PUC-Rio - Departamento de Informática

Ginga-NCL com objetos de mídia SSML embutidos

Relatório Técnico: Arquitetura

Rafael Diniz
Matrícula: 1312398

5 de agosto de 2014

Sumário

1 Introdução

1.1 Propósito

Este documento especifica os aspectos arquiteturais do projeto do sistema, fornecendo aos desenvolvedores as informações necessárias para a construção do sistema. Como este projeto foi desenvolvido no escopo da disciplina INF2102 - Projeto Final de Programação, este documento também fornece material para que o professor possa avaliar o trabalho desenvolvido.

1.2 Público Alvo

Este documento se destina aos engenheiros de software e testadores interessados no sistema.

1.3 Escopo

Este documento descreve os casos de uso, modelo de arquitetura adotada e componentes do projeto: Ginga-NCL com objetos de mídia SSML embutidos.

1.4 Definições, Acrônimos e Abreviações

Ginga	Middleware padrão do Sistema Brasileiro de TV Digital
Ginga-NCL	Subsistema Declarativo do middleware Ginga
NCL	Nested Context Language
XML	eXtensible Markup Language
SSML	Speech Synthesis Markup Language

1.5 Referências

- [1] Soares, L. F. G., Rodrigues, R. F., Moreno, M. F. (2007). Ginga-NCL: the declarative environment of the Brazilian digital TV system. Journal of the Brazilian Computer Society, 12(4), 37-46.
- [2] Soares, L. F. G. S. (2009). Programando em NCL 3.0: desenvolvimento de aplicaçoes para middleware Ginga: TV digital e Web. Elsevier.
- [3] W3C Recommendation (2010). Speech Synthesis Markup Language (SSML) Version 1.1.
- [4] Garrido, J., Bofias, E., Laplaza, Y., Marquina, M., Aylett, M., & Pidcock, C. (2008). The Cerevoice speech synthesiser. Actas de las V Jornadas de Tecnología del Habla (Bilbao).
- [5] Documento de requisitos do projeto de nome *requisitos.pdf*, presente no mesmo diretório deste documento.

1.6 Visão geral do documento

A seção 2 apresenta a modelagem dos casos de uso relacionados aos requisitos levantados no documento de requisitos[5], assim como os atores envolvidos. Na seção 3 é descrito o modelo de arquitetura adotado, e sua adequação à solução proposta, bem como os componentes do sistema. Na seção 4 estão os diagramas de classes do que deverá ser alterado no Ginga-NCL pelo presente trabalho, e na seção 5 constam os diagramas de sequência.

2 Modelagem dos casos de uso

Esta seção traz o diagrama de caso de uso baseado nos requisitos já levantados[5]e os atores envolvidos.

2.1 Diagrama

A Figura 1 apresenta o diagrama de casos de uso.

