Pontifícia Universidade Católica
do Rio de Janeiro

Carlos Eduardo Coelho Freire Batista

GINGA-MD: Uma Plataforma para Suporte à Execução de Aplicações Hipermídia Multi-Dispositivo Baseada em NCL

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do título de Doutor pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Luiz Fernando Gomes Soares

Rio de Janeiro

Fevereiro de 2013.

Carlos Eduardo Coelho Freire Batista

GINGA-MD: Uma Plataforma para Suporte à Execução de Aplicações Hipermídia Multi-Dispositivo Baseada em NCL

Tese apresentada como requisito parcial para obtenção do título de Doutor pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Luiz Fernando Gomes Soares
Orientador
Departamento de Informática - PUC-Rio

Profa. Noemi de La Rocque Rodriguez
Departamento de Informática - PUC-Rio

Prof. Renato Fontoura de Gusmão Cerqueira
Departamento de Informática - PUC-Rio

Prof. Carlos André Guimarães Ferraz
UFPE

Prof. Guido Lemos de Souza Filho
UPFB

Prof. José Eugênio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 28 de fevereiro de 2013

Carlos Eduardo Coelho Freire Batista

Graduou-se em Ciência da Computação pela UFPB em 2004. Obteve em 2005 o título de Mestre em Informática também pela UFPB. É pesquisador do Laboratório Telemídia da PUC-Rio.

Ficha Catalográfica

Batista, Carlos Eduardo Coelho Freire

GINGA-MD: uma plataforma para suporte à execução de aplicações hipermídia multi-dispositivo baseada em NCL / Carlos Eduardo Coelho Freire Batista ; orientador: Luiz Fernando Gomes Soares. – 2012.

160 f. : il. (color.) ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2012.

Inclui bibliografia

1. Informática – Teses. 2. Aplicações multi-dispositivo. 3. NCL. 4. Hipermídia distribuída. I. Soares, Luiz Fernando Gomes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Agradeço a Deus por tudo. Agradeço aos meus pais pelo amor incondicional e por tudo que sou. Agradeço às minhas irmãs Daniela, Clemens Rachel, Maria Carmen e Renata, meus cunhados (em especial Jankees, quem mais me incentivou na vida acadêmica e que agora torce pra mim do céu) e aos meus sobrinhos por todo amor e alegria proporcionada em minha vida.

Agradeço ao meu orientador, professor Luiz Fernando Gomes Soares, por toda paciência, dedicação e por tudo que me ensinou nos últimos anos; por ser mais do que um orientador acadêmico, pelo exemplo de excelência no trato profissional e principalmente pelo exemplo de caráter e valores que representa.

Agradeço a todos os membros do laboratório TeleMídia, pela amizade, pelo convívio agradável e pelas inúmeras contribuições essenciais para a realização deste trabalho. Agradeço ao Coordenador Márcio Moreno pelo suporte e pelo grande volume de conhecimento que pude adquirir. Marcelo Moreno (antigo coordenador e exemplo de líder), Álvaro, Amparito, Bruno Lima, Carlos Salles, Eduardo, Felipe Nagato, Felipe Nogueira, Guilherme, José Geraldo, Luciana, Rafael Savignon, Ricardo, Roberto, Romualdo, Vinicius e todos os que passaram pelo laboratório e que fizeram parte do legado de pesquisa que pude continuar.

Agradeço ao professor Guido Lemos e à equipe do Laboratório de Aplicações de Vídeo Digital da UFPB, pelo suporte e colaboração. À professora Tatiana Aires e aos pesquisadores Álan Livio, Ana Paula Nunes, José Ivan, Luís Feliphe, Lucenildo, Rafael Rossi, Sttiwe e todos os outros membros da equipe.

Agradeço aos meus amigos e todos aqueles com que tive contato profissional durante o desenvolvimento deste trabalho, durante as reuniões do Fórum Brasileiro de TV Digital e da União Internacional de Telecomunicações.

Finalmente, agradeço à CAPES, ao CNPq e à RNP pelo apoio para o desenvolvimento dos trabalhos associados à realização desta tese.

Resumo

Batista, Carlos Eduardo Coelho Freire; Soares, Luiz Fernando Gomes. **GINGA-MD: Uma Plataforma para Suporte à Execução de Aplicações Hipermídia Multi-Dispositivo Baseada em NCL.** Rio de Janeiro, 2012. 160p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O crescente número de formatos de mídias digitais fomentou a criação de aplicações multimídia interativas, incluindo o desenvolvimento de cenários interativos para múltiplos usuários. Existem muitos espaços onde artefatos multimídia são consumidos por grupos de pessoas, tais como ambientes domésticos com TV Digital, salas de Cinema Digital e salas de conferência com apresentações multimídia. Nesta tese, uma plataforma para execução de aplicações hipermídia distribuídas é proposta para ser utilizada em tais espaços onde ocorre consumo coletivo de multimídia interativa. A plataforma proposta usa a linguagem NCL como seu formato de descrição de aplicações, pois NCL é uma linguagem de cola para autoria de documentos hipermídia que suporta o conceito de aplicações multi-dispositivo através de abstrações declarativas. A plataforma estabelece uma arquitetura de software de referência, definindo mecanismos e interfaces para a integração de dispositivos heterogêneos. Um protótipo foi implementado e validado em diferentes cenários de uso, nos quais aplicações hipermídia usam recursos de mídia capturados por e sendo transmitidos para múltiplos dispositivos.

Palavras-chave

Aplicações multi-dispositivo; NCL; Hipermídia Distribuída.

Abstract

Batista, Carlos Eduardo Coelho Freire; Soares, Luiz Fernando Gomes (Advisor). **GINGA-MD: An NCL Based Platform for Supporting the Execution of Multi-Device Hypermedia Applications.** Rio de Janeiro, 2012. 160p. DSc. Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The increase of digital media formats fostered the creation of interactive multimedia applications, including the development of multi-user interactive scenarios. There are many spaces where digital interactive multimedia artifacts are consumed by groups of people, such as homes with Digital TV, theaters with Digital Cinema and conferences with interactive lecture presentations. In this thesis, a platform to support the execution of distributed hypermedia applications is proposed aiming at these spaces of collective digital multimedia consumption. The proposed platform uses the NCL language as the application description format, since it is a hypermedia glue-language that supports the concept of multi-device applications following a declarative abstraction level. The platform establishes a reference software architecture, defining mechanisms and interfaces for heterogeneous device integration. A prototype is implemented and validated against different usage scenarios, in which hypermedia applications use media resources coming from and going to multiple devices.

Keywords

Multi-device applications, NCL; Distributed Hypermedia.

Sumário

1 Introdução 12
1.1. Motivação e Objetivos 14
1.2. Organização da Tese 15

2 Cenários de Uso 17
2.1. Descrição dos cenários de uso 18
2.2. Requisitos de Alto Nível 23

3 Trabalhos Relacionados 26
3.1. Hipermídia distribuída em aplicações de TV Digital interativa 28
3.2. Hipermídia distribuída em aplicações de Cinema Digital interativo 40
3.3. Hipermídia distribuída em apresentações de eslaides multimídia 42
3.4. Outros sistemas de hipermídia distribuída relevantes 44
3.5. Resumo comparativo 46

4 Extensões para linguagem NCL 50
4.1. Estendendo as classes de dispositivos NCL 51
4.2. Classe Captura de Mídia 55
4.3. API Lua 56

5 Arquitetura de Software 60
5.1. Visão geral – Requisitos 62
5.2. Visões da Arquitetura de *Software* da Plataforma 70

6 Implementação do Protótipo 106
6.1. Recuperação e Tolerância a falhas na execução de aplicações hipermídia distribuídas 107
6.2. Visão de Processos 114

7 Testes Sistêmicos 132

7.1. Aplicações de Teste 133
7.2. Testes do protótipo 141

8 Conclusão e Trabalhos Futuros 151
8.1. Contribuições da Tese 151
8.2. Trabalhos Futuros 152

9 Referências 154

Lista de figuras

Figura 1 – Entidades funcionais de alto-nível e arquitetura de rede para execução de aplicações hipermídia distribuídas 63
Figura 2 – Diagrama de segmentação em camadas para o *Base Device Subsystem*. 90
Figura 3 – Diagrama de segmentação em camadas para o *Secondary Device Subsystem* 92
Figura 4 – Diagrama de segmentação em camadas para o *Application Provider Subsystem* 93
Figura 5 – Diagrama de pacotes para o *Base Device Subsystem* 94
Figura 6 – Diagrama de pacotes para o *Secondary Device Subsystem* 95
Figura 7 – Diagrama de pacotes para o *Application Provider Subsystem* 96
Figura 8 – Diagrama de componentes UML ilustrando principais componentes do *Base Device Subsystem*. 97
Figura 9 – Diagrama de componentes para o *Secondary Device Subsystem* 103
Figura 10 – Diagrama de componentes para o *Application Provider Subsystem* 105
Figura 11 – Diagrama de sequência para processo de carregamento de aplicação NCL realizado pelo *Base Device Subsystem* 115
Figura 12 – Diagrama de sequência para o processo de atualização de aplicação NCL em execução no *Base Device Subsystem*. 117
Figura 13 – Diagrama de sequência para processo de ativação e desativação de serviços das classes de dispositivos (quando da atualização da aplicação NCL em execução no *Base Device Subsystem*). 119
Figura 14 –. Diagrama de sequência para processo de pareamento entre Dispositivo Base e Dispositivo Secundário 121
Figura 15 – Diagrama de sequência para o processo de finalização do pareamento entre os subsistemas. 122
Figura 16 – Diagrama de sequência para as etapas de orquestração de objetos associados à Classe Passiva 124
Figura 17 – Diagrama de sequência para as etapas de orquestração de

objetos associados à Classe Ativa 126
Figura 18 – Diagrama de sequência as atividades de orquestração de objetos associados à Classe Captura de Mídia 128
Figura 19 – Diagrama de sequência as atividades de orquestração de objetos associados à Classe *UPnP AV MediaServer ControlPoint* 129
Figura 20 – Visão estrutural da aplicação "Brazil 14 Bis" 134
Figura 21 – Visão estrutural da aplicação "Luzia e a Vaca Andorinha" 137
Figura 22 – Visão estrutural para parte da aplicação "Aula sobre NCL" (parte executada pelo Dispositivo Base) 138
Figura 23 – Visão estrutural para parte da aplicação "Aula sobre NCL" (parte executada pelo Dispositivo Secundário registrado na Classe Ativa Fértil) 140
Figura 24 – Ambiente de Testes no LAVID/UFPB 141
Figura 25 – Diagrama de implantação UML para cenário de TV Digital interativa 142
Figura 26 – Testes no Laboratório TeleMídia da PUC-Rio (Aplicação "Brazil 2014 Bis") 144
Figura 27 – Diagrama de implantação UML para o cenário de Cinema Digital interativo 144
Figura 28 – Execução da aplicação "Luzia e a Vaca Andorinha" em ambiente simulado (Ginga for Windows) - Janela do Dispositivo Base apresenta o vídeo principal enquanto as janelas dos Dispositivos Secundários apresentam aplicação NCL para votação nas opções para o desenrolar da trama. 146
Figura 29 – Diagrama de implantação UML para cenário de Apresentação de eslaides multimídia interativa 147
Figura 30 – Telas para aplicação "Aula sobre NCL" (a – Tela do Dispositivo Base; b – Tela Dispositivo Secundário Pai; c – Tela dos Dispositivos Secundários filhos) 148
Figura 31 – Testes com usuários no LAVID/UFPB 149

Lista de tabelas

Tabela 1 – Resumo comparativo dos Trabalhos Relacionados 49
Tabela 2 – Atributos acrescentados às classes definidas pelo UAProf 54
Tabela 3 – Definição da classe *NCLDeviceClass* 54
Tabela 4 – Métodos síncronos – módulo *multidevice.* 56
Tabela 5 – Descrição da API para componente *Device Integration Manager* (interface *IDeviceIntegrationManager*) 101
Tabela 6 – Descrição da API para componente *Secondary Device Resource Manager* (interface *IDeviceResourceManager*). 104
Tabela 7 – Tipos de comunicação, Tipos de Falhas e Respectivas Ações. 113
Tabela 8 – Configuração dos nós físicos - Cenário de TV Digital interativa 143
Tabela 9 – Configuração dos nós físicos - Cenário de Cinema Digital interativo 145
Tabela 10 – Configuração dos nós físicos - Cenário de Apresentação de eslaides multimídia interativa 148

1
Introdução

A última década registrou avanços significativos nas tecnologias de comunicação em rede e também uma popularização massiva dos dispositivos computacionais portáteis. Surgem, então, aplicações que exploram recursos oferecidos simultaneamente por múltiplos dispositivos interconectados, em diferentes contextos de uso. Dentre as possíveis aplicações distribuídas voltadas para o usuário final, destacam-se as aplicações interativas que lidam com objetos multimídia ou aplicações hipermídia.

O desenvolvimento de aplicações hipermídia distribuídas é uma tarefa de alta complexidade, pois o sincronismo entre os dispositivos envolvidos é dificultado por características inerentes à comunicação em rede e pelas diferenças das configurações dos dispositivos. Para facilitar o desenvolvimento de aplicações hipermídia que utilizem recursos de dispositivos heterogêneos interconectados, é necessário estabelecer plataformas de software interoperáveis (padronizadas) para o desenvolvimento e execução de tais aplicações. Uma plataforma com tal intuito deve oferecer abstrações (através de linguagens e/ou API) que escondam ou minimizem a complexidade de se lidar com um ambiente distribuído e heterogêneo, garantindo que os requisitos das aplicações (definidos por tais abstrações) possam ser corretamente interpretados pelos dispositivos durante a execução dessas aplicações hipermídia distribuídas.

O uso de múltiplos dispositivos para a execução de aplicações hipermídia não é uma funcionalidade nova (Cesar, 2007; Silva 2007; Costa 2009; Soares 2009). São vários os registros na literatura científica acerca de aplicações onde vários dispositivos são utilizados para entrada de dados ou para exibição dos conteúdos das mídias. Comercialmente, já existem plataformas que oferecem algum suporte ao desenvolvimento e à execução de aplicações multimídia que utilizem recursos distribuídos (tela, alto falantes, armazenamento etc.). Tecnologias como UPnP (*Universal Plug and Play*) (Miller 2001), OSGi (*Open Services Gateway initiative*) (OSGi 2011) e Bluetooth integram dispositivos

heterogêneos e podem ser utilizadas para lidar com informação multimídia, porém, em aplicações dentro de um contexto de *hardware* e *software* restrito. UPnP e Bluetooth focam na entrega distribuída e compartilhamento de objetos em redes domésticas, enquanto OSGi foca na comunicação entre dispositivos nessas redes para fins de automação residencial.

As aplicações já existentes dependem, em sua maioria, de implementações proprietárias de *software* (Holmes 2012; Cortez 2012), ou instalações com configurações específicas de *hardware* e *software* (Izadi 2003; McGinity 2007; Jagodic 2011), situações que impõem várias restrições quanto aos tipos de dispositivos integráveis e também quanto aos tipos de conteúdo multimídia envolvidos.

No âmbito das redes domésticas, plataformas para execução de aplicações de TV Digital Interativa vislumbram o suporte ao desenvolvimento de aplicações hipermídia distribuídas. As especificações de *middleware* ARIB (*Association of Radio Industries and Businesses*) (ARIB 2004), do sistema japonês ISDB (*Integrated Services Digital Broadcasting*), e do *middleware* Ginga (ABNT 2007), do Sistema Brasileiro de TV Digital (SBTVD ou ISDB-Tb), incorporam funcionalidades que permitem que o receptor de TV Digital utilize recursos dos dispositivos localmente conectados durante a execução de uma aplicação interativa. O *middleware* Ginga oferece em suas especificações suporte à execução de aplicações em múltiplos dispositivos através de seus dois ambientes de execução: o imperativo, chamado Ginga-J (Souza 2007) e o declarativo, obrigatório, chamado Ginga-NCL (Soares 2007).

NCL (*Nested Context Language*, linguagem utilizada pelo Ginga-NCL) (Soares 2006) é uma linguagem de autoria hipermídia que age como uma linguagem de cola, relacionando objetos de mídia no tempo e no espaço. NCL não restringe ou prescreve nenhum tipo de objeto de mídia: imagens, vídeos, áudio, texto, aplicações imperativas (scripts Lua, aplicações Java) e declarativas (HTML, SMIL etc.) são suportadas pela linguagem, contanto que os exibidores e *softwares* decodificadores necessários estejam presentes no ambiente de execução. NCL oferece mecanismos de adaptação de conteúdo e de apresentação, tratamento de eventos de interação do usuário, e edição da estrutura da apresentação em tempo de execução.

A linguagem NCL dá suporte ao desenvolvimento de aplicações que associam a execução de seus objetos de mídia a exibidores remotos (dispositivos conectados a um receptor de TV por uma rede IP, por exemplo), através de um modelo hierárquico de orquestração distribuída. A linguagem não define quais protocolos devem ser utilizados para a comunicação necessária para a orquestração e nem quais são as características consideradas para o estabelecimento dos canais de comunicação entre os dispositivos envolvidos em uma aplicação.

Outras linguagens declarativas também dão suporte à definição de documentos apresentados por múltiplos dispositivos cooperativamente, notadamente SMIL (W3C 2008) (e suas extensões StoryML (Hu 2003) e IPML (Hu 2009)). Porém, tais abordagens possuem especificações em um nível de abstração muito alto e, como resultado, nenhuma plataforma, até a conclusão desta tese, implementa a execução de aplicações hipermídia distribuídas baseadas em StoryML e IPML. Dentre as poucas plataformas implementadas que dão suporte à execução de aplicações hipermídia distribuídas, nenhuma oferece uma solução confiável de interoperabilidade, com a possibilidade de integrar recursos heterogêneos dos mais variados dispositivos através de abstrações facilmente usáveis.

A descrição e a implementação de uma plataforma para exibição de documentos NCL de forma distribuída é proposta nesta tese. A linguagem é estendida e utilizada para descrever aplicações que integrem recursos heterogêneos de dispositivos interconectados, viabilizando cenários de uso inéditos em diferentes contextos de exibição.

1.1. Motivação e Objetivos

A complexidade vinculada ao desenvolvimento de aplicações hipermídia distribuídas e o fato de que os atuais sistemas que lidam com tais aplicações não são capazes de suprir um conjunto significativo de requisitos são elementos basilares da justificativa deste trabalho. O suporte dado pela linguagem NCL às aplicações hipermídia distribuídas, em particular, motivou investigações mais

profundas sobre as possibilidades de integrar dispositivos de acordo com a lógica de uma aplicação hipermídia.

Assim, a motivação para esta tese partiu da necessidade de facilitar o desenvolvimento e aumentar o universo de possibilidades das aplicações hipermídia distribuídas em NCL. Novas funcionalidades são elencadas em cenários de uso hipotéticos (apresentados no Capítulo 2), formulados a partir de requisitos não supridos em conjunto até a realização desta tese. As funcionalidades aumentam o escopo de uso da linguagem, possibilitando seu uso para o desenvolvimento de aplicações hipermídia distribuídas voltadas para ambientes de execução (infraestrutura) diferentes, facilitando a integração de dispositivos heterogêneos.

Assim, esta tese tem por objetivo principal provar que é possível se estabelecer uma plataforma capaz de dar suporte à execução de aplicações hipermídia multi-dispositivo, a partir de uma descrição de alto nível que abstraia características dos diferentes dispositivos e mecanismos de comunicação e viabilize um universo significativo de funcionalidades. A validação do modelo de apresentação distribuída da linguagem NCL e contribuições para a linguagem serão propostas a partir da modelagem de uma arquitetura de *software* que atenda aos requisitos extraídos dos cenários motivadores (do Capítulo 2).

A implementação de um protótipo que cubra a arquitetura de *software* modelada, face aos requisitos estabelecidos, e a definição de mecanismos de avaliação para esta arquitetura são objetivos associados à verificação da solução proposta nesta tese.

1.2. Organização da Tese

A tese se organizada como apresentado a seguir:

Capítulo 2 – "Cenários de Uso": são apresentados cenários de uso relevantes para aplicações hipermídia distribuídas, dos quais são extraídos requisitos de alto-nível.

Capítulo 3 – "Trabalhos Relacionados": apresenta uma avaliação do estado da arte, analisando o atendimento dos requisitos extraídos dos cenários de uso.

Capítulo 4 – "Extensões para a linguagem NCL": propõe extensões para a linguagem NCL, com novas funcionalidades que viabilizem os cenários de uso considerados.

Capítulo 5 – "Arquitetura de *Software*": apresenta uma proposta de arquitetura de *software* para o atendimento dos requisitos estabelecidos.

Capítulo 6 – "Implementação do Protótipo": descreve uma implementação de protótipo da arquitetura de *software* proposta como solução.

Capítulo 7 – "Testes Sistêmicos": descreve uma metodologia de testes e o resultado de sua utilização com o protótipo desenvolvido.

Capítulo 8 – "Conclusão e Trabalhos Futuros": conclui a tese, sumarizando as contribuições realizadas e discutindo possíveis trabalhos futuros.

2
Cenários de Uso

A digitalização das mídias de representação proporcionou o surgimento de aplicações multimídia interativas, cujos mecanismos de interação, a princípio, permitiam o controle e navegação a partir de um único dispositivo. Posteriormente, aplicações que lidam com conteúdo multimídia digital passaram a ser desenvolvidas para cenários de execução distribuída (Silva 2007; Costa 2009), graças à evolução das redes de computadores.

Os *softwares* que suportam a exibição de conteúdo multimídia digital oferecem cada vez mais funções para execução cooperativa. Vários *softwares* para exibição e produção multimídia já dão suporte aos padrões que oferecem facilidades para busca e compartilhamento de conteúdo multimídia em rede (por exemplo, suporte à UPnP em exibidores como Windows Media Player[1], VLC[2] e vários outros). É crescente também o número de *softwares* exibidores que oferecem API e/ou interfaces para controle através de serviços de rede (via HTTP, por exemplo, como é o caso do VLC, ou o suporte dado ao protocolo OSC[3] em várias ferramentas de produção de áudio digital). Esses novos exibidores e ferramentas facilitam a construção de cenários sofisticados para consumo de conteúdo multimídia de forma distribuída (Melo 2010).

Diversas pesquisas abordam o desenvolvimento de formas diferentes de reprodução e navegação em aplicações distribuídas que lidam com conteúdo multimídia. Alguns sistemas já proporcionam uma experiência multimídia coletiva e interativa usando recursos disponibilizados por múltiplos dispositivos (chamaremos de **aplicação hipermídia distribuída** a descrição lógica dessa experiência). Este capítulo apresentará cenários de uso inovadores para aplicações hipermídia distribuídas.

Os dispositivos computacionais se popularizaram massivamente com o advento dos portáteis e agregam cada vez mais recursos destinados à reprodução e

[1] http://windows.microsoft.com/en-US/windows/windows-media-player
[2] http://www.videolan.org

manipulação de conteúdo multimídia. Sistemas que se inserem no ambiente dos usuários desses dispositivos de forma transparente (chamados de pervasivos ou ubíquos (Helal 2002; Paterno 2008)) hoje oferecem os mais variados serviços, a partir de mecanismos de manuseio fácil. Espaços destinados ao consumo de conteúdo multimídia de forma coletiva possuem hoje infraestrutura necessária para a reprodução de conteúdo digital, além de prover suporte a usuários munidos de dispositivos computacionais portáteis, fato que recentemente vem sendo explorado para criar novas experiências multiusuário.

Diversas aplicações interativas destinadas aos espaços de assistência coletiva (como residências e outros locais que possuem um receptor de TV Digital (Silva 2008; Costa 2009; Holmes 2012), salas de projeção de Cinema Digital (Lew 2004; McGinity 2007) e salas para apresentações de eslaides multimídia (Glasberg 2004; Gaspar 2007) são encontradas na literatura científica, e muitas delas utilizam dispositivos portáteis. Tais aplicações exploram características dos espaços coletivos, e nelas os dispositivos são utilizados como recursos auxiliares (secundários) de visualização e interação (Soares 2008; Holmes 2012).

Exemplos descritivos do comportamento de três aplicações hipermídia distribuídas, executadas em espaços coletivos distintos, são abordados nesta tese. Tais exemplos, na forma de cenários de uso, serão utilizados para dar suporte à elaboração dos requisitos para uma plataforma que dê suporte à execução de aplicações hipermídia distribuídas. A próxima subseção discorrerá sobre esses cenários de uso. Os cenários contemplam aplicações que exploram funcionalidades inovadoras, não oferecidas, em conjunto, por nenhum outro sistema (como será realçado na seção seguinte). A descrição dos cenários inclui as características dos espaços considerados e as etapas de execução das aplicações usando uma plataforma idealizada. A *posteriori*, são discutidos requisitos de alto nível extraídos a partir dos cenários apresentados.

2.1. Descrição dos cenários de uso

Três aplicações hipermídia distribuídas utilizando diferentes espaços para assistência coletiva são descritas nesta seção. Cada etapa relevante da execução

[3] http://opensoundcontrol.org/

das aplicações é apresentada, assim como as características basilares das entidades funcionais presentes e a maneira como se comunicam durante a execução. O objetivo é definir um universo significativo de novos requisitos para uma arquitetura de *software* que viabilize a execução de aplicações hipermídia distribuídas.

Listamos abaixo alguns conceitos utilizados para a descrição dos cenários de uso e que são utilizados no restante da tese:

- Um **dispositivo computacional multimídia** ou simplesmente **dispositivo** é um equipamento com capacidade de processamento, reprodução de conteúdos de mídia digital, interação com o usuário e comunicação em rede. Tais dispositivos podem ser utilizados em conjunto para a execução de uma aplicação hipermídia distribuída. Exemplos: Computadores *desktop*, receptores de TV Digital, *laptops*, *smartphones*, *tablets* etc.
- Um **objeto de mídia** é, no escopo desta tese, uma estrutura de dados que contempla uma referência para um conteúdo de mídia digital e parâmetros de execução associados. Uma aplicação hipermídia especifica objetos de mídia referenciando: mídias estáticas (cuja reprodução não é modificada com o tempo, e.g. imagens estáticas e texto), mídias contínuas (cuja reprodução se dá pela decodificação de amostras com base no tempo, e.g. arquivos de vídeo e de áudio) e também programas em código declarativo (e.g. páginas HTML, documentos NCL) e imperativo (e.g. scripts Lua, aplicações Java) (Soares 2005; Soares 2006; Soares 2007). Um objeto de mídia é reproduzido através dos *softwares* exibidores presentes nos dispositivos computacionais multimídia.
- Uma **plataforma para execução de aplicações hipermídia distribuídas** ou simplesmente **plataforma de hipermídia distribuída** é uma arquitetura de *software* que viabiliza a execução de aplicações hipermídia utilizando recursos oferecidos por múltiplos dispositivos conectados. Uma plataforma de hipermídia distribuída realiza a execução de uma aplicação através de um formato padronizado, que deve oferecer mecanismos para que autores explorem as funcionalidades associadas aos recursos dos dispositivos.
- **Pareamento** é o processo de estabelecimento de um canal de comunicação entre dois dispositivos. O pareamento considerado nesta tese estabelece um

relacionamento hierárquico entre os dois dispositivos, onde um, o pai, pode orquestrar recursos oferecidos pelo outro, o filho.

2.1.1. Cenário 1: TV Digital Interativa

Em uma sala de estar, uma família está reunida para assistir um programa de TV Digital interativa. O programa é recebido pelo receptor de TV Digital de uma estação transmissora de TV, através de transmissão *broadcast* em uma rede terrestre. A família é composta pelo pai, pela mãe e por dois filhos adolescentes, e todos possuem dispositivos computacionais portáteis: o pai um *tablet* e a mãe e os filhos *smartphones*. Esses dispositivos portáteis são capazes de tratar e executar partes de uma aplicação hipermídia. Quando o programa interativo começa – um especial sobre todas as edições da Copa do Mundo de Futebol – os dispositivos se pareiam com o receptor de TV Digital, habilitando assim a interação com múltiplos dispositivos.

Em um dado momento a aplicação interativa passa a ser exibida na tela do receptor de TV Digital, enquanto que os dispositivos pareados passam a exibir um menu de opções adaptado para as configurações de visualização em cada um deles. A aplicação na tela da TV apresenta informações sumarizadas sobre os campeões de todas as edições da Copa do Mundo (texto, imagens e animações). O menu apresentado nos dispositivos oferece a visualização dos créditos da aplicação e uma opção para que a aplicação interativa seja apresentada apenas nos dispositivos e não mais na tela do receptor de TV Digital. Quando essa opção é selecionada pela mãe, o vídeo principal da aplicação toma a tela toda e todos os outros objetos de mídia visuais deixam de ser exibidos na tela do receptor de TV Digital; surgindo, em versão adaptada, nos dispositivos dos membros da família, individualizando a navegação que antes era compartilhada pelos dispositivos de entrada do receptor de TV Digital.

Um dos membros da família pareia seu dispositivo apenas após a exibição da interatividade nos dispositivos já anteriormente pareados, porém também passa a exibir imediatamente a versão adaptada da aplicação, logo após parear-se.

No intervalo comercial, outra aplicação é exibida nos dispositivos. É apresentada uma página *Web* com um formulário para a compra de pacotes

turísticos para a Copa do Mundo de 2014. Depois dos comerciais, a aplicação anterior (com informações sobre os campeões) volta a ser exibida nos dispositivos pareados. Pouco depois do início do segundo bloco, mais um integrante se junta à audiência local do programa: um amigo de um dos filhos, que está de posse de um *smartphone*. O *smartphone* do amigo não possui o *software* necessário para interagir com o programa de TV sendo apresentado nos demais dispositivos, porém, é capaz de visualizar uma lista de vídeos com todos os gols exibidos no programa e recuperá-los sob demanda. Tais vídeos são disponibilizados pelo receptor de TV Digital dentro da lógica da aplicação interativa, utilizando protocolos e mecanismos padronizados que permitam dispositivos compatíveis com outras plataformas (como por exemplo UPnP/DLNA) acessarem também recursos da aplicação interativa.

No fim do segundo e último bloco do programa, um jogo interativo é exibido nos dois primeiros dispositivos que se parearam com o receptor (os outros exibem os créditos do jogo). O jogo simula a cobrança de um pênalti; um dos dispositivos atua como goleiro e outro como jogador de linha. O usuário que controla o jogador de linha escolhe para qual lado o mesmo deve bater o pênalti e, independentemente, o outro usuário escolhe para que lado o goleiro deve pular. O resultado da escolha de ambos é combinado pelo receptor de TV Digital e exibido em sua tela: se o usuário que controla o goleiro acertou o canto que o usuário que controla o jogador de linha escolheu ocorre a defesa; em caso contrário, o gol. Quando todos os dispositivos finalizam o pareamento com o receptor, a aplicação, em seu todo é novamente exibida na tela da TV.

2.1.2. Cenário 2: Cinema Digital

Em uma sala de Cinema Digital um curta metragem interativo é exibido antes de um longa metragem convencional. As pessoas na plateia poderão interagir com a exibição do curta depois de terem seus dispositivos pessoais pareados com o projetor digital, que tem acesso ao conteúdo do curta interativo a partir do acesso a um servidor em uma rede privada.

O curta apresenta uma história onde o protagonista deve fazer uma série de decisões de múltipla escolha que influenciam o desenrolar da trama apresentada

em vídeo. Cada decisão do protagonista é tomada a partir do resultado de uma votação realizada pela plateia (Souto 2008) por meio dos seus dispositivos pessoais. Quando o protagonista do curta metragem se depara com instantes de decisão, um menu com as possíveis escolhas é apresentado nos dispositivos dos espectadores para que escolham uma das opções. A cena a ser apresentada após cada instante de decisão será escolhida pelo projetor interativo com base na opção mais votada pela plateia.

Durante a exibição de certas cenas, informações adicionais (que complementam a trama) são apresentadas nos dispositivos. Dispositivos pareados com a exibição do curta em andamento exibem o conteúdo adicional, de forma sincronizada com as informações referentes ao momento atual do curta.

Após a última cena do filme interativo, é apresentada uma peça publicitária onde um personagem realiza uma ligação telefônica e um dos dispositivos pareados é sorteado. O dispositivo sorteado passará a exibir (só ele) uma aplicação que simula o recebimento da chamada feita pelo protagonista exibido na tela do projetor. Se o usuário selecionar a opção de atender, ouvirá uma mensagem de áudio (o personagem perguntando seu nome e assento) e poderá também capturar uma amostra de áudio, para que seja reproduzida pelo dispositivo de áudio de alta fidelidade acoplado ao projetor digital. Caso o espectador não selecione a opção de atender ou desconecte-se sem realizar a captura do trecho de áudio, uma mensagem alternativa é exibida.

Também após a última cena, os vídeos do final não selecionado pela votação também são transmitidos, codificados em menor qualidade, para os dispositivos que possuem apenas um exibidor de fluxos de vídeo, associado a um mecanismo de configuração automática. A transmissão dos vídeos é feita pelo projetor digital para esses dispositivos utilizando protocolos de comunicação de grupo (transmissão *multicast* ou *broadcast*), de forma que todos os dispositivos consumam o conteúdo dos vídeos simultaneamente.

2.1.3. Cenário 3: Apresentações de Eslaides Multimídia

Uma apresentação de eslaides multimídia interativa é conduzida por um apresentador em um auditório como parte de uma palestra em um congresso. O

dispositivo acoplado a um projetor (um *laptop*) passa a executar uma aplicação hipermídia que caracteriza a apresentação. A aplicação é recuperada através do acesso a um repositório na *Web*. O palestrante utiliza um *tablet* como dispositivo secundário que executa uma parte da aplicação hipermídia da apresentação, recebida após o pareamento com o *laptop*. O *tablet* passa a exibir uma aplicação de navegação, que oferece um menu com as opções disponíveis para uma apresentação com dispositivos integrados.

A partir daí os espectadores podem parear-se com o *tablet* do palestrante e interagir com a apresentação utilizando seus dispositivos computacionais (como *smartphones*, *laptops* e *tablets*). Durante a apresentação, são exibidos nos dispositivos questionários interativos, que capturam aspectos qualitativos e quantitativos relacionados ao tema da palestra.

Na tela do dispositivo do apresentador é exibido o estado da apresentação e um indicador de quais dos espectadores já responderam seus questionários, para que assim ele possa determinar quando devem deixar de ser exibidos e prosseguir com a exposição na tela do projetor. A qualquer momento membros da plateia podem se integrar ao cenário, quando então seus dispositivos recebem as anotações e questionários corretamente sincronizados com a parte da apresentação que está sendo exibida na tela do projetor.

Durante a conclusão da apresentação, todos os usuários que possuírem dispositivos compatíveis com outras plataformas que oferecem serviços multimídia em redes locais (exemplo: UPnP/DLNA) poderão receber um vídeo relacionado com a apresentação interativa. A apresentação encerra-se com a exibição na tela do projetor dos dados colhidos através dos questionários interativos respondidos pelos usuários de dispositivos secundários.

2.2. Requisitos de Alto Nível

Apesar de os cenários apresentarem a execução de aplicações hipermídia distribuídas em espaços com finalidades diferentes, é possível identificar um conjunto significativo de características em comum empregadas por elas. Assim, podem-se elencar requisitos de alto nível para o desenvolvimento de uma

plataforma que desse suporte à execução das aplicações apresentadas nos cenários. São eles:

- **Modelo de orquestração hierárquico** – a comunicação entre os dispositivos é realizada através de uma organização lógica hierárquica, definida pela descrição da aplicação hipermídia distribuída em execução, e materializada pelo pareamento dos dispositivos. Os cenários descreveram aplicações que eram inicialmente executadas por um dispositivo compartilhado, e que ofereceram pareamento para outros dispositivos (individuais). Os recursos dos dispositivos foram orquestrados, de acordo com a lógica da aplicação de cada cenário, através da troca de mensagens entre os dispositivos de acordo com a hierarquia estabelecida.
- **Integração dinâmica de dispositivos** – a configuração da topologia de comunicação é realizada dinâmica e transparentemente nos cenários apresentados. Os dispositivos individuais se pareaam com o dispositivo compartilhado em momentos arbitrários e se engajaram normalmente na aplicação.
- **Comunicação com dispositivos individuais e grupos de dispositivos** – nos cenários descritos, os dispositivos são tratados em grupo ou como entidades individuais, de acordo com a lógica da aplicação.
- **Sincronismo entre objetos de mídia executados em diferentes dispositivos** – os cenários apresentam o sincronismo da execução de objetos de mídia em dispositivos diferentes. O controle do ciclo de vida e o monitoramento remoto da execução dos objetos é realizado de acordo com a hierarquia de comunicação estabelecida entre os dispositivos.
- **Interação multimodal distribuída** – informações heterogêneas trafegaram entre os dispositivos nos cenários apresentados: sinalização de orquestração, partes das aplicações etc. Essas informações são geradas e tratadas em paralelo, sem nenhuma restrição quanto ao tipo de informação manipulada, e são distribuídas de acordo com a hierarquia definida pela aplicação hipermídia distribuída em execução (de ascendente para descendente e vice-versa).
- **Adaptação de conteúdo e apresentação** – a exibição das partes das aplicações associadas aos dispositivos (*smartphones*, *tablets* etc.) foi adaptada de acordo com os dispositivos. A adaptação foi realizada através da seleção de

conteúdo (dentre um conjunto pré-definido de possibilidades) e também pela manipulação dos parâmetros de exibição dos objetos de mídia.

- **Mecanismos de interoperabilidade** – os cenários integraram na lógica de uma mesma aplicação hipermídia dispositivos com características heterogêneas de *hardware*, *software*, e que suportavam diferentes recursos de comunicação (*hardware* e protocolos de rede associados).
- **Mecanismos de tolerância à falha** – a infraestrutura de comunicação e os dispositivos envolvidos nos cenários apresentados são inerentemente propensos a falhas no funcionamento; como descrito nos cenários, falhas específicas (fim abrupto da conexão, falha de sincronismo etc.) foram contornadas automática ou semi-automaticamente (comportamento padrão ou com tratamento específico dentro da lógica de uma aplicação).

Requisitos relacionados com segurança da informação e escalabilidade são importantes, porém, não foram considerados para o conjunto de requisitos supracitado (tais requisitos são discutidos no Capítulo 8). Na próxima seção serão analisados trabalhos que possuem afinidade funcional com as aplicações apresentadas nos cenários de uso. O propósito da análise é resumir quais são os avanços acerca da realização dos requisitos de alto nível definidos, realçando quais são as questões em aberto abordadas nesta tese.

3
Trabalhos Relacionados

No escopo da realização desta tese, foram feitas consultas regulares na literatura científica relacionada em busca de trabalhos com soluções para a implementação de sistemas com requisitos similares aos apresentados no capítulo anterior. Até a conclusão desta tese, não foi encontrada outra abordagem que supriu todos os requisitos necessários para a realização dos três cenários descritos.

A literatura registra avanços relevantes que concretizam parcialmente as funcionalidades exploradas nos cenários de uso. Alguns dos trabalhos considerados nesta seção possuem foco similar ao definido para esta tese, no tocante aos espaços e grupos de pessoas considerados para os cenários. Já outros suprem funcionalidades similares às descritas nos cenários de uso, porém através de sistemas genéricos ou voltados para outros espaços de uso coletivo. A análise contemplada por esta seção aborda, em sua maior parte, sistemas onde há um forte acoplamento entre os subsistemas distribuídos envolvidos (sistemas onde há interface coerente (Dix 2008)). Nesses sistemas os usuários percebem que o processo de interação (intermediado por seus dispositivos) é parte da visualização multimídia compartilhada pelo coletivo.

Com a popularização dos dispositivos computacionais portáteis e a evolução das tecnologias de comunicação em rede, surgiram protocolos e plataformas de *hardware* e *software* que têm por objetivo integrar os recursos abundantes em redes que possuem muitos dispositivos conectados (redes domésticas, coorporativas etc.). Um conjunto representativo de tais tecnologias está relacionado ao desenvolvimento de aplicações multimídia distribuídas, que são utilizadas como base para grande parte dos trabalhos referenciados por esta seção. As principais tecnologias citadas são resumidas abaixo:

- UPnP – *Universal Plug and Play* (Miller 2001) é o nome dado a um conjunto de protocolos que visam facilitar (tornar transparente) a descoberta, descrição, apresentação e controle de serviços oferecidos por dispositivos em uma rede local. Baseia-se em padrões como HTTP, HTML, XML e SOAP e é requisito

básico para a certificação DLNA (*Digital Living Network Alliance* (Kim 2007)), popular no mercado de dispositivos multimídia.

