UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

INTEGRAÇÃO ENTRE O MIDDLEWARE BRASILEIRO DE TV DIGITAL E SERVIÇOS DE DISPOSITIVOS ELETRÔNICOS EM REDES OSGi

NAIRON SARAIVA VIANA

MANAUS
2009

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

NAIRON SARAIVA VIANA

INTEGRAÇÃO ENTRE O MIDDLEWARE BRASILEIRO DE TV DIGITAL E SERVIÇOS DE DISPOSITIVOS ELETRÔNICOS EM REDES OSGi

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica, área de concentração Automação de Sistemas.

Orientador: Prof. Dr.-Ing. Vicente Ferreira de Lucena Junior

MANAUS
2009

Ficha Catalográfica
(Catalogação realizada pela Biblioteca Central da UFAM)

Viana, Nairon Saraiva

V614i Integração digital entre o middleware brasileiro de TV digital e serviços de dispositivos eletrônicos em redes OSGi / Nairon Saraiva Viana. - Manaus: UFAM, 2009.
185 f.; il.

Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Amazonas, 2009.
Orientador: Prof. Dr. Vicente Ferreira de Lucena Junior

1. Televisão interativa 2. OSGi (Linguagem de programação de computador) 3. Ginga (Linguagem de programação de computador) 4. Processamento de sinais - Técnicas digitais I. Lucena Junior, Vicente Ferreira de II. Universidade Federal do Amazonas III. Título

CDU 004.932(043.3)

NAIRON SARAIVA VIANA

INTEGRAÇÃO ENTRE O *MIDDLEWARE* BRASILEIRO DE TVDI E SERVIÇOS DE DISPOSITIVOS ELETRÔNICOS EM REDES OSGi.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como parte do requisito para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 28 de Agosto de 2009.

BANCA EXAMINADORA

Prof. Dr. –Ing. Vicente Ferreira de Lucena Junior

Universidade Federal do Amazonas

Prof. Dr. José Pinheiro de Queiroz Neto

Instituto Federal do Amazonas

Profa. Dra. Maria da Graça Campos Pimentel

Universidade de São Paulo

Este trabalho é dedicado à minha família, em especial à Sr^a Fátima Viana, minha mãe, pelo exemplo de responsabilidade, bondade e honestidade, e por todos os valores ensinados que contribuíram e ainda contribuem para minha formação não só profissional, mas como cidadão.

Agradecimentos

À Deus, que me oferece todas as condições para viver e buscar meus objetivos, meu agradecimento principal.

À família, pela presença constante apesar da grande distância, pelo incentivo, por sempre acreditar em mim, e por nunca deixar que o tempo, a distância, o cansaço e os problemas da vida diária fossem maiores que o objetivo fixo em alcançar o final deste trabalho. Tenho certeza de que estão tão felizes quanto eu. Meus sinceros agradecimentos.

Aos professores do IFPI, Fábio de J. Lima Gomes e Rogério Batista, pelo incentivo ao ingresso no Programa de Mestrado em Engenharia Elétrica da UFAM.

Aos amigos de Teresina e aos novos amigos de Manaus, pelos momentos de descontração e pelo apoio nas horas difíceis; aos colegas do Mestrado, Orlewilson Maia, Luciano Lima, Dayse Rivera e Marcos Rogério, pela força para continuar até o fim da jornada. Aos colegas do grupo de pesquisa em TV Digital, Antônio Santos, Vandermi Silva, Luciano Pinto, Lady Oliveira, Ricardo Erikson e Alexandre Martiniano, pelas sugestões, revisões de artigos, grupos de estudo, etc.

Aos professores do Programa de Mestrado em Engenharia Elétrica da UFAM, em especial ao coordenador do curso em 2007, Prof. Dr. João Edgar Chaves Filho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro concedido através de bolsa de estudo.

Ao orientador, professor Vicente Lucena, pelo suporte durante o Mestrado e principalmente pelas cobranças dos artigos científicos, que contribuíram para minha formação intelectual e serviram de motivação para continuar na área acadêmica.

"Everything should be made as simple as possible, but not simpler."

Albert Einstein (1879-1955)

Resumo

Este trabalho apresenta o processo de desenvolvimento de uma plataforma que integra ambientes de aplicações para TV Digital Interativa (TVDI) e *Home Network*, sobretudo usando tecnologias compatíveis com o modelo brasileiro de TV Digital. A pesquisa nessa área se faz útil por refletir o atual estado da arte em TVDI onde, graças à convergência entre tecnologias de comunicação e de dispositivos eletrônicos no ambiente doméstico, novos tipos de aplicações interativas de TVDI têm surgido, permitindo a entrega de novos serviços ao usuário. O poder de processamento da TVDI tem se expandido de modo a dar suporte à execução de aplicações que integram os demais dispositivos eletrônicos em uma Rede Doméstica (a chamada *Home Network*).

Para a criação de tal plataforma, foram investigadas tecnologias abertas existentes para ambientes de TV Digital e *Home Network*. Com o intuito de fornecer novas aplicações para a TV Digital brasileira, foi escolhido o *middleware* Ginga como plataforma para aplicações de TVDI. Para gerência de aplicações e serviços de dispositivos eletrônicos no ambiente doméstico, dentre as alternativas analisadas, a mais apropriada foi o uso do *framework* OSGi (*Open Services Gateway Initiative*). De modo a explorar as funcionalidades das duas especificações de *middleware* Ginga (a máquina de execução, *middleware* procedural Ginga-J, e a máquina de apresentação, *middleware* declarativo Ginga-NCL), foram criados dois modelos de integração com o OSGi: um procedural (Ginga-J–OSGi) e outro declarativo (Ginga-NCL–OSGi). Ao final, para cada modelo, foram construídos cenários de validação das plataformas, com aplicações Ginga de TV Digital (*xlets* do Ginga-J ou *scripts* do Ginga-NCL) que gerenciam serviços de dispositivos eletrônicos *Home Networks* OSGi.

Palavras-chave: Ginga, OSGi, TVDI, Redes Domésticas.

Abstract

This work presents the development process of a platform that integrates interactive Digital TV (iDTV) and Home Network applications, specially using technologies compliant with the Brazilian Digital TV system. The research on this field is important because it represents the most recent state of the art in Digital TV. With the convergence of communication technologies and electronic devices in the home environment, a wide range of interactive applications have emerged, allowing to deliver new services to the user. The iDTV processing power has been expanded to support applications that interact with other electronic appliances in a Home Network.

For creating such platform, the work investigates some open technologies and standards for iDTV and Home Network. In order to provide new applications to the Brazilian iDTV, the Ginga middleware was chosen as the application platform. For managing applications and services of electronic devices, among the investigated alternatives, the most appropriate was the OSGi (Open Services Gateway Initiative) framework. With the aim to explore the capabilities of the two Ginga specifications (the procedural Ginga-J and the declarative Ginga-NCL middleware) we created two OSGi integration models: a procedural Ginga-J–OSGi and a declarative Ginga-NCL–OSGi. At the end, we built some validating scenarios for both platforms, allowing Ginga iDTV applications (Ginga-J xlets or Ginga-NCL scripts) to manage services of electronics devices in an OSGi Home Network.

Keywords: Ginga, OSGi, iDTV, Home Network.

Lista de Figuras

Lista de Tabelas

Lista de Abreviações e Siglas

8-VSB	*Vestigial Sideband Modulation*
AAC	*Advanced Audio Coding*
ABNT	*Associação Brasileira de Normas Técnicas*
ACAP	*Advanced Common Application Platform*
ANATEL	*Agência Nacional de Telecomunicações*
API	*Application Programming Interface*
ATSC	*Advanced Television Systems Committee*
AWT	*Abstract Window Toolkit*
BML	*Broadcast Markup Language*
CableLabs	*Cable Television Laboratories*
CDC	*Connected Device Configuration*
CETELI	*Centro de P&D em Tecnologia Eletrônica e da Informação*
COFDM	*Coded Orthogonal Frequency Division Multiplex*
CSS	*Cascading Style Sheets*
DASE	*DTV Application Software Environment*
DHCP	*Dynamic Host Configuration Protocol*
DIBEG	*Digital Broadcasting Experts Group*
DLNA	*Digital Living Network Alliance*
DMB	*Digital Multimedia Broadcasting*
DNS	*Domain Name System*
DOM	*Document Object Model*
DSMCC	*Digital Storage Media Command Control*
DTV–HNF	*Digital TV–Home Network Framework*
DVB	*Digital Video Broadcasting*
DVR	*Digital Video Recorder*
EPG	*Electronic Program Guide*
ETSI	*European Telecommunications Standards Institute*
FMC	*Fixed-Mobile Convergence*
FMCA	*Fixed-Mobile Convergence Alliance*
GEM	*Globally Executable MHP*
HAVi	*Home Audio Video Interoperability*
HDTV	*High Definition Television*
HomePlug	*Home-Plug Powerline Alliance*
HomePNA	*Home Phoneline Network Alliance*
HTML	*HyperText Markup Language*
HVAC	*Heating, Ventilation Air-conditioning control and lighting Control*

IM	*Instant Messaging*
IPTV	*Internet Protocol Television*
IrDA	*Infrared Data Association*
ISDB	*Integrated Services Digital Broadcasting*
ISDTV-T	*International Standard for Digital Television Terrestrial*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
JAR	*Java ARchive*
JMF	*Java Media Framework*
JVM	*Java Virtual Machine*
LAN	*Local Area Network*
MAN	*Metropolitan Area Network*
MHP	*Multimedia Home Platform*
MHEG	*Multimedia and Hypermedia Experts Group*
MoCA	*Multimedia over Coax Alliance*
MPEG	*Moving Picture Experts Group*
NCL	*Nested Context Language*
NCM	*Nested Context Model*
OCAP	*OpenCable Application Platform*
OSGi	*Open Services Gateway Initiative*
OSI	*Open Systems Interconnection*
PAN	*Personal Area Network*
PBP	*Personal Basis Profile*
PLC	*Power Line Communication*
PnP	*Plug and Play*
PSK	*Phase Shift Keying*
QAM	*Quadrature Amplitude Modulation*
QoS	*Quality of Service*
QPSK	*Quadrature Phase Shift Keying*
RTOS	*Real-Time Operating System*
SBTVD	*Sistema Brasileiro de Televisão Digital*
SDTV	*Standard Definition Television*
SOA	*Service Oriented Architecture*
SSDP	*Simple Service Discovery Protocol*
STB	*Set-Top Box*
TVD	*TV Digital*
TVDi	*TV Digital Interativa*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
UMB	*Universal Middleware Brigde*
URD	*Unidade de Recepção e Decodificação*
USB	*Universal Serial Bus*
UPN	*User Personal Network*
UPnP	*Universal Plug and Play*
VoD	*Video On Demand*
VoIP	*Voice Over IP*
WAN	*Wide Area Network*
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Markup Language*

Sumário

Capítulo 1 - Introdução

Sistemas de TV Digital oferecem novas possibilidades para o desenvolvimento de aplicações e a exploração de modelos de negócio. Desde dezembro de 2007, quando oficialmente inaugurada, a TV Digital vem se expandindo pelo país, promovendo uma mudança em todo o processo de produção e distribuição de conteúdo de algumas emissoras brasileiras. Entretanto, por ainda encontrar-se em fase de transição, a TV Digital brasileira está distante de oferecer ao consumidor os serviços já existentes na Europa e nos Estados Unidos, principalmente quando se trata de interatividade. O que há hoje no Sistema Brasileiro de TV Digital (SBTVD) é uma gradativa mudança na estrutura de transmissão do sinal, sem no entanto explorar os potenciais de uso do receptor (*Set-Top Box* ou STB) para entregar, junto com o conteúdo de áudio e vídeo, aplicações que permitirão, por exemplo, que o usuário compre algum produto exibido em um filme, ou obtenha informações da bolsa de valores, ou ainda que participe de um *chat* com amigos durante uma partida de futebol[1].

A interatividade dessas aplicações estende o papel da TV, na medida em que conecta o usuário com o mundo externo, permitindo além de receber conteúdo, produzir e compartilhar dados com a emissora ou outros telespectadores. O escopo das aplicações interativas está delimitado pelas capacidades do receptor. Com o passar dos anos, os STBs evoluíram de modo a suportar aplicações mais complexas, desde as que possibilitam o acesso à Internet até as fazem uso das interfaces de hardware do dispositivo (aplicações que acessam a USB ou leitores de *smart cards*). Hoje, os STBs são beneficiados pelos avanços em tecnologias de comunicação e eletrônica, assim como dispositivos móveis, *Laptops*, *PDAs*, e demais dispositivos eletrônicos. A conectividade é explorada nesses

[1]Embora já existam alguns exemplos de casos de transmissão de aplicações interativas em redes de TV aberta no Brasil, estas ainda encontram-se em fase de testes.

dispositivos e encontra uma série de novas aplicações na TV Digital. Além de aplicações que integram na TV Digital Interativa (TVDI) emissora e telespectador, os potenciais do STBs podem ser explorados de modo a permitir que estes compartilhem dados com os demais dispositivos eletrônicos da residência, criando um ambiente doméstico conectado onde redes (de telefonia, de Internet, de TV, etc.), aplicações (enviar um SMS para um celular através da TV) e dispositivos (assistir TV no celular) convergem.

Este trabalho está inserido nesse contexto. É feito um estudo sobre a convergência da TV Digital e dispositivos eletrônicos do ambiente doméstico conectado, de modo a verificar o uso de aplicações interativas de TVDI que permitam que ela gerencie os demais dispositivos. Tecnicamente, serão investigadas as tecnologias que tornam possível tal integração e será desenvolvida uma proposta compatível com o modelo brasileiro de TV Digital, com o objetivo de oferecer novas aplicações de interatividade e conectividade à TV brasileira.

Neste capítulo será introduzido o trabalho de Dissertação desenvolvido sobre o tema acima. A motivação (seção 1.1) procurará identificar os benefícios e alguns desafios da integração entre TVDI e dispositivos, segundo a estratégia de centralizar os serviços dos dispositivos na TVDI. O escopo do trabalho é definido, com a apresentação dos objetivos gerais e específicos (seção 1.2) e em seguida, com a elaboração da metodologia utilizada no trabalho (seção 1.3). Por fim, é apresentada a estruturação dos capítulos desta Dissertação (seção 1.4).

1.1 Motivação

A evolução das tecnologias dos chamados dispositivos eletrônicos de consumo (*Consumer Electronics Devices*) é caracterizada pelo aumento do nível de conectividade entre eles. Nos dias de hoje isso encontra aplicação na utilização de dispositivos eletrônicos como meios de integração e compartilhamento de serviços entre o mundo interno e externo às chamadas redes residenciais/domésticas (*Home Networks*, HN) (Gong, 2001a). Com a evolução do hardware e o surgimento de novas tecnologias de dispositivos, um volume maior de informação passa a estar disponível: serviços de conteúdo através de celulares,

pagers (áudio, vídeo, entretenimento, etc.) bem como informações sobre dispositivos eletrônicos em geral, como em uma residência (controlar temperatura do ar condicionado, acender lâmpadas, atender ao videofone, etc.).

Entre as mais recentes abordagens de convergência entre dispositivos eletrônicos, destacam-se as que inserem a TVDI nesse contexto. Assim como qualquer outro dispositivo da HN, a TVDI pode prover serviços e integrar-se a outros dispositivos da rede. Soma-se a isso o fato de as características dos receptores de TV Digital, agregando cada vez mais interfaces para suportar novas tecnologias de comunicação, apontarem os STBs como equipamentos apropriados para centralizar o gerenciamento dos demais dispositivos em uma HN (Ge et al., 2007; den Hartog et al., 2004), papel exercido pelos *Home Gateways* (HG). Dessa forma, os serviços convencionais da TVDI podem ser integrados aos serviços das plataformas de gerenciamento de *Home Networks*, o que estende o escopo das aplicações que podem ser criadas e contribui para uma nova experiência do usuário com o ambiente interativo interconectado.

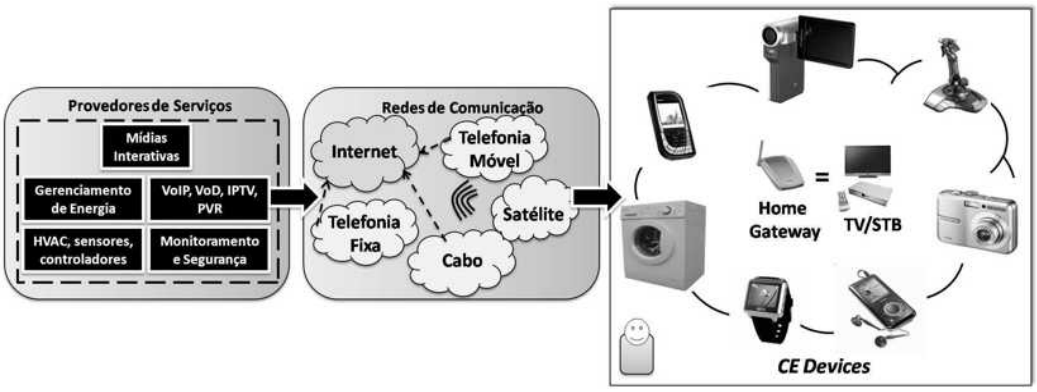


Figura 1.1: Utilização da TVDI como um *Home Gateway*.
FONTE: Do Autor.

A Figura 1.1 ilustra um modelo de utilização de STBs como *Home Gateways*, onde provedores de serviços, redes de comunicação e dispositivos convergem no ambiente doméstico centralizados na plataforma comum STB-HG. O uso dos receptores de TV Digital como *Home Gateways* permite a construção de cenários que variam dos mais simples, desde a administração da residência por meio da interface do televisor ou a realização de tarefas comuns ao ambiente doméstico (como o controle de portas, câmeras e aplicações em segurança), até cenários mais específicos como a impressão de

um documento qualquer associado a um programa de TV ou o controle do ambiente sincronizado com o áudio e vídeo na transmissão de um filme. Duas situações comuns ao dia-dia da vida moderna, possíveis aplicações dessa abordagem, são descritas abaixo:

Cenário 1 : *No receptor de Carlos, um guia de recomendação de conteúdo é encarregado de gravar os programas de TV de seu interesse. Visto que ele é interessado em cozinhar, um documentário sobre o tema está sendo exibido. Quando Carlos decide assisti-lo é notificado que há uma receita disponível que pode ser acessada e gravada localmente. Carlos, entretanto, prefere uma cópia impressa e então seleciona a opção "imprimir a receita", que prontamente envia o documento à impressora mais próxima disponível na residência.*

Cenário 2 : *Enquanto Ana está assistindo TV, uma pequena janela aparece na tela mostrando que há alguém chamando à campainha. Ana então reconhece que se trata de sua amiga e seleciona a opção "abrir a porta". Alguns minutos depois, outra pessoa chama à campainha, mas Ana não a reconhece. Então, ela ativa a comunicação via áudio e, depois de uma rápida conversa, decide não abrir a porta e continua assistindo TV.*

Entretanto, a implementação do STB como um *Home Gateway* não é tão simples quanto os cenários que podem ser criados. Uma das razões para esta dificuldade é o número de tecnologias e padrões distintos atuando em uma mesma rede doméstica. Num único ambiente podem co-existir, centralizados no *Home Gateway*, um computador com uma interface de comunicação IEEE 1394 (IEEE, 1996), um ar-condicionado com *Universal Plug and Play* (UPnP) (UPnP, 2009), câmeras com a tecnologia HomePNA (*Home Phoneline Network Alliance*) (HomePNA, 2009), uma TV com CE Bus (*Consumer Electronic Bus*)(CEBus, 2009) e um celular com a tecnologia Bluetooth. A diversidade de padrões e a necessidade de interoperabilidade são portanto questões importantes no projeto de arquiteturas de *Home Gateways* (Boucouvalas, 2002).

Para resolver os problemas comuns ao projeto de *Home Gateways*, algumas entidades oferecem especificações de referência para a definição da arquitetura desses dispositivos a fim de prover compatibilidade entre as aplicações produzidas e as plataformas de execução. Sob o ponto de vista da arquitetura de software para *Home Gateways*, destaca-se o consórcio *Open Services Gateway Initiative*

(OSGi)(OSGi-Alliance, 2009) que especifica uma plataforma com componentes que permitem a conexão da *Home Network* com o mundo externo, tendo como base um modelo orientado a serviços. O OSGi é um ambiente construído sobre conceitos de modularidade e componentização de software. O *framework* provê um mecanismo para gerenciamento de serviços e unidades de aplicações conhecidas como *bundles*(Tavares e Valente, 2008).

De forma semelhante, para TV Digital existem algumas soluções concorrentes. Os padrões de TV Digital definem especificações em todos os níveis da arquitetura de sistema. No que se refere ao desenvolvimento de aplicações, destacam-se *middlewares* como MHP (*Multimedia Home Platform*), ACAP (*Advanced Common Application Platform for Interactive Television*) e mais recentemente o *middleware* brasileiro Ginga. Esses *middlewares* oferecem modelos para funções genéricas no STB como gerenciamento de recursos, canal de retorno e controle do ciclo de vida dos aplicativos, os *xlets* (Morris e Smith-Chaigneau, 2004).

Dada a existência desses padrões, o uso da TVDI (ou STB) como um *Home Gateway* pode ser realizado por meio da construção de modelos de colaboração entre plataformas de software que representem cada domínio: a TVDI (MHP, ACAP, GINGA, etc) e as aplicações na *Home Network* (OSGi, UPnP, etc). Essa é a proposta desta Dissertação: investigar os mecanismos que caracterizam a integração de aplicações para STBs (*middleware*) e *Home Gateways* (*frameworks*) dando enfoque principal na utilização da plataforma brasileira para aplicações de TVDI, o Ginga.

1.2 Objetivos e Contribuição

Este trabalho fornece uma plataforma para a criação de aplicações baseada em um conjunto de componentes que habilitam a comunicação entre aplicações de TV Digital e de *Home Network*. Esses componentes atuarão em conjunto para oferecer ao programador de aplicações interativas uma interface de desenvolvimento apropriada, de modo a facilitar a construção de aplicações para o ambiente TVDI-HG. O trabalho explora a utilização, em um ambiente comum, das estruturas de software que representam duas tecnologias distintas: o *middleware* brasileiro, dividido em suas duas especificações (Ginga-J e Ginga-

NCL) e o *framework* para a gerência dos serviços dos dispositivos na rede doméstica, o OSGi. Com isso é estabelecido um modelo que dá suporte à colaboração entre aplicações Ginga e OSGi, exportando serviços e funções de um domínio para o outro, explorando as funcionalidades providas pelas duas especificações e ao mesmo tempo considerando as limitações de cada uma.