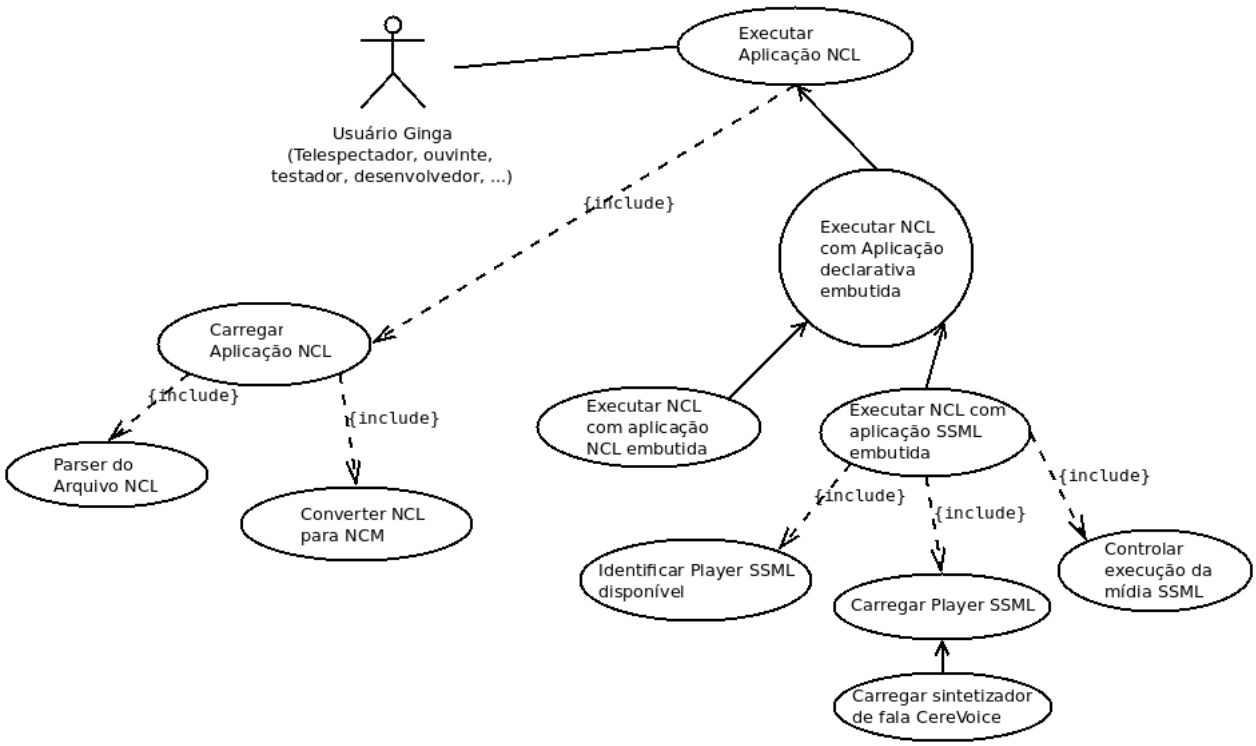

Figura 1. Diagrama de casos de uso.

2.2 Atores

Do ponto de vista do Ginga-NCL em execução, os atores são os espectadores de TV e rádio digitais. Também podem ser considerados como atores os testadores, desenvolvedores e designers de aplicativos NCL quando executam o aplicativo em desenvolvimento com o objetivo de testá-lo.

3 Arquitetura

Esta seção descreve o modelo da implementação de referência do Ginga-NCL como um todo, descrevendo resumidamente cada um dos componentes. Neste projeto será utilizada a versão em C++ da implementação de referência disponível em *http://git.telemidia.puc-rio.br/*. Nesta seção, também são evidenciados os principais pontos que a implementação foi alterada ou onde foram adicionados novos elementos pelo presente projeto de forma a permitir que o Ginga-NCL execute apropriadamente documentos SSML embutidos em documentos NCL, segundo os requisitos definidos.

A Figura 2 evidencia os principais componentes da arquitetura do Ginga-NCL e do Ginga Common Core. O Ginga-NCL é a máquina de apresentação responsável por executar documentos NCL, enquanto o Ginga Common Core é o responsável por suprir funcionalidades básicas de exibição de mídias, gerenciamento de contexto, dentre outras, as quais serão úteis tanto para a máquina Ginga-NCL como para outro subsistema que possa ser adicionado, como o Ginga-J, para suporte à linguagem Java (tópico não abordado nesse trabalho).

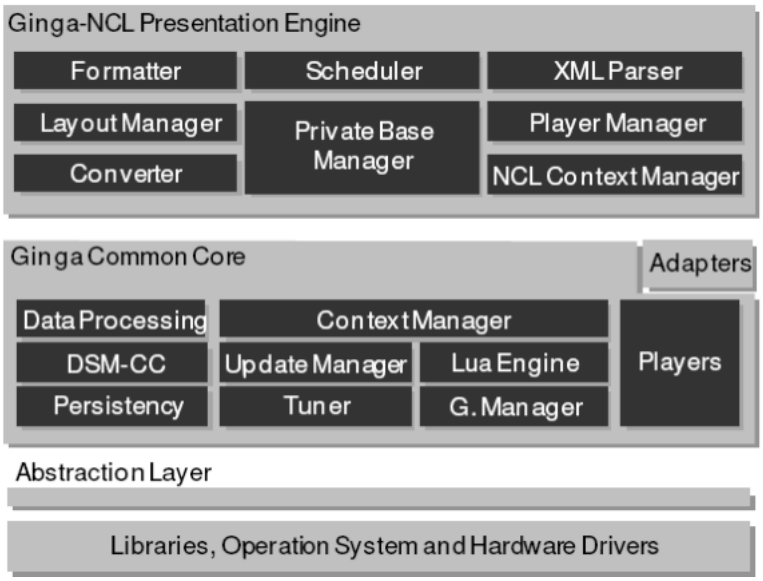

Figura 2: Arquitetura do Ginga-NCL e Ginga Common Core.

