4
O Fomatador NCL

O Formatador NCL compõe a máquina de apresentação de documentos NCL. Ele é o elemento responsável por receber a especificação de um hiperdocumento (sua estrutura, seus relacionamentos e a descrição da forma de exibição desejada) e concretizá-la na apresentação propriamente dita (Rodrigues, 2003).

No entanto, a implementação Java do Formatador NCL é voltada para a plataforma Java SE. Para que essa implementação possa executar com o comportamento esperado em sistemas de TV digital, são necessárias algumas adaptações e otimizações a serem discutidas neste capítulo.

Para cumprir seus propósitos o capítulo está organizado da forma a seguir. A Seção 4.1 faz uma breve descrição da versão Java do Formatador NCL. A Seção 4.2 sugere uma otimização no método utilizado pelo Formatador para leitura e conversão do documento NCL. A Seção 4.3 sugere o desmembramento do Formatador em componentes de software. E, por fim, a Seção 4.4 sugere algumas mudanças nos em cada módulo do formatador.

4.1.
O Formatador NCL Java

A implementação Java atual do Formatador pode ser estruturada em módulos como mostrado na **Figura 16**. Para realizar suas tarefas, tal implementação conta com o auxílio do conversor, do escalonador e informações sobre o contexto de execução. Além dessas entidades, o Formatador faz uso dos serviços oferecidos por mais três elementos: o Gerenciador de Leiaute, o Gerenciador de Documentos e o Gerenciador de Adaptadores para Exibidores.

Figura 16 – Arquitetura do Formatador NCL.

O processo de apresentação se inicia quando o Formatador recebe um comando de edição específico para iniciar a apresentação de um documento. A partir desse comando, o Formatador requisita os serviços do Gerenciador de Documentos.

O Gerenciador de Documentos tem a responsabilidade de processar os comandos de edição delegados pelo Formatador e realizar a manutenção dos documentos NCL ativos em **uma** base de documentos. Ao receber um comando de edição, o gerenciador poderá: realizar operações sobre a base de documentos (ex.: exclusão de documento, exclusão de um elo em um documento, entre outras); ou, caso o comando de edição contenha especificações XML (por exemplo, para adição de documentos, elos, conectores, entre outros) (Soares et al., 2006), o Conversor NCL é acionado.

O Conversor NCL é a entidade responsável por transformar especificações XML em entidades do modelo conceitual NCM. São essas entidades que poderão ser, efetivamente, armazenadas na base de documentos pelo Gerenciador de Documentos.

As entidades inseridas na base de documentos estão aptas a serem apresentadas. Para iniciar esse processo, no entanto, é preciso convertê-las para o

modelo de execução interno do Formatador. Isso é feito através do Conversor do Modelo de Execução.

Essa segunda conversão é realizada sob demanda. O critério utilizado é a distância, contada em elos, a partir do objeto de mídia em execução, ou seja, à medida que novos objetos de mídia são executados, aqueles a uma certa distância serão compilados. Cabe ressaltar que essa distância é um dos parâmetros de configuração do Formatador.

Outra entidade envolvida no processo de apresentação é o Escalonador. À medida que novos elos causais são adicionados ao modelo, o Formatador notifica tal entidade. A partir daí, o Escalonador se cadastra como observador de tais elos para receber uma notificação quando a condição prevista nestes for satisfeita. Quando, enfim, o Escalonador receber a notificação do elo, este executará a ação por ele representada.

O Formatador tem, ainda, a responsabilidade de emitir notificações quando novos objetos de execução são adicionados ao seu modelo de execução. Cada objeto de execução representa a instância de um nó a ser exibido, especificado no documento NCL, contendo todas as suas informações, inclusive as suas características de apresentação provenientes do descritor de apresentação associado. Quando esse descritor existir, o Gerenciador de Leiaute, se necessário, cria a superfície para a exibição do conteúdo do nó.

Ainda no momento em que novos objetos de execução são adicionados, é feita uma avaliação das alternativas de objetos que podem ser resolvidas estaticamente. São avaliadas as informações do contexto de execução (preferências do usuário, poder de processamento, recursos disponíveis) para que, a partir das regras definidas no documento, seja feita a escolha do objeto de execução adequado.