Os objetivos foram organizados da seguinte forma:

Objetivo Geral: Construir uma solução de integração entre plataformas de software para TV Digital e redes de dispositivos eletrônicos através de um conjunto de componentes de software que promovam serviços e implementem funções seguindo um modelo colaborativo, dando suporte à verificação de cenários de convergência TVDI-HG; em especial, no contexto do modelo brasileiro de TV Digital.

Objetivos Específicos:

- Pesquisar os mecanismos de funcionamento definidos pelas especificações de software para TVDI e *Home Network*.
- Analisar os principais trabalhos associados, estudando suas abordagens e identificando as melhores características de cada.
- Criar um modelo próprio de colaboração TVDI-HG, o qual dará suporte à construção dos componentes da plataforma.
- Estender o conjunto de componentes criado de forma a promover compatibilidade com o *middleware* Ginga, sobretudo pela utilização da especificação declarativa do *middleware*, o Ginga-NCL.
- Validar os resultados do trabalho por meio da verificação de alguns cenários de colaboração TVDI-HG.

Ao longo da Dissertação os objetivos traçados foram desenvolvidos, utilizando a metodologia descrita a seguir.

1.3 Metodologia de Trabalho

O processo adotado para o projeto da Dissertação envolve atividades iniciais como a fundamentação da teoria relacionada, com o estudo das normas que definem as tecnologias MHP/Ginga e OSGi, os componentes principais de cada modelo e identificação dos mais relevantes para a implementação da proposta. Modelos complementares às definições dos padrões Ginga e OSGi foram também relacionados e comparados de forma a justificar a escolha das tecnologias na integração TVDI-HG.

Depois de estabelecidos os conceitos fundamentais, a próxima fase foi o estudo dos modelos de software para redes de dispositivos em uma *Home Network*, sobretudo os que inserem a TV Digital como centro dessa arquitetura. Os métodos nesses trabalhos relacionados foram analisados e forneceram os parâmetros de comparação com a nova plataforma. Paralelamente a essas atividades, um levantamento sobre as principais soluções abertas para desenvolvimento e testes do projeto foi realizado. A montagem do ambiente real de execução da plataforma foi realizada aproveitando-se do suporte oferecido pelo Laboratório de TV Digital do Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação (CETELI), do Mestrado em Engenharia Elétrica da Universidade Federal do Amazonas. Para validação da proposta alguns estudos de caso (descritos no Capítulo 7) foram realizados.

As atividades identificadas a seguir compõem o processo de desenvolvimento do projeto, dividido em 4 grupos distintos: (1) *Estudo sobre o estado da arte*; (2) *Modelagem e Implementação*; (3) *Experimentos práticos* e (4) *Elaboração da dissertação.*

A etapa de estado da arte envolve: (1) Introdução, onde são analisados conceitos de TV Digital e *Home Network* (Arquitetura, Protocolos, Padrões); (2) Arquiteturas de *software* para *Home Network* e TV Digital (OSGi e MHP/GINGA; especificações, modelos e componentes); e (3) Modelos de colaboração em *software* para TV Digital e *Home Netowrk*, com a análise das estratégias dos trabalhos relacionados.

Na etapa de modelagem e implementação tem-se: (1) Estudo das API's e componentes (MHP/GINGA e OSGi); (2) Levantamento das principais soluções abertas

para desenvolvimento e testes do plataforma; (3) Estudo comparativo e implementação dos modelos propostos, onde os resultados obtidos são analisados para a construção da plataforma; (4) Criação de um modelo de colaboração Ginga-OSGi e desenvolvimento da plataforma de execução de aplicações; (5) Testes iniciais e análise dos resultados nas plataformas abertas; e (6) Finalização da primeira versão do protótipo.

A etapa de experimentos práticos envolve: (1) Levantamento das soluções para implementação real do projeto (montagem da infra-estrutura no laboratório do CETELI-UFAM); (2) Montagem do cenário para os estudos de caso e execução da plataforma; (3) Testes da implementação compatível com o Ginga; e (4) Ajustes finais e entrega do projeto.

Já na última etapa foram realizadas atividades referentes ao documento desta Dissertação: (1) Elaboração dos capítulos; (2) Entrega da primeira versão; (3) Revisões; e (4) Entrega da versão final.

1.4 Organização dos Capítulos

A organização dos capítulos procura refletir as atividades realizadas bem como os objetivos definidos. Após a introdução deste capítulo, será apresentada no Capítulo 2 uma visão geral dos Sistemas de TV Digital e de *Home Network*, estudando seus principais componentes, sobretudo os relacionados às aplicações (para TVDI) e serviços (para *Home Network*). Os papéis dos STBs e HG são apresentados, dentro do modelo geral dos sistemas.

No Capítulo 3 uma análise mais detalhada das especificações de software abertas para *middleware* de TV Digital e *framework* de *Home Network* será feita. Para a TV Digital, serão analisados os principais padrões e identificado o mais apropriado para a criação dos componentes que serão integrados ao *Home Gateway*. Por sua vez, para o *Home Gateway* serão analisadas plataformas de serviços disponíveis, seus modelos de operação e qual plataforma apresenta as melhores características para integrar-se com o *middleware* Ginga.

Escolhidas as plataformas de software, no capítulo seguinte (Capítulo 4) será feito

um estudo das principais referências bibliográficas sobre o tema de integração STB-HG. Será justificado o uso de STBs como HGs e, na literatura, serão organizados os principais trabalhos que implementam tal uso, de modo a definir quais características serão mais adequadas para a construção da plataforma usando o *middleware* Ginga e o *framework* OSGi.

Nos capítulos seguintes (Capítulos 5 e 6) serão definidos o modelo e a implementação dos componentes das plataformas Ginga-J–OSGi e Ginga-NCL–OSGi. Em seguida, no Capítulo 7 serão apresentados os cenários construídos sobre a plataforma Ginga-OSGi para, por fim, identificar alguns resultados obtidos no trabalho, pontos de melhoria e sugestões para trabalhos futuros (Capítulo 8).

Capítulo 2 - Visão Geral das Tecnologias

A fim de oferecer uma base elementar para a análise dos principais assuntos referentes à pesquisa desenvolvida sobre o tema desta Dissertação, neste capítulo as principais tecnologias que compõem os atuais Sistemas de TV Digital Interativa (TVDI) e Sistemas de Redes Domésticas (*Home Networks*) são apresentadas. Para o primeiro, a TVDI é caracterizada, dentro de um contexto de padrões e protocolos para as principais funções do sistema (como *multiplexação*, *modulação* e *middleware*). Como complemento, o Sistema Brasileiro de TV Digital é apresentado. Para o segundo item, são enumeradas as principais tecnologias que compõem as atuais Redes Domésticas, e alguns conceitos importantes que definem esse tipo de rede, como *Home Gateway* e serviços. Para cada sistema, é feita uma breve introdução sobre as tecnologias de software associadas (*middleware* para TV Digital e *framework* para serviços em Redes Domésticas), que posteriormente serão analisadas em maiores detalhes, quando da definição sobre as plataformas de software nas quais será desenvolvido o trabalho.

2.1 TV Digital Interativa

Um sistema de TV Digital Interativa apresenta componentes que dão suporte à distribuição de conteúdo digital em redes de difusão (*Broadcasting*). Assim como o modelo analógico convencional, o sistema de TVDI tem por base o conceito de difusão de dados, no modelo de distribuição de serviços. Segundo Piccione (2004), a *difusão de dados* é definida como a entrega de um programa ou serviço, televisivo ou de rádio, para consumidores que possuam equipamentos apropriados para a recepção dos mesmos.

Já o modelo de distribuição de serviços define componentes da cadeia de valor em um sistema de TV (seja digital ou analógico, conforme Schwalb (2004)), tais como emissoras de conteúdo, operadores de canais de difusão e telespectadores. Na TV Digital, a difusão de dados e todo o modelo da cadeia de valor são realizados sob a forma de dados digitais, ou *datacasting*. Isso leva à possibilidade de aplicação de tecnologias existentes na Internet (como técnicas de compressão e multiplexação de dados) nos dados digitais das transmissões de TV, oferecendo diversas vantagens em relação à transmissão analógica, tais como: maior quantidade de conteúdo transmitido; diminuição de interferências (chuviscos na tela); melhor qualidade de som e imagem; além da capacidade de execução de aplicações vinculadas ao conteúdo, as chamadas aplicações interativas (Crinon et al., 2006).

Para promover o *datacasting*, os sistemas de TV Digital apresentam alguns componentes e conceitos definidos a seguir.

2.1.1 Caracterização: Componentes do Sistema

Um típico sistema de TVDI apresenta três componentes principais: (1) difusor ou provedor de serviços de difusão, (2) meios de difusão e (3) receptor (ver Figura 2.1) (Montez e Becker, 2005). Esses três componentes ou subsistemas tratam da informação digital na forma de serviços, unidades elementares de áudio/vídeo/dados que compõem um programa. Serviços são organizados fisicamente como fluxos de transporte, sob os quais são realizadas operações como compressão e multiplexação, definidas pela norma ISO-IEC 13818-x (ISO, 2009a).

O difusor (1) é representado pela emissora de TV ou qualquer entidade capaz de realizar o processo de edição, formatação e posterior disponibilização de conteúdo na rede de difusão. Esse processo envolve etapas de codificação áudio/vídeo (ISO-IEC 13818-1, 2), inserção de dados/aplicativos (padrão DSMCC - *Digital Storage Media Command Control*, ISO-IEC 13818-6) (ISO, 2009b) e multiplexação (ISO-IEC 13818-2). Nessa etapa, os dados que serão difundidos estão em formato de Fluxo de Transporte (definido pelo *MPEG-2 Systems* (ISO, 2009a)). O último passo no difusor é a modulação para o meio de

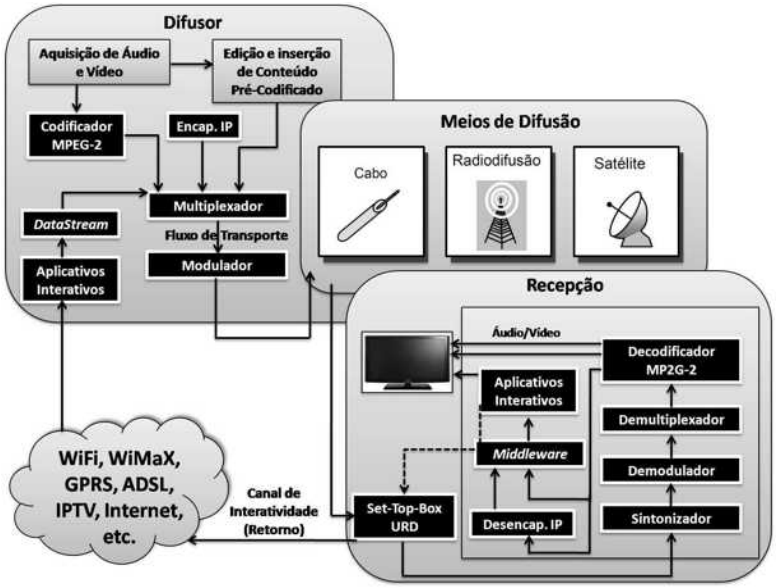


Figura 2.1: Componentes do Sistema de TVDI.

FONTE: Adaptado de Montez e Becker (2005) e Barbosa e Soares (2008).

difusão, que dependerá do padrão de TVDI adotado. Para a difusão podem ser adotados três meios: cabo, radiodifusão (usado em transmissões convencionais) e satélite.

Ao receber o sinal digital da rede de difusão, o STB (*Set-top Box* ou URD - *Unidade de Recepção e Decodificação*) realiza o processo inverso sobre o Fluxo de Transporte. Para isso, o STB apresenta componentes de hardware para operações específicas de demodulação, demultiplexação e decodificação. O sinal recebido é demodulado (sintonizado e retirado do canal de freqüência), demultiplexado e entregue para decodificadores de áudio, vídeo e dados para processamento e apresentação. Uma característica importante dos STBs é o suporte à execução de aplicativos interativos os quais podem ou não estar vinculados ao conteúdo de áudio/vídeo e permitem o envio de informações no sentido inverso da difusão, do usuário para a emissora. No sistema de TVDI essa função é habilitada pelo subsistema de canal de retorno (ou canal de interatividade).

Um sistema de TVDI pode ser também definido pelo conjunto de tecnologias utilizadas em cada nível de sua arquitetura. Para cada função do sistema, há uma variedade de tecnologias disponíveis. Por exemplo, no nível de modulação podem ser utilizadas tecnologias 8-VSB (*Vestigial Sideband Modulation*), COFDM (*Coded*

Orthogonal Frequency Division Multiplex), QAM (*Quadrature Amplitude Modulation*), PSK (*Phase Shift Keying*), QPSK (*Quadrature Phase Shift Keying*), etc. No nível de codificação áudio/vídeo podem ser usados MPEG AAC ou Dolby AC3. Assim, considerando essa diversidade de soluções, há a necessidade de padronização de um conjunto de tecnologias para um sistema de TVDI.

Em todo o mundo, consórcios entre indústrias, emissoras, empresas e governos surgiram com o objetivo de promover padrões para sistemas de TV Digital de forma a garantir compatibilidade entre tecnologias, fabricantes e operadores de redes de difusão. Fatores de ordem econômica, social, política, e geográfica contribuíram para a divisão de partes do globo no uso de alguns padrões. Dessa forma, como esquematizado por Morris e Smith-Chaigneau (2004) e Fernandes et al. (2004):

- Na Europa, foi definido o *Digital Video Broadcast* (DVB) (DVB, 2009), com variações para os três tipos de meios de transmissão: DVB-S (Satélite), DVB-C (Cabo) e DVB-T (Terrestre). Para modulação são usados QPSK (DVB-S), QAM (DVB-C) e COFDM (DVB-T). Para codificação de áudio e vídeo escolheu-se MPEG2 BC e MPEG2 SDTV, respectivamente. Para transporte, *MPEG-2 Systems*.

- Nos Estados Unidos, Canadá e México, definiu-se o *Advanced Television Systems Committee* (ATSC) (ATSC, 2009) com modulação 8-VSB, transporte MPEG-2 Systems e codificação de áudio e vídeo Dolby AC3 e MPEG2 HDTV, respectivamente.

- O *Digital Broadcasting Experts Group* (DiBEG) definiu no Japão o *Integrated Services Digital Broadcasting, Terrestrial* (ISDB-T) (ISDB, 2009), com modulação COFDM, transporte MPEG-2 Systems e codificação de áudio e vídeo MPEG2 AAC e MPEG2 HDTV, respectivamente.

Esses três padrões são os dominantes ao redor do mundo. Um padrão que também merece destaque é o *OpenCable*, definido pela *CableLabs* (CableLabs, 2009), uma organização associada a indústria de TV a cabo norte americana. Entretanto, países como China e Brasil decidiram produzir seus próprios padrões, dadas suas

características peculiares e tomando por base as funcionalidades mais apropriadas dos padrões existentes. Dessa forma, a China definiu em 2006 o *Digital Multimedia Broadcasting Terrestrial/Handheld* (DMB-T/H), focado principalmente na convergência entre serviços de difusão e móvel (Burger et al., 2007). Já o Brasil, definiu o ISDTV (*International Standard for Digital Television*) (ISDTV, 2009), detalhado na seção 2.1.3.

2.1.2 Middleware, Interatividade e Aplicativos

Como meio de interface entre o consumidor/usuário de serviços e os provedores, o STB exerce papel central no sistema de TVDI. Visto como um conjunto de camadas, o STB apresenta níveis que agregam um conjunto de componentes de hardware e software. O'Driscoll (2000) divide as funções do STB em 3 camadas:

- *Hardware*: Funções para tratamento do sinal digital, com técnicas de decodificação, demodulação e demultiplexação, além de interface física de comunicação (USB, Ethernet, RF, Serial RS-232, smart card, etc).

- *Sistema Operacional*: Inclui um *Sistema Operacional de Tempo Real* (Real-Time OS, RTOS) e *device drivers*. Entre RTOSs mais comuns estão VxWorks, pSOSystem, Microware's DAVID OS-9 e Microsoft Windows CE.

- *Middleware*: Camada constituída por um conjunto de tecnologias e ferramentas de software que fornecem às aplicações um ambiente adequado para execução independente de plataforma (hadrware ou sistema operacional). O *middleware* pode ser caracterizado como um conjunto de API's (*Application Programming Interface*) que oferecem funções sobre operações no STB como persistência de dados, acesso à Internet, sintonização de canais, etc. Cada padrão de TV Digital define um *middleware* específico; entre os principais estão: DVB-MHP (*Multimedia Home Platform*); ATSC-DASE/ACAP (*DTV Application Software Environment*)/(*Advanced Common Application Platform*), OpenCable-OCAP (*OpenCable Application Platform*) e o ISDTV-Ginga. No Capítulo 3 cada plataforma será analisada separadamente.

Na camada de aplicação da arquitetura, estão os aplicativos e serviços interativos. Esses serviços podem ser agrupados de acordo com o nível de interatividade de cada um. Monteiro (2005) adota uma classificação que leva em conta o nível de utilização do *canal de retorno* e agrupa os aplicativos em três níveis de interatividade: *local* (sem canal de retorno), *intermitente* (uso parcial) e *permanente* (uso total).

Já Montez e Becker (2005) utilizam como critério a *natureza do serviço* e organizam as aplicações interativas em grupos: EPG (*Electronic Program Guide*); Portais de TV Interativa (*Walled Gardens*); Internet; Comércio Eletrônico (*T-Commerce*); Banco Eletrônico (*T-Banking*); Aplicações Educacionais (*T-learning*), Sociais (*T-Government*) e de saúde (*T-Health*); Aplicações Transversais aos Canais ou independentes de conteúdo (*Stand Alone Applications*); Console de Jogos e Publicidade Interativa.

Além dos serviços citados, cada vez mais os STBs incorporam interfaces de hardware que dão suporte a novas tecnologias como *Bluetooth*, *Zigbee*, *WiFi* e *USB* (conforme será discutido no Capítulo 4), o que expande a natureza dos serviços oferecidos para o usuário de TVDI. Por exemplo, num STB dotado com a tecnologia Bluetooth é possível enviar o Guia de Programação Eletrônica (EPG) para um celular; caso haja uma interface WiFi é possível a integração com sensores, como de temperatura, permitindo o monitoramento pela tela da TVDI; caso haja uma interface USB é possível acessar uma impressora e enviá-la um documento associado a um programa sendo exibido; através da interface *Ethernet* é possível acessar câmeras em outros ambientes e controlar portas ou janelas. Esses novos cenários incrementam o modelo atual de serviços da TVDI. Um novo modelo, que insere a TV Digital como centro de controle de dispositivos eletrônicos, é o objeto de estudo desta Dissertação, que se concentra em oferecer uma solução de software que expanda as funcionalidades do *middleware* de TV Digital para o acesso a serviços em redes de dispositivos eletrônicos com essas tecnologias.

2.1.3 O Modelo Brasileiro de TV Digital

O Brasil optou pela definição de um modelo próprio de TV Digital. Para isso instituiu, por meio do Decreto-Lei n. 4091, de 26 de novembro de 2003, o Sistema

Brasileiro de TV Digital (SBTV). O decreto, além de dispor sobre as funcionalidades técnicas da arquitetura de sistema digital, deixou claro que o modelo a ser criado deveria contemplar características próprias da realidade brasileira, preocupando-se com a inclusão social por intermédio da TV e com o desenvolvimento da indústria nacional (Brasil, 2003).

A cargo da Agência Nacional de Telecomunicações (ANATEL) estava a tarefa de realizar testes para definir de qual padrão (dentre DVB, ATSC e ISDB) derivaria o SBTVD. Em 2006, o governo brasileiro definiu que o Sistema Brasileiro de Televisão Digital Terrestre (SBTVD-T) adotaria como base o padrão de sinais digitais terrestres do ISDB-T japonês (artigo 3º) e possibilitaria (artigo 6º) a transmissão digital em alta definição (HDTV); em definição padrão (SDTV); a transmissão digital simultânea para recepção fixa, móvel e portátil; e a interatividade. A transição completa do sistema analógico para o digital ocorrerá em 10 anos (artigo 10º) (TeleBrasil, 2008). Atualmente, o padrão encontra-se normatizado pela ABNT (Associação Brasileira de Normas Técnicas) sob o nome ISDTV (*International Standard For Digital Television*).

No que se refere a plataformas de *middleware* para a TVDI brasileira, o ISDTV instituiu o Ginga. A especificação é compatível com as recomendações ITU J.200, J.201 e J.202 (Souza Filho et al., 2007) fornecendo um ambiente que dá suporte a conteúdo procedural e declarativo, por meio uma máquina de execução para aplicações baseadas em linguagem Java, e uma máquina de apresentação para o tratamento de conteúdo declarativo baseado principalmente na linguagem NCL (*Nested Context Language*) (Soares e Moreno, 2007). Na pilha de software do Ginga há também um Núcleo Comum (*Ginga Common Core*) responsável por prover serviços que integram os outros dois ambientes. O *Common Core* contém decodificadores comuns de conteúdo que servem tanto ao Ginga-NCL quanto ao Ginga-J, além de implementar um mecanismo-ponte para habilitar a interação entre conteúdo procedural (miniaplicativos Java) e NCL.

As características do Ginga (tanto da especificação procedural como declarativa) serão analisadas e confrontadas com as características de outros *middleware* para TVDI quando da definição do *middleware* utilizado na plataforma criada nesta Dissertação, no Capítulo 3.

