Uma Arquitetura para Experimentação de Novos Serviços na Plataforma Orientada a Aplicações da TV 3.0

Joel dos Santos
Lab. MultiSenS, Cefet/RJ
Rio de Janeiro, RJ, Brasil
joel.santos@cefet-rj.br

Rodrigo da Mota Sodré
Lab. MídiaCom, UFF
Niterói, RJ, Brasil
rmsodre@id.uff.br

Luís Claudio Ribeiro Junior
Lab. MultiSenS, Cefet/RJ
Rio de Janeiro, RJ, Brasil
luis.junior.1@aluno.cefet-rj.br

Glauco Amorim
Lab. MultiSenS, Cefet/RJ
Rio de Janeiro, RJ, Brasil
glauco.amorim@cefet-rj.br

Débora C. Muchaluat-Saade
Lab. MídiaCom, UFF
Niterói, RJ, Brasil
debora@midiacom.uff.br

ABSTRACT

TV 3.0 brings several new functionalities to the Brazilian DTV system. Among these functionalities, it is possible to highlight the Application-Oriented Platform (AoP), execution of Ginga applications, viewer profile creation, and connection with smart devices using a local network. Given the wide spectrum of functionalities present in TV 3.0, this work proposes an architecture for experimenting with TV 3.0 AoP services. This architecture is designed to be extensible such that developers can easily create/extend its functionalities. To indicate its usefulness and versatility, this work presents two use cases implemented over this proposed architecture.

KEYWORDS

TV 3.0 AoP, TV 3.0 experimentation, Multisensory, Virtual Reality, SE Presentation Engine, NCL360

1 INTRODUÇÃO

O Brasil está em processo de definição de uma nova geração para o Sistema de Televisão Digital (SBTVD) brasileiro, chamada de TV 3.0 ou DTV+. Os padrões de TV 3.0 deverão estar disponíveis para a sociedade ainda em 2025[1]. A TV 3.0 traz uma série de novas funcionalidades, incluindo uma plataforma orientada a aplicações (AoP - *Application Oriented Platform*), que norteará o funcionamento de todo o sistema. A AoP implementa a Camada de Codificação de Aplicações [6] proposta pela arquitetura da TV 3.0, ilustrada na Figura 1. Dentre suas funcionalidades, destacam-se a execução de aplicações Ginga NCL e HTML5, o suporte a criação de perfis de telespectadores, a comunicação com dispositivos de segunda tela e outros dispositivos inteligentes presentes na rede doméstica do receptor.

A implementação de referência[2] do *middleware* da TV 2.0, que vem sendo utilizada como plataforma para extensão e teste de evoluções que foram incorporadas à TV 2.5 [3] e que serviram para prototipar algumas das soluções propostas e integradas à TV 3.0 [4, 5], é um software desenvolvido há algum tempo e de difícil manutenção. Com as novas tecnologias e recursos ofertados pelo novo sistema, é imprescindível uma plataforma aberta que permita o desenvolvimento, experimentação e testes de novos componentes ou aplicações, facilitando sua evolução, disseminação e adoção em larga escala.

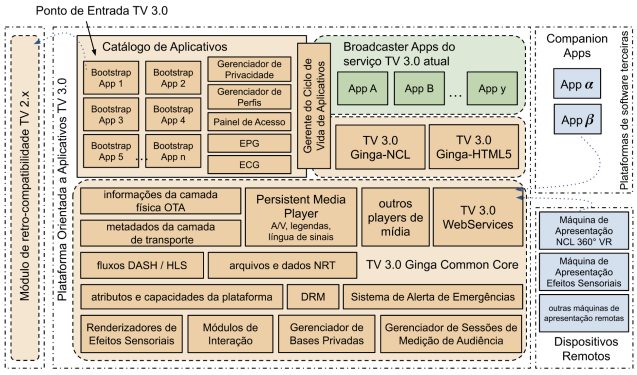

Figure 1: Arquitetura da Plataforma Orientada a Aplicações da TV 3.0. Fonte: [6]

Diante desse cenário, este trabalho propõe uma nova plataforma para experimentação de novos serviços da TV 3.0 AoP. A presente proposta já foi utilizada em dois casos de uso distintos, demonstrando sua utilidade e versatilidade. Um dos casos de uso é o suporte a aplicações de realidade virtual (RV), incluindo cenas 360 graus interativas especificadas em NCL360 [7] e consumidas através da integração com o player Guaraná que roda em um HMD (*Head Monted Display*) conectado ao receptor da TV 3.0 através da rede doméstica. O outro caso de uso é a implementação da *Sensory Effect Presentation Engine* (máquina de execução de efeitos sensoriais), como um componente externo ao receptor conectado à rede doméstica do telespectador.