- OSGi – *Open Services Gateway initiative framework* é um conjunto de ferramentas e bibliotecas de *software* para o desenvolvimento de aplicações que podem ser remotamente instaladas e controladas. O *framework* é baseado em Java e é utilizado para dar suporte a vários sistemas distribuídos (Lin 2009), principalmente aqueles que oferecem funcionalidades de automação residencial (Viana 2009). Também é usado para o desenvolvimento de sistemas de computação em grade, automação industrial e multimídia (Lin 2009). Possui diversas implementações comerciais e de código aberto (Lin 2009; Viana 2009).
- OSC – *Open Sound Control* é um formato para mensagens de rede utilizado para facilitar a comunicação entre computadores e dispositivos multimídia em geral. OSC foi originalmente concebido para troca de mensagens entre dispositivos envolvidos em uma performance musical, como instrumentos musicais eletrônicos (controladoras MIDI, sintetizadores etc.) e computadores munidos de *softwares* para produção musical. OSC é comumente utilizado sobre redes IP e diversas ferramentas estão disponíveis com o intuito de facilitar a criação de redes de dispositivos que se comunicam via OSC.
- Jini (*Apache River*[4]) – é uma arquitetura de *software* orientada a serviços que utiliza o ambiente de execução Java. Jini define um modelo de programação para a construção de sistemas distribuídos, oferecendo mecanismos para descoberta e estabelecimento de serviços, além de mobilidade de código Java.
- Bonjour[5] – é uma arquitetura de *software* e um conjunto protocolos de rede desenvolvidos pela Apple com o intuito de oferecer mecanismos para configuração automática de serviços em redes IP. Utiliza mDNS (*multicast Domain Name System*) para localização e identificação de dispositivos em uma rede local. É nativamente suportado pelos sistemas operacionais da Apple, Mac OS X e iOS, e também possui implementação para outras plataformas (incluindo Microsoft Windows e Linux).

[4] http://river.apache.org/
[5] http://www.apple.com/support/bonjour/

As tecnologias apresentadas possuem propósito funcional similar: facilitar a integração de recursos de dispositivos interconectados dentro da lógica de um sistema. Porém, as soluções supracitadas focam camadas diferentes da pilha de *software* e protocolos necessários para a comunicação em redes de computadores (considerando a classificação utilizada pelo modelo OSI - *Open Systems Interconnection*). Em alguns casos é possível interoperar essas tecnologias, como é o caso do suporte dado pelo OSGi a serviços UPnP (Kang 2005), mas ainda assim são muitas as dificuldades para a integração de dispositivos heterogêneos em rede as quais motivaram o desenvolvimento de várias pesquisas na área de sistemas distribuídos.

A quatro próximas subseções apresentarão resumidamente trabalhos que agregam um conjunto significativo de funcionalidades, à luz dos requisitos de alto nível apresentados na Seção 2.2. Os trabalhos foram organizados em subseções de acordo com os três espaços para assistência multimídia coletiva utilizados nos cenários de uso, além de uma seção adicional que agrega abordagens genéricas e de foco distinto. A Seção 3.5 finaliza a discussão sobre os trabalhos relacionados com um resumo comparativo dos trabalhos mais relevantes.

3.1. Hipermídia distribuída em aplicações de TV Digital interativa

O uso de múltiplos dispositivos em aplicações de TV Digital interativa já é vastamente explorado, inclusive comercialmente. Aplicações que oferecem o recurso popularmente conhecido como "segunda tela" (*second screen*) sofisticam a experiência de ver TV. A partir do uso de dispositivos portáteis, programas de TV são associados a aplicativos de redes sociais, de recomendação de conteúdo (Holmes 2012), e também há possibilidade de interação multimodal com o programa sendo exibido (Teixeira 2010).

Já existem também esforços para a definição de plataformas de *software* para receptores de TV Digital visando a integração com dispositivos secundários. Algumas especificações de *middleware* para sistemas de TV Digital oferecem funcionalidades com tal intuito e também as plataformas para TVs conectadas (como o Yahoo Connected TV (Cortez 2012) e o Google TV[6]). O conteúdo das

[6] http://www.google.com/tv/

próximas subseções analisará as abordagens mais relevantes dentro do contexto no qual se insere esta tese.

3.1.1. Aplicações distribuídas baseadas no *middleware* Ginga

Ginga[7] é o nome dado ao *middleware* para aplicações de TV Digital interativa, que é padrão do sistema ISDB-T (*International Standard for Digital Broadcasting*) (ABNT 2007) e recomendação ITU-T para serviços IPTV interativos (ITU-T 2009). As especificações do Ginga definem como requisito obrigatório a presença do subsistema Ginga-NCL (ABNT 2007; Soares 2007), que é responsável por executar aplicações declarativas escritas em NCL (*Nested Context Language*) (Soares 2005; Soares 2006; Soares 2007). A arquitetura definida para o Ginga define uma camada de *software* para abstração de exibidores de mídia, sistema operacional e *hardware*, chamada "*Ginga Common Core*" (Ginga-CC). Através desse núcleo comum, são manipulados os recursos oferecidos por um receptor de TV Digital (Soares 2007). No sistema ISDB-T o Ginga pode acomodar outro subsistema para execução de aplicações, o Ginga-J (Souza 2007), que é responsável por executar *Xlets* Java.

As definições do subsistema Ginga-NCL oferecem suporte declarativo a aplicações para múltiplos dispositivos. O subsistema é baseado na linguagem NCL, que especifica aplicações hipermídia a partir de definições para o sincronismo entre diferentes objetos de mídia (Soares 2007). A linguagem NCL permite que tais objetos sejam exibidos de forma distribuída, introduzindo o conceito de classes de dispositivos. No Ginga-NCL as classes de dispositivos agregam logicamente dispositivos pareados com o receptor de TV Digital, oferecendo uma abstração que é usada para associação da execução dos objetos de mídia aos dispositivos. Assim, os recursos oferecidos pelos dispositivos (telas, alto falantes) podem ser orquestrados de acordo com a lógica da aplicação NCL sendo executada em um receptor de TV Digital ISDB-T (ABNT 2007).

O subsistema Ginga-NCL (ABNT 2007; Soares 2007) define dois tipos de classes de dispositivos para a apresentação de aplicações NCL distribuídas. No primeiro tipo, um mesmo conteúdo é apresentado nos dispositivos registrados na

[7] http://www.ginga.org.br

classe, sobre controle de navegação único; o segundo tipo permite que os dispositivos registrados na classe controlem a apresentação do seu conteúdo associado de forma independente, i.e. o controle de navegação é individualizado. O primeiro tipo define uma classe chamada passiva, enquanto o segundo define uma classe ativa. Subclasses podem ser definidas como extensões das classes ativa e passiva (Soares 2009).

Dispositivos secundários se registram em classes junto a um dispositivo base (que é a raiz única da árvore de comunicação hierárquica) ou dispositivos descendentes (recursivamente registrados) do dispositivo base e formam, junto ao dispositivo pai, um domínio. É chamado de dispositivo pai do dispositivo registrado aquele onde o registro é efetuado (que oferece o serviço de pareamento). No modelo de controle hierárquico da NCL dispositivos em uma classe só podem exibir conteúdos de objetos de mídia vindos do mesmo dispositivo pai, independente do tipo de classe. Uma classe, portanto, não pode ter mais de um dispositivo pai por vez. Adicionalmente, o dispositivo base não pode se registrar em nenhum outro dispositivo do domínio. Portanto, não é possível ter um dispositivo como ascendente ou descendente de si mesmo (Soares 2009).

O dispositivo base orquestra a exibição da aplicação utilizando um modelo hierárquico. Um dispositivo pai deve transmitir amostras de áudio e/ou vídeo para serem exibidas pelos dispositivos que gerencia e estão registrados em classes passivas, de forma que todos esses dispositivos apresentem sempre o mesmo conteúdo. No caso de classes ativas, o dispositivo pai deverá transmitir para os dispositivos que gerencia (dispositivos filhos) partes da aplicação (composta por objetos de mídia, inclusive código NCL) para que sejam decodificadas, apresentadas e localmente controladas. É importante ressaltar que o modelo permite a criação de subdomínios a partir de aplicações NCL associadas à dispositivos ativos, de forma que parte das mesmas possa ser distribuída novamente para outros dispositivos descendentes, recorrentemente (Costa 2009; Soares 2009).

As funcionalidades da NCL relacionadas à utilização de recursos de dispositivos secundários são suportadas por módulos específicos na arquitetura da implementação de referência do Ginga. Os módulos que compõem o componente de integração com múltiplos dispositivos são responsáveis pelo registro de

dispositivos e pela comunicação entre dispositivos, para apresentação de um documento multimídia distribuído (Costa 2009; Soares 2009).

O componente Controlador de Serviços manipula os serviços dos dispositivos em um domínio, que é definido pelo conjunto dos dispositivos em suas classes. O Gerenciador de Dispositivos (Costa 2009; Soares 2009) controla o registro de dispositivos junto a um conjunto de classes de dispositivo pré-definidas: as classes “0”, “1” e “2”. A classe "0” é associada apenas ao dispositivo base, que executa o código NCL inicial. A classe passiva "1" agrega dispositivos registrados que devem ser capazes de receber fluxos de mídia vídeo e áudio decodificados, prontos para serem exibidos e navegados em grupo. A classe ativa "2" registra dispositivos capazes de receber qualquer objeto de mídia NCL especificado pelo padrão Ginga-NCL (Costa 2009; Soares 2009) para apresentação de forma independente. A implementação de referência original (para Linux) oferecia também o modo de execução como dispositivo secundário passivo. Uma versão de cliente passivo para dispositivos iOS foi desenvolvida e utilizada para testes da especificação do *middleware* Ginga (Costa 2009; Soares 2009).

O Gerenciador de Dispositivos da implementação de referência do Ginga guarda todas as informações relativas à configuração dos dispositivos para todas as classes registradas, bem como a associação dos dispositivos às classes. Informações sobre os recursos dos dispositivos que serão utilizados pelas aplicações NCL (ex. tamanho de tela, número de canais de áudio etc.) estão associados a um conjunto de variáveis de ambiente do Ginga-NCL, que podem ser acessadas através do seu "nó *settings*" (nó de definições) (Costa 2009; Soares 2009). Essas informações podem ser utilizadas para adaptação de aplicações às características de uma determinada classe de dispositivos (Costa 2009; Soares 2009), porém não é possível especificar outras características específicas para cada classe de dispositivo.

O Controlador de Serviços é responsável por realizar as ações necessárias para que objetos de mídia possam ser apresentados em dispositivos secundários registrados nas classes ativas e passivas (Costa 2009; Soares 2009). Para as classes passivas, o controlador deverá acessar o conteúdo já decodificado do objeto de mídia a ser apresentado remotamente (o mapa de memória de vídeo, por exemplo) e utilizar o componente de transporte para entrega do objeto

decodificado. Já para classes ativas, o controlador envia aos dispositivos da classe objetos de mídia por completo, em sua codificação original (uma aplicação NCL e suas mídias associadas, por exemplo). Na implementação de referência anterior a esta tese, o dispositivo pai recebia apenas notificações dos eventos para o formatador do Ginga-NCL (eventos de apresentação, de seleção e de atribuição) – não era possível, por exemplo, reproduzir no dispositivo base um fluxo de vídeo ou um trecho de áudio capturado por um dispositivo filho.

Até a conclusão desta tese, apenas a implementação de referência do Ginga oferecia o suporte à execução de aplicações NCL em múltiplos dispositivos. A implementação de referência oferecia registro de dispositivos apenas nas classes pré-definidas "1" e "2", através de um protocolo específico, que define mecanismos para a busca e estabelecimento de serviços em redes locais (IP). Um modelo para descrição de outras classes não foi alvo das especificações do Ginga-NCL, podendo ser definido pelo fabricante do dispositivo base, sendo atrelado a uma implementação específica do Ginga-NCL.

Mecanismos para interoperação com outras tecnologias e plataformas de integração de dispositivos não são definidos pelo Ginga-NCL e também não foram explorados na implementação de referência (Costa 2009; Soares 2009). Cabe ressaltar que tais limitações não são do middleware Ginga-NCL, mas sim de sua implementação de referência original (Batista 2010). Implementações comerciais podem apresentar soluções próprias para cada questão mencionada. Como será visto posteriormente (no Capítulo 5), o desenvolvimento desta tese incluiu a extensão da implementação de referência do subsistema Ginga-NCL, integrando-a a plataforma para hipermídia distribuída proposta.

Alguns trabalhos presentes na literatura relacionada propõem extensões ao suporte a múltiplos dispositivos da implementação de referência do subsistema Ginga-NCL, como por exemplo, o Ginga WaC (*Watch and Comment*) (Teixeira 2010), que acrescenta ao subsistema mecanismos para anotação de conteúdo a partir de dispositivos portáteis, através de uma API Lua (Ierusalimschy 2006) e de um serviço UPnP específico. Outros trabalhos exploram características oferecidas pelo subsistema que estende o Ginga nas definições do ISDB-T, o Ginga-J.

O subsistema Ginga-J (Souza 2007) é responsável por executar aplicações (*Xlets*) Java e possui uma especificação de API opcional para construção de aplicações para múltiplos dispositivos (Silva 2007; Silva 2008) (pacote

br.org.sbtvd.interactiondevices). A classe Java responsável pela gerência dos dispositivos conectados a um receptor de TV Digital com Ginga-J é a *GRemoteDeviceManager*. Durante a execução de um *Xlet* no Ginga-J, uma instância dessa classe armazena uma lista dos dispositivos registrados junto ao receptor de TV Digital, e cada dispositivo é representado por uma instância da classe *GRemoteDevice*.

A classe *GRemoteDevice* representa um dispositivo de interação, oferecendo métodos que possibilitam recuperar de informações acerca dos dispositivos registrados (tipo do dispositivo, recursos disponíveis etc.) e explorar suas funcionalidades (recursos de gravação de áudio, de vídeo, captura de imagens). Cada instância de *GRemoteDevice* referencia uma instância de uma classe contêiner da API gráfica do Ginga, que pode ser utilizada para compor interfaces gráficas para exibição nos dispositivos secundários, enquanto controlada pelo receptor de TV. Originalmente compatível com a família de *middlewares* para TV Digital GEM (*Globally Executable MHP*) (ETSI 2005), o Ginga-J incorporava em sua primeira especificação a API gráfica HAVi (ETSI 2005). A especificação corrente incorpora a API JavaDTV (Kulesza 2011), referenciando a classe contêiner *DTVContainer*. Ouvintes (*GRemoteDeviceActionListener*) podem ser registrados junto às instâncias de *GRemoteDevice* para notificação de eventos do usuário capturados pelos dispositivos. Instâncias desses eventos (*GRemoteEvent*) podem encapsular dados de áudio e vídeo ou códigos de teclas pressionadas (Silva 2008).

Não há na especificação do pacote *br.org.sbtvd.interactiondevices* mecanismos voltados para comunicação em grupo e nem mecanismos específicos de tolerância à falhas durante as aplicações distribuídas. A API não possui funcionalidades específicas para o sincronismo entre mídias distribuídas e a captura de objetos de mídia por parte dos dispositivos é feita através de chamadas assíncronas, onde amostras de mídia podem ser recuperadas dos dispositivos de interação para posterior decodificação no dispositivo receptor de TV Digital. Os dispositivos secundários devem possuir uma implementação cliente compatível com um protocolo que não faz parte da especificação (Silva 2008). A API foi validada através de um protótipo, e é parte opcional do Ginga-J (Silva 2007; Silva 2008). Não há atualmente nenhuma implementação comercial da API de integração de dispositivos do Ginga-J presente em um receptor de TV com Ginga.

O único protótipo existente utilizou um protocolo de comunicação simples, que foi testado sobre redes IEEE 802.11 e Bluetooth, integrando dispositivos Symbian munidos de uma implementação cliente. O protótipo não oferece integração dinâmica de dispositivos, assumindo que uma aplicação residente (para configuração do *middleware*) registre os dispositivos (Silva 2007; Silva 2008).

Extensões para o middleware Ginga-J foram propostas no intuito de facilitar a integração com dispositivos computacionais presentes em redes domésticas. No trabalho intitulado Ginga-OSGi (Viana 2009), foi desenvolvida uma camada de interoperabilidade do Ginga com a plataforma OSGi. A integração das API Java permite que serviços OSGi sejam acessados através de aplicações Ginga-J. Também foi desenvolvido um componente que oferece mecanismos para consulta de estado e acesso a serviços OSGi a partir de aplicações executadas no Ginga-NCL. O protótipo desenvolvido possibilitou que aplicações Ginga pudessem incorporar funcionalidades para automação residencial. Duas aplicações validaram o modelo: uma aplicação permitia o acesso a serviços de impressão e outra possibilitava a exibição do conteúdo capturado por uma câmera de segurança e também controlava uma fechadura eletrônica (para controle de acesso (Viana 2009)) .

3.1.2. Aplicações distribuídas baseadas no *middleware* MHP

MHP (*Multimedia Home Platform*) é o nome do *middleware* que suporta aplicações de TV Digital interativa no sistema de TV Digital europeu DVB (*Digital Video Broadcasting*). MHP foi utilizado como base para definir uma família de *middlewares* chamada GEM (*Globally Executable MHP*), que inclui o *middleware* americano ACAP (*Advanced Common Application Platform*) (ATSC 2005) e o japonês ARIB B.23 (*Application Execution Engine Platform for Digital Broadcasting*) (ARIB 2004).

Apesar de embutirem as API gráficas do *middleware* HAVi (ETSI 2005) (que é voltado para integração de dispositivos de automação residencial), as especificações de *middleware* baseadas no MHP não oferecem funcionalidades voltadas para o desenvolvimento de aplicações que integrem recursos de dispositivos em uma rede doméstica.

A integração de receptores GEM com dispositivos foi o mote de uma série de trabalhos acadêmicos (Lin 2009). Alguns trabalhos definem protocolos e API própria, como é o caso do HoNeY, que oferece suporte a integração de dispositivos de automação residencial, a partir da definição de um conjunto de protocolos e serviços (Forno 2006). Grande parte dos trabalhos relevantes, porém, define uma camada de interoperabilidade entre o MHP e a plataforma de integração de dispositivos OSGi (Lin 2009).

A relação entre o *middleware* MHP e a plataforma OSGi é documentada em (Lin 2009), que compara e classifica diferentes abordagens arquiteturais para a cooperação entre os dois sistemas. Como MHP e OSGi dependem do ambiente de execução Java, o trabalho categoriza as abordagens de acordo com a disposição dos elementos de *software* com relação à máquina virtual Java (Lin 2009) e diferentes abordagens para a harmonização dos modelos de ciclo de vida de aplicações (*bundles*) OSGi e os *Xlets* MHP. Não há nesses trabalhos suporte à exibição sincronizada de objetos de mídia em dispositivos distribuídos. Não foram criados mecanismos específicos para a tolerância de falhas de sincronismo entre objetos de mídia distribuídos, e a integração dinâmica de dispositivos em todos os casos obedece ao modelo definido pelo OSGi (Lin 2009).

3.1.3. Especificação de aplicações distribuídas no *middleware* ARIB

Uma API para a integração de dispositivos é definida pela especificação do *middleware* ARIB STD-B23 (*Application Execution Engine Platform for Digital Broadcasting*) (ARIB 2004), e é destinada a receptores de TV Digital compatíveis com o sistema japonês ISDB (*Integrated Services Digital Broadcasting*).

O pacote núcleo dessa API Java (*jp.or.arib.tv.peripheral* (ARIB 2004)) é composto de classes e interfaces com funcionalidades para descoberta e registro de dispositivos, obtenção de propriedades e estado dos dispositivos, além de funcionalidades para envio e recebimento de mensagens. A API oferece alguns serviços compatíveis com UPnP (pacote *jp.or.arib.tv.peripheral.protocol*), e registro de dispositivos através de Bluetooth e USB. Não oferece recursos específicos para sincronismo entre mídias distribuídas e nem de adaptação de conteúdo e apresentação. Não há, até o limite das pesquisas realizadas durante o

desenvolvimento desta tese, nenhum registro de uma implementação completa dessa API para receptores ARIB.

3.1.4. Yahoo Connected TV

Yahoo Connected TV (Cortez 2012) é o nome da plataforma de aplicações interativas para TVs conectadas desenvolvida pela Yahoo. Já presente em vários modelos de TV de diferentes fabricantes, utiliza tecnologias *Web* (HTML e *JavaScript*) (Cortez 2012) para oferecer suas funcionalidades aos desenvolvedores. Inclui uma API para integração de dispositivos, que permite o desenvolvimento de aplicações multiusuário, com capacidade de interpretar gestos e de utilizar múltiplas telas para exibição de conteúdo.

A plataforma possui um protocolo de comunicação próprio, que é utilizado para troca de mensagens entre os dispositivos durante a execução das aplicações. As mensagens carregam informações de interação (teclas de navegação) e dados arbitrários (em formato de texto apenas, i.e. não permite troca de objetos de mídia entre dispositivos).

A API do Yahoo Connected TV não oferece mecanismos de interoperabilidade com outras plataformas para aplicações multi-dispositivo. A plataforma da Yahoo possui um módulo responsável pela manutenção e estabelecimento dos canais de comunicação entre os dispositivos, utilizando a biblioteca mDNSResponder (Cortez 2012) (parte da arquitetura Bonjour). Os serviços podem utilizar comunicação segura entre os dispositivos envolvidos, através do uso de conexões SSL (*Secure Socket Layer*) (Cortez 2012).

3.1.5. Google TV

Google TV[8] é uma plataforma de *software* para TVs conectadas desenvolvida pelo Google em parceria com fabricantes de eletrônicos de consumo. Google TV é baseado no sistema operacional Android e no navegador Chrome e utiliza API própria para o suporte à execução de aplicações interativas em TVs conectadas. Através da plataforma do Google é possível utilizar

[8] http://www.google.com/tv/

dispositivos móveis como um controle remoto avançado, em aplicações que utilizam o conceito de segunda tela ("*second screen*"). A API para aplicações Google TV permite que um celular ou *tablet* seja usado como controle de cursor ou teclado da TV, além da possibilidade de captura de voz para busca por conteúdo. Aplicações Google TV podem compartilhar referências para páginas da *Web* e vídeos com os dispositivos em uma rede doméstica.

Em uma arquitetura cliente-servidor, a aplicação cliente (Google TV Android Remote) é utilizada para integrar dispositivos móveis (celulares e tablets) e dispositivos Google TV (*set-top boxes* e TVs, que atuam como servidores). A aplicação cliente está disponível para dispositivos Android e iOS . O pareamento entre os dispositivos é feito através do protocolo *Google TV Pairing* (com especificação e implementação de referência aberta), que dá suporte para o estabelecimento serviços de rede para troca de mensagens. As mensagens são codificadas de acordo com o protocolo Anymote, que encapsula informações referentes a diferentes tipos de interação suportados pelos dispositivos móveis integráveis ao Google TV.

O processo de pareamento é realizado através da solução de um "*challenge*" (desafio). Um *challenge* é um processo de interação que envolve os dispositivos Google TV, no qual um usuário captura uma informação oferecida pela TV em seu dispositivo móvel, de acordo com as capacidades oferecidas pelo dispositivo (i.e. os tipos de entrada capturados pelo dispositivo). Quando um dispositivo cliente deseja se parear com o dispositivo servidor Google TV ele deve notificar quais tipos de informação pode capturar utilizando o Google TV Pairing (e.g. se possuir uma câmera poderá capturar QRCodes, se possuir teclado pode capturar eventos de teclas etc.), definindo quais tipos de *challenge* poderá responder. O dispositivo Google TV servidor cria um *challenge* que deve ser respondido corretamente pelo dispositivo cliente (e.g. dispositivo captura um código QRCode exibido na tela, dispositivo digita texto exibido na tela etc.) para que ele possa interagir com aplicações Google TV.

O protocolo Anymote permite que dispositivos cliente enviem notificações de eventos de interação para uma aplicação Google TV. Após o pareamento, tanto o sistema cliente quanto o servidor possuem certificados específicos para uma aplicação, e podem se comunicar outras vezes sem nova autenticação (o pareamento é persistente). A camada de transporte usa TSL/SSL para criar um

canal seguro para troca das mensagens. O protocolo Anymote descreve mensagens específicas para encapsular informações de descrição dos dispositivos, códigos de teclas e de seleção via cursor, além de *strings* arbitrárias (que podem encapsular sequências de teclas ou comandos de semântica arbitrária) e *Flings*. *Flings* são estruturas de dados que encapsulam informações referentes à ativação de exibidores de mídia nos dispositivos cliente (i.e. para a exibição de um vídeo do Youtube ou de uma página *Web*). Não há na atual versão da API e dos protocolos suporte a interação a partir de mídias capturadas pelos dispositivos.

Apesar dos dispositivos Google TV possuírem certificação DLNA[9], sua API não contempla funcionalidades de integração com os mecanismos oferecidos por tais tecnologias (como serviços UPnP). A comunicação e registro sempre são realizados através de suas API e protocolos próprios. A API de integração com dispositivos do Google TV possui funções para verificação do estado da conexão entre os dispositivos envolvidos, porém não oferece funcionalidades específicas para o sincronismo entre objetos de mídia distribuídos, nem para tolerância às falhas de sincronismo em objetos distribuídos.

3.1.6. neXtream

O projeto neXtream (Martin 2010) propõe uma abordagem particular para entrega de conteúdo audiovisual, através da integração de dispositivos móveis, redes sociais e provedores de conteúdo de TV. Durante o projeto foi desenvolvido um *framework* e uma aplicação que oferece recursos para personalização de conteúdo e também interação social entre os usuários. O *framework* oferece a possibilidade de diferentes mecanismos para entrega de conteúdo (para TV, PC e dispositivos móveis) e para interatividade a partir dos dispositivos conectados.

Um protótipo (Martin 2010) foi implementado, com um módulo para Apple TV e outro para dispositivos móveis (*smartphones* Apple iPhone). Bonjour foi utilizado para descoberta e estabelecimento dos serviços de interatividade sobre uma rede IP, e a pacotes OSC são usados para troca de mensagens entre os dispositivos.

[9] http://www.dlna.org/

O dispositivo secundário é usado como controle para navegação entre as mídias exibidas pelo receptor de TV, e também para utilização dos mecanismos de interação nas redes sociais. O protótipo foi implementado em Objective-C para plataforma iOS, e não há informações sobre um formato que permita a definição de diferentes aplicações, com funcionalidades limitadas às oferecidas pelo protótipo (Martin 2010). Não há mecanismos para interoperabilidade, nem específicos para adaptação de apresentação. Não há também tratamento de falhas de sincronismo durante a execução da aplicação distribuída.

3.1.7. MDCS

O MDCS (*Multimedia Delivery and Control System*) (Kernchen 2010) é um sistema para geração dinâmica de apresentações multimídia, com foco em adaptação de conteúdo para múltiplos dispositivos e mobilidade para as sessões de visualização. O MDCS possui uma implementação de código aberto (Kernchen 2010) que facilita a construção de aplicações multimídia que integram dispositivos presentes em um ambiente doméstico.

O sistema gera dinamicamente o código de uma aplicação SMIL (Bulterman 2004; W3C 2008), a partir da escolha por uma configuração ótima de dispositivos, associando as características dos objetos de mídia utilizados pela aplicação com as capacidades e restrições dos dispositivos. O sistema adapta dinamicamente a execução de acordo com as mudanças na configuração dos dispositivos utilizados. Um cenário de exemplo: quando um usuário deixa a sala de TV, o conteúdo visto na tela do receptor passa a ser exibido no seu dispositivo móvel (Kernchen 2010).

A arquitetura do MDCS possui uma camada para abstração dos mecanismos de descrição dos dispositivos e seus recursos, podendo integrar diferentes modelos de descrição (Kernchen 2010). Tal característica possibilita a interoperabilidade com diferentes formatos de descrição de dispositivos, porém a implementação não explora outros mecanismos de interoperação com outras plataformas para integração de dispositivos (via protocolos de rede ou API, por exemplo). O sistema permite seleção de conteúdo através dos dispositivos móveis.

As aplicações SMIL geradas a partir da avaliação do contexto de execução não tratam possibilidades arbitrárias de interações com o usuário, constituindo

apenas apresentações lineares. O MDCS adapta conteúdo em função de diferentes ambientes de execução, mas não apresenta funcionalidades específicas para a integração dinâmica de dispositivos nesses ambientes. Não há também mecanismos para tratamento de falhas de sincronismo durante a execução de uma aplicação distribuída.

3.2. Hipermídia distribuída em aplicações de Cinema Digital interativo

Cinema interativo e instalações de cinema digital performático estão cada vez mais populares. Sistemas de cinema interativo estão explorando cada vez mais recursos tecnológicos para oferecer uma experiência onde cada usuário possa interferir na trama compartilhada por todos em uma sala de cinema (Lew 2004; Vasilakos 2008). A maioria das abordagens documentada na literatura científica são realizações na forma de instalação, onde mecanismos de interação não reaproveitáveis são utilizadas para criar uma experiência de filme interativo.

"*Quanticum Man*" (Vasilakos 2008) é uma instalação que usa de realidade mista (Realidade Virtual associada com captura 3D, em tempo real) e ambiência inteligente para oferecer uma experiência de cinema interativo. "*Last Call*" foi um experimento de cinema interativo[10] realizado pelo canal de *TV 13st street*[11] , onde um dos integrantes da plateia é selecionado em determinadas partes do filme para determinar certas escolhas realizadas pelos protagonistas na trama. "*Last Call*" permitia a interação com o filme através da voz capturada dos aparelhos celulares – a retaguarda responsável por processar os comandos de voz e modificar o andamento do filme projetado foi composta pelos softwares AixVox e Powerflasher[12].

Não há dentre as abordagens encontradas durante a pesquisa aqui relatada nenhum formato de aplicação ou sistema que se proponha a descrever filmes interativos para espaços onde múltiplos usuários possuem dispositivos computacionais. Algumas das abordagens encontradas na literatura, porém, suprem alguns dos requisitos de alto nível extraídos dos cenários apresentados na seção anterior, possibilitando Cinema com interatividade multiusuário.

[10] http://www.jvm.com/
[11] http://www.13thstreetuniversal.nl/
[12] http://www.filmdeluxe.com/de/

O projeto AVIE (McGinity 2007) criou uma sala de cinema interativo com uma tela de 360°. A sala é capaz de capturar movimentos dos usuários, permitindo assim que os mesmos interajam com a projeção. Um cluster de seis PCs *dual Xeon* (Windows) é usado para controlar doze projetores. Doze câmeras com lentes infravermelhas são distribuídas no espaço da tela de 360°, cobrindo toda a área disponível para os usuários. Um *cluster* com quatro PCs (Linux) executa dois algoritmos de captura de movimento diferentes em paralelo, funcionando como interface primária de entrada, que é ativada através de gestos e movimentos dos usuários. O pacote de *software* Virtools (McGinity 2007) foi usado como base para o sistema, e diferentes aplicações foram desenvolvidas para a sala interativa, fazendo com que o AVIE seja uma plataforma também para planejamento em arquitetura, simulação de situações de perigo (em ambientes como minas de carvão), além de outras aplicações lúdicas (como jogos) (McGinity 2007).

O sistema Cinematrix (Maynes-Aminzade 2002) captura parâmetros limitados de interatividade para que seus usuários possam fazer escolhas que interferem em um filme interativo, através da exibição de diferentes lados de uma placa sinalizadora especial. Outras instalações como o Carrossel (Caires 2007) e sistemas como o LiveCinema (Lew 2004) oferecem mecanismos para que seus usuários alterem o andamento da projeção de um vídeo em uma tela de cinema convencional.

A instalação Carrossel (Caires 2007) oferece como interface de entrada um dispositivo modelado como uma caixa de música (um carrossel), que pode ser manipulado por um usuário apenas. A projeção reage à velocidade e direção da manipulação do carrossel através da informação capturada por um sensor PowerMate (Caires 2007) e tratada pelo computador responsável pela projeção. Utilizando o *software* Pure Data as informações recebidas são processadas e utilizadas para alterar o vídeo sendo projetado (Caires 2007).

O projeto LiveCinema desenvolveu um sistema para edição em tempo de exibição do conteúdo projetado em uma tela de cinema. O sistema protótipo, desenvolvido para a plataforma Mac OS X (Lew 2004), possui uma interface para seleção de trechos de vídeo e uma ferramenta de edição controlada por toques em uma tela (*touch screen*). Um dispositivo de interação na forma de uma plataforma giratória foi construído (similar a um toca discos (Lew 2004)), e é utilizado para controle do andamento dos trechos de vídeo em exibição.

3.3. Hipermídia distribuída em apresentações de eslaides multimídia

Sistemas criados para dar suporte às apresentações baseadas em eslaides digitais multimídia já suportam mecanismos para interação com múltiplos usuários e múltiplos dispositivos. Abordagens diferentes para a execução distribuída de tais apresentações são encontradas na literatura. Discutiremos alguns trabalhos que suprem requisitos relevantes face ao proposto nesta tese.

A extensão de sistemas de apresentação originalmente não distribuídos é explorada em algumas abordagens. O DiS (*Distributed Slideshow*) (Gaspar 2007) estende o Microsoft PowerPoint, oferecendo a possibilidade de sincronismo da apresentação de um conjunto de eslaides em diferentes computadores interconectados. O sistema oferece a possibilidade de controle do andamento da apresentação a partir de um dos computadores integrados – somente comandos podem ser enviados, não há possibilidade de troca de mensagens contendo objetos de mídia. O DiS utiliza IP *multicast* para troca de mensagens de sincronismo, e SDP (*Service Discovery Protocol*) para anúncio e descrição das sessões de apresentação. As mensagens de sincronismo do DiS são codificadas em XML, e a transmissão dos arquivos da apresentação é feita, a partir de um dispositivo raiz, utilizando MFTP (*Multicast File Transfer Protocol*) (Gaspar 2007).

O *middleware* Moheet (Al-Muhtadi 2006) foi desenvolvido para ser uma solução para construção de espaços inteligentes através de sistemas ubíquos. A plataforma de execução foi desenvolvida em Java (J2ME), utilizando RMI para construção de aplicações distribuídas e Jini para descoberta e estabelecimento de serviços de rede entre os dispositivos (Al-Muhtadi 2006). Aplicações podem ser descritas utilizando scripts Lua, que definem a lógica através da qual os objetos e aplicações Java nos dispositivos devem interagir. O protótipo desenvolvido implementa a apresentação de eslaides multimídia distribuídos. O controle do andamento da apresentação distribuído é feito a partir dos dispositivos móveis, que também podem ser utilizados para visualização de conteúdo extra e para realizar anotações. A implementação, porém, não apresenta mecanismos específicos para sincronismo entre objetos de mídia distribuídos, nem de recuperação e tolerância a falhas durante a apresentação (Al-Muhtadi 2006).

O sistema Active Presentation (Glasberg 2004) cria um espaço ativo para realização de apresentações distribuídas baseada em eslaides multimídia, utilizando o paradigma de computação ubíqua. A infraestrutura para as apresentações foi concebida a partir do desenvolvimento de cinco protótipos, que integraram exibidores multimídia como o Microsoft PowerPoint e o Windows Media Player. Os protótipos controlaram as funcionalidades dos exibidores através de suas interfaces COM e utilizaram mecanismos CORBA para distribuição de partes de uma aplicação, como parte de uma apresentação de eslaides. LuaOrb é utilizado como camada de interoperabilidade, permitindo a integração do sistema com outros *middlewares*.

Os protótipos do Active Presentation (Glasberg 2004) utilizaram dispositivos móveis, como celulares CDMA (implementação utilizando Brew) e também sensores (UbiSensors). Tais dispositivos foram integrados a uma plataforma de Navegação 3D desenvolvida pelo TecGraf da PUC-Rio. Para o desenvolvimento de um dos protótipos, o sistema HyperProp (Soares 2000) foi testado para controle da apresentação, o qual utilizava a linguagem NCL (versão 2.1) como formato de descrição de suas apresentações. A partir desse protótipo, uma versão simplificada do formatador do HyperProp foi desenvolvida em Lua, com suporte parcial aos recursos da NCL. Os protótipos permitiam pontos de sincronismos entre as mídias envolvidas. Havia, porém, a necessidade de indicação explícita do endereço de rede associado aos dispositivos que executam cada parte da apresentação (não há integração dinâmica durante a apresentação). Não foi oferecida interação multimodal distribuída, nem comunicação com grupos de dispositivos.

O trabalho desenvolvido por West, Foster e Clayton (West 2005), por outro lado, converte apresentações de eslaides do Microsoft PowerPoint para arquivos HTML que são disponibilizados para dispositivos conectados ao dispositivo controlador da apresentação através de um servidor HTTP. Tal abordagem permite a carga seletiva dos elementos da apresentação para visualização individualizada, porém sem mais funcionalidades para execução distribuída.

3.4. Outros sistemas de hipermídia distribuída relevantes

Foram identificados na literatura trabalhos que propuseram soluções para cenários diferentes dos abordados nesta tese, mas que também suprem um subconjunto parcial, porém relevante, dos requisitos de alto nível extraídos dos cenários definidos na seção anterior.

HTML5[13] especifica a possibilidade de construção de documentos que se comunicam assincronamente, especificados através de funções *JavaScript* da API *Cross-window messaging*. O trabalho proposto por Concolato et al. (Concolato 2011) apresenta um *framework* para comunicação entre documentos multimídia, o que permite que *widgets* MPEG-4 BIFS/XMT ou SVG possam se comunicar assincronamente. A proposta estende o exibidor GPAC[14] e define um modelo através do qual é possível associar-se documentos multimídia a serviços e mecanismos de descoberta de serviços (o protótipo utiliza UPnP). O modelo de comunicação utilizado por esses dois últimos ambientes de execução de aplicações hipermídia não é suficiente para acomodar os requisitos especificados por esta tese.

Alguns sistemas focam espaços onde há telas públicas. Uma tela é dita pública quando o espaço para sua visualização é de público acesso e suficientemente grande para comportar grupos de pessoas em trânsito (telão para publicidade em uma rua movimentada; telas para informação e publicidade presentes em aeroportos, lojas etc.) (Dix 2008). A interação com telas públicas é explorada através de mecanismos de interação compartilhados (como um teclado adaptado em uma calçada), e também a partir da identificação de presença e gestos (Dix 2008). Recentemente surgiram sistemas que se destinam a viabilizar mecanismos de interação entre dispositivos pessoais e telas públicas. Com diferentes características técnicas e níveis de interação e acoplamento (Izadi 2003; Dix 2008; Jagodic 2011), tais sistemas permitem a captura de informações a partir de dispositivos dinamicamente conectados ao sistema que controla uma tela pública.

O sistema MobiLenin (Scheible 2005) viabiliza que um grupo de pessoas em um mesmo espaço físico interaja com a apresentação de um vídeo com

[13] http://www.w3.org/html/wg/drafts/html/master/

múltiplas trilhas, exibido em uma tela grande e pública, a partir de seus celulares. O sistema foi modelado a partir de uma arquitetura cliente-servidor, onde a aplicação servidor reside no dispositivo responsável pelo controle da tela pública, e conecta os dispositivos portáteis através de uma aplicação cliente que oferece recursos para interação.

O protótipo do MobiLenin (Scheible 2005) utilizou um ambiente real (um bar) para validar as possibilidades de interação de múltiplos usuários com uma tela pública. Celulares com o *software* cliente instalado foram disponibilizados para alguns dos frequentadores presentes e por eles utilizados para seleção de quais trilhas em um clipe de música deveriam ser exibidas na tela (definindo ações para o protagonista do clipe). A aplicação teste incluiu o sorteio de brindes, visando incentivar o uso dos mecanismos de interatividade. O protótipo implementado não ofereceu mecanismos de interoperabilidade e o *software* cliente foi desenvolvido apenas para plataforma Symbian (e utilizado em um modelo de celular apenas: Nokia Série 60). Mecanismos de pareamento automático e de integração dinâmica de dispositivos não são explorados pelo MobiLenin. O sistema limita a natureza das informações que podem ser enviadas para o dispositivo controlador da tela pública (não há possibilidade de envio de fluxos nem objetos de mídia).

O sistema Dynamo (Izadi 2003) define mecanismos para o desenvolvimento de aplicações com interação entre múltiplos dispositivos, com abordagem genérica, não somente para espaços com telas compartilhadas. Utiliza um serviço de registro de dispositivos centralizado, que permite associação, reconfiguração e remoção em tempo real de serviços oferecidos por dispositivos. Na infraestrutura Dynamo (Izadi 2003), um dispositivo é representado por um serviço ou um conjunto de serviços que encapsula as funcionalidades por ele oferecidas. Os serviços dos dispositivos são descritos através de parâmetros com informações referentes à suas funcionalidades e restrições.