2.2 Tecnologias para Home Network

O termo *Home Network*[1] ou Rede Doméstica vem sendo usado com freqüência para designar redes de dispositivos eletrônicos que compartilham entre si, no ambiente doméstico, *serviços* de maneira independente de tecnologia, bem como são capazes de utilizar meios de comunicação para prover ou utilizar serviços remotos, sobretudo através da Internet (Kim et al., 2005). Segundo Dixit e Prasad (2008) o surgimento e a evolução das redes domésticas têm sido guiados pela necessidade constante de provedores (operadores de TV, celular, Internet) em diferenciar seus serviços, o que acabou levando-os a convergir para o ambiente doméstico. Como exemplos de convergência, podem ser citados serviços de banda larga e de acesso a redes de TV por operadoras de telefonia; serviços de entretenimento pela TV Digital integrados ao *e-commerce* típico da Internet; serviços de Voz sobre IP (*Voice Over IP*, VoIP) ou *Instant Messaging* providos por operadores de TV a cabo, entre outros.

A evolução das tecnologias de telecomunicações e computação e o barateamento de microprocessadores e chips eletrônicos também tem contribuído para a convergência de serviços e aplicações (Decina e Trecordi, 1997), pois permitiu o aparecimento de diversos dispositivos eletrônicos com funções que variam desde entretenimento (TV móvel, PDA, *Palmtop*, iPOD) até o gerenciamento da residência (como medidores de consumo de energia, sistemas de controle de temperatura, sistemas de segurança, etc.).

A convergência de aplicações, serviços e dispositivos em *Home Networks* resulta atualmente num mundo conectado, levando à evolução de modelos de negócios para prover novos tipos de serviços e atrair um maior número de consumidores. A capacidade de adaptação desses novos tipos de rede em meio ao surgimento constante de novas tecnologias e formas de integração será o critério fundamental para tornar a rede extensível a novos requisitos e necessidades (Alam e Prasad, 2008). Adicionalmente, os modelos de negócios também precisarão ser mais flexíveis e dinâmicos de modo a acompanhar essa constante evolução.

[1] A partir desse ponto, em toda a Dissertação o termo *Home Network* (ou HN) será utilizado para referenciar Redes Domésticas.

Para garantir a flexibilidade que permitirá uma integração maior entre fabricantes, serviços, dispositivos e tecnologias de comunicação, tornando os negócios do futuro em *Home Networks* dinâmicos, Alam e Prasad (2008) levantam duas principais questões que precisam ser respondidas:

- Onde a *inteligência* deverá residir no sentido de prover uma nova geração de serviços: nas redes, nos dispositivos (serviços), nas aplicações, ou em todos eles?

- Qual será a plataforma apropriada para a criação desses serviços de nova geração? Isso trará uma série de disputas entre diferentes abordagens e visões de diferentes segmentos da indústria, tais como gigantes em *redes* (Cisco), gigantes *players de mídia* (Apple, Sony, Microsoft), gigantes em *redes (móveis/wireless) e dispositivos* (Nokia, Ericsson, Motorola, Samsung) e gigantes no ramo de *aplicações/serviços* (Google, Yahoo, Microsoft). Essas companhias poderão impor seus modelos de negócios para aceitação no mercado mas, em último caso, a plataforma apropriada deverá ser aberta para negócios e desenvolvedores, sobre a qual estes poderão construir seus serviços (Dixit e Prasad, 2008).

A seguir será feita uma descrição dos principais componentes da arquitetura de *Home Networks*, seja do ponto de vista de provedores de serviços, de tecnologias de comunicação e dispositivos ou de consumidor.

2.2.1 Caracterização de Home Networks

Atualmente uma das principais dificuldades em se definir uma arquitetura genérica para *Home Networks* está na diversidade de provedores de serviços, fabricantes de dispositivos e tecnologias de comunicação. O esquema apresentado em Marples e Kriens (2001) pode ser estendido para redes em geral, podendo-se identificar os seguintes elementos em *Home Networks*, ilustrados na Figura 2.2:

- *Provedores de Serviços* (*Service Providers*): representa os provedores de serviços como difusão de dados (*Broadcast*), Telefonia, Multimídia, VoIP, Internet Banda

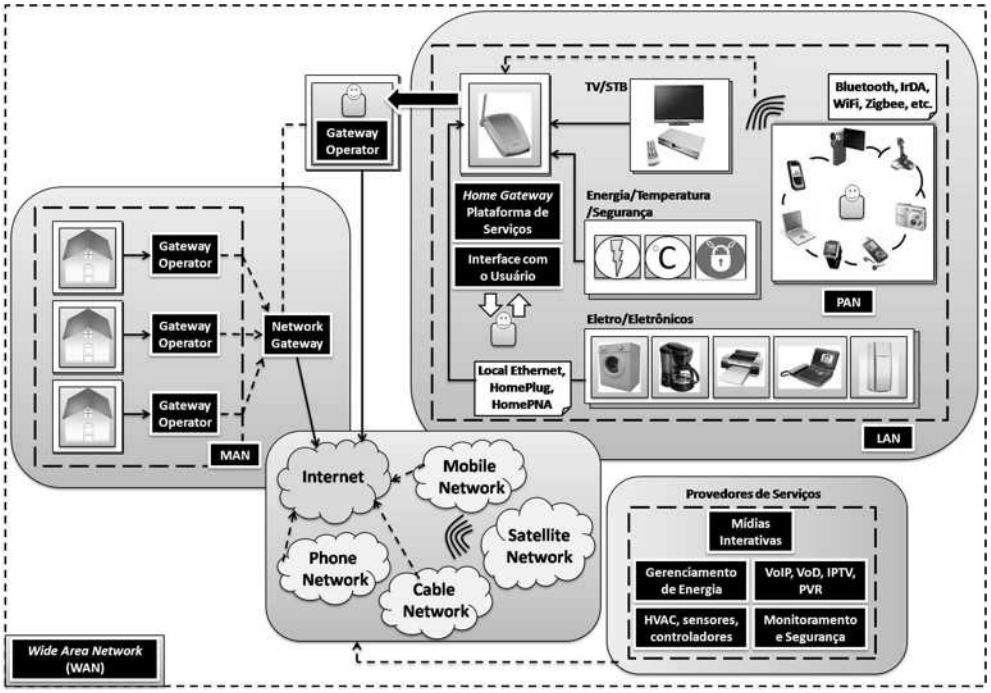


Figura 2.2: Principais Componentes do Modelo Geral das Home Networks.
FONTE: Adaptado de Marples e Kriens (2001).

Larga, etc. Cada um desses serviços pode estar associado a um dispositivo da *Home Network*.

- *Operador do Gateway* (*Gateway Operator*): entidade que concentra serviços de provedores distintos e os distribui para as Home Networks.

- *Gateway de Rede* (*Network Gateway*): elemento de interconexão de um conjunto de *Home Networks* a redes distintas dos provedores de serviços do sistema (redes de Difusão, redes a cabo, Telefonia, Internet, etc).

- *Plataforma de Serviços e Aplicações* (*Home Gateway*): Também chamada Gateway Doméstico ou Residencial, é o ambiente para a execução de aplicações e entrega de serviços, locais ou remotos, relacionados a dispositivos da residência.

Na Figura 2.2 os elementos apresentados compõem redes que variam de pequena a larga escala (Gerasimenko, 2007). Redes PAN (*Personal Area Network*) agregam unidades de dispositivos como celulares e PDAs sob tecnologias como *Bluetooth* ou Infravermelho (IrDA). Redes LAN (*Local Area Network*) cobrem pequenas áreas como

redes de computadores ou de dispositivos sob tecnologias IEEE 802.3 *Ethernet* (com fio) ou IEEE 802.11 *WiFi* (sem fio). No modelo da Figura 2.2, as LANs são centralizadas no Gateway Residencial e este permite ao operador de gateway controlar os dispositivos da rede. Redes MAN (*Metropolitan Area Network*) cobrem áreas maiores como conjuntos de prédios ou até uma cidade, sendo conectadas principalmente por tecnologias sem fio ou de fibra óptica. O Gateway de rede interconecta várias MANs a redes maiores como as dos provedores de serviços ou a Internet. No último nível, a rede em larga escala ao redor do globo é chamada WAN (*Wide Area Network*).

O modelo apresentado na Figura 2.2 permite a entrega de serviços de natureza diversa à *Home Networks*. Levando em conta esses serviços, Alam e Prasad (2008) e Dixit e Prasad (2008) categorizam *Home Networks* em quatro áreas, ilustradas na Figura 2.3 e explicadas a seguir: *Home Computing*, *Home Network Entertainment*, *Home Communications* e *Home Monitoring and Management*.

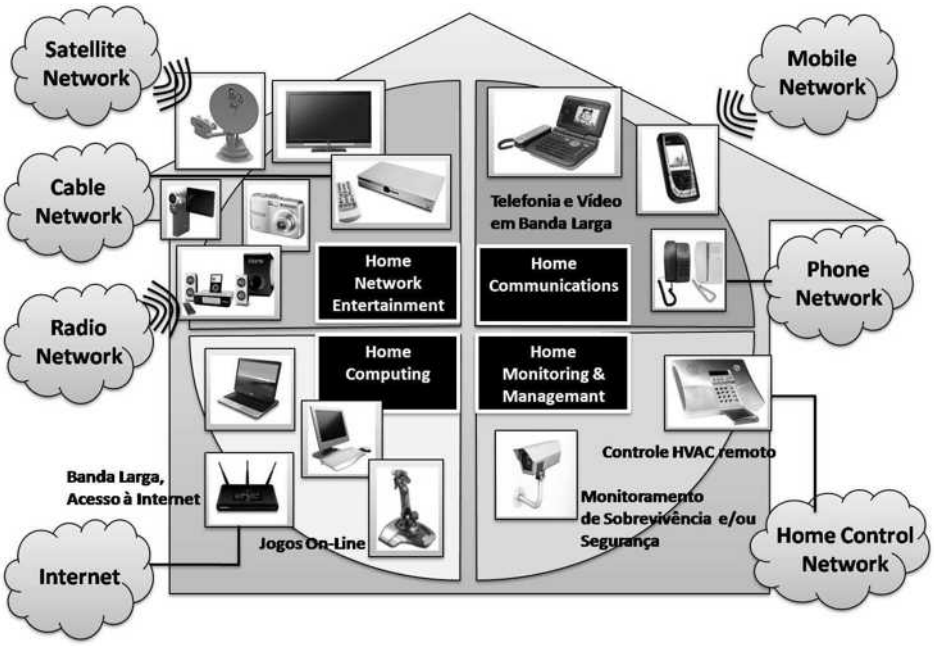


Figura 2.3: As quatro principais categorias das *Home Networks*
FONTE: Adaptado de Alam e Prasad (2008); Dixit e Prasad (2008).

A principal função de *Home Computing* (ou *Personal Computing*) está em conectar redes internas (*indoor networks*) e externas (*wide area network*) bem como interconectar múltiplos dispositivos e prover compartilhamento de serviços entre eles no

ambiente doméstico. *Home Network Entertainment* engloba dispositivos de consumo da rede que promovam serviços de entretenimento, principalmente associados ao uso de mídias (áudio/vídeo), tais como: Xbox, PS3, DVR, TV, *Home Theater*, Câmera Fotográfica, entre outros. *Home communications* inclui dispositivos de telefonia por voz/vídeo, *Wi-fi* fones, *dual-mode* fones, entre outros. Por último, *Home Monitoring and Management* inclui dispositivos de controle remoto sem fio (como sistemas HVAC), controle remoto de pacientes (*remote surveillance*), sistemas de gerenciamento de residência (alarmes, câmeras), etc.

Home Computing

Segundo Alam e Prasad (2008), são seis os requisitos básicos em uma *Home Network* sob a perspectiva de *Home Computing*: (1) Interconectar múltiplos dispositivos na residência e prover largura de banda para distribuir serviços entre eles; (2) Interconexão entre o Gateway Doméstico e WANs; (3) Comunicação segura no ambiente doméstico e fora dele (com as WANs); (4) Melhorias de serviços sem fio em velocidade e faixa de transmissão; (5) Suporte a distribuição de mídias em redes com ou sem fio; e (6) Compartilhamento de arquivos.

Dentre as novas tecnologias para redes de alta velocidade em ambientes domésticos (garantindo requisitos de segurança e/ou privacidade) que têm surgido, destacam-se:

- Redes *wireless* 100Mbps (IEEE 802.11n). Redes de alta banda *wireless* possibilitam ao ambiente doméstico a distribuição de grande volume de dados como vídeo em alta definição.
- Tecnologias sem novos fios (no new wires) tais como:
  - Redes de alta velocidade sobre tecnologias de cabo coaxial (MoCA).
  - Redes de alta velocidade sobre tecnologias de linhas de telefone (HomePNA).
  - Redes de alta velocidade sobre utilizando o suporte da rede elétrica (HomePlug AV). HomePlug AV dá suporte à aplicações de entretenimento na rede

doméstica: dispositivos para redes HomePlug AV permitem a provedores de serviços e consumidores a distribuição de áudio/vídeo em alta definição usando a rede elétrica residencial.

Algumas das principais tecnologias que estão migrando para o ambiente doméstico, suas respectivas taxas de transmissão e fabricantes são sistematizados na Tabela 2.1.

Meio Físico	**Taxa de Transferência**	**Consórcio Associado**
Infravermelho	9600 bps – 115 Kbps; até 4Mbps	*The Infrared Data Association* (IrDA) `http://www.irda.org`
Linha Telefônica	1 – 10Mbps	*Home Phoneline Network Alliance* (HomePNA) `http://www.homepna.org`
Freqüência de Rádio	1 – 2Mbps	*Home Radio Frequency Working Group* (HomeRF) `http://www.homerf.org`
Rede Elétrica	até 14Mbps	*Home-Plug Powerline Alliance* (HomePlug) `http://www.homeplug.org`
Rede de TV à Cabo	27Mbps ou 500 Kbps – 10Mbps	*Cable Television Laboratories, Inc.* (CableLabs) `http://www.cablelabs.com`

Tabela 2.1: Meios físicos para transmissão de dados em *Home Networks*
FONTE: Adaptado de Gong (2001a).

Entre as iniciativas em redes de alta velocidade para o ambiente doméstico destaca-se o *Digital Living Network Alliance* (DLNA) (DLNA, 2009), que engloba empresas de dispositivos eletrônicos, PCs, e dispositivos móveis, entre outras. O objetivo é permitir a consumidores que obtenham, armazenem, acessem e compartilhem conteúdo digital de qualquer lugar dentro ou fora da residência. Isso permite a usuários despender menos esforço em gerenciar, visualizar, imprimir e/ou compartilhar seu conteúdo. Por exemplo, em uma rede doméstica DLNA, é possível acessar um vídeo de um DVR (*Digital Video Recording*) e assisti-lo em um PC, ou celular, de qualquer lugar da residência.

Home Entertainment

Geralmente um dispositivo que provê serviços de entretenimento doméstico está associado à dispositivos das outras três áreas de *Home Networks*. Estes tipos de dispositivos são chamados *consumer place-shifting devices* (Dixit e Prasad, 2008): dispositivos ou software que usam serviços de outros próximos para prover serviços aos consumidores. Como exemplo, a *Sling Media* (Sling, 2009) oferece um dispositivo de consumo que, localizado próximo ao Set-top Box, permite a usuários visualizar seus programas de TV via Internet em um PC ou dispositivo móvel.

Um dos principais desafios de *Home Entertainment* refere-se ao mercado de mídias para armazenamento. Este se encontra fragmentado, havendo uma baixa demanda por esse serviço em redes domésticas (Alam e Prasad, 2008). Consumidores armazenam arquivos e mídias digitais em discos de seus PC/laptop, ou discos externos, USB, e/ou CDs/DVDs. Conteúdo de TV é armazenado em DVRs e/ou PC com placa de TV. O mercado para serviços/dispositivos de armazenamento de conteúdo em redes domésticas está ainda em desenvolvimento, mas com crescente demanda dos consumidores para compartilhamento de arquivos, conteúdo de vídeo pessoal, transferências de arquivos P2P, transferência de áudio/vídeo, IPTV (*Internet Protocol Television*), entre outras.

Home Communications

A comunicação no ambiente doméstico pode ocorrer de três formas:

- **Comunicação integrada entre fixo e móvel** (do inglês, FMC, *Fixed-Mobile Communication*). Como exemplo pode-se citar a iniciativa FMCA (*Fixed-Mobile Convergence Alliance*) (FCMA, 2009), um consórcio global entre operadores de telecomunicações cujo objetivo é acelerar o desenvolvimento de produtos e serviços de convergência.

- **Comunicação com grupos de usuários de outras redes domésticas**. Também chamadas *User Personal Network* (UPN), o suporte para esse tipo de serviço permitirá que usuários adicionem/removam outros usuários de suas redes. Usuários

pertencentes a uma UPN podem compartilhar qualquer forma de conteúdo tais como voz, vídeo, IM (*Instant Messaging*) com outros na mesma rede.

- **Sistemas de Presença integrada a Comunicação via Voz**. Sistemas de comunicação por voz integrados com aplicações de presença em tempo real permitirão a usuários saber quem está conectado na rede e em que instante, e se a pessoa do outro lado do *link* de comunicação está recebendo uma chamada. Isto reduz o número de chamadas desnecessárias e melhora a interação entre os usuários. Este serviço é comumente disponibilizado por usuários de empresas. Entretanto espera-se que ele entre no mercado doméstico tão logo serviços de VoIP também adentrem esse mercado (Dixit e Prasad, 2008).

Para prover fluxo de conteúdo livre de meios físicos (cabos) entre dispositivos, redes com conectividade *wireless* aparecem como a nova área de crescimento em pesquisa e oportunidades de negócios (Dixit e Prasad, 2008). Isso permite o controle, acesso e compartilhamento de informações entre todos os dispositivos, contribuindo para a melhoria da experiência do usuário e qualidade de vida no ambiente doméstico conectado.

Home Monitoring and Management

Serviços de monitoramento e gerenciamento local ou remoto de residências são as aplicações que melhor representam o processo de automação de tarefas no ambiente doméstico. *Home Monitoring and Management* permite o controle remoto de sistemas de iluminação, ventilação e segurança, por meio de alertas e relatórios sobre condições do ambiente através de uma variedade de sensores de movimento, atuadores, sensores de temperatura, etc. Esse segmento da *Home Network* divide-se ainda em três subcategorias:

- **Monitoramento Remoto de Sobrevivência** (*Remote Surveillance*): útil principalmente para monitoramento de atividades de pessoas dependentes como crianças e idosos. Inclui sistemas de alerta e emergência, por exemplo, acionar um alarme quando a atividade de um idoso não mudar em um determinado intervalo de tempo.

- **Sistemas de Gerenciamento de Residência**: inclui sistemas de segurança, câmeras e sistemas de detecção de intrusos.

- **Sistemas HVAC** (*Heating, Ventilation Air-conditioning control and lighting Control*): inclui sistemas de ventilação, controle de temperatura, controle de incêndio, gás, iluminação, etc.

Para oferecer um ambiente apropriado para o provimento de serviços complexos a redes domésticas, é necessário uma plataforma com características específicas. O *Home Gateway*, ou Gateway Residencial é a plataforma adequada para isso e será brevemente discutido a seguir.

2.2.2 Caracterização de Home Gateways

O termo *Home Gateway*[2] é usado para designar o dispositivo eletrônico que centraliza no ambiente doméstico interconectado serviços providos por todos os dispositivos da rede. O *Home Gateway* é também o elemento da arquitetura de *Home Networks* (apresentada na Figura 2.2) que habilita a conexão da rede com as redes externas dos provedores de serviços, outras Home Networks e a Internet (Gong, 2001a). Ao mesmo tempo, esses aparelhos servem como plataformas para entrega de serviços e execução de aplicações associadas aos dispositivos da rede doméstica (Zhang e Zheng, 2009).

A complexidade de serviços entregues à *Home Network* e a heterogeneidade de dispositivos presentes na rede exigem que Home Gateways agreguem muitas interfaces de comunicação (Serial RS-232, Porta Paralela, USB, IDE, SCSCI, IEEE 1394 *Firewire*) e funções (compartilhamento de mídia, VoIP, HDTV, VoD, etc), o que acaba tornando difícil a existência de um equipamento genérico. Gupta (2009) sistematiza quatro tipos principais de Home Gateway: (1) *Modems de Banda Larga* (modem a cabo, DSL - *Digital Subscriber Line*, modem wireless); (2) *Gateways de Automação*, que incluem funções de controle da residência (por exemplo, Gateway de sistemas HVAC); (3) *Gateways de*

[2]Nesta Dissertação será usado o termo *Home Gateway* (em vez de *Service Gateway* ou Gateway Residencial ou ainda Gateway Doméstico) para referenciar o dispositivo que interconecta todos os demais em uma *Home Network.*

Segurança e (4) *Digital STB Home Gateway*, que estende as funções normais do STB convencional (HDTV, PVR, IPTV, EPG, etc) integrando-as com outros dispositivos da rede. Sobre esta última categoria é realizado este trabalho de Dissertação. Mais adiante serão discutidas as características que apontam os STBs como apropriados para atuarem como *Home Gateways* (Capítulo 4).

Do ponto de vista da arquitetura do dispositivo, o *Home Gateway* apresenta componentes de *hardware* (processador, memória, armazenamento flash/RAM, interfaces para conexão com dispositivos e redes de comunicação como TCP/IP), *um sistema operacional de tempo real* e *uma plataforma para gerência de serviços e aplicações*, semelhante ao *middleware* de TV Digital.

2.2.3 Aplicações e Serviços em Home Network

A camada de software da arquitetura de *Home Gateways* agrega componentes que oferecem aos aplicativos interfaces sobre as funções de baixo nível do sistema. Os chamados *frameworks* ou protocolos para *Home Network* definem padrões para acesso a funções e serviços dos dispositivos da rede. Da mesma forma que em outros segmentos do modelo de *Home Network* existem soluções concorrentes de *frameworks* tanto de propósito específico como protocolos LonWorks (LonWorks, 2009), CEBus (CEBus, 2009), X10 (X10, 2009) e *Bluetooth*, como padrões de funções gerais, destacando-se as plataformas HAVi (*Home Audio Video Interoperability*) (Havi, 2009), Jini (Jini, 2009b), UPnP (*Universal Plug and Play*) (UPNPForum, 2009) e OSGi (*Open Service Gateway Initiative*) (OSGi-Alliance, 2009).