O restante do artigo está estruturado da seguinte forma. A Seção 2 apresenta uma descrição conceitual da TV 3.0 AoP. A Seção 3 discute os trabalhos relacionados que inspiraram esta proposta. A Seção 4 apresenta a plataforma proposta em detalhes. A Seção 5 discute os dois casos de uso integrando a implementação existente

[1] https://forumsbtvd.org.br/tv3_0/
[2] https://github.com/TeleMidia/ginga

In: VIII Workshop Futuro da TV Digital Interativa (WTVDI 2025). Anais Estendidos do XXXI Simpósio Brasileiro de Sistemas Multimídia e Web (WTVDI'2025). Rio de Janeiro/RJ, Brazil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2596-1683

do Guaraná e da *Sensory Effect Presentation Engine*, como proposto no projeto de norma da TV 3.0. Finalmente, a Seção 6 conclui o trabalho sugerindo desdobramentos futuros.

2 TV 3.0 AOP

Na TV 3.0, a Camada de Codificação de Aplicações evolui para a especificação de uma plataforma abrangente de software [6]. Esta especificação considera tanto componentes de *back-end*, quanto de *front-end* para o ambiente de TV. Sendo assim, a Plataforma Orientada a Aplicações da TV 3.0 (TV 3.0 AoP) abrange desde a navegação entre serviços de TV, escolha de perfil de telespectador, comunicação com dispositivos remotos e execução de efeitos sensoriais, como ilustrado na Figura 1.

Como pode ser observado na Figura 1, a AoP possui diferentes componentes associados às mais diversas funcionalidades previstas para a TV 3.0. Em particular, destaca-se o *Catálogo de Aplicativos* que apresenta ao telespectador uma lista de aplicações disponíveis, considerando os serviços de radiodifusão presentes num determinado local. O acesso a esses serviços é feito por meio das aplicações iniciais (*Bootstrap Applications*) instaladas automaticamente após a varredura do sinal. A partir destas aplicações, todo o conteúdo transmitido por um radiodifusor é apresentado na TV. Outros componentes que merecem destaque são o *TV 3.0 WebServices* que, assim como sua versão na TV 2.5, permite o acesso a informações da AoP para aplicações locais (no receptor) ou na rede local, o *Gerenciador de Privacidade*, para controle de acesso a propriedades do perfil do telespectador e a possibilidade de utilização de dispositivos remotos conectados à AoP.

Todo este cenário apresentado acima, com a especificação de diferentes componentes para inclusão de funcionalidades na AoP, resulta de pesquisas realizadas pela comunidade, e comumente possuem suas próprias implementações. Em geral, tais implementações são baseadas na implementação de referência disponível do componente Ginga-NCL para TV 2.0. Por se tratar de apenas parte de uma plataforma mais ampla, essa abordagem dificulta a integração entre as diferentes implementações e, portanto, a oferta de um ambiente para experimentação com o novo padrão.

3 TRABALHOS RELACIONADOS

Com a crescente adoção das Smart TVs, testar aplicações desenvolvidas para esses dispositivos é essencial para melhorar a qualidade de experiência do usuário. O artigo [1] analisa as características específicas das aplicações de Smart TV, apresenta os pontos fundamentais para a realização dos testes, identifica os principais desafios enfrentados e propõe soluções viáveis para superá-los.