O Dynamo utiliza a abstração de bolha ("*bubble*" (Izadi 2003)) para representar porções autossuficientes da aplicação distribuída, que geralmente agrupam elementos fortemente acoplados (e.g. elementos de interface, elementos de controle de saída de áudio etc.). As bolhas são associadas dinamicamente aos

[14] http://gpac.wp.mines-telecom.fr/player/

dispositivos registrados de acordo com suas capacidades. Dynamo utiliza Jini e RMI para o estabelecimento e controle dos serviços oferecidos pelos dispositivos.

O Dynamo não possui funcionalidades específicas para o tratamento de sincronismo entre objetos de mídia, porém possui recursos para a apresentação de partes de uma aplicação em múltiplos dispositivos simultaneamente: define tipos "sincronizados" através de classes que encapsulam dados e que permitem que a modificação de seus membros seja propagada entre os dispositivos integrados. Funcionalidades de *proxying* (via Java Surrogate Architecture) são definidas para integração dispositivos que não são compatíveis com Jini (Izadi 2003).

O *middleware* ReWire (Vanderhulst 2008) oferece uma plataforma para desenvolvimento de aplicações distribuídas capazes de se adaptar a configurações de uso dinâmicas. O *middleware* oferece uma camada de abstração para mecanismos OSGi, acrescentando uma camada de serviços de *software* descritos semanticamente. A comunicação entre dispositivos se dá através da camada de comunicação (W2P (Vanderhulst 2008)), com informações codificadas em XML e utilizando o protocolo HTTP. O uso tecnologias para descrição semântica, como RDF (*Resource Description Framework*) e OWL (*Web Ontology Language*), além da linguagem de consulta SPARQL, acrescentam semântica descritiva para permitir a interoperação de componentes OSGi com outros serviços arbitrários. O protótipo desenvolvido utiliza páginas HTML para oferecer recursos disponibilizados por dispositivos dinamicamente integrados ao ambiente de execução OSGi. Não oferece funcionalidades específicas para manutenção do sincronismo entre objetos de mídia distribuídos, nem para captura e manipulação de mídias.

3.5. Resumo comparativo

A tabela apresentada a seguir (Tabela 1) pretende materializar um panorama com as soluções mais relevantes para suprir os requisitos que sustentam os cenários propostos anteriormente. O objetivo é realçar quais características necessárias para a concretização dos cenários apresentados no Capítulo 2 estão além do estado da arte.

Na tabela, quando um requisito é total ou parcialmente suprido, há uma breve referência ao que foi apresentado em cada abordagem. Um parâmetro adicional foi considerado na tabela: qual o escopo da solução (se é um *middleware*, um *framework*, uma API, uma instalação dedicada etc.). O escopo da solução é, quando necessário, associado à forma como são descritas as aplicações hipermídia distribuídas (formato de descrição da aplicação).

	Escopo (Formato)	Modelo de Orquestração Distribuída	Integração dinâmica	Comunicação	Sincronismo intermídia distribuído	Interação Multimodal Distribuída	Adaptação	Interoperabilidade	Tolerância a falhas
Ginga-J (Silva 2008)	*Middleware* (Aplicação Java TV - *Xlet*)	Hierárquico, dois níveis apenas.	-	Unicast.	-	Amostras de mídia são capturadas, porém apenas para reprodução.	Via código imperativo (Java).	Não oferece em suas especificações. Alguns trabalhos exploram (Viana 2009).	-
Ginga-NCL (Soares 2009) versão pré-tese	*Middleware* (Documento NCL)	Hierárquico, múltiplos níveis (implementação com dois)	Através de mecanismos não flexíveis.	Multicast.	Possível, via código declarativo (NCL).	Possibilita captura de eventos de toque, mas não trata informação multimodais.	Via código declarativo (NCL) e código Lua	Não oferece em suas especificações. Alguns trabalhos exploram (Teixeira 2010).	-
ARIB B.23 (ARIB 2004)	*Middleware* (especificação) (Aplicação Java TV - *Xlet*)	Hierárquico, dois níveis apenas.	Via UPnP, USB e Bluetooth.	Unicast e Multicast.	-	-	Via código imperativo (Java).	Com USB, UPnP e Bluetooth.	-
MHP + HoNeY (Forno 2006)	*Middleware* (Aplicação Java TV - *Xlet*)	Hierárquico, dois níveis apenas.	Via mecanismos e protocolos HoNeY	Unicast e Multicast	-	-	Via código imperativo (Java).	-	-
MHP + OSGi (Lin 2009)	*Middleware* (Aplicação Java TV - *Xlet*)	Hierárquico, múltiplos níveis.	Via OSGi *Device Access*.	Unicast e Multicast.	-	-	Via código imperativo (Java).	Com OSGi.	-
neXtream (Martin 2010)	Aplicação residente em objective C	Hierárquico, dois níveis apenas.	Bonjour.	Unicast.	-	-	Via código imperativo (Objective C).	Com Bonjour e OSC.	-
MDCS (Kernchen 2010)	*Framework* (Aplicação SMIL)	Hierárquico, dois níveis apenas.	-	Unicast.	Possível, via código declarativo (SMIL).	-	Via código declarativo (SMIL).	-	-
Yahoo TV (Cortez 2012)	API (Aplicação HTML + JavaScript)	Hierárquico, dois níveis apenas.	Via mDNSResponder.	Unicast.	Possível, via código imperativo (JavaScript).	Possibilita a captura de gestos, mas protocolo não lida com informações multimodais.	Via CSS e JavaScript	-	-
Google TV	API e Protocolos (Aplicação Android e iOS)	Hierárquico, dois níveis apenas.	Via Google TV Pairing Protocol	Unicast	Possível, via código imperativo (Java, iOS)	Possibilita captura de toques e busca por voz, mas não lida com informações multimodais.	Via código imperativo (Java e iOS)	GoogleTV Pairing Protocol, Google TV Anymote Protocol	-
AVIE (McGinity 2007)	Instalação / Aplicação Virtools	Não utiliza dispositivos secundários.	-	Captura de movimentos dos usuários.	-	-	-	-	-

	Escopo (Formato)	Modelo de Orquestração Distribuída	Integração dinâmica	Comunicação	Sincronismo intermídia distribuído	Interação Multimodal Distribuída	Adaptação	Interoperabilidade	Tolerância a falhas
Cinematrix (Maynes-Aminzade 2002)	Instalação	Não se aplica.	Sim (placas sinalizadoras).	Captura de informações visuais.	-	-	-	-	-
Carrossel (Caires 2007)	Instalação / Aplicação Puredata	Hierárquico, dois níveis apenas	-	Unicast	-	-	-	-	-
LiveCinema (Lew 2004)	Instalação / Aplicação OSX	Hierárquico, dois níveis apenas	-	Unicast	-	-	-	-	-
DiS (Gaspar 2007)	*Framework* (Apresentação PowerPoint)	Hierárquico, dois níveis apenas.	Via SDP e mecanismo próprio de pareamento	Multicast.	Possível, apresentação de eslaides distribuída e sincronizada.	-	-	Com SDP e XML	-
Moheet (Al-Muhtadi 2006)	*Middleware* (Aplicação Java)	Hierárquico, dois níveis apenas.	Via Jini	Unicast e Multicast	-	-	Via código imperativo (Java/J2ME)	Com Jini e RMI	-
Active Presentation (Glasberg 2004)	*Framework* (Documento NCL)	Hierárquico, dois níveis apenas.	-	Unicast.	Possível, através de código declarativo (NCL)	-	Via código declarativo (NCL)	Com LuaOrb	-
MobiLenin (Scheible 2005)	Aplicação cliente e servidor (Symbian)	Hierárquico, dois níveis apenas.	-	Unicast	Sim, entre vídeo na tela pública e aplicação nos dispositivos.	-	-	-	-
ReWire (Vanderhulst 2008)	*Middleware* (Aplicação Java)	Não hierárquico.	Via OSGi *Device Access*	Unicast e Multicast.	-	-	Via código imperativo (Java)	Com OSGi	-
Dynamo (Izadi 2003)	*Middleware* (Aplicação Java)	Não hierárquico.	Via Jini	Unicast e Multicast.	-	-	Via código imperativo (Java)	Com Jini e RMI	-
Ginga-NCL versão pós-tese	*Middleware* (Documento NCL)	Hierárquico, múltiplos níveis (aplicações com três níveis)	Mecanismos flexíveis (API). Protocolo próprio e UPnP foram integrados,	Unicast, Broadcast e Multicast.	Possível, via código declarativo (NCL).	Sinais de orquestração, aplicações hipermídia e fluxos de mídia podem ser enviados entre dispositivos.	Via código declarativo (NCL) e código Lua	Com UPnP; oferece API e protocolos para integração.	Através de mecanismos automáticos e semiautomáti-cos.

Tabela 1 – Resumo comparativo dos Trabalhos Relacionados

4
Extensões para linguagem NCL

Como já discutido na seção de trabalhos relacionados, a implementação de referência original do Ginga-NCL possuía limitações que não permitiam a sua utilização ou fácil adaptação para viabilizar os cenários de uso apresentados na Seção 2.2. Uma característica importante da linguagem NCL era pouco explorada pela implementação de referência original do Ginga-NCL: a capacidade de abstrair e integrar diferentes mecanismos de exibição. Ao definir apenas dois serviços de comunicação com dispositivos (que são associados a duas classes de dispositivos predefinidas estaticamente (Costa 2009; Soares 2009)), a implementação de referência original dificultava a integração de dispositivos que suportassem outros serviços de comunicação em rede (com diferentes semânticas de mensagens e protocolos necessários).

Para atender a totalidade dos requisitos definidos para a plataforma proposta nesta tese (que a partir de agora será denominada plataforma Ginga-MD – Ginga *Multi-Dispositivos*) algumas modificações para a linguagem NCL foram consideradas para a implementação do protótipo. As modificações apresentadas neste capítulo suprem diferentes lacunas, muitas das quais existiam apenas por conta do grau de abstração das especificações que utilizam a linguagem (como é o caso do mecanismo de associação e descrição de classes de dispositivos NCL e as definições de tolerância à falha e recuperação). Porém, outras modificações incorporadas pelo protótipo da plataforma Ginga-MD introduzem aspectos inovadores (como é o caso do suporte à Classe Captura de Mídia e à integração de dispositivos UPnP), aumentando o escopo de possibilidades oferecido pela linguagem no tocante ao desenvolvimento de aplicações hipermídia distribuídas. É importante ressaltar que tais modificações são válidas apenas dentro do contexto da plataforma Ginga-MD, pois no momento da conclusão desta tese as modificações propostas não são parte dos perfis da linguagem NCL utilizados pelo *middleware* Ginga e suas especificações derivadas (padrões de escopo nacional (ABNT 2007) e internacional (ITU-T 2009)).

4.1. Estendendo as classes de dispositivos NCL

Para facilitar a manipulação das classes de dispositivo NCL na plataforma Ginga-MD, um formato de metadados foi utilizado para expressar os requisitos de infraestrutura para uma aplicação NCL distribuída. Para se adequar às diferentes possibilidades, o modelo de associação de classes de dispositivos proposto é genérico o suficiente para acomodar os diferentes parâmetros relacionados a quaisquer plataformas de integração de dispositivos multimídia (Batista 2010). A abordagem adotada para a associação de classes de dispositivos a aplicações NCL, proposta para a plataforma Ginga-MD, possui os requisitos apresentados abaixo:

- Possibilidade de estabelecer um limite de quantidade de dispositivos (máximo e mínimo) para uma classe associada a uma parte da aplicação em execução em um Dispositivo Pai, somada a um mecanismo de identificação único para cada dispositivo da classe.
- Provisão de mecanismos para a descrição dos recursos (serviços) oferecidos pelos Dispositivos Secundários, para que possam ser associados pelo Dispositivo Pai a uma classe. Esses recursos podem traduzir a semântica de descrição própria a uma plataforma específica, sendo que a requisição por tais informações deve ser suportada pela plataforma Ginga-MD.
- Possibilidade de associação de parâmetros e estados às classes, de forma que o registro a uma classe possa ser condicionado à avaliação de um parâmetro arbitrário disponibilizado pelo dispositivo (por exemplo, o valor de uma assinatura digital, a posição geográfica do dispositivo etc.).

4.1.1. Variáveis NCL

Limitar a quantidade de dispositivos registrados é importante para aplicações que só façam sentido com um limite máximo ou mínimo de participantes. Assim, ações relacionadas a um objeto de mídia só serão notificadas pelo Dispositivo Pai aos Dispositivos Secundários filhos quando a classe contiver uma quantidade de dispositivos registrados compatíveis com sua definição.

A identificação única de dispositivos poderá ser utilizada pelos Dispositivos Filhos para personalizar a exibição dos objetos de mídia que recebe (em particular aplicações NCL). Assim, além das variáveis relacionadas à execução de aplicações multi-dispositivo utilizadas pela versão

da implementação de referência do Ginga anterior ao desenvolvimento desta tese (apresentadas originalmente em (Costa 2009; Soares 2009)), foram incorporadas novas variáveis, refletindo as novas características consideradas para as classes de dispositivos NCL.

Dispositivos Pais poderão capturar informações desses Dispositivos Secundários através do *namespace parent:*

- *parent.class(i)* – e.g. *parent.class(i).var*, onde i é o índice associado à classe de dispositivo NCL e *var* é o nome da variável utilizada.
- *parent.class(i).device(j)* – e.g. *parent.class(i).device(j).var*, onde *i* é o índice associado à classe de dispositivo NCL, *j* é o valor do índice do registro do Dispositivo Secundário filho na classe e *var* é o nome da variável utilizada.

A variável *child.index* deverá conter o índice de registro e pode ser usada por Dispositivos Secundários (registrado em uma Classe Ativa e que possuam um exibidor de aplicações NCL) para personalização do conteúdo que exibem (e.g. exibição condicional através do elemento <*switch*> da linguagem NCL).

4.1.2. Descrição e associação de classe de dispositivos NCL

A descrição das capacidades necessárias para os dispositivos se juntarem a uma classe é utilizada pelo Dispositivo Pai para flexibilizar a associação de diferentes serviços de rede às classes de dispositivos NCL. O uso da mesma semântica e sintaxe para essas especificações simplifica a comparação. Plataformas como Jini, OSGi e UPnP oferecem mecanismos para que seus serviços sejam descritos, utilizando semântica própria e especializada para seu contexto de uso (Helal 2002).

UAProf (*User Agent Profile*) (WAP Forum 2001) é uma implementação concreta do framework W3C CC/PP (*Composite Capabilities/Preference Profiles*) (Klyne 2004), que utiliza RDF (*Resource Description Framework*) para descrição de capacidades de dispositivos móveis genéricos, e que se adequa à descrição de classes de dispositivos. A descrição é composta de informações como: tamanho de tela, exibidores e geradores de mídia, conjunto de caracteres que devem ser suportados etc. As especificações foram originalmente concebidas para que as informações do perfil do dispositivo fossem enviadas junto ao cabeçalho de suas requisições HTTP, de forma que o conteúdo a ser recebido pudesse ser adaptado pelo servidor respondendo à requisição. O modelo CC/PP também é suficiente para definição de parâmetros e estados

associados a classes de dispositivos. A associação de parâmetros dinâmicos em ontologias baseadas no framework CC/PP já foi proposta em trabalhos em adaptação de conteúdo sensível a contexto (Viterbo 2006).

Para a plataforma Ginga-MD, uma extensão da especificação CC/PP é definida para descrição das classes de dispositivos NCL, a partir de entidades do modelo UAProf (WAP Forum 2001). A extensão contempla o acréscimo de alguns elementos específicos (Tabela 4) do modelo de múltiplos dispositivos proposto para o Ginga-NCL, de forma a também incorporar outras semânticas de descrição de capacidades (como a descrição UPnP). Para a especificação e associação de classes de dispositivos NCL são utilizadas as entidades *HardwarePlatform*, *SoftwarePlatform* e *NetworkCharacteristics* do UAProf (que são subclasses do componente genérico definido pelo CC/PP), e uma nova classe *NCLDeviceClass*. Um novo *namespace* é usado para a definição da nova classe, bem como os novos parâmetros específicos da plataforma Ginga-MD e atributos genéricos para comportar definições de capacidades com semântica associada a outro modelo de descrição diferente do UAProf.

A Tabela 4 sumariza as propriedades adicionadas aos elementos definidos pelo UAProf. Na tabela seguinte (Tabela 5) são apresentados os atributos que um elemento *NCLDeviceClass* suporta.

Classe: SoftwarePlatform		
Atributo	*Tipo*	*Exemplo*
gmd:supportedServices	*rdf*:Bag	"UPnP AV MediaServer ControlPoint", "Active Orchestration Service"
gmd:softwareParams	rdf:Bag	"KEY=123456"
Classe: NetworkCharacteristics		
gmd:pairingMethod	Literal	"UPnP", "Ginga-MD Active Pairing"

gmd:supportedProtocols	rdf:Bag	“HTTP”, “HTTPS”, “UDP”, “RTP”

Tabela 2 – Atributos acrescentados às classes definidas pelo UAProf

Classe: NCLDeviceClass	
Atributo	*Tipo*
gmd:maxDevices	Número
gmd:minDevices	Número
gmd:Hardware	prf:HardwarePlatform
gmd:Software	prf:SoftwarePlatform
gmd:Network	prf:NetworkCharacteristics

Tabela 3 – Definição da classe *NCLDeviceClass*

A classe *NCLDeviceClass* possui atributos para determinar os limites máximo e mínimo para quantidade de dispositivos que poderão se registrar em uma classe, sendo responsável por encapsular os três elementos da descrição de perfil UAProf. O componente *SoftwarePlatform* (Tabela 4) foi acrescido de três novos atributos. O atributo *supportedServices* agrega identificadores para os serviços que devem ser providos pelos Dispositivos Secundários filhos, que podem ser utilizados pelo Dispositivo Pai; e o atributo *softwareParams* especifica parâmetros arbitrários para a utilização desses serviços. O componente *NetworkCharacteristics* também recebeu três novos atributos: o atributo *pairingMethod*, que armazena o identificador do mecanismo de pareamento utilizado pelo Dispositivo Secundário filho junto ao Dispositivo Pai; o atributo *supportedProtocols* define quais são os protocolos que podem ser utilizados para a comunicação com o Dispositivo Secundário filho; e, finalmente, o atributo *networkParams* define parâmetros arbitrários para o uso dos serviços de rede dos Dispositivos Secundários filhos.

A descrição das classes de dispositivos NCL associada a uma aplicação acessada pelo componente Gerente de Classes de Dispositivos a partir de um arquivo de definições RDF. Nas aplicações NCL, os identificadores das classes (atributo ID do elemento de tipo RDF *NCLDeviceClass*) são associados ao elemento <*regionBase*> (atributo *device*), que contém as regiões que agregam os objetos de mídia a serem apresentados pelos dispositivos registrados nessa classe.

4.2. Classe Captura de Mídia

A plataforma Ginga-MD suporta o conceito de captura de mídia por parte dos Dispositivos Secundários filhos para envio do fluxo capturado para o Dispositivo Pai. Foi definida a classe de dispositivos NCL "Classe Captura de Mídia". Instâncias do subsistema do Dispositivo Secundário registradas na Classe Captura de Mídia são capazes de capturar e enviar, individualmente, fluxos de mídia para o Dispositivo Pai.

É definido um esquema de URL (*URL schema*) para que os recursos dos Dispositivos Secundários filhos possam ser referenciados na definição de objetos de mídia (elemento <*media*>) em aplicações NCL em execução no Dispositivo Pai. O esquema é apresentado a seguir:

- *ncl-device*://<*device_id*>/<*resource_id*>
  - device_id – identificador do dispositivo (de acordo com as variáveis apresentadas na Seção 4.1.1).
  - resource_id – identificador do recurso utilizado.

Como é apresentado no Capítulo 6 ("Implementação do Protótipo"), o serviço *Media Capture Device Service* (Ginga-MD MCS) implementado (parte do *Secondary Device Subsystem*) oferece a funcionalidade de captura de áudio, e o identificador *AudioCapture* é utilizado para sua identificação.

Um exemplo de URL:

- ncl-device://*parent.class(3).device(1)*/*AudioCapture*
  - parent.class(3).device(1) – primeiro (identificador 1) dispositivo pareado na Classe Captura de Mídia (identificador 3).

4.3. API Lua

Uma API Lua foi definida para oferecer funcionalidades relacionadas ao modelo de integração de dispositivos da linguagem NCL utilizado pela plataforma Ginga-MD. A API foi também incorporada como parte das especificações LuaTV (Brandão 2010), e oferece métodos síncronos e assíncronos (através de uma nova classe de eventos NCLua (Sant'anna 2008), que possui como identificador a *string multidevice*). A API Lua oferece funcionalidades para aplicações em execução no Dispositivo Pai, e oferece métodos síncronos para manipulação de informações disponíveis localmente e métodos assíncronos para manipulação de informações que requerem troca de mensagens com os Dispositivos Secundários. A Tabela 6 apresenta a API Lua com os métodos síncronos:

Retorno do método	**Operação e parâmetros de entrada**
bool (resultado da operação de cadastro de classe de dispositivos NCL a partir da sua descrição)	multidevice:**addDeviceClass**(string desc)
-	multidevice:**removeDeviceClass**(*string* devClassId)
string (descrição da classe de dispositivos NCL)	multidevice:**getDeviceClassMetadata**(string classId)

Tabela 4 – Métodos síncronos – módulo *multidevice.*

Os métodos síncronos oferecidos pelo módulo *multidevice* (Tabela 6) oferecem funcionalidades para a associação (método *addDeviceClass*) e dissociação (método *removeDeviceClass*) de classes de dispositivos NCL junto às aplicações em execução, e também um método para recuperação dos metadados que descrevem as classes de dispositivos NCL (método *getDeviceClassMetadata*).

Os métodos assíncronos devem possuir tratador de eventos pré-registrado para a classe "*multidevice*" (seguindo o modelo de tratamento de eventos Lua e a API NCLua *event* (Sant'anna 2008)).

Para recuperar o estado de um evento de um objeto de mídia associado a uma classe de dispositivos, envia-se primeiramente um evento Lua com o formato:

- *evt* = {class='multidevice', type='object_state', *device_class*=string, *object_id*=string, *anchor_id*=string, *ncl_event_id*=string, *request*_id=identifier }
    - *device_class – string* com identificação da classe de dispositivos;
    - *object_id – string* com identificador do objeto de mídia (elemento <*media*>);
    - *anchor_id – string* com identificador (opcional) para âncora de objeto de mídia (elemento <*area*>);
    - *ncl_event_id – string* com identificador do evento (máquina de estados) a ser consultado (evento de apresentação, atribuição ou seleção);
    - *request_id* – identificador (arbitrário) para a requisição.

A resposta para a requisição pelo estado do objeto possui o seguinte formato:

- evt = {class='multidevice', type='object_state', device_*class*=string, object_id=string, anchor_id=string, *state*=identifier, *error*=<err_msg>, *request_id*=identifier }
    - *resquest_id* – identificador arbitrário utilizado na solicitação;
    - *state* – código identificador para o estado consultado;
    - *error* – mensagem de erro (campo mutuamente exclusivo com o campo *state*).

Para recuperar a propriedade de um objeto de mídia associado a uma classe de dispositivos, posta-se primeiro um evento Lua com o formato:

- evt = {class='multidevice', type='object_property', *device_class*=string, *object_id*=string, *property_id*=string, *request_id*=identifier }
    - *device_class – string* com identificação da classe de dispositivos;
    - *object_id – string* com identificador do objeto de mídia (elemento <*media*>);
    - *property_id – string* com identificador da propriedade a ser consultada;
    - *request_id* – identificador (arbitrário) para a requisição.

A resposta para a requisição pelo estado do objeto possui o seguinte formato:

- evt = {class='multidevice', type='object_property', device_class=string, object_id=string, *property_id*=string, *property_value*=string, *error*=<err_msg>, *request_id*=identifier }
    - *resquest_id* – identificador arbitrário utilizado na solicitação;
    - *property_id – string* com identificador da propriedade consultada;
    - *property_value – string* com valor da propriedade consultada;
    - *error* – mensagem de erro (campo mutuamente exclusivo com o campo *property_value*).

Para recuperar a propriedade de uma classe de dispositivos, posta-se primeiro um evento Lua com o formato:

- evt = {class='multidevice', type='class_property', *device_class*=string, *property_id*=string, *request_id*=identifier }
    - *device_class – string* com identificação da classe de dispositivos;
    - *property_id – string* com identificador da propriedade a ser consultada;
    - *request_id* – identificador (arbitrário) para a requisição.

A resposta para a requisição pela propriedade da classe de dispositivos possui o seguinte formato:

- evt = {class='multidevice', type='class_property', device_class=string, *property_id*=string, *property_value*=string, *error*=<err_msg>, *request_id*=identifier }
    - *resquest_id* – identificador arbitrário utilizado na solicitação;
    - *property_id – string* com identificador da propriedade consultada;
    - *property_value – string* com valor da propriedade consultada;
    - *error* – mensagem de erro (campo mutuamente exclusivo com o campo *property_value*).

Além dos métodos síncronos e assíncronos, a API Lua da plataforma Ginga-MD permite o registro de ouvintes (*listeners*) para o estado de objetos associados a classes de dispositivos NCL. Os eventos gerados possuem o seguinte formato:

- evt = {class='multidevice', type='object_state_change', *device_class*=string, *anchor_id*=string, *action_id*=string, *state*=identifier, *ncl_event_id*=string, *payload*=string, *timestamp*=time }

- *device_class* – *string* com identificação da classe de dispositivos;
- *object_id* – *string* com identificador do objeto de mídia (elemento <*media*>);
- *anchor_id* – *string* com identificador (opcional) para âncora de objeto de mídia (elemento <*area*>);
- *action_id* – identificador da transição realizada no evento do objeto de mídia associado;
- *state* – estado do evento do objeto de mídia, resultado da transição realizada;
- *ncl_event_id* – *string* com identificador do evento (máquina de estados) no qual a ação foi realizada (evento de apresentação, atribuição ou seleção);
- *payload* – parâmetros associados à transição;
- *timestamp* – marcação de tempo associada à realização da ação no Dispositivo Secundário.

5
Arquitetura de Software

É possível modelar um sistema distribuído que viabilize todos os cenários motivadores apresentados na Seção 2.2. Esta tese propõe um sistema que seja capaz de realizar cenários funcionalmente equivalentes aos descritos, a partir de uma descrição de alto nível de uma aplicação hipermídia. Tal descrição deve contemplar todos os elementos necessários e toda a lógica de execução para uma aplicação hipermídia distribuída, através de uma linguagem que seja capaz de expressar tais definições. Uma implementação da lógica de execução é apresentada no Capítulo 6 ("Implementação do Protótipo").

Como mencionado no capítulo anterior, a solução proposta por esta tese utiliza NCL (*Nested Context Language*) (Soares 2007) como linguagem de descrição para as aplicações hipermídia distribuídas que executa. NCL já era utilizada para especificar aplicações hipermídias distribuídas e também já existiam implementações de exibidores NCL que as executassem (Soares 2009), porém o leque de funcionalidades disponíveis nos exibidores era sensivelmente menor do que o considerado pelos cenários de uso que motivaram esta tese.

Os cenários motivadores foram concebidos a partir do estabelecimento de requisitos que não eram atendidos pelos exibidores que seguiam o modelo proposto para a linguagem NCL. O conjunto dos requisitos evoluiu com o desenvolvimento desta tese (Batista 2010; Batista 2011), a partir da pesquisa por funcionalidades relevantes e a implementação de protótipos intermediários, tomando como base a versão original da implementação de referência do *middleware* Ginga (Soares 2007). Gradativamente, acréscimos e adaptações foram planejados para a linguagem NCL, atendendo diferentes aspectos do seu modelo de exibição distribuída (Batista 2010), e novos módulos foram adaptados ou incorporados à implementação. Finalmente, materializou-se a definição de uma plataforma e seus subsistemas, que aumentava, então, o escopo de cenários de uso da linguagem.

A plataforma Ginga-MD, cuja arquitetura de *software* será apresentada neste capítulo, utiliza um perfil estendido da linguagem NCL (incorporando os acréscimos apresentados no capítulo anterior) para a descrição das aplicações hipermídia distribuídas. NCL tem por propósito

ser uma linguagem de cola, permitindo a integração e o sincronismo entre diferentes tipos de mídias e aplicações (Soares 2005; Soares 2007), e é usada neste trabalho como camada de controle para serviços que se baseiam em dispositivos distribuídos e sistemas heterogêneos (Soares 2009; Batista 2010).

Denominamos o conjunto dos subsistemas de plataforma porque essa deve ser capaz de incorporar outros subsistemas cuja compatibilidade é definida em termos mais abstratos do que os comumente utilizados para o estabelecimento de arquiteturas de *software* para *middlewares* e *frameworks*. No escopo desta tese, uma plataforma se posiciona num nível de abstração acima, utilizando NCL para oferecer abstrações para os recursos dos dispositivos heterogêneos, no tocante às capacidades e características relevantes para a execução de mídias e aplicações, e com relação às suas configurações e mecanismos de conectividade.

É de suma importância a existência de uma arquitetura de *software* de referência para a plataforma e assim facilitar a integração de diferentes recursos de dispositivos heterogêneos e seus serviços de comunicação. Deste modo facilita-se a implementação de serviços de dispositivos que possam ser acoplados a uma determinada implementação da plataforma e também a implementação de variações da plataforma como um todo. A plataforma Ginga-MD define mecanismos de interoperabilidade através da especificação de API e protocolos de comunicação, que são usados para integrar os subsistemas.

A descrição da arquitetura de *software* de referência para a plataforma Ginga-MD inicia-se com a definição de seus requisitos, quais as entidades funcionais os suprem e uma arquitetura de rede necessária para comunicação entre essas entidades. As entidades funcionais são posteriormente mapeadas em subsistemas a serem executados em dispositivos diferentes e interconectados, concretizando o modelo de comunicação e orquestração hierárquica estabelecido como requisito.

A descrição da arquitetura segue, então, com a apresentação de visões distintas para a plataforma Ginga-MD – as visões contemplam um conjunto de informações suficientemente relevante para uma correta implementação de seus subsistemas. A metodologia, definida em (Kruchten, 1995), estabelece um "modelo arquitetural de visões 4+1". Tal modelo tem seu uso difundido na indústria e representa um conjunto suficiente de informações para representar uma vasta gama de sistemas, possuindo também vários registros de uso na literatura científica (Clements 2003). Quatro visões apresentam diferentes pontos de vista para o sistema, partindo

dos casos de uso (a visão extra, que é a especialização dos requisitos). As visões apresentam, sequencialmente, a estruturação lógica do sistema, qual o comportamento e a dinâmica dos processos que realizam os casos de uso, qual a distribuição e implantação dos módulos de *software* que compõem a plataforma e finalmente um esquema resumido para a implementação do sistema como um todo. Este capítulo contempla as visões de casos de uso e lógica. A visão de processo é discutida no Capítulo 6 ("Implementação do Protótipo") enquanto as visões de implementação e implantação são discutidas no Capítulo 7 ("Testes Sistêmicos").

As visões utilizam diagramas UML com o intuito de definir precisamente as características mais importantes da plataforma, que são aquelas diretamente relacionadas ao atendimento dos requisitos levantados, para a realização dos objetivos desta tese. Graças à constante evolução das especificações e da pluralidade de ferramentas de modelagem disponíveis, UML se popularizou na última década (Pandey 2010) e, ao contrário de qualquer outra ADL (Linguagem de descrição de arquitetura, do inglês *Architecture Description Language*) (Pandey 2010), passou a ser usada largamente na indústria. As atuais especificações para UML a fazem ser, não só considerada como uma ADL suficiente (Pandey 2010), como também reconhecida como padrão *de fato* para documentação de arquitetura de sistemas de *software* (Eden 2006;Pandey 2010).

A descrição e avaliação da solução para os problemas levantados por esta tese partem da arquitetura de *software* introduzida neste capítulo. A análise posterior dos resultados dos testes sistêmicos do protótipo configura a verificação do comportamento da execução do protótipo considerando conjuntos de funcionalidades, o que, dentro do estabelecido para esta tese, materializa a prova de conceito da implementação do protótipo.

5.1. Visão geral – Requisitos

Esta seção estabelece os requisitos funcionais e não funcionais para a plataforma Ginga-MD. Os requisitos estão associados às entidades funcionais da plataforma, que são subsistemas genéricos que desempenham facilidades necessárias para viabilização de cada um dos três cenários apresentados no Capítulo 2. São definidos requisitos para a plataforma como um todo e também requisitos específicos para cada uma das entidades funcionais.

Aplicações hipermídia distribuídas possuem requisitos inerentes, de acordo com a natureza da informação que manipulam. A execução de uma aplicação hipermídia distribuída, como as

que são apresentadas nos cenários de uso, consiste pragmaticamente de conteúdos de mídia sendo consumidos simultaneamente por múltiplos dispositivos interconectados. Para que a lógica de execução (na forma de definições de sincronismo intra e inter fluxos de mídia), pretendida pelo autor da aplicação, seja mantida, os dispositivos devem manter atualizado o estado da execução da aplicação – para tal, os dispositivos trocam entre si comandos, fluxos e objetos de mídia.

Na descrição dos cenários de uso, são apresentados ambientes diferentes nos quais uma aplicação hipermídia era executada utilizando recursos de múltiplos dispositivos (com usuários únicos diferentes), porém, em todos os casos, um dispositivo é compartilhado por todos os usuários simultaneamente (um receptor de TV, um computador com projetor etc.). Os ambientes considerados são ricos em dispositivos com características e capacidades heterogêneas e que oferecem possibilidades diferentes de conectividade. Uma representação gráfica para a visão geral da plataforma, contemplando as entidades funcionais e a arquitetura de rede, é apresentada na Figura 1.

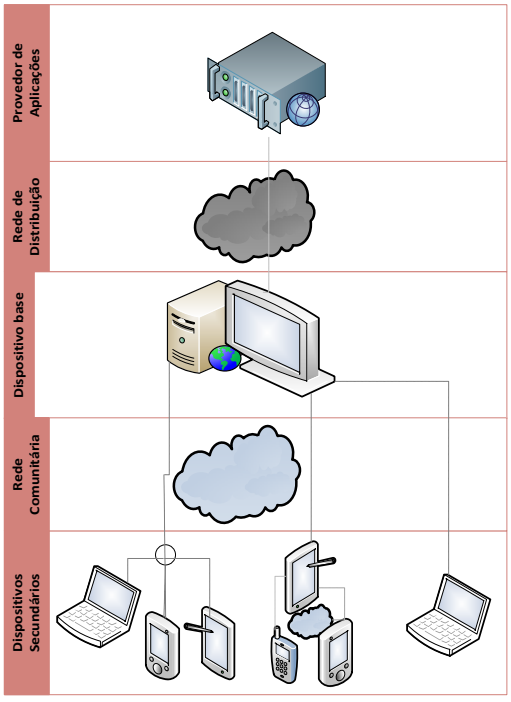


Figura 1 – Entidades funcionais de alto-nível e arquitetura de rede para execução de aplicações hipermídia distribuídas

As entidades funcionais da plataforma Ginga-MD são descritas a seguir.

- **Provedor de Aplicações** – é o dispositivo computacional com um subsistema responsável por armazenar o que compõe a aplicação a ser executada na plataforma. Envia e atualiza a aplicação e seus elementos para o Dispositivo Base.
- **Dispositivo Base** – é o dispositivo computacional com um subsistema responsável por executar a aplicação hipermídia (NCL) recebida do Provedor de Aplicações. A aplicação contempla definições para a orquestração de recursos locais (e.g. exibidores de mídias) e também de recursos remotos (e.g. serviços de exibidores de dispositivos conectados), além de uma descrição dos requisitos necessários para sua execução.
- **Dispositivo Pai** – é o dispositivo computacional capaz de orquestrar recursos locais (e.g. exibidores de mídias) e também de recursos remotos (e.g. serviços de exibidores de dispositivos conectados), além de uma descrição dos requisitos necessários para sua execução. Todo Dispositivo Base é necessariamente um Dispositivo Pai, mas Dispositivos secundários podem ser ou não Dispositivos Pai, como definido a seguir.
- **Dispositivo Secundário** – é o dispositivo computacional que possui um subsistema que viabiliza que seus recursos (e.g. exibidores de mídias) possam ser orquestrados remotamente, de acordo com a lógica de uma aplicação recebida pelo Dispositivo Base**.** Para juntar-se à plataforma, um Dispositivo Secundário deverá parear-se a um dispositivo ascendente (Dispositivo Pai) definido pela aplicação, que pode ser o Dispositivo Base (raiz da árvore de orquestração hierárquica) ou outro Dispositivo Secundário. A lógica da aplicação no Dispositivo Base define uma árvore de orquestração hierárquica onde um dispositivo secundário (pai) pode orquestrar os recursos de outros dispositivos secundários (filhos) a ele pareados. Tal relacionamento entre dispositivos secundários pode se repetir recursivamente, desde que a hierarquia definida na aplicação seja obedecida sempre e que diferentes classes de dispositivos sejam utilizadas para cada nível da hierarquia (dentro de uma mesma rede local, de acordo com a definição de domínio de aplicação NCL (Soares 2009)). Ao parear-se com um Dispositivo Pai, um Dispositivo Secundário registra-se em uma ou mais classes de dispositivos. Um Dispositivo Secundário poderá se registrar em um tipo específico de classe de dispositivos se possuir o módulo de *software* (serviço) correspondente à classe. Visando acomodar as características dos dispositivos apresentados nos cenários da Seção 2.2, são definidas as seguintes classes de dispositivos:
    - Classe Ativa Estéril – Dispositivos Secundários que possuírem o módulo de *software* correspondente à Classe Ativa Estéril são capazes de receber, decodificar e executar o conteúdo dos objetos de mídia, a partir de comandos de orquestração recebidos do Dispositivo Pai.
    - Classe Ativa Fértil – Dispositivos Secundários que podem se associar a essa classe possuem, além do módulo de *software* correspondente à classe Ativa Estéril, módulos capazes de orquestrar os recursos de outros Dispositivos Secundários filhos. Assim sendo, parte da aplicação recebida do Dispositivo Pai pode ser redistribuída para seus Dispositivos Filhos.

- Classe Passiva – Os Dispositivos Secundários que possuírem o módulo de *software* correspondente à Classe Passiva são capazes de decodificar fluxos de mídia renderizados, recodificados e transmitidos pelo Dispositivo Pai. O Dispositivo Pai processa e codifica os fluxos a partir das definições de composição (por exemplo, entre elementos visuais) presentes na aplicação hipermídia que executa, e envia o resultado da renderização para o grupo de dispositivos dinamicamente registrados na Classe Passiva.
- Classe Captura de Mídia – Dispositivos Secundários que possuírem o módulo de *software* correspondente, capaz de capturar e enviar, individualmente, fluxos de mídia para o Dispositivo Pai.
- Classe UPnP *AV MediaServer ControlPoint* – Os dispositivos capazes de se registrar na classe de dispositivos UPnP *AV MediaServer ControlPoint* são aqueles que suportam o serviço UPnP de mesmo nome (Kang 2005). Dispositivos registrados na Classe *UPnP AV MediaServer ControlPoint* poderão visualizar uma lista de vídeos compartilhados pelo Dispositivo Pai e recuperá-los individualmente. O Dispositivo Pai harmoniza o ciclo de vida de objetos de mídia referenciados pela aplicação hipermídia que executa com o compartilhamento de vídeos suportados pelo serviço UPnP *AV MediaServer ControlPoint*.

Além das entidades funcionais de alto nível, a plataforma depende da arquitetura de rede composta por:

- **Rede de Distribuição** – elemento da arquitetura de rede que conecta os Provedores de Aplicações a instâncias de Dispositivo Base (e.g. uma rede de TV IPTV ou *broadcast*, uma conexão internet com um repositório de aplicações etc.). Tem capacidade para transmitir os dados das aplicações hipermídia enviados pelo Provedor de Aplicações para o Dispositivo Base.
- **Rede Comunitária** – elemento da arquitetura de rede que representa uma rede disponível em um espaço de uso coletivo, utilizada para comunicação entre Dispositivo Base e os Dispositivos Secundários e entre Dispositivos Secundários. Tem capacidade para transmitir as mensagens de pareamento e orquestração, enviadas e recebidas pelos dispositivos de acordo com a árvore hierárquica definida na aplicação em execução no Dispositivo Base.