Mais adiante, no momento em que o Escalonador requisita a execução de um objeto de mídia, ocorre a interação de mais um módulo do Formatador: o Gerenciador de Adaptadores para Exibidores. Esse gerenciador tem à sua disposição um conjunto de adaptadores para os exibidores que tratam determinados tipos de conteúdo suportados pelo ambiente do Formatador NCL. A partir do objeto de execução, o gerenciador seleciona o adaptador de um exibidor capaz de reproduzir o conteúdo de tal objeto.

Como exemplo de exibidores pode-se citar: XHTML, LUA, imagens estáticas (JPEG, PNG, GIF), áudio (WAVE, MPEG-1, MPEG-2) e vídeo (MPEG-1, MPEG-2).

4.2. O GEM processando um documento NCL

Como visto na Seção 4.1, o primeiro passo para a execução de um documento NCL é a sua conversão de uma especificação XML para objetos Java NCL. Essa operação consome tempo, processamento e espaço no receptor para armazenar uma biblioteca que realize a tradução do documento XML. Esses requisitos levam à elaboração de dois cenários diferentes para tratar o processo de conversão: a pré-conversão do documento e a conversão no receptor.

No primeiro cenário, a conversão ocorre no provedor de conteúdo. Nesse caso, ao invés de receber documentos NCL, os receptores processariam diretamente objetos Java serializados[2], tornando opcional a presença de uma biblioteca capaz de processar documentos XML. Esse cenário é mais indicado na presença de um receptor com recursos de processamento/memória bastante escassos. Contudo, os receptores que se basearem nesse modelo perderão a capacidade de processar diretamente documentos NCL. Eles não seriam capazes de atuar em cenários onde lhes fosse exigido, por exemplo, processar documentos NCL dinâmicos presentes na WEB via canal de retorno. Sua capacidade de processamento restringir-se-ia apenas a objetos serializados.

Na Figura 17 é ilustrado esse processo de pré-conversão do documento NCL. Nele o documento é convertido pelo provedor de conteúdo, depois é serializado e enviado para o carrossel de objetos para, em seguida, ser desserializado e processado pelos receptores.

[2] A serialização de objetos JAVA permite a codificação de um objeto e de todos os objetos por ele referenciados direta ou indiretamente em um fluxo de bytes. Permite ainda a reconstrução do "grafo" de objetos a partir de um fluxo. Este fluxo possibilita o envio por rede e o armazenamento em arquivos, por exemplo.

