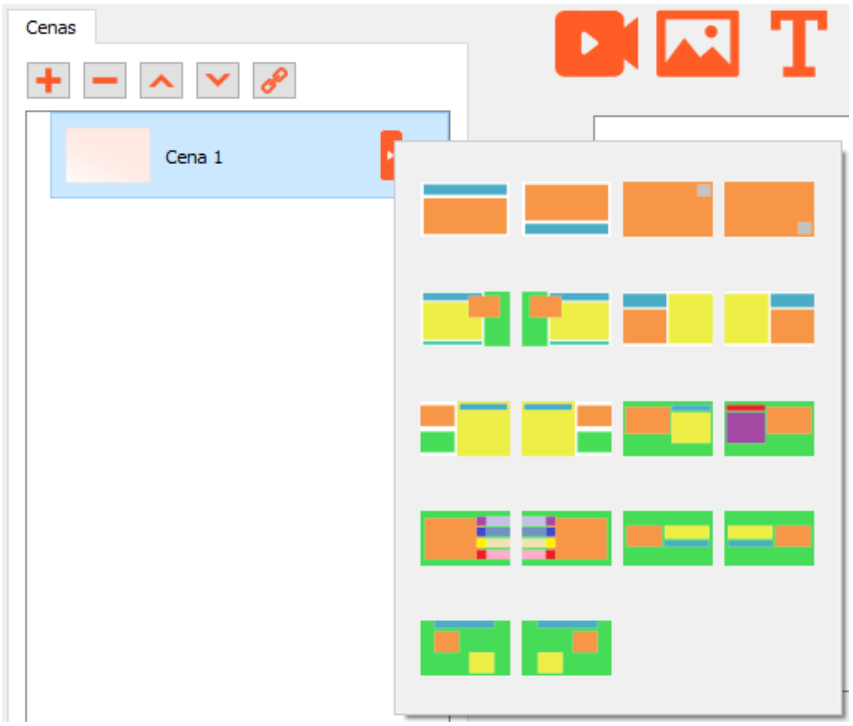


O template 1 é composto por duas regiões, uma região de cor laranja, e uma região de cor cinza, que são identificadas respectivamente pelas letras *A* e *B*. O template 2 define dez regiões: entre elas, a região *C* de cor verde, a região *D* de cor branca, a região *E* de cor azul e, assim como o template 1, a região *A* de cor laranja. As demais regiões especificadas no template 2 não são utilizadas no desenvolvimento desse OA. Dessa forma, o vídeo principal apontará para a

região *A* de todos os templates, uma imagem com a marca do laboratório de pesquisa apontará para a região *B*, a região *C* será ocupada por uma imagem de plano de fundo e as imagens com as opções insertsort e quicksort apontarão para as regiões *D* e *E*, respectivamente.

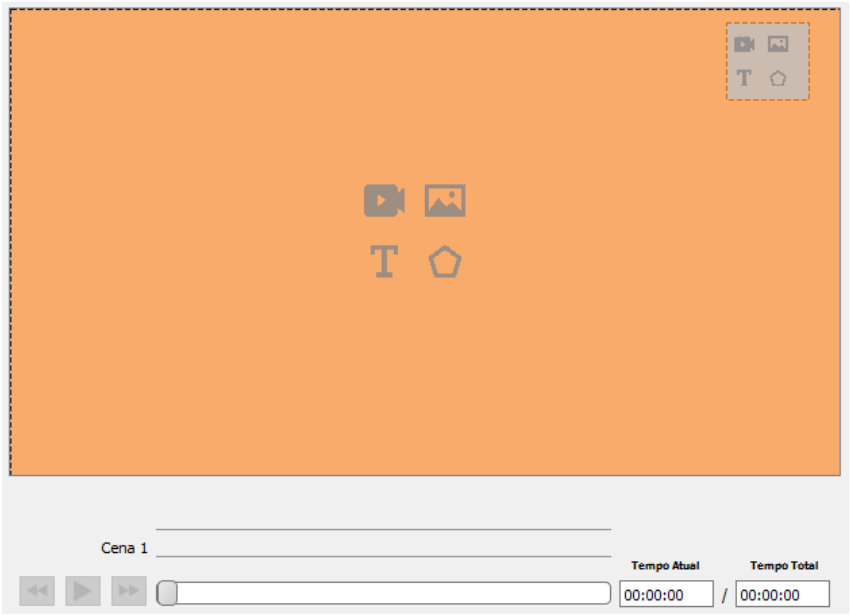
O primeiro passo para a construção da Cena 1, é inserir o template 1. Isso é feito pressionando o botão “Inserir Template” situado na Visão de Cenas. Como consequência, é aberto um menu com templates disponíveis na ferramenta onde o usuário pode escolher o template de leiaute que deseja utilizar, conforme mostra a Figura 22. Pode-se notar que na mesma figura estão presentes as opções correspondentes a todos templates disponíveis. Então, como mostra a Figura 23, a opção correspondente ao template 1 é escolhida, resultando na apresentação do mesmo na Visão de Leiaute.

Figura 22 – Menu de templates de leiaute.



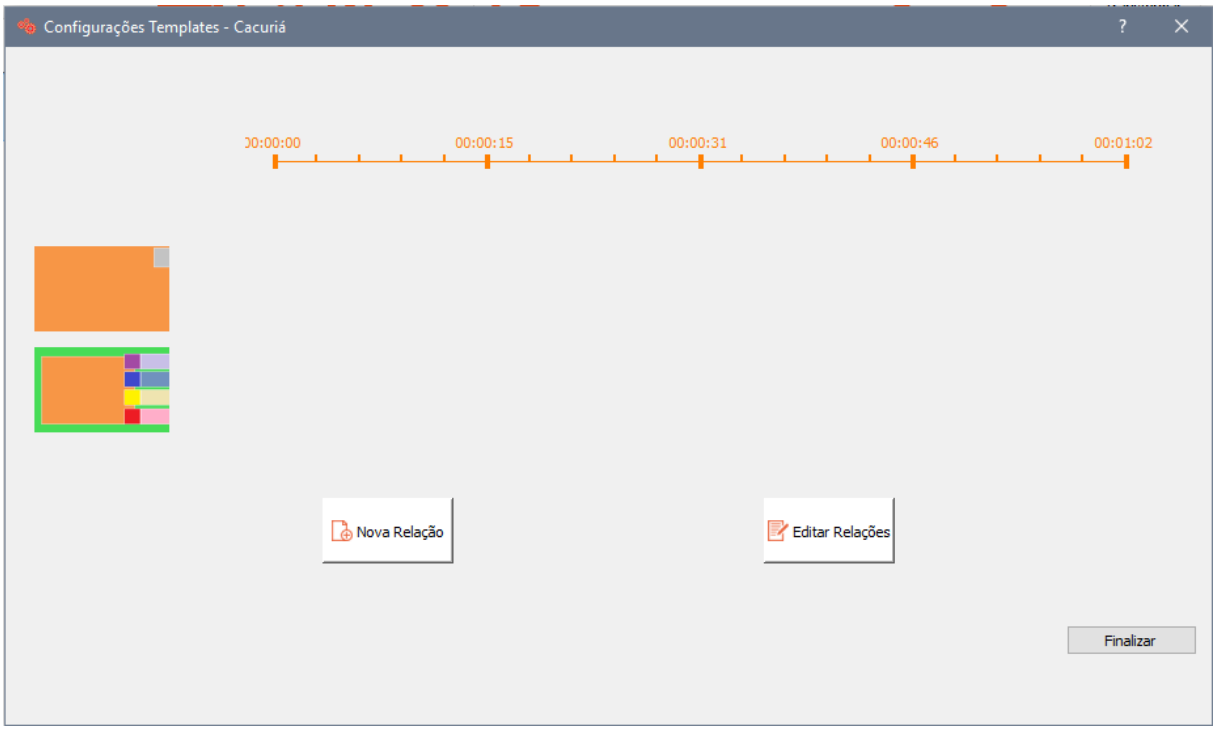
O próximo passo é inserir o template 2. Para isso, repete-se o processo anterior. Porém, como a cena já contém um template, é aberta automaticamente uma janela onde o usuário deverá configurar como os templates irão se relacionar. Como mostra a Figura 24, a janela é composta pelos ícones dos templates inseridos atualmente na cena e de três botões: “Nova Relação”, “Editar Relações” e “Finalizar”. Ao lado dos ícones dos templates são demonstrados os intervalos de tempo em que cada template é visível na cena. Como, ainda não foi determinado os tempos de início e fim dos templates, esse espaço não está preenchido. Para