A seguir são apresentados os subsistemas que compõem a plataforma e seus requisitos funcionais e não funcionais, elaborados a partir das entidades funcionais e arquitetura de rede definidas acima.

5.1.1. Descrição dos requisitos dos subsistemas

Os usuários utilizarão a plataforma a partir da interação com o subsistema usado pelo *Dispositivo Base* e com as instâncias do subsistema usado pelos *Dispositivos Secundários*. Os autores de aplicações hipermídia distribuídas utilizarão a plataforma a partir de mecanismos oferecidos pelo subsistema usado pelo *Provedor de Aplicações*, podendo registrar e atualizar aplicações a serem requisitadas pelo *Dispositivos Base*.

Assim, considerando as entidades funcionais mapeadas anteriormente, são definidos os seguintes subsistemas para a plataforma Ginga-MD:

- *Application Provider Subsystem* (APS) – subsistema utilizado pelo Provedor de Aplicações.
- *Base Device Subsystem* (Ginga-MD BDS) – subsistema utilizado pelo Dispositivo Base.
- *Secondary Device Subsystem* (Ginga-MD SDS) – subsistema utilizado pelo Dispositivo Secundário. O Ginga-MD SDS deve conter os módulos de *software* para pelo menos um dos seguintes serviços:
    - *Passive Device Service* (Ginga-MD PDS) – serviço necessário para registro na Classe Passiva.
    - *Sterile Active Device Service* (Ginga-MD SADS) – serviço necessário para registro na Classe Ativa Estéril.
    - *Fertile Active Device Service* (Ginga-MD FADS) – serviço necessário para registro na Classe Ativa Fértil.
    - *Media Capture Device Service* (Ginga-MD MCS) – serviço necessário para registro na Classe Captura de Mídia.

Os requisitos não funcionais que devem ser observados para a especificação dos subsistemas que compõem a plataforma objeto desta tese são os seguintes:

- **Interoperabilidade** – a arquitetura de *software* deve oferecer mecanismos que permitam a integração de dispositivos que utilizem protocolos de comunicação definidos pela plataforma Ginga-MD. Deve ser possível integrar implementações diferentes para os subsistemas usados pelos Provedores de Aplicações e Dispositivos Secundários, além das que são implementadas nesta tese. A plataforma, então, deve oferecer interfaces e protocolos padronizados para que um Dispositivo Base que suporte NCL possa integrar novas ou diferentes tecnologias, tanto para recuperação de aplicações quanto para integração de dispositivos.
- **Confiabilidade** – a execução distribuída de uma aplicação NCL deve ser realizada a partir de definições consistentes, e o comportamento dos dispositivos envolvidos deverá ser o mesmo

quando a execução for realizada em ambientes com a mesma configuração de *hardware*, *software* e rede.

- **Resiliência** – a plataforma deve prover mecanismos automáticos ou semiautomáticos para manutenção de níveis aceitáveis de eficiência na utilização dos recursos disponíveis para a execução da aplicação hipermídia distribuída, face um conjunto representativo de falhas e dificuldades de operação na infraestrutura (dispositivos envolvidos e arquitetura de rede) utilizada.
- **Segurança** – a execução de uma aplicação hipermídia distribuída não pode expor os dispositivos integrados a violações de segurança. A plataforma deve garantir que cada dispositivo atue de acordo com a hierarquia de orquestração que o autor da aplicação especificou.

Os requisitos gerais listados abaixo foram observados para a especificação de todos os componentes da plataforma:

- **Aplicações devem ser descritas através de um formato aberto de aplicações hipermídia distribuídas.** Tal formato deve: permitir a manipulação de mídias estáticas e contínuas; oferecer sincronismo intermídia e intramídia; oferecer suporte à adaptação de exibição e de conteúdo para diferentes configurações e perfis de uso; oferecer suporte à manipulação de interação dos usuários; permitir a descrição de apresentações hipermídia distribuídas a partir de abstrações de alto nível e compatível com as entidades funcionais definidas nesta seção.
- **Suporte a um modelo de orquestração hierárquica de recursos**. A plataforma deve realizar a orquestração dos recursos distribuídos a partir da troca de mensagens entre dispositivos pareados, de acordo com a hierarquia definida na aplicação, considerando o modelo de orquestração hierárquica utilizado nos cenários do Capítulo 2, que é uma extensão do modelo originalmente definido para os exibidores da linguagem NCL.
- **Suporte à comunicação em grupo**. Tendo em vista que os cenários de uso demandam a transmissão de fluxos de mídia e sinais de sincronismo para grupos de dispositivos, é importante que a plataforma possibilite a criação de aplicações que usem protocolos de comunicação em rede que permitam estabelecer não só canais de comunicação *unicast* como também *broadcast* e *multicast*. A plataforma deve suportar um mecanismo flexível para definição de grupos, incluindo quais são os critérios e requisitos de agrupamento dos dispositivos secundários (características dos dispositivos, protocolos de comunicação necessários etc.). A definição dos grupos deve estar associada à descrição da aplicação hipermídia.

5.1.2. Requisitos para *Base Device Subsystem*

Abaixo são listados os requisitos funcionais e não funcionais para o subsistema que deve estar presente no Dispositivo Base, o *Base Device Subsystem* (Ginga-MD BDS). O Dispositivo

Base é a raiz da árvore de orquestração, que carrega aplicações NCL do Provedor de Aplicações e controla Dispositivos Secundários na sua rede comunitária.

Os seguintes requisitos funcionais são considerados para o *Base Device Subsystem*:

- Receber aplicações sob demanda (*pulled data*) via Rede de Distribuição (e.g. Internet).
- Receber aplicações sem solicitação (*pushed data*) via Rede de Distribuição (e.g. canal de difusão terrestre).
- Interpretar os sinais de atualização de aplicação enviados pelo *Application Provider Subsystem* através da Rede de Distribuição.
- Executar as aplicações hipermídia (NCL) solicitadas pelo usuário. O *Base Device Subsystem* deverá, a partir das definições contidas na aplicação, orquestrar seus recursos locais (exibidores, interfaces de entrada) e também recursos remotos disponibilizados por Dispositivos Secundários filhos pareados através da Rede Comunitária.
- Permitir o pareamento de instâncias de *Secondary Device Subsystem* através da Rede Comunitária. O pareamento estabelece os canais de comunicação necessários para que os serviços dos Dispositivos Secundários sejam orquestrados pelo *Base Device Subsystem*, de acordo com a lógica de uma aplicação NCL.
- Orquestrar os serviços oferecidos por instâncias de *Secondary Device Subsystem* pareadas através da Rede Comunitária, com base nas definições da aplicação NCL em execução. Tais definições devem incluir os critérios de agrupamento (classes de dispositivos NCL) e como deve ser feita a orquestração dos recursos oferecidos pelo grupo (serviços associados às classes, que oferecem abstrações para os protocolos e mecanismos utilizados para a troca de mensagens de orquestração).

Os requisitos não funcionais adicionais para o *Base Device Subsystem*:

- Implementação do subsistema deve ser portável para sistemas operacionais diferentes.

5.1.3. Requisitos para *Secondary Device Subsystem*

Os Dispositivos Secundários apresentados nos cenários do Capítulo 2 são associados a subcategorizações: classes diferentes de dispositivos NCL. As classes NCL de dispositivos agrupam Dispositivos Secundários a partir as características que devem possuir e serviços que devem oferecer para se registrar na classe. Dispositivos Secundários foram utilizados nos cenários para seleção de conteúdo, entrada de texto, identificação pessoal, e como interface genérica de entrada. Os Dispositivos Secundários deverão possuir o *Secondary Device Subsystem*, para o qual são definidos os seguintes requisitos funcionais:

- Parear-se com um Dispositivo Pai, através da Rede Comunitária, de acordo com as definições da aplicação hipermídia em execução no *Base Device Subsystem*. A aplicação

define quais os requisitos necessários para o pareamento entre os dispositivos e uma estrutura hierárquica para a comunicação, que é utilizada para a orquestração dos dispositivos.

- Ter seus recursos orquestrados de acordo com a lógica da aplicação no seu Dispositivo Pai, após o pareamento. O *Secondary Device Subsystem* deverá utilizar seus recursos locais de acordo com as mensagens de orquestração vindas de um dispositivo único, que é seu ascendente imediato na hierarquia definida na aplicação no Dispositivo Base (seu Dispositivo Pai).

A seguir são apresentados os requisitos funcionais adicionais considerados para os serviços que poderão ser oferecidos pelo *Secondary Device Subsystem*.

O serviço *Passive Device Service* possui os seguintes requisitos funcionais:

- Decodificar e exibir o fluxo de mídia renderizado e enviado pelo Dispositivo Pai.
- Notificar eventos de interação do usuário para seu Dispositivo Pai.

O serviço *Sterile Active Device Service* possui os seguintes requisitos:

- Receber objetos de mídia do seu Dispositivo Pai, e executá-los localmente.
- Receber e processar mensagens de orquestração.
- Notificar seu Dispositivo Pai acerca do processamento das mensagens de orquestração e de notificações advindas da execução dos objetos de mídia que executa.
- Orquestrar recursos locais de acordo com a lógica da aplicação hipermídia em execução no seu Dispositivo Pai.

O serviço *Fertile Active Device Service* possui, além dos requisitos definidos para o *Sterile Active Device Service*, os seguintes requisitos funcionais:

- Ser capaz de receber partes da aplicação hipermídia executada pelo seu Dispositivo Pai.
- Notificar seu Dispositivo Pai acerca do processamento das mensagens de orquestração e de notificações advindas da execução de sua parte da aplicação.
- Parear Dispositivos Secundários filhos.

O serviço *Media Capture Device Service* possui os seguintes requisitos:

- Enviar, quando solicitado, fluxo de mídia ou pacote contendo objeto de mídia estática para o Dispositivo Pai.

O serviço oferecido pela plataforma UPnP (Kang 2005) denominado *UPnP AV MediaServer ControlPoint* possui seus requisitos definidos em (Ritchie 2002).

O *Secondary Device Subsystem* possui os seguintes requisitos não funcionais:

- Implementação do subsistema deve ser portável para sistemas operacionais diferentes.

5.1.4.
Requisitos para *Application Provider Subsystem*

Abaixo são apresentados os requisitos funcionais para o subsistema utilizado pelo Provedor de Aplicações, o *Application Provider Subsystem*:

- Possibilitar o armazenamento de todos os dados das aplicações hipermídia que podem ser solicitadas ou recebidas sem solicitação pelo *Base Device Subsystem* através da Rede de Distribuição.
- Oferecer mecanismos para que autores publiquem suas aplicações.
- Entregar a aplicação NCL, e os dados necessários para sua execução, para o *Base Device Subsystem*.
- Atualizar uma aplicação em execução no *Base Device Subsystem*. Quando o autor altera uma aplicação em execução, o *Application Provider Subsystem* deve enviar os dados alterados e a sinalização adequada para o *Base Device Subsystem* através da Rede de Acesso.

5.1.5.
Arquitetura de Rede – Requisitos não funcionais

A arquitetura de rede é composta por dois elementos de alto nível: a Rede de Distribuição e a Rede Comunitária.

A Rede de Distribuição determina alguns requisitos não funcionais que devem observados pelas entidades funcionais com as quais se relaciona. A Rede de Distribuição deve:

- Suportar o envio das informações (dados de aplicações e sinalizações de atualização) do Provedor de Aplicações para o Dispositivo Base.

A Rede Comunitária possui requisitos não funcionais associados, que devem ser observados pelas entidades funcionais com as quais se relaciona. A Rede Comunitária deve:

- Estabelecer um canal de comunicação entre dois dispositivos, de acordo com a hierarquia definida dentro da aplicação hipermídia sendo executada pelo Dispositivo Base – comumente a conexão é criada através de uma LAN, HAN, LAN, PAN etc.
- Suportar os protocolos e mecanismos necessários para pareamento e orquestração entre os dispositivos conectados (comunicação *unicast* e *multicast*), de acordo com a lógica da aplicação hipermídia sendo executada no Dispositivo Base.

5.2.
Visões da Arquitetura de *Software* da Plataforma

A seguir são apresentadas visões diferentes da arquitetura de software modelada para a plataforma Ginga-MD. Os artefatos associados às visões têm por intuito definir precisamente o

comportamento de cada subsistema a partir do ponto de vista de todos os atores que com eles interagirão. Como previamente mencionado, a notação UML é utilizada para descrição dos artefatos apresentados.

5.2.1. Visão de Casos de Uso

Esta seção lista os casos de uso mais relevantes para cada subsistema que compõe a plataforma Ginga-MD. Casos de uso significativos são aqueles fundamentais para o atendimento dos requisitos estabelecidos para a plataforma, e realizam uma ampla cobertura da arquitetura dos subsistemas (envolvimento dos elementos arquiteturais).

Os casos de uso são descritos textualmente, realçando elementos significativos do comportamento dos subsistemas: atores envolvidos, pré-condição para a ativação das funcionalidades, sequência lógica de passos para a realização do caso de uso e, fluxos alternativos de eventos e, finalmente, as pós-condições.

5.2.1.1. Casos de uso para *Base Device Subsystem*

5.2.1.1.1.Execução de aplicações sob demanda (*como pulled data*)

Esse caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para solicitação de aplicações NCL servidas pelo *Application Provider Subsystem*.

Atores envolvidos: Usuário do Dispositivo Base e o *Application Provider Subsystem*.

Pré-condição: o *Base Device Subsystem* deve estar conectado ao *Application Provider Subsystem* via Rede de Distribuição.

Fluxo de eventos:

1. Usuário do Dispositivo Base seleciona, de uma lista, uma aplicação NCL e, então, o *Base Device Subsystem* envia para o *Application Provider Subsystem* uma mensagem de requisição com o identificador da aplicação selecionada.

2. *Application Provider Subsystem* responde a requisição enviando, através da Rede de Distribuição, um pacote contendo a aplicação NCL e os arquivos necessários para o início de sua execução.

3. *Base Device Subsystem* recebe a mensagem e armazena a aplicação NCL e seus objetos associados que necessitam de armazenamento.

4. *Base Device Subsystem* carrega os recursos necessários para iniciar a execução da aplicação recebida. A conexão com o *Application Provider Subsystem* é mantida.

5. Após confirmação do Usuário do Dispositivo Base, a execução da aplicação NCL é iniciada e a conexão com o *Base Device Subsystem* é mantida.

Pós-condição: A especificação da aplicação NCL deve estar armazenada e acessível para o *Base Device Subsystem*; aplicação NCL deve permanecer em execução durante período de acordo com suas definições (ou intervenção do Usuário do Dispositivo Base). Conexão com *Application Provider Subsystem* deve permanecer ativa durante a execução da aplicação NCL.

5.2.1.1.2. Execução de aplicações sem solicitação (como pushed data)

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para recebimento de aplicações NCL a partir de um canal de transmissão de TV Digital (e.g. uma rede de difusão).

Atores envolvidos: Usuário do Dispositivo Base, *Application Provider Subsystem* (Retaguarda de transmissão).

Pré-condição: *Base Device Subsystem* e *Application Provider Subsystem* estão conectados à Rede de Distribuição.

Fluxo de eventos:

1. *Base Device Subsystem* recebe, através da Rede de Distribuição, notificação de nova aplicação NCL (e.g. a notificação é gerada a partir de uma retaguarda de transmissão de TV Digital).

2. *Base Device Subsystem* descarrega e armazena a aplicação NCL e os arquivos associados necessários para o início de sua execução.

3. *Base Device Subsystem* carrega os recursos necessários para iniciar a execução da aplicação NCL; com a carga completa, a execução da aplicação NCL é iniciada.

Pós-condição: A especificação da aplicação NCL deve estar armazenada e acessível para o *Base Device Subsystem*; aplicação NCL deve permanecer em execução durante período de acordo com suas definições (ou intervenção do Usuário do Dispositivo Base).

5.2.1.1.3. Atualização de aplicação NCL recebida do *Application Provider Subsystem*

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para atualização de aplicações NCL servidas pelo *Application Provider Subsystem*.

Ator envolvido: *Application Provider Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar conectado ao *Application Provider Subsystem* via rede de distribuição. *Base Device Subsystem* deve estar executando aplicação NCL servida pelo *Application Provider Subsystem*.

Fluxo de eventos:

1. *Base Device Subsystem* recebe uma mensagem do *Application Provider Subsystem* (via Rede de Distribuição) notificando a existência modificações para a aplicação NCL em execução.

2. *Base Device Subsystem* processa modificações na aplicação NCL e arquivos associados.

3. *Base Device Subsystem* recebe os recursos que se fizerem necessários para execução após a atualização; descarrega recursos não mais utilizados pela versão modificada da aplicação NCL.

Pós-condição: Aplicação NCL deve ter sua versão atualizada armazenada e deve permanecer em execução considerando definições atualizadas; conexão com o *Application Provider Subsystem* deve ser mantida.

5.2.1.1.4. Pareamento com *Secondary Device Subsystem*

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para realizar o pareamento com o *Secondary Device Subsystem*.

Ator envolvido: *Secondary Device Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a classe de dispositivos associada à instância de *Secondary Device Subsystem* que solicita o pareamento; Dispositivo Base e Dispositivo Secundário estão conectados à Rede Comunitária.

Fluxo de eventos:

1. *Base Device Subsystem* recebe mensagem difundida pelo *Secondary Device Subsystem*; a mensagem contém informações relativas à busca por serviço associado à classe de dispositivos NCL (características do dispositivo, serviços que oferece) que quer se associar.

2. *Base Device Subsystem* recebe solicitação, verifica se há classe de dispositivos NCL compatível com a requisitada pelo *Secondary Device Subsystem* e, em caso positivo, realiza o registro do dispositivo na classe.

3. *Base Device Subsystem* configura um canal de orquestração de acordo com as características especificadas pelo *Secondary Device Subsystem* e com os serviços associados à classe de dispositivos na qual foi realizada o registro.

Fluxo secundário: No passo 3 do fluxo de eventos, o *Base Device Subsystem* verifica se o *Secondary Device Subsystem* está se reconectando. Em caso positivo, *Base Device Subsystem* envia mensagens para a recuperação do sincronismo (de acordo com serviços oferecidos pelo *Secondary Device System*).

Pós-condição: Canal de orquestração entre o *Base Device Subsystem* e *Secondary Device Subsystem* é estabelecido; *Base Device Subsystem* registra dispositivo na classe de dispositivos NCL adequada.

5.2.1.1.5. Envio de mensagem de orquestração para instâncias de *Secondary Device Subsystem* registradas na Classe Passiva

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para a manutenção do sincronismo temporal dos objetos de mídia associados à Classe Passiva de dispositivos NCL, que agrupa instâncias de *Secondary Device Subsystem* que possuem o serviço *Passive Device Service*.

Ator envolvido: *Secondary Device Subsystem* com serviço *Passive Device Service*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Passiva; Dispositivo Base e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados.

Fluxo de eventos:

1. *Base Device Subsystem* realiza transição em alguma das máquinas de estado de um objeto de mídia associado à Classe Passiva.

2. A transição modifica ou cria (dependendo do estado dos objetos de mídia envolvidos) um fluxo de amostras de mídia renderizado pelo *Base Device Subsystem*, que é enviado (com suas amostras codificadas) para o grupo que agrega as instâncias pareadas de *Secondary Device Subsystem* que possuam o serviço *Passive Device Service*. Os conteúdos dos objetos de mídia são mixados, renderizados (de acordo com as configurações de exibição definidas na aplicação NCL no Dispositivo Base), e codificados (o serviço *Passive Device Service* define um formato específico), para que, então, um fluxo único seja enviado.

3. *Secondary Device Subsystem* recebe, decodifica e realiza a exibição das amostras recebidas (através do seu serviço *Passive Device Service*).

Fluxo secundário: No passo 1 do fluxo de eventos, caso a transição efetuada culmine com a desativação de todos os objetos de mídia associados à Classe Passiva, o fluxo para de ser gerado até que alguma outra transição modifique o estado de algum objeto de mídia associado à mesma classe de dispositivos.

Pós-condição: *Base Device Subsystem* deve permanecer enviando fluxo codificado para o *Secondary Device Subsystem*.

5.2.1.1.6. Envio de mensagem de orquestração para instâncias de *Secondary Device Subsystem* registradas nas Classes Ativas

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para a manutenção do sincronismo temporal dos objetos de mídia associados à classe de dispositivos NCL que agrupa instâncias de *Secondary Device Subsystem* que possuem o serviço *Sterile Active Device Service* ou *Fertile Active Device Service.*

Ator envolvido: *Secondary Device Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Ativa Estéril ou Classe Ativa Fértil; Dispositivo Base e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados.

Fluxo de eventos:

1. *Base Device Subsystem* realiza transição em alguma das máquinas de estado de um objeto de mídia associado à Classe Ativa Estéril ou Classe Ativa Fértil.

2. *Base Device Subsystem* envia mensagem de orquestração contendo informações da transição de máquina de estados (ação, qual máquina de estados, qual objeto de mídia, qual âncora do objeto, quais parâmetros da transição) para cada uma das instâncias de *Secondary Device Subsystem* registradas na Classe Ativa Estéril ou Classe Ativa Fértil que processam a mensagem.

Fluxo secundário: No passo 1 do fluxo de eventos, caso a transição efetuada seja a primeira em um objeto associado à Classe Ativa Estéril ou Classe Ativa Fértil, o *Base Device Subsystem* deve antes enviar um pacote (arquivo zip) contendo o objeto de mídia (e os arquivos necessários para o início de sua execução) para os subsistemas pareados.

Pós-condição: *Base Device Subsystem* deve ter atualizado o estado dos objetos de mídia associados; instâncias do *Secondary Device Subsystem* registradas na Classe Ativa Fértil ou Classe Ativa Estéril terem processado a ação de orquestração enviada pelo *Base Device Subsystem*.

5.2.1.1.7. Envio de mensagem de orquestração para dispositivos que possuírem o serviço UPnP *AV MediaServer ControlPoint*

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para a manutenção do sincronismo temporal dos objetos de mídia associados à Classe *UPnP AV MediaServer ControlPoint*.

Ator envolvido: Dispositivo *UPnP AV MediaServer ControlPoint*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe UPnP *AV MediaServer ControlPoint*; Dispositivo Base e dispositivo UPnP estão conectados à Rede Comunitária e pareados.

Fluxo de Eventos:

1. *Base Device Subsystem* realiza a transição da máquina de estados de um objeto de mídias associado à Classe UPnP *AV MediaServer ControlPoint* e notifica o serviço *UPnP AV Media Server* para que tal objeto seja incluído ou excluído (dependendo da transição na máquina de estados de apresentação efetuada) da lista de elementos compartilhados para recuperação sob demanda por parte dos dispositivos UPnP.

Pós-condição: lista de vídeos compartilhados pelo *Base Device Subsystem* (através do serviço *UPnP AV MediaServer*) com dispositivos registrados na Classe *UPnP AV MediaServer ControlPoint* é atualizada de acordo com a transição efetuada pelo *Base Device Subsystem* durante a execução da aplicação NCL.

5.2.1.1.8. Recebimento de mensagem de orquestração de instâncias de *Secondary Device Subsystem* registradas na Classe Passiva

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para a captura e processamento de mensagens de orquestração relacionadas aos objetos de mídia associados à Classe Passiva.

Ator envolvido: *Secondary Device Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Passiva; Dispositivo Base e o Dispositivo Secundário estão conectados à

Rede Comunitária e pareados; pelo menos um objeto de mídia associado à Classe Passiva deve estar em execução (de acordo com a sua máquina de estados de apresentação NCL).

Fluxo de eventos:

1. *Base Device Subsystem* recebe mensagem de orquestração enviada por um *Secondary Device Subsystem* registrado na Classe Passiva. A mensagem de orquestração contém informações de interação do usuário, ocorrida durante a exibição do fluxo de mídia codificado.

2. *Base Device Subsystem* processa mensagem de interação e realiza modificações necessárias nas máquinas de estado dos objetos de mídia afetados.

3. *Base Device Subsystem* renderiza objetos de mídia associados à Classe Passiva e passa a enviar fluxo de amostras de mídia codificado refletindo a interação para o grupo de Dispositivos Secundários registrados na Classe Passiva.

Pós-condição: Instâncias de *Secondary Device Subsystem* registradas na Classe Passiva passam a exibir (um mesmo) conteúdo que reflete o processamento da interação pelo *Base Device Subsystem*.

5.2.1.1.9. Recebimento de mudança de variável NCL associada às Classes Ativas

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para processar mensagens recebidas das instâncias pareadas de *Secondary Device Subsystem* que possuam o serviço *Sterile Active Device* Service ou *Fertile Active Device Service*. Tais mensagens são relacionadas à modificação de variáveis NCL associadas à Classe Ativa Estéril ou Classe Ativa Fértil.

Ator envolvido: *Secondary Device Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Ativa Estéril ou Classe Ativa Fértil; Dispositivo Base e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados; pelo menos um objeto de mídia associado à Classe Ativa Estéril ou Classe Ativa Fértil deve estar em execução (de acordo com a sua máquina de estados de apresentação NCL).

Fluxo de eventos:

1. *Base Device Subsystem* recebe mensagem enviada por instância de *Secondary Device Subsystem* (registrada na Classe Ativa Estéril ou Classe Ativa Fértil) contendo informações relativas à mudança de uma variável NCL (pertencente a um *namespace* específico da plataforma Ginga-MD), ocorrida durante a execução da aplicação NCL recebida.

2. *Base Device Subsystem* processa a modificação da variável (registra o valor da variável no seu nó de definições locais), identificando também qual o índice associado ao *Secondary Device Subsystem* que gerou a modificação.

Pós-condição: *Base Device Subsystem* realiza ações relacionadas à modificação da variável.

5.2.1.1.10. Recebimento de fluxo codificado de mídia de instâncias de *Secondary Device Subsystem* registradas na Classe Captura de Mídia

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para receber fluxos de mídia codificados de uma instância de *Secondary Device Subsystem* registrada na Classe Captura de Mídia.

Ator envolvido: *Secondary Device Subsystem*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Captura de Mídia; Dispositivo Base e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados.

Fluxo de eventos:

1. *Base Device Subsystem* realiza transição de ativação na máquina de estado de apresentação de objeto de mídia NCL que identifica (através de uma URL com esquema específico da plataforma Ginga-MD) recurso de captura de mídia oferecido (e.g. captura de áudio) por uma instância particular de *Secondary Device Subsystem*. A instância é identificada na URL através de seu índice de registro, assim como cada recurso a ser oferecido possui seu identificador próprio definido pela plataforma Ginga-MD (como é discutido no Capítulo 4).

2. *Base Device Subsystem* gera uma requisição por conteúdo para a instância específica de *Secondary Device Subsystem* (que processará a requisição através do seu serviço *Media Capture Device Service*).

3. *Base Device Subsystem* recebe e exibe conteúdo enviado pela instância de *Secondary Device Subsystem*. O conteúdo é exibido pelo *Base Device Subsystem* através de um exibidor local (que tratará uma URL convertida do esquema específico da plataforma Ginga-MD para uma que o exibidor possa tratar – e.g. um esquema de URL utilizado pelo protocolo HTTP).

Fluxo secundário: No passo 2 do fluxo de eventos, caso o dispositivo identificado pelo índice associado ao objeto de mídia não esteja conectado ou não possua o serviço necessário para

realizar a captura, o processo de conversão notificará o exibidor de falha (objeto de mídia não disponível).

Pós-condição: *Base Device Subsystem* realizar a exibição do conteúdo recebido de uma instância específica de *Secondary Device Subsystem*.

5.2.1.1.11. Recebimento de solicitação de dispositivos compatíveis com serviço *UPnP AV MediaServer ControlPoint*

O caso de uso descreve as etapas realizadas pelo *Base Device Subsystem* para a recepção e o e tratamento das requisições relacionadas à Classe *UPnP AV MediaServer ControlPoint*. O serviço UPnP AV MediaServer ControlPoint é especificado em (Ritchie 2002).

Ator envolvido: Dispositivo compatível com serviço *UPnP AV MediaServer ControlPoint*.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a classe de dispositivos associada aos dispositivos compatíveis com o serviço UPnP *AV MediaServer ControlPoint*; Dispositivo Base e dispositivo UPnP estão conectados à Rede Comunitária e pareados; pelo menos um objeto de mídia associado à Classe UPnP *AV MediaServer ControlPoint* deve estar em execução (máquina de estados de apresentação NCL).

Fluxo de Eventos:

1. *Base Device Subsystem* recebe solicitação do dispositivo compatível com o serviço UPnP *AV MediaServer ControlPoint* para objeto de mídia presente na lista de compartilhamento (objetos de mídia associados à Classe UPnP *AV MediaServer ControlPoint* que foram iniciados). O processo de solicitação é detalhado em (Ritchie 2002).

2. *Base Device Subsystem* entrega (através de seu serviço UPnP *AV MediaServer*, como definido em (Ritchie 2002)) o objeto de mídia para o dispositivo compatível com o serviço UPnP *AV MediaServer ControlPoint* que fez a solicitação.

Pós-condição: *Base Device Subsystem* deve ter entregado o objeto de mídia solicitado ao dispositivo compatível com o serviço UPnP *AV MediaServer ControlPoint.*

5.2.1.2. Casos de uso para *Secondary Device Subsystem*

A seguir são definidos os casos de uso para *Secondary Device Subsystem*. Também são apresentados os casos de uso para os serviços que o subsistema pode oferecer: *Passive Device Service*, *Sterile Active Device Service*, *Fertile Active Device Service, Media Capture Device*

Service e o serviço *UPnP AV MediaServer ControlPoint* (incorporado da plataforma UPnP (Ritchie 2002; Kang 2005)).

5.2.1.2.1. Casos de uso para *Secondary Device Subsystem*

5.2.1.2.1.1.Pareamento com subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para realizar o pareamento com um subsistema ascendente (Dispositivo Pai), que pode ser o *Base Device Subsystem* ou uma instância de *Secondary Device Subsystem* que possua o serviço *Fertile Active Device Service*.

Atores envolvidos: Usuário do Dispositivo Secundário, Subsistema do Dispositivo Pai.

Pré-condição: Subsistema do Dispositivo Pai deve estar executando uma aplicação NCL que referencie a classe de dispositivos associada ao *Secondary Device Subsystem*; Dispositivo Pai e Dispositivo Secundário estão conectados à Rede Comunitária.

Fluxo de eventos:

1. Usuário do Dispositivo Secundário seleciona a opção de busca por serviços para sua classe de dispositivos NCL.

2. *Secondary Device Subsystem* difunde mensagem de busca por serviço na Rede Comunitária, informando na mensagem de solicitação suas características (inclusive os serviços que oferece).

3. Subsistema do Dispositivo Pai recebe solicitação e realiza o registro na classe, se conectando aos serviços oferecidos pelo *Secondary Device Subsystem*. Usuário do Dispositivo Secundário é notificado do pareamento.

Fluxos secundários: No passo 3, mensagens de orquestração para recuperação de sincronismo podem ser recebidas, caso o *Secondary Device Subsystem* já tenha se pareado junto ao subsistema do Dispositivo Pai durante a mesma execução de uma aplicação NCL.

Pós-condição: Canal de orquestração entre o subsistema do Dispositivo Pai e *Secondary Device Subsystem* é estabelecido.

*5.2.1.2.2.*Casos de uso para *Passive Device Service*

Os casos de uso do serviço *Passive Device Service* foram definidos com base nas funcionalidades apresentadas nos cenários de uso (Capítulo 2) associadas a dispositivos que

possuíam apenas um exibidor de mídia contínua e mecanismos de pareamento e notificação de interação simples. Tais funcionalidades foram incluídas nos cenários considerando as definições estabelecidas para o tipo *passivo* de classe de dispositivos NCL (Soares 2009).

5.2.1.2.2.1.Recebimento e exibição de conteúdo enviado por Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para a recepção e exibição de conteúdo enviado pelo subsistema do Dispositivo Pai ao qual se pareou.

Ator envolvido: Subsistema do Dispositivo Pai.

Pré-condição: *Base Device Subsystem* deve estar executando uma aplicação NCL que referencie a Classe Passiva; Dispositivo Pai e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados.

Fluxo de eventos:

1. *Secondary Device Subsystem* recebe do subsistema do Dispositivo Pai um fluxo de amostras de mídia codificadas.

2. *Secondary Device Subsystem* realiza a decodificação e exibição das amostras recebidas.

Pós-condição: *Secondary Device Subsystem* deve permanecer decodificando e exibindo fluxo enquanto subsistema do Dispositivo Pai enviar.

5.2.1.2.2.2. Envio de mensagem de orquestração para subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para a captura e envio de sinais de interação ocorridos durante a decodificação e exibição do fluxo de amostras de mídia recebido do subsistema do Dispositivo Pai.

Ator envolvido: Subsistema do Dispositivo Pai, Usuário do Dispositivo Secundário.

Pré-condição: Subsistema do Dispositivo Pai deve estar executando uma aplicação NCL que referencie a Classe Passiva; Dispositivo Pai e Dispositivo Secundário estão conectados à Rede Comunitária e pareados; pelo menos um objeto de mídia associado à Classe Passiva deve estar em execução (de acordo com sua máquina de estados de apresentação NCL).

Fluxo de eventos:

1. Usuário do Dispositivo Secundário interage (usando mecanismo de seleção, e.g. cursor controlado ou tela sensível a toque) com conteúdo recebido do Dispositivo Pai.

2. *Secondary Device Subsystem* envia mensagem de orquestração contendo as informações relacionadas à interação do usuário (coordenadas da seleção realizada) para o subsistema do Dispositivo Pai.

3. *Secondary Device Subsystem* recebe do subsistema do Dispositivo Pai fluxo de amostras de mídia codificadas geradas considerando o processamento mensagem de orquestração.

Pós-condição: *Secondary Device Subsystem* deve exibir conteúdo que reflete o processamento da interação realizado pelo subsistema do Dispositivo Pai.

5.2.1.2.3. Casos de uso para *Sterile Active Device Service* e *Fertile Active Device Service*

Os casos de uso para *Sterile Active Device Service* e *Fertile Active Device Service* foram definidos com base nas funcionalidades apresentadas nos cenários de uso (Capítulo 2) associadas a dispositivos capazes de controlar localmente a execução objetos de mídia. Tais funcionalidades foram incluídas nos cenários considerando as definições estabelecidas para o tipo *ativo* de classe de dispositivos NCL (Soares 2009).

5.2.1.2.3.1. Recebimento de objeto de mídia de subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para receber e executar objetos de mídia recebidos do subsistema do Dispositivo Pai ao qual se pareou.

Atores envolvidos: Usuário do Dispositivo Secundário, Subsistema do Dispositivo Pai.

Pré-condição: Subsistema do Dispositivo Pai deve estar executando uma aplicação NCL que referencie a Classe Ativa Estéril ou a Classe Ativa Fértil; Dispositivo Pai e Dispositivo Secundário estão conectados à Rede Comunitária e pareados.

Fluxo de eventos:

1. *Secondary Device Subsystem* recebe do subsistema do Dispositivo Pai uma mensagem de orquestração contendo um pacote (e.g. arquivo zip) que carrega um objeto de mídia (e.g. um vídeo ou uma aplicação NCL) e os arquivos relacionados necessários para o início de sua execução (e.g. objetos de mídia parte de uma aplicação NCL).

2. *Secondary Device Subsystem* descompacta e armazena o conteúdo do pacote, registrando o identificador do objeto de mídia.

3. *Secondary Device Subsystem* carrega recursos necessários para iniciar a execução do objeto de mídia recebido.

Pós-condição: *Secondary Device Subsystem* passa a armazenar os arquivos referentes ao objeto de mídia enviado pelo subsistema ascendente.

5.2.1.2.3.2. Recebimento de mensagem de orquestração de subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para receber e processar mensagens de orquestração do subsistema de um Dispositivo Pai.

Ator envolvido: Subsistema em Dispositivo Pai.

Pré-condição: Subsistema do Dispositivo Pai deve estar executando uma aplicação NCL que referencie a Classe Ativa Estéril ou a Classe Ativa Fértil; Dispositivo Pai e o Dispositivo Secundário estão conectados à Rede Comunitária e pareados. *Secondary Device Subsystem* possui pelo menos um objeto de mídia armazenado.

Fluxo de eventos:

1. Subsistema do Dispositivo Pai, durante a execução de uma aplicação NCL, realiza transição em alguma das máquinas de estado de um objeto de mídia associado à Classe Ativa Estéril ou a Classe Ativa Fértil. *Secondary Device Subsystem* recebe, então, mensagem de orquestração contendo as informações da transição (seu tipo, qual máquina de estados afetou, qual objeto de mídia, qual âncora do objeto e quais parâmetros estão associados).

2. *Secondary Device Subsystem* processa a mensagem de orquestração, realizando as ações descritas (aplica as transições às máquinas de estado dos objetos que possui).

Pós-condição: *Secondary Device Subsystem* deve ter processado a ação de orquestração enviada pelo subsistema do Dispositivo Pai.

5.2.1.2.3.3. Envio de mudança de variável NCL para subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para enviar mensagens relacionadas à modificação de variáveis NCL associadas à Classe Ativa Estéril ou a Classe Ativa Fértil para o subsistema do seu Dispositivo Pai.

Ator envolvido: Subsistema do Dispositivo Pai.

Pré-condição: Subsistema ascendente deve estar executando uma aplicação NCL que referencie a Classe Ativa Estéril ou a Classe Ativa Fértil; Dispositivo Pai e Dispositivo

Secundário devem estar conectados à Rede Comunitária e pareados; pelo menos um objeto de mídia associado à Classe Ativa Estéril ou a Classe Ativa Fértil deve estar em execução (de acordo com sua máquina de estados de apresentação NCL). *Secondary Device Subsystem* possui aplicação NCL armazenada (recebida do subsistema do Dispositivo Pai) e em execução.

Fluxo de eventos:

1. *Secondary Device Subsystem* executa aplicação NCL recebida de subsistema do seu Dispositivo Pai, e altera uma variável NCL de nomenclatura específica da plataforma Ginga-MD.

2. *Secondary Device Subsystem* envia mensagem para subsistema do Dispositivo Pai contendo informações acerca da mudança da variável.

3. Subsistema do Dispositivo Pai recebe e processa a mensagem, realizando ações definidas para a ocorrência de mudanças na variável NCL alterada.

Pós-condição: Subsistema do Dispositivo Pai deve ter realizado ações relacionadas à modificação da variável NCL.

5.2.1.2.4. Casos de uso para *Fertile Active Device Service*

Os casos de uso exclusivos para o serviço *Fertile Active Device Service* foram definidos com base nas funcionalidades apresentadas nos cenários de uso (Capítulo 2) associadas a dispositivos capazes de controlar localmente a execução objetos de mídia. Tais funcionalidades foram incluídas nos cenários considerando as definições estabelecidas para o tipo *ativo* de classe de dispositivos NCL (Soares 2009). Diferentemente do que acontece com a Classe Ativa Estéril, Dispositivos Secundários registrados na Classe Ativa Fértil poderão parear outros Dispositivos Secundários como seus filhos, de forma que possa orquestrar os recursos dos mesmos.

Assim, são incorporados os casos de uso definidos para o *Base Device System* nas seções 4.2.1.1.4 (Pareamento com *Secondary Device Subsystem*), 4.2.1.1.5 (Envio de mensagem de orquestração para instâncias de *Secondary Device Subsystem* registradas na Classe Passiva), 4.2.1.1.6 (Envio de mensagem de orquestração para instâncias de Secondary Device Subsystem registradas nas Classes Ativas), 4.2.1.1.7 (Envio de mensagem de orquestração para instâncias de Secondary Device Subsystem registradas na Classe *UPnP AV MediaServer ControlPoint*), 4.2.1.1.8 (Recebimento de mensagem de orquestração de instâncias de Secondary Device Subsystem registradas na Classe Passiva), 4.2.1.1.9 (Recebimento de mudança de variável NCL associada às Classes Ativas), 4.2.1.1.10 (Recebimento de fluxo codificado de mídia de instâncias

de Secondary Device Subsystem registradas na Classe Captura de Mídia) e 4.2.1.1.11 (Recebimento de solicitação de dispositivos compatíveis com serviço *UPnP AV MediaServer ControlPoint*).

*5.2.1.2.5.*Casos de uso para *Media Capture Device Service*

Os casos de uso para *Media Capture Device Service* foram definidos com base nas funcionalidades apresentadas nos cenários de uso (Capítulo 2) associadas a Dispositivos Secundários capazes de capturar mídias e enviá-las para o seu Dispositivo Pai.