Para os propósitos deste estudo são considerados os quatro últimos *frameworks* apresentados. Seus modelos de execução serão detalhadamente analisados e confrontados no sentido de escolher o que apresenta as melhores características para integração com *middlewares* de TV Digital, sobre tudo o *middleware* brasileiro. Isso fornecerá a base para a construção de um modelo em software que permita verificar a terceira categoria de *Home Gateways* apresentada neste tópico (*Digital Set-Top Box Home Gateway*).

2.3 Resumo

Neste capítulo foi apresentado o estado da arte sobre tecnologias para sistemas de TV Digital e *Home Networks*, como base essencial para o desenvolvimento dos capítulos posteriores.

Inicialmente, foram abordados conceitos sobre tecnologias para modulação, multiplexação, *middleware* e componentes de Sistemas de TVDI (difusor de conteúdo, meios de difusão e receptor), citando alguns dos principais padrões existentes na atualidade, com destaque para o Sistema Brasileiro de TV Digital (SBTVD). Também foram apresentados os principais cenários de aplicações interativas e discutida a possibilidade de extensão desses cenários para o provimento de novos serviços, tais como os que integram a TV Digital com outros dispositivos eletrônicos em uma *Home Network*.

Na perspectiva de *Home Network*, foram abordadas algumas das tendências atuais de convergência entre serviços, provedores e aplicações no ambiente doméstico, bem como as dificuldades de integração entre as diversas tecnologias presentes neste ambiente. Foram apresentados os componentes de um sistema de *Home Network*, seguido de um critério para a classificação dessas redes em quatro categorias: *Home Computing*, que agrega dispositivos que integram os demais na mesma rede ou em redes externas; *Home Entertainment*, composto por dispositivos que oferecem entretenimento ao usuário; *Home Communications*, que inclui dispositivos de telefonia, voz/vídeo; e *Home Monitoring and Management*, que inclui dispositivos de monitoramento e gerenciamento de informação. Por fim, foram caracterizados os dispositivos centrais de *Home Networks*, os *Home Gateways*, bem como citadas as principais aplicações e serviços que podem ser oferecidos por esta plataforma para o ambiente doméstico.

A partir deste ponto, o trabalho se concentrará nas camadas de software tanto da arquitetura de TV Digital (*middleware* no STB) como de *Home Networks* (*framework* no *Home Gateway*), a fim de definir tecnologias usadas na construção de uma plataforma-base para aplicações no ambiente integrado *STB-HG*. No próximo capítulo serão feitas análises dos *middlewares* para TV Digital e *frameworks* para *Home Networks*.

Capítulo 3 - Definição das Plataformas de Software

Neste capítulo serão abordadas as principais plataformas de software para ambientes de TVDI (*middleware* no STB) e *Home Network* (*framework* no *Home Gateway*). Para cada domínio, serão analisadas as características, funcionalidades e limitações das plataformas de software a fim de escolher a mais adequada para a construção do modelo de colaboração TVDI-HG descrito nesta Dissertação.

Considerando a TV Digital, o *middleware* é apresentado como a plataforma para execução de aplicativos interativos. Arquitetura e funções dos principais *middlewares* (MHP, DASE, ARIB, Ginga; e APIs relacionadas, javaTV, HAVi, DAVIC e javaDTV) serão discutidos, bem como aspectos importantes para a integração dos aplicativos de TVDI com os componentes de software do *framework* para HN, tais como *gerenciamento do ciclo de vida de aplicações* e *comunicação entre aplicações*. Ao final são apresentadas as razões que motivaram a escolha do *middleware* Ginga para a construção do modelo de colaboração com o *framework* para HN. Esse modelo deve explorar as potencialidades do Ginga, sejam funções comuns ao Ginga-J e MHP/GEM (relativo à parte procedural do *middleware*) ou funcionalidades baseadas na interpretação de documentos NCL no Ginga-NCL (relativo à parte declarativa do *middleware*).

Sob o ponto de vista de software para *Home Networks*, serão discutidos conceitos sobre os principais padrões para a construção de aplicações e gerenciamento de serviços no ambiente doméstico: modelo, arquitetura e funções dos principais *frameworks* (UPnP, OSGi, Jini, HAVi) são confrontados com o intuito de escolher o mais apropriado para integração com o *middleware* de TV Digital. A plataforma definida pelo *Open Services Gateway Initiative* (OSGi) é discutida em detalhes, sob aspectos do modelo orientado a

serviços, modularização e componentização de software. Os principais elementos do OSGi que servirão como base para a integração com componentes do *middleware* Ginga/MHP são abordados (conceitos como unidades de aplicativos *bundles*, e *gerenciamento de serviços e dependências*).

Ao final do capítulo espera-se identificar o *middleware* e o *framework* para o modelo TVDI-HN na TVDI brasileira, bem como destacar os componentes de software das duas especificações que deverão ser posteriormente modificados e integrados de modo a garantir a implementação do modelo.

3.1 Escolha da Plataforma para TVDI

Conforme apresentado no capítulo anterior, em TV Digital a principal estrutura responsável por prover um ambiente para a execução de aplicações interativas e entrega de serviços aos terminais de TVDI é o *middleware*. Issarny et al. (2007) definem *middleware* como uma camada de software entre o sistema operacional de rede e aplicações, que abstrai do desenvolvedor de software especificidades da plataforma e oferece soluções de reuso de software, principalmente para problemas de heterogeneidade, interoperabilidade e segurança.

De uma maneira geral, a definição de *middleware* está associada a sistemas distribuídos; trata-se de uma estrutura que permite aplicações distribuídas comunicarem-se de maneira transparente e usarem funções da plataforma de maneira independente do hardware, sistema operacional, interfaces de rede e armazenamento. Um *middleware* fornece entidades ou componentes de software reusáveis que permitem a construção de aplicações interoperáveis entre plataformas heterogêneas (Schantz e Schmidt, 2002; Schmidt e Buschmann, 2003).

3.1.1 Tecnologias em *Middleware* para TVDI

A evolução das tecnologias em *middleware* para TV Digital foi ditada pela crescente necessidade de harmonização entre os padrões e tecnologias existentes. O cenário antigo do mercado em *middleware* é caracterizado por um conjunto de tecnologias

fechadas, num mercado vertical dominado por poucas empresas. A não existência de padrões abertos dificultava o crescimento dos serviços para a TV Digital e o avanço da tecnologia em geral. Entre os principais *middlewares* proprietários podem ser citados *Microsoft TV*, *Galio*, *Myrio*, *MediaHighWay, NDS*, *Imagenio*, *OpenTV* e *PowerTV* (Morris e Smith-Chaigneau, 2004).

Com o surgimento de padrões abertos (MHP, OCAP, ACAP, DASE, ARIB, GINGA) muitas tecnologias já existentes puderam ser incorporadas às especificações de *middleware*, através de um conjunto de API's que tratam de funções específicas do STB. Como exemplos podem ser citados os consórcios HAVi (*Home Audio Video Interoperability*), DAVIC (*Digital Audio Video Council*) e JavaTV, tecnologias XML, HTML, ECMAScript, JavaScript, NCL, SMIL, LUA, etc. A Figura 3.1 ilustra a convergência dos *middlewares* de um mercado fechado e vertical para um aberto, harmonizado e horizontal. Conforme a figura a tendência de harmonização é crescente; os esforços das principais entidades que definem padrões de *middleware* dos sistemas de TV Digital culminaram com a especificação de um conjunto de componentes comum a todos os padrões, definido como GEM (*Globally Exacutable MHP*).

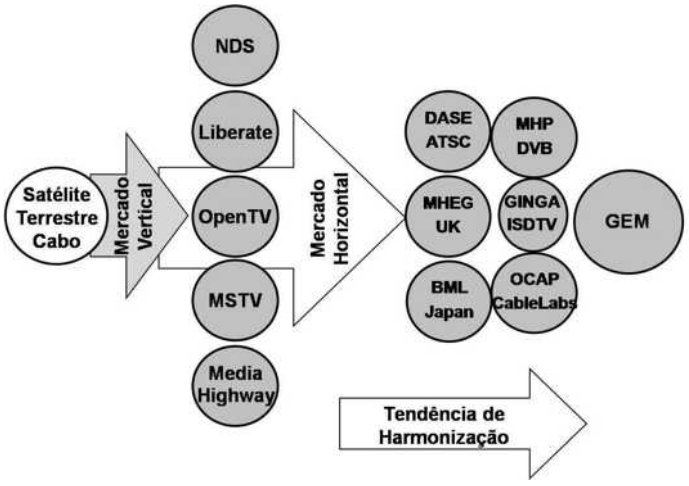


Figura 3.1: Evolução dos padrões de *middleware*.
FONTE: Adaptado de (Morris e Smith-Chaigneau, 2004).

As aplicações interativas utilizam API's do *middleware* para acesso a funcionalidades gerais da plataforma, como sintonização de canais, tratamento de áudio e vídeo, persistência de dados, gerenciamento do ciclo de vida de aplicações, gerenciamento

de componentes gráficos, funções do canal de interatividade (canal de retorno), entre outras. Os recursos acessados pelas aplicações são mapeados por componentes do *middleware* da plataforma. As funções e recursos do STB que devem ser abstraídos pelo *middleware* podem ser organizados em quatro grupos (Schwalb, 2004), ilustrados na Figura 3.2: (1) controle de mídias; (2) controle do fluxo de transporte (MPEG2-TS), (3) interface gráfica com o usuário e (4) canal de retorno.

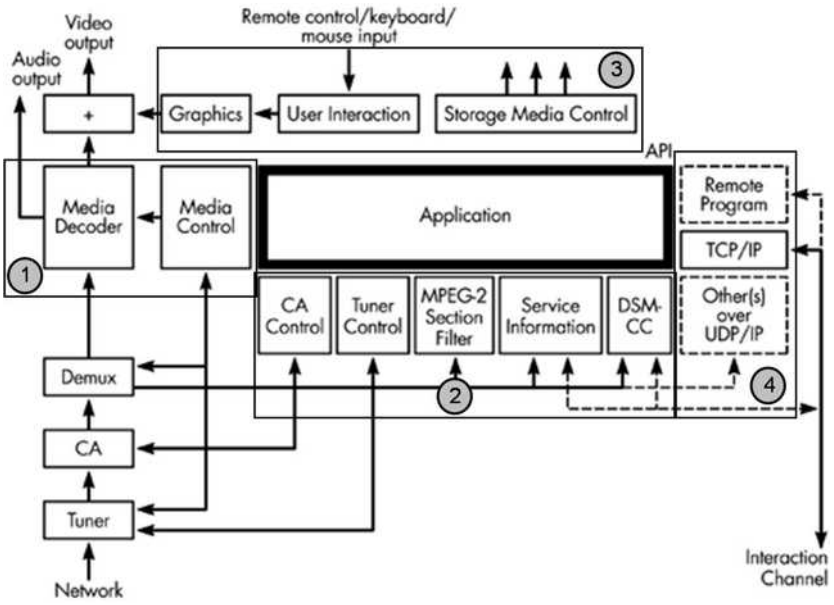


Figura 3.2: Funções de interface entre aplicações e a plataforma (STB).
FONTE: Adaptado de (Schwalb, 2004).

3.1.2 Paradigmas de Desenvolvimento

As tecnologias que agregam *middlewares* em TVDI podem ser agrupadas em dois grandes grupos, diferenciados principalmente pelo tipo de suporte dado ao desenvolvimento de aplicações para a plataforma. Dessa forma, os *middlewares* para TVDI dos padrões existentes apresentam ambientes para programação procedural e declarativa de aplicativos. Na primeira, os *middlewares* oferecem máquinas de execução que interpretam código escrito em alguma linguagem de programação (por exemplo, Java). Nesses ambientes, geralmente um miniaplicativo controlado por uma máquina de estados executa sobre o *middleware*, comunicando-se com um gerenciador de aplicações, utilizando e liberando recursos do sistema. Na segunda forma uma *máquina de apresentação*

trata de interpretar um documento escrito em algum padrão de gramática, criando uma estrutura em árvore correspondente, a partir da qual realiza o controle sobre o aplicativo (Soares et al., 2006a).

Aplicações que executam sobre *middleware* procedural utilizam linguagem de programação cujo estilo decompõe o problema numa implementação algorítmica. Embora possuam um nível de abstração menor que aplicações declarativas, apresentam vantagens como o reuso de componentes ou a possibilidade de criação dos próprios componentes para executar tarefas específicas. Como exemplos de *middleware* procedural podem ser citados: ARIB (B.23), OCAP (OCAP-J), ACAP (ACAP-J), DASE, MHP (DVB-J), GEM e GINGA-J.

As normas ITU (*International Telecommunication Union*) J.200 (*Worldwide Common Core - Application Environment for Digital Interactive Television Services*) e J.202 (*Harmonization of Procedural Content Format for Interactive TV Applications*) (ITU, 2009) definem algumas das funções para ambientes procedurais que devem ser implementados nos *middlewares* de TV Digital. Basicamente, há um conjunto de componentes derivados da tecnologia Java que oferecem funcionalidades básicas. Além destes, há o suporte para apresentação de mídia com o *Java Media Framework* (JMF); controle de áudio/vídeo com API's HAVi, DAViC; acesso a informações de serviços e canal de transmissão com JAVATV. A Figura 3.3 ilustra a arquitetura definida para ambientes procedurais pela ITU J.200 e J.202.

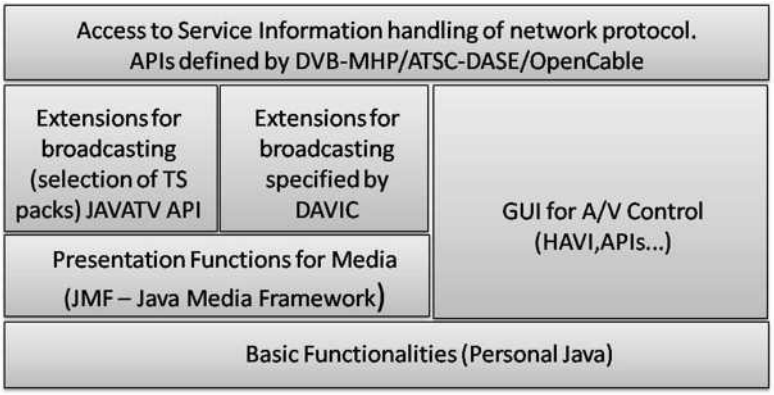


Figura 3.3: Norma ITU J.202 para *middleware* procedural.

FONTE: Adaptado de ITU (2009).

As três últimas tecnologias citadas, HAVi, DAVIC e JAVATV, compõem a maioria dos *middlewares* procedurais existentes. Foram desenvolvidas independentes dos padrões de *middleware*, mas acabaram sendo posteriormente incorporadas a eles. Essas APIs são apresentadas a seguir:

- *Home Audio Video Interoperability* (HAVi) - Padrão definido por um consórcio formado por companhias de produtos de consumo audiovisuais que especifica uma forma comum para conexão e interoperação de dispositivos de áudio e vídeo digital. A especificação permite que dispositivos de áudio e vídeo interajam entre si, definindo mecanismos que permitem que as funcionalidades de um dispositivo possam ser remotamente usadas e controladas por outro dispositivo (Havi, 2009). Para a TVDI, esse padrão foi adotado por prover um conjunto de APIs com suporte específicos para TV tais como funções para lidar com gráficos e construção de interface com o usuário. O *HAVi Level 2UI* (HAVi nível 2, user interface), por exemplo, é um subconjunto do HAVi que possui funcionalidades extras que tornam possível o desenvolvimento de interfaces com o usuário com componentes específicos para o ambiente de TV Digital. Além dos recursos citados, o HAVi define ainda uma arquitetura de software distribuída cujos elementos asseguram interoperabilidade e implementam serviços de gerenciamento de rede e comunicação entre dispositivos, conforme será discutido no tópico 3.2.3.

- *Digital Audio Visual Council* (DAVIC) - O DAVIC é uma associação de diversos setores da indústria audiovisual, agregando companhias entre fabricantes de produtos eletrônicos e de telecomunicações, emissoras, agências de governo e organizações de pesquisa em mais de 25 países (Davic, 2009). A idéia por traz do consórcio é de fornecer interoperabilidade, com ênfase nas informações audiovisuais digitais interativas. Para obter essa interoperabilidade, as especificações DAVIC definem interfaces em diversos pontos de referência onde cada uma é caracterizada como um conjunto de fluxos de informações e protocolos, que tratam de padrões específicos sobre disposição de informações. Assim, as especificações DAVIC dispõem sobre formatos de conteúdo para diversos tipos de objetos (fonte, texto,

hipertexto, áudio e vídeo), incluindo também facilidades para o controle de áudio e legenda.

- *JavaTV* - É uma extensão da plataforma Java que fornece funcionalidades aos desenvolvedores para a produção de conteúdo para TVDI (Sun, 2009a). O principal objetivo da tecnologia é viabilizar o desenvolvimento de aplicações interativas portáveis, que são independentes da tecnologia da rede de difusão. A API provê uma abstração que permite aos aplicativos escritos em Java (denominados *Xlets*, descritos na próxima subseção) e interpretados na máquina virtual (JVM - *Java Virtual Machine*) dentro da arquitetura de um receptor de TV Digital, obter informações sobre determinado serviço de uma forma independente do protocolo utilizado. Dessa forma uma aplicação pode ser reusada em uma variedade de ambientes. Embora desenvolvida para ser utilizada em arquiteturas específicas para STBs, a API pode ser utilizada em ambiente computacional convencional (Calder et al., 2000), tanto que a maioria dos emuladores atuais de TVDI dispõem de recursos do javaTV. A implementação de referência javaTV consiste, em essência, de um conjunto de grupos funcionais ou *packages* (pacotes) Java, a maioria sub-pacotes de *javax.tv*.

No que se refere a *middleware* declarativo, aplicações para esses ambientes utilizam linguagens e estruturas (geralmente baseadas em XML) que enfatizam uma descrição declarativa do problema. Embora apresentem um maior nível de abstração, são projetadas para um domínio específico. Em TVDI, linguagens declarativas são utilizadas principalmente para criar apresentações multimídia com separação entre conteúdo e estrutura da apresentação e sincronização temporal de mídias (vídeo, áudio, imagens, texto, etc.) (Barbosa e Soares, 2008). Como exemplos de *middleware* declarativos podem ser citados DVB-HTML, Ginga-NCL, ISDB-BML, OCAP-X, DASE-HTML.

As normas ITU J.200 e J.201 (*Harmonization of Declarative Content Format for Interactive TV Applications*) definem algumas das funções para ambientes declarativos que devem ser implementados nos *middleware* de TV Digital. Conforme a Figura 3.4 o *middleware* declarativo possui suporte à conteúdo ECMAScript e XHTML (*Extensible HiperText Markup Language*), através de máquinas apropriadas, além de suporte à

formatação com CSS (*Cascading Style Sheets*), extensão para XML (XML Parser) e um *browser* sobre o qual o conteúdo é executado. O *browser* permite ainda acesso às funcionalidades do receptor (mídias, interfaces de rede, canal de retorno, etc.).

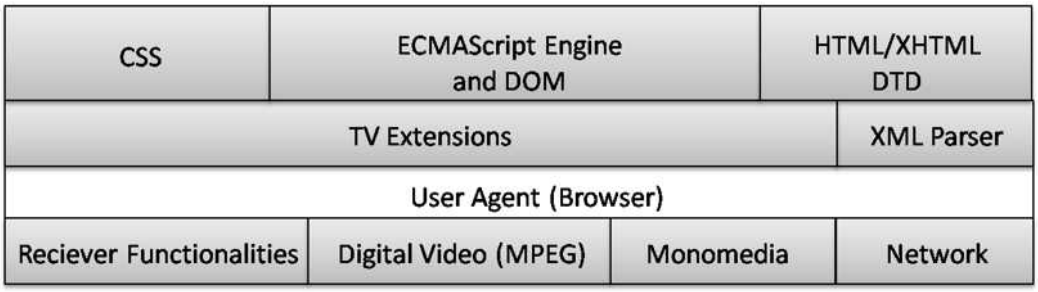


Figura 3.4: Norma ITU J.201 para *middleware* declarativo.
FONTE: Adaptado de ITU (2009).

Todos os padrões abertos de TV Digital oferecem suporte à *middleware* procedural e declarativo compatíveis com as normas ITU. Entretanto, para cada um dos ambientes, os *middlewares* utilizam além das tecnologias padrão (HAVi, DAVIC, JavaTV, XHTML, XML), tecnologias específicas criadas para atender às necessidades de cada *middleware*. Assim, o DVB-MHP definiu o DVB-HTML declarativo, o ARIB definiu o ARIB-BML (*Broadcast Markup Language*) e o ISDTV definiu o Ginga-NCL, com base no modelo NCM (*Nested Context Model*) discutido adiante. É interessante ressaltar que as normas ITU ainda especificam um mecanismo de interação entre os dois ambientes, através de um componentes-ponte que permite a troca de recursos entre a parte procedural e a parte declarativa dos *middlewares*. A Tabela 3.1 resume os ambientes de aplicações para receptores fixos e móveis dos padrões de TV Digital.

3.1.3 *Middleware* MHP e convergência para o GEM

O MHP é um padrão aberto adotado para a entrega de aplicações e serviços aos terminais (STBs) da rede de difusão do sistema DVB. O *middleware* é composto por API's e tecnologias associadas dentre as citadas anteriormente, além de um conjunto de testes de conformidade definidos pelas normas do DVB e do *European Telecommunications Standards Institute* (ETSI) (ETSI, 2009b). O MHP é um padrão consolidado; é derivado

Middleware	Sistema de TVDI	Ambiente Declarativo	Ambiente Procedural
ACAP	Americano ATSC	ACAP-X [ATSC A-101 2005] (linguagem declarativa = XHTML like; linguagem não-declarativa = ECMAScript)	ACAP-J [ATSC A-101 2005](linguagem não declarativa =Java)
MHP	Europeu DVB/T	DVB-HTML [ETSI TS 102 812 v 1.2.2 2006] (linguagem declarativa = XHTML like; linguagem não-declarativa = ECMAScript)	MHP [ETSI TS 102 812 v1.2.2 2006] (linguagem não declarativa =Java)
ARIB-BML	Japonês ISDB/T	ARIB-BML [ARIB B-24 2004] (linguagem declarativa = BML (XHTML like); linguagem não-declarativa = ECMAScript)	Opcional (GEM [ETSI TS 102 819 v1.3.1 2005] like; não implementado
GINGA	Brasileiro SBTVD	Ginga-NCL [ABNT NBR 15606-2 2007] (linguagem declarativa = NCL; linguagem não-declarativa = Lua)	Ginga-J (linguagem não declarativa = Java)

Tabela 3.1: Ambientes declarativo e procedural dos *middlewares* de TV Digital.
FONTE: Adaptado de Barbosa e Soares (2008).

do antigo *Multimedia and Hypermedia Experts Group* (MHEG) baseado em Java, que incorporou componentes HAVi, DAVIC e JavaTV.