Os componentes do Ginga-NCL apresentados na Figura 2 são descritos na de forma geral na seção 4, sendo que os componentes modificados no escopo deste trabalho são apresentados com detalhes.

4 Componentes do Ginga

O Ginga, conforme discutido em [1], é subdividido em dois subsistemas principais: Ginga-NCL e Ginga Common-Core. O Ginga Common-Core que provê funcionalidades comuns que podem utilizados por outros subsistemas, como para o caso do Ginga-J. Neste trabalho serão tratados o Ginga-NCL e o Ginga Common-Core, os quais serão adaptados para permitir a execução de objetos de mídia SSML [3], conforme discutido no documento [5]. Nas próximas seções serão descritos os principais componentes da arquitetura do Ginga-NCL e Ginga Common Core. Atenção especial será dada àqueles componentes que sofrerão mudanças para a realização deste trabalho.

4.1 Ginga-NCL

4.1.1 Formatter

O Formatter tem com principais funcionalidades receber e controlar a exibição de aplicações multimídia escritas em NCL. As aplicações são entregues ao Formatter pelo Ginga Common-Core.

O Formatter requisita ao componentes XMLParser e Converter para traduzir o código fonte da aplicação NCL em uma estrutura de dados interna necessária para a apresentação do documento. A partir de então o componente Scheduler é iniciado para orquestrar a apresentação da aplicação.

4.1.2 Layout Manager

Este componente é responsável por mapear todas as regiões definidas em uma aplicação NCL para sua respectiva área do Canvas. Adicionalmente, esse Canvas também pode estar em um outro dispositivo, permitindo ao Ginga-NCL suportar múltiplos dispositivo de exibição.

4.1.3 Converter

O componente Converter é responsável por traduzir o formato XML de documentos NCL, lido através do XML Parser, em uma estrutura de dados interna ao Ginga-NCL, denominada Modelo de Apresentação que servirá como base para o Scheduler orquestrar a aplicação.

4.1.4 Scheduler

O componente Scheduler é responsável por orquestrar a apresentação do documento NCL. A pré-busca de objetos de mídia, a avaliação de condições nos elos causais e o agendamento de ações que guiam o fluxo da apresentação estão entre as funções deste módulo.

O Scheduler também é responsável por comandar o componente Player Manager, responsável por instanciar o Player apropriado, de acordo com o conteúdo da mídia que será exibida em um determinado momento.

4.1.5 Private Base Manager

A máquina de apresentação Ginga-NCL (Scheduler) trata de um conjunto de aplicações que estão organizadas em uma estrutura de dados denominada Base Privada (ou Private Base). O componente Private Base Manager é responsável por receber comandos de edição ao vivo sobre documentos NCL e manter os documentos NCL que estão sendo apresentados.

4.1.6 XML Parser

XML Parser é o componente responsável por ler e gravar documentos NCL em um formato XML.

4.1.7 Player Manager

O Player Manager é o componente responsável por gerenciar os diversos Players dos diversos tipos de conteúdo de mídia que o Ginga-NCL é capaz de reproduzir.

Na implementação tomada como base, o Player Manager é configurável através de um arquivo que associa cada tipo MIME a uma classe C++ que implementa o Player Adapter para aquele tipo de mídia. Assim, uma nova entrada nesse arquivo deve ser realizada associando o tipo MIME do SSML às seus respectivos Players Adapter.

4.1.8 NCL Context Manager

Responsável por, juntamente com informações obtidas do Context Manager, disponibilizar variáveis globais do sistema e do perfil do usuário possibilitando adaptar o conteúdo a ser apresentado conforme os valores destas variáveis.

4.2 Ginga Common Core

4.2.1 Data Processing

Responsáveis por oferecer suporte e obter os dados que chegam através de um carrossel de dados, seja via TV ou rádio digital.

4.2.2 Context Manager

É responsável por obter informações das características da plataforma e perfil do usuário e atualizar variáveis globais da aplicações NCL, permitindo, dentre outras coisas, que a apresentação seja adatada à plataforma ou perfil do usuário do serviço.