5.2.1.2.5.1.Envio de conteúdo para subsistema do Dispositivo Pai

O caso de uso descreve as etapas realizadas pelo *Secondary Device Subsystem* para capturar e enviar fluxos de mídia, quando requisitados pelo subsistema do Dispositivo Pai.

Ator envolvido: Subsistema do Dispositivo Pai.

Pré-condição: Subsistema do Dispositivo Pai deve estar executando uma aplicação NCL que referencie a Classe Captura de Mídia; Dispositivo Pai e Dispositivo Secundário devem estar conectados à Rede Comunitária e pareados; índice associado à solicitação por conteúdo na aplicação NCL em execução no subsistema do Dispositivo Pai possui referência para *Secondary Device Subsystem* correspondente.

Fluxo de eventos:

1. Subsistema do Dispositivo Pai gera requisição por conteúdo para uma instância específica de *Secondary Device Subsystem* que oferece o serviço *Media Capture Device Service*. *Secondary Device Subsystem* recebe e processa solicitação, iniciando a captura e codificação de amostras de mídia (e.g. áudio) que são enviadas para o subsistema do Dispositivo Pai.

Pós-condição: Subsistema do Dispositivo Pai deve realizar a exibição do conteúdo enquanto *Secondary Device Subsystem* enviar (a partir das definições da aplicação NCL em execução).

*5.2.1.2.6.*Casos de uso para *UPnP AV MediaServer ControlPoint*

Os casos de uso para o serviço *UPnP AV MediaServer ControlPoint* são definidos nas especificações da plataforma UPnP (Ritchie 2002; Kang 2005).

5.2.1.3. Casos de uso para o *Application Provider Subsystem*

5.2.1.3.1. Cadastro de aplicações NCL no repositório do *Application Provider Subsystem*

O caso de uso descreve as etapas realizadas pelo *Application Provider Subsystem* para realizar o cadastro de aplicações NCL em seu repositório.

Ator envolvido: Administrador do Provedor de Aplicações.

Pré-condição: Administrador deve estar devidamente autorizado, e estar de posse dos arquivos de uma aplicação NCL (objetos de mídia, scripts Lua etc.).

Fluxo de eventos:

1. Administrador seleciona opção de cadastro na aplicação de gerenciamento do repositório de aplicações do *Application Provider Subsystem*.

2. Administrador fornece as informações necessárias para o cadastro da aplicação (nome da aplicação, NCL principal, descrição, classificação etc.).

3. Administrador envia pacote (arquivo zip) contendo a aplicação NCL e seus arquivos associados.

4. Administrador submete informações de cadastro.

5. O sistema verifica as informações fornecidas (preenchimentos dos campos obrigatórios, presença do arquivo NCL principal etc.), e caso sejam validadas, o sistema comunica o sucesso ao Administrador.

Fluxos de eventos secundários:

A qualquer momento o cadastro pode ser cancelado e as informações não são registradas junto ao *Application Provider Subsystem*.

No passo 5 do fluxo de eventos, o sistema deve pedir ao Administrador que corrija as informações incorretas ou submeta uma aplicação válida.

Pós-condição: A aplicação NCL deve ter sido cadastrada no repositório de aplicações do *Application Provider Subsystem*.

5.2.1.3.2. Entrega de aplicações solicitadas (*pulled data*) para *Base Device Subsystem*

O caso de uso descreve as etapas realizadas pelo *Application Provider Subsystem* para entregar aplicações NCL para o *Base Device Subsystem*, mediante solicitação.

Ator envolvido: *Base Device Subsystem*.

Pré-condição: *Application Provider Subsystem* deve possuir um repositório com aplicações NCL já cadastradas.

Fluxo de eventos:

1. *Base Device Subsystem* solicita ao *Application Provider Subsystem* (via Rede de Distribuição) uma aplicação NCL, utilizando seu identificador único.

2. *Application Provider Subsystem* verifica se aplicação está registrada em seu repositório e em caso positivo entrega a especificação da aplicação NCL e os arquivos necessários para o início de sua execução.

Fluxos secundários:

No passo 2 do fluxo de eventos, caso a aplicação solicitada não esteja registrada o *Base Device Subsystem* deve ser notificado.

Pós-condição: o pacote da aplicação NCL deve ser completamente transferido para o *Base Device Subsystem*, que iniciará sua execução.

5.2.1.3.3. Entrega de aplicações sem solicitação (*pushed data*) para *Base Device Subsystem*

O caso de uso descreve as etapas realizadas pelo *Application Provider Subsystem* para entregar (sem solicitação) aplicações NCL para o *Base Device Subsystem*.

Ator envolvido: *Base Device Subsystem*.

Pré-condição: *Application Provider Subsystem* deve possuir um repositório com aplicações NCL já cadastradas.

Fluxo de eventos:

1. *Application Provider Subsystem* envia (através da Rede de Distribuição) aplicação NCL para instâncias de Dispositivo Base conectadas.

2. *Base Device Subsystem* recebe do *Application Provider Subsystem* (via Rede de Distribuição) a especificação de uma aplicação NCL e os arquivos necessários para o início de sua execução.

Pós-condição: a especificação da aplicação NCL deve ser transferida para o *Base Device Subsystem*, que iniciará sua execução.

5.2.1.3.4. Atualização de aplicação NCL em execução em instâncias do *Base Device Subsystem*

O caso de uso descreve as etapas realizadas pelo *Application Provider Subsystem* para atualizar aplicações NCL junto a instâncias do *Base Device Subsystem*.

Atores envolvidos: Administrador do Provedor de Aplicações, instâncias de *Base Device Subsystem* conectadas ao *Application Provider Subsystem* (via Rede de Distribuição).

Pré-condição: instâncias do *Base Device Subsystem* terem recebido uma aplicação NCL do *Application Provider Subsystem*; instâncias de *Base Device Subsystem* estão executando aplicação NCL; Dispositivo Base e Provedor de aplicações estão conectados à Rede de Distribuição.

Fluxo de eventos:

1. Administrador do Provedor de Aplicações seleciona opção de atualizar aplicação NCL.

2. Administrador seleciona uma aplicação cadastrada no repositório, a partir do seu identificador único.

3. Administrador atualiza a especificação da aplicação NCL.

4. Administrador submete informações para *Application Provider Subsystem*.

5. O sistema verifica as informações de atualização fornecidas, e caso sejam validadas o sistema comunica o sucesso ao Administrador.

6. *Application Provider Subsystem* envia comandos de atualização para instâncias de *Base Device Subsystem* conectadas e executando a aplicação identificada.

Fluxos secundários:

No passo 5 do fluxo de eventos, o sistema deve pedir ao Administrador que corrija as informações de atualização incorretas.

Pós-condição: instâncias do *Base Device Subsystem* passam a executar aplicação considerando os comandos de atualização recebidos do *Application Provider Subsystem*.

5.2.2. Visão Lógica

Esta seção visa descrever a plataforma Ginga-MD do ponto de vista de suas estruturas estáticas significativas e como elas colaboram para atingir os requisitos estabelecidos. São apresentados os subsistemas da plataforma através de três subvisões: camadas, pacotes e componentes. Estas subvisões mapeiam a organização das principais estruturas lógicas do

sistema, especificando quais suas responsabilidades, como se distribuem e quais suas dependências.

A visão lógica é de suma importância para a definição de uma arquitetura de referência e também para avaliar uma implementação de sistema à luz dos requisitos estabelecidos para a plataforma defendida nesta tese.

5.2.2.1. Subvisão de Camadas

A decomposição em camadas do modelo arquitetural adotado visa definir qual a hierarquia entre os pacotes de *software* de cada subsistema, apresentando quais suas dependências e como se comunicam. Cada camada está associada a um conjunto de responsabilidades, oferecendo serviços para as camadas superiores e utilizando serviços das camadas inferiores.

A divisão em camadas visa reduzir o acoplamento entre os módulos de cada subsistema, facilitando o reuso de componentes. A modelagem em camadas facilita a distribuição das responsabilidades para os componentes do sistema, e oferece uma noção inicial da disposição estrutural lógica para cada subsistema. São apresentados a seguir diagramas de segmentação em camadas para cada subsistema, e posteriormente a descrição de cada camada e seus principais pacotes e componentes de *software*.

5.2.2.1.1 Subvisão de Camadas – *Base Device Subsystem*

A Figura 2 logo abaixo apresenta a subvisão de camadas do *Base Device Subsystem*.

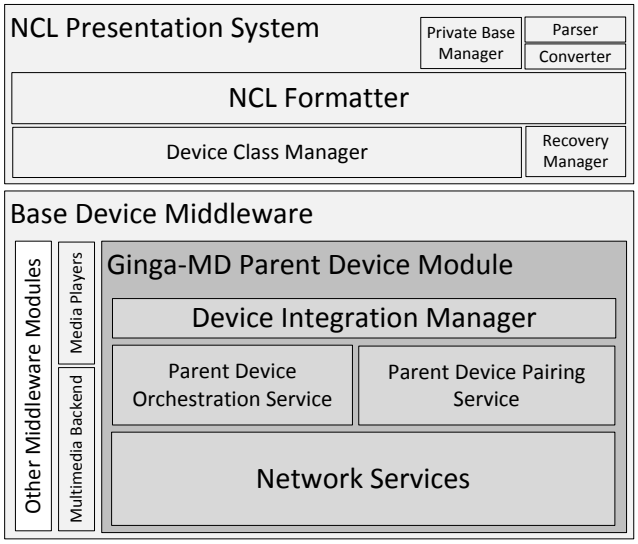


Figura 2 – Diagrama de segmentação em camadas para o *Base Device Subsystem*.

As principais camadas de *software* definidas para o *Base Device Subsystem* são as seguintes:

- *NCL Presentation System* – o subsistema de apresentação de aplicações NCL é a camada de *software* composta por subcamadas e componentes de *software* responsáveis pela manipulação e execução de aplicações NCL. Os principais componentes dessa camada são o Formatador NCL (*NCL Formatter*) e seus conversores e analisadores sintáticos (*Converter* e *Parser*), o Gerente de Bases Privadas (Soares 2007; Soares 2009) (*Private Base manager*), o Gerente de Recuperação (*Recovery Manager* – apresentado na Seção 6.1) e, por fim, o Gerente de Classes de Dispositivos NCL (*Device Class Manager*). O *NCL Presentation System* utiliza serviços e recursos oferecidos pelo *middleware* do Dispositivo Base (Ginga-MD *Base Device Middleware*).
    - *NCL Formatter* – o formatador NCL é o componente responsável por processar aplicações NCL. Utiliza conversores e analisadores sintáticos para capturar as definições de sincronismo entre objetos de mídia NCL, que podem ser referências para arquivos ou fluxos de mídia (arquivos de áudio, fluxo de vídeo etc.), ou para arquivos com código declarativo (páginas HTML, outras aplicações NCL embutidas etc.) ou ainda elementos imperativos (*scripts Lua*). Esse componente também controla a execução de partes remotas da aplicação (associadas às classes de dispositivos NCL), através do componente Gerente de Classes de Dispositivos.
    - *Device Class Manager* – o Gerente de Classes de Dispositivos é o componente responsável por gerenciar as classes de dispositivos NCL utilizadas pelas aplicações em execução. O módulo realiza a associação de instâncias de subsistema Dispositivo

Secundário a uma ou mais classes, dependendo das características e serviços oferecidos. O Gerente de Classes de Dispositivos mantém as variáveis dos objetos de mídia associados às classes de dispositivos e intermedia a comunicação entre o formatador NCL e o módulo de integração com dispositivos, parte do *middleware* do Dispositivo Base, facilitando o controle de serviços remotos através da transição das máquinas de estado associadas a objetos de mídia NCL.

- *Base Device Middleware* – camada de *software* que oferece interfaces de *software* padronizadas para o acesso a recursos e serviços oferecidos pelos dispositivos que hospedarão o *Base Device Subsystem*. As definições dos componentes essenciais para o *middleware* do Dispositivo Base partiram da arquitetura definida para o *middleware* Ginga. Por conseguinte, o *middleware* do Dispositivo Base é composto por um subconjunto dos componentes do núcleo (*Ginga Common Core* (Soares 2007)) da implementação de referência do Ginga além do componente de integração de dispositivos da plataforma Ginga-MD.
    - *Ginga-MD Parent Device Module* – o módulo de integração com dispositivos da plataforma Ginga-MD é responsável por capturar e traduzir as notificações de transição das máquinas de estado dos objetos de mídia NCL, de forma que as notificações possam ser enviadas para os componentes de controle dos serviços oferecidos pelos dispositivos pareados com o *Base Device Subsystem*. Os principais componentes desse módulo são *Device Integration Manager* (Gerente de Integração de Dispositivos), *Parent Device Orchestration Manager* (Serviço de Orquestração – Dispositivo Pai), *Parent Device Pairing Service* (Serviço de Pareamento – Dispositivo Pai), que serão descritos na subvisão de componentes. Este módulo estará presente também no subsistema dos Dispositivos Secundários capazes de se registrar na Classe Ativa Fértil.
    - *Multimedia Backend* – camada de *software* do *middleware* do Dispositivo Base que agrega os componentes responsáveis pelo controle dos recursos de entrada e saída, necessários para a reprodução de mídias e captura de interação do usuário (e.g. exibidores de mídias, controle de interfaces de saída de áudio e vídeo).

5.2.2.1.2 Subvisão de Camadas – *Secondary Device Subsystem*

A Figura 3 apresenta a subvisão de camadas do *Secondary Device Subsystem*. Um diagrama de segmentação em camadas apresenta a organização hierárquica das camadas de *software* que compõem o *Secondary Device Subsystem*.

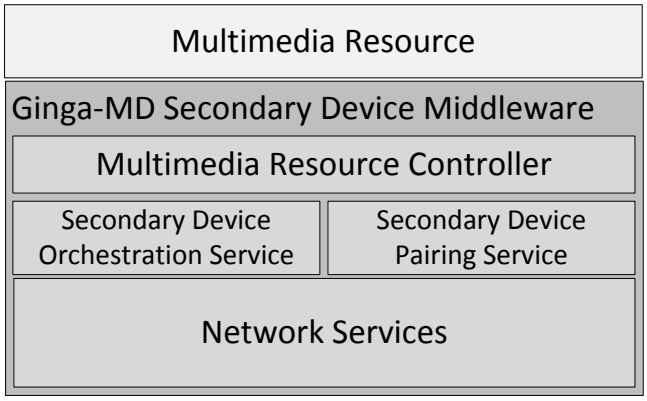


Figura 3 – Diagrama de segmentação em camadas para o *Secondary Device Subsystem*

- *Multimedia Resource* – um "recurso multimídia" representa uma abstração para um serviço de exibição ou captura de mídia oferecido por um dispositivo que possui o subsistema Dispositivo Secundário. Tal recurso poderá ser controlado (orquestrado) remotamente a partir dos serviços oferecidos pelo *middleware* de Dispositivo Secundário da plataforma Ginga-MD. Exemplos de recursos multimídia são exibidores de objetos de mídia (reprodutores de vídeo, ambiente de exibição NCL etc.) ou serviços de captura de mídia.
- *Ginga-MD Secondary Device Middleware* – camada de *software* que permite que recursos de captura e exibição de mídia possam ser orquestrados remotamente por um Dispositivo Pai, dentro da lógica de execução de uma aplicação NCL.
    - *Multimedia Resource Controller* – interface de controle para serviços arbitrários de captura e exibição de mídia. Essa interface é análoga à interface de controle de exibidores utilizada pelo formatador NCL (Moreno 2011).
    - *Secondary Device Orchestration Service* – componente que oferece uma abstração através da qual os recursos disponibilizados pelos dispositivos podem ser orquestrados. A orquestração dos seus serviços ocorre a partir das transições das máquinas de estado dos objetos de mídia dentro de uma aplicação NCL, e este componente é responsável por realizar a comunicação com o serviço de orquestração Dispositivo Pai ao qual se pareou. Os serviços diferentes do *Secondary Device Subsystem* se comunicarão com o subsistema do Dispositivo Pai através de uma implementação de interface definida por esse componente.
    - *Secondary Device Pairing Service* – componente responsável por oferecer mecanismos específicos para o pareamento com subsistemas ascendentes. Os serviços de pareamento diferentes do *Secondary Device Subsystem* se parearão com o subsistema do Dispositivo Pai através da interface definida por este componente.

5.2.2.1.3 Subvisão de Camadas – *Application Provider Subsystem*

A Figura 4 apresenta a subvisão de camadas para o *Application Provider Subsystem*.

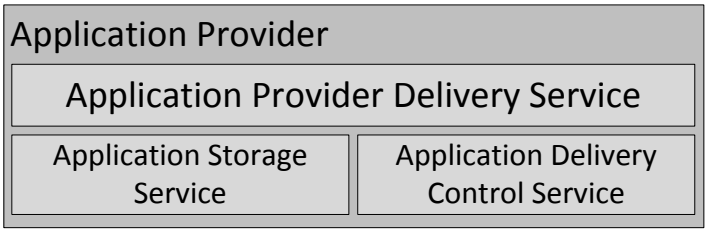


Figura 4 – Diagrama de segmentação em camadas para o *Application Provider Subsystem*

O *Application Provider Subsystem* oferece um serviço de entrega de aplicações NCL – *Application Delivery Service*. Esse serviço utiliza dois componentes de *software*: *Application Storage Service* e o *Application Delivery Control Service*, para acesso ao repositório de aplicações e para o controle da entrega das aplicações.

5.2.2.2. Subvisão de Pacotes

Seguindo a visão de camadas, são apresentados os pacotes de *software* mais relevantes que compõem a plataforma Ginga-MD. Um pacote é um agrupamento lógico de componentes de *software*, que possui responsabilidades, atributos e operações modeladas de acordo com os requisitos mapeados para cada módulo de cada subsistema.

Os pacotes definem o escopo de cada módulo (mapeados em código através de *namespaces*), de forma que o agrupamento dos componentes em pacotes visa facilitar a caracterização das responsabilidades entre os módulos que compõem a plataforma.

São apresentados os pacotes e componentes mais relevantes para cada subsistema através de diagramas de pacotes UML e da descrição textual de cada elemento. O objetivo é concretizar a organização lógica da plataforma e também as interfaces e dependências entre os pacotes.

5.2.2.2.1. Pacotes do *Base Device Subsystem*

Os pacotes mais relevantes que compõem o *Base Device Subsystem* da plataforma Ginga-MD são apresentados na Figura 5, através de um diagrama de pacotes UML e da descrição textual dos pacotes caracterizados.

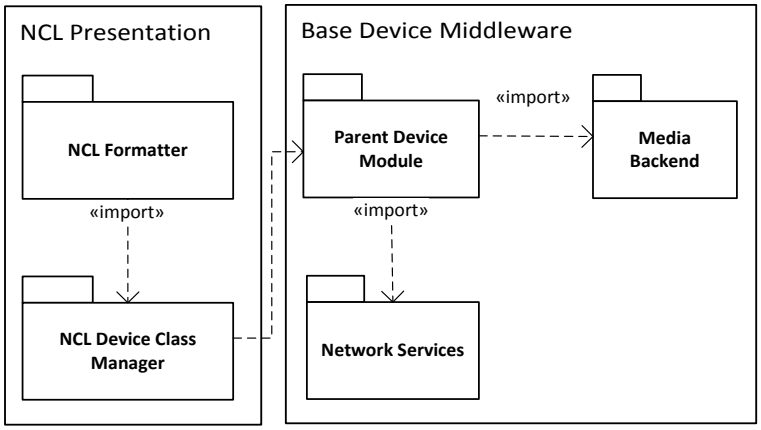


Figura 5 – Diagrama de pacotes para o *Base Device Subsystem*

Os principais pacotes de *software* que compõem o *Base Device Subsystem* da plataforma Ginga-MD se organizam logicamente de acordo com a hierarquia apresentada na subvisão anterior (de camadas).

O pacote *NCL Presentation* agrupa pacotes e componentes que estão relacionados à manipulação e execução de aplicações NCL. Tal pacote vem a ser uma instância do pacote *GingaNCL*, conforme modelado e construído na implementação de referência do *middleware* Ginga[15]. Internamente, o subpacote *NCL Device Class Manager* vem a ser o pacote que inclui o componente responsável pela interface com os serviços de integração com dispositivos oferecidos pelo *middleware* do Dispositivo Base – pacote *Parent Device Module.*

O pacote *Base Device Middleware* inclui pacotes responsáveis por oferecer interfaces para os serviços oferecidos pelo dispositivo executando o *Base Device Subsystem*. O pacote vem a ser uma instância do chamado núcleo comum modelado e construído como parte da implementação de referência do *middleware* Ginga (pacote *Ginga*-CC). Os pacotes mais relevantes, dentro do contexto da plataforma Ginga-MD, são aqueles relacionados à execução de aplicações que envolvam a orquestração de serviços oferecidos por dispositivos remotos (pacote *Parent Device Module*), utilizando serviços de comunicação em rede (parte do *middleware* do Dispositivo Base, pacote *Network Services)*. O pacote *Media Backend* é realçado, pois é necessário para renderização local do fluxo de vídeo a ser enviado para os Dispositivos Secundários registrados

[15] http://www.ginga.org.br

na Classe Passiva e também por ser necessário para controlar o processo de decodificação do conteúdo recebido de dispositivos que oferecem o serviço associado à Classe Captura de Mídia.

5.2.2.2.2. Pacotes do Dispositivo Secundário

Os pacotes mais relevantes que compõem o subsistema Dispositivo Secundário da plataforma Ginga-MD são apresentados a seguir, através de um diagrama de pacotes UML e da descrição textual dos pacotes caracterizados.

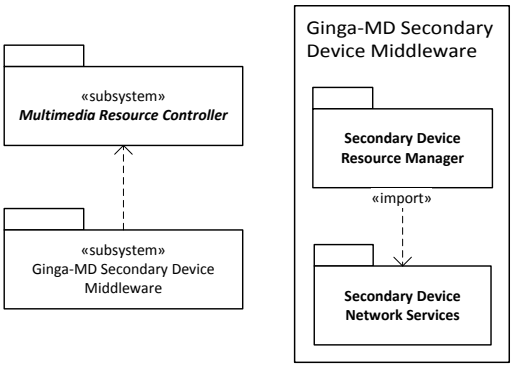


Figura 6 – Diagrama de pacotes para o *Secondary Device Subsystem*

Os principais pacotes de software do *Secondary Device Subsystem* se organizam logicamente de acordo com a hierarquia apresentada na subvisão anterior (de camadas). O diagrama da Figura 6 apresenta uma visão da organização dos pacotes envolvidos.

O pacote *Ginga-MD Secondary Device Middleware* agrega todos os componentes necessários para que um ou mais recursos (representados pelo pacote *Multimedia Resource Controller*) do dispositivo hospedeiro possa ser orquestrado remotamente por um Dispositivo Pai. O pacote *Secondary Device Resource Manager* agrega componentes responsáveis por tratar os sinais de orquestração recebidos e realizar a interface entre o *middleware* do Dispositivo Secundário e o módulo controlador do recurso a ser orquestrado. O envio e recebimento de mensagens de orquestração para o subsistema ascendente e os mecanismos de pareamento são contemplados por componentes presentes no pacote *Secondary Device Network Services*.

5.2.2.2.3. Pacotes do *Application Provider Subsystem*

Os pacotes mais relevantes que compõem o *Application Provider Subsystem* são apresentados na Figura 7, através de um diagrama de pacotes UML e da descrição textual dos pacotes caracterizados.

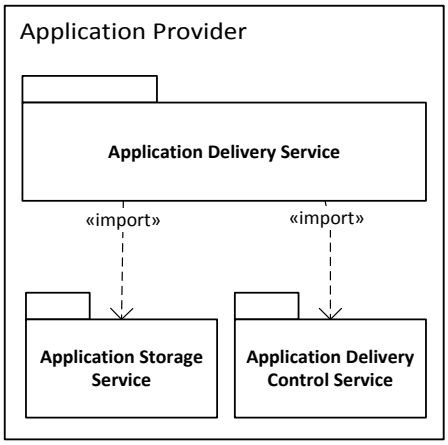


Figura 7 – Diagrama de pacotes para o *Application Provider Subsystem*

O serviço de entrega de aplicações NCL provido pelo *Application Provider Subsystem* (pacote *Application Delivery Service)* utiliza dois pacotes de *software*: *Application Storage Service* e o *Application Delivery Control Service*. O pacote *Application Storage Service* agrega componentes de *software* relacionados ao acesso aos arquivos das aplicações NCL registradas. O pacote *Application Delivery Control Service* agrega componentes que lidam com a lógica de entrega e atualização das aplicações NCL para *Base Device Subsystem*.

5.2.2.3. Subvisão de Componentes

A subvisão de componentes especifica quais componentes relevantes foram construídos ou reutilizados para concretizar a solução proposta nesta tese. Cada componente representa um empacotamento físico (na forma de arquivos executáveis, bibliotecas etc.) de elementos lógicos relacionados (outros componentes internos, classes e interfaces).

Os componentes modelados para os subsistemas da plataforma Ginga-MD definem suas dependências através de interfaces, seguindo o modelo de componentes utilizado pela implementação de referência do *middleware* Ginga (Moreno 2010b). Tais interfaces mapeiam

quais são as funcionalidades oferecidas por cada componente e são importadas por aqueles componentes que utilizam tais funções. A modelagem dos componentes segue a hierarquia de camadas e pacotes da subvisão anterior e será apresentada através de diagramas de componentes UML. Os componentes críticos de cada subsistema são descritos, bem como as principais interfaces entre eles.

5.2.2.3.1. Componentes do *Base Device Subsystem*

Os componentes mais relevantes do *Base Device Subsystem* (dentro do contexto da plataforma Ginga-MD) são apresentados no diagrama de componentes representado na Figura 8.

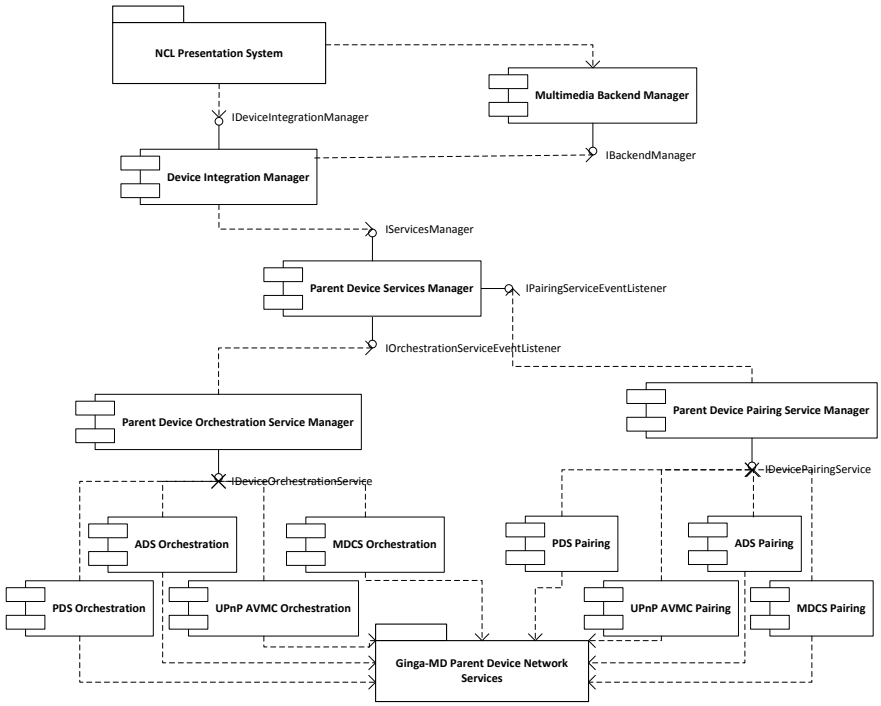


Figura 8 – Diagrama de componentes UML ilustrando principais componentes do *Base Device Subsystem*.

O desenvolvimento da plataforma Ginga-MD utilizou e estendeu a implementação de referência do *middleware* Ginga. De tal modo, o *Base Device Subsystem* incorpora o ambiente de apresentação NCL (o subsistema Ginga-NCL) e também grande parte do núcleo comum utilizado pelo *middleware* Ginga (o subsistema Ginga-CC, Ginga *common core* (Soares 2007)). O diagrama de componentes apresentado na Figura 8 realça os componentes do *Base Device*

Subsystem relacionados com os requisitos estabelecidos para a plataforma Ginga-MD, os quais foram concebidos ou modificados durante o desenvolvimento desta tese.

- *NCL Presentation System* – agrupa pacotes e componentes que estão relacionados à manipulação e execução de aplicações NCL. De acordo com as aplicações NCL que executa, e as classes de dispositivo que tais aplicações utilizam, o *NCL Presentation System* utiliza (através da interface *IDeviceIntegrationManager*) o componente *Device Integration Manager* para orquestrar os recursos oferecidos por dispositivos pareados com o *Base Device Subsystem*.
- *Device Integration Manager* – componente responsável por realizar o controle da comunicação entre o *NCL Presentation System* e o componente de gerência de componentes controladores para os serviços dos Dispositivos Secundários (componente *Parent Device Services Manager*).
- *Multimedia Backend Manager* – componente que oferece funcionalidades para o controle dos recursos de entrada e saída disponíveis para o *middleware* do Dispositivo Base, alinhado às configurações do dispositivo (*hardware* e *software*) que possui o subsistema instalado. Os recursos são controlados por componentes específicos, que incorporam diretivas específicas do sistema operacional do dispositivo hospedeiro, permitindo que tais recursos possam ser utilizados para renderização e exibição dos objetos de mídia em uma aplicação NCL.
- *Parent Device Services Manager* – componente responsável por gerenciar os componentes de controle de orquestração e pareamento dos serviços associados as classe de dispositivos, (serviços oferecidos por instâncias de *Secondary Device Subsystem*).
- *Parent Device Orchestration Service Manager* – componente responsável por oferecer uma abstração (interface comum) para os diferentes componentes de orquestração suportados pelo *Base Device Subsystem*.
    - *PDS Orchestration* (*Passive Device Service Orchestration)* - componente responsável por orquestrar instâncias de *Secondary Device Subsystem* que ofereçam o serviço *Passive Device Service*. É capaz de interpretar transições relacionadas aos objetos de mídia associados à Classe Passiva, e, a partir dessa interpretação, realizar a transmissão do fluxo de amostras gerado a partir da renderização destes objetos de mídia; finalmente, o componente *PDS Orchestration* é responsável por interpretar mensagens de interação vindas de instâncias de *Secondary Device Subsystem*

registradas na Classe Passiva. O componente utiliza serviços de comunicação em rede (pacote *Network Services*) oferecidos pelo *middleware* do Dispositivo Base.

- *ADS Orchestration* (*Active Device Service Orchestration)* – componente responsável por orquestrar instâncias pareadas de *Secondary Device Subsystem* que ofereçam o serviço *Sterile Active Device Service* e *Fertile Active Device Service*. É capaz de interpretar transições relacionadas aos objetos de mídia associados à Classe Ativa Estéril e Classe Ativa Fértil, gerar e transmitir mensagens de orquestração que encapsulem as informações das transições relacionadas a estes objetos de mídia, receber mensagens contendo informações sobre mudança de variáveis realizadas por instâncias de *Secondary Device Subsystem* registradas na Classe Ativa Estéril e Classe Ativa Fértil.
- *MDCS Orchestration* (*Media Capture Device Service Orchestration*) – componente responsável por orquestrar instâncias pareadas de *Secondary Device Subsystem* que ofereçam o serviço *Media Capture Device Service.* É capaz de interpretar transições relacionadas aos objetos de mídia que referenciam recursos da Classe Captura de Mídia, controlando o serviço *Media Capture Device Service do Secondary Device Subsystem* de forma que exibidores de mídia locais possam exibir conteúdo capturado por estes dispositivos.
- *UPnP AV MC Orchestration* – componente responsável por orquestrar dispositivos compatíveis com o serviço *UPnP AV MediaServer ControlPoint* e já pareados. É capaz de interpretar transições relacionadas aos objetos de mídia associados à Classe *UPnP AV MediaServer ControlPoint* e servir para estes dispositivos os objetos de mídia através do serviço *UPnP AV MediaServer*.

- *Parent Device Pairing Service Manager* – componente responsável por gerenciar os diferentes serviços de pareamento suportados pelo *Base Device Subsystem*.
  - *PDS Pairing* (*Passive Device Service Pairing*) – componente responsável por parear instâncias de *Secondary Device Subsystem* que ofereçam o serviço *Passive Device Service*.
  - *ADS Pairing* (*Active Device Service Pairing*) – componente responsável por parear instâncias de *Secondary Device Subsystem* que ofereçam o serviço *Sterile Active Device Service* ou o serviço *Fertile Active Device Service*.

- *MDCS Pairing* (*Media Capture Device Service Pairing*) – componente responsável por parear instâncias de *Secondary Device Subsystem* que ofereçam o serviço *Media Capture Device Service.*
- *UPnP AV MC Pairing* – componente responsável por parear dispositivos compatíveis com o serviço *UPnP AV MediaServer ControlPoint*.

A seguir é apresentada a descrição da interface do componente *Device Integration Manager*.

Retorno do método	**Operação e parâmetros de entrada**
int (código de resultado da operação de carga dos serviços do *Base Device Subsystem* necessário para orquestrar serviços associados às classes de dispositivos NCL)	**addDeviceClass**(*string* devClassDesc)
Void	**removeDeviceClass**(*string* devClassId)
string (descrição da classe de dispositivos NCL)	**getDeviceClassDescription**(*string* devClassId)
int (quantidade de dispositivos pareados a uma determinada classe de dispositivos NCL)	**getDeviceClassSize**(*string* devClassId)
int (código de resultado da ação de associar um objeto de mídia a uma classe de dispositivos NCL)	**addObject**(*string* devClassId, *string* objId, *string* objType, *string* objURL)
void	**removeObject**(*string* devClassId, *string* objId)
int (código de resultado do envio da ação de orquestração relativa a um objeto de mídia associado a uma classe de dispositivos NCL)	**postAction**(*string* devClassId, *string* objId, *string* objType, *int* nclEventType, *string* anchorId, *int* action, *string* value)
int (estado de um objeto de mídia	**getObjectState**(*string* devClassId, *string*

associado a uma classe de dispositivos NCL)	objId, *string* anchorId)
string (valor da propriedade associada a uma classe de dispositivos NCL)	**getDeviceClassPropertyValue**(*string* devClassId, *string* propId)
string (valor da propriedade de objeto de mídia associado a classe de dispositivos NCL)	**getObjectPropertyValue**(*string* devClassId, *string* objId, *string* paramId)
string (valor da URL a ser repassada para um exibidor local, relativo a um serviço de captura de conteúdo oferecido por um dispositivo pareado)	**getObjectResourceURL**(*string* devClassId, *string* objId, *string* objType, *string* anchorId)
void	**registerDeviceClassEventListener**(*DeviceClassEventListener* *dcel)
void	**registerDeviceClassObjectEventListerner**(*DeviceClassObjectEventListerner* *dcoel)

Tabela 5 – Descrição da API para componente *Device Integration Manager* (interface *IDeviceIntegrationManager*)

A interface oferecida pelo componente de integração de dispositivos da plataforma Ginga-MD (*Device Integration Manager*) tem sua API apresentada na Tabela 2 e estabelece os mecanismos de comunicação entre o exibidor de aplicações hipermídia distribuídas (*NCL Presentation System*) e os serviços de orquestração de recursos de dispositivos remotos. Tais recursos estão associados às classes de dispositivos NCL através de seus objetos de mídia. A API apresentada na Tabela 2, portanto, estabelece mecanismos para:

- Gestão de classes de dispositivo NCL – para que o *NCL Presentation System* possa definir quais classes de dispositivos são utilizadas durante a execução de aplicações NCL (métodos *addDeviceClass* e *removeDeviceClass*), cujo formato de descrição utilizado como parâmetro é discutido no Capítulo 4.

- Orquestração de objetos de mídia das classes de dispositivos NCL – funcionalidades utilizadas pelo *NCL Presentation System* para associação e dissociação de objetos de mídia a classes de dispositivos (métodos *addObject* e *removeObject*), notificação de transição da máquina de estados dos eventos NCL associados aos objetos (método *postAction*), recuperação do estado dos eventos NCL de um objeto de mídia associado a uma classe de dispositivos (método *getObjectState*), recuperação do valor de propriedade de objeto de mídia associado a uma classe de dispositivos (método *getObjectPropertyValue*) e para recuperar URL a ser utilizada por um exibidor de mídia local (e.g. um *software* reprodutor de áudio) e que referencia serviço de captura de mídia oferecido por uma instância específica de subsistema *Secondary Device Subsystem* (método *getObjectResourceURL*).
- Controle das variáveis NCL associadas às classes de dispositivos – funcionalidades para que o *NCL Presentation System* possa recuperar a quantidade de dispositivos registrados em uma classe (método *getDeviceClassSize*) e também de outras propriedades definidas pela plataforma Ginga-MD (método *getDeviceClassPropertyValue*).
- Notificação de eventos relacionados às classes de dispositivos – através do registro de ouvintes (*listeners*) para notificações dos serviços associados às classes de dispositivos (método *registerDeviceClassObjectEventListener*), bem como para notificações acerca das variáveis NCL associadas aos parâmetros (e.g. quantidade de dispositivos pareados) das classe de dispositivos (método *registerDeviceClassEventListener*).

5.2.2.3.2. Componentes dos Dispositivos Secundários

A seguir são descritos os componentes mais relevantes que compõem o *Secondary Device Subsystem,* os quais são apresentados no diagrama de componentes da Figura 9.

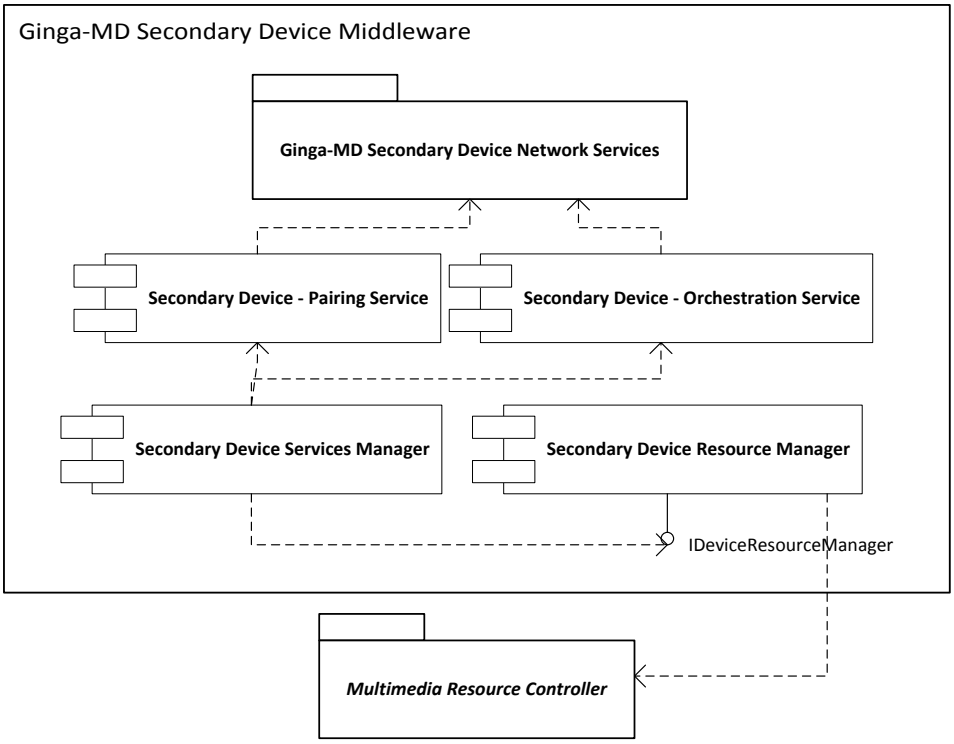


Figura 9 – Diagrama de componentes para o *Secondary Device Subsystem*

Multimedia Resource Controller é o componente responsável por controlar um recurso multimídia oferecido pelo dispositivo hospedeiro do *Secondary Device Subsystem*. Esse componente é embutido pelo componente *Secondary Device Resource Manager* (interno do *middleware* de Dispositivo Secundário da plataforma Ginga-MD), que oferece uma interface comum de controle (*IDeviceResourceManager*) para os diferentes recursos orquestráveis.