Figura 23 – Template 1 na Visão de Leiaute.



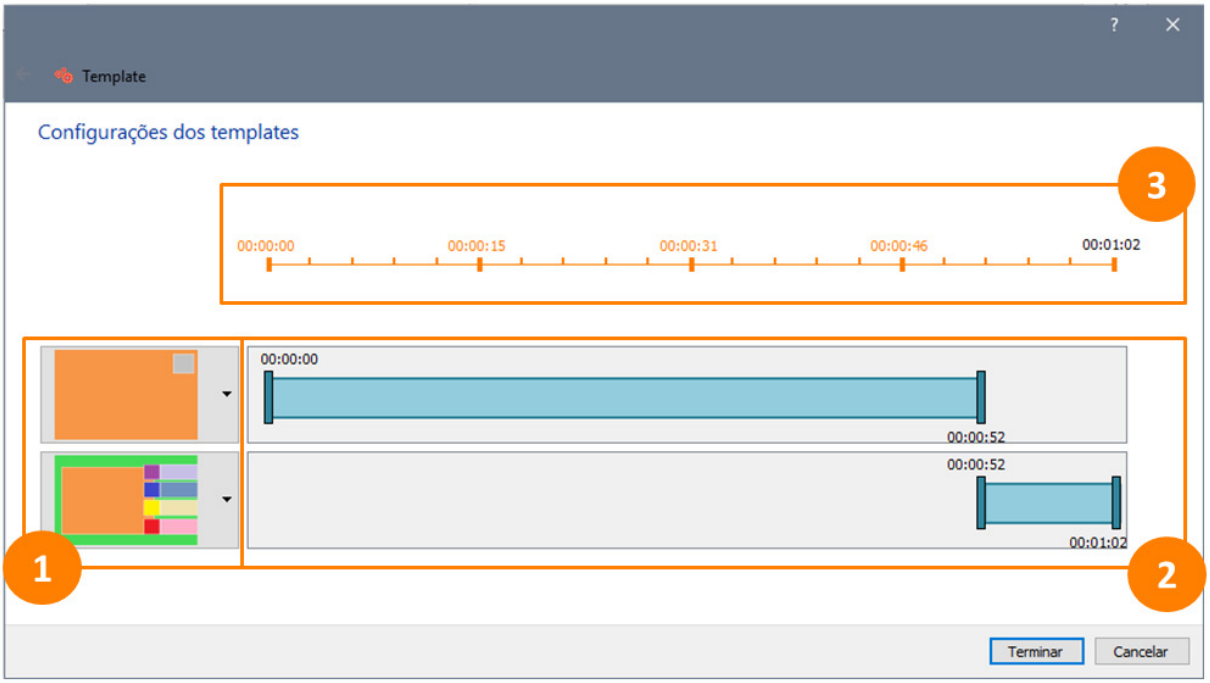
configurar a relação entre os templates 1 e 2, o botão “Nova Relação” é clicado. Com isso, o autor do OA passa a visualizar a tela de definir relação, mostrada na Figura 25.

Figura 24 – Janela de configuração de templates.



A tela “Nova Relação”, possui três visões: Visão de Templates (1), Visão de

Figura 25 – Tela de Nova Relação.



Edição (2) e Visão Temporal (3). Como as relações são compostas por um par de templates, são exibidas na Visão de Templates duas caixas de combinação. Cada caixa de combinação contém uma lista com todos os templates inseridos na cena. Assim, seleciona-se nas caixas os templates que devem compor a relação. Vale ressaltar que a interface não permite que o mesmo template seja selecionado nas duas caixas, já que não é possível um template relacionar-se com si mesmo. Na Visão de Edição é possível editar o tempo de início é fim de cada template. Isso pode ser feito pela movimentação da alça de dimensionamento ou pela linha de edição do tempo de início e fim. A Visão Temporal exibe uma linha do tempo com objetivo de auxiliar a edição dos tempos. Nela, também é possível editar o tempo final da cena.

Aos 52 segundos do vídeo “Algoritmos.mp4”, o professor questiona o aluno sobre qual algoritmo ele quer saber mais. Logo, nesse instante, devem ser exibidas as imagens com as opções e o vídeo deve ser redimensionado. Sendo assim, o template 1 deve ser finalizado no instante 52 segundos e o template 2 deve ser iniciado. Essa configuração pode ser vista também na Figura 25.

Para finalizar a definição dessa relação, é pressionado o botão “Terminar”. Como consequência, volta a ser exibida a tela de configurações de relações. Como os tempos de início e fim dos templates 1 e 2 foram definidos, é possível visualizar esses intervalos, como mostra a Figura 26. Pressionado o botão “Finalizar”, a relação é definida e os templates são exibidos na Visão de Leiaute do Cacuriá.

Assim, é possível visualizar a exibição das regiões ao longo do tempo, como mostra a Figura 27.

Figura 26 – Visualização dos templates da Cena 1.