O componente NCL Context Manager consulta o Context Manager para manter as variáveis na máquina de apresentação atualizada e vice-versa.

4.2.3 Update Manager

O Update Manager é o componente responsável por gerenciar versões do Ginga-NCL bem como dos seus componentes individuais. Não está no escopo deste trabalho discutí-lo detalhadamente.

4.2.4 Tuner

Responsável por oferecer uma API para o gerenciamento da sintonia de canais de TV ou rádio. Não está no escopo deste trabalho utilizar ou modificar este componente.

4.2.5 Lua Engine

Além da apresentação de objetos de mídia pré-definidos (imagem, áudio, vídeo, texto, etc.), o Ginga-NCL também possibilita a execução de objetos de mídia imperativos e declarativos.

No que se refere à apresentação de objetos de mídia imperativos, o Ginga-NCL suporta código Lua por padrão. O componente Lua Engine é responsável por fazer essa comunicação de forma apropriada com a Máquina de Apresentação.

4.2.6 Graphics Manager Component

Responsável por controlar o modelo gráfico definido por uma determinada plataforma de exibição. Não será modificado pelo presente trabalho.

4.2.7 Players

Cada componente Player é responsável por reproduzir determinados tipos de conteúdo de mídia. O Player Manager é responsável por escolher qual Player é utilizado para tocar determinado tipo de mídia.

O Ginga-NCL foi desenvolvido de forma a facilitar a integração de novos tipos de mídia. Para isso, cada Player deve implementar uma API comum que permite a comunicação entre ele e a máquina de apresentação (componente Scheduler).

Os Players são responsáveis por notificar a máquina de apresentação sobre eventos que ocorrem em um determinado objeto de mídia, tais como a ocorrência de alguma âncora, início ou fim da apresentação, seleção ou alteração de alguma propriedade do nó de mídia. Players que não seguem a especificação da API Ginga devem se utilizar de componentes Adapters para fazer essa mediação.

Com relação ao projeto desenvolvido no escopo da disciplina INF2102, um novo Player será adicionado à implementação do Ginga-NCL: o CereVoice [4], responsável por tocar apropriadamente objetos de mídia SSML [3]. Um novo Adapter também será necessário para manter a compatibilidade com a API Ginga.

4.2.8 Adapters

Os Adapters, na arquitetura do Ginga, são componentes de software responsáveis por intermediar a comunicação entre o Ginga e os Players, de forma que seja possível utilizar Player que não implementem a API determinada pelo Ginga. No que se refere a este projeto, como será visto, será definido um novo Adapter para se comunicar com o CereVoice, player responsável por tocar objetos de mídia declarativos SSML.

5 Diagramas de classe

Esta seção apresenta os diagramas de classe relacionado com as mudanças que devem ser realizadas no Ginga-NCL.

5.1 SSMLPlayerAdapter

Uma das principais alterações necessárias para permitir ao Ginga-NCL tocar uma mídia SSML é implementar um novo Adapter. No nosso caso o SSMLPlayerAdapter. Um novo Adapter deve implementar a interface FormatterPlayerAdapter, já definida no Ginga-NCL.

A Figura 3 apresenta o diagrama de classes que será discutido nesta seção.

Figura 3. Diagrama de classes com a classe SSMLPlayerAdapter.

5.2 SSMLPlayer

Outra classe que também deve ser implementada por este projeto é a classe que representa o player SSML, denominada SSMLPlayer. Assim como para o Adapter, a interface Player, já definida pelo Ginga-NCL, deverá ser implementada pelo SSMLPlayer.

A Figura 4 evidencia esse diagrama de classes.