As diferenças entre aplicações de Smart TV e aplicações web tradicionais se concentram principalmente na forma de interação (controle remoto em vez de mouse ou toque), na navegação direcional (baseada em foco), na variedade de plataformas e dispositivos, e nas limitações técnicas (como recursos de hardware e desempenho). Os autores destacam três pontos essenciais para um teste eficaz: modelo de navegação baseado em foco, que representa como o usuário navega usando as setas do controle remoto; geração automática de testes, a partir do modelo de navegação, para simular os fluxos reais de uso; e infraestrutura de execução de testes adaptada a diferentes dispositivos e contextos. Além disso, os principais desafios identificados incluem: fragmentação de plataformas e dispositivos, ausência de ferramentas automatizadas específicas para Smart TVs, dificuldade em observar e verificar comportamentos corretos em tempo de execução e limitações de acesso ao código-fonte em alguns contextos. Como soluções, os autores sugerem: uso de abstrações baseadas em foco para representar interações, estratégias de modelagem e geração de testes automatizados, ferramentas para coleta de dados em tempo real e desenvolvimento de repositórios de testes reutilizáveis.

A arquitetura proposta neste trabalho possibilita a modelagem e implementação de testes automatizados, facilita a construção de ferramentas para coleta em tempo real e o desenvolvimento de repositórios de testes reutilizáveis.

Os autores em [2] propõem uma estrutura automatizada para teste de aplicações de Smart TV, baseada na separação entre dois modelos: o modelo de navegação e o modelo funcional. Essa abordagem foi projetada para lidar com os desafios específicos do ambiente de Smart TVs, como a interação via controle remoto (navegação direcional) e a escassez de ferramentas especializadas. A proposta se baseia na separação dos modelos e na implementação de um Framework automatizado. O modelo de navegação representa a lógica de transição entre telas e componentes da interface, considerando a navegação por setas (cima, baixo, esquerda, direita). Já, o modelo funcional representa os eventos e ações dentro de cada componente, como pressionar "OK" ou enviar dados. O framework foi implementado com base em tecnologias web e usa a linguagem JavaScript para instrumentação. Uma interface de simulação do controle remoto permite executar os testes em navegadores ou dispositivos reais. O Framework possibilita gerar casos de teste automaticamente a partir dos dois modelos e executar os testes em um ambiente instrumentado de Smart TV ou emulador, capturando falhas e comportamentos inesperados. Os autores validaram a abordagem com testes em aplicações reais de Smart TV e demonstraram que a separação dos modelos permite uma cobertura mais sistemática das interações e reduz o esforço manual necessário na criação dos testes.

É possível verificar que a arquitetura proposta em [2] é baseada na separação dos componentes, possibiltando uma construção mais modular e facilitando a realização de testes de cada componente.

Já em [7], os autores propõem uma arquitetura para possibilitar a integração de tecnologias de VR em apresentações multimídia interativas na TV. A arquitetura proposta se baseia no uso de dispositivos HMD como uma "segunda tela" da aplicação na TV. Desta forma, enquanto a TV apresenta o conteúdo tradicional de uma aplicação, usuários com HMD podem assistir a um formato imersivo como um conteúdo adicional.

O controle da sincronização entre o conteúdo do programa e a cena 360 (o termo *cena 360* denota uma cena imersiva criada a partir de vídeos 360^o interativos) é feito pelo dispositivo principal (*e.g.*, TV). Como sistema interativo de TV foi utilizado o *middleware* Ginga e sua linguagem NCL. Já o *player* de cena 360 proposto, chamado *Guaraná*, é uma aplicação para HMDs que foi desenvolvida em Unity. *Guaraná* interage com o *middleware* Ginga recebendo uma cena 360 a ser rendenizada, bem como comandos para ativação dos elementos de cena (*e.g.*, iniciar um vídeo), e enviando ao Ginga eventos (*e.g.*, interação do usuário) que tenham ocorrido durante a apresentação.

Apesar de um passo importante em facilitar a utilização de um novo dispositivo como segunda tela, a implementação realizada em [7] é complexa e pontual, não abarcando os diferentes componentes necessários para implementar plenamente a funcionalidade de comunicação com dispositivos remotos prevista na TV 3.0. A arquitetura proposta neste trabalho, por permitir a implementação separada dos diferentes componentes, tem como um dos objetivos facilitar a introdução destes novos dispositivos.