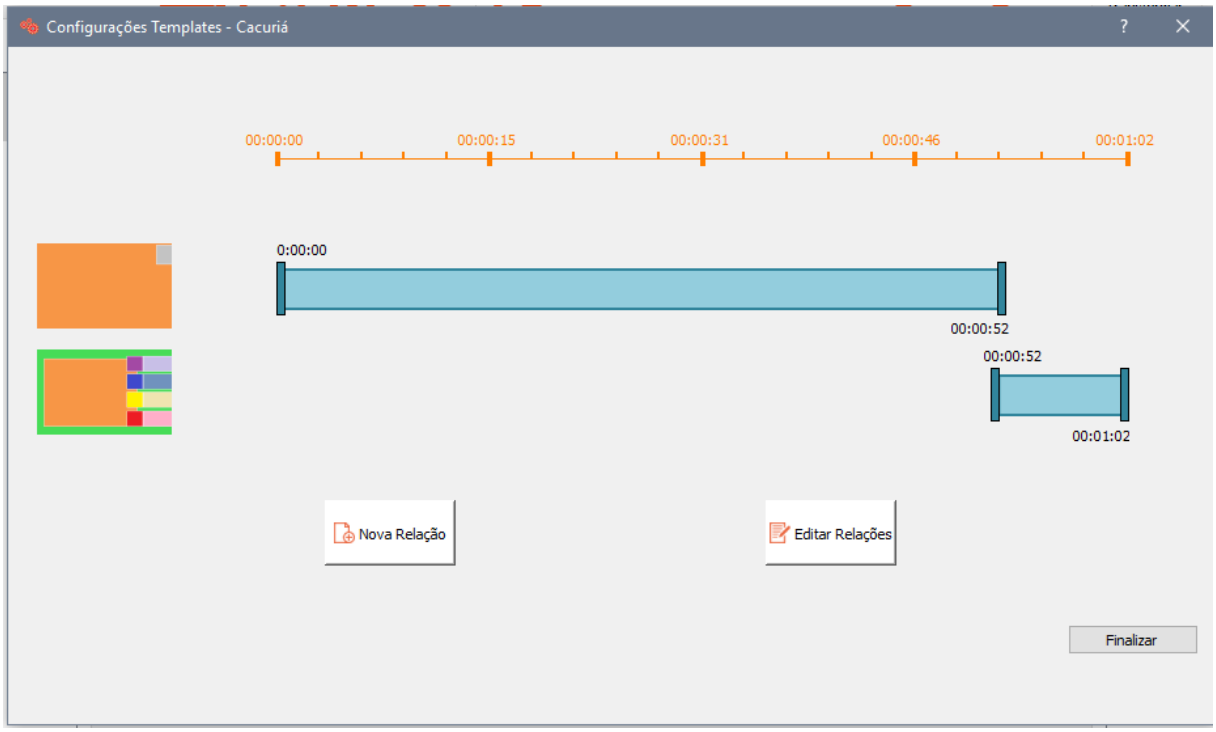
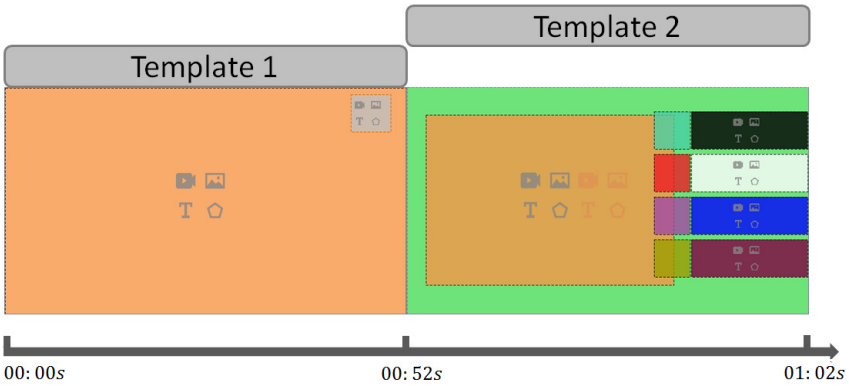
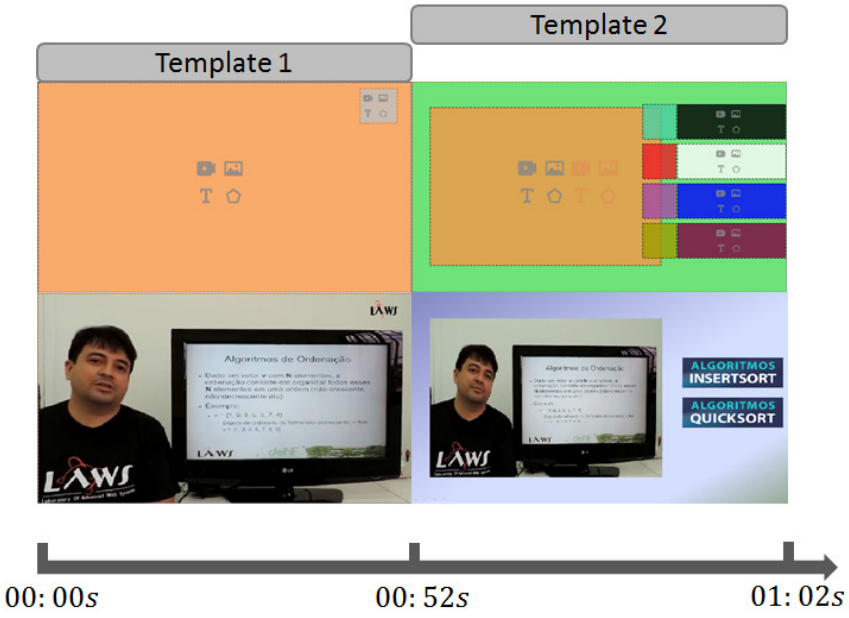


Figura 27 – Exibição das regiões ao longo do tempo.



A construção da Cena 1 é concluída com a inserção das mídias nas regiões em que devem ser exibidas. Vale ressaltar que as regiões dos templates que possuem cores iguais compartilham o mesmo identificador. Assim, quando o autor do conteúdo insere, por exemplo, o vídeo na região representada pela cor laranja no template 1, o sistema associa o mesmo vídeo nas regiões de mesma cor (laranja), no template 2. O resultado da Cena 1 é ilustrado na Figura 28.

Figura 28 – Exibição dos templates 1 e 2 ao longo do tempo.



É possível perceber que na Cena 1 foi definida a relação template 1 *meets* template 2, isso porque, o fim do template 1 coincide com o início do template 2. Assim, nota-se que os autores das aplicações podem criar relações sem conhecer os operadores de Allen e no Cacuriá, utilizando o perfil TemplateSync, as relações são identificadas e as regiões dos templates são exibidas de acordo com elas.

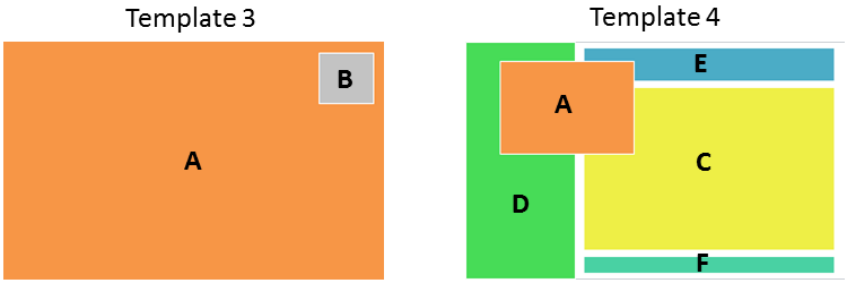
- Construção da Cena 2

  Assim como na Cena 1, a Cena 2 deve ter tempo total igual ao do vídeo exibido na mesma, chamado “Insertsort.mp4”. Logo, o tempo total da Cena 2 é definido como 7 minutos e 55 segundos.

  Na Cena 2, é exibido o vídeo com a explicação sobre o algoritmo insertsort. Durante a explicação, o professor analisa o código referente ao algoritmo. Assim, essa cena pode ser construída utilizando dois templates. Um template, chamado template 3, será exibido nos momentos em que o vídeo explicativo deve ocupar toda a tela e o segundo template, chamado template 4, será exibido quando for apresentado na tela o algoritmo. Nesse instante, o vídeo é redimensionado, ocupando uma parte menor da tela. Os templates 3 e 4 são ilustrados na Figura 29.

