4
Arquitetura para aplicações NCL dinâmicas

Além de ser composta por objetos de mídia com conteúdo audiovisual, uma aplicação NCL dinâmica precisa ter objetos NCLua geradores de conteúdo em sua composição. A autoria de aplicações NCL dinâmicas é mais complexa em relação às aplicações com documentos especificados inteiramente em tempo de autoria. Tal complexidade ocorre porque a autoria dessas aplicações envolve tanto a construção dos documentos NCL iniciais, compostos por objetos de mídia e relacionamentos conhecidos em tempo de autoria, quanto a especificação dos geradores de conteúdo, escritos em linguagem imperativa Lua, o que requer certa experiência em programação.

A priori, observa-se que cada aplicação NCL dinâmica possui geradores de conteúdo com lógica específica à aplicação. Além disso, o autor da aplicação precisa determinar como esses objetos geradores de conteúdo estão relacionados com os outros objetos de mídia da aplicação. Portanto, cabe ao autor da aplicação as tarefas de estruturar logicamente os objetos NCLua geradores de conteúdo e especificar a semântica de relacionamento entre esses e os demais objetos de mídia.

As próximas seções deste capítulo apresentam uma proposta de arquitetura que visa amenizar o impacto do aumento de complexidade nas *aplicações que se recriam dinamicamente*. Em suma, arquitetura é o particionamento cuidadoso de um sistema, incluindo as relações específicas entre as partes (Clements, 2010). Esse particionamento permite, por exemplo, que grupos de indivíduos trabalhem cooperativamente para solucionar um problema maior. A arquitetura é o que faz as partes trabalharem juntas fazendo o todo funcionar coerentemente e corretamente.

Essa proposta se baseia em identificar uma estrutura lógica recorrente em aplicações que se recriam dinamicamente. Tal estrutura recorrente evidencia parte de um processamento que é genérico para quaisquer aplicações desse tipo, deixando para o autor de uma aplicação específica uma base de conhecimento que

torna mais direta a implementação dos geradores de conteúdo e sua integração com a aplicação. O trabalho de identificação dessa estrutura recorrente partiu da arquitetura de um exemplo de aplicação NCL que se recria dinamicamente, como demonstram as seguintes seções deste capítulo.

Antes de entrar em detalhes específicos sobre a arquitetura proposta, a Seção 4.1 apresenta a descrição dos requisitos da aplicação NCL cuja arquitetura é especificada a seguir. Em seguida, a Seção 4.2 faz uma breve introdução sobre a importância de arquiteturas para a geração de conteúdo dinâmico, e apresenta os elementos presentes na arquitetura proposta sob algumas visões, a partir dos requisitos da aplicação descritos na Seção 4.1.

4.1. Exemplo de aplicação NCL que se recria dinamicamente

A ideia da solução de arquitetura proposta nesta dissertação surgiu a partir de requisitos do projeto de pesquisa denominado **SAGGA** (do inglês, *Support to Automatic Generation of Ginga Applications*). O projeto **SAGGA** está sendo desenvolvido em parceria entre o Laboratório TeleMídia da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e o Laboratório LAWS da Universidade Federal do Maranhão (UFMA). Esse projeto propõe a implementação de uma infra-estrutura para a geração automática de aplicações NCL 3.0 baseadas em conteúdos disponíveis nos *sites* Zappiens[4], Overmundo[5] e Clube NCL[6]. Esses *sites* são administrados pelo CGI.br, Instituto Overmundo e PUC-Rio, respectivamente, que concederam a licença de uso de seus conteúdos no âmbito do projeto. O projeto, no entanto, não se limita a esses *sites*, e poderia ser igualmente utilizado em *sites* de emissoras de TV, ou mesmo outros *sites* de conteúdo audiovisual. Como o Ginga-NCL é padrão brasileiro para TV digital terrestre e padrão ITU-T para serviços IPTV, as aplicações poderão ser executadas tanto no Sistema Brasileiro de TV Digital (SBTVD), quanto em serviços de entrega de TV via redes IP.