4 PLATAFORMA PROPOSTA

A implementação de referência do Ginga-NCL disponível para a comunidade acadêmica[3] encontra-se em descompasso com as normas de TV digital, ainda mais considerando os avanços recentes da TV 3.0. Ainda, cabe destacar que tal código tem seu escopo limitado à execução de aplicações NCL. Assim, incluir novas funcionalidades previstas nos novos padrões, bem como alterar funcionalidades existentes, requerem um grande esforço.

Por outro lado, dada a complexidade da TV 3.0 AoP, conforme apresentado na Seção 2, a implementação de todos os seus componentes em um único código também é uma tarefa complexa.

Este artigo, portanto, propõe uma nova arquitetura de microsserviços para a implementação de componentes de *front-end* e *back-end* aderentes às especificações contidas na camada de codificação de aplicações da TV 3.0. O racional por trás dessa proposta é permitir que diferentes componentes representando funcionalidades da AoP possam ser integrados mesmo partindo de códigos diferentes. Com isso, torna-se possível obter uma base para teste de aplicações e experimentação de funcionalidades da TV 3.0.

A Figura 2 ilustra a arquitetura distribuída proposta. A arquiteutura é baseada num padrão publish/subscribe com um broker centralizado. Conforme ilustrado na figura, diferentes componentes da AoP são implementados independentemente. A implementação desses componentes pode ainda seguir diferentes abordagens e rodar em plataformas distintas.

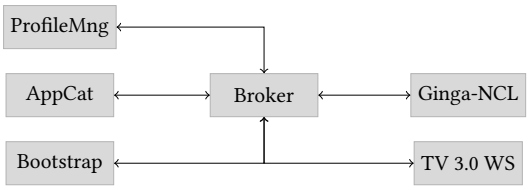

Figure 2: Arquitetura distribuída da AoP

A Figura 2 apresenta uma arquitetura parcial da AoP, apesar de suficiente para demonstrar o funcionamento da arquitetura proposta. Nela são representados os componentes *Application Catalog* (`AppCat`) responsável por apresentar o catálogo de aplicações das emissoras transmitidas via sinal de TV. O componente *Bootstrap App* (`Bootstrap`) representa uma destas aplicações das emissoras. O componente *Profile Manager* (`ProfileMng`) gerencia os perfis de telespectadores cadastrados na AoP e permite a identificação do telespectador atual. O componente *TV 3.0 WebServices* (`TV 3.0 WS`) exporta APIs para acesso tanto de aplicações locais (internas à AoP) quanto externas (na rede local da AoP) acessarem informações como conteúdos de aplicações Ginga, dados sobre o serviço em uso, alertas, etc. Por fim, o componente *TV 3.0 Ginga-NCL* (`Ginga-NCL`) é responsável por interpretar aplicações interativas veiculadas pelas emissoras e codificadas usando a linguagem NCL.

Para garantir a correta integração entre componentes, a comunicação entre eles é realizada por meio de um *broker*. Esse *broker* é responsável por armazenar descentralizadamente o estado da AoP. Assim, cada componente acessa a parte do estado relevante para seu funcionamento e é notificado pelo *broker* quando modificações no estado são realizadas.

A implementação apresentada neste artigo utiliza um *broker* MQTT cuja estrutura de tópicos reflete o estado da AoP. A Tabela 1 apresenta os tópicos, bem como o formato e descrição do conteúdo de cada tópico.

Na Tabela 1, o padrão `<subtópico>` representa uma parte do tópico que é parametrizada por algum identificador, como *serviceId*, *appId* ou *class*. Por exemplo, o tópico `aop/0/100/doc/nodes` indica que algum componente deseja obter informação dos nós da aplicação `100` veiculada pelo serviço identificado por `0`. É importante ressaltar que todos os tópicos apresentados na Tabela 1 ficam retidos no broker, sendo (re)enviados sempre que um cliente subscreve em um destes tópicos.

O prefixo `aop/<serviceId>/<appId>/doc` endereça tópicos que acessam informação sobre um determinado nó de uma aplicação Ginga. Complementam este prefixo identificadores dos nós representando a estrutura do documento.