  O template 3 tem o mesmo <templatebase> que o template 1. O vídeo “Insertsort.mp4” apontará para a região A e nenhuma mídia apontará para a região *B*. O template 4 possui cinco regiões: a região *A* (que também está especificada no template 3), a região *C* de cor amarela, a região *D* de cor verde e as região *E* e *F*

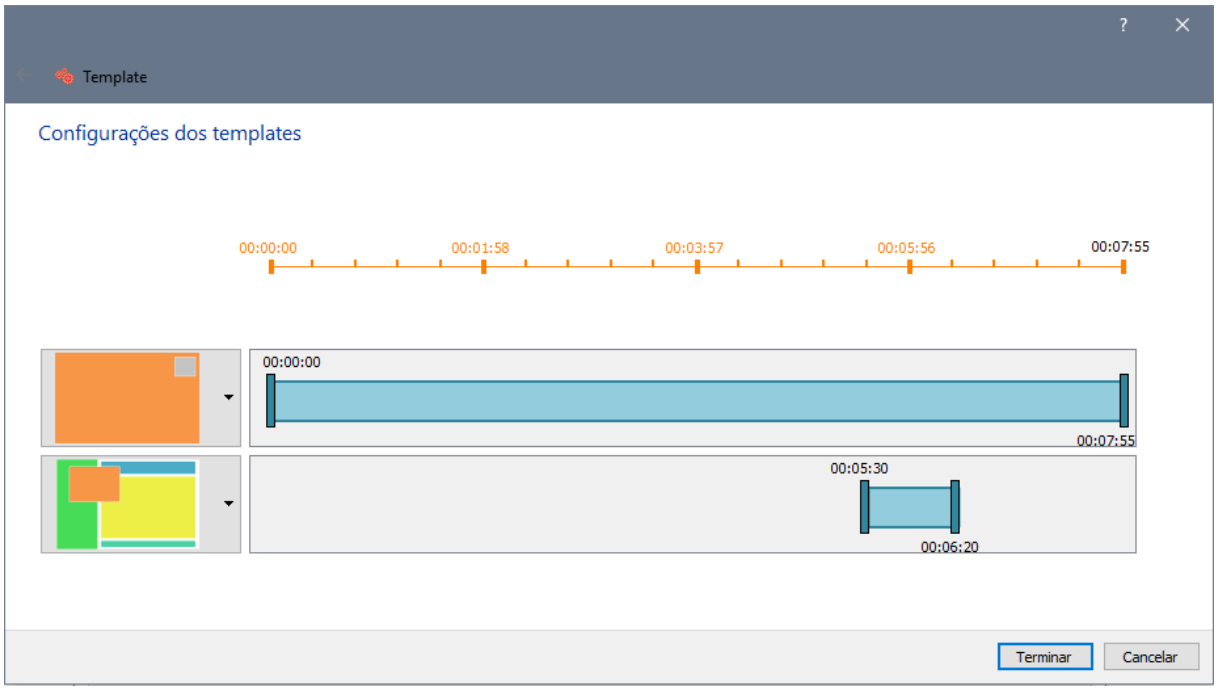
Figura 29 – Templates utilizados na Cena 2.



de cores azul e azul claro, respectivamente. A imagem com o código do algoritmo insertsort apontará para a região *C*, uma imagem de fundo apontará para a região *D* e as regiões *E* e *F* serão utilizadas para exibir os textos “Algoritmo Insertsort” e “Curso de Algoritmos”, respectivamente.

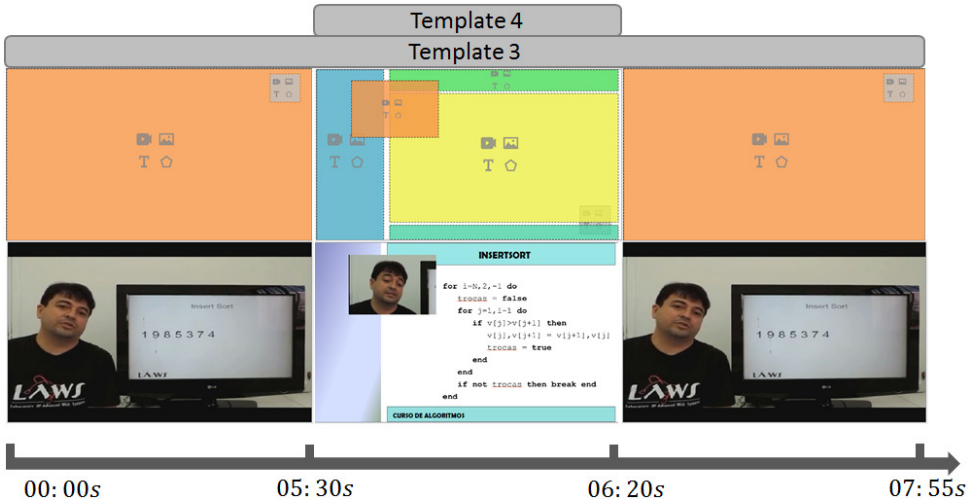
Nessa cena, é desejável que o template 4 seja visível dos instantes 05:30 ao 06:20, momento em que é apresentada a imagem com o código. No restante da cena deve ser visível o template 3, ou seja, o vídeo deve ocupar toda a tela. Para obter a configuração desejada, define-se na janela “Nova Relação” que o template 3 inicia no instante 0 segundos e termina no fim da cena, ou seja, seu tempo de fim é igual à 07:55. O template 4 tem seu início definido como 05:30 e fim como 06:20. A Figura 30 ilustra a definição dessa relação.

Figura 30 – Configuração da relação dos templates 3 e 4.



Nota-se que o template 4 é exibido durante a execução do template 3. Portanto, a relação definida foi template 3 *contains* template 4. Assim, durante a execução do template 4, o vídeo da cena é redimensionado e reposicionado, como as dimensões da região A definida no template 4. E, com o fim da execução do template 4, o vídeo volta a preencher a toda a tela, como definido no template 3. O resultado da Cena 2 é mostrado na Figura 31.

Figura 31 – Exibição dos templates 3 e 4 ao longo do tempo.



- Construção da Cena 3

  A Cena 3 é similar Cena 2. Nela, é exibido um vídeo com a explicação sobre o algoritmo quicksort e em um determinado momento da explicação o professor faz análise de um código que é apresentado na tela. Logo, serão utilizados os templates 5 e 6, que possuem a mesma base dos templates 3 e 4, respectivamente.

  O tempo total da cena é definido como 6 minutos e 40 segundos. No instante 01:00 o vídeo passa a focar no professor, que está explicando o código. Portanto, nesse momento, deve ser iniciada a execução do template 6 para que seja exibido o algoritmo. Após 50 segundos, o professor termina a explicação do código e passa a apresentar uma abordagem gráfica, exemplificando o funcionamento do quicksort. Com essas informações, é definida uma relação entre os templates 5 e 6. A Figura 32 mostra a configuração feita na janela “Nova Relação”. O resultado da construção da Cena 3 é ilustrado na Figura 33.