Um dos módulos do projeto **SAGGA**, denominado **SAGGA1**, consiste na geração automática de aplicações contendo vídeo, oriundo de algum *site Web*

[4] http://www.zappiens.br
[5] http://www.overmundo.com.br
[6] http://www.clube.ncl.org.br

(inicialmente limitados aos *sites* colaborativos Overmundo e Zappiens), ao qual podem ser agregadas legendas especificadas em SRT (*Subrip subtitle format*). Além da agregação de legendas, a aplicação NCL permitirá a navegação e exibição de outros vídeos presentes nos mesmos *sites* (Overmundo e Zappiens) e de aplicações NCL (oriundas inicialmente apenas do *site* Clube NCL), que estejam semanticamente relacionados ao vídeo original de entrada.

A aplicação do módulo **SAGGA1** foi escolhida como ponto de partida para a especificação da arquitetura proposta nesta dissertação. A aplicação de **SAGGA1** possui as características de uma aplicação NCL que se recria dinamicamente. Mesmo após ser recriada, a aplicação continua conforme o *template* especificado em tempo de autoria. Ela se recria quando ocorre um evento de interação do usuário, pois a seleção de um dos vídeos semanticamente relacionados interrompe a aplicação para iniciar uma nova exibição. O vídeo principal é substituído pelo vídeo selecionado, a legenda em formato SRT é substituída pela legenda correspondente ao vídeo selecionado, e o menu de vídeos relacionados é substituído pelos vídeos semanticamente relacionados ao vídeo selecionado.

4.2. Arquitetura da aplicação SAGGA1

Em processos de produção de *software*, a documentação de arquiteturas auxilia os *arquitetos* a decidir com mais correção as decisões de sistema, informa aos *desenvolvedores* como implementar essas decisões e registra essas decisões para dar informações sobre a solução do arquiteto a futuros *mantenedores do sistema* (Clements, 2010).

Segundo o que foi publicado em (Clements, 2010), um documento de arquitetura de *software* consistente é composto por um conjunto de visões relevantes, isto é, visões que possuem informações importantes de serem comunicadas a arquitetos, desenvolvedores e mantenedores do sistema. Cada visão descreve sob seu ponto de vista os elementos da arquitetura e suas propriedades, as relações entre esses elementos e as propriedades dessas relações, e também as interfaces e comportamento dos mesmos elementos da arquitetura.

A arquitetura da aplicação **SAGGA1** é especificada nesta seção através de um conjunto de visões. Inicialmente, a visão de casos de uso descreve a funcionalidade dessa aplicação por meio da associação entre os atores (usuários) e os casos de uso. Em seguida, são apresentados os elementos que compõem a arquitetura da aplicação e uma visão da estrutura lógica da aplicação.

4.2.1. Visão de casos de uso

A visão de casos de uso ilustrada pelo diagrama da Figura 4.1 tem a finalidade de apresentar as funcionalidades providas pelo módulo **SAGGA1** a partir dos requisitos que foram descritos na Seção 4.1. O Anexo I contém os cenários principais de cada caso de uso desse diagrama.