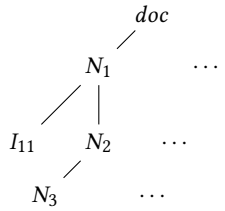

Figure 3: Exemplo de estrutura de nós de uma aplicação Ginga

A Figura 3 apresenta um exemplo de estrutura de nós (representados por N_i) e interfaces (representadas por I_{ij}) para uma aplicação identificada por `100` do serviço identificado por `0`. Considerando esse exemplo, o tópico `aop/0/100/doc/N1/N2/N3` acessa informação sobre o nó N_3. Esquema semelhante é utilizado para interfaces. Assim, o tópico `aop/0/100/doc/N1/I11` acessa informação sobre a interface I_{11} filha do nó N_1. Um componente é capaz de montar a correta estrutura do tópico para acesso à informação de um nó por meio da estrutura disponível no tópico `aop/<serviceId>/<appId>/doc/nodes` conforme apresentado na Tabela 1. Já a informação sobre as interfaces e o estado de um nó são providas conforme apresentado na Tabela 2. Note que, por simplicidade, a Tabela 2 omite o prefixo até o identificador de um nó.

A Figura 4 complementa a Tabela 2 ilustrando a estrutura de tópicos relacionados a um nó da aplicação. Na figura, subtópicos destacados em negrito representam tópicos que não ficam retidos no broker. Portanto, só são recebidos pelos clientes conectados no momento em que alguma informação é publicada nestes tópicos.

[3] https://github.com/TeleMidia/ginga

Tópico	Formato	Descrição
aop/users	URL	Caminho para arquivo de dados dos telespectadores e *thumbnails*
aop/currentUser	UUID	Identificador do perfil de telespectador selecionado
aop/services	JSON	Vetor com informação dos serviços
aop/currentService	Inteiro	Identificador do serviço em uso
aop/<serviceId>/apps	JSON	Vetor com informação das aplicações de um serviço
aop/<serviceId>/currentApp	Inteiro	Identificador da aplicação em execução
aop/<serviceId>/<appId>/path	URL	Caminho para conteúdo da aplicação
aop/<serviceId>/<appId>/doc/nodes	JSON	Vetor com informação sobre os nós de uma aplicação
aop/devices/<class>	JSON	Vetor com identificadores de dispositivos remotos registrados numa classe

Table 1: Esquema de tópicos do broker

Tópico	Formato	Descrição
.../interfaces	JSON	Vetor com informação sobre as interfaces de um nó
.../<event>/state	String	Estado da máquina de estados de evento do nó
.../<event>/occurrences	Inteiro	Número de ocorrências de um nó
.../<event>/actionNotification	String	Ação a ser realizada sobre a máquina de estados de evento do nó
.../<event>/eventNotification	String	Transição da máquina de estados de evento do nó a ser notificada
.../preparationEvent/prepared	Boleano	Indica se a preparação foi bem sucedida
.../selectionEvent/<key>/user	String	Identificação do usuário que realizou a interação
.../<ifaceid>/attributionEvent/value	String	Valor de uma interface do tipo propriedade