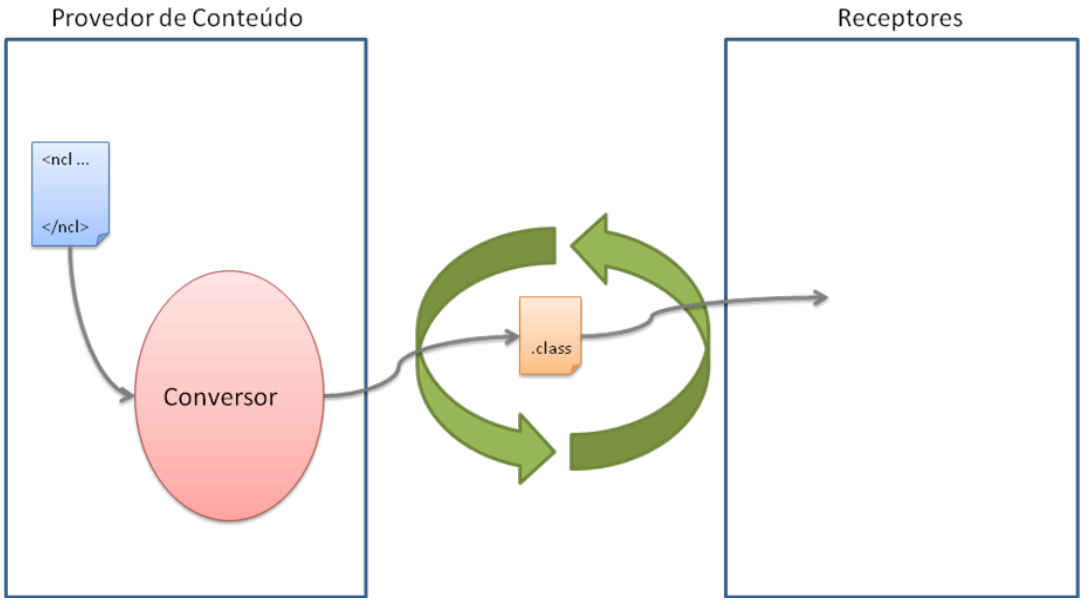

Figura 17 – Pré-processamento do documento NCL.

No segundo cenário, a conversão ocorre no receptor. Esse cenário é antagônico ao primeiro: requer um receptor com um maior poder de processamento/armazenamento e permite o processamento de documentos NCL em seu estado bruto. A Figura 18 ilustra esse processo em que o documento é enviado para o carrossel de objetos sendo convertido apenas no ambiente dos receptores.

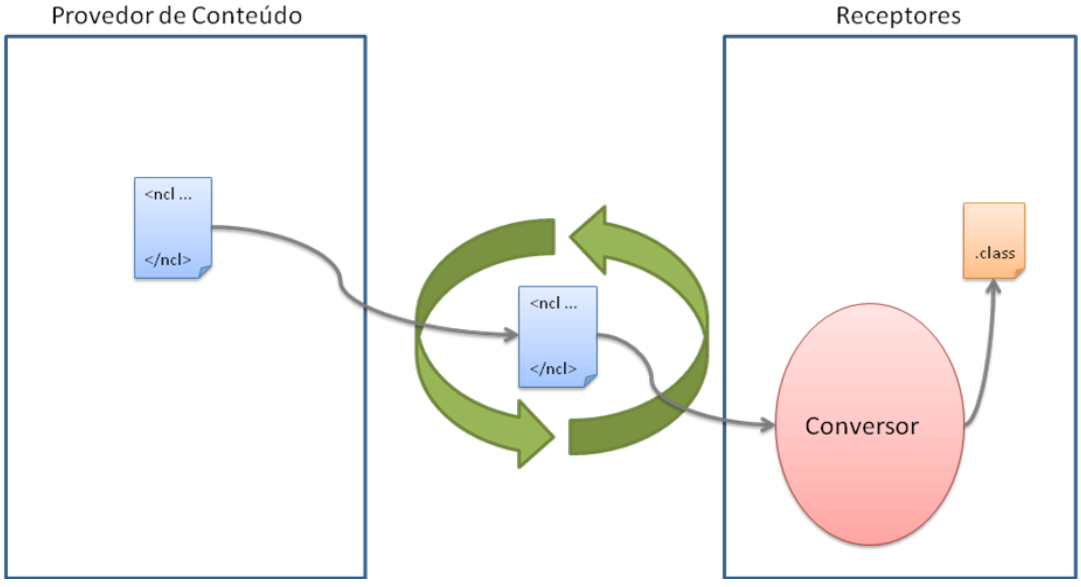

Figura 18 – Processamento do documento NCL no receptor

Essa é a forma convencional de operação do Formatador NCL. Ela requer, da parte do receptor, a presença de uma biblioteca para o processamento de arquivos XML.

4.3. A Arquitetura de Implantação

Na Seção 4.1 foi mostrado o caráter modular da arquitetura do Formatador NCL. Essa característica possibilita a realização de otimizações em seu processo de implantação. Os seguintes módulos podem ser identificados:

- Núcleo do Formatador, formado pelo Formatador e compreendendo o Escalonador, o Conversor e as Informações do Contexto;
- Gerenciador de Leiaute;
- Gerenciador de Documentos; e
- Gerenciador de Adaptadores para Exibidores. Sendo que cada exibidor por ele gerenciado constitui um módulo específico.