Ao longo de sua evolução, as funcionalidades do MHP foram separadas, para cada versão do *middleware*, em conjuntos denominados *MHP Profiles*. Cada *profile* agrega tecnologias adicionais em relação ao anterior. Os *MHP profiles* foram definidos ao longo das versões MHP1.0.x (a especificação original com atualizações, versão mais comumente usada do MHP) e MHP 1.1.x (versão anterior com suporte à HTML, aplicações residentes, API's de acesso à Internet e canal de retorno) (Morris e Smith-Chaigneau, 2004; Piesing, 2006). São os seguintes *MHP profiles*, sistematizados na Figura 3.5:

- *Transmissão Melhorada*: Perfil que combina a transmissão de áudio e vídeo e serviços de *download* de aplicações que permitem serviços de interação local (não suporta canal de retorno).

- *Transmissão Interativa*: Perfil que contém as funcionalidades do perfil de transmissão melhorada e permite diferentes formas de interação global com ou sem associação com serviços de difusão (Por exemplo, suporte ao IPTV).

- *Acesso à Internet*: Além de conter as funcionalidades dos perfis anteriores, permite

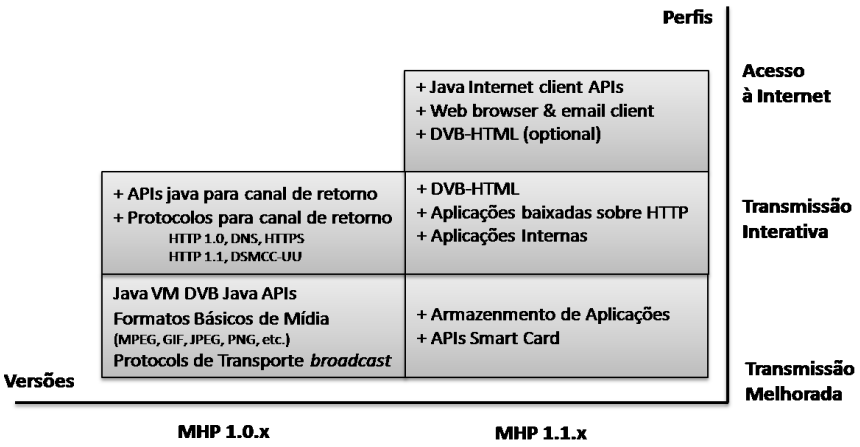


Figura 3.5: MHP profiles.

FONTE: Adaptado de Morris e Smith-Chaigneau (2004).

a interação entre serviços de Internet e difusão. Exemplos de aplicações desta configuração incluem *Browser* para E-mail e API Java para acesso a Internet.

Além do MHP 1.0.x e 1.1.x um outro componente especificado é o GEM. Definido pela norma ETSI TS 102 819 (ETSI, 2009a) como subconjunto do MHP 1.0.2, apresenta uma API estável e genérica como base de referência para interoperabilidade entre padrões. Aplicações GEM-compatíveis podem executar em qualquer *middleware* que agregue o subconjunto GEM. O GEM define um conjunto de funcionalidades MHP que são tecnologicamente neutras, ou seja, remove qualquer API que seja específica do MHP, adicionando mecanismos de substituição para adequar-se às outras tecnologias.

Modelo de Gerenciamento de Ciclo de Vida

Uma das funções mais importantes em um *middleware* de TV Digital é o gerenciamento de aplicativos e recursos da plataforma, além da possibilidade de intercomunicação entre aplicativos em plataformas distribuídas (por exemplo, em serviços interativos de *chat* pela TV). Os aplicativos MHP e de outros *middleware* em geral têm um rigoroso mecanismo para gerenciar o ciclo de vida das aplicações e a liberação para uso de recursos da plataforma. O principal modelo adotado para gerenciamento de aplicações é o definido pelo JavaTV, denominado *Xlet Lifecycle Model* (Calder et al., 2000), no qual figuram três elementos:

- *Xlet* - Miniaplicativo Java, tem o ciclo de vida caracterizado como máquina de estados. Um *Xlet* possui os estados de *Carregado* (pronto para ser ativado), *Ativo* (em execução), *Pausado* (em espera) ou *Destruído* (eliminado da plataforma).

- *Application Manager* - Gerenciador de *Xlets*. Entidade que controla os recursos do ambiente de software do receptor, bem como aplicações que nele executam.

- *XletContext* - Componente através do qual *Xlet* e *Application Manager* interagem.

Para cada miniaplicativo *Xlet* no receptor, um ciclo de vida é definido desde o momento em que ele é criado até quando for destruído. O *Xlet* atua no ambiente executando sucessivas mudanças de estado, controladas pelo *Application Manager* por meio do *XletContext*. Sobre o *middleware*, uma típica seqüência de execução de um *Xlet*, em interação com o gerenciador, é caracterizada pelos seguintes passos descritos na Tabela 3.2 e ilustrados na Figura 3.6:

Application Manager	***Xlet***
O *Application Manager*, ao receber um fluxo de dados e decodificá-lo, identifica um possível aplicativo que irá executar e cria uma nova instância de um *xlet*.	Um construtor *default* é chamado para o *xlet* que entra em estado de *loaded*.
O *Application Manager* obtém o contexto necessário para a inicialização do *xlet* através da criação de um objeto *XletContext*.	O *xlet* usa o objeto *XletContext* para inicializar a si próprio, entrando no estado de *paused*.
O *Application Manager* decide se é o momento oportuno para o *xlet* realizar seu serviço, podendo notificá-lo a entrar no estado de *active*.	O *xlet* adquire os recursos necessários a sua execução e e começa a realizar seu serviço.
O *Application Manager* não necessita mais da realização do serviço pelo *xlet* sinalizando-o a interromper sua execução.	O *xlet* interrompe a execução de seu serviço liberando os recursos utilizados até o momento entrando em estado de *paused* e aguardando o próximo comando do gerenciador.
O *Application Manager* decide que não é mais necessária a execução do *xlet*. Então o gerenciador sinaliza ao *xlet* que este deve ser destruído.	O *xlet* prepara-se para ser finalizado, salvando as configurações atuais como preferências do usuário. O *xlet* entra no estado de *destroyed* e deixa de existir no ambiente de execução.

Tabela 3.2: Etapas da Interação entre *Xlet* e *Application Manager*.

FONTE: Adaptado de Calder et al. (2000).

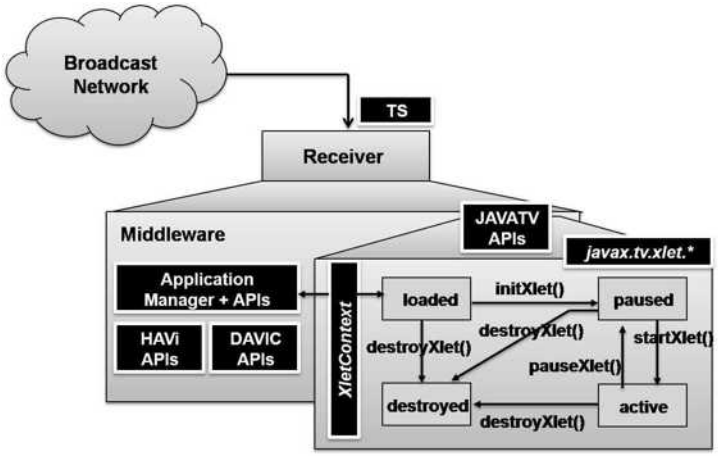


Figura 3.6: Estados do *Xlet* e interação com o *Application Manager*.
FONTE: Adaptado de Morris e Smith-Chaigneau (2004).

Modelo de Comunicação entre aplicativos

Outra característica importante do ambiente de aplicações para TVDI é o suporte para a comunicação entre aplicativos. O *Inter-Xlet Communication* (IXC) é um mecanismo de interação entre aplicações de TVDI que tem como base o compartilhamento de objetos relacionados a funcionalidades providas por *Xlets*, registrados numa entidade pública, visíveis e acessíveis por outros *Xlets*. O modelo do IXC baseia-se no *Remote Method Invocation* (RMI), sistema de comunicação distribuída baseado em Java que permite a aplicativos acessarem funcionalidades presentes em JVMs remotas.

O IXC existe principalmente para garantir segurança no acesso aos dados compartilhados por *Xlets* diferentes (Morris e Smith-Chaigneau, 2004). MHP e OCAP permitem a *Xlets* registrar funcionalidades (através de uma operação BIND) em uma entidade padrão, o *IXC Registry*, que por sua vez publica um objeto que protege o acesso direto ao objeto registrado pelo *Xlet*. Quando um outro *Xlet* necessita usar uma função registrada no IXC ele realiza uma consulta (por meio da operação *LOOKUP*), obtendo uma cópia do objeto (conhecida como classe *Stub*) que previamente registrou a função. Isso garante a proteção ao objeto original, assim como ocorre no Java RMI. No MHP o IXC é implementado nos grupos funcionais de *org.dvb.ixc.**.

3.1.4 Arquitetura e Componentes do GINGA

A arquitetura do *middleware* para TVDI definido pelo SBTVD para compor o sistema ISDTV é compatível com as especificações ITU J.200, J.201 e J.202, ao apresentar tecnologias que suportam a programação de aplicações interativas sob os dois paradigmas de programação: o procedural, com suporte dado pela máquina de execução Ginga-J e o declarativo, representado pela máquina de apresentação Ginga-NCL. Conforme ilustra a Figura 3.7, a especificação do Ginga em ABNT (2008a) determina um ambiente de aplicações composto pelos seguintes elementos estruturais:

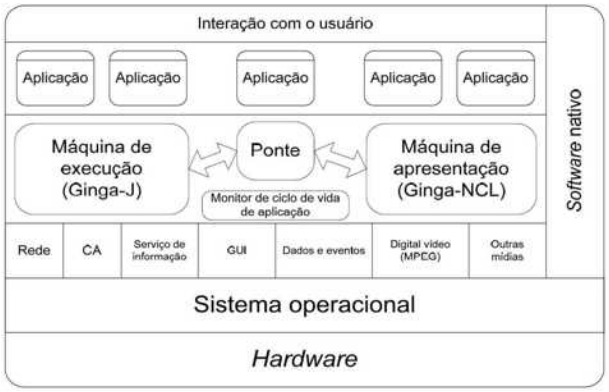


Figura 3.7: Estrutura do ambiente do *Middleware* Ginga.
FONTE: Adaptado de ABNT (2008a).

- *Máquina de apresentação e Máquina de execução*, de acordo com a ABNT NBR 15606-2.

- *Ponte* entre as duas máquinas - Mecanismo para aplicações que permite o mapeamento bidirecional entre as APIs Java e os objetos e métodos do DOM (*Document Object Model*), ECMAScript, LUAScript que compõem o domínio declarativo.

- *Monitor do ciclo de vida da aplicação* - Aplicação ou recurso do sistema operacional para controle do estado do software. Sua função inclui a gerência de todo o ciclo de vida da aplicação, incluindo inicialização, término e controle. O monitor do ciclo de vida de aplicações é definido no ambiente procedural.

- *Aplicações* - Programadas para a máquina de apresentação, execução ou para ambas.
- *Outras Mídias* - Incluem stream de mídia como áudio/dados ou monomídias como imagens/textos.
- *Software Nativo* - Inclui software legado ou escrito usando APIs com novas funcionalidades.

Considerando o Ginga-J, o *middleware* apresenta funções comuns de gerenciamento de recursos do STB, controle de mídias e ciclo de vida de aplicações, com o uso de algumas APIs existentes, principalmente do GEM. Além destas, o Ginga-J foi concebido considerando a convergência digital como um requisito fundamental (Souza Filho et al., 2007). Dessa forma o *middleware* apresenta componentes que suportam a integração do STB com dispositivos móveis como celulares ou PDAs. As API's do Ginga-J foram sistematizadas em três grupos funcionais (ver Figura 3.8). São eles:

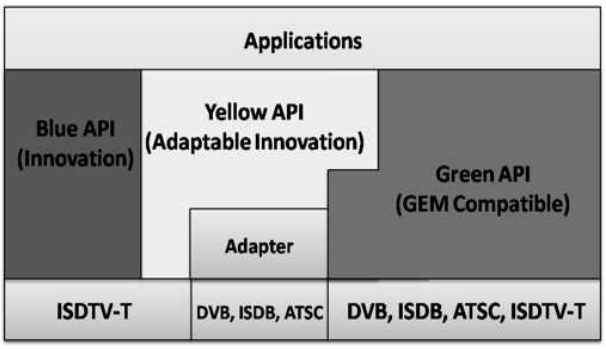


Figura 3.8: Grupos funcionais de APIs que compõem o Ginga-J.
FONTE: Adaptado de Souza Filho et al. (2007).

- *Green API* - API de interoperação com outros padrões de *middleware*. É GEM compatível, portanto aplicações Ginga escritas com componentes puros da *Green API* executarão em terminais MHP, OCAP, ACAP e ARIB B.23. Composto por pacotes do JavaTV, HAVi, DAVIC e DVB.
- *Yellow API* - API compatível com outros padrões através de um adaptador. Composto pelo *Java Media Framework* 2.1 para suporte à fluxos de vídeo, bem como extensões adaptadas do GEM para canal de retorno e acesso a informações de

serviços (em tabelas *Program Specific Information*, PSI, que compõem o Fluxo de Transporte).

- *Blue API* - API de inovação do Ginga-J com suporte à aplicações específicas do ISDTV. Compõe-se de três elementos: (1) *Device Integration*, que permite ao receptor a comunicação com outros dispositivos usando meios com fio (Ethernet, PLC) ou sem fio (infravermelho, bluetooth); (2) *Multi-User API*, que permite a interação simultânea de vários usuários com aplicações de TVDI; e (3) *Bridge NCL*, que permite a aplicações Java incorporarem conteúdo declarativo do NCL.

O cenário de definição do *middleware* procedural brasileiro está, até o período em que foi escrita esta Dissertação, com alguns pontos em aberto. Apesar de a especificação apontar o GEM como API integrável às demais do ISDTV, o fórum do SBTVD, por razões de ordem econômica e/ou política, levantou a discussão sobre a possibilidade de incorporar, em detrimento do GEM, APIs fabricadas pela Sun Microsystems, sob a tecnologia JavaDTV. O JavaDTV é derivado da especificação JavaME para dispositivos de consumo portáteis, composto pelo *Connected Device Configuration* (CDC), *Personal Basis Profile* (PBP), *Foundation Profile* (FP) e JavaTV, apresentado algumas particularidades como a API de *widgets* LWUIT, derivado dos pacotes gráficos do *Abstract Window Toolkit* (AWT) e Swing Java (Lagally e Patzold, 2009).

Sob o ponto de vista declarativo o *middleware* Ginga-NCL, especificado pela ABNT NBR 15606-2 (ABNT, 2008b), fornece uma máquina de apresentação para conteúdo multimídia baseado em scripts. O subsistema lógico do Ginga-NCL processa documentos no padrão da linguagem NCL, própria para a apresentação e sincronização de conteúdo multimídia, baseada no modelo NCM (Soares e Rodrigues, 2006). Além de NCL há suporte também para conteúdo ECMAScript, CSS e XHTML. O Ginga-NCL permite também a inserção de programação dinâmica às aplicações, por meio do suporte dado a execução de código escrito na linguagem LUA. Para cada tipo de mídia que executa sobre o Ginga-NCL (JPEG, PNG, MPEG, scripts, ou LUA) existem componentes correspondentes, numa camada abaixo do *middleware*, no *Ginga Common Core*, os chamados *Adapters*. A Figura 3.9 ilustra a arquitetura do Ginga-NCL.

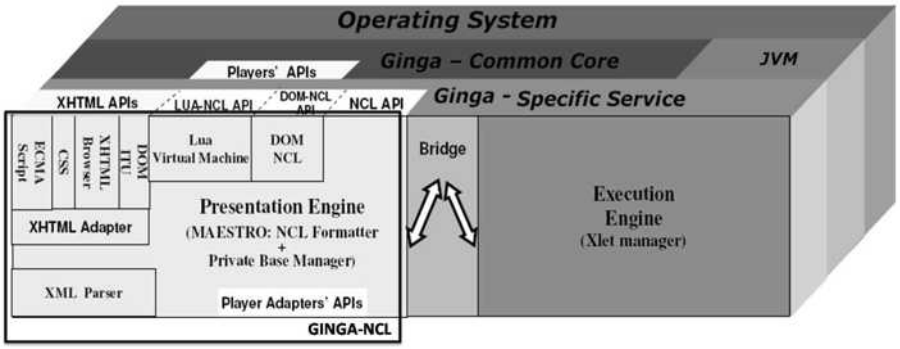


Figura 3.9: Arquitetura do Ginga-NCL no contexto do Ginga.
FONTE: Adaptado de Soares e Moreno (2007).

Diferentemente do *Xlet* como miniaplicativo procedural, a principal unidade de aplicação declarativa no Ginga é o documento NCL. O NCL define componentes de linguagem declarativa (na forma de *tags* XML) para representar todos os elementos do modelo NCM associados às mídias de um documento hipermídia e as relações entre elas, que podem ser de ordem estrutural (como estão definidas as mídias no documento NCL), temporal (quando uma mídia deve iniciar a apresentação) ou espacial (onde uma mídia deve ser apresentada, em que região da tela). Um documento NCL especifica para as mídias o que deve ser tocado, quando se define o tipo de mídia (*nós de mídia NCL*); onde deve ser tocado, quando se definem as *regiões NCL* para apresentação das mídias; como deve ser tocado, através de *descritores NCL* que definem a associação de uma mídia a uma região da tela; e quando deve ser tocado, onde se definem elementos de conexão entre as mídias (*portas*, *elos* e *conectores NCL*) (Soares et al., 2006b).

3.1.5 Definição do *Middleware* para a Plataforma

As características do MHP, como a interoperabilidade do GEM e o uso de APIs estáveis, o torna o principal *middleware* de referência aberto. Qualquer API construída sobre a base GEM do MHP pode ser, conforme apresentado, extensível a outros padrões. Por essa razão o MHP (com o subconjunto comum à base do Ginga-J) destaca-se como apropriado para a construção da plataforma integrada a ambientes de software de *Home Network* para a TVDI brasileira, propósito desta Dissertação. Outro ponto positivo do MHP é o suporte para desenvolvimento de aplicações com software

livre, com a existência de implementações de referência de código aberto, em emuladores como *XleTView* (XleTView, 2009) e *OpenMHP* (OpenMHP, 2009). Dessa forma, os mecanismos de *middleware* descritos nessa seção serão incorporados ao modelo TVDI-HN como a parte procedural da plataforma.

Do ponto de vista declarativo, a solução compatível com o *middleware* brasileiro é a que utiliza o Ginga-NCL. Desse modo, o modelo de execução do *middleware* será integrado à plataforma de software para *Home Network*, respeitando as características nativas de cada parte. Um modelo declarativo permitirá utilizar conceitos do Ginga-NCL de forma a integrar as mídias exibidas no *middleware* com os dispositivos da rede doméstica, aumentando consideravelmente o escopo dos cenários que podem ser criados. Na próxima seção, serão analisadas as plataformas abertas para gerenciamento de serviços em *Home Network*, escolhendo-se a mais apropriada para integração com o Ginga, seja no modelo procedural centralizado no GEM do Ginga-J, ou no declarativo.

3.2 Escolha da Plataforma para HN

Algumas plataformas de software para HN são apresentadas neste tópico. A maior parte delas define mecanismos e componentes que permitem o compartilhamento de serviços de dispositivos com tecnologias distintas em redes distintas, no ambiente doméstico, de forma a facilitar o desenvolvimento de aplicações para este ambiente e diminuir a intervenção do usuário na configuração do mesmo. A seguir, dentre as tecnologias existentes, são avaliadas: *Jini*, *UPnP*, *HAVi* e *OSGi*.

3.2.1 Arquitetura Jini

Jini (Jini, 2009a) é uma arquitetura de rede para sistemas distribuídos, desenvolvida pela *Sun Microsystems* baseada na tecnologia Java. O objetivo da arquitetura é fazer da rede um ambiente dinâmico e de fácil configuração, onde serviços são adicionados e removidos de maneira flexível. Mais precisamente, o propósito é chegar a um sistema onde o usuário final é capaz de compartilhar serviços e recursos na rede; onde usuários têm acesso simples e fácil a serviços, mesmo se a localização dos usuários ou

dos serviços muda; onde a construção, manutenção, e mudança de dispositivos, software, ou usuários podem ser gerenciadas. De modo geral, Jini pode ser considerada como uma extensão do ambiente Java de uma máquina (aplicações não distribuídas) para um ambiente de rede (aplicações distribuídas).

Um típico ambiente Jini é composto pelos seguintes elementos: *serviços Jini*, *Clientes Jini* e pelo menos um *Jini Lookup Service (JLS)*, que interagem conforme ilustrado na Figura 3.10:

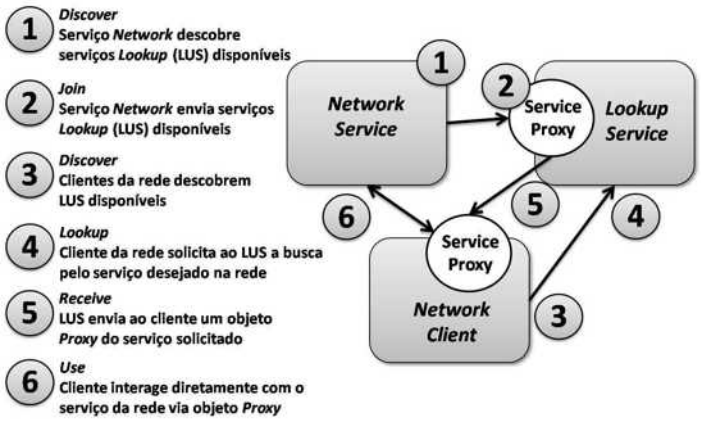


Figura 3.10: Esquema de Operação do Jini.
FONTE: Adaptado de 6Power (2009).