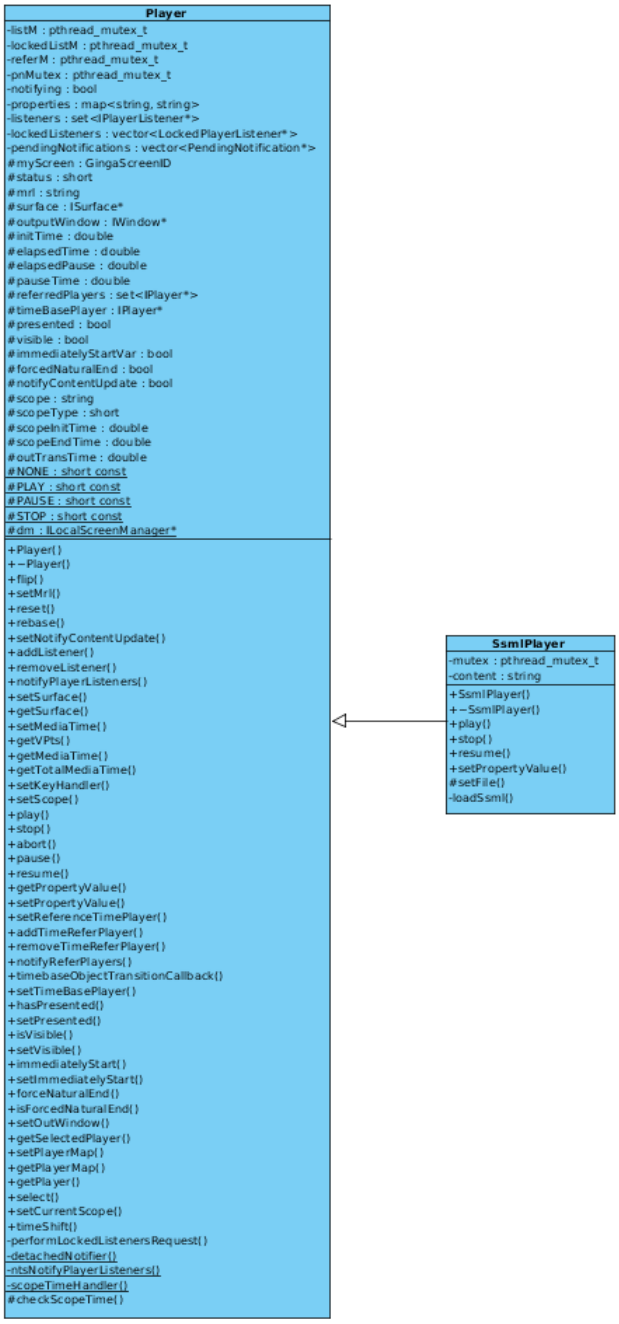

Figura 4. Diagrama de classes com a classe SSMLPlayer.

6 Diagramas de sequência

Esta seção apresenta os diagramas de sequência mais importantes relativos às mudanças que serão necessárias para a implementação do projeto.

6.1 Identificar e Carregar Player SSML

Este diagrama de sequência (Figura 5) está relacionado com os Casos de Uso "Identificar Player SSML disponível" e "Carregar Player SSML".

Conforme discutido na seção sobre a arquitetura do Ginga-NCL, o componente Scheduler (classe FormatterScheduler) é o responsável por orquestrar a apresentação do documento NCL. Ele é o responsável por manter a apresentação conforme o autor do documento NCL especificou. Para isso, deve chamar o PlayerAdapterManager quando surgir a necessidade de tocar uma determinada mídia, o qual irá identificar, em tempo de execução, qual o adaptador correto a ser criado para uma determinada mídia. Através da mensagem getPlayerClass o PlayerAdapterManager descobre qual o adaptador

correto a ser criado. Neste trabalho só estamos interessados em discutir quando tal adaptador é o SSMLPlayerAdapter. O SSMLPlayerAdapter é criado quando o tipo da mídia a ser tocada é um documento SSML.

Ao FormatterScheduler também cabe a responsabilidade de informar ao SSMLPlayerAdapter quando um determinado evento ocorre, seja pela ação do usuário ou porque o autor da NCL assim especificou. Ainda mais, cabe também ao FormatterScheduler realizar mecanismos de pré-busca, fazendo com que toda o carregamento da mídia seja realizado com antecedência à ocorrência de um determinado evento, não prejudicando assim o sincronismo da aplicação. Assim sendo, no momento correto, o FormatterScheduler deve então chamar o método prepare do SSMLPlayerAdapter o qual irá instanciar o player SSML, deixando-o apto a receber comandos de start, stop, abort ou set.