5.2 Telejornal

Nesta seção é apresentada a construção de um telejornal apresentado por dois âncoras, que é iniciado com um vídeo dos apresentadores ocupando toda a tela. Ao

Figura 32 – Configuração da relação dos templates 5 e 6.

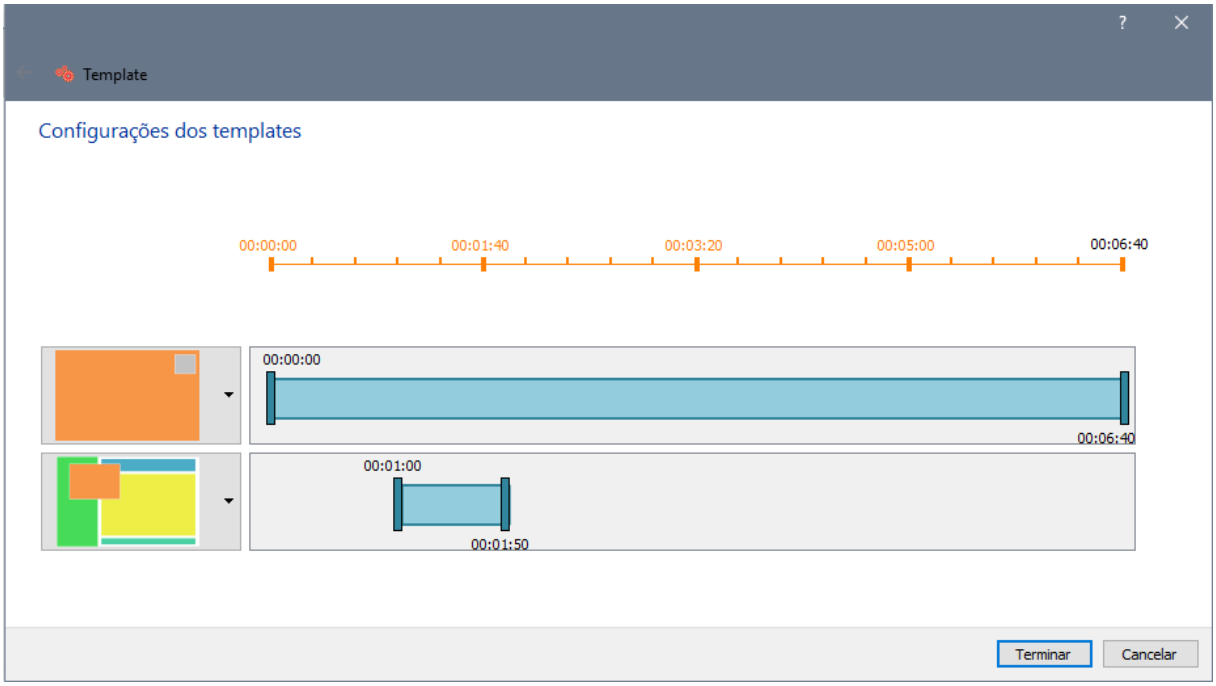
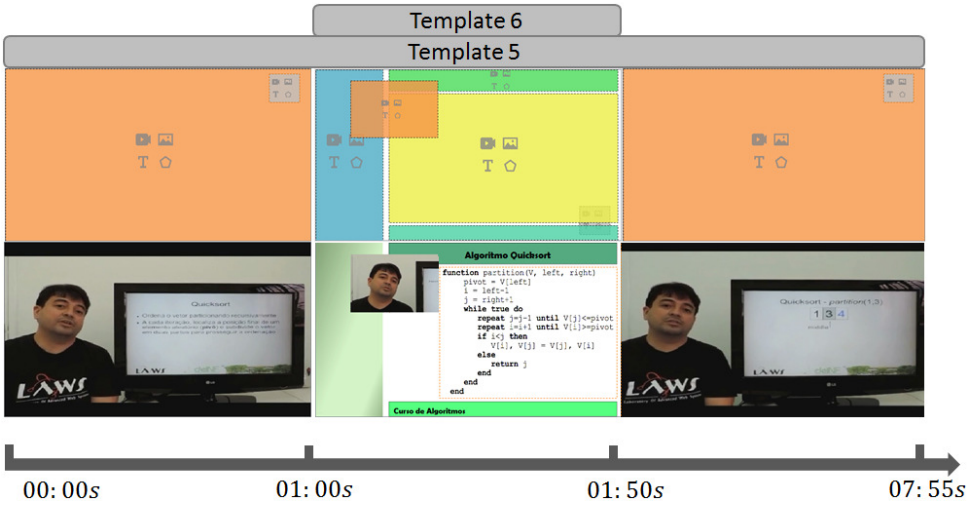


Figura 33 – Exibição dos templates 5 e 6 ao longo do tempo.

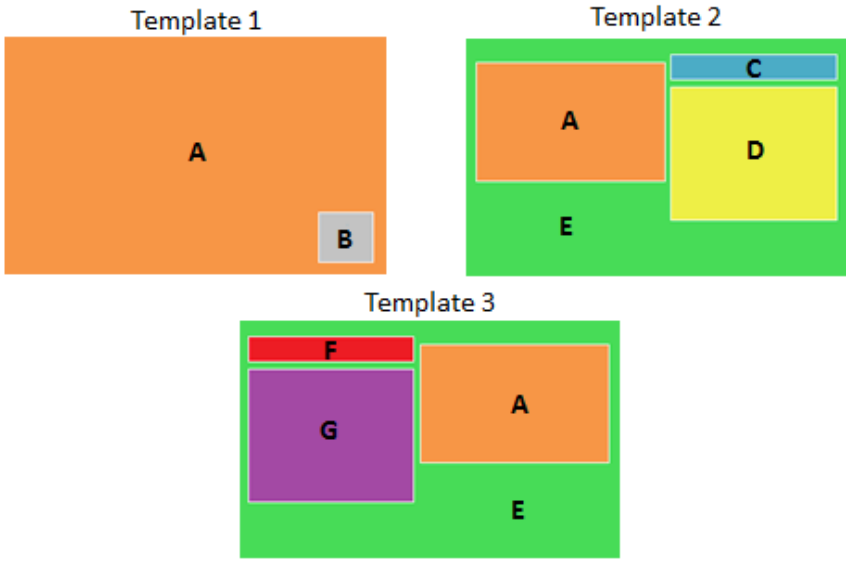


longo da aplicação, o vídeo dos apresentadores é redimensionado e reposicionado para possibilitar a exibição em sequência das notícias sobre economia e esporte. Em seguida, o vídeo dos apresentadores volta a preencher toda a tela e a aplicação é finalizada.

A construção da aplicação é feita utilizando três templates, mostrados Figura 34. O template 1 é usado nos instantes em que vídeo dos apresentadores preenche toda tela. O template 2 é utilizado quando o apresentador fala sobre economia e o template

3 quando o outro apresentador fala sobre esportes.

Figura 34 – Templates 1, 2 e 3.