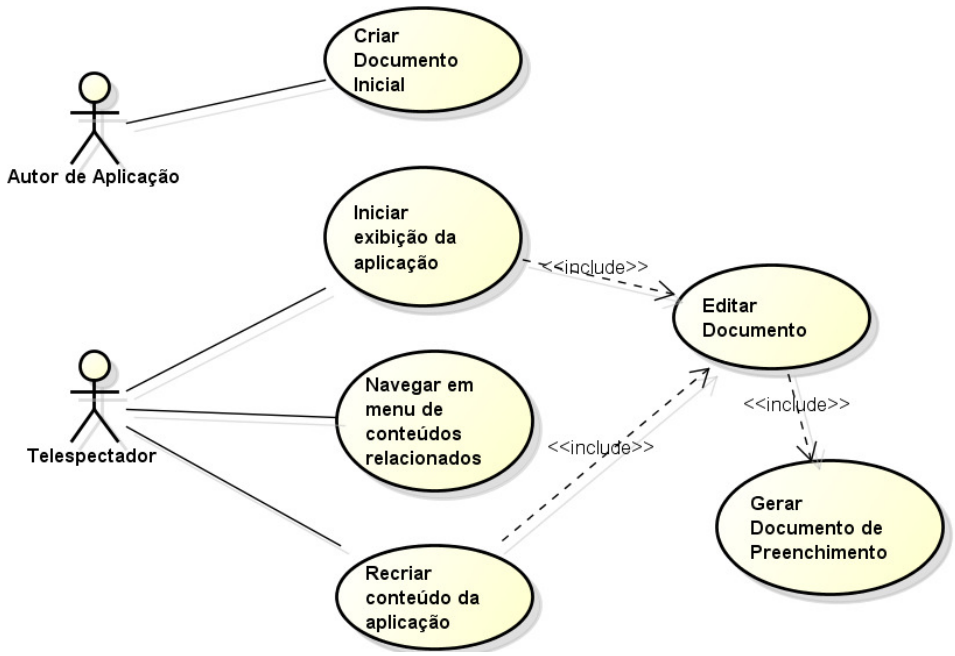

Figura 4.1 - Diagrama de casos de uso de SAGGA1

O caso de uso associado ao autor de aplicação é “Criar documento inicial”. Em suma, o autor de aplicação cria um documento inicial, escolhendo o vídeo principal e o arquivo de legenda em formato SRT associado ao vídeo. A partir dessa escolha, o documento inicial é estruturado de forma a estar pronto para ser iniciado. A Figura 4.2 apresenta uma listagem de um documento criado pelo autor. O documento criado pelo autor de aplicação é chamado de “inicial” porque

os objetos de mídia que compõem o documento serão adicionados em tempo de apresentação.

```
1.  <ncl id="generated" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
2.      <head>
3.          <regionBase> ... </regionBase>
4.          <descriptorBase> ... </descriptorBase>
5.          <connectorBase> ... </connectorBase>
6.      </head>
7.      <body id="saggaBody">
8.          <port id="p1" component="generator"/>
9.          <media id="main_settings" type="application/x-ginga-settings">
10.                 <property name="service.currentFocus"/>
11.                 <property name="service.currentKeyMaster"/>
12.                 <property name="set_data"/>
13.         </media>
14.         <media id="generator" src="generator.lua" descriptor="dlib">
15.                 <property name="builder"/>
16.                 <property name="template_processor"/>
17.                 <property name="template"/>
18.                 <property name="scope"/>
19.                 <property name="data"/>
20.                 <property name="action"/>
21.                 <area id="execute" label="execute"/>
22.                 <area id="start_component" label="start_component"/>
23.                 <area id="stop_component" label="stop_component"/>
24.         </media>
25.         <link id="link_init" xconnector="saggaBase#onBeginSet">
26.                 <bind role="onBegin" component="generator"/>
27.                 <bind role="set" component="generator" interface="builder">
28.                         <bindParam name="val" value="sagga1Builder"/>
29.                 </bind>
30.                 <bind role="set" component="generator"
    interface="template_processor">
31.                         <bindParam name="val" value="xtemplate-in"/>
32.                 </bind>
33.                 <bind role="set" component="generator" interface="scope">
34.                         <bindParam name="val" value="saggaBody,sagga_composite"/>
35.                 </bind>
36.                 <bind role="set" component="generator" interface="template">
37.                         <bindParam name="val" value="template_sagga1.ncl"/>
38.                 </bind>
39.                 <bind role="set" component="generator" interface="action">
40.                         <bindParam name="val" value="replace"/>
41.                 </bind>
42.                 <bind role="set" component="generator" interface="data">
43.                         <bindParam name="val"
    value="site=overmundo|video=http://overmundo.com.br/download_banco/outro-olhar-
    violencia-domestica|srt=http://overmundo.com.br/download_banco/violencia.srt"/>
44.                 </bind>
45.         </link>
46.     </body>
47. </ncl>
```

Figura 4.2 - Exemplo de documento NCL inicial de SAGGA1

O documento NCL inicial presente na listagem da Figura 4.2 contém um objeto NCLua identificado como *generator* (*id* = *generator*) (linha 14 da Figura 4.2). O ator telespectador, ao iniciar a exibição da aplicação, inicia também o NCLua *generator* (ou gerador), pois esse objeto foi colocado como uma porta (*port*) – que aparece na linha 8 da Figura 4.2 – para o corpo (*body*) do documento (em NCL, os objetos que são portas de uma composição são iniciados junto com essa composição).