A orquestração de recursos multimídia está associada ao uso de serviços de orquestração e de pareamento com dispositivos orquestradores (Dispositivo Pai). O componente *Secondary Device Services Manager* encapsula instâncias de controladores de recurso que disponibilizam a interface *IDeviceResourceManager* e é responsável por associar os recursos multimídia aos diferentes serviços que viabilizam a orquestração dos recursos oferecidos pelo Dispositivo Secundário (serviços de orquestração e pareamento, componente *Secondary Device Orchestration Service* e *Secondary Device Pairing Service*). A descrição da API para a interface *IDeviceResourceManager* é apresentada a seguir (Tabela 3). A modelagem da interface seguiu as definições utilizadas para a API da interface utilizada pelos os exibidores NCL (Moreno 2011).

Retorno do método	**Operação e parâmetros de entrada**
void	**start(***string* objId, *list<string, string>* <propertyName, propertyValue>, *string* objSrc, *string* delay)
void	**stop()**
void	**abort()**
void	**pause()**
void	**resume()**
string (valor da propriedade consultada)	**getPropertyValue(***string* propertyName)
int (código de resultado da definição de propriedade)	**setPropertyValue(***string* propertyName, *string* propertyValue)
int (código de resultado para envio de ação)	**postAction**(*string* objId, *string* objType, *int* nclEventType, *string* anchorId, *int* action, *string* value)
void	**registerActionListener(***MultimediaResource ActionListener* *dal**)**

Tabela 6 – Descrição da API para componente *Secondary Device Resource Manager* (interface *IDeviceResourceManager*).

5.2.2.3.3. Componentes do *Application Provider Subsystem*

Os componentes mais relevantes do *Application Provider Subsystem* (dentro do contexto da plataforma Ginga-MD) são apresentados no diagrama de componentes representado na Figura 10.

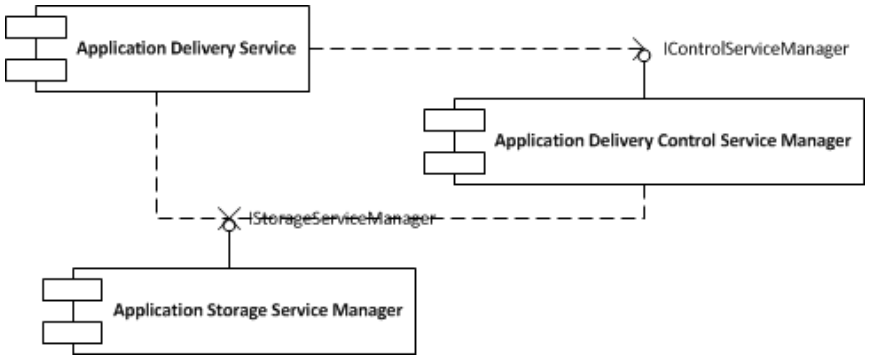


Figura 10 – Diagrama de componentes para o *Application Provider Subsystem*

Application Delivery Service é o componente responsável por enviar e receber (se houver suporte) informações da Rede de Distribuição, que é utilizada para entregar aplicações NCL às instâncias de *Base Device Subsystem*. Fornece mecanismos para entrega de aplicações compatíveis com a Rede de Distribuição.

Application Delivery Control Service Manager é o componente que oferece funcionalidades para o controle do serviço de entrega, definindo os parâmetros de configuração para o serviço (para integração com a Rede de Distribuição utilizada), e assim controlando a entrega e a atualização de aplicações realizadas pelo *Application Delivery Service*.

Application Storage Service Manager é o componente que oferece funcionalidades para controle e acesso do repositório de arquivos das aplicações NCL cadastradas.

6
Implementação do Protótipo

Este Capítulo descreve a implementação de um protótipo para a plataforma proposta por esta tese. São apresentados detalhes do desenvolvimento dos componentes significativos da plataforma Ginga-MD, especificados no Capítulo 5.

O desenvolvimento desta tese foi vinculada às pesquisas realizadas para a especificação do *middleware* Ginga, que é o motor de interatividade do Sistema Brasileiro de TV Digital Terrestre (ISDB-Tb) e Recomendação ITU para serviços interativos IPTV e terrestre. A implementação de referência original do Ginga, aderente às especificações vigentes (ABNT 2007), foi também utilizada para o desenvolvimento de parte dos subsistemas que serão apresentados a seguir. A implementação do protótipo da plataforma Ginga-MD incorporou os motores de exibição da linguagem NCL da sua implementação de referência (o subsistema Ginga-NCL, requisito obrigatório do Ginga) (Soares 2007) e também o motor de exibição NCL para dispositivos Android (Daher 2010).

Da mesma forma que NCL e seus motores de exibição (como o Ginga-NCL) facilitam a integração de diferentes exibidores de mídia, esses deveriam também facilitar a integração de diferentes serviços de rede utilizados para comunicação com dispositivos a serem integrados em uma aplicação hipermídia distribuída. NCL oferece uma semântica de alto nível para um conjunto de mensagens de orquestração (transições das máquinas de estado definidas para os objetos de mídia NCL) (Soares 2005; Soares 2007) que acomoda o controle de recursos locais e também recursos (serviços) remotos. O protótipo da plataforma apresentado neste capítulo utiliza a semântica das mensagens de orquestração da linguagem NCL para criar mecanismos registro e comunicação, de forma a acomodar diferentes dispositivos e serviços de rede, e integrá-los na execução de uma aplicação hipermídia distribuída.

A implementação do protótipo da plataforma Ginga-MD valida os conceitos definidos para o modelo de orquestração distribuída da linguagem NCL que não haviam sido realizados na prática (através das implementações dos seus motores de exibição, conforme apresentado na Seção 3.1.1). Além disso, a elaboração dos cenários (Capítulo 2) teve como mote extrapolar as

possibilidades que já eram oferecidas pela linguagem NCL, assim, os acréscimos à linguagem apresentados no Capítulo 4 foram implementados no protótipo.

O protótipo apresentado neste capítulo é a realização de uma instância da arquitetura conceitual apresentada no Capítulo 5. A implementação do protótipo é conforme os testes apresentados no Capítulo 7 e, portanto configura uma validação formal da arquitetura de *software* modelada para atender os requisitos definidos que sustentam a solução proposta por esta tese.

Definições de recuperação e tolerância a falhas para aplicações hipermídia distribuídas foram introduzidas na implementação do protótipo da plataforma Ginga-MD, e são discutidas na Seção 6.1. A Seção 6.2 apresenta o comportamento dinâmico dos subsistemas protótipo, em função dos processos que executam (a Visão de Processos).

6.1. Recuperação e Tolerância a falhas na execução de aplicações hipermídia distribuídas

O plano de recuperação utilizado na implementação de referência do Ginga tem por objetivo tornar tolerante a falhas não apenas o subsistema em si, mas também a apresentação de aplicações interativas NCL que o subsistema executa. Assim, como apresentado em (Moreno 2010a), os módulos da arquitetura do subsistema Ginga-NCL (Soares 2007) foram classificados com o objetivo de definir agrupamentos funcionais para a aplicação de técnicas de recuperação.

Os módulos de Risco são aqueles que podem comprometer a confiabilidade da execução do Ginga-NCL, por agregarem bibliotecas de terceiros ou serem atualizados no processo de evolução dinâmica do middleware (Soares 2007) (exemplo: módulos dos exibidores, *parser* XML, máquina Lua etc.).

Os módulos de Recuperação do Ginga-NCL são aqueles responsáveis pela detecção, controle e recuperação de falhas de software. O módulo Gerente de Recuperação é o único módulo desse conjunto que, internamente, divide-se em outros módulos (Moreno 2010a), que atuam sobre os outros conjuntos de módulos do Ginga-NCL para criar o plano de recuperação e tolerância a falhas, utilizando técnicas de recuperação proativa e reativa. A versão da implementação de referência do Ginga cujos componentes foram incorporados pelos subsistemas da plataforma Ginga-MD já suportava a tolerância e recuperação de falhas durante a execução de

aplicações NCL, suporte que foi estendido como parte das pesquisas que resultaram nesta tese, de forma que também as aplicações distribuídas dispusessem de mecanismos de recuperação.

Esta seção descreve o plano de recuperação adotado pela implementação de referência do *middleware* Ginga e em seguida apresenta quais são as definições de tolerância a falhas e ações de recuperação durante a execução de aplicações NCL em múltiplos dispositivos.

6.1.1. Recuperação proativa na implementação de referência do Ginga (Moreno 2010a)

Na recuperação proativa as técnicas de recuperação são aplicadas pelo componente Qualificador através de rejuvenescimento dos módulos de Controle. Quando um módulo de Controle é rejuvenescido, os módulos de Apresentação e os módulos de Risco, controlados por esse módulo de Controle, são, indiretamente, rejuvenescidos.

Para definir quando realizar o rejuvenescimento de cada módulo de Controle, sem comprometer a disponibilidade do sistema, foi definido um componente Monitor, que deve ser cadastrado como ouvinte do comportamento dos módulos de Controle. Quando alguma operação que possibilita rejuvenescimento é permitida nesses módulos, o Monitor é notificado. O Monitor então repassa a informação para que o componente Qualificador avalie se há alguma ação de recuperação a ser realizada.

Na avaliação realizada pelo Qualificador, pode ser levada em consideração uma política de recuperação proativa especificada para o dispositivo receptor, através de uma consulta ao componente Política. As políticas tratadas por esse componente estão comumente associadas às restrições do ambiente de execução. Por exemplo, um dispositivo receptor capaz de decodificar apenas um vídeo (de um determinado formato) por vez deve possuir como política a existência de apenas um exibidor instanciado com tais características. Antes de executar a ação de recuperação, o Qualificador consulta o estado dos módulos de Controle para certificar-se que o rejuvenescimento pode ser realizado.

6.1.2. Recuperação reativa na implementação de referência do Ginga (Moreno 2010a)

Na recuperação reativa, um Monitor de falhas foi definido para receber notificações de falhas ocorridas nos módulos de Risco. Para isso, o Monitor é cadastrado como ouvinte de falhas

dos módulos de Risco. Ao receber uma notificação, o Monitor repassa a informação de falha ao Identificador de Falha. De posse das informações sobre a falha, o Identificador de Falha solicita ao Registrador que a falha seja registrada no sistema.

O Registrador de falhas retorna ao Identificador de Falha quantas vezes a falha ocorreu em um intervalo de tempo. Com essa informação, o Identificador de Falha consulta a Política de recuperação reativa para avaliar se a falha deve ser tratada. Por exemplo, um dispositivo receptor pode definir como política que uma determinada falha não seja mais tratada após um determinado número de tentativas de recuperação. Caso o Identificador de Falha defina que a falha não seja tratada, nenhuma ação de recuperação será realizada. Caso contrário, o Identificador de Falha repassa as informações sobre a falha para o Identificador de Ação. Ao receber as informações sobre a falha ocorrida, o Identificador de Ação avalia qual a ação de recuperação a ser realizada. O Identificador de Ação então solicita ao Validador que o estado do módulo a ser recuperado seja validado. Para isso, o Validador consulta os módulos de Controle e de Apresentação, retornando as informações para o Identificador de Ação. O estado retornado pelo Validador pode indicar que, apesar da falha ocorrida, o módulo que consiste na origem da falha realizou suas operações corretamente. Nesse caso, nenhuma ação de recuperação será realizada. Caso contrário, após definir a ação de recuperação e adquirir as informações necessárias para realizá-la, o Identificador de Ação aciona o Recuperador.

O Recuperador é responsável por garantir que a ação de recuperação recebida através do Identificador de Ação seja realizada. Antes de executar a ação de recuperação, no entanto, o Recuperador consulta a Política de recuperação reativa para determinar os requisitos da ação. Um exemplo de requisito é o intervalo de tempo em que a ação de recuperação é válida. De posse de todas as informações necessárias, o Recuperador realiza a ação de recuperação sobre o módulo de Risco origem da falha e, em seguida, notifica o módulo de Apresentação, que controla esse módulo de Risco, que uma recuperação foi realizada.

6.1.3. Falhas de comunicação na execução de aplicações multi-dispositivo

Plataformas para execução de aplicações hipermídia multi-dispositivo são sistemas distribuídos propensos a um conjunto significativo de falhas, muitas das quais são passíveis de tratamento e recuperação. Sistemas distribuídos estão suscetíveis a falhas arbitrárias nas

entidades computacionais envolvidas, falhas nos meios de comunicação, atrasos na comunicação e também a problemas de consenso (estado compartilhado) entre as entidades distribuídas. O tratamento das falhas relacionadas à comunicação entre os dispositivos é de suma importância quando o sistema distribuído em questão dá suporte à execução de uma aplicação hipermídia, principalmente para a manutenção da consistência temporal definida pelo autor para a aplicação.

Como discutido anteriormente, a implementação de referência do subsistema Ginga-NCL suporta a tolerância e recuperação de falhas, especificando um plano de recuperação. Tal plano torna a implementação de referência resiliente, oferecendo também resiliência para as aplicações interativas em execução. O tratamento das falhas discutido, no entanto, contemplava originalmente apenas os componentes do Ginga-NCL relacionados à execução de aplicações locais.

As subseções seguintes discorrem sobre um conjunto representativo de falhas que podem acontecer durante a apresentação de uma aplicação Ginga-NCL multi-dispositivo, as quais foram consideradas para a implementação de mecanismos de proteção adicionais para a implementação de referência do Ginga-NCL, incorporada pela plataforma Ginga-MD. Munido de tais mecanismos, o plano de recuperação disponível para um dispositivo base tratará também falhas causadas por dispositivos secundários quando da execução de aplicações NCL multi-dispositivo. Tais falhas foram elencadas e tratadas para as Classes Ativas e para a Classe Passiva da plataforma Ginga-MD.

6.1.3.1. Falhas de comunicação: Classe Passiva

Para realizar a orquestração de dispositivos registrados na Classe Passiva, o formatador NCL instancia apenas um objeto de apresentação a partir do objeto de mídia associado a uma classe de dispositivos passiva. O objeto de apresentação é transmitido (transcodificado) para todos os dispositivos registrados. O tratamento das notificações de entrada (interações do usuário) geradas pelos dispositivos em classes passivas não é individualizado (durante a apresentação a navegação é compartilhada) e um mesmo conteúdo é reproduzido nos dispositivos.

A implementação de referência do Ginga-NCL conta com mecanismos de recuperação e tolerância a falha de seus módulos exibidores (Moreno 2010a), o que inclui também o tratamento

local dos componentes que manipulam (renderizam e transcodificam) objetos de mídia a serem transmitidos para os dispositivos passivos.

Levando em consideração as características da comunicação com dispositivos registrados na Classe Passiva (Soares 2009), a implementação do módulo de orquestração utilizado pelo Dispositivo Pai realiza a comunicação com dispositivos passivos por meio de datagramas UDP *multicast*, sobre uma rede IP *multicast*, com a utilização do protocolo IGMP para criação de grupos de dispositivos que integram a Classes Passivas (outros protocolos, como RTP, também podem ser utilizados para dar suporte ao modelo de comunicação passiva). Uma vez que um Dispositivo Pai inicia a execução de um objeto de mídia associado à Classe Passiva, o fluxo de bits oriundo do adaptador de exibidor (parte do *Multimedia Backend*, um renderizador ou transcodificador) é empacotado em datagramas e enviado em uma taxa adequada para o grupo multicast que foi criado para agregar os dispositivos da classe.

O Dispositivo Pai deve ser capaz de detectar quando uma classe passiva não possui mais membros (e.g. em grupos IP multicast através de uma mensagem IGMP "*group specific query*") (Costa 2009; Soares 2009), o que deve ser tratado como uma falha quando essa classe está associada a um objeto de mídia ainda em execução.

6.1.3.2. Falhas de comunicação: Classe Ativa

Durante a comunicação com os dispositivos registrados nas Classes Ativas, o formatador NCL em um Dispositivo Pai controla remotamente exibidores de mídia, determinando quais objetos de mídia esses exibidores reproduzirão com controle local. Cada Dispositivo Filho associado a uma Classe Ativa instancia seu próprio objeto de apresentação, navegando individualmente e notificando o formatador no Dispositivo Pai sobre as transições das máquinas de estado dos eventos dos objetos de mídia orquestrados remotamente.

A comunicação necessária para a orquestração (envio e recebimento de notificações) de dispositivos registrados nas Classes Ativas é feita através de um canal *unicast* (troca de mensagens via TCP) e cada dispositivo possui um identificador único. A quantidade máxima e mínima de dispositivos definida para uma Classe Ativa pode ser delimitada (vide Capítulo 4). Como um objeto de apresentação é instanciado em cada Dispositivo Filho registrado nas Classes Ativas, as falhas que ocorrerem nos exibidores remotos devem ser detectadas e notificadas ao Dispositivo Pai.

Falhas na orquestração de dispositivos registrados nas Classes Ativas podem acontecer durante o registro dos dispositivos e durante a troca de mensagens relativas à apresentação distribuída. Uma possível situação de falha se dá quando um dispositivo deseja se registrar em uma Classe Ativa e a mesma já possui objetos de mídia em execução. Se o registro se proceder normalmente, o dispositivo não receberia as notificações de orquestração anteriores ao seu registro. A mesma inconsistência de estado pode ser causada por falhas arbitrárias nos exibidores durante a apresentação nos dispositivos registrados em Classes Ativas, o que deve ser detectado pelo orquestrador no Dispositivo Pai e tratado como falha.

Como mencionado em (Batista 2010), um objeto de mídia associado a uma Classe Ativa só pode receber notificações de transição de seu evento de apresentação se a quantidade de dispositivos registrados na classe for compatível com as da definição da classe (e maior que zero). O valor é dinâmico e durante uma apresentação distribuída, por um conjunto de fatores não determinísticos, o limite inferior pode ser violado, o que deve ser tratado como falha.

6.1.4. Recuperação e tolerância a falhas: NCL em múltiplos dispositivos

A recuperação proativa é realizada pelo rejuvenescimento dos serviços de comunicação necessários para a orquestração de múltiplos dispositivos. Os serviços que dão suporte ao módulo Gerente de Classes de Dispositivos do subsistema do Dispositivo Pai são monitorados pelo módulo de Recuperação Proativa que, quando necessário (por exemplo, quando um serviço de rede torna-se indisponível) realiza rejuvenescimento do serviço.

Os mecanismos adicionais, que serão descritos nesta seção, foram desenvolvidos no intuito de viabilizar que o Gerente de Classes de Dispositivos notificasse o Gerente de Recuperação (componente do Módulo de Recuperação) das falhas ocorridas na comunicação com Dispositivos Secundários filhos, viabilizando a recuperação e tratamento de falhas durante a apresentação de uma aplicação hipermídia distribuída, baseado em políticas pré-definidas. As falhas consideradas durante a comunicação para a orquestração de dispositivos secundários demandam recuperação reativa.

As falhas descritas anteriormente são detectadas pelo Gerente de Integração de Dispositivos (parte do middleware do Dispositivo Pai) e notificadas ao Gerente de Recuperação através do Gerente de Classes de Dispositivos (parte do NCL Presentation System). Ao receber

uma notificação de falha, o Módulo de Recuperação as classifica conforme os tipos apresentados na coluna "Falha" da Tabela 7 e realiza seu registro e consulta da política de recuperação reativa. A definição da política oferece apenas a opção de quantas falhas podem ocorrer e em um período de tempo. Caso esses valores não sejam definidos, não é realizada a validação da política. Após a consulta da política de recuperação, de acordo com o tipo de falha, é definida a ação, como apresentado na coluna "Ação" na Tabela 7.

Classe	Falha	Ação
Ativas e Passiva	Não há mais dispositivos registrados a uma classe que possui uma mídia associada em execução.	Mídia é abortada.
Ativas	Número de dispositivos da classe torna-se inferior ao de sua definição.	Mídia é abortada.
Ativas	Dispositivo apresenta estado inválido (inconsistência temporal).	Dispositivo Pai envia mensagens de sincronismo para Dispositivo Secundário filho de acordo com andamento da apresentação.

Tabela 7 – Tipos de comunicação, Tipos de Falhas e Respectivas Ações.

Um objeto de mídia deve ser abortado (Tabela 7) quando a classe de dispositivos ao qual está associado passa a ter um número de dispositivos inferior ao de sua definição (no caso das classes ativas) ou quando o número de dispositivos torna-se zero (em ambas as classes). O Gerente de Recuperação implementado possui uma interface de comunicação com o Formatador NCL e notifica-o para garantir que os relacionamentos do objeto não sejam mais considerados na apresentação.

Para que o estado dos dispositivos filhos não seja inválido, os dispositivos devem receber informações sobre o andamento da apresentação do dispositivo pai. O Módulo de Recuperação se comunica com o Formatador NCL e o Módulo de Integração de Dispositivos para que o dispositivo pai possa sincronizar os dispositivos filhos, o que deve ser feito através cadeias

temporais e Grafos Temporais Hipermídia (GTH) (Costa 2008), que oferecem suporte ao início ou retomada síncrona da apresentação. No entanto, na versão da implementação de referência incorporada na plataforma Ginga-MD, o suporte ao GTH não está disponível, de forma que o sincronismo é mantido pelo Gerente de Classes de Dispositivos do Dispositivo Pai através do registro das mensagens de orquestração das mídias associadas a uma classe de dispositivos, que é entregue a um Dispositivo Secundário em caso de detecção de inconsistência temporal.

6.2. Visão de Processos

A visão de processos (ou de concorrência) descreve o comportamento dinâmico dos subsistemas e dos módulos que implementam os subsistemas que são previamente descritos na visão lógica. Os principais processos e *threads* dos componentes de cada subsistema da implementação corrente são apresentados através de diagramas de sequência, que visam elencar quais são as responsabilidades e relacionamentos entre os módulos relevantes para a realização dos principais casos de uso definidos para a plataforma. A implementação corrente reflete a arquitetura o suficiente para validar os requisitos estabelecidos para a plataforma Ginga-MD.

Para representar suficientemente o comportamento esperado para o sistema modelado, os diagramas de sequência propostos definem um contexto específico para os relacionamentos relevantes entre os módulos do sistema. O contexto de uma sequência é composto pelo conjunto de mensagens que podem ser trocadas entre os módulos do sistema e pela semântica de cada mensagem (como está associada à responsabilidade dos módulos para concretizar uma funcionalidade da plataforma).

São apresentados a seguir os diagramas de sequência para os casos de uso para a plataforma Ginga-MD. Os diagramas de sequência enfatizam a ordem cronológica das alocações dos objetos e das interações entre eles, definindo também quais mensagens são utilizadas nessas interações, considerando uma linha de vida para cada objeto ativo. Contempla-se assim um conjunto representativo de informações sobre os aspectos dinâmicos e comportamentais dos casos de uso em função dos componentes dos subsistemas da plataforma.

As informações reunidas nesta seção focam em aspectos que servem de base para analisar o modelo de concorrência, além de aspectos de desempenho e escala para o sistema distribuído definido nesta tese. A partir da definição dos pontos de disparo e qual a semântica de mensagens

utilizada nos casos de uso, estabelecem-se critérios relevantes para validar suas etapas de execução, o que será utilizado (no Capítulo 7 – "Testes Sistêmicos") para a prova de conceito associada à implementação do protótipo descrita nesta seção.

6.2.1. Casos de uso relacionados ao carregamento de aplicações NCL

Esta subseção agrega diagramas de sequência relacionados aos casos de uso que lidam com o carregamento de aplicações NCL por parte do *Base Device Subsystem*.

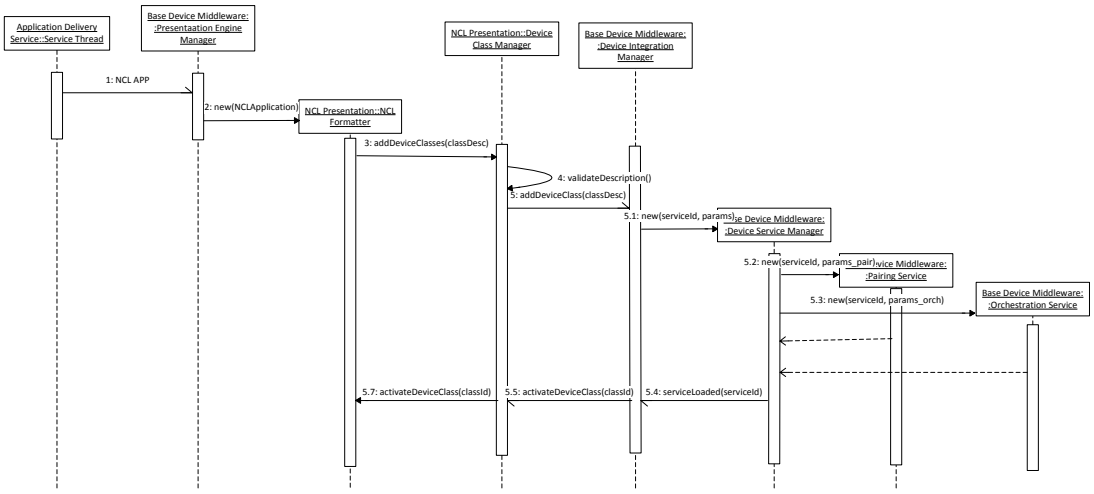


Figura 11 – Diagrama de sequência para processo de carregamento de aplicação NCL realizado pelo *Base Device Subsystem*

O diagrama de sequência apresentado na Figura 11 representa um processo realizado pelo *Base Device Subsystem* para descarregar aplicações presentes no repositório do *Application Provider Subsystem* e carregar os serviços associados às classes de dispositivos que esta aplicação referencia. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.1, 4.2.1.2 (para o *Base Device Subsystem*), 4.2.1.3.2 e 4.2.1.3.3 (para o *Application Provider Subsystem*). A seguir, são detalhadas as etapas apresentadas no diagrama da Figura 11:

1. *Thread* do Serviço de Entrega do *Application Provider Subsystem* entrega a especificação de uma aplicação NCL e os arquivos necessários para sua inicialização (via *pushed*

data ou *pulled data*, como definido nos casos de uso do *Application Provider Subsystem* apresentados nas seções 4.2.1.3.2 e 4.2.1.3.3).

2. Componente *Presentation Engine Manager* (parte do *middleware* do Dispositivo Base) armazena conteúdo da aplicação e entrega referência para o Formatador NCL (*NCL Formatter*).

3. Formatador NCL identifica quais classes de dispositivos estão associadas à aplicação NCL carregada e registra as descrições das classes de dispositivo NCL junto ao Gerente de Classes de Dispositivos (*Device Class Manager*, parte do *NCL Presentation System*).

4. Gerente de Classes de Dispositivos valida a descrição de classes de dispositivos NCL.

5. Gerente de Classes de Dispositivos notifica Gerente de Integração de Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Base) para que este, a partir da descrição das classe de dispositivos NCL carregadas, ative os serviços do *Base Device Subsystem* relacionados a estas classes.

5.1. Gerente de Integração de Dispositivos solicita ao Gerente de Serviços de Dispositivos (componente *Device Service Manager*) que crie os serviços de pareamento e orquestração associados a uma classe de dispositivos carregada.

5.2. Gerente de Serviços de Dispositivos instancia serviço de pareamento associado a uma classe de dispositivos carregada.

5.3. Gerente de Serviços de Dispositivos instancia serviço de orquestração associado a uma classe de dispositivos carregada.

5.4. Gerente de Serviços de Dispositivos notifica Gerente de Integração de Dispositivos do sucesso da criação dos serviços associados à classe de dispositivos referenciada pela aplicação NCL carregada.

5.5. Gerente de Integração de Dispositivos notifica Gerente de Classes de Dispositivos da ativação da classe de dispositivos.

6. Formatador NCL processa ativação da classe de dispositivos NCL (a partir de notificação do Gerente de Classes de Dispositivos).

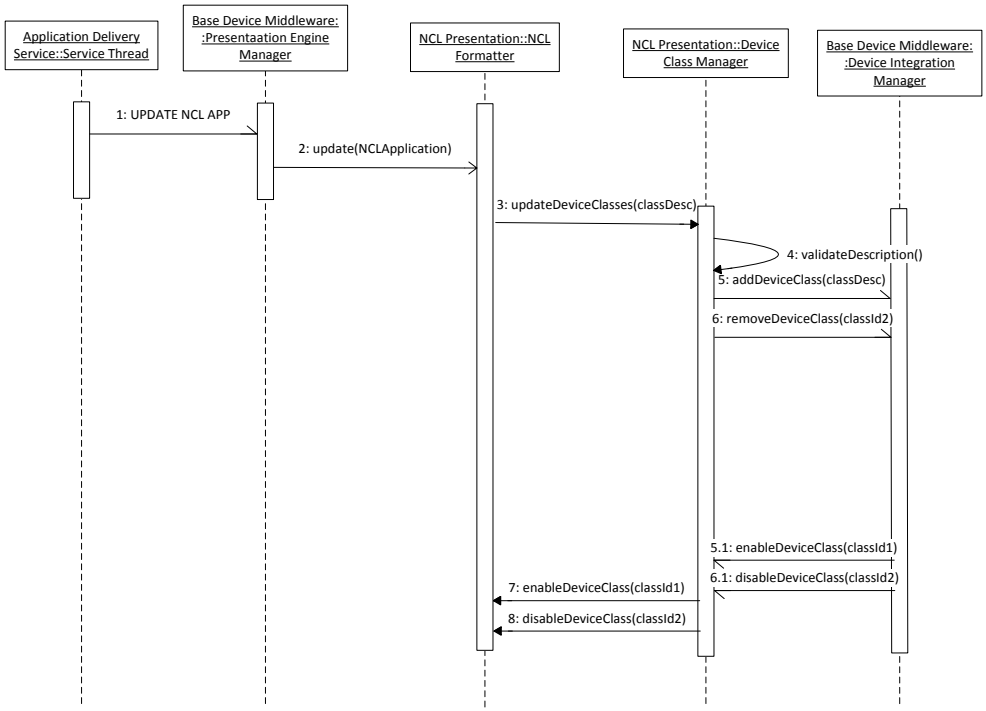


Figura 12 – Diagrama de sequência para o processo de atualização de aplicação NCL em execução no *Base Device Subsystem*.

O processo de atualização de aplicações é caracterizado pelo diagrama de sequência apresentado na Figura 12. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.3 (para o *Base Device Subsystem*) e 4.2.1.3.4 (para o *Application Provider Subsystem*). A seguir, são detalhadas as etapas apresentadas no diagrama da Figura 12:

1. *Thread* do Serviço de Entrega (*Application Delivery Service)* do *Application Provider Subsystem* notifica componente *Presentation Engine Manager* (parte do *middleware* do Dispositivo Base) acerca da atualização da especificação da aplicação NCL em execução.

2. Componente *Presentation Engine Manager* notifica modificações ao Formatador NCL (*NCL Formatter*).

3. Formatador NCL identifica quais classes de dispositivos foram ou deixaram de ser associadas à aplicação NCL atualizada, e registra as descrições das classes de dispositivo

utilizadas pela versão atualizada da aplicação junto ao Gerente de Classes de Dispositivos (*Device Class Manager*, parte do *NCL Presentation System*).

4. Gerente de Classes de Dispositivos valida nova descrição de classes de dispositivos NCL.

5. Gerente de Classes de Dispositivos notifica Gerente de Integração de Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Base) do acréscimo de classes de dispositivos NCL na nova versão da aplicação NCL carregada, para que o Gerente de Integração de Dispositivos ative os serviços do *Base Device Subsystem* relacionados às classes de dispositivos da nova versão aplicação NCL carregada.

5.1. Gerente de Integração de Dispositivos notifica Gerente de Classes de Dispositivos da ativação da classe de dispositivos (mais detalhes desse processo são apresentados no diagrama da Figura 13).

6. Gerente de Classes de Dispositivos notifica Gerente de Integração de Dispositivos da não mais utilização de classes de dispositivos NCL na aplicação NCL atualizada, para que o Gerente de Integração de Dispositivos desative os serviços do *Base Device Subsystem* relacionados à classe de dispositivos removida.

6.1. Gerente de Integração de Dispositivos notifica Gerente de Classes de Dispositivos da desativação da classe de dispositivos (mais detalhes desse processo são apresentados no diagrama da Figura 13).

7. Formatador NCL processa ativação da classe de dispositivos NCL (a partir de notificação do Gerente de Classes de Dispositivos).

8. Formatador NCL processa desativação da classe de dispositivos NCL (a partir de notificação do Gerente de Classes de Dispositivos).

O processo de ativação e desativação de serviços associados à atualização de uma aplicação NCL é detalhado no diagrama de sequência da Figura 13.

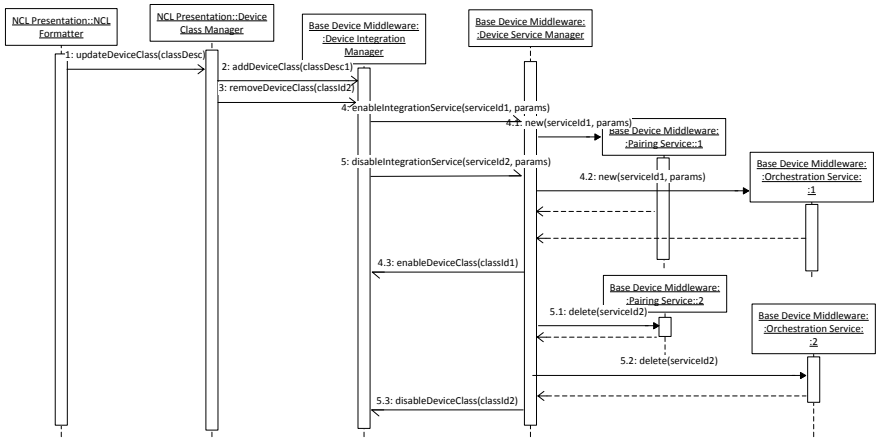


Figura 13 – Diagrama de sequência para processo de ativação e desativação de serviços das classes de dispositivos (quando da atualização da aplicação NCL em execução no *Base Device Subsystem*).

As etapas apresentadas no diagrama da Figura 13 são apresentadas a seguir:

1. Formatador NCL (componente *NCL Formatter*, parte do *NCL Presentation System*) registra descrições de classes de dispositivo NCL (utilizadas pela versão atualizada da aplicação) junto ao Gerente de Classes de Dispositivos (*Device Class Manager*, parte do *NCL Presentation System*).

2. Gerente de Classes de Dispositivos, após validação (etapa 4 do diagrama de sequências da Figura 12), notifica Gerente de Integração de Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Base) do acréscimo de uma classe de dispositivos NCL na nova versão da aplicação NCL carregada, para que assim o Gerente de Integração de Dispositivos ative os serviços do *Base Device Subsystem* relacionados às classes de dispositivos da aplicação NCL atualizada.

3. Gerente de Classes de Dispositivos notifica Gerente de Integração de Dispositivos da não mais utilização de uma classe de dispositivos NCL na aplicação NCL atualizada, para que o Gerente de Integração de Dispositivos desative os serviços do *Base Device Subsystem* relacionados à classe de dispositivos removida.

4. Gerente de Integração de Dispositivos notifica Gerente de Serviços de Dispositivos (*Device Service Manager*, parte do *middleware* do Dispositivo Base) da ativação de classe de dispositivos.

4.1. Gerente de Serviços de Dispositivos instancia serviço de pareamento associado à classe de dispositivos carregada na atualização.

4.2. Gerente de Serviços de Dispositivos instancia serviço de orquestração associado à classe de dispositivos carregada na atualização.

4.3. Gerente de Serviços de Dispositivos notifica Gerente de Integração de Dispositivos do sucesso da criação dos serviços associados à classe de dispositivos referenciada pela aplicação NCL atualizada.

5. Gerente de Integração de Dispositivos notifica Gerente de Serviços de Dispositivos da desativação de classe de dispositivos.

5.1. Gerente de Serviços de Dispositivos remove instância de serviço de pareamento associado à classe de dispositivos descarregada na atualização.

5.2. Gerente de Serviços de Dispositivos remove instância de serviço de orquestração associado à classe de dispositivos descarregada na atualização.

5.3. Gerente de Serviços de Dispositivos notifica Gerente de Integração de Dispositivos do sucesso da desativação dos serviços associados à classe de dispositivos não mais referenciada pela aplicação NCL atualizada.

6.2.2. Casos de uso relacionados ao pareamento com Dispositivos Secundários

Esta subseção agrega diagramas de sequência dos casos de uso relacionados ao pareamento entre Dispositivos Pai (dispositivo com *Base Device Subsystem* e *Secondary Device Subsystem* com o serviço *Fertile Active Device Service*) e Dispositivos Secundários (*Secondary Device Subsystem*).

O processo de pareamento é caracterizado pelo diagrama de sequência apresentado na Figura 14. O diagrama de sequência da Figura 15 apresenta o processo de finalização do pareamento entre os subsistemas. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.4 e 4.2.1.2.1.1.

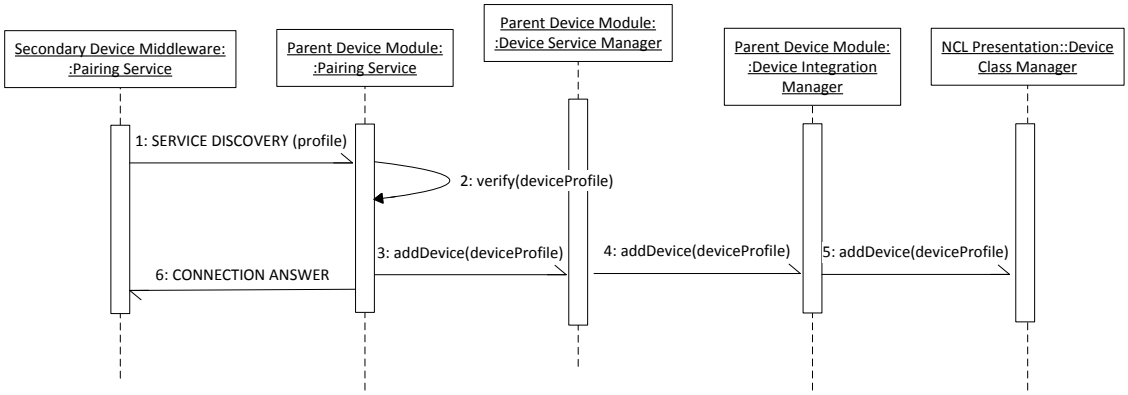


Figura 14 –. Diagrama de sequência para processo de pareamento entre Dispositivo Base e Dispositivo Secundário

As etapas caracterizadas pelo diagrama de sequência da Figura 14 são descritas abaixo:

1. Serviço de Pareamento do *Secondary Device Subsystem* (componente *Pairing Service*) realiza busca por serviço, informando qual o seu perfil (e, assim, qual serviço busca).

2. Serviço de Pareamento do subsistema do Dispositivo Pai (componente *Pairing Service*) recebe solicitação e verifica que há classe de dispositivos associada ao subsistema Dispositivo Secundário que realizou a busca.

3. Subsistema do Dispositivo Pai notifica *Secondary Device Subsystem* do sucesso do pareamento e estabelece os canais de comunicação necessários para a orquestração.

4. Serviço de Pareamento do subsistema do Dispositivo Pai notifica Gerente de Serviços de Dispositivo (componente *Device Service Manager*, parte do subsistema do Dispositivo Pai) do pareamento.

5. Gerente de Serviços de Dispositivo notifica o Gerente de Integração de Dispositivos (*Device Integration Manager*) do pareamento, e este atualiza as variáveis da classe de dispositivos e realiza a configuração dos canais de comunicação para orquestração dos recursos oferecidos pelo Dispositivo Secundário pareado.

6. Gerente de Serviços de Dispositivo verifica se há atividades de recuperação de sincronismos pendentes associadas ao novo pareamento (i.e. se dispositivo está restaurando pareamento realizado anteriormente, durante uma mesma execução de aplicação NCL).

7. *NCL Presentation System* é notificado, através do Gerente de Classes de Dispositivos (*Device Class Manager*) do pareamento de um novo dispositivo.

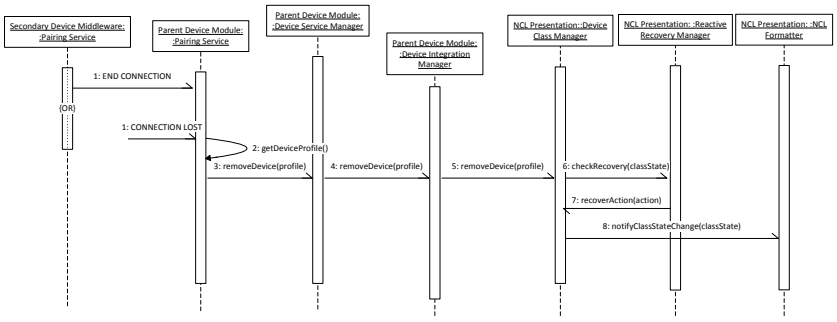


Figura 15 – Diagrama de sequência para o processo de finalização do pareamento entre os subsistemas.