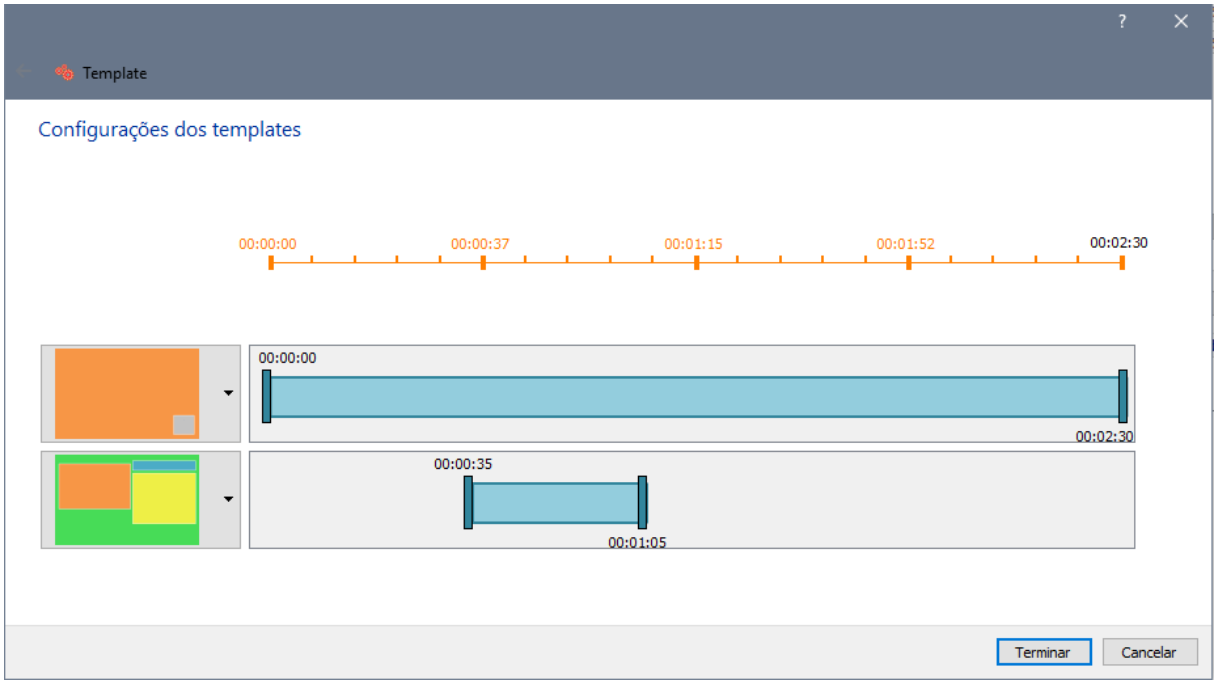


O template 1 é composto por duas regiões, uma região de cor laranja, e uma região de cor cinza, que são identificadas respectivamente pelas letras *A* e *B*. O template 2 define quatro regiões: a região *C* de cor azul, a região *D* de cor amarela, a região *E* de cor verde, e assim como o template 1, a região *A* de cor laranja. O template 3 também define quatro regiões: a região *F* representada pela cor vermelha, a região *G* de cor roxa e as regiões *A* e *E* que estão representadas respectivamente pelas cores laranja e verde. Dessa forma, o vídeo principal apontará para a região *A* de todos os templates, uma imagem com a marca da emissora de televisão apontará para a região *B*, as mídias referentes às informações sobre economia apontarão para as regiões *C* e *D*, as mídias referentes às informações sobre esportes apontarão para as regiões *F* e *G* e uma imagem de fundo apontará para as regiões *E* dos templates 2 e 3.

Primeiramente, insere-se o template 1 e o template 2. E, na configuração dos templates, define-se seus tempos de início e fim. Essa configuração é mostrada na Figura 35. O template 1 será exibido durante toda a aplicação e o template 2 será inciado aos 35 segundos e finalizado em 01:05. Isso foi definido porque o intervalo em que o template 2 é exibido, é o momento em que o apresentador passa a abordar o assunto economia. Assim, com o início do template 2, o vídeo será redimensionamento e o conteúdo sobre economia será exibido. É possível observar que o template 2 inicia depois do template 1 e é finalizado antes do fim do template 1, o que caracteriza a relação template 2 *during* template 1.

Em seguida, é inserido o template 3. Como anteriormente, é necessário especificar seus tempos de início e fim. Na janela de configuração dos templates, em uma

Figura 35 – Configuração da relação entre os templates 1 e 2.

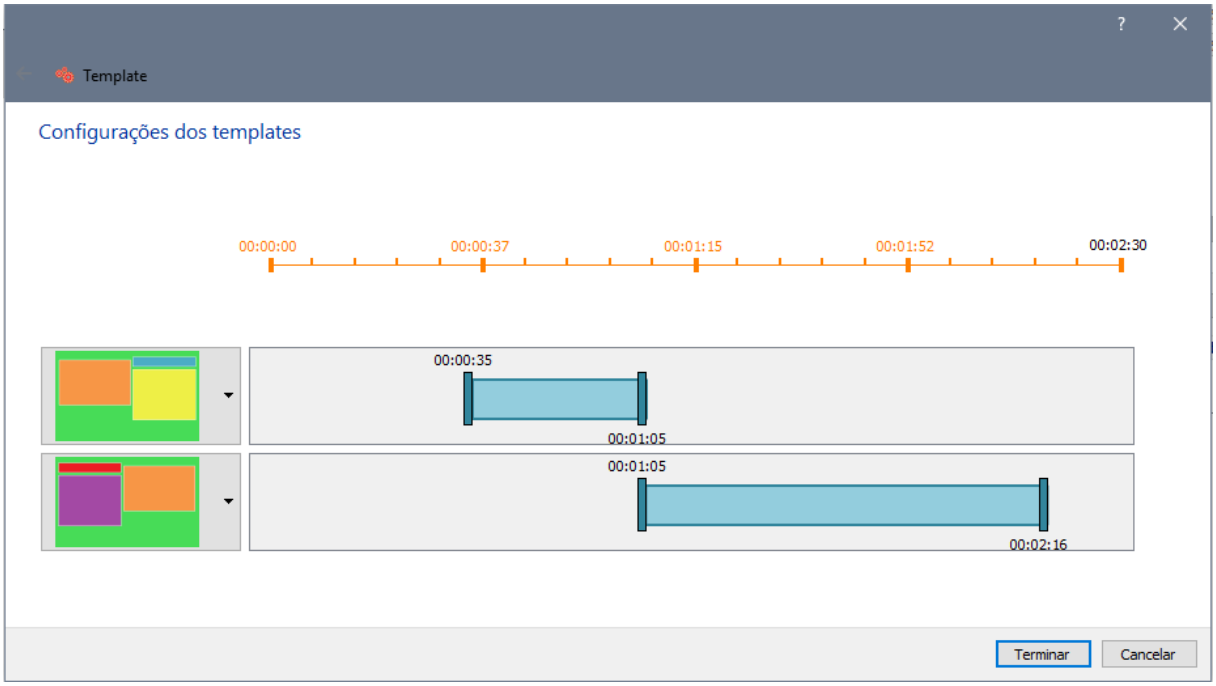


das caixas de combinação escolhe-se o template 3 e na outra o template 2. Quando seleciona-se um template que já possui tempos de início e fim definidos, esses são exibidos na Visão de Edição, o que acontece quando escolhe-se o template 2. Assim, só precisa ser configurada a exibição do template 3. Como mostra a Figura 36, o template 3 inicia no mesmo instante em que o template 2 é finalizado, ou seja, no momento em que um apresentador encerra suas informação sobre economia e o outro apresenta as notícias sobre esportes. Portanto, com a adição do template 3 na cena e com as configurações descritas, pode-se notar definição da relação template 3 *meets* template 2.