Ao ser iniciado o objeto NCLua *generator,* o documento faz atribuições a propriedades desse objeto, por meio do elo (*link*) identificado por *link_init*, entre as linhas 25 e 45 da Figura 4.2. A função das atribuições para cada propriedade será detalhada posteriormente neste capítulo. Aqui se dá ênfase à atribuição da propriedade *data*. A ação de atribuir um valor a essa propriedade ativa no gerador o caso de uso "Editar documento", que prepara o documento com o vídeo principal, a legenda em formato SRT e com os menus de vídeos relacionados oriundos dos *sites* envolvidos no projeto **SAGGA**. O caso de uso "Gerar documento de preenchimento" é incluído pelo "Editar documento" porque a edição do documento passa pela geração automática do documento de preenchimento, baseado no valor da propriedade *data* configurada por *link_init*.

A preparação feita no documento pelo caso de uso "Editar documento" se baseia em adicionar, por comando de edição, um objeto de composição do tipo contexto (*context*) que representa a nova estrutura lógica do documento. Esse contexto possui um objeto NCLua que controla a navegação no menu de conteúdos relacionados e elo que atribui um valor à propriedade *data* do objeto *generator* quando um dos vídeos é selecionado, através de uma ação semelhante à ação do elo *link_init*. O caso de uso "Recriar conteúdo da aplicação" consiste na ação de seleção de um vídeo relacionado, e é semelhante a "Iniciar exibição da aplicação" porque a ação de selecionar um vídeo relacionado representa semanticamente o reinício da exibição da aplicação.

O caso de uso "Navegar em menu de conteúdos relacionados" representa a funcionalidade de que o usuário dispõe para navegar, por meio de teclas direcionais, no menu de *sites* (Overmundo, Zappiens e Clube NCL) e no menu de conteúdos. Ao selecionar uma das opções de *sites*, o usuário tem acesso aos conteúdos relacionados ao vídeo principal hospedados no *site* selecionado.

4.2.2.
Visão estrutural da aplicação

A visão estrutural representa a estrutura lógica do documento NCL, ou seja, nela estão representados os objetos de composição, os objetos de mídia e os elos (relacionamentos). Na visão estrutural, os objetos de composição e os objetos de mídia são representados por vértices de um grafo (compostos e atômicos, respectivamente), enquanto os elos são representados por arestas entre vértices do grafo.

A Figura 4.3 representa uma visão estrutural de documentos gerados no módulo **SAGGA1**. Nessa visão estrutural estão presentes: um vértice *Vídeo* representando o objeto de vídeo principal; um vértice *Lua-SRT* representando um objeto NCLua exibidor (*player*) de legendas no formato SRT; objetos de mídia *Z*, *O* e *C* representando, respectivamente, objetos de imagens referentes aos *sites* Zappiens, Overmundo e Clube NCL do menu de *sites*; e objetos de composição contendo imagens que representam o menu de vídeos semanticamente relacionados ao vídeo principal de cada *site*[7]. Vale ressaltar que o exibidor NCLua *Lua-SRT* foi desenvolvido para esta dissertação porque a Norma ABNT NBR 15606-2 (ABNT NBR, 2007) do SBTVD não prevê na implementação do Ginga-NCL a existência de um exibidor para o tipo de mídia SRT.

As arestas pontilhadas, originadas dos objetos de composição na Figura 4.3, representam portas para os objetos de mídia, simbolizando que estes são exibidos quando o objeto de composição se inicia.

Os elos representados pelas arestas contínuas com papéis *onSelection* e *start* partindo das imagens *Z*, *O* e *C* significam que ao selecionar uma dessas imagens é iniciada a composição contendo o menu de vídeos relacionados do *site* selecionado.