Um *Serviço* Jini é o elemento que representa uma interface para acesso a recursos da rede. Podem ser usados por uma pessoa, um programa ou outro serviço. Alguns exemplos de serviços: armazenamento, computação, dispositivos de hardware, um usuário, dispositivos como impressoras, informações em bancos de dados (Waldo, 1999). Um sistema Jini é feito de serviços que podem ser combinados de modo a completar uma tarefa particular (esse é um conceito geral em sistemas baseados em serviços, chamado *composição de serviços*). O Jini oferece um conjunto de interfaces Java que especificam conceitos chave para a definição do relacionamento entre serviços.

Um *Cliente Jini* é um cliente para serviços. Clientes Jini procuram por novos serviços no *Jini Lookup Service*. Uma vez que uma consulta ao JLS retorna um serviço o cliente recebe um objeto (*Proxy*, um bytecode Java que representa a interface para o serviço) que é executado na JVM do próprio cliente (este mecanismo é característico do Java RMI).

Um *Jini Lookup Service* é usado para encontrar e validar serviços. O que ele faz é o mapeamento de interfaces de um serviço em um conjunto de objetos que implementam esse serviço. Um serviço tem de ser registrado em pelo menos um serviço *Lookup*. Um *Jini Service Provider* (provedor de serviços Jini) registra a si mesmo para cada serviço *Lookup* descoberto. Um objeto pode consistir de um ou mais serviços. Além disso, objetos podem também encapsular nomes de outros serviços ou diretórios, o que possibilita a integração do sistema de *Lookup* Jini com mecanismos de busca de serviços de outras plataformas.

A base da arquitetura Jini está nos esquemas de descoberta, junção e busca (*lookup*) (Jini, 2009a). O protocolo de descoberta é usado quando um dispositivo procura por um serviço de *lookup* para se registrar na rede. O protocolo de junção é usado quando um serviço encontra o *Jini Lookup Service* ao qual deseja se associar. O protocolo de lookup é usado quando um cliente precisa localizar e invocar um serviço. Descoberta/Junção representa como um serviço é adicionado ao sistema. O provedor de serviços notifica todos os serviços *lookup* para que se identifiquem (descoberta). Após isso, um objeto de serviço é carregado no *lookup service* (*join*). Este objeto de serviço possui a interface Java para o serviço, incluindo os métodos da interface a ser invocada quando do uso do serviço, além de outras informações descritivas. O processo *lookup* garante que uma cópia do objeto de serviço seja carregada para o cliente. Dessa forma, o cliente pode utilizar o serviço.

3.2.2 Arquitetura UPnP

O *Universal Plug and Play* (UPnP) é uma iniciativa da Microsoft para oferecer conectividade simples, robusta e de fácil configuração entre PCs (UPnP, 2009). O *Plug and Play* (PnP) torna simples a tarefa de configurar e adicionar periféricos a PCs, enquanto que o propósito do UPnP é estender esta simplicidade para uma plataforma distribuída, de rede, habilitando o registro, descoberta e controle de dispositivos. Suas metas são *configuração-nula* (mínimo de interferência do usuário), *rede transparente* (*invisible networking*) bem como descoberta automática de dispositivos de diferentes fabricantes.

Os componentes básicos de uma rede UPnP são: *dispositivos*, *serviços* e *pontos de controle.* Um *dispositivo* UPnP contém serviços e dependências com outros dispositivos. Um documento XML descreve cada dispositivo e lista o conjunto de serviços que este pode prover, bem como algumas propriedades associadas ao dispositivo. Um *serviço* em UPnP é a unidade de controle elementar. Um serviço consiste de *uma tabela de estados*, um *servidor de controle* e um *servidor de eventos.* A primeira controla o estado do serviço através de variáveis de estado. O segundo atualiza o estado de acordo com requisições realizadas. Já o último elemento notifica a disponibilidade do serviço a outros serviços ou entidades interessadas em utilizá-lo (Bishaj, 2007). Um *ponto de controle* UPnP é o mecanismo através do qual os dispositivos serão descobertos na rede. Como no UPnP não existe uma entidade central de registro de serviços, isso deve ser feito no próprio dispositivo, através do ponto de controle (Friday et al., 2004).

Com o objetivo de facilitar a integração de dispositivos na rede, o UPnP utiliza alguns protocolos padrão. O TCP/IP é a base sobre a qual são construídos UPnP e alguns protocolos como UDP (*User Datagram Protocol*), DHCP (*Dynamic Host Configuration Protocol*) e DNS (*Domain Name System*). O protocolo HTTP é parte do núcleo principal do UPnP. Todos os aspectos de comunicação são baseados no HTTP e suas variantes.

Entre os protocolos do UPnP está o *Simple Service Discovery Protocol* (SSDP), que define como encontrar serviços em redes UPnP (UPnP, 2009). O SSDP serve tanto para os pontos de controle localizar serviços na rede, como para dispositivos notificarem a rede de sua disponibilidade. Uma unidade de controle UPnP, quando inicializada, pode enviar uma requisição de busca para descobrir serviços e dispositivos. Dispositivos UPnP, por outro lado, aguardam alguma requisição para seus serviços. Se o critério de busca é satisfeito, uma resposta é enviada ao ponto de controle, indicando a localização do serviço.

3.2.3 Arquitetura HAVi

HAVi é um consórcio fundado por empresas e instituições de pesquisa em eletrônica e dispositivos de consumo. O foco do grupo está em dispositivos de

entretenimento doméstico, especialmente áudio/vídeo. HAVi habilita comunicação entre dispositivos para detecção, busca e controle. Adicionalmente, grupos de dispositivos podem prover serviços melhorados por meio de cooperação (composição de serviços). Entretanto, dada a variedade de dispositivos eletrônicos, a arquitetura apresenta alguns mecanismos de extensão para a integração com dispositivos não HAVi da rede doméstica (Bishaj, 2007).

De acordo com Lea et al. (2000) os dispositivos HAVi são classificados em: *Full AV Devices* (FAV), *Intermediate AV Devices* (IAV), *Base AV Devices* (BAV) e *Legacy AV Devices* (LAV). Os dispositivos FAV são compatíveis com o HAVi por completo, os IAV não possuem o ambiente de execução, os BAV contém funcionalidades básicas e os LAV são dispositivos não compatíveis com HAVi.

No HAVi, cada dispositivo é representado como um elemento de software na rede, através do módulo de controle de dispositivo (*Device Control Module*, DCM), que contém uma série de módulos de controle funcional (*Functional Control Modules*, FCM). Cada FCM é responsável por uma função do dispositivo, enquanto que o DCM é uma interface para acessar as funções do dispositivo. Cada dispositivo possui dados adicionais que provêem uma descrição das propriedades do mesmo. O HAVi possui ainda um sistema de mensagens, que tem basicamente duas funções: (1) criar identificadores únicos para um objeto quando ele é registrado e (2) gerenciar a informação trocada entre dispositivos. O *Registry* é o elemento que atua como um serviço de diretórios no HAVi. Ele mantém a informação sobre objetos registrados e disponíveis, bem como suas propriedades.

Outro elemento importante na arquitetura HAVi é o gerenciador de comunicação de mídias 1394, baseado no padrão IEEE 1394 (*FireWire*). Este gerenciador provê um mecanismo para envio de requisições e recebimento de informações de dispositivos remotos. O uso do padrão IEEE 1394 traz benefícios como a transmissão de dados em alta taxa com garantia de qualidade de serviço (QoS), importante para tráfego de dados na rede de dispositivos de áudio/vídeo (IEEE, 1996).

A conexão entre os elementos de um dispositivo e entre dispositivos HAVi é configurada pelo *Gerenciador de Fluxo*. Esses fluxos são recebidos pelos FCMs dos dispositivos. O papel do Gerenciador de Fluxo é lidar com requisições de conexões,

por meio da alocação e gerenciamento de recursos (principalmente largura de banda). Dessa forma, HAVi provê QoS para *stream* multimídia. Outro elemento importante na arquitetura HAVi é o *Gerenciador de Recursos*, o qual gerencia os recurso usados para cada dispositivo.

No que diz respeito à segurança, HAVi fornece suporte de duas formas. A primeira é através de níveis de acesso, onde as APIs são divididas em *confiáveis* e *não-confiáveis*. A outra forma é usando o esquema derivado do RSA com chaves liberadas por uma autoridade certificadora HAVi. Isso é feito de forma que somente código certificado Java possa executar no ambiente.

3.2.4 Arquitetura OSGi

OSGi é uma especificação definida pelo consórcio *Open Services Gateway Initiative*, que provê um padrão aberto para programação remota de dispositivos (OSGi, 2009a). Além de permitir o gerenciamento remoto de dispositivos, o OSGi em seu conceito vai mais além: trata-se de um *framework* que fornece suporte à implementação de aplicações baseadas em componentes de software e no modelo orientado à serviços, escritas principalmente na linguagem Java. O *framework* é uma implementação da arquitetura orientada à serviços (ou SOA, *Service Oriented Architecture*) num ambiente caracterizado por serviços e unidades modulares de componentes (as aplicações da plataforma, denominadas *bundles*).

O modelo do OSGi é caracterizado por um *framework* servidor de aplicações Java para o ambiente de dispositivos conectados, sejam de pequeno, médio ou grande poder de processamento. Aplicações do modelo variam desde Automação Industrial, Automação Doméstica, Comunicação, *Smart Phones*, Telemática, Servidores de Aplicações, Aplicações Desktop (que tenham por base o modelo de gerenciamento de aplicações/serviços), Veículos, etc. (OSGi, 2009b). O *framework* é implementado na Plataforma de Serviços OSGi (*OSGi Service Platform*) que gerencia as interfaces do dispositivo no qual está sendo implementado por meio de unidades de aplicações baseadas em componentes de software.

O OSGi dá suporte à integração da plataforma de serviços a algumas das plataformas para *Home Network* existentes, tais como *LonWorks*, *Jini*, *UPnP* e *HAVi*, além de simplificar a instalação e operação de múltiplos serviços em uma única plataforma, o *Home Gateway*. A Figura 3.11 ilustra o modelo OSGi de operação do *Home Gateway* no ambiente de *Home Network* e algumas tecnologias integradas.

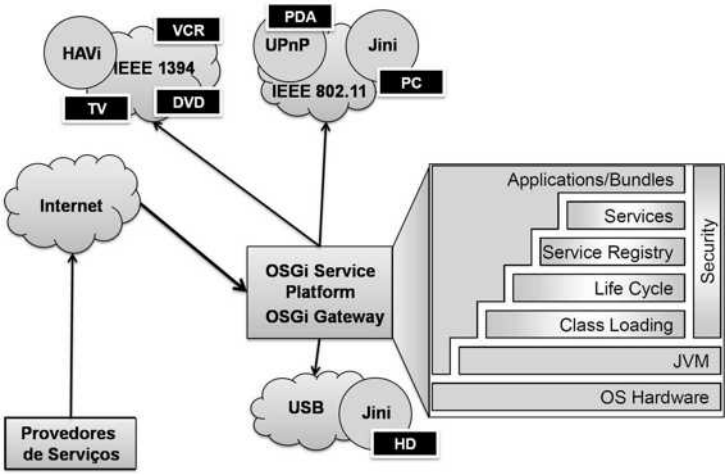


Figura 3.11: Arquitetura do OSGi e *OSGi Service Platform*.
FONTE: Adaptado de Hoang (2007); OSGi (2009b).

Arquitetura da Plataforma de Serviços

O ambiente de execução do OSGi apresenta-se segmentado em quatro camadas. Cada camada apresenta um conjunto de componentes com um propósito específico. Na arquitetura estão presentes desde componentes de carga de classe (*Class Loading*), próximos às interfaces da JVM, até os níveis mais altos de abstração que incluem os serviços e as unidades de aplicações modulares OSGi, os *bundles*. Conforme ilustrado na Figura 3.11, as principais camadas do modelo OSGi, que agregam todos os componentes da arquitetura, conforme em (OSGi, 2009a), são: (1) *Execution Environment* (*Security*), (2) *Module* (*Class Loading*), (3) *Life Cycle* e (4) *Service* (*Services* e *Service Registry*).

O *OSGi Execution Environment* (e *Security Layer*) é a especificação do ambiente Java. Oferece mecanismos de segurança e API's do J2SE, CDC, CLDC e MIDP. Todos são ambientes válidos de execução para o OSGi. O OSGi possui também um ambiente de execução baseado no *Foundation Profile* e uma pequena variação que especifica os

requisitos mínimos de plataforma (chamado *OSGi Minimum*) no que se refere às APIs Java exigidas.

A camada OSGi *Module* define um mecanismo rigoroso para carga de classes Java na plataforma. A camada utiliza mecanismos de Java para garantir segurança no acesso aos objetos e recursos da plataforma. O *Module* define, por exemplo, a hierarquia na carga de classes, e como carregar classes e recursos empacotados nos *bundles*. *Bundles* são definidos nesta camada como unidades de aplicações e recursos empacotados como um arquivo JAR padrão de Java. Além de classes, um *bundle* pode carregar arquivos HTML, XML, TXT, imagens, etc. Um *bundle* contém um arquivo principal, no formato de manifesto Java, que funciona como um meta-arquivo que auxilia o *framework* a mapear todas as propriedades do *bundle* de modo a gerenciar seu ciclo de vida e sua relação com os serviços.

A camada *OSGi Life Cycle* gerencia o ciclo de vida dos *bundles*. *Bundles* executam no ambiente em estados definidos e controlados pelo *framework*. Podem ser dinamicamente instalados, removidos ou ativados na plataforma. Um *bundle* pode se encontrar nos seguintes estados: *installed* (o *bundle* é instalado e suas dependências para ativação são identificadas); *resolved* (o *framework* consegue inicializar todas as dependências do *bundle*; este está pronto para ser ativado); *starting* (o *bundle* está em processo de ativação, carga de classes, etc); *active* (*bundle* em execução) *stopping* (o *bundle* se tornará inativo) e *uninstalled* (o *bundle* é descarregado da plataforma).

A última camada, o *OSGi Service*, adiciona elementos importantes para o modelo, como os conceitos de serviços e Registro de Serviços (*OSGi Service Registry*). Este último provê um modelo de cooperação entre *bundles*, baseado no modelo orientado a serviços, que considera aspectos dinâmicos da plataforma. *Bundles* podem cooperar entre si de duas formas na plataforma: por compartilhamento direto de classes ou pelo mecanismo de descoberta e utilização de serviços. Neste último o *Service Registry* tem papel importante pois é a entidade na qual serviços de um *bundle* são publicados e disponibilizados a outros *bundles*. Um serviço OSGi é uma interface Java associada às operações que podem ser realizadas sobre o objeto que representa. Este objeto pode implementar métodos sobre operações em um dispositivo tais como ligar uma lâmpada, abrir uma porta, visualizar um

stream de vídeo, etc. No núcleo do OSGi vários serviços padrão já vêm disponíveis para o desenvolvedor de aplicações como serviços de HTTP, Log, Administração de usuário, Gerenciamento de permissões, Parser XML, Conexões com Jini, UPnP, entre outros. Ao longo de sua evolução, o OSGi, em 4 versões, incorporou cada vez mais serviços, estendendo sua aplicação para mais ambientes.

3.2.5 Definição do Framework para a Plataforma

Comparar arquiteturas diferentes é difícil, pois exige a identificação de parâmetros em comum de forma a avaliá-las sob uma mesma perspectiva. Dessa forma, a comparação que é feita aqui é estruturada analisando o maior subconjunto de características comuns às quatro plataformas apresentadas. Alguns parâmetros identificados para comparação foram organizados na Tabela 3.3. São eles:

- *Licença de uso* - HAVi e OSGi são plataformas de código aberto; suas especificações encontram-se livremente distribuídas. Para o UPnP, sendo uma iniciativa Microsoft, não é código aberto. Jini é código aberto, porém com taxas para uso comercial.

- *Transporte na Rede* - Jini, OSGi são independentes da tecnologia da rede de dispositivos, enquanto que HAVi é baseada no IEEE 1394, o que permite implementar QoS. UPnP baseia-se na rede TCP/IP como uma estratégia de uso de um dos mais populares padrões de rede.

- *Linguagem de Programação* - HAVi e UPnP são independentes de linguagem de programação. Para o Jini, embora o padrão seja Java, a especificação dá suporte para qualquer outra linguagem que produza bytecodes para a JVM. Já o OSGi, embora Java também predomine, oferece suporte a código nativo em C e programação de serviços de forma declarativa, usando XML.

- *Repositório Central de objetos* - Observando as arquiteturas HAVi e UPnP, pode ser verificado que, em oposição ao modelo cliente-servidor tradicional, tanto cliente como servidor estão alocados no dispositivo, não possuindo uma entidade para

registro das funcionalidades/serviços dos dispositivos. Já OSGi e Jini oferecem o *Service Registry* e o *Lookup Service*, respectivamente.

- *Segurança* - O UPnP não contempla mecanismos de segurança. Para Jini e OSGi há a segurança estendida do Java enquanto que o HAVi oferece os esquemas de *níveis* e *assinaturas*.

Framework	**Licença**	**Transporte na Rede**	**Linguagem de programação**	**Repositório de objetos**	**Segurança**
Jini (v1.2)	aberta para uso não comercial	independente da tecnologia de rede	Java	centralizado	Plataforma Java
UPnP (v0.91)	aberta somente para membros	TCP/IP	independente	não	–
HAVi (v1.1)	aberta	IEEE 1394	independente	não	níveis e assinaturas
OSGi (vR4)	aberta	independente da tecnologia de rede	Java	sim (*Service Registry*)	Plataforma Java

Tabela 3.3: Comparação entre os protocolos em *frameworks* para *Home Network*.

Os propósitos do projeto desta Dissertação incluem a criação de uma estrutura comum sobre a plataforma STB-HG que permita o uso de serviços centralizados no *Home Gateway* por aplicações do STB, sendo que para este último, a plataforma de *middleware* escolhida deriva da especificação GEM do MHP e do Ginga declarativo (Ginga-NCL), que operam sobre a JVM. Dessa forma, analisando os parâmetros levantados na Tabela 3.3 percebe-se que o OSGi é mais adequado que os demais pelas seguintes razões: (1) tecnologia baseada em Java; (2) modelo orientado à serviços, o que facilita a integração com aplicações de TV Digital; (3) independência da tecnologia da *Home Network*, o que garante a compatibilidade com outras redes (incluindo as do HAVi, UPnP e Jini) e, principalmente (4) os serviços são centralizados numa única plataforma, e não distribuídos para cada dispositivo, o que facilitará o uso dos serviços pelas aplicações de TV Digital.

3.3 Resumo

Neste capítulo foi feita uma análise dos principais padrões existentes para o desenvolvimento de aplicações e gerenciamento de serviços, sejam eles no grupo da TV Digital ou das *Home Networks*. Sobre o primeiro grupo, foi feita uma análise das tecnologias e práticas de desenvolvimento de aplicações para ambientes de *middleware* (procedural ou declarativo), focando na arquitetura genérica do MHP (GEM compatível, de modo a estender sua implementação ao Ginga-J) e do Ginga-NCL, a fim de identificar os elementos das arquiteturas que farão parte do modelo descrito nos próximos capítulos. Sobre o segundo grupo, que inclui tecnologias em *frameworks* para *Home Networks*, foi feita uma introdução sobre o modelo de funcionamento dos principais padrões existentes.

Como plataforma de aplicações de TVDI, chegou-se a conclusão de que para uma abordagem compatível com o *middleware* brasileiro e extensível a outros ambientes, o ideal é a utilização do MHP-GEM, juntamente com o Ginga-NCL. Como *framework* para gerência de aplicações no ambiente de redes de dispositivos eletrônicos, após a comparação com outros *frameworks* identificou-se o OSGi como o mais adequado. Assim, o modelo de colaboração criado nesta Dissertação e descrito com detalhes no Capítulo 5 integra elementos Ginga-J (GEM), Ginga-NCL e OSGi.

No próximo capítulo, a abordagem de colaboração TVDI-HN desta Dissertação será contextualizada em meio ao conjunto de trabalhos que existem na mesma linha de pesquisa. Isso é importante pois permite identificar parâmetros que, no final do trabalho servirão como base de comparação da plataforma aqui desenvolvida com outras abordagens de integração TVDI-HN existentes.

Capítulo 4 - Análise de Modelos TVDI-HG

Esse capítulo representa a parte central do referencial bibliográfico coletado e analisado durante a pesquisa para a elaboração e desenvolvimento de projeto desta Dissertação. Aqui conceitos já estabelecidos nos capítulos anteriores (TV Digital, *Home Network*, *Middleware* e *Framework*) são utilizados no estudo dos principais trabalhos relacionados na pesquisa bibliográfica, na construção de modelos ou arquiteturas de *Home Networks* nas quais STB e *Home Gateway* interagem de forma a compartilhar serviços no ambiente doméstico.

As principais referências são organizadas em grupos e analisadas de acordo com seu nível de importância para o projeto da Dissertação. Os grupos apresentam trabalhos que variam gradativamente desde aqueles que propõem a interação STB-HG até projetos mais específicos, que abordam a colaboração a nível de plataformas de software (o uso de *middleware* e *framework*) para promover tal interação. Esta última categoria de trabalhos é o referencial base para a estratégia adotada no protótipo desenvolvido nesta Dissertação, apresentado nos Capítulos 5 e 6.