Com as relações definidas entre os templates, é possível visualizar a exibição das regiões ao longo do tempo, como mostra a Figura 37. Nota-se que o template 1 é executado do início ao fim da aplicação, ou seja, dos 0s aos 2:30s. Entretanto, os templates 2 e 3 são exibidos em sequência tendo seus tempos de início e fim definidos respectivamente de 35s a 1:05s e 1:05s a 2:15s. Para finalizar o desenvolvimento da aplicação, as mídias são inseridas nas regiões em que devem ser exibidas. O resultado da aplicação pode ser visto na Figura 37.

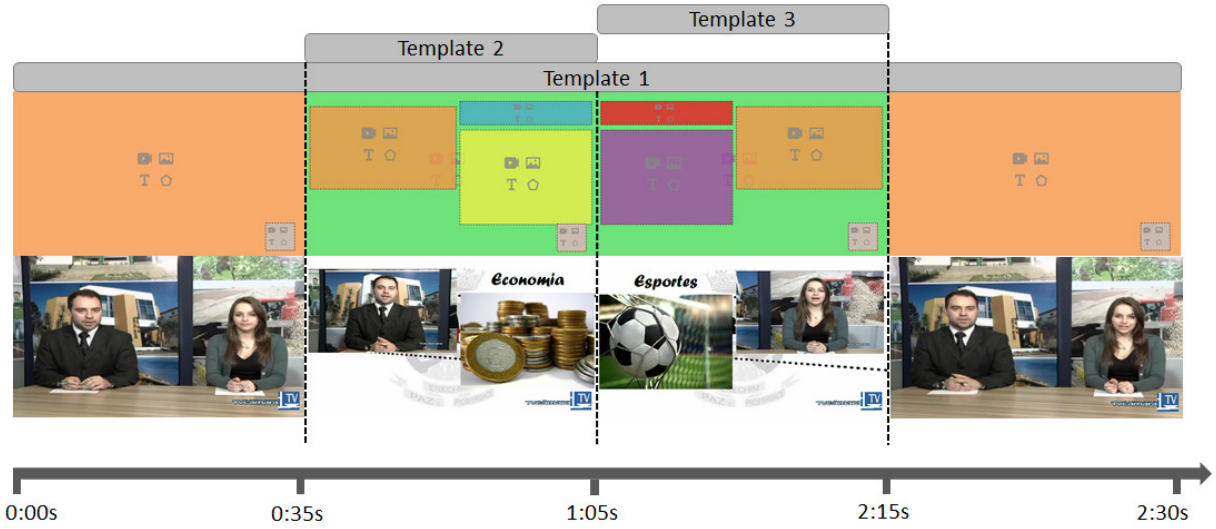
Construindo essas aplicações, é possível perceber como a utilização de templates facilita a autoria de uma aplicação multimídia e aumenta o reúso na especificação do sincronismo temporal. O exemplo permite, também, visualizar como é feita a sincronização temporal dos templates como entidades de primeira classe dentro da ferramenta de autoria. Além disso, é possível perceber como o *start* de um template

Figura 36 – Configuração da relação entre os templates 2 e 3.



cria implicitamente *stops* e *starts* em mídias.

Figura 37 – Exibição dos templates 1, 2 e 3 ao longo do tempo.



6 Conclusão

Este trabalho apresenta uma proposta para o emprego de templates de leiaute para a autoria de aplicações multimídia. O sincronismo entre os templates de leiaute é feito por *timeline*. Além disso, baseada nos operadores de Allen (ALLEN, 1983), a relação temporal entre os templates é identificada pelo *software*. A ideia é permitir que templates de leiaute sincronizados entre si sejam o conceito central da lógica temporal na autoria. Ou seja, o trabalho permite que os templates sejam sincronizados temporalmente como entidades de primeira classe. A proposta é concretizada pela definição do perfil XML TemplateSync, passível de ser incorporado a outras linguagens XML.

É descrito como os templates de leiaute são utilizados na ferramenta de autoria multimídia Cacuriá e são apresentados dois cenários de uso que descrevem a aplicação da proposta. São apresentados em detalhes os dois exemplos e é feito o passo-a-passo para suas criações. A partir disso, percebe-se que é possível aplicar o TemplateSync em uma ferramenta real, alcançando o objetivo geral deste trabalho.

Os dois exemplos apresentados podem ser criados na ferramenta Cacuriá sem a utilização do recurso template de leiaute. Entretanto, para isso, o autor da aplicação necessitaria redimensionar e reposicionar todas as mídias individualmente nas situações em que foram utilizados os templates. Isso demandaria mais tempo e precisão por parte do autor para alcançar o resultado exatamente como esperado. Ou seja, utilizando os templates de leiaute, é necessário especificar apenas o tempo no qual as mídias deverão ser rearranjadas. Assim, o tempo de desenvolvimento da aplicação é reduzido e a autoria é simplificada.

Na criação das aplicações apresentadas, também é possível perceber o aumento do reúso. Em dois momentos do desenvolvimento, diferentes mídias precisaram ser dispostas de maneira semelhante. Com a utilização de templates de leiaute, foi preciso apenas instanciar em cada um dos momentos o leiaute adequado e especificar as mídias. Sem a utilização desse recurso, seria necessário definir a posição e as dimensões das mídias com bastante precisão a fim de obter o resultado similar nas duas situações. Nesse caso, também é possível notar que a utilização de templates de leiaute promoveu a coerência da aplicação. Ou seja, o autor pôde definir e seguir um mesmo estilo de disposição das mídias ao longo da aplicação.

Com objetivo de aumentar o reúso dentro da ferramenta Cacuriá, tem-se como um trabalho futuro criar dentro da ferramenta um ambiente para autoria de templates de leiaute. Assim, além do usuário utilizar os templates de leiaute predefinidos fornecidos

atualmente, ele poderá criar seus próprios templates. Isto é, o autor de aplicações poderá definir as dimensões das regiões e identificá-las da forma que lhe for mais conveniente. Isso permitirá também, que um autor especifique um template de leiaute e compartilhe com outros usuários que poderão usá-los em suas aplicações.

Como trabalho futuro pretende-se incorporar a funcionalidade fornecida por TemplateSync em outras linguagens de autoria hipermídia hospedeiras. A intenção é ilustrar mais cenários de uso da proposta desta monografia. Outro trabalho futuro passível de ser feito é fazer uma avaliação da produtividade e usabilidade de autores quando da criação de aplicações baseadas no emprego de templates de composição sincronizados entre si.

Referências

ALLEN, J. F. Maintaining knowledge about temporal intervals. *Magazine Communications of the ACM*, v. 26, n. 11, p. 832–843, Novembro 1983. Citado 3 vezes nas páginas 16, 25 e 50.