As etapas realizadas para o fim do registro de um Dispositivo Secundário em uma classe de dispositivos no Dispositivo Pai são caracterizadas pelo diagrama de sequência da Figura 15:

1. Serviço de Pareamento do subsistema do Dispositivo Pai (componente *Pairing Service*) recebe mensagem de sinalização para fim de pareamento de instância de *Secondary Device Subsystem* ou detecta que a conexão com uma instância foi perdida (falha no canal de comunicação).

2. Serviço de Pareamento do subsistema do Dispositivo Pai recupera identificador associado ao dispositivo que teve o pareamento finalizado.

3. Serviço de Pareamento do Dispositivo Ascendente notifica Gerente de Serviços de Dispositivo (componente *Device Service Manager*, parte do *middleware* do Dispositivo Ascendente) do fim do pareamento.

4. Gerente de Serviços de Dispositivo notifica o Gerente de Integração de Dispositivos (*Device Integration Manager*) do fim do pareamento, e este atualiza as variáveis da classe de dispositivos NCL envolvida e realiza a configuração dos canais de comunicação para orquestração dos recursos oferecidos pelo subsistema pareado.

5. Gerente de Classes de Dispositivos (*Device Class Manager*) é notificado do pareamento de um novo dispositivo pelo Gerente de Integração de Dispositivos.

6. Gerente de Classes de Dispositivos verifica junto ao Gerente de Recuperação Reativa (componente *Reactive Recovery Manager*, parte do *NCL Presentation System*) se há atividades de recuperação associadas à finalização do pareamento realizada.

7. Gerente de Recuperação Reativa envia ações a serem realizadas para o Gerente de Classes de Dispositivos, para que este realize as ações de recuperação associadas.

8. Gerente de Classes de Dispositivos notifica Formatador NCL (*NCL Formatter*) das ações de recuperação realizadas.

6.2.3. Casos de uso relacionados à manutenção do sincronismo entre dispositivos pareados

Esta subseção contempla os diagramas de sequência para os processos associados aos casos de uso necessários para a manutenção do sincronismo entre dispositivos para a execução distribuída de uma aplicação NCL, obedecendo à hierarquia definida para os subsistemas da plataforma Ginga-MD. A manutenção do sincronismo é realizada através da orquestração feita por Dispositivos Pai (dispositivo com *Base Device Subsystem* ou *Secondary Device Subsystem* com o serviço *Fertile Active Device Service*) de serviços oferecidos por Dispositivos Secundários (*Secondary Device Subsystem*).

6.2.3.1. Classe Passiva

O envio e o recebimento de mensagens de orquestração do Dispositivo Pai para os seus Dispositivos Secundários filhos registrados na Classe Passiva é caracterizado pelo diagrama de sequência apresentado na Figura 16. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.5, 4.2.1.1.8, 4.2.1.2.2.1 e 4.2.1.2.2.2.

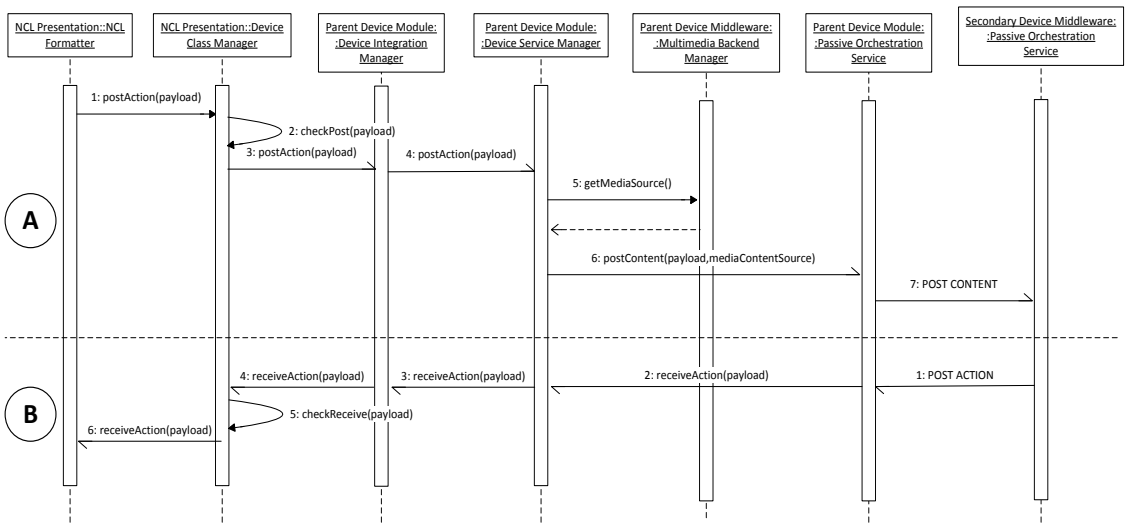


Figura 16 – Diagrama de sequência para as etapas de orquestração de objetos associados à Classe Passiva

As etapas apresentadas pelo diagrama de sequência da Figura 16 são descritas a seguir.

Sequência A – Envio de mensagens de orquestração do Dispositivo Pai para Dispositivos Secundários registrados na Classe Passiva:

1. Formatador NCL (componente *NCL Formatter* parte do *NCL Presentation System*) realiza uma transição em alguma das máquinas de estado de um objeto de mídia associado à Classe Passiva e notifica o Gerente de Classes de Dispositivos (componente *Device Class Manager*).

2. Gerente de Classes de Dispositivos verifica se modificação no estado dos objetos deve ser notificada ao Gerente de Integração de Dispositivos (componente *Device Integration Manager* do *middleware* do Dispositivo Base).

3. Gerente de Classes de Dispositivos notifica Gerente de Integração de Dispositivos que a alteração de estados do objeto de mídia provocou mudança no conteúdo a ser renderizado e enviado para os Dispositivos Secundários registrados na Classe Passiva.

4. Gerente de Integração de Dispositivos notifica Gerente de Serviços de Dispositivos (componente *Device Service Manager,* parte do *middleware* do Dispositivo Pai).

5. Gerente de Serviços de Dispositivos (através do seu componente controlador dos serviços associados à Classe Passiva) é notificado da alteração e solicita ao componente *Multimedia Backend Manager* que renderize os objetos de mídia associados à Classe Passiva e gere as amostras renderizadas em um fluxo codificado.

6. Gerente de Serviços de Dispositivos notifica componente controlador de Serviço de Orquestração Passiva (componente *Passive Device Orchestration Service*, parte do *middleware* do Dispositivo Pai).

7. Componente controlador de Serviço de Orquestração Passiva procede com o envio do fluxo codificado de amostras para Dispositivos Secundários registrados na Classe Passiva.

Sequência B – Envio de mensagens de orquestração de Dispositivo Secundário registrado na Classe Passiva para Dispositivo Pai:

1. Usuário do Dispositivo Secundário registrado na Classe Passiva interage (via mecanismo de entrada) com conteúdo recebido do Dispositivo Pai. *Secondary Device Subsystem* gera mensagem de notificação da interação e envia para o Dispositivo Pai ao qual se pareou.

2. Componente controlador de Serviço de Orquestração Passiva (componente *Passive Device Orchestration Service*, parte do *middleware* do Dispositivo Pai) recebe mensagem de interação e notifica o Gerente de Serviços de Dispositivo (componente *Device Service Manager*, parte do *middleware* do Dispositivo Pai).

3. Gerente de Serviços de Dispositivo notifica Gerente de Integração com Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Pai).

4. Gerente de Integração com Dispositivos notifica *NCL Presentation System* via Gerente de Classes de Dispositivos (*Device Class Manager*).

5. Gerente de Classes de Dispositivos verifica informações da mensagem de orquestração recebida.

6. Formatador NCL (*NCL Formatter*) é notificado e realiza a ação decorrente do processamento da mensagem de orquestração recebida.

6.2.3.2. Classes Ativas

O envio e o recebimento de mensagens de orquestração do Dispositivo Pai para os seus Dispositivos Secundários filhos registrados nas Classes Ativas é caracterizado pelo diagrama de sequência apresentado na Figura 17. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.6, 4.2.1.1.9, 4.2.1.2.3.2 e 4.2.1.2.3.3.

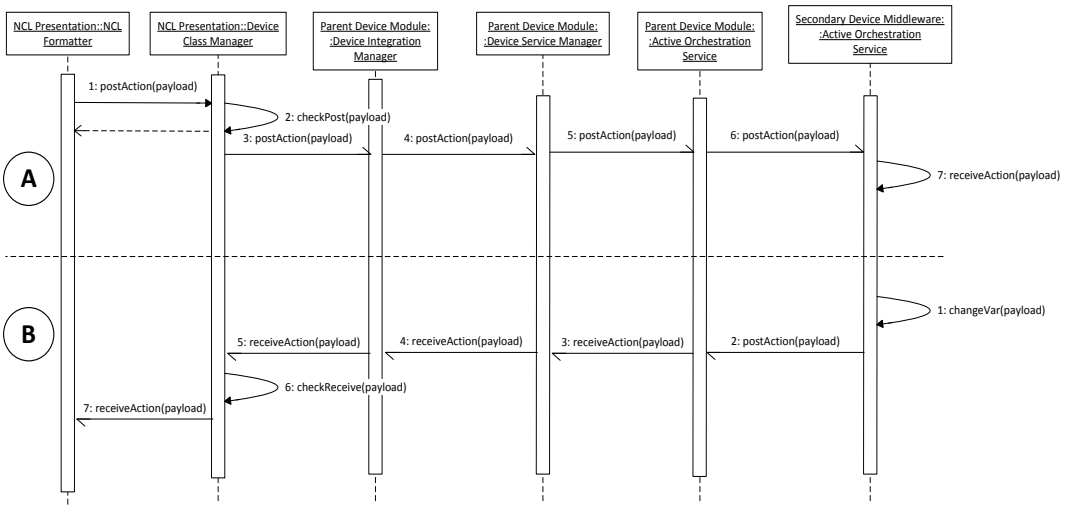


Figura 17 – Diagrama de sequência para as etapas de orquestração de objetos associados à Classe Ativa

As etapas apresentadas pelo diagrama de sequência da Figura 17 são descritas a seguir.

Sequência A – Envio de mensagens de orquestração do Dispositivo Pai para Dispositivos Secundários registrados em uma Classe Ativa:

1. Formatador NCL (componente *NCL Formatter* do *NCL Presentation System*) realiza transição em alguma das máquinas de estado de um objeto de mídia associado a uma Classe Ativa e notifica o Gerente de Classes de Dispositivos (*Device Class Manager*).

2. Gerente de Classes de Dispositivos valida o conteúdo associado à transição para gerar uma mensagem de orquestração contendo informações da transição de máquina de estados (ação, qual máquina de estados, qual objeto de mídia, qual âncora do objeto, quais parâmetros da transição).

3. Mensagem é repassada para o Gerente de Integração de Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Pai).

4. Gerente de Integração de Dispositivos processa a mensagem e notifica ao Gerente de Serviços de Dispositivo (componente *Device Service Manager*).

5. Gerente de Serviços de Dispositivo notifica ao componente controlador de Serviço de Orquestração Ativa (componente *Active Device Orchestration Service*).

6. Componente controlador de Serviço de Orquestração Ativa processa a mensagem e a envia com semântica adequada para os Dispositivos Secundários registrados na Classe Ativa.

7. As instâncias de *Secondary Device Subsystem* recebem e processam a mensagem de orquestração.

Sequência B – Envio de mensagens de orquestração de Dispositivo Secundário (referente à mudança de variável NCL) registrado em uma Classe Ativa para Dispositivo Pai:

1. *Secondary Device Subsystem* executa aplicação NCL recebida de subsistema do seu Dispositivo Pai, e altera uma variável NCL de nomenclatura específica da plataforma Ginga-MD.

2. *Secondary Device Subsystem* envia mensagem contendo informações da mudança de variável realizada para Dispositivo Pai.

3. Componente controlador de Serviço de Orquestração Ativa (componente *Active Orchestration Service* do *middleware* do Dispositivo Pai) processa a mensagem e notifica Gerente de Serviços de Dispositivo (componente *Device Service Manager*).

4. Gerente de Serviços de Dispositivo notifica Gerente de Integração de Dispositivos (componente *Device Integration Manager*).

5. Gerente de Integração de Dispositivos notifica *NCL Presentation System* via Gerente de Classes de Dispositivos (componente *Device Class Manager*).

6. Gerente de Classes de Dispositivos verifica notificação de mudança de variável.

7. Formatador NCL (*NCL Formatter*) é notificado pelo Gerente de Classes de Dispositivos acerca da mudança da variável e processa a ação de acordo (realiza ações resultantes da mudança da variável).

6.2.3.3. Classe Captura de Mídia

A captura e o envio de fluxos de mídia codificados pelos Dispositivos Secundários filhos registrados na Classe Captura de Mídia é caracterizado pelo diagrama de sequência apresentado na Figura 18. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.10 e 4.2.1.2.5.1.

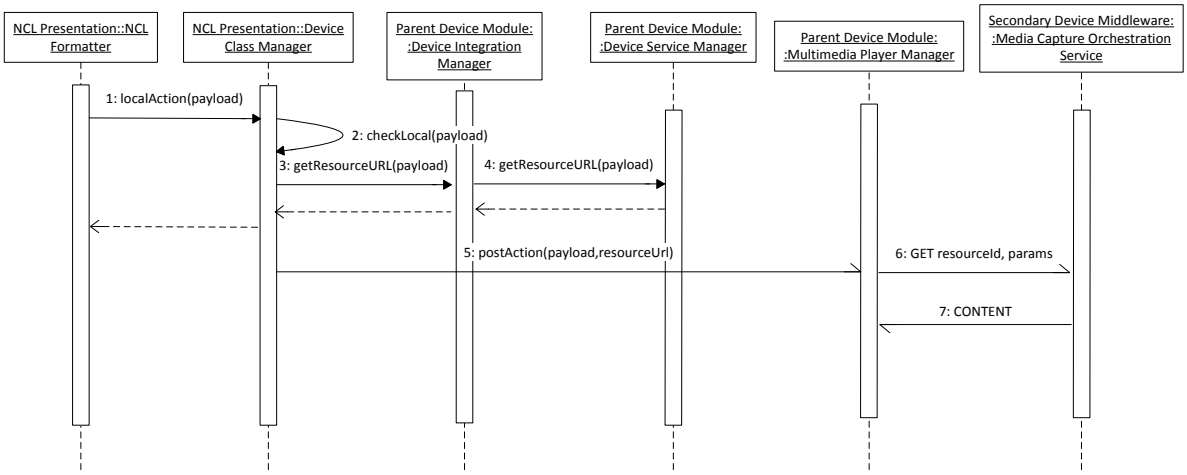


Figura 18 – Diagrama de sequência as atividades de orquestração de objetos associados à Classe Captura de Mídia

As etapas apresentadas pelo diagrama de sequência da Figura 18 são descritas a seguir:

1. Formatador NCL (*NCL Formatter*, parte do *NCL Presentation System*) no Dispositivo Pai realiza transição de ativação na máquina de estado de apresentação de objeto de mídia que identifica (através de uma URL com esquema específico) recurso de captura de mídia oferecido (e.g. captura de áudio) por uma instância particular de *Secondary Device Subsystem* que oferece o serviço *Media Capture Device Service*. Formatador NCL notifica Gerente de Classes de Dispositivos (componente *Device Class Manager*, do *NCL Presentation System*) da transição realizada.

2. Gerente de Classes de Dispositivos verifica e valida conteúdo da notificação de transição recebida do Formatador NCL.

3. Gerente de Classes de Dispositivos notifica transição a Gerente e Integração com Dispositivos (componente *Device Integration Manager*, parte do *middleware* do Dispositivo Pai).

4. Gerente de Integração de Dispositivos realiza solicitação de conversão de URL para o Gerente de Serviços de Dispositivos (componente *Device Service Manager*), para que um exibidor local (no Dispositivo Pai) trate a exibição do conteúdo capturado pelo Dispositivo

Secundário. URL convertida é enviada para o Gerente de Classes de Dispositivos do *NCL Presentation System*.

5. Gerente de Classes de Dispositivos envia a URL convertida para Gerente de Exibidores Multimídia (*Multimedia Player Manager*, parte do *middleware* do Dispositivo Pai).

6. Gerente de Exibidores Multimídia associa URL ao exibidor adequado, que realiza a solicitação por conteúdo para o serviço de captura de mídia oferecido pela instância de Dispositivo Secundário registrada na Classe Captura de Mídia.

7. *Secondary Device Subsystem* procede com o envio do conteúdo capturado para o Dispositivo Pai (através do serviço de captura de mídia – *Media Capture Device Service*, parte do *middleware* do Dispositivo Secundário).

6.2.3.4. Classe UPnP AV MediaServer ControlPoint

O envio e o recebimento de mensagens de orquestração do Dispositivo Pai para os seus Dispositivos Secundários filhos registrados na Classe *UPnP AV MediaServer ControlPoint* é caracterizado pelo diagrama de sequência apresentado na Figura 19. Os casos de uso que estabeleceram quais as etapas para esse processo são apresentados nas seções 4.2.1.1.7, 4.2.1.1.11 e 4.2.1.2.

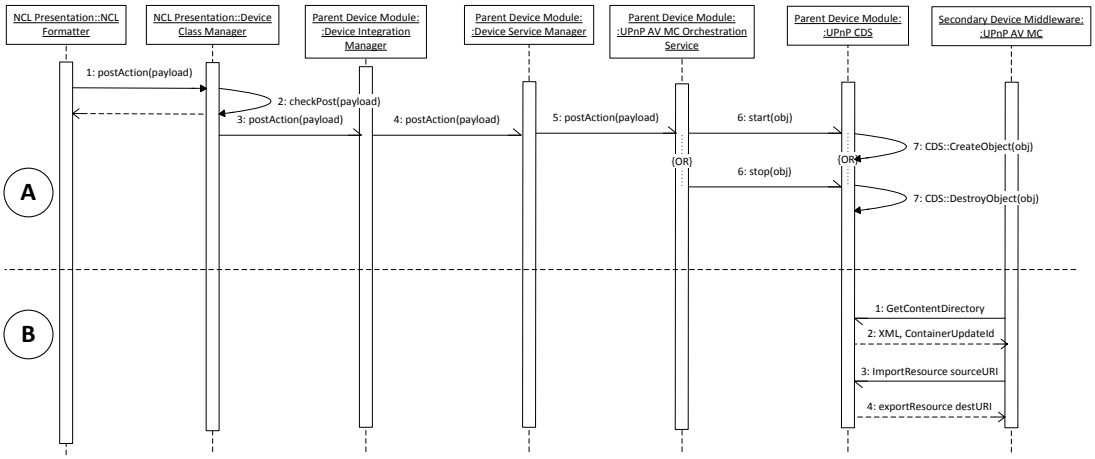


Figura 19 – Diagrama de sequência as atividades de orquestração de objetos associados à Classe *UPnP AV MediaServer ControlPoint*

As etapas apresentadas pelo diagrama de sequência da Figura 19 são descritas a seguir.

Sequência A – Envio de mensagens de orquestração do Dispositivo Pai para Dispositivos Secundários registrados na Classe *UPnP AV MediaServer ControlPoint*:

1. Formatador NCL (*NCL Formatter*) no Dispositivo Pai realiza transição na máquina de estados de um objeto de mídias associado à Classe *UPnP AV MediaServer ControlPoint* e notifica Gerente de Classes de Dispositivos (componente *Device Class Manager*, parte do *NCL Presentation System*).

2. Gerente de Classes de Dispositivos verifica e valida conteúdo da notificação de transição recebida do Formatador NCL.

3. Gerente de Classes de Dispositivos notifica transição a Gerente e Integração com Dispositivos (componente *Device Integration Manager*, parte do middleware do Dispositivo Pai).

4. Gerente de Integração de Dispositivos processa a mensagem e notifica ao Gerente de Serviços de Dispositivo (componente *Device Service Manager*).

5. Gerente de Serviços de Dispositivo notifica ao componente controlador de Serviço *UPnP AV MediaServer ControlPoint* (componente *UPnP AV MC Orchestration Service*).

6. Dependendo da transição efetuada (se a transição ativa ou desativa objeto de mídia), componente controlador de Serviço *UPnP AV MediaServer ControlPoint* notifica ao serviço UPnP *Content Directory* (Ritchie 2002; Kang 2005) (componente *UPnP CDS*).

7. Componente UPnP CDS (parte do Subsistema do Dispositivo Pai) realiza a ação de compartilhar ou finalizar o compartilhamento do objeto de mídia (como definido em (Ritchie 2002; Kang 2005)) com Dispositivos Secundários registrados na Classe *UPnP AV MediaServer ControlPoint*.

Sequência B – Solicitação por objeto de mídia partindo de Dispositivo Secundário registrado na Classe *UPnP AV MediaServer ControlPoint* para Dispositivo Pai:

1. Dispositivo Secundário solicita (através de seu serviço *UPnP AV MediaServer ControlPoint*) lista de objetos de mídia compartilhados ao Dispositivo Pai (mensagem *GetContentDirectory,* de acordo com definições da plataforma UPnP (Ritchie 2002; Kang 2005)).

2. Dispositivo Pai envia uma lista com os objetos compartilhados para o Dispositivo Secundário que realizou a solicitação (mensagem *ContainerUpdate*, de acordo com definições da plataforma UPnP (Kang 2005)).

3. Dispositivo Secundário, de posse da lista de objetos de mídia compartilhados, solicita um dos objetos ao Dispositivo Pai (mensagem *importResource*, de acordo com definições da plataforma UPnP (Ritchie 2002; Kang 2005)).

4. Dispositivo Pai envia objeto de mídia solicitado para Dispositivo Secundário (mensagem *exportResource*, de acordo com definições da plataforma UPnP (Ritchie 2002; Kang 2005)).

7
Testes Sistêmicos

Este Capítulo apresenta os procedimentos de teste necessários para a verificação do atendimento das funcionalidades da arquitetura de *software* apresentada no Capítulo 5 e reporta os resultados da realização de tais procedimentos sobre o protótipo da plataforma Ginga-MD apresentado no Capítulo 6.

Um conjunto de testes que verifica a aderência de uma implementação a uma especificação é de suma importância. A definição do conjunto de testes definido para a plataforma Ginga-MD baseou-se na metodologia definida para a criação da Suíte de Testes para a linguagem NCL (Araújo 2011). Testes de conformidade são testes do tipo Caixa Preta, e verificam funcionalidades e o comportamento observável de um sistema (Araújo 2011).

A Suíte de Testes da linguagem NCL (Araújo 2011) foi desenvolvida baseada nas técnicas de testes unitários (ou de unidade), testes de integração e testes de sistemas. A metodologia foi desenvolvida através de adaptações dessas técnicas considerando aspectos específicos da linguagem como interdependência entre atributos de um mesmo elemento, e a interdependência (hierárquica ou não) dos diversos elementos da linguagem relacionados (Araújo 2011).

Foram definidas aplicações para a plataforma Ginga-MD que materializam definições lógicas que contemplam sequências de testes integrados que validam os requisitos definidos para esta tese. Foram desenvolvidas três aplicações NCL, baseadas nos cenários de uso apresentados no Capítulo 2, que estão diretamente associados aos requisitos propostos para a plataforma. Fatores como escala e segurança, apesar de importantes, não foram considerados para a prova de conceito materializada pelo protótipo implementado (considerações acerca desses fatores são feitas na Seção 8.2 – "Trabalhos Futuros"). A Seção 7.1 apresenta as características e funcionalidades abrangidas por essas aplicações. Finalmente, a Seção 7.2 reporta a realização da execução de testes dessas aplicações NCL, utilizando o protótipo da plataforma Ginga-MD.

7.1. Aplicações de Teste

Serão apresentados a seguir os comportamentos de três aplicações NCL. A descrição do comportamento de cada aplicação contempla uma sequência de testes integrados, de forma que cada uma das aplicações valide a totalidade dos requisitos considerados para a plataforma Ginga-MD, a partir da verificação da realização de etapas e resultados associados diferentes.

7.1.1. Aplicação “Brazil 2014 Bis” (TV Digital interativa)

Durante a Copa do Mundo de 2010 na África do Sul, o Brasil apresentou algumas inovações tecnológicas tendo em vista o que pode ser esperado para as transmissões da Copa do Mundo de 2014. Uma dessas apresentações foi a aplicação interativa "Brazil 2014 Bis", sendo executada no Ginga-NCL com múltiplos dispositivos de apresentação. O vídeo principal da aplicação é composto por clipes de edições passadas da Copa do Mundo FIFA, e durante a exibição o uso de múltiplos dispositivos é extensivamente explorado. A aplicação foi desenvolvida para ter um receptor de TV Digital como Dispositivo Base, que orquestra dispositivos registrados em duas classes de dispositivos NCL (Classe Ativa Estéril e Classe *UPnP AV MediaServer ControlPoint*).

A aplicação inicia com um vídeo principal, que é composto por clipes de edições passadas da Copa do Mundo FIFA. Em um dado momento, na tela do Dispositivo Base, são apresentadas graficamente informações sobre os países que já foram campeões da Copa do Mundo. Neste instante, os Dispositivos Secundários que se parearam com o Dispositivo Base e foram registados na Classe Ativa Estéril passam a exibir uma aplicação que oferece uma opção que, se selecionada, fará com que a aplicação sendo exibida no Dispositivo Base seja finalizada e uma versão adaptada da aplicação seja exibida apenas nos Dispositivos Secundários.

Em um instante posterior, os Dispositivos Secundários da Classe Ativa passam a exibir outra aplicação, através da qual pode ser realizada a compra de pacotes turísticos para a Copa de 2014. No instante em que o vídeo principal passa a exibir uma cobrança de pênalti realizada na final de uma Copa do Mundo, os dois primeiros dispositivos que se registraram na Classe Ativa Estéril passam a exibir uma aplicação que simula uma disputa de pênaltis (os demais exibem uma mensagem com os créditos da aplicação). O primeiro usuário a se registrar atuará como o jogador de linha e o segundo usuário como o goleiro. Em um momento específico o Dispositivo

Base requisita que os usuários, através das telas dos seus Dispositivos Secundários registrados na Classe Ativa Estéril, escolham para que lado chutar ou pular, e o resultado da disputa (gol ou defesa) é apresentado na tela do Dispositivo Base.

Durante a exibição do vídeo principal, uma lista com vídeos dos melhores momentos das Copas do Mundo é compartilhada pelo Dispositivo Base para os Dispositivos Secundários registrados na Classe *UPnP AV MediaServer ControlPoint*.

A visão estrutural a seguir (Figura 20) refere-se a uma aplicação NCL que possui o comportamento apresentado acima. Os principais elos entre os objetos de mídia manipulados pela aplicação NCL apresentada na visão estrutural são discutidos a seguir.

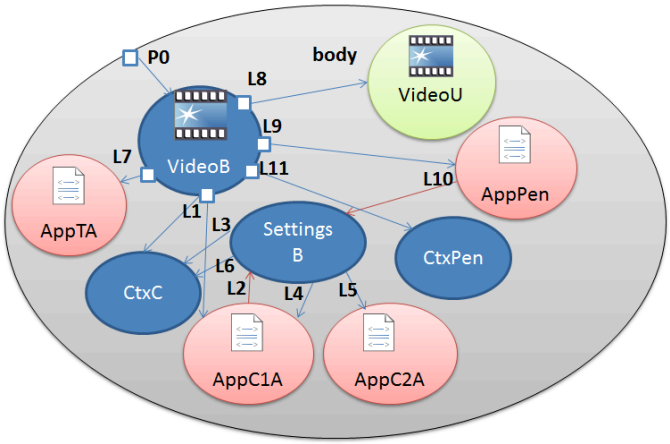


Figura 20 – Visão estrutural da aplicação "Brazil 14 Bis"

P0 – Aplicação inicia (definição de elemento <*port*>) com o vídeo principal (*VideoB*);

L1 – Elo condicionado a início de instante do vídeo (âncora em *VideoB*) inicia contexto que contém elementos gráficos (contendo informações relativas aos campeões da Copa do Mundo) que são apresentados na tela do Dispositivo Base (*CtxC*) e inicia também uma aplicação NCL associada à Classe Ativa Estéril (*AppC1A* – a aplicação que possui uma opção para exibição de conteúdo apenas nos Dispositivos Secundários).

L2 – Elo na aplicação *AppC1A* associada à Classe Ativa Estéril que define alteração de valor de variável (variável *parent.class(2).appToBase*), que refletirá no nó de definições do exibidor NCL no Dispositivo Base (*SettingsB*).

L3 – Elo que define que modificação de variável (variável do nó de definições *SettingsB parent.class(2).appToBase*) finaliza a exibição dos elementos gráficos na tela do Dispositivo Base (*CtxC*).

L4 – Elo que define que modificação de variável (variável do nó de definições *SettingsB parent.class(2).appToBase*) finaliza a exibição da aplicação associada à Classe Ativa Estéril (*App1CA*).

L5 – Elo que define que modificação de variável (variável do nó de definições *SettingsB parent.class(2).appToBase*) inicia a exibição da aplicação com as informações sobre os campeões associada à Classe Ativa Estéril (*App2CA*).

L6 – Elo que define que se a aplicação associada à Classe Ativa Estéril (*App2CA*) for abortada (i.e. não há mais Dispositivos Secundários registrados) os elementos gráficos (*CtxB*) devem ser exibidos na tela do Dispositivo Base.

L7 – Elo condicionado a início de instante do vídeo (âncora em *VideoB*) que inicia aplicação NCL associada à Classe Ativa Estéril (aplicação para compra de pacotes para Copa de 2014 – *AppTA*).

L8 – Elo condicionado a início de instante do vídeo (âncora em *VideoB*) que inicia vídeo (*VideoU*) associado à Classe UPnP AV *MediaServer ControlPoint* (fazendo que o mesmo passe a ser compartilhado com dispositivos registrados na classe associada).

L9 – Elo condicionado a início de instante do vídeo (âncora em *VideoB*) que inicia aplicação NCL (*AppPen*) associada à Classe Ativa Estéril (aplicação da simulação da disputa de pênalti – exibição condicionada ao valor da variável *child.index*).

L10 – Elo na aplicação *AppC1A* associada à Classe Ativa Estéril (disponível para Dispositivos Secundários cujo valor de *child.index* seja 1 ou 2) que define alteração de valor de variável (variáveis *parent.class(2).device(1).shoot* e *parent.class(2).device(2).jump*, alteradas por dispositivos distintos), que refletirá no nó de definições do exibidor NCL no Dispositivo Base (*SettingsB*).

L11 – Elo condicionado a início de instante do vídeo (âncora em *VideoB*) que inicia animação na tela do Dispositivo Base com o resultado da simulação de pênalti – contexto (*CtxPen*) com exibição do resultado condicional, realizada através do elemento <*switch*> (utilizando regras definidas a partir das variáveis do nó de definições *SettingsB: parent.class(2).device(1).shoot* e *parent.class(2).device(2).jump*).

7.1.2.
Aplicação "Luzia e a Vaca Andorinha" (Cinema Digital interativo)

"Luzia e a Vaca Andorinha" é um filme interativo dirigido por Eliezer Rolim (Souto 2008), cuja viabilização técnica foi realizada pelo Laboratório de Aplicações em Vídeo Digital (LAVID) da Universidade Federal da Paraíba (UFPB). O filme possui um roteiro não linear, no qual os espectadores são convidados a decidir os rumos da trama através dos seus dispositivos pessoais. Antes de momentos determinantes para o rumo da trama, uma enquete com diferentes versões para uma escolha do protagonista é apresentada aos espectadores. A partir da votação realizada, o filme procede na versão escolhida sem nenhuma interferência abrupta. Uma versão da aplicação foi desenvolvida para a plataforma Ginga-MD, na qual outras possibilidades de uso de múltiplos dispositivos são exploradas, como é descrito a seguir. A aplicação foi desenvolvida para ter um projetor de Cinema Digital como Dispositivo Base, que orquestra dispositivos registrados em três classes de dispositivos NCL (Classe Ativa Estéril, Classe Passiva e Classe Captura de Mídia).

A aplicação inicia com um vídeo principal, que exibe uma história onde os protagonistas realizam uma série de decisões que influenciam na composição da história. As decisões realizadas pelos protagonistas dependem da escolha dos membros da audiência, o que é realizado através de uma aplicação NCL exibida em Dispositivos Secundários registrados na Classe Ativa Estéril junto ao Dispositivo Base (que controla a exibição do filme interativo). A aplicação é exibida nos dispositivos nos momentos de decisão, e a sequência de vídeos exibidos pelo Dispositivo Base depende do resultado da votação em cada momento definido. Dispositivos Secundários registrados na Classe Passiva poderão visualizar um vídeo de baixa qualidade (e sem áudio) referente à opção não escolhida em cada momento de decisão. Durante a exibição da última cena do filme interativo, o Dispositivo Base reproduz um fluxo de áudio capturado pelo primeiro Dispositivo Secundário registrado na Classe Captura de Mídia.

A visão estrutural da Figura 21 apresenta uma estrutura simplificada para a aplicação NCL cujo comportamento é apresentado acima (elementos recorrentes das diferentes votações são omitidos, i.e. apenas uma escolha é apresentada na visão). Os principais elos entre os objetos de mídia manipulados pela aplicação NCL são discutidos a seguir.

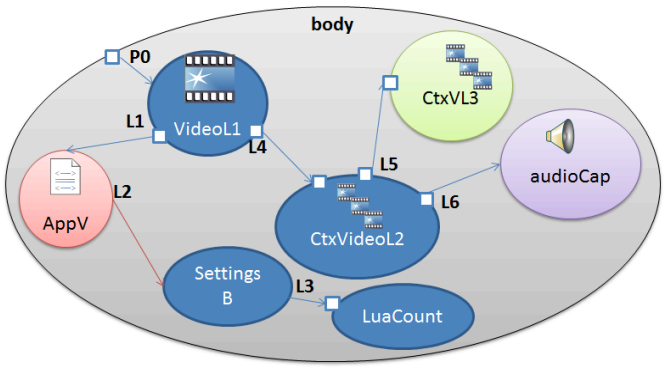


Figura 21 – Visão estrutural da aplicação "Luzia e a Vaca Andorinha"

P0 – Aplicação inicia (definição de elemento <*port*>) com a exibição de um vídeo, no Dispositivo Base, contendo as primeiras cenas da trama (*VideoL1*).

L1 – Elo condicionado a início de instante do vídeo (âncora em *VideoL1*) inicia uma aplicação NCL associada à Classe Ativa Estéril (*AppV*).

L2 – Elo na aplicação *AppV* associada à Classe Ativa Estéril que define alteração de valor de variável (variável *parent.class(2).voto1* ou variável *parent.class(2).voto2* dependendo da escolha do usuário), que refletirá no nó de definições do exibidor NCL no Dispositivo Base (*SettingsB*).

L3 – Elo que define que modificação de variáveis relativas à votação (*parent.class(2).voto1* ou variável *parent.class(2).voto2,* nó de definições *SettingsB*) ativa *script* Lua que realiza o cálculo da quantidade de votos para cada opção.

L4 – Elo condicionado a início de instante do vídeo (âncora em *VideoL1*) inicia contexto que apresenta vídeo correspondente à opção mais votada (*CtxVideoL2* onde as opções são condicionalmente exibidas através de um elemento <*switch*> da linguagem NCL).

L5 – Elo condicionado a início do vídeo escolhido (*CtxVideoL2*), que envia fluxo com renderização de baixa qualidade do vídeo não escolhido (*CtxVL3*) para Dispositivos Secundários registrados na Classe Passiva.

L6 – Elo condicionado a início de instante (âncora) do vídeo escolhido (*CtxVideoL2*) onde é iniciada a execução de um objeto de mídia (*audioCap*) no Dispositivo Base que se refere a um fluxo de áudio capturado pelo primeiro Dispositivo Secundário registrado na Classe Captura de Mídia (URL: ncl-device://*parent.class(3).device(1)*/*AudioCapture*).

7.1.3. Aplicação “Aula sobre NCL”

A aplicação “Aula sobre NCL” foi desenvolvida para uma palestra ou aula onde há uma apresentação de eslaides multimídia interativa sobre a linguagem NCL. O computador acoplado ao projetor apresenta os elementos da apresentação cujo andamento é controlado pelo professor palestrante através de uma aplicação executada em um *tablet* pareado com o computador. Os espectadores podem utilizar seus dispositivos pessoais para receber do *tablet* do apresentador questionários interativos. Na tela do dispositivo do apresentador há um indicador de quantos dos espectadores já responderam seus questionários. A aplicação realiza o compartilhamento de um vídeo (através de serviços UPnP) relacionado ao tema quando da projeção dos seus últimos eslaides multimídia.

Para realizar tal aplicação em NCL de forma que fosse executada na plataforma Ginga-MD, foram utilizadas as seguintes classes de dispositivos NCL: Classe Ativa Fértil e Classe UPnP *AV MediaServer ControlPoint* (referenciada na aplicação NCL a ser executada no Dispositivo Base) e Classe Ativa Estéril (referenciada na aplicação NCL a ser executada no Dispositivo Secundário registrado na Classe Ativa Fértil). A visão estrutural na Figura 22 apresenta uma estrutura simplificada para a lógica de exibição necessária para viabilizar a aplicação NCL no Dispositivo Base e a visão estrutural apresentada na Figura 23 refere-se à aplicação executada no *tablet* do apresentador (Dispositivo Secundário registrado na Classe Ativa Fértil).

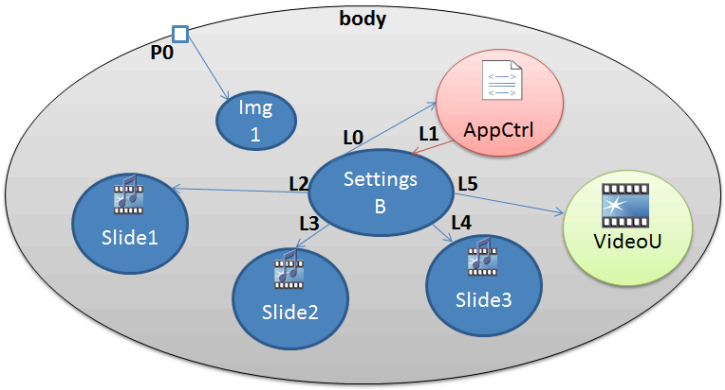


Figura 22 – Visão estrutural para parte da aplicação "Aula sobre NCL" (parte executada pelo Dispositivo Base)

P0 – Aplicação inicia (definição de elemento <*port*>) com a exibição de uma imagem, no Dispositivo Base, com uma mensagem solicitando pareamento do dispositivo do professor (*Img1*).

L0 – Elo que define que modificação de variável relativas à quantidade de dispositivos na Classe Ativa Fértil (variável *system.devNumber(2)* (Costa 2009; Soares 2009), no nó de definições *SettingsB*) e inicia aplicação associada a esta classe de dispositivos (*App1*).

L1 – Elo definido na aplicação NCL *App1* (associada à Classe Ativa Fértil) que define alteração de valor de variável (variável *parent.class(2). device(1).currentSlide*), que refletirá no nó de definições do exibidor NCL no Dispositivo Base (*SettingsB*).

L2 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "1" no nó de definições (*SettingsB*) iniciará a exibição do contexto com os elementos relacionados ao primeiro eslaide multimídia (*Slide1).*

L3 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "2" no nó de definições (*SettingsB*) iniciará a exibição do contexto com os elementos relacionados ao segundo eslaide multimídia (*Slide2).*

L4 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "3" no nó de definições (*SettingsB*) iniciará a exibição do contexto com os elementos relacionados ao terceiro eslaide multimídia (*Slide3).*

L5 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "3" no nó de definições (*SettingsB*) iniciará objeto de mídia (*VideoU*) associado à Classe UPnP AV *MediaServer ControlPoint*.