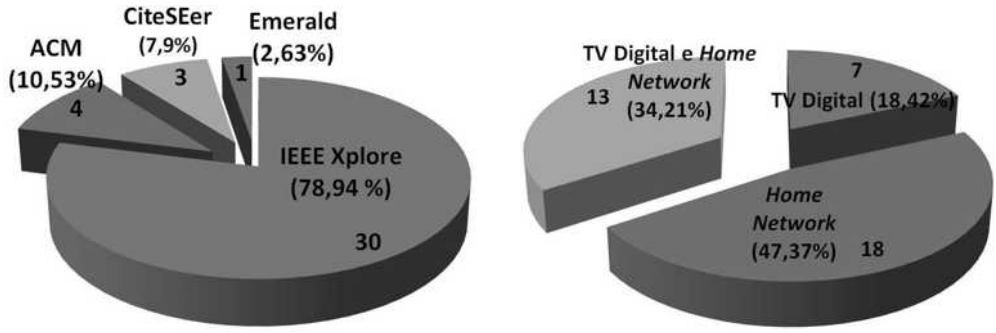
Ao final deste capítulo, espera-se contextualizar teoricamente o trabalho a ser descrito nos capítulos posteriores no conjunto de trabalhos da mesma natureza. Estes são confrontados e suas características são analisadas sob aspectos comuns de modo a identificar pontos não contemplados ou que exigem melhorias, particularmente no que se refere ao modelo brasileiro de TV Digital. Estes pontos serão posteriormente implementados na plataforma da Dissertação para, ao final do projeto, ser possível a comparação da abordagem desenvolvida com as dos trabalhos relacionados.

4.1 Estratégia de Pesquisa

A pesquisa de artigos científicos para formar a base deste trabalho de Dissertação foi realizada principalmente em sítios de pesquisa de artigos e periódicos do *IEEE Xplore* (IEEE, 2009) e *ACM Digital Library* (ACM, 2009). Outras bases consultadas foram *Computer and Information Science Papers* (CiteSEer) (CiteSEer, 2009) e *Emerald Group Publishing Limited* (Emerald, 2009). Os documentos pesquisados incluem trabalhos nas áreas de *Sistemas Computacionais*, *Automação de Dispositivos*, *Dispositivos Eletrônicos de Consumo*, *Sistemas Embarcados* e *Engenharia de Software*. Os principais termos e palavras-chaves utilizados nos critérios de pesquisa foram: *Digital TV*, *Home Network*, *Framework*, *Middleware*, *Digital Convergence*, *MHP*, *OSGi*, *Ginga Middleware*, *Set-Top-Box*, *Home Gateway*, *Architecture*, *Interoperability*, e *MHP OSGi Convergence*.

Dentre os trabalhos encontrados ao longo dessas fontes, foram separados os 38 que mais contribuíram para a pesquisa realizada. Os trabalhos estão divididos entre as fontes da seguinte forma: dos 38, 30 pertencem ao *IEEE Xplore* (a maioria - 22 - retirada de periódicos *IEEE Transactions on Consumer Electronics*); 4 pertencem a *ACM Digital Library*; 3 pertencem ao *CiteSEer* e 1 à *Emerald*. Outra classificação para os trabalhos leva em consideração artigos que discorrem sobre 3 categorias distintas: (1) *TV Digital* (7 artigos); (2) *Home Network* (18 artigos) e (3) *Interação TV Digital e Home Network* (14 artigos). Esta última categoria serve de base para a construção deste capítulo. A Figura abaixo (Figura 4.1) ilustra os gráficos das distribuições dos trabalhos pesquisados separados por bases de pesquisa (Figura 4.1(a)) e categorias de trabalhos (Figura 4.1(b)).

A coleta e filtragem de trabalhos foram organizadas em 4 etapas, e suas análises foram divididas ao longo de três capítulos da Dissertação, sistematizados no diagrama de blocos da Figura 4.2. Na primeira fase (1), coletou-se material sobre TVDI e HN (Capítulo 2); na segunda fase (2), foram organizadas referências sobre *middleware* para TVDI e *framework* para HN (Capítulo 3); na terceira fase (3), foram realizadas pesquisas em trabalhos que caracterizam STBs como *Home Gateways*, verificando as interfaces que os dois dispositivos têm em comum, sugerindo o uso das plataformas de STBs como



(a) Por bases de pesquisa. (b) Por categorias.

Figura 4.1: Distribuição de artigos pesquisados.

Home Gateway e apresentando algumas aplicações deste uso (Capítulo 4); na última fase (4) foram coletadas as principais referências do trabalho: as que propõem modelos de colaboração em nível de software entre *middleware* e *framework* (Capítulo 4).

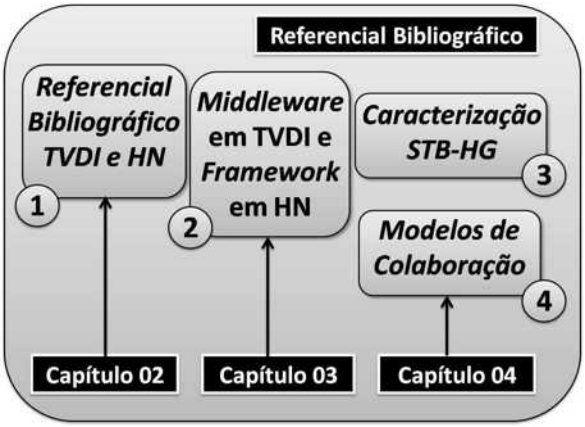


Figura 4.2: Etapas de Coleta de Material Bibliográfico.

4.2 Abordagens de Integração TVDI-HN

Esta seção se concentra nas duas últimas das 4 fases de pesquisa descritas na seção anterior. São apresentados modelos que apontam o potencial dos STBs como plataformas de *Home Gateways*. Em seguida são analisados trabalhos que propõem soluções de integração STB-HG em nível de software, utilizando tecnologias abertas em *middleware* e *frameworks*. A análise desses últimos trabalhos é fundamental para definir a estratégia sobre a qual será construído o modelo de colaboração baseado no *middleware* Ginga.

4.2.1 Caracterização de STBs como HG

Conforme apresentado anteriormente, um *Home Gateway* possui interfaces para a comunicação com dispositivos heterogêneos em *Home Networks*, além de habilitar a comunicação desta com o meio externo. Já um STB é um equipamento que dá suporte à funções de decodificação de sinal digital e execução de aplicações interativas. Entretanto, alguns trabalhos apresentam propostas de integração entre os dois dispositivos numa única plataforma que agrega ao STB interfaces para comunicação com dispositivos eletrônicos (como WiFi, Bluetooth, ZibBee, etc.).

Os trabalhos de den Hartog et al. (2004) e Ge et al. (2007) apontam evidências do potencial uso dos STBs como *Home Gateways*, ou o uso de *Home Gateways* como STBs. No primeiro, um novo modelo baseado na arquitetura de rede OSI (*Open Systems Interconnection*) é usado para a classificação de *Home Gateways*. Esse modelo é utilizado para analisar as atuais soluções em *Home Gateway* e esboçar tendências futuras para o uso deste equipamento. É feito um estudo geral sobre o quão viável é implementar um *Home Gateway* em equipamentos como *STBs*, *Console de Jogos*, *PC's* e *Modems de Banda Larga*.

Segundo den Hartog et al. (2004) são sete os requisitos para analisar o potencial de um destes dispositivos como um *Home Gateway*: (1) estar sempre ligado e operacional; (2) ser bastante confiável; (3) ser gerenciável, de preferência remotamente; (4) ter um tempo de depreciação não inferior a três anos; (5) ter preços acessíveis e produção subsidiada por modelos de negócios; (6) suporte à integração com múltiplos periféricos e redes e (7) suporte a uma variedade de serviços. Segundo o referido trabalho, os STBs contemplam as características 2, 5, 6 e 7, o que os torna, entre os quatro equipamentos analisados, o mais apropriado para ser um *Home Gateway* com todas as camadas de rede OSI implementadas.

Já o trabalho de Ge et al. (2007) analisa as características separadas de STBs e *Home Gateways* de modo a identificar pontos em comum. Segundo o trabalho, integrar as duas plataformas numa só reduz custos ao mesmo tempo em que provê mais funções de qualidade. O trabalho propõe uma nova geração de STBs, projetando e implementando

um STB capaz de prover serviços de difusão, entretenimento, segurança, administração remota, comunicação e automação doméstica. O projeto do STB tem por objetivo atender aos seguintes requisitos que, segundo o trabalho caracterizam a convergência STB-HG: (1) *requisitos de interfaces de redes e tecnologias de comunicação*; (2) *gerenciamento de serviços locais/remotos* e (3) *custos operacionais e de consumo (desempenho).*

Outras categorias de trabalhos são as que consideram soluções que integram STB e HG através de mecanismos de comunicação (local ou remota) entre elementos que constituem as tecnologias em *middleware/framework* para estas plataformas. Estes trabalhos são descritos a seguir.

4.2.2 Integração STB-HG

Tanto do ponto de vista da TV Digital como das *Home Networks* existe uma variedade de padrões e tecnologias que podem ser úteis para esta pesquisa. Isto pode ser utilizado para justificar o fato de, na literatura, existirem abordagens muito diferentes para a solução do mesmo problema o que, algumas vezes, dificulta a identificação de parâmetros para avaliar esses trabalhos sob uma mesma perspectiva. Dentro do escopo do trabalho desta Dissertação, no que se refere à esta categoria de trabalhos pesquisados, foram considerados 3 critérios gerais para a classificação das abordagens de integração STB-HG, descritas a seguir: (1) Abordagem baseada na Home Network; (2) Abordagem baseada na TVDI e (3) Colaboração entre as duas plataformas.

Abordagem Centrada na Home Network

Nesta categoria um trabalho merece destaque. Matsubara et al. (2005) apresentam uma arquitetura centrada na TV Digital que permite o gerenciamento de múltiplos tipos de conteúdo de dispositivos da *Home Network* baseada nos padrões UPnP e DLNA. O trabalho implementa um sistema de TV Digital, com componentes UPnP e DLNA adicionados ao sistema de modo a fornecer suporte ao tratamento de conteúdo transportado em redes IP e IEEE 1394. Para os propósitos do trabalho desta Dissertação, uma desvantagem significativa deste trabalho é a necessidade de se desenvolver os módulos

da TV Digital com todas as interfaces para comunicação com *Home Network* sem o uso das funcionalidades já oferecidas pelos *middleware* de TVDI.

Abordagem Centrada na TVDI

Na segunda categoria, destaca-se o trabalho de Portelli et al. (2008). Nele é criado um mecanismo para gerenciar redes domésticas a partir de uma extensão do *middleware* de TV Digital. É apresentado um esquema de descoberta e gerenciamento de serviços usando um subconjunto do MHP customizado para estes fins. Um novo protocolo é implementado de modo a permitir o registro, a descoberta e o uso de serviços em redes baseadas em TCP/IP. Embora o modelo tenha se mostrado útil, a necessidade de modificar componentes da especificação MHP e criar um novo modelo para serviços de dispositivos da *Home Network* em meio a uma variedade de protocolos para este propósito acaba sendo uma desvantagem deste trabalho.

Colaboração entre as duas plataformas

Os trabalhos da última categoria apresentam estratégias de explorar as características dos modelos de software já existentes para plataformas de STBs e *Home Gateways*, promovendo um ambiente de colaboração entre os componentes de cada modelo. No método descrito em Tkachenko et al. (2004) tanto aplicações DASE-ATSC quanto OSGi são entregues ao consumidor por meio do operador da rede, que atua como difusor ATSC e operador de gateway OSGi. Os aplicativos são inseridos no carrossel de dados, encapsulados em fluxos MPEG-2 TS (*Transport Stream*) com o áudio e vídeo relacionado. Do lado do consumidor, o STB DASE recebe o fluxo, extrai os serviços OSGi e os distribui pelos dispositivos da rede por meio do gateway OSGi. A conexão entre um aplicativo de TVDI e um dispositivo da rede é realizada através de comandos usando uma rede IP.

Tkachenko et al. (2005) apresentam um *framework* chamado DTV-HNF (*Digital TV - Home Network Framework*), que gerencia solicitações de aplicativos de TVDI para acessar serviços em um dispositivo conectado à rede residencial através do uso de *services plugins*. Um *service plugin* é uma unidade de software especial carregada para o STB

através da rede de difusão. Essa unidade provê uma interface para aplicações de TVDI usarem serviços dos dispositivos da rede. A conexão entre o aplicativo e um *service-plugin* é realizada por meio do *service object* (objeto de serviço). Uma aplicação de TVDI que necessite interagir com algum dispositivo solicita ao *framework* um *service object*; este por sua vez busca o *service-plugin* correspondente; caso não esteja presente, o *service-plugin* é buscado na rede de difusão e instalado na plataforma. O trabalho em Tkachenko et al. (2006) é uma extensão do mesmo conjunto de componentes de Tkachenko et al. (2005) com o adicional de que todos os elementos de software do *framework* estão sob uma mesma plataforma iDTV-HN.

O trabalho de Bae et al. (2006) utiliza outras tecnologias para a integração TVDI-HN. Nele é proposta uma arquitetura de convergência entre serviços de difusão e de *Home Network*, baseado no *middleware* ACAP e na pilha de protocolos UMB (*Universal Middleware Brigde*). O modelo apresenta dois conjuntos de componentes de software: um *Service Proxy* sob a arquitetura ACAP e um *Service Broker* sob a pilha UMB que estabelecem conexões através do protocolo *Simple Service Discovery Protocol* (SSDP) permitindo a descoberta e uso de serviços da rede residencial pelas aplicações ACAP-J.

Em Tkachenko et al. (2004, 2005) e Bae et al. (2006) os *frameworks* estão em ambientes separados, dependentes do sistema de comunicação, e os aplicativos de TVDI acessam os serviços de dispositivos através da rede de difusão. Uma desvantagem significativa desses métodos é que eles não suportam uma comunicação bidirecional, ou seja, somente aplicativos de TVDI podem acessar serviços dos dispositivos. Em contrapartida, alguns trabalhos como os apresentados em Tkachenko et al. (2006); Cabrer et al. (2006); Redondo et al. (2007); Yang et al. (2007); Lin et al. (2008) propõem uma interação entre TVDI e HN em um mesmo ambiente, integrando os componentes de software das duas plataformas envolvidas (MHP/DASE - OSGi).

A Figura 4.3 ilustra os dois tipos de interação usados nos trabalhos pesquisados. No modelo inicial de Tkachenko et al. (2004) o provedor OSGi e o provedor de TVDI entregam seus serviços ao operador de difusão. Este então entrega os aplicativos ao STB que os repassa ao gateway OSGi por meio da rede TCP/IP. Na outra abordagem (linhas tracejadas) o provedor OSGi e operador de TVDI podem ser a mesma entidade,

entregando serviços ao dispositivo comum STB-Gateway que os distribui para os dispositivos da *Home Network*.

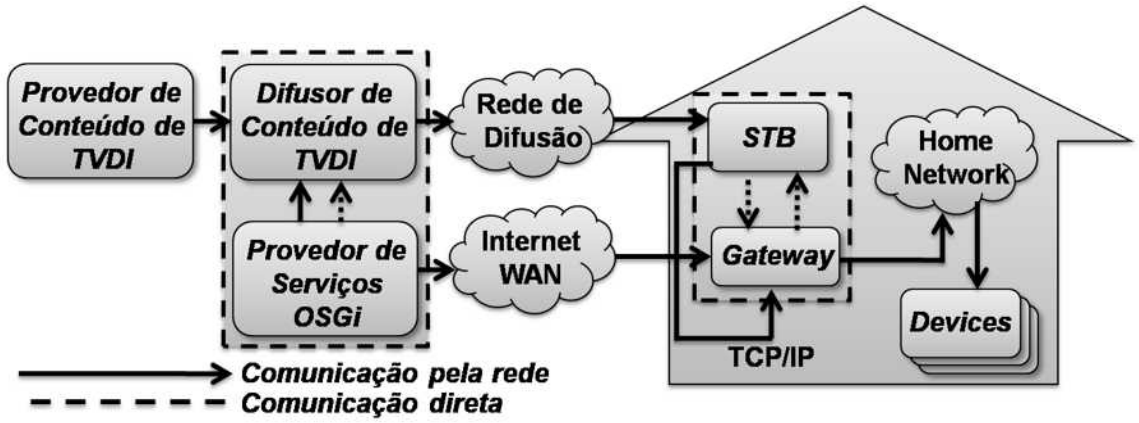


Figura 4.3: Interação entre plataformas de STB e *Home Gateway*.
FONTE: Adaptado de Tkachenko et al. (2004).

O trabalho em Cabrer et al. (2006); Redondo et al. (2007) sugere a criação de uma estrutura híbrida, que contemple características tanto de *xlets* MHP quanto de *bundles* OSGi. Chamada *XBundLET*, essa entidade obedece às especificações que definem *xlets* e *bundles*: ao mesmo tempo em que é controlada por um gerenciador de aplicações, utilizando e liberando recursos e comunicando-se com outros *xlets* via IXC, o *XBundLET* pode prover e/ou utilizar serviços de outros *bundles*, em interação com o *Service Registry*. O *XBundLET* definido segue um modelo de execução como máquina de estado que contém todos os estados de *xlet* e *bundle*. Adicionalmente, como forma de melhoria da proposta apresentada, os autores elaboram um processo de descrição semântica para auxiliar a descoberta e utilização de serviços OSGi no ambiente pelo aplicativo híbrido *XBundLET*. Uma desvantagem desse modelo é a necessidade da criação de uma estrutura sob um ambiente comum MHP-OSGi que por atuar como um *xlet* e um *bundle* exige um complexo modelo para garantir o correto gerenciamento do aplicativo híbrido, principalmente sob questões de ciclo de vida. Some-se a isso a o fato de o desenvolvedor de aplicações TVDI e serviços OSGi precisar conhecer esse novo componente. A Figura 4.4 ilustra o modelo deste trabalho.

A proposta em Yang et al. (2007) analisa a estratégia de implementação de um modelo colaborativo que modifique o mínimo possível as características dos componentes

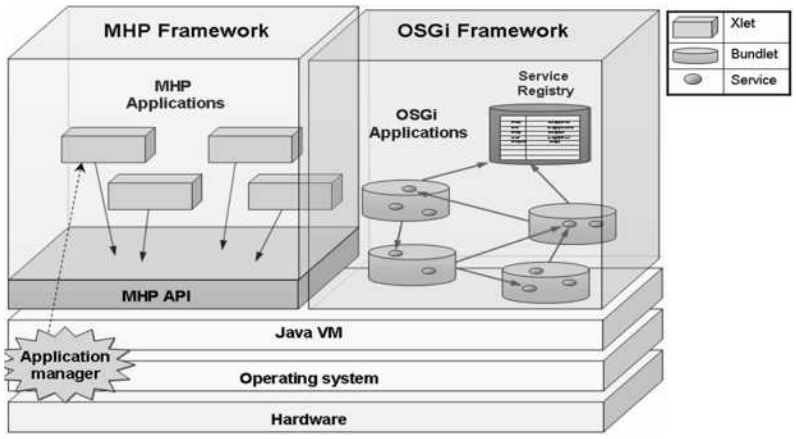


Figura 4.4: Modelo de Colaboração proposto por Cabrer et al. (2006); Redondo et al. (2007). FONTE: Adaptado de Cabrer et al. (2006).

nativos dos dois *frameworks* (MHP e OSGi). Embora ainda se utilize a idéia de criar uma estrutura ponte entre os dois ambientes, esta não se constitui de um componente híbrido e sim de uma entidade que explora as potencialidades da Máquina Virtual Java. A idéia da proposta é a criação de estruturas-ponte entre as duas plataformas (*BunldeContext Bridge*, *XletContext Bridge* e *Collaboration Permission*) que permitam a passagem de parâmetros de contexto (*XletContext*, *IXC Registry* e *BundleContext*) de *xlets* MHP para *bundes* OSGi e vice-versa. Essas estruturas exploram as características dos *Java ClassLoaders*, assim como implementam um mecanismo de segurança para garantir que acessos indevidos não aconteçam. Embora esse modelo seja mais simples que o anterior, a grande desvantagem é que o desenvolvedor tanto de *xlets* como de *bundles* precisa conhecer os novos componentes, além da necessidade de exportar os componentes-ponte como bibliotecas estáticas da JVM, fora do ambiente MHP-OSGi. A Figura 4.5 ilustra o modelo deste trabalho.

O último trabalho analisado, proposto por Lin et al. (2008), apresenta uma abordagem semelhante à de Bae et al. (2006), utilizando componentes sob padrões de projetos *Wrapper* e *Broker* para isolar os domínios TVDI e HN. A diferença entre eles é que no primeiro MHP e OSGi estão em uma mesmo dispositivo. *Xlets* MHP e *bundles* OSGi se comunicam com *proxys* específicos para cada plataforma, implementados dentro de um componente ponte *Broker*. Esse esquema apresenta o *MHP Application Manager* como um serviço do domínio OSGi (*MHPAPPmanager* bundle). Dentro dele está o componente

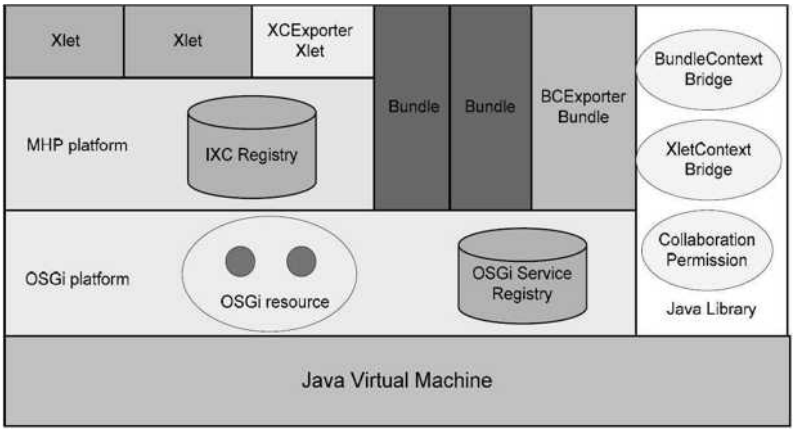


Figura 4.5: Modelo de colaboração de Yang et al. (2007).

FONTE: Adaptado de Yang et al. (2007).

Broker e os *proxys*. Uma vez ativado o *MHPAPPmanager bundle*, é inicializado o gerenciamento do ciclo de vida do *xlet*. Assim como a abordagem de Cabrer et al. (2006) e de Redondo et al. (2007), é necessário modificar componentes MHP e OSGi para dar suporte ao modelo, o que acaba aumentando sua complexidade. A Figura 4.6 ilustra o modelo deste trabalho.