Cada módulo abrange um componente de software diferente. Esses componentes serão enviados para os receptores para a reprodução do documento NCL. Contudo, apenas os módulos necessários em um dado cenário deverão ser carregados.

Caso o receptor forneça a capacidade de armazenamento de aplicações, é possível otimizar-se o tempo de iniciação do Formatador ao armazenar localmente os módulos comumente utilizados. No entanto, algum mecanismo de controle de versão torna-se necessário, afim de evitar problemas de compatibilidade e suporte a atualizações dos módulos.

4.3.1. Estrutura do Sistema Baseada em Componentes

Ao modelar o Formatador NCL de forma a baseá-lo em componentes de software, estes devem compor unidades binárias desenvolvidas de forma independente que cooperem através de interfaces bem definidas (Szyperski, 2002). Têm-se, assim, as seguintes vantagens:

- Capacidade de reutilizar esses componentes em outras aplicações. Ou seja, um mesmo componente ser utilizado para TV, desktop, ferramentas de autoria ou até outro tipo de aplicação de forma independente;
- Fácil manutenção e customização desses componentes para oferecer novas funcionalidades e recursos. Ou seja, alguns componentes

podem sofrer customizações individualmente, como, por exemplo, para IPTV (Weber & Newberry, 2006);

- Possibilidade de fornecer unidades de implantação (*deployment*) independentes; e
- A linguagem de programação de cada componente não precisa ser necessariamente a mesma.

No entanto, em sistemas GEM não se pode tirar proveito de todas as vantagens citadas. Seu ambiente, por exemplo, permite apenas uma linguagem de programação. Além disso, devido ao fato do programa ter como requisito a necessidade de operar em uma plataforma com recursos limitados, nenhum framework/infra-estrutura de componentes (como, por exemplo, CORBA CCM) deve ser utilizado para a implementação dos mesmos. Porém, é válido para a elaboração do sistema o uso dos conceitos de componentes independentes de infra-estruturas comerciais adotadas (Brown & Wallnau, 1999), tornando possível uma futura adoção de tais infra-estruturas, como sugerido na Seção 6.1.

4.4. Adaptações e Otimizações no Formatador NCL

O primeiro aspecto a levar em consideração na implementação do Formatador NCL *Xlet* é que este deve utilizar apenas as bibliotecas fornecidas pelo JVM mínimo necessário para o GEM, ou seja, portar o código escrito para uma configuração de máquina virtual Java SE para uma configuração Java ME (CDC com PBP). A seguir são abordados aspectos particulares de cada componente.

4.4.1. A Pré-conversão de Documentos NCL e o Módulo Gerenciador de Documentos

O conversor de documentos NCL está contido dentro do módulo Gerenciador de Documentos. Ele é responsável por converter os documentos NCL (descritos no formato XML) em objetos Java. Esse processo, no entanto, requer a presença de uma biblioteca XML capaz de traduzir tais documentos. Num ambiente de TV Digital, a adição de tal biblioteca ocupa tempo de

transmissão e, conseqüentemente, ocasiona um maior retardo na iniciação da apresentação.

A solução proposta na Seção 4.2 prevê a criação de um ambiente de pré-conversão de documentos NCL. No entanto, de forma a não alterar a interface fornecida pelo Componente Gerenciador de Documentos e, ainda assim, permitir a pré-conversão, optou-se por adotar o seguinte processo. No ambiente de pré-conversão (no produtor de conteúdo, por exemplo), o Gerenciador de Documentos é instanciado. Em seguida, todos os documentos necessários para a apresentação são convertidos e adicionados à sua base de documentos. Ao final do processo, o componente é serializado e enviado aos receptores. Dessa forma, elimina-se a necessidade do envio da biblioteca XML para o receptor tornando, porém, o Gerenciador de Documentos incapaz de processar comandos de edição NCL que necessitem da conversão de especificações XML.