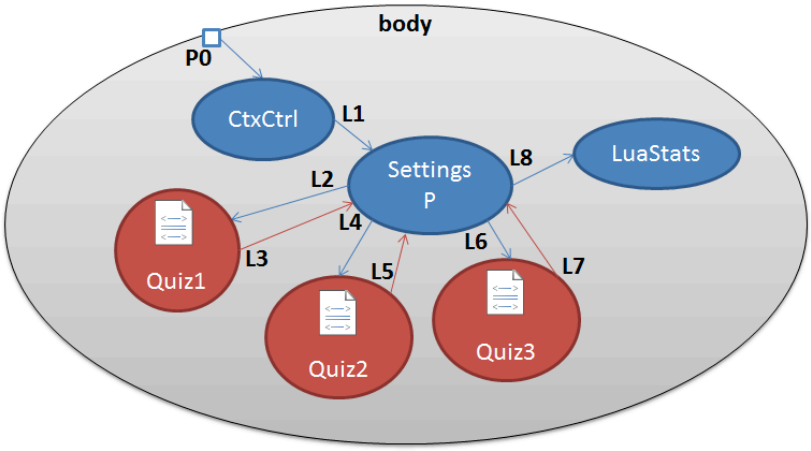


Figura 23 – Visão estrutural para parte da aplicação "Aula sobre NCL" (parte executada pelo Dispositivo Secundário registrado na Classe Ativa Fértil)

P0 – Aplicação inicia (definição de elemento <*port*>) com a exibição de elementos gráficos (botões) utilizados para controle do andamento da apresentação de eslaides no Dispositivo Base e que compõem o contexto *CtxCtrl*.

L1 – Elo que define alteração de valor de variável (variável *parent.class(2). device(1).currentSlide*) do nó de definições do Dispositivo Secundário pai (*SettingsP*). Essa modificação será refletida pelo nó de definições do exibidor NCL no Dispositivo Base.

L2 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "1" iniciará a exibição da aplicação NCL *Quiz1* associada à Classe Ativa Estéril.

L3 – Elo definido na aplicação NCL *Quiz1* (associada à Classe Ativa Estéril) que define alteração de valor de variável (variável *parent.class(4).device(i).answerQuiz1,* onde i é o índice do Dispositivo Secundário registrado na Classe Ativa Estéril) que refletirá no nó de definições do exibidor NCL no Dispositivo Secundário Pai (*SettingsP*).

L4 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "2" iniciará a exibição da aplicação NCL *Quiz2* associada a Classe Ativa Estéril

L5 – Elo definido na aplicação NCL *Quiz1* (associada à Classe Ativa Estéril) que define alteração de valor de variável (variável *parent.class(4).device(i).answerQuiz2,* onde i é o índice do Dispositivo Secundário registrado na Classe Ativa Estéril) que refletirá no nó de definições do exibidor NCL no Dispositivo Secundário Pai (*SettingsP*).

L6 – Elo que define que modificação de variável (*parent.class(2).device(1).currentSlide*) para valor "3" iniciará a exibição da aplicação NCL *Quiz3* associada a Classe Ativa Estéril

L7 – Elo definido na aplicação NCL *Quiz3* (associada à Classe Ativa Estéril) que define alteração de valor de variável (variável *parent.class(4).device(i).answerQuiz3,* onde i é o índice do Dispositivo Secundário registrado na Classe Ativa Estéril) que refletirá no nó de definições do exibidor NCL no Dispositivo Secundário Pai (*SettingsP*).

L8 – Elo que representa conjunto de elos que definem que modificação de variáveis realizadas em L3, L5 e L7 (no nó de definições *SettingsP*) sejam computadas e apresentadas ao usuário do Dispositivo Secundário Pai através de um script Lua (*LuaStats*).

7.2. Testes do protótipo

Vários testes foram realizados com as aplicações descritas na Seção 7.1. Para a execução dos testes, foram utilizadas as instalações do Laboratório TeleMídia da PUC-Rio e também do Laboratório de Aplicações de Vídeo Digital da UFPB (Figura 24). A Seção 7.2.1 descreve um conjunto representativo de especificações de configurações de *hardware* e *software* utilizado para os testes das aplicações. A Seção 7.2.2 finaliza com considerações acerca da realização dos testes e da avaliação dos resultados tendo em vista o atendimento dos requisitos estabelecidos por esta tese.

Figura 24 – Ambiente de Testes no LAVID/UFPB

7.2.1. Visão de Implantação e de Implementação

Esta seção descreve especificações de configurações de *hardware*, *software* e rede diferentes, nas quais a plataforma Ginga-MD foi implantada e executada (Visão de Implantação).

Cada especificação contemplará quais os nós físicos (dispositivos que executarão os subsistemas) e suas interconexões. Uma representação para os executáveis e subsistemas envolvidos (a Visão de Implementação) também é contemplada nesta seção. As especificações de configurações elaboradas tiveram como base os cenários descritos no Capítulo 2, e serão utilizadas para condução dos testes sistêmicos da plataforma proposta nesta tese.

7.2.1.1. Aplicação "Brazil 14 Bis" – TV Digital interativa

O diagrama de implantação UML (Figura 25) apresentado abaixo remete ao primeiro cenário apresentado no Capítulo 2, onde uma aplicação NCL distribuída é executada por espectadores de um programa de TV Digital interativa. A aplicação "Brazil 14 Bis" (Figura 26) foi executada várias vezes utilizando esta plataforma de testes e também outras com configurações equivalentes.

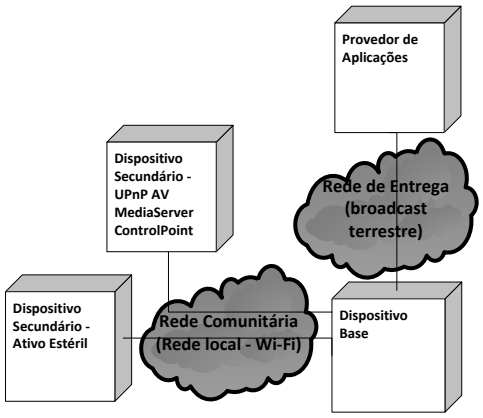


Figura 25 – Diagrama de implantação UML para cenário de TV Digital interativa

A configuração e a cardinalidade dos elementos apresentados no diagrama de implantação para o cenário de TV Digital interativa (Figura 25) são apresentados resumidamente na Tabela 8.

Nó Físico	Especificação da Configuração	Quantidade
Provedor de Aplicações	EiTV Playout[16] (Retaguarda de transmissão ISDB-Tb)	1

[16] http://www.eitv.com.br/playout_en.php

Dispositivo Base	MiniPC AOne (Core 2 Duo, 2Gb RAM, Linux), *Base Device Subsystem.*	1
Dispositivo Secundário – Classe Ativa Estéril	Smartphone Motorola Atrix (Android), *Secondary Device Subsystem* (Ginga-MD SADS).	1
Dispositivo Secundário – Classe Ativa Estéril	Smartphone HTC Legend (Android), *Secondary Device Subsystem* (Ginga-MD SADS).	1
Dispositivo Secundário – Classe Ativa Estéril	Notebook Sony Vaio S series Windows (Core i7, 8Gb RAM, Windows), *Secondary Device Subsystem* (Ginga-MD SADS).	1
Dispositivo Secundário – Classe Ativa Estéril	Smartphone Motorola Milestone (Android), *Secondary Device Subsystem* (Ginga-MD SADS).	1
Dispositivo Secundário – Classe *UPnP AV MediaServer* ControlPoint	Notebook MacBook (Core 2 Duo, 2Gb RAM, Mac OS X) – Player VLC instalado	1

Tabela 8 – Configuração dos nós físicos - Cenário de TV Digital interativa

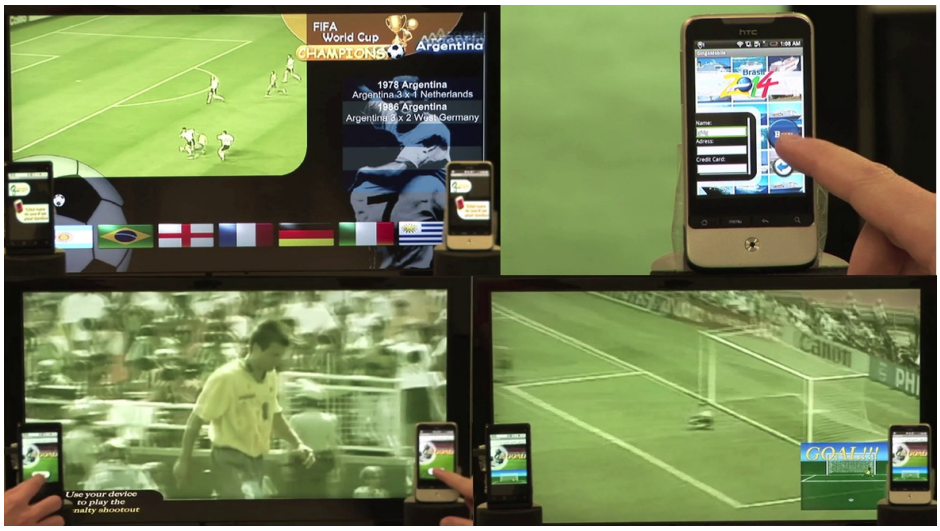


Figura 26 – Testes no Laboratório TeleMídia da PUC-Rio (Aplicação "Brazil 2014 Bis")

7.2.1.2.Aplicação "Luzia e a Vaca Andorinha" – Cinema Digital interativo

O diagrama de implantação UML (Figura 27) apresentado abaixo se refere ao segundo cenário apresentado no Capítulo 2, onde uma aplicação NCL distribuída é executada durante a apresentação de um filme interativo em uma sala de Cinema Digital. A aplicação "Luzia e a Vaca Andorinha" (simulada na Figura 28) foi executada várias vezes utilizando esta plataforma de testes e também outras com configurações equivalentes.

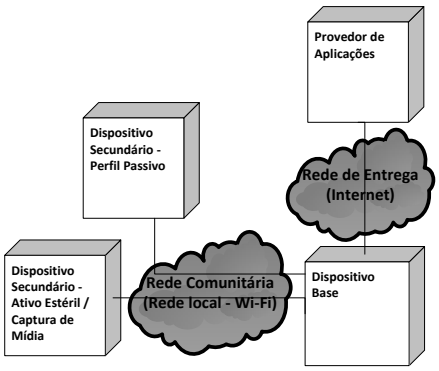


Figura 27 – Diagrama de implantação UML para o cenário de Cinema Digital interativo

A configuração e a cardinalidade dos elementos apresentados no diagrama de implantação para o cenário de Cinema Digital interativo são apresentados resumidamente na Tabela 9.

Nó Físico	Especificação da Configuração	Quantidade
Provedor de Aplicações	Notebook Sony Vaio S series Windows (Core i7, 8Gb RAM, Linux) – GingaSpace (Guedes 2012)	1
Dispositivo Base	MiniPC AOne (Core 2 Duo, 2Gb RAM, Linux), Base Device Subsystem	1
Dispositivo Secundário – Classe Ativa Estéril e Classe Captura de Mídia	Smartphone Motorola Atrix (Android), Secondary Device Subsystem (Ginga-MD SADS e Ginga-MD MCDS)	1
Dispositivo Secundário – Classe Ativa Estéril e Classe Captura de Mídia	Smartphone HTC Legend (Android), Secondary Device Subsystem (Ginga-MD SADS e Ginga-MD MCDS)	1
Dispositivo Secundário – Classe Passiva	Notebook Sony Vaio (Windows), Secondary Device Subsystem (Ginga-MD PDS)	1
Dispositivo Secundário – Classe Passiva	Notebook Sony Vaio (Linux), Secondary Device Subsystem (Ginga-MD PDS)	1

Tabela 9 – Configuração dos nós físicos - Cenário de Cinema Digital interativo

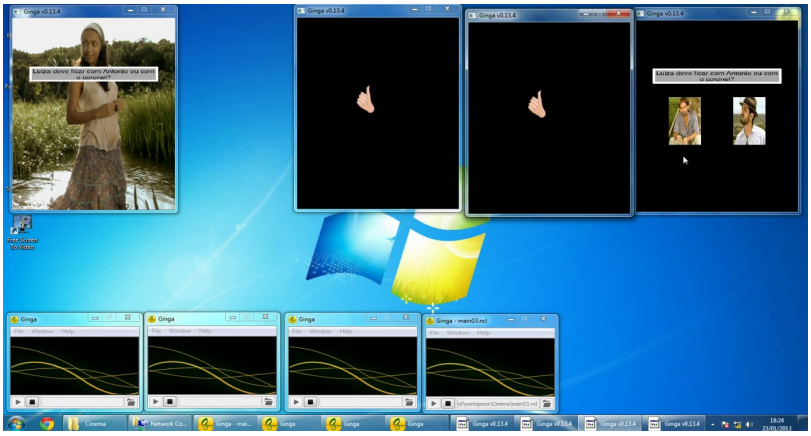


Figura 28 – Execução da aplicação "Luzia e a Vaca Andorinha" em ambiente simulado (Ginga for Windows [17]) - Janela do Dispositivo Base apresenta o vídeo principal enquanto as janelas dos Dispositivos Secundários apresentam aplicação NCL para votação nas opções para o desenrolar da trama.

7.2.1.3. Aplicação “Aula sobre NCL” – Apresentação de eslaides multimídia interativa

O diagrama de implantação UML (Figura 29) apresentado abaixo se refere ao terceiro cenário apresentado no Capítulo 2, onde uma aplicação NCL distribuída é executada durante a uma apresentação de eslaides multimídia interativa. A aplicação “Aula sobre NCL” (Figura 30) foi executada várias vezes utilizando esta plataforma de testes e também outras com configurações equivalentes.

[17] http://www.ginga.org.br

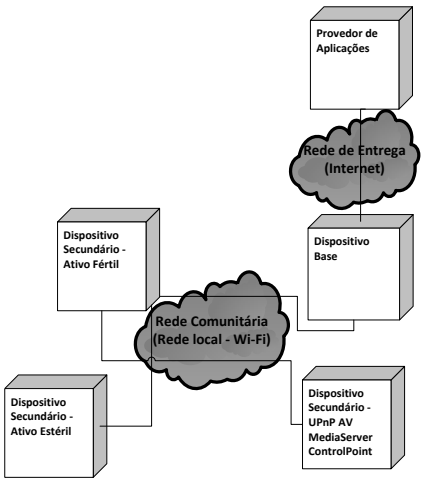


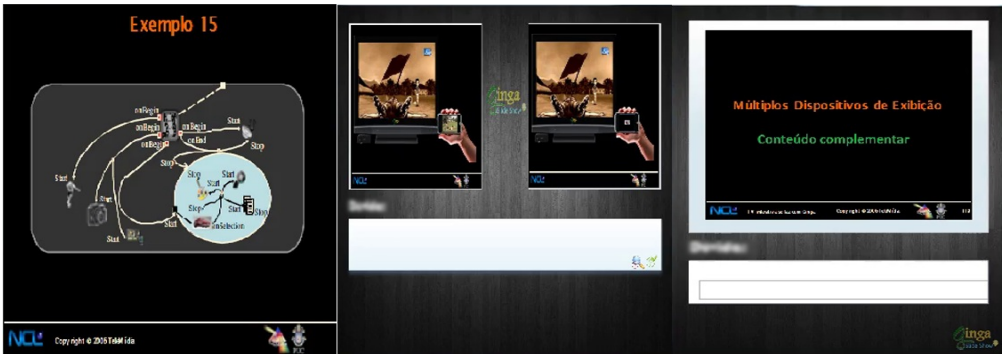
Figura 29 – Diagrama de implantação UML para cenário de Apresentação de eslaides multimídia interativa

A configuração e a cardinalidade dos elementos apresentados no diagrama de implantação para o cenário de Apresentação de eslaides multimídia interativa são apresentados na Tabela 10.

Nó Físico	Especificação da Configuração	Quantidade
Provedor de Aplicações	Notebook Sony Vaio S series Windows (Core i7, 8Gb RAM, Linux)	1
Dispositivo Base	MiniPC AOne (Core 2 Duo, 2Gb RAM, Linux), Base Device Subsystem	1
Dispositivo Secundário – Classe Ativa Fértil	Notebook Sony Vaio (Linux), Secondary Device Subsystem (Ginga-MD FADS).	1
Dispositivo Secundário – Classe Ativa Estéril	Tablet AOC Breeze 2 (Android), Secondary Device Subsystem (Ginga-MD SADS).	1
Dispositivo Secundário – Classe Ativa Estéril	Smartphone Motorola DEFY (Android), Secondary Device	1

	Subsystem (Ginga-MD SADS)	
Dispositivo Secundário – Classe Ativa Estéril	Smartphone Motorola Milestone (Android), Secondary Device Subsystem (Ginga-MD SADS)	1
Dispositivo Secundário – Classe UPnP AV MediaServer ControlPoint	Smartphone Motorola Atrix (Android) – aplicação Andromote[18] instalada	1
Dispositivo Secundário – Classe UPnP AV MediaServer ControlPoint	Notebook MacBook (Core 2 Duo, 2Gb RAM, Mac OS X) – Player VLC instalado	1

Tabela 10 – Configuração dos nós físicos - Cenário de Apresentação de eslaides multimídia interativa



(a) (b) (c)

Figura 30 – Telas para aplicação "Aula sobre NCL" (a – Tela do Dispositivo Base; b – Tela Dispositivo Secundário Pai; c – Tela dos Dispositivos Secundários filhos)

7.2.2. Realização dos testes

Dezenas rodadas de execução de testes das aplicações foram realizadas, e nestas foram registradas toda a comunicação entre os dispositivos (registro automático das mensagens de

[18] http://www.andromote.de/

orquestração) e também o comportamento observável (por um observador humano) dos subsistemas, em avaliação de conformidade similar a definida em (Araújo 2011). Os resultados coletados foram utilizados para verificar a realização de todas as ações de orquestração previstas nas especificações das aplicações apresentadas na Seção 7.1.

Os testes foram realizados entre Junho e Agosto de 2012 no LAVID/UFPB e entre Setembro e Novembro de 2012 no TeleMídia/PUC-Rio utilizando infraestrutura como descrita na Seção 7.2.1. O sincronismo entre mídias distribuídas, a interação multimodal distribuída e a comunicação entre grupos de dispositivos pode ser verificada meramente pela avaliação dos registros de comunicação. De acordo com a especificação da aplicação testada e a manipulação orientada dos subsistemas por múltiplos usuários, foram verificados que os instantes do envio e do processamento de mensagens de orquestração correspondiam à sequência esperada em todos os casos.

Figura 31 – Testes com usuários no LAVID/UFPB

Os registros e o comportamento observável dos subsistemas envolvidos puderam comprovar a integração dinâmica de dispositivos e a ativação de mecanismos de tolerância à falha quando necessários (restauração do sincronismo). A integração dinâmica da plataforma mostrou-se efetiva em todos os casos.

A integração dinâmica e o acoplamento entre os subsistemas também foram aspectos avaliados em pesquisas realizadas em paralelo aos testes da plataforma Ginga-MD. A execução dos testes do protótipo da plataforma Ginga-MD serviu de base para um estudo na área de usabilidade (Guimarães 2012), no qual usuários de diferentes perfis utilizaram aplicações multi-dispositivo através da plataforma Ginga-MD.

Os testes com usuários (Figura 31 (Guimarães 2012)) contaram com a presença de um avaliador e um observador com objetivo de registrar e avaliar o desempenho dos usuários durante a execução das aplicações. Antes da execução dos testes, o avaliador explicava aos usuários o contexto da aplicação, dando um mínimo de informações sobre como o usuário deveria utilizar os dispositivos (muitos dos usuários não possuíam prática na manipulação de dispositivos móveis (Guimarães 2012)).

A avaliação de usabilidade levou em consideração o comportamento dos usuários durante a execução das aplicações e também o relato de uso dos mesmos, na forma do preenchimento de formulários que qualificavam diferentes aspectos da experiência. Questionários qualitativos e quantitativos foram aplicados aos usuários, onde aspectos como a facilidade de utilizar os subsistemas, elementos da interface das aplicações, entendimento de acoplamento entre os subsistemas e efetividade da integração dinâmica dos dispositivos foram avaliados.

O resultado (Guimarães 2012) demonstra que a maioria dos usuários registrou impressões positivas sobre o uso de dispositivos secundários, entendendo corretamente o acoplamento entre os subsistemas e a forma que é realizada a integração dinâmica de dispositivos na plataforma.

Além disso, a plataforma Ginga-MD foi utilizada para dar suporte ao desenvolvimento de uma nova abordagem para repositório de aplicações interativas, focado em distribuir aplicações multi-dispositivo (a pesquisa e desenvolvimento deste repositório foram materializados em uma dissertação de mestrado (Guedes 2012)).

8
Conclusão e Trabalhos Futuros

Esta tese apresentou aspectos relacionados à execução de aplicações hipermídia distribuídas. A definição do problema tratado neste trabalhou partiu da elaboração de cenários de uso, dos quais foram extraídos requisitos não atendidos em conjunto por nenhuma outra abordagem investigada.

Para atender os requisitos, foi modelada uma plataforma que desse suporte à execução de aplicações hipermídia distribuídas, baseada na linguagem NCL, incorporando funcionalidades relevantes e inéditas. Um protótipo da arquitetura de *software* foi implementado e, a partir de metodologia descrita, testado e avaliado. Os resultados dos testes comprovaram que os requisitos estabelecidos para a realização da solução proposta por esta tese foram atendidos.

A tese conclui-se com um resumo das contribuições realizadas e dos possíveis trabalhos futuros.

8.1.
Contribuições da Tese

As seguintes contribuições desta tese podem ser salientadas:

- Especificação de extensões para a linguagem NCL, agregando novas funcionalidades à linguagem: diferentes mecanismos para extensão e associação de classes de dispositivos (usando a ontologia mencionada no item anterior), variáveis para o tratamento individualizado de dispositivos, novas possibilidades para captura e transmissão de mídia entre dispositivos. As extensões propostas utilizam um modelo semântico de ontologia do domínio dos elementos envolvidos na plataforma para execução de aplicações hipermídia distribuídas. Tal modelo estabelece uma semântica comum para a definição dos requisitos necessários para os dispositivos envolvidos em uma apresentação hipermídia distribuída. A semântica de descrição permite que a plataforma orquestre recursos de dispositivos heterogêneos, definindo quais os serviços e mecanismos de comunicação utilizados

pela aplicação hipermídia (documentos NCL com anotações semânticas dos requisitos alvo).

- Especificação dos componentes de *software* de uma plataforma interoperável para execução de aplicações hipermídia distribuídas escritas em NCL – arquitetura de *software* de referência da plataforma Ginga-MD. Implementação de protótipos dos subsistemas da plataforma Ginga-MD de acordo com a arquitetura de *software* especificada, incorporando mecanismos de tolerância a falhas e recuperação durante a execução de aplicações hipermídia distribuídas.
- Validação do modelo de comunicação e diretivas de orquestração hierárquica da linguagem NCL, utilizada para descrever aplicações distribuídas, executadas por dispositivos heterogêneos.

8.2. Trabalhos Futuros

Os trabalhos realizados nesta tese abrem várias possibilidades para o desenvolvimento de outros trabalhos.

- Modelo de Segurança – há espaço para investigação acerca de mecanismos que permitam que os dispositivos possam utilizar canais seguros de comunicação, viabilizando a manipulação de objetos de mídia sensíveis ou de uso restrito (e.g. que utilizam alguma técnica de DRM – *Digital Rights Management* (Abbadi 2006)).
- Requisitos de rede mais complexos – esta tese considerou cenários de uso onde dispositivos eram orquestrados em redes locais, utilizando quantidades moderadas de dispositivos (dezenas) sem verificar questões de escalabilidade. Cabe, então, a investigação acerca de cenários de uso que extrapolem esta limitação, levando em consideração diferentes infraestruturas de comunicação (i.e. devem ser considerados aspectos de manutenção de QoS – Qualidade de Serviço) e a modelagem de uma arquitetura de *software* utilizando elementos que possam garantir requisitos de escalabilidade mais robustos (Muhammad 2012).
- Evolução da linguagem NCL – é de suma importância a investigação acerca de melhores elementos e "açúcares sintáticos" para a implementação de aplicações

hipermídia multi-dispositivo em NCL, principalmente no que diz respeito ao tratamento dos mecanismos de interação (de entrada) dos dispositivos.

- Integração do protótipo com outras plataformas – é desejável a investigação acerca da possibilidade de interoperabilidade de outras plataformas, tais como OSGi e Bluetooth.
- Ferramentas de autoria – há a necessidade de um melhor suporte por parte das ferramentas de autoria para a especificação de aplicações multi-dispositivo. Tais ferramentas de autoria deveriam possuir funcionalidades intuitivas que permitam a fácil manipulação dos elementos da linguagem NCL para construção de aplicações multi-dispositivo.
- Metadados e ontologia – para escalar o uso de classes de dispositivos (e assim adequar o uso da linguagem NCL às mais diversas situações multi-dispositivo), é necessário um robusto suporte semântico, como discutido nesta tese. A investigação por melhorias na ontologia utilizada nesta tese é importante para permitir que mais tipos de dispositivos possam ser integrados à plataforma.
- Suíte de Testes de Conformidade – este trabalho definiu uma metodologia de testes e avaliação a partir de três aplicações que agregavam funcionalidades representativas para os testes sistêmicos da plataforma. Porém, para facilitar a verificação de conformidade de diferentes implementações dos subsistemas propostos, é importante também o estabelecimento de uma Suíte de Testes de Conformidade a partir de testes minimalistas.

9
Referências

ABBADI, I. M. **Digital asset protection in personal private networks**. In 8th International Symposium on Systems and Information Security (SSI 2006), Sao Jose dos Campos, Sao Paulo, Brazil, 2006.

ABNT – Associação Brasileira de Normas e Técnicas. **Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações.** ABNT, NBR 15606-2, 2007.

AL-MUHTADI, J. **Moheet: A Distributed Middleware for Enabling Smart Spaces**. In First National IT Symposium (NITS) Riyadh, Arábia Saudita. 2006.

ARAÚJO E. C. et al. **Suíte de teste de conformidade para o Ginga-NCL**, Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia, pp. 9-17, Florianópolis, Brasil, 2011.

ARIB – Association of Radio Industries and Businesses. ***Application Execution Engine Platform for Digital Broadcasting***, ARIB STD-B23 Version 1.2, 2004.

ATSC - Advanced Television Systems Committee. **ATSC Standard: Advanced Common Application Platform (ACAP)**. Padrão A/101. 2005.

BATISTA, C.E.C.F, et al. **Estendendo o uso das classes de dispositivos Ginga-NCL**. Proceedings of the Brazilian Symposium on Multimedia and the Web – WebMedia. Belo Horizonte, Brasil, 2010.

BATISTA, C. E. C. F. et al. **Recuperação e Tolerância a Falhas em Apresentações de Aplicações Hipermídia Distribuídas**. Proceedings of the Brazilian Symposium on Multimedia and the Web – WebMedia. Florianópolis, Brasil, 2011

BRANDÃO R. R. de M., et al. **Extended features for the Ginga-NCL environment - Introducing the LuaTV API**. Proceedings of the 19th International Conference on Computer Communication Networks (2nd Workshop on Multimedia Computing and Communications). Zurique, Suíça, 2010.

BULTERMAN, D.C.A.; JANSEN, J.; KLEANTHOUS, K.; BLOM, K.; BENDEN, D. **AMBULANT: A Fast, Multi-Platform Open Source SMIL layer**. ACM International Conference on Multimedia. Nova York, EUA. 2004.

CAIRES, C. **Da narrativa fílmica interactiva Carrossel e Transparências: dois projectos experimentais**. In: Estéticas do digital: cinema e tecnologia, Manuela Penafria e Índia Mara Martins (org), LabCom, p. 33-48, 2007.

CESAR, P., BULTERMAN, D.C.A., OBRENOVIC, Z., DUCRET, J., CRUZLARA, S. **An Architecture for Non-Intrusive User Interfaces for Interactive Digital Television Experiences**. European Interactive TV Conference. Amsterdam, Holanda. 2007.

CLEMENTS, P. **Documenting software architectures: views and beyond**. In Software Engineering, 2003. Proceedings. 25th International Conference on (pp. 740-741). IEEE.

CONCOLATO, C., DUFORD, J. C., et al. **Communicating and migratable interactive multimedia documents**. Multimedia Tools and Applications, 1-24. 2011

CORTEZ, J., SHAMMA, D. A., CAI, L. **Device Communication: A Multi-Modal Communication Platform for Internet Connected Televisions**. Proceedings of the 10th European conference on Interactive TV and video, pp. 19-26. ACM, 2012

COSTA, R.M.R; MORENO, M.F.; RODRIGUES, R.F.; Soares L.F.G. **Live Editing of Hypermedia Documents.** ACM Symposium on Document Engineering. Amsterdam, Holanda. 2006.

COSTA R.M.R, MORENO M.F., SOARES L.F.G. **Intermedia Synchronization Management in DTV Systems.** ACM Symposium on Document Engineering. São Paulo, Brasil. 2008.

COSTA, R.M.R, MORENO, M.F., SOARES, L.F.G. **Ginga-NCL: Suporte a Múltiplos Dispositivos.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Natal, Brasil. 2009.

DIX, A., SAS, C. **Public displays and private devices: A design space analysis**. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 2008). 2008.

DAHER, G., et al. **Ginga-NCL em Dispositivos Portáteis: Uma Implementação para a Plataforma Android**. Proceedings of the Brazilian Symposium on Multimedia and the Web – WebMedia. Belo Horizonte, Brasil, 2010.

EDEN, A. H. et al. **Abstraction Classes in Software Design**. IEEE Software 153:4, pp.163-182. Agosto, 2006.

ETSI – European Telecommunications Standards Institute. **Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1.** Especificação Técnica ETSI TS 102 B12. 2005.

FORNO, F., et al.. **HoNeY: a MHP-based Platform for HOme NEtwork interoperability. In Advanced Information Networking and Applications**, 2006. AINA 2006. 20th International Conference on (Vol. 2, pp. 102-110). IEEE. 2006.

GASPAR, B. et al. **A New Approach for Slideshow Presentation at Working Meetings**. 6th Conference on Telecommunications, Conftele 2007. Peniche, Portugal, 2007.

GLASBERG, M. S., CERQUEIRA, R. **ActivePresentation: Uma Infra-estrutura de Software para Controle de Apresentações em Espaços Ativos**. Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia 2004 (pp. 28-35). 2004.

GUEDES, A. L. V. **Um repositório de aplicações sensível ao contexto de execução multidispositivo: Um Estudo de caso em BroadBand TV com o GingaSpace**. Dissertação de mestrado do Programa de Pós-Graduação em Informática da UFPB, 2012.

GUIMARÃES, A. P. N. **Experiência de uso de dispositivos convergentes na TV Digital Brasileira: um estudo de caso baseado no Ginga**. Monografia de Conclusão de Curso de Graduação em Ciência da Computação da UFPB, 2012.

HELAL S. **Standards for Service Discovery and Delivery**. IEEE Pervasive Computing archive. Volume 1, Issue 3, pp 95-100, 2002.

HOLMES M. E., et al. **Visual attention to television programs with a second-screen application**. In Proceedings of the Symposium on Eye Tracking Research and Applications (pp. 397-400). ACM. 2012.

HU, J. et al. **IPML: Structuring Distributed Multimedia Presentations in Ambient Intelligent Environments**. International Journal of Cognitive Informatics & Natural Intelligence, vol. 3, no. 2, pp. 37-60, 2009.

HU, J. **Enabling Distributed interfaces for Interactive Media**. 12th World Wide Web Conference. Budapeste, Hungria, 2003.

IERUSALIMSCHY R. **Programming in Lua**. 2. ed. Rio de Janeiro. s.l. : Lua.org, p. 308, 2006.

IZADI, S. et al. **Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media**. In Proceedings of the 16th annual ACM symposium on User interface software and technology (pp. 159-168). ACM. 2003.

ITU-T – International Telecommunication Union - **Recommendation H.761 - Nested Context Language (NCL) and Ginga-NCL for IPTV Services**. Genebra, Suiça. Abril, 2009.

JAGODIC, R., et al. **Enabling multi-user interaction in large high-resolution distributed environments**. Future Generation Computer Systems, 27(7), 914-923. 2011.

KANG, D. O., KANG, K., CHOI, S., LEE, J. **UPnP AV architectural multimedia system with a home gateway powered by the OSGi platform**. Consumer Electronics, IEEE Transactions on, 51(1), 87-93. 2005.

KERNCHEN, R. et al. **Intelligent multimedia presentation in ubiquitous multidevice scenarios. MultiMedia**, IEEE, 17(2), 52-63. 2010.

KIM J. et al. **Implementation of the DLNA Proxy System for Sharing Home Media Contents**, IEEE Transactions on Consumer Electronics, Vol. 53, No. 1, 2007.

KLYNE G. et al. **Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies**. W3C working draft, 2004.

KOLBERG M., et al. **Compatibility issues between services supporting networked appliances**. Communications Magazine, IEEE, 41(11), 136-147. 2003.

KULESZA, R. **Ginga-J-An Open Java-Based Application Environment for Interactive Digital Television Services**. Open Source Systems: Grounding Research, 34-49. 2011.

KRUCHTEN, P. B. **The 4+ 1 view model of architecture**. Software, IEEE, 12(6), 42-50. 1995.

LEW, M. **Live cinema: an instrument for cinema editing as a live performance**. In ACM SIGGRAPH 2004 Sketches (p. 117). ACM. 2004.

LIN, C. L., WANG, P. C., HOU, T. W. **Classification and Evaluation of Middleware Collaboration Architectures for Converging MHP and OSGi in a Smart Home**. Journal of Information Science and Engineering, 25(5), 1337-1356. 2009.

MARTIN, R., et al. **neXtream: a multi-device, social approach to video content consumption.** In Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE (pp. 1-5). IEEE. 2010.

MAYNES-AMINZADE, D., et al. **Techniques for interactive audience participation**. In Proceedings of the 4th IEEE International Conference on Multimodal Interfaces (p. 15). IEEE Computer Society. 2002.

MCGINITY, M., et al. **AVIE: a versatile multi-user stereo 360 interactive VR theatre** Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interacton. ACM, 2007.

MELO, E. et al**.. Arthron 1.0: Uma ferramenta para transmissão e gerenciamento remoto de fluxos de mídia**. XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Gramado, Brasil, 2010.

MILLER B. A., et al. **Home Networking with Universal Plug and Play**. *IEEE Communications Magazine*, pp 105-109, 2001.

MORENO, M. F. et al. **Ginga-NCL Architecture for Plug-ins**. TOPI '11 Proceedings of the 1st workshop on Developing tools as plug-ins (33rd International Conference on Software Engineering). Honolulu, Estados Unidos, 2011.

MORENO, M.F. et al. **Resilient Hypermedia Presentations**. Proc. of XXI IEEE International Symposium on Software Reability Engineering (Workshop W2, Software Aging and Rejuvenation). San Jose, Estados Unidos, 2010.

MORENO, M.F. SOARES, L.F.G; CERQUEIRA, R. **Uma Arquitetura Orientada a Componentes para o Middleware Ginga-NCL.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Belo Horizonte, Brasil, 2010.

MUHAMMAD, W. A., et al. **SOA4DERTS: A Service-Oriented UML profile for Distributed Embedded Real-Time Systems.** In Computers & Informatics (ISCI), 2012 IEEE Symposium on, pp. 64-69. IEEE, 2012.

PANDEY, R. K. **Architectural description languages (ADL) vs UML: a review**. ACM SIGSOFT Software Engineering Notes 35, no. 3 1-5. 2010.

PATERNO, F.; SANTORO, C.; MANTYJARVI, J.; MORI, G.; SANSONE, S. **Authoring Pervasive Multimodal User Interfaces.** International Journal of Web Engineering and Technology, v. 4. ISSN: 1476-1289. 2008.

RITCHIE, J., KUEHNEL, T. **UPnP AV Architecture: 1, For Universal Plug and Play Version 1.0**. UPnP Forum. 2002.

SANT'ANNA, F., et al. **NCLua - Objetos Imperativos Lua na Linguagem Declarativa NCL**. Proceedings of the Brazilian Symposium on Multimedia and the Web – WebMedia. Vitória, Brasil, 2008.

SCHEIBLE, J., OJALA, T. **MobiLenin combining a multi-track music video, personal mobile phones and a public display into multi-user interactive entertainment**. In Proceedings of the 13th annual ACM international conference on Multimedia (pp. 199-208). ACM. 2005.

SILVA L.D.N., TAVARES T.A., SOUZA G.L. **Desenvolvimento de programas de TVDI Explorando as Funções Inovadoras do Ginga-J**. XIV Simpósio Brasileiro de Sistemas Multimídia e Web Vila Velha, Brasil, 2008.

SILVA L. D. N. e, et al. **Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV Digital com Ginga**. T&C Amazônia Magazine. N. 12, ISSN 1678-3824, pp 75-84. Manaus, AM : s.n., 2007.

SOARES, L.F.G; RODRIGUES, R.F.; MUCHALUAT-SAADE, D.C. **Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System**. Multimedia Systems, v. 8, n. 2. Alemanha. 2000.

SOARES, L.F.G.; RODRIGUES R.F. **Nested Context Model 3.0 Part 1 – NCM Core.** Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 18/05. ISSN 0103-9741. Rio de Janeiro, Brasil. Maio, 2005.

SOARES, L.F.G; RODRIGUES, R.F. **Nested Context Language 3.0 Part 8 – NCL Digital TV Profiles**. Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 35/06. ISSN 0103-9741. Rio de Janeiro, Brasil. Outubro, 2006.

SOARES, L.F.G.; RODRIGUES, R.F.; MORENO, M.F. **Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System**. Journal of the Brazilian Computer Society. ISSN 0104-6500. 2007.

SOARES, L.F.G., COSTA, R.M.R; MORENO, M.F.; MORENO, M.F. **Multiple Exhibition Devices in DTV Systems.** ACM International Conference on Multimedia. Beijing, China. 2009.

SOARES, L.F.G.; BARBOSA, S.D.J. **Programando em NCL 3.0 – Desenvolvimento de Aplicações para o Middleware Ginga, TV Digital e Web**. ISBN 978-85-352-3457-2. Elsevier. 2009.

SOARES, L.F.G., RODRIGUES, R.F CERQUEIRA, R. BARBOSA, S.D.J. **Variable and State Handling in NCL**. Multimedia Tools and Applications. ISSN: 1380-7501, 2010.

SOUTO, N. I., FERNANDES, J.D.C. **Os Bastidores de Uma Nova Era: A Interatividade na Televisão Digital Brasileira**. In XXXI Congresso Brasileiro de Ciências da Comunicação, Natal, Brasil, 2008.

SOUZA, G.L.; LEITE, L.E.C; BATISTA, C.E.C.F. **Ginga-J: The Procedural Middleware for the Brazilian Digital TV System**. Journal of the Brazilian Computer Society, v. 13, n. 4. ISSN: 0104-6500. 2007.

TEIXEIRA, C. A., et al. **Distributed discrimination of media moments and media intervals: a Watch-and-Comment approach**. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 1929-1935). ACM. 2010.

TSEKLEVES, E., et al. **Investigating media use and the television user experience in the home**. Entertainment computing, 2(3), 151-161. 2011.

VANDERHULST, G., et al. **ReWiRe: Creating interactive pervasive systems that cope with changing environments by rewiring**. In Intelligent Environments, 2008 IET 4th International Conference on (pp. 1-8). IET. 2008.

VASILAKOS, A. V., et al. **Interactive theatre via mixed reality and Ambient Intelligence.** Information Sciences, 178(3), 679-693. 2008.

VEERAVALLI, B. et al. **Fault-tolerant analysis for multiple servers movie retrieval strategy for distributed multimedia applications**. Springer Multimedia Tools and Applications, Volume 32, Number 1, 1-27.

VIANA, N. S., MAIA, O. B., et al. **Convergence model between the IDTV Brazilian middleware and home networking software platforms**. Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia 2009 (p. 7). ACM. Fortaleza, Brasil, 2009.

VITERBO J. et al. **An ontology based on the CC/PP framework to support content adaptation in context-aware systems**, Proc. of WOMSDE 2006, in conjunction with SBES, Florianópolis, pp. 1-10, ISBN 85-7669-086-1, 2006.

VOGEL, D., BALAKRISHNAN, R. **Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users**. In Proceedings of the 17th annual ACM symposium on User interface software and technology (pp. 137-146). ACM. 2004.

WAP FORUM – WAG UAProf. Technical Report WAP-248-UAPROF-20011020-a, 2001.

W3C - World-Wide Web Consortium. **Synchronized Multimedia Integration Language (SMIL 3.0) Specification**. W3C Recommendation. Dezembro, 2008.

WEST, P., FOSTER, G. G., CLAYTON, P. G. **Content Exposure of Slide Show Presentations for Selective Download and Annotation via Mobile Devices**. In mLearn 2005-4th World conference on mLearning. 2005.