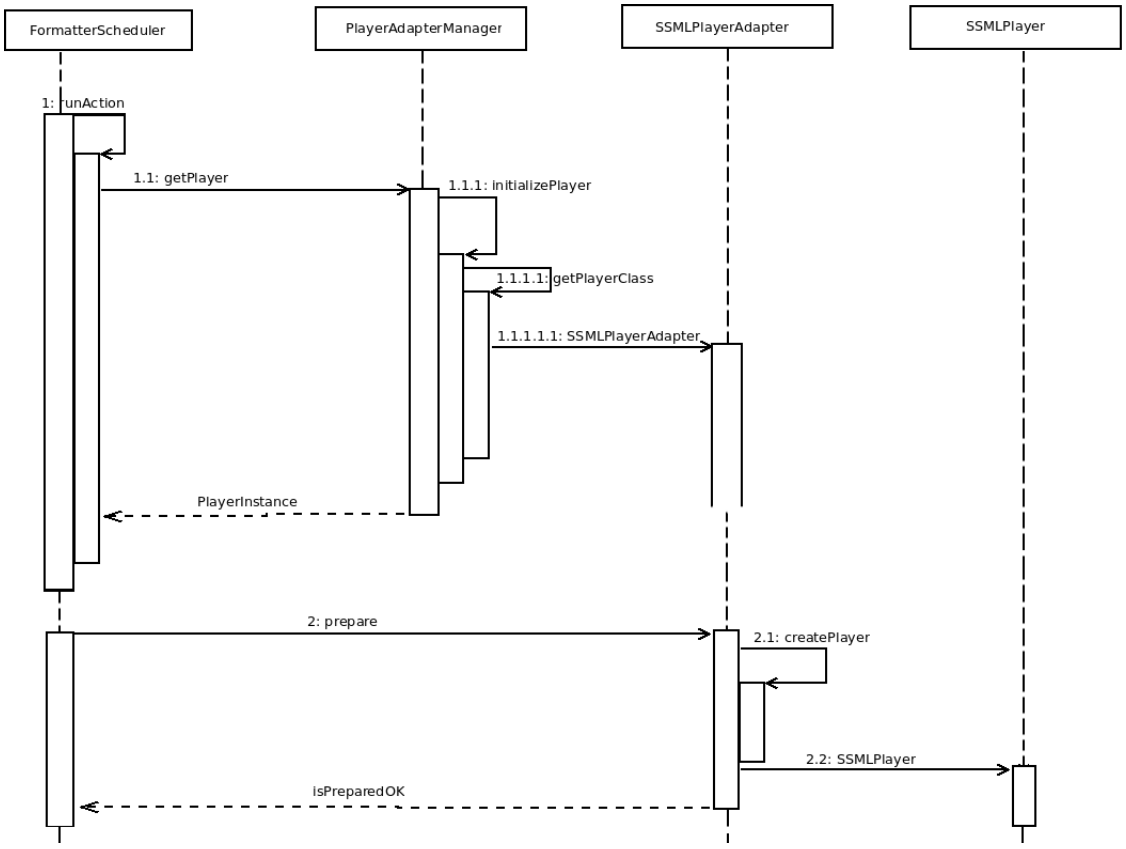

Figura 5. Diagrama de sequência da identificação e carregamento do Player SSML.

6.2 Controlar Exibição de Mídia SSML

O caso de uso "Controlar exibição de mídia SSML" inclui, na verdade, vários comandos que podem ser executados pelo FormatterScheduler sobre o player, sendo eles: como chamar comandos de start, stop, pause ou abort; mudar propriedades do player, através de comandos set; como também diversos eventos que podem ser notificados do player para o FormatterScheduler, tais como início ou fim da ocorrência de uma âncora e eventos de seleção sobre determinada âncora.

Por simplificação, e pela similaridade entre os diagramas, as figuras 6 e 7 só mostram os diagramas de sequência referentes ao comando de play e stop das mídias SSML, respectivamente.