Esse cenário leva à criação de um padrão de apresentação de **Documentos NCL Orientados a Sincronização por Contexto**. Nesse padrão, o produtor de conteúdo criará vários contextos (composições ou documentos) que serão pré-convertidos e enviados para os receptores no componente Gerenciador de Documentos serializado antes do início da apresentação. No momento desejado, a emissora enviará um comando de edição para a iniciação do contexto. É importante notar que os documentos desse padrão caracterizam-se por possuir vários contextos sem elos definidos entre eles.

O exemplo mostrado na Figura 19 ilustra a transmissão de uma partida de futebol seguindo o padrão de apresentação de Documentos NCL Orientados a Sincronização por Contexto. Primeiramente, é enviado o Gerenciador de Documentos juntamente com todos os contextos envolvidos na apresentação da partida de futebol (no caso, os dois documentos com os identificadores “Futebol” e “intervalo”). A seguir, é enviado um comando de edição NCL do tipo *startDocument* para iniciar a exibição do documento “Futebol”. Durante o jogo é enviado o mesmo comando no momento em que ocorre o gol de alguma das equipes, iniciando a exibição de um contexto responsável por mostrar uma animação comemorativa. Ao longo do jogo, podem ser enviados outros comandos e pode ser acionado o início da exibição de um outro documento pré-convertido, representado na Figura 19 pelo identificador (“*id”*) “intervalo” e que representa a apresentação do intervalo de jogo.

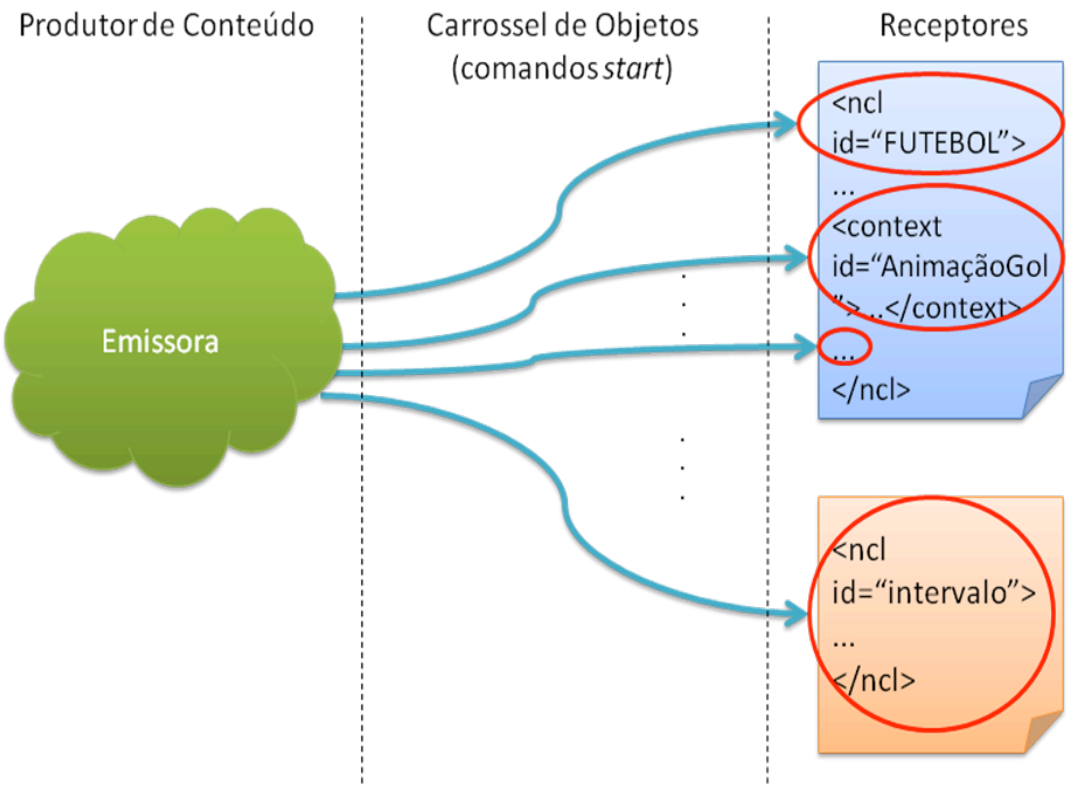

Figura 19 – Modelo de uma apresentação NCL orientada a sincronização por contexto.