[7] Na Figura 3.2, os objetos de composição dos vídeos semanticamente relacionados aos *sites* Zappiens e Clube NCL foram omitidos por questões didáticas, podendo ser inferidos pela representação do objeto de composição do *site* Overmundo.

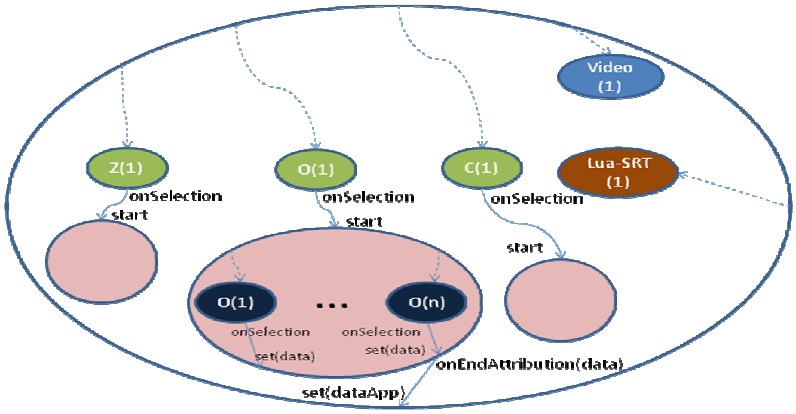

Figura 4.3 - Visão estrutural de composição hipermídia SAGGA1

Quando uma das imagens do menu de vídeos relacionados é selecionada, inicia-se a atribuição de uma propriedade denominada *data*. Esses relacionamentos estão representados na Figura 4.3 para o *site* Overmundo pelas setas contínuas partindo dos objetos *O(i)* (onde *i* é um inteiro que varia entre 1 e o número de vídeos relacionados do Overmundo) com papéis *onSelection* e *set(data)*. De forma análoga tais relacionamentos existem para os *sites* Zappiens e Clube NCL, apesar de terem sido omitidos da Figura 4.3 por questões didáticas. A propriedade *data* é utilizada para comunicar informações sobre o objeto que foi selecionado, ou seja, o valor de *data* corresponde a metadados de objetos do menu de conteúdos relacionados. Podem ser alguns desses metadados: a URL que representa onde está localizado o vídeo; *tags* que associem o vídeo a determinados assuntos como esporte, política, religião etc.; o *site* ao qual o vídeo pertence etc. O valor da propriedade *data* permite à aplicação determinar qual vídeo foi selecionado no menu e extrair as informações necessárias para a busca de vídeos relacionados realizada pelo gerador de conteúdos. O elo com os papéis *onEndAttribution(data)* e *set(dataApp)*, quando disparado, atribui à propriedade *dataApp* da composição mais externa o valor de *data*.

A visão estrutural da Figura 4.3 omite o gerador de conteúdos dinâmicos. Essa omissão se justifica porque a Figura 4.3 foca na representação da composição hipermídia da aplicação, isto é, nos objetos de mídia e os relacionamentos entre eles. Essa composição hipermídia é a estrutura lógica que será *recriada dinamicamente* pelo gerador de conteúdo.

A complementação dessa visão estrutural com a representação do gerador de conteúdo é ilustrada na Figura 4.4. A composição apresentada por essa figura

incorpora um objeto Lua Gerador que é responsável por gerar o conteúdo quando há seleção de um vídeo relacionado. Observa-se que há um elo entre a composição que contém a lógica da aplicação (a mesma composição retratada na Figura 4.3) e o gerador. Esse elo estabelece que, ao fim da atribuição de *dataApp* (papel *onEndAttribution(dataApp)*), atribui-se à propriedade *data* do gerador (papel *set(data)*) o valor de *dataApp*, encerra-se a exibição da aplicação (papel *stop*) e inicia-se a nova composição da aplicação (papel *start*) após o processamento feito pelo gerador.