AMORIM, G. F.; SANTOS, J. A. F. dos; MUCHALUAT-SAADE, D. C. Adaptive layouts for authoring ncl programs. In: *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web (WebMedia'13)*. [S.l.: s.n.], 2013. p. 205–208. Citado 3 vezes nas páginas 18, 21 e 22.

AMORIM, G. F.; SANTOS, J. A. F. dos; MUCHALUAT-SAADE, D. C. Adaptive layouts and nesting templates for hypermedia composite templates. In: *Proceedings of the 21th Brazilian Symposium on Multimedia and the Web (WebMedia'15)*. [S.l.: s.n.], 2015. p. 189–196. Citado 4 vezes nas páginas 15, 18, 21 e 22.

ATKINS, C. B.; LYONS, N. P.; ZHANG, X.; TRETTER, D. R. Templated recursive image composition. In: *Proceedings of the 18th ACM International Conference on Multimedia (MM'10)*. [S.l.: s.n.], 2010. p. 655–658. Citado 2 vezes nas páginas 18 e 19.

BEZERRA, D. H. D.; SOUSA, D. M. T. de; FILHO, G. L. S.; BURLAMAQUI, A. M. F.; SILVA, I. R. de M. Luar: A language for agile development of ncl templates and documents. In: *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web (WebMedia'12)*. [S.l.: s.n.], 2012. p. 395–401. Citado 2 vezes nas páginas 17 e 18.

BLANCHETTE, J.; SUMMERFIELD, M. *C++ GUI.Programming with Qt 4*. [S.l.]: Prentice Hall, 2008. Citado na página 36.

CELENTANO, A.; GAGGI, O. A laboratory for prototyping and testing multimedia presentations. *International Journal of Software Engineering and Knowledge Engineering*, v. 16, n. 4, p. 615–642, Agosto 2006. Citado na página 23.

CHEN, J.; XIAO, Y. G. J. islideshow: a content-aware slideshow system. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI'10)*. [S.l.: s.n.], 2010. p. 293–296. Citado na página 19.

CLEMENTS, P. C. A survey of architecture description languages. In: INTERNATIONAL WORKSHOP ON SOFTWARE SPECIFICATIONS AND DESIGN. *Proceedings of the 8th International Workshop on Software Specification and Design*. [S.l.], 1996. p. 16. Citado na página 17.

DAMASCENO, A. L. B.; GALABO, R. J. F.; SOARES NETO, C. S. Cacuriá: Authoring tool for multimedia learning objects. In: *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia'14)*. [S.l.: s.n.], 2014. p. 59–66. Citado 2 vezes nas páginas 16 e 36.

DAMASCENO, A. L. de B. *Cacuriá: uma ferramenta de autoria para criação de objetos de aprendizagem*. Dissertação (Mestrado) — Universidade Federal do Maranhão, São Luís, 2015. Citado na página 36.

DAMASCENO, J. R.; SANTOS, J. A. F. dos; MUCHALUAT-SAADE, D. C. Editec: Editor gráfico de templates de composição para facilitar a autoria de programas para tv digital interativa. In: *Proceedings of the 16th Brazilian Symposium on Multimedia and the Web (WebMedia'10)*. [S.l.: s.n.], 2010. p. 211–218. Citado na página 23.

DELTOUR, R.; LAYAIDA, N.; WECK, D. Limsee2: A cross-platform smil2.0 authoring tool. *The European Research Consortium for Informatics and Mathematics*, n. 61, p. 41–42, Julho 2014. Citado na página 23.

DELTOUR, R.; ROISIN, C. The limsee3 multimedia authoring model. In: *Proceedings of the 2006 ACM Symposium on Document Engineering (DocEng'06)*. [S.l.: s.n.], 2006. p. 173–175. Citado na página 17.

MUCHALUAT-SAADE, D. C.; RODRIGUES, R. F.; SOARES, L. F. G. Xconnector: Extending xlink to provide multimedia synchronization. In: *Proceedings of the 2002 ACM symposium on Document engineering*. [S.l.: s.n.], 2002. p. 49–56. Citado na página 17.

MUCHALUAT-SAADE, D. C.; SOARES, L. F. G. Xconnector e xtemplate: Estendendo xlink para aumentar expressividade e reuso. In: *VIII Simpósio Brasileiro de Sistemas Multimídia e Hipermídia-SBMedia2002*. [S.l.: s.n.], 2002. Citado na página 17.

MUCHALUAT-SAADE, D. C.; SOARES, L. F. G. Xconnector & xtemplate: Improving the expressiveness and reuse in web authoring languages. *The New Review of Hypermedia and Multimedia*, v. 8, n. 1, p. 139–169, 2003. Citado na página 17.

MYODO, E.; UENO, S.; TAKAGI, K.; SAKAZAWA, S. Automatic comic-like image layout system preserving image order and important regions. In: *Proceedings of the 19th ACM International Conference on Multimedia (MM'11)*. [S.l.: s.n.], 2011. p. 795–796. Citado 2 vezes nas páginas 19 e 20.

SANTOS, J. A. F. dos; MUCHALUAT-SAADE, D. C. Linguagem xtemplate 3.0: Facilitando a autoria de programas ncl para tv digital interativa. In: *Proceedings of the 15th Brazilian Symposium on Multimedia and the Web (WebMedia'09)*. [S.l.: s.n.], 2009. Citado na página 17.

SILVA, J. V. da; MUCHALUAT-SAADE, D. C. Next - graphical editor for authoring ncl documents supporting composite templates. In: *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web (WebMedia'12)*. [S.l.: s.n.], 2012. p. 387–394. Citado 2 vezes nas páginas 23 e 24.

SOARES, L. F. G.; RODRIGUES, R. F.; MUCHALUAT-SAADE, D. C. Modelo de contextos aninhados - versão 3.0. *Relatório Técnico, Laboratório Telemídia, PUC-Rio*, 2003. Citado na página 15.

SOARES NETO, C. d. S. *Autoria de Documentos Hipermídia Orientada a Templates*. Tese (Doutorado) — PUC-RIO, Departamento de Informática, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2010. Citado na página 17.

SOARES NETO, C. S.; SOARES, L. F. G.; SOUZA, C. S. de. Tal - linguagem para autoria de templates de documentos hipermídia. In: *Proceedings of the 16th Brazilian Symposium on Multimedia and the Web (WebMedia'10)*. [S.l.: s.n.], 2010. p. 147–154. Citado 2 vezes nas páginas 15 e 17.

TIAN, A.; ZHANG, X.; TRETTER, D. R. Content-aware photo-on-photo composition for consumer photos. In: *Proceedings of the 19th ACM International Conference on Multimedia (MM'11)*. [S.l.: s.n.], 2011. p. 1549–1552. Citado na página 21.

VUORIMAA, P.; BULTERMAN, D.; CESAR, P. Smil timesheets 1.0. *Working draft, W3C*, 2008. Citado na página 15.

W3C. *HTML5*. 2014. <http://www.w3.org/TR/html5/>. [Acessado em 20/05/2016]. Citado na página 15.

W3C. *CSS Snapshot 2015*. 2015. <https://www.w3.org/TR/CSS/>. [Acessado em 20/05/2016]. Citado na página 15.

W3C, W. W. W. C. et al. *XML Schema*. [S.l.]: World Wide Web Consortium, 2003. Citado na página 31.