Quando não houver a necessidade de pré-conversão, o componente Gerenciador de Documentos poderá ser instanciado no receptor. Este é o caso, por exemplo, de receptores que já possuam nativamente uma biblioteca para a tradução de documentos XML, ou, ainda, sistemas que possibilitem a persistência de dados.

4.4.2. Adaptações do Módulo do Núcleo do Formatador

A porta de entrada para este componente é a entidade que implementa a interface *IFormatter*. Atualmente, essa interface encontra-se especificada de acordo com o diagrama da Figura 20. A figura mostra que essa interface é responsável por:

- Processar comandos para o controle da apresentação, como *reset* (reiniciar), *close* (fechar), *startDocument* (iniciar a apresentação de um documento a partir de uma âncora), *stopDocument* (finalizar a apresentação de um documento), *pauseDocument* (pausar a apresentação de um documento) *resumeDocument* (continuar a apresentação de um documento);

- Realizar a conversão de um documento (através do método *compileDocument*) para a estrutura de execução interna do Formatador. Como já mencionado na Seção 4.1, essa conversão é feita de acordo com uma profundidade definida pelo método *setDepth*; e
- Processar os comandos de edição (que são tratados pelos métodos abaixo do método *resumeDocument* na Figura 20 e, em seguida, enviados para o Gerenciador de documentos).

Essa interface, no entanto, apresenta-se fortemente acoplada ao componente Gerenciador de Documentos. De forma a diminuir o acoplamento mencionado, foram realizadas as seguintes adaptações:

- Os eventos de edição passaram a ser enviados diretamente para o componente Gerenciador de Documentos;
- Foi criada uma interface do tipo *Listener* (de acordo com o padrão de desenho "Observador") (Gamma, 2002), permitindo ao Gerenciador de Documentos a emissão de notificações na ocorrência de um evento de edição bem sucedido;
- Foram retirados os métodos que representam os comandos de edição da interface *IFormatter*.

Figura 20 – Digrama de classe da interface *IFormatter*.

Assim, caso o Núcleo do Formatador ofereça suporte a eventos de edição basta que, internamente, alguma entidade implemente a interface representada no digrama da Figura 21 e se cadastre como *Listener* do Gerenciador de Documentos. No momento que um documento for alterado por um comando de edição todos os *Listeners* serão notificados.

Figura 21 – Diagrama da interface *NCLEditingCommandListener*.

4.4.3. Adaptações do Módulo Gerenciador de Adaptadores para Exibidores e os Adaptadores Criados

A adaptação necessária a este componente se dá ao definir a forma como a classe que implementa a interface *IFormatterPlayerAdapter* obtém a instância do adaptador do exibidor. Sempre que for requisitado um adaptador ao gerenciador, este verifica sua existência no receptor. Caso não logre sucesso, o gerenciador tentará realizar o carregamento do exibidor necessário para a apresentação através do *IGingaXletDeployer* que é detalhado na Seção 5.2.2. Essa interface abstrai do

Gerenciador de Adaptadores a localização e o protocolo para a obtenção do adaptador.

Os adaptadores criados para a validação da implementação foram o de texto, o de imagens estáticas e o *NCLet*. Esse último, em especial, é responsável por realizar a ponte com o ambiente procedural. O conteúdo de sua exibição são programas descritos em código binário Java.

4.4.4. Adaptações do Módulo Gerenciador de Leiaute

No modelo utilizado pelo Formatador NCL, a construção da interface gráfica, ou seja, os componentes gráficos onde os objetos de mídia serão exibidos, é feita pelo Gerenciador de Leiaute.

Este componente fornece a implementação para duas entidades: o *IFormatterLayout* que, de fato, gerencia o leiaute; e o *IFormatterRegion* que contém a informação do elemento da interface gráfica em si.

A Figura 22 mostra o diagrama de classes do módulo de leiaute, introduzido pelo Formatador, apresentando uma modificação: a adição do método *createRegion* na interface *IFormatterLayout*. Dessa forma, as regiões serão criadas apenas a partir de seu gerenciador, permitindo que este mantenha uma referência para todas aquelas criadas.