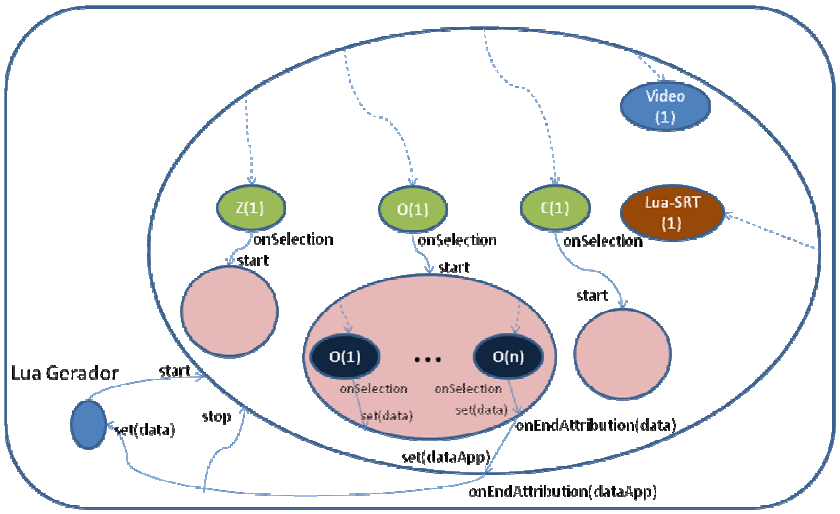

Figura 4.4 - Visão estrutural de SAGGA1 com gerador de conteúdo monolítico

Visualmente, a ação de selecionar um vídeo relacionado em um dos menus corresponde à interrupção do vídeo principal sendo exibido no momento e ao início do vídeo selecionado, além do surgimento dos novos menus de vídeos relacionados. Estruturalmente, ocorre uma recriação da aplicação. É como se houvesse a autoria de uma nova aplicação com a mesma estrutura da Figura 4.3 sempre que um vídeo é selecionado. Ao invés da autoria ser feita por intervenção humana, cabe ao Lua Gerador desempenhar essa função.

Nesta dissertação está presente o intuito de utilizar os benefícios da autoria de aplicações com o auxílio de *templates*. O objeto Lua Gerador – cuja função é realizar a recriação (autoria) da aplicação dinamicamente – representado na Figura 4.4 não é conforme ao método de autoria orientado a *templates*, haja vista que tal visão estrutural não é compatível com o fluxo de trabalho desse método, o qual foi ilustrado na Figura 2.5. A visão da Figura 4.4 apresenta um objeto *Lua Gerador*

monolítico, que gera a composição inteira da aplicação sem considerar a separação entre a autoria de documentos e a autoria de *templates*.

A visão estrutural completa da aplicação **SAGGA1** compatível com o fluxo da Figura 2.5 está representada na Figura 4.5. *Lua Controlador* é um objeto NCLua que recebe solicitações de geração de conteúdo oriundas de *Lua Gerador* por meio de *getContext(data)*, uma interface do *Lua Controlador* que recebe metadados (parâmetro *data*) e retorna a composição da aplicação que o *Lua Gerador* deve adicionar por comando de edição para recriar a aplicação.

O objeto *Lua Controlador* se comunica com o objeto gerador de documentos de preenchimento (objeto *FillDocGen*) por meio da interface *getDoc(data)*, que retorna o documento de preenchimento baseado nos metadados contidos no parâmetro *data*. Após receber de *FillDocGen* o documento de preenchimento, *Lua Controlador* invoca *getContext(doc, template)* do objeto *Processador de Template*. O processador de *template* recebe através de *getContext* o documento de preenchimento (parâmetro *doc*) e o *template* da aplicação (parâmetro *template*) e retorna a composição da aplicação pronta para ser encaminhada ao *Lua Gerador*.