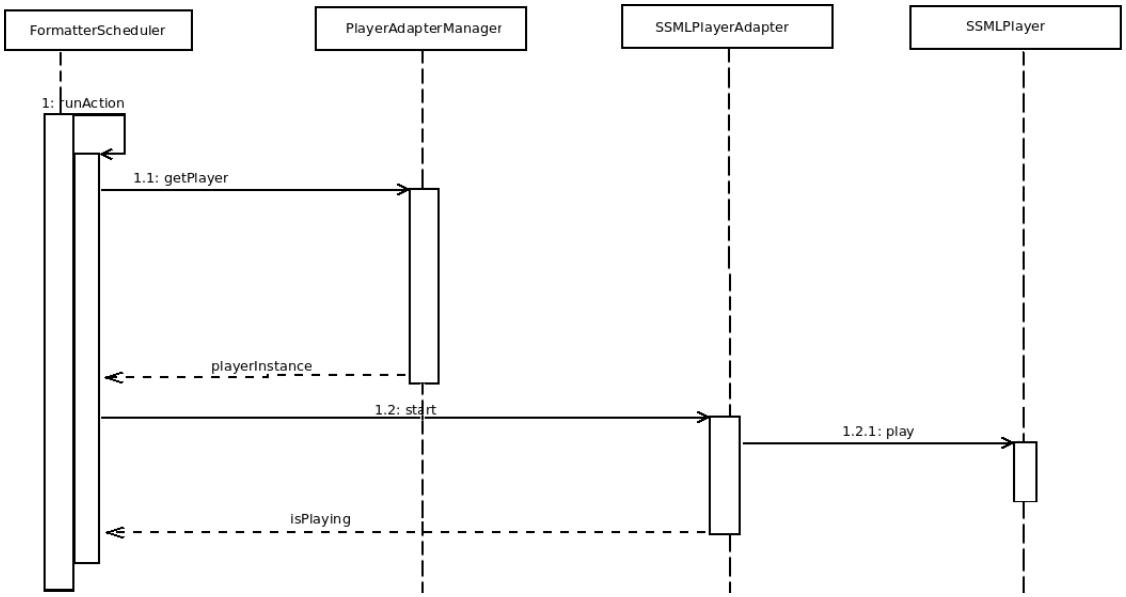

Figura 6. Diagrama de sequência do *start* de uma mídia SSML.

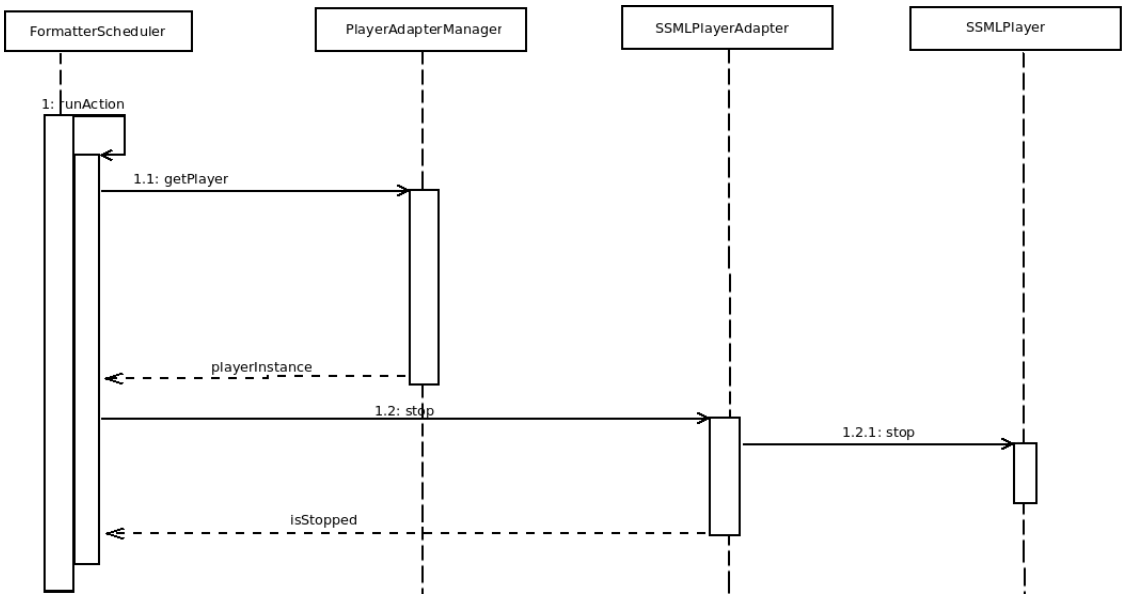

Figura 7. Diagrama de sequência do *stop* de uma mídia SSML.