Diferente do ambiente do *desktop*, o perfil PBP da configuração CDC não oferece suporte à biblioteca gráfica SWING (Robinson & Vorobiev, 2003), apenas AWT (Geary & McClellan, 1997). Nesse perfil, é permitida apenas a criação de uma única janela (*Frame*). Com isso, a implementação da classe *IFormatterRegion*, responsável por criar os elementos de interface gráfica, foi alterada para atender tais requisitos.

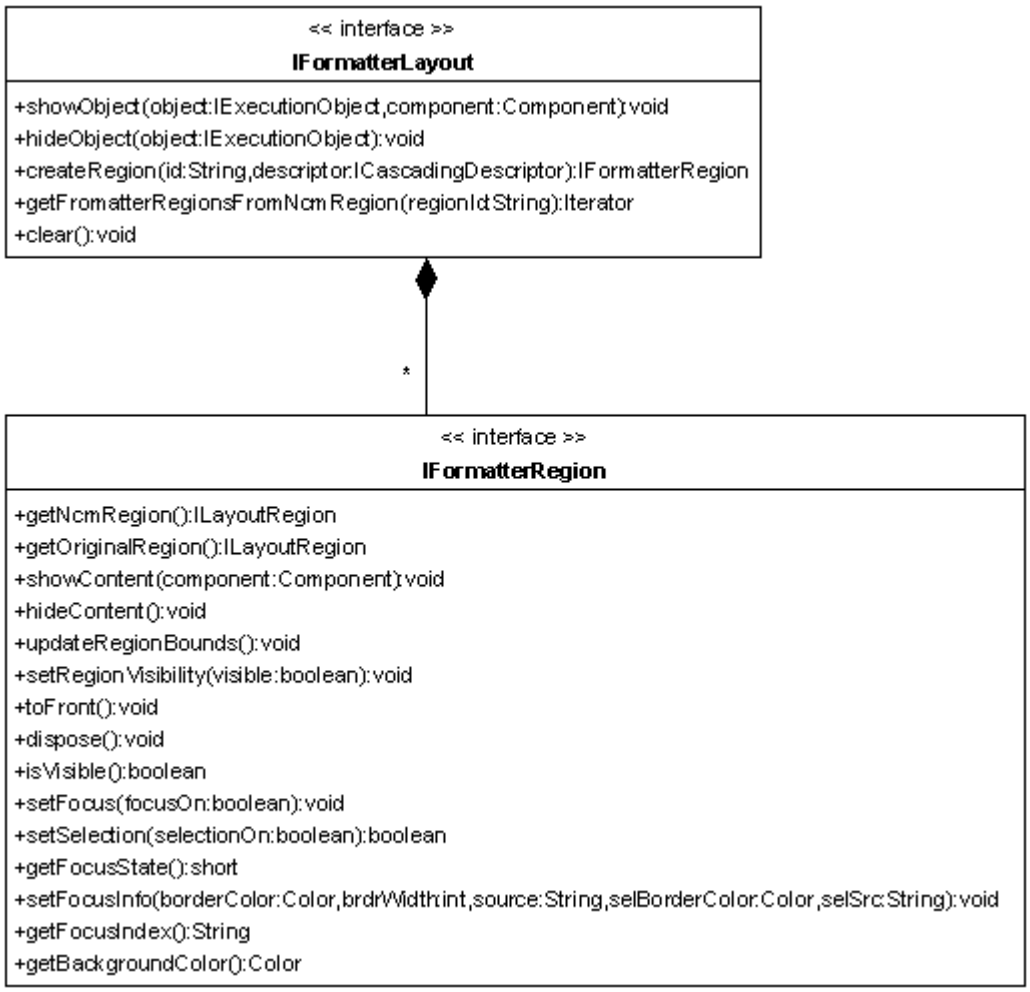

Figura 22 – Diagrama de classes do modelo de leiaute do Formatador NCL.