Table 2: Esquema de tópicos relacionados a um nó ou interface

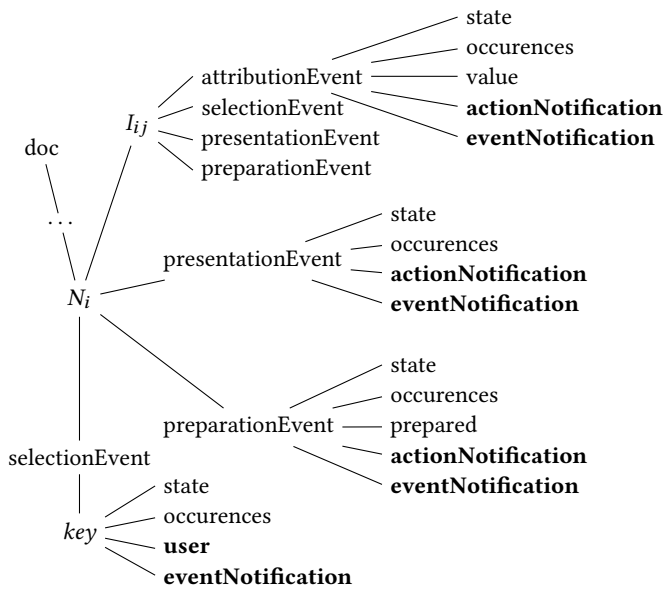

Figure 4: Exemplo de estrutura de nós de uma aplicação Ginga-NCL

A estrutura de tópicos proposta para o estado da AoP busca uma granularidade que permita que componentes mais simples, *i.e.*, que tratem de uma parte específica da AoP, não precisem filtrar a informação recebida para acessar o que é relevante.

Com a estrutura de tópicos proposta, *e.g.*, a implementação de um novo *player* de um determinado tipo de nó de uma aplicação Ginga, precisa apenas se registrar e publicar no tópico relacionado àquele nó. Assim, este *player* do exemplo, será capaz de alterar o estado do nó e da aplicação Ginga como um todo apenas se concentrando no tópico e subtópicos relativos ao nó.

5 CASOS DE USO

5.1 *Sensory Effect Presentation Engine*

A Máquina de Apresentações de Efeitos Sensoriais, ou *Sensory Effect Presentation Engine* (SEPE), é um componente de software concebido para o ecossistema da TV 3.0, atuando como uma ponte entre o middleware Ginga e dispositivos remotos heterogêneos capazes de reproduzir efeitos sensoriais, como os de Internet das Coisas (IoT). Seu objetivo principal é receber comandos originados de uma aplicação de TV, traduzi-los e orquestrar a atuação de diferentes aparelhos, como lâmpadas inteligentes e difusores de aroma, para criar experiências imersivas sincronizadas com o conteúdo audiovisual.

O princípio de funcionamento da SEPE baseia-se em uma arquitetura em camadas e no padrão de projeto *Adapter*, o que lhe confere modularidade e extensibilidade. Para cada dispositivo físico é criado um adaptador específico que traduz os comandos genéricos vindos da TV 3.0 Web Services para o protocolo proprietário do hardware. Essa estrutura permite que novos dispositivos sejam adicionados sem impactar o núcleo do sistema.

A comunicação entre a SEPE e a AoP é iniciada pela prórpia SEPE pelo processo de registro via requisição HTTP ao TV 3.0 WS, com o corpo da requisição informando a classe de dispositivo como

"sensory-effect" e seus tipos de efeitos sensoriais disponíveis. Após isso, um canal WebSocket é aberto para comunicação bidirecional persistente entre a TV 3.0 AoP e a SEPE. Através desse canal a SEPE envia metadados sobre os renderizadores que gerencia e, quando uma aplicação de TV deseja ativar um efeito, ela envia um comando HTTP para o TV 3.0 WebServices, que é encaminhado pelo Ginga ao canal WebSocket da SEPE. A SEPE então processa o comando e aciona os adaptadores correspondentes, acionando os dispositivos físicos através de seus protocolos específicos.

5.2 Motor de Apresentação Guaraná

O suporte para execução de aplicações NCL 360 no motor de apresentação Guaraná também foi implementada na arquitetura proposta. A API *Remote Device* do TV 3.0 WS foi implementada de forma a relacionar os identificadores do dispositivos conectados (handle) no tópico aop/devices/<class>, onde *class* pode ser tanto "sensory-effect", para a SEPE, quanto "ncl-360-vr", para dispositivos que implementam o motor Guaraná.

Uma implementação simplificada do componente Ginga-NCL, portanto, avalia a lista de dispositivos da classe "ncl-360-vr" conectados e, possuindo nós do tipo "ncl360", duplica estes nós associando cada cópia deste nó a um dispositivo conectado. Em seguida, o componente Ginga-NCL posta no tópico aop/0/100.doc/nodes a lista de nós, bem como cria a estrutura de tópicos da Tabela 2 para cada nó da aplicação.