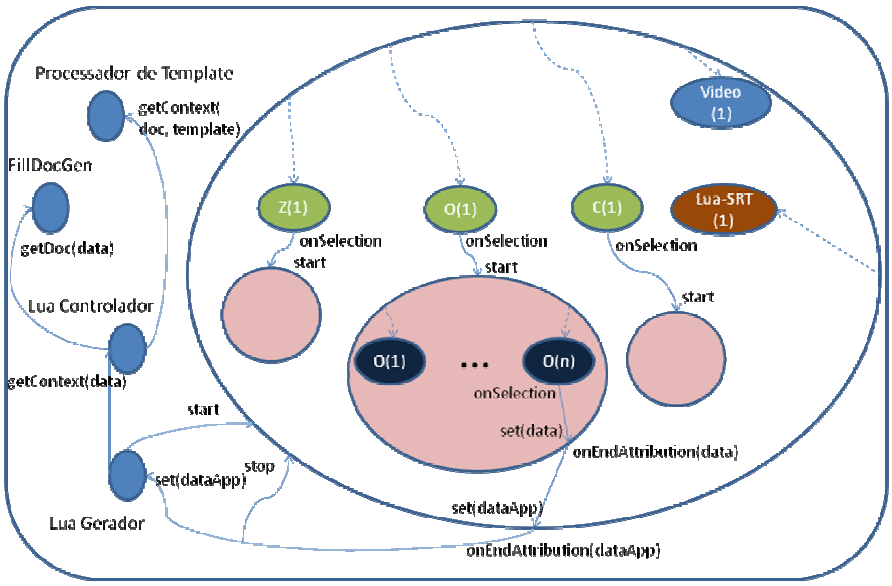

Figura 4.5 - Visão estrutural completa de documentos SAGGA1

4.2.3. Generalização da arquitetura

A visão estrutural representada pela Figura 4.5 corresponde a uma arquitetura específica para a aplicação do módulo **SAGGA1**. Observa-se nessa visão estrutural que é possível isolar a composição hipermídia – como aparece na Figura 4.3 – do gerador de conteúdos, e estabelecer uma interface entre eles por meio de eventos de atribuição a propriedades do gerador. Dessa forma, a arquitetura abrange aplicações com composições hipermídia genéricas, como mostra a Figura 4.6.

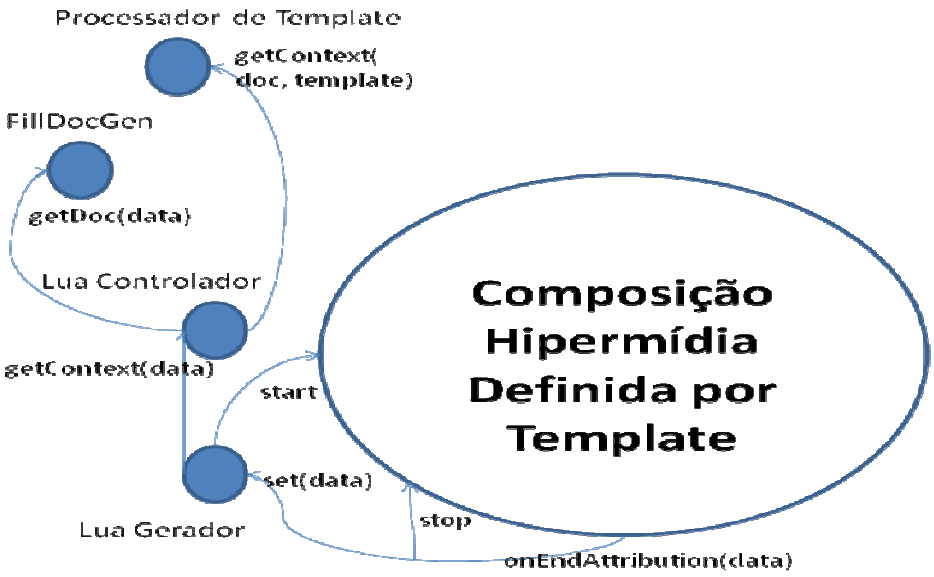

Figura 4.6 - Visão estrutural genérica para aplicações que se recriam dinamicamente

A Figura 4.6 é a visão estrutural da arquitetura genérica para aplicações que se recriam dinamicamente. Essa arquitetura, que é baseada na utilização de *templates*, é uma importante contribuição deste trabalho. Os detalhes de como esses módulos interagem entre si são apresentados no capítulo seguinte desta dissertação, que aborda a parte concreta de implementação do presente trabalho.