O componente TV 3.0 WS, por sua vez, é responsável por associar nós da aplicação Ginga-NCL a um dos dispositivos remotos, conforme o identificador do dispositivo indicado. Feita essa associação, o componente traduz notificações de ação publicadas nos tópicos relacionados a um nó, em metadados JSON que são enviados via WebSocket ao dispositivo correspondente. O contrário também é feito, traduzindo uma notificação de evento recebida pelo dispositivo remoto na publicação no tópico correspondente do nó associado ao dispositivo.

6 CONCLUSÃO

Este trabalho apresentou uma proposta de arquitetura distribuída para a criação de uma plataforma de experimentação de aplicações e funcionalidades da TV 3.0. Na arquitetura proposta, diferentes componentes implementam funcionalidades previstas da Plataforma orientada a aplicações (AoP) mantendo o estado armazenado em um *broker*. Cada diferente componente se comunica com o *broker* para acessar/publicar informações relevantes ao funcionamento do sistema.

Para apresentar a utilidade e versatilidade da arquitetura proposta, esse artigo apresentou dois casos de uso. Ambos implementam a possibilidade de conexão com dispositivos remotos, seja para renderizar efeitos sensoriais, ou para apresentar cenas 360 em HMDs. Cada caso de uso ainda implementa funcionalidades específicas, como o controle da renderização de efeitos sensoriais por aplicações Ginga-HTML, o parsing/duplicação de nós Ginga-NCL do tipo "ncl360" e a criação/atualização da estrutura de tópicos referentes aos nós de uma aplicação Ginga-NCL.

Como trabalhos futuros, espera-se estender a implementação da plataforma proposta, incluindo mais funcionalidades da AoP, como a criação de perfis de telespectadores e uma versão mais completa do *middleware* Ginga-NCL. Além disso, será feita uma análise de outros brokers com base em critérios como consumo de recursos, suporte à validação de payloads, facilidade de configuração e disponibilidade de plugins. Esses fatores impactam diretamente a escalabilidade, a consistência dos dados e a manutenção da solução proposta. Por isso, considera-se a adoção de um broker com suporte à validação de esquemas, visando padronizar a comunicação e aumentar a robustez da arquitetura distribuída.

ACKNOWLEDGMENTS

Este trabalho foi parcialmente financiado pelo CNPq, CAPES, FAPERJ, RNP e MCOM.

REFERENCES

[1] Bestoun S Ahmed and Miroslav Bures. 2018. Testing of smart tv applications: Key ingredients, challenges and proposed solutions. In *Proceedings of the future technologies conference*. Springer, 241–256.

[2] Bestoun S Ahmed, Angelo Gargantini, and Miroslav Bures. 2020. An automated testing framework for smart tv apps based on model separation. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 62–73.

[3] Associação Brasileira de Normas Técnicas 2023. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiofusão digital Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações.* Associação Brasileira de Normas Técnicas.

[4] FÁBIO BARRETO, RAPHAEL S. DE ABREU, MARINA I. P. JOSUÉ, EYRE BRASIL B. MONTEVECCHI, PEDRO ALVES VALENTIM, and Débora C. Muchaluat-Saade. 2023. Providing multimodal and multi-user interactions for digital tv applications. *Multimedia Tools and Applications* 82 (2023), 4821–4846.

[5] Marina Josué, Marcelo F Moreno, and Débora C Muchaluat-Saade. 2024. Automatic Preparation of Sensory Effects: Managing Synchronization in Mulsemedia Applications. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 100–108.

[6] Marcelo F. Moreno, Débora Muchaluat-Saade, Guido Lemos, Sérgio Colcher, Carlos Soares Neto, Li-Chang Shuen C. S. Sousa, and Joel dos Santos. 2024. *TV 3.0: Especificações da Camada deCodificação de Aplicações.* Sociedade Brasileira de Computação, Chapter 4.

[7] Gabriel Souza, Daniel Silva, Matheus Delgado, Renato Rodrigues, Paulo R. C. Mendes, Glauco Fiorott Amorim, Alan L. V. Guedes, and Joel dos Santos. 2020. Interactive 360-degree Videos in Ginga-NCL Using Head-Mounted-Displays as Second Screen Devices. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (São Luís, Brazil) *(WebMedia '20)*. Association for Computing Machinery, New York, NY, USA, 289–296. https://doi.org/10.1145/3428658.3430972