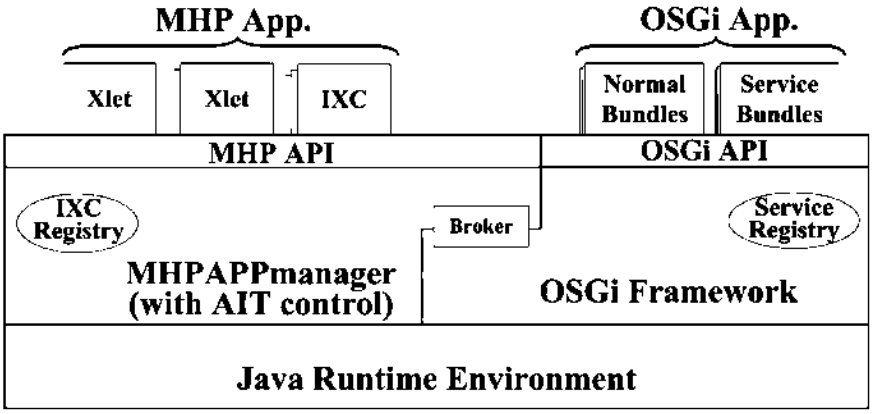


Figura 4.6: Modelo de colaboração de Lin et al. (2008).

FONTE: Adaptado de Lin et al. (2008)

Análise Comparativa

Há duas principais vantagens nos quatro últimos trabalhos: (1) os modelos não estão limitados às falhas de comunicação como em Tkachenko et al. (2004, 2005); e (2) a comunicação é permitida em ambas as direções (tanto aplicações de TVDI acessam

serviços da HN quanto aplicações em outros dispositivos da HN podem acessar serviços da TVDI). A Tabela 4.1 mostra uma comparação entre todos os trabalhos das três categorias discutidas, tomando como base alguns aspectos relevantes, tais como: a tecnologia adotada para as plataformas *Home Network* e TVDI; a interação entre as plataformas que pode ser direta (ambiente comum STB-HG) ou sujeita à rede de comunicação (STB e HG separados); a forma de comunicação entre as aplicações do STB e *Home Gateway* que pode ser unidirecional (apenas aplicações de TVDI utilizam serviços de aplicações de *Home Network*) ou bidirecional (as aplicações dos dois ambientes compartilham serviços entre si).

Publicação	Interação entre os ambientes	Formas de Comunicação	Middleware de TVDI	Framework para HN
Matsubara et al. (2005)	TVDI e HG separados	TVDI para HN (unidirecional)	Nenhum	UPnP e DLNA
Portelli et al. (2008)	TVDI e HG separados	TVDI para HN (unidirecional)	MHP	None
Tkachenko et al. (2004) Tkachenko et al. (2005)	TVDI e HG separados	DASE para OSGi (unidirecional)	NIST RI (DASE)	OSCAR/JEFFREE (OSGi)
Bae et al. (2006)	TVDI e HG separados	ACAP para UMB (unidirecional)	ACAP RI	UMB RI
Tkachenko et al. (2006)	Ambiente comum TVDI-HG	DASE para OSGi (unidirecional)	NIST RI (DASE)	OSCAR (OSGi)
Cabrer et al. (2006) Redondo et al. (2007)	Ambiente comum TVDI-HG	MHP para OSGi (bidirecional)	XleTView (MHP)	OSCAR (OSGi)
Yang et al. (2007)	Ambiente comum TVDI-HG	MHP para OSGi (bidirecional)	XleTView (MHP)	Knopflerfish (OSGi)
Lin et al. (2008)	Ambiente comum iDTV-HG	MHP para OSGi (bidirecional)	XleTView (MHP)	OSCAR OSGi

Tabela 4.1: Comparação entre os trabalhos relacionados.

Este trabalho enquadra-se na terceira das categorias de modelos TVDI-HN. Uma nova arquitetura de componentes de software que habilitam interoperabilidade entre GINGA e OSGi é criada. Propõe-se uma abordagem compatível com Cabrer et al. (2006); Redondo et al. (2007); Yang et al. (2007); Lin et al. (2008) por meio da criação de um

modelo procedural e, adicionalmente, explora-se as capacidades do GINGA declarativo, criando um mecanismo de colaboração entre aplicações GINGA e serviços OSGi, trazendo uma série de novas aplicações e cenários para a TV Digital brasileira. Nos próximos capítulos, projeto, arquitetura, implementação e cenários de testes são descritos a fim de validar o os componentes desenvolvidos.

4.3 Resumo

Este capítulo analisou as principais referências em modelos de convergência STB-HG existentes na literatura. Foram agrupadas referências de acordo com os veículos de pesquisa utilizados e as categorias de cada trabalho, desde aqueles que abordam tecnologias de TV Digital e *Home Neworks*, utilizados nos Capítulos 2 e 3, até os que propõem modelos e arquiteturas de integração entre os dois ambientes. Esta última categoria apresenta os trabalhos dos quais a pesquisa desta Dissertação deriva.

Entre os trabalhos de colaboração TVDI-HN, foram criadas mais 3 subcategorias, com o objetivo de analisar cada abordagem mais especificamente. O primeiro subgrupo engloba um trabalho que centraliza a integração TVDI-HN nas tecnologias de *Home Networks*; o segundo ilustra uma abordagem centrada no padrão de TVDI para gerenciar serviços em *Home Networks*; já a última subcategoria ilustra trabalhos que propõem soluções de comunicação entre *frameworks* de *Home Network* (OSGi) e *middleware* de TV Digital (MHP, DASE). Dentre essas abordagens, as que utilizam as duas plataformas num mesmo ambiente, sem interferência da rede, são analisadas com o objetivo de identificar possíveis características para o modelo criado neste trabalho. Com base nas últimas abordagens de trabalhos relacionados, um modelo de integração GINGA-OSGi será apresentado no próximo capítulo e posteriormente comparado com os modelos apresentados neste capítulo, sob alguns dos parâmetros identificados.

Capítulo 5 - Mecanismo TVDI-HG para a TVDI Brasileira

Neste capítulo é especificada a plataforma de integração TVDI-HG baseada no *middleware* Ginga e no *framework* OSGi. Após identificadas algumas características desejadas para uma plataforma Ginga-OSGi, com base nas análises feitas nos capítulos anteriores, é criado um conjunto de componentes de software sobre o ambiente integrado Ginga-OSGi que permitirá ao programador de aplicações desenvolver aplicativos que integrem aplicações de TV Digital com serviços de dispositivos em redes OSGi. Mais especificamente, serão definidos os dois modelos de colaboração Ginga-OSGi: o primeiro utiliza uma abordagem procedural, GingaJ-OSGi, e o segundo utiliza uma abordagem baseada no *middleware* declarativo Ginga-NCL. Para cada modelo, serão apresentados os elementos de software que o compõem, além dos mecanismos através dos quais esses componentes interagem para permitir a construção de cenários de aplicações.

Para o modelo GingaJ-OSGi procurou-se seguir uma abordagem compatível com os trabalhos de Cabrer et al. (2006); Redondo et al. (2007); Yang et al. (2007) e Lin et al. (2008), eliminando as características indesejáveis de cada modelo e adicionando novas funcionalidades do *middleware* Ginga-J. O modelo baseia-se na construção de componentes-ponte que permitem o compartilhamento de objetos de serviços entre OSGi (registrados no *OSGi Service Registry*) e Ginga-J (armazenados na entidade de registro de objetos, que permite a comunicação entre aplicativos *xlet*, definida para o Ginga-J, o *IXC Registry*). A plataforma GingaJ-OSGi habilita o acesso a serviços OSGi por aplicações Ginga-J e, de modo correspondente, habilita o acesso à objetos Ginga-J por aplicações OSGi.

Para o modelo declarativo, procurou-se explorar algumas funcionalidades do

middleware Ginga-NCL. Por ser um *middleware* declarativo, no qual as especificações das funções e procedimentos do programa são configuradas num arquivo baseado em XML, as possibilidades de exploração da plataforma são bem menores, pois os recursos do *middleware* são reduzidos quando comparado a um *middleware* procedural. Isso de certa forma justifica a menor complexidade do modelo GingaNCL-OSGi em relação ao modelo GingaJ-OSGi. No modelo declarativo, foram estendidas algumas funções de objetos padrão do Ginga-NCL, os *Adapters*, de modo a associá-los aos serviços OSGi, permitindo tanto a exportação de *Adapters* para o domínio OSGi quanto o uso de serviços OSGi pelos *Adapters* Ginga-NCL.

Após a definição dos modelos, são construídos dois mecanismos adicionais à rede OSGi, no sentido de lidar com problemas comuns do *framework*. Os dois mecanismos são: *invocação automática de serviços OSGi*, que lida com o problema de registrar muitos serviços OSGi para uma única referência exportada ao Ginga-J/Ginga-NCL; e *gerenciamento automático de dependências entre serviços* que lida com um problema comum do OSGi, a perda da referência a um serviço em tempo de execução.

5.1 Visão Geral do Modelo

Com relação à classificação de abordagens de integração TVDI-HG definida no capítulo anterior, a proposta desenvolvida aqui pertence à terceira categoria de trabalhos, *colaboração entre as duas plataformas*. Isso quer dizer que o *middleware* de TVDI (Ginga) e o *framework* para *Home Network* (OSGi) estão num ambiente comum *Set Top Box/Home Gateway*. O esquema da Figura 5.1 ilustra o contexto geral do sistema em que está inserida a plataforma. Naquela figura, provedores de serviços OSGi criam as aplicações relacionadas aos dispositivos da residência, que executarão no *Home Gateway*. Ao mesmo tempo, operadores de redes de TV Digital receberão aplicativos a serem entregues às residências que, juntamente com os aplicativos OSGi, serão transmitidos pela rede de difusão. O sinal é decodificado no STB-HG, e cada aplicação será registrada em seu domínio específico: os serviços ou *bundles* serão registrados do lado OSGi enquanto que as aplicações *xlet* ou NCL serão registradas do lado Ginga-J/Ginga-NCL.

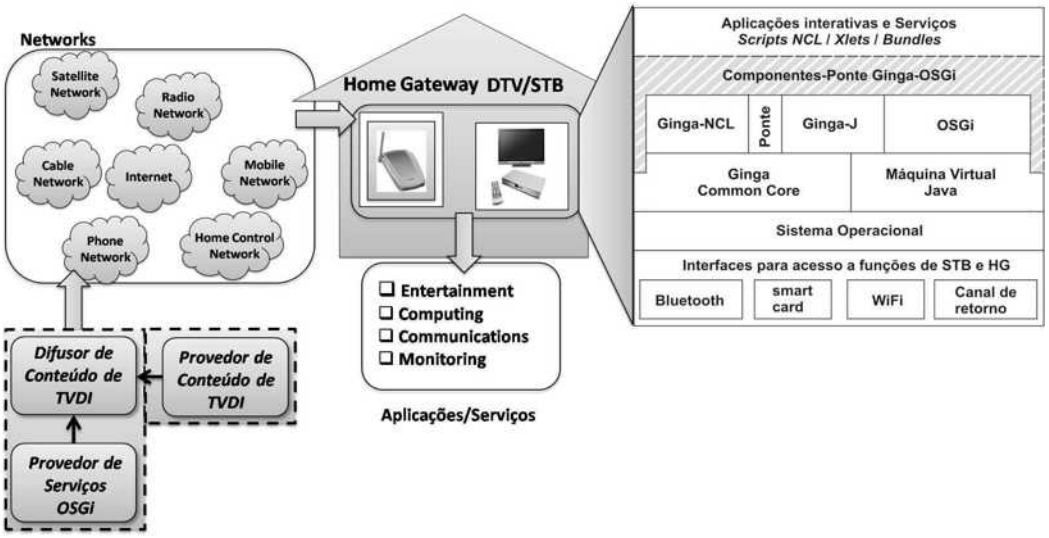


Figura 5.1: Esquema geral de entrega de serviços e aplicativos aos terminais STB-HG.
FONTE: Do Autor.

Uma vez registrados os aplicativos nos dois domínios, os componentes de software da plataforma proposta entrarão em ação para interoperar os componentes registrados de um domínio para o outro. Para isso, foram adotadas duas estratégias separadas: uma que considera componentes de software do Ginga-J e outra que usa componentes do Ginga-NCL. Ainda na Figura 5.1 é ilustrada a arquitetura geral de software do STB-HG, na qual podem ser notadas as camadas do sistema, variando desde as interfaces comuns do STB (canal de retorno, leitor *smart card*, etc) ou *Home Gateway* (suporte à Bluetooth, WiFi, ZigBee, IEEE 802.11, RS-232, etc), passando por sistema operacional, máquina Virtual Java e componentes do núcleo do Ginga e o ambiente comum Ginga-J/Ginga-NCL-OSGi, até o nível dos aplicativos, que permitirão o controle interativo dos dispositivos da *Home Network*, dentro das quatro categorias de aplicações/serviços em *Home Networks* citadas no item 2.2.1.

Os componentes criados, tanto para o Ginga-J quanto Ginga-NCL, devem garantir a execução correta de dois mecanismos padrão de comunicação com o OSGi; são eles:

a) *Aplicações Ginga-J/Ginga-NCL acessam serviços OSGi* - O processo consiste em registrar, em uma entidade apropriada, os serviços providos por componentes OSGi que representam algum dispositivo da *Home Netowrk*. Em seguida, uma estrutura

de conversão da plataforma cria uma referência para o serviço no ambiente de aplicações Ginga-J/Ginga-NCL. Assim, eles podem ser acessados como um conjunto de funcionalidades e interpretados para o usuário, permitindo em último caso que este controle algum dispositivo por meio da TV. O cenário 1 apresentado no capítulo 1 ilustra uma aplicação desse processo.

b) *Aplicações OSGi acessam funcionalidades Ginga-J/Ginga-NCL* - Nesse mecanismo as aplicações Ginga-J (*xlets*) ou componentes Ginga-NCL registram suas funcionalidades em uma entidade definida. A mesma estrutura ponte do item anterior é responsável por converter a funcionalidade em uma estrutura que possa ser registrada na entidade do domínio OSGi e interpretada como um serviço. Assim, aplicativos OSGi acessam funcionalidades Ginga-J/Ginga-NCL como serviços. Um exemplo da aplicação desse processo é ilustrado pelo cenário 2 do capítulo 1.

Conforme será descrito nas próximas seções, cada estratégia de integração apresenta suas características, refletindo as diferenças entre os middlewares Ginga-J e Ginga-NCL. Para o primeiro, foi construída uma plataforma baseada no subconjunto Ginga-J compatível com o GEM, enquanto que para a segunda foram estendidos alguns componentes do núcleo do Ginga-NCL de modo a permitir a busca e registro de serviços na rede OSGi. Apesar das diferenças, a plataforma como um todo deve apresentar algumas características comuns. Entre as mais importantes estão:

- *Permitir o registro e a descoberta de serviços OSGi na rede* - Serviços formam a base do modelo OSGi. Um serviço é uma interface para as operações que podem ser realizadas sobre o dispositivo que representa. Este pode implementar protocolos como UPnP, HomePNA, IEEE 1304, USB, Bluetooth, etc. A descoberta de serviços na rede é etapa chave para a integração com as aplicações de TV Digital.

- *Interoperar serviços OSGi e aplicações Ginga* - Em outras palavras, permitir a troca entre componentes de um domínio para o outro. O ambiente deve permitir que aplicações Ginga acessem objeos OSGi e vice-versa. A princípio isso não é uma tarefa fácil, pois requer esforços no sentido de integrar dois ambientes construídos sobre

modelos totalmente distintos (o modelo orientados a funcionalidades do Ginga-J que deriva do MHP, além do modelo do Ginga-NCL, e o modelo orientado a serviços do OSGi).

- *Implementar um mecanismo de comunicação entre os componentes elementares Ginga e OSGi* - O mecanismo de comunicação depende do modelo adotado. Ele especifica os componentes de software que irão habilitar o acesso a funções de um domínio por elementos do outro domínio.

- *Oferecer um mecanismo para gerenciar dependências entre serviços no domínio OSGi* - O mecanismo de gerenciamento de serviços permite que todas as dependências de serviços entre bundles OSGi sejam resolvidas em tempo de execução. Isso garantirá, por exemplo, que nunca existirá uma referência inconsistente a um serviço OSGi que foi removido em algum momento do sistema.

O ambiente com tais características, dividido em duas implementações distintas é descrito nas seções a seguir.

5.2 Modelo GingaJ-OSGi

A Figura 5.2 ilustra a abordagem procedural do modelo Ginga-OSGi. A arquitetura está em conformidade com os requisitos desejados na seção 5.1, e segue um modelo apoiado por componentes do conjunto de APIs GEM, sendo portanto extensível para outros *middlewares* GEM compatíveis como plataformas MHP ou OCAP. Dessa forma, com as alterações necessárias, os componentes do modelo Ginga-J-OSGi poderão operar sobre outros middlewares.

O modelo procedural segue a mesma linha das abordagens de Cabrer et al. (2006), Redondo et al. (2007), Yang et al. (2007) e Lin et al. (2008). Entretanto, para esse modelo, procurou-se modificar o mínimo possível os dois ambientes de forma que programadores de *xlets* Ginga-J ou de serviços OSGi não precisem conhecer mecanismos complexos da plataforma. O programador trabalhará apenas com os componentes padrão

do Ginga-J e OSGi e os componentes da plataforma farão o trabalho de integrar os dois ambientes.

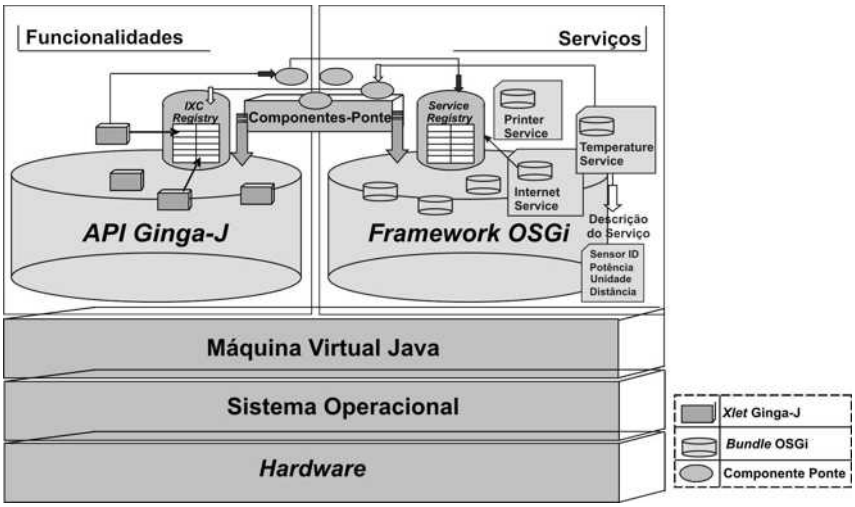


Figura 5.2: Arquitetura geral do modelo GingaJ-OSGi.
FONTE: Do Autor.

Conforme a Figura 5.2, a camada de software sob Ginga-J(GEM)-OSGi apresenta componentes que habilitam o fluxo de objetos de serviços entre os dois domínios seguindo um mesmo padrão. Todo o processo de interação é centrado nas entidades de registros de serviços OSGi e funcionalidades Ginga-J, o *OSGi Service Registry* e o *MHP IXCRegsitry*. Uma vez registrada alguma funcionalidade no *IXCRegistry* um componente notifica o domínio OSGi por meio do registro de um serviço correspondente. Da mesma forma, sempre que um *bundle* registrar seu serviço no *OSGi Service Registry*, uma outra entidade passará essa referência para o domínio do Ginga-J para que ela possa ser acessada por *xlets*. Essas entidades atuam como *Listeners* nos dois ambientes e os objetos são registrados através de uma interface genérica seguindo os padrões de projeto *Proxy* e *Decorator*, formalizados por Gamma et al. (1995): a interface protege o acesso direto a métodos do objeto, oferecendo novos métodos para acessar o objeto com segurança, aproveitando-se das funcionalidades da API *Java Reflection*. Os sete componentes do modelo são descritos a seguir:

- ***GingaJ-OSGiRegister*** - Este componente é responsável por exportar funcionalidades do Ginga-J para o OSGi na forma de serviços. Ele implementa um *Listener*

para os eventos de registro no *IXCRegistry* (bind, rebind ou unbind), ou seja, sempre verifica a ocorrência de um desses eventos no *IXC Registry*. Por meio do *ContextBridge*, o *GingaJ-OSGiRegister* consegue obter o contexto dos dois ambientes e registrar no *Service Registry* o serviço correspondente como um objeto *WrappedProxyObject*. Entretanto, para registrar um serviço no OSGi o *GingaJ-OSGiRegister* deve utilizar um serviço do bundle *GingaJ-OSGiBundle*.

- ***GingaJ-OSGiBundle*** - Este componente disponibiliza no *Service Registry* um serviço através do qual é possível registrar objetos *WrappedProxyObject* referentes a interfaces do Ginga-J. Isso é necessário porque uma vez que o componente *GingaJ-OSGiRegister* não é um *bundle* ele não está habilitado a registrar diretamente serviços no OSGi.

- ***OSGi-GingaJRegister*** - Este componente dá suporte ao processo inverso, ou seja, implementa um *Listener* para registro de serviços no OSGi e acessa o *ContextBridge* para exportar um *WrappedProxyObject* correspondente ao serviço OSGi para o domínio Ginga-J, por meio de um registro no *IXCRegistry*. Nesse caso, não há nenhuma restrição para que o serviço OSGi seja registrado no IXCRegistry como um *WrappedProxyObject*.

- ***XletContextExporter*** - Um *xlet* responsável por exportar um objeto *XletContext* (em *javax.tv.xlet*) para o *ContextBridge*. O objeto exportado servirá como ponto de entrada para elementos do OSGi acessarem o domínio Ginga-J.

- ***BundleContextExporter*** - Um *bundle* responsável por exportar um objeto *BundleContext* (em *org.osgi.framework*) para o *ContextBridge*. O objeto exportado servirá como ponto de entrada para elementos do Ginga-J acessarem o domínio OSGi.

- ***ContextBridge*** - Este componente atua como uma ponte que permite o acesso a um contexto genérico tanto de *xlet* quanto de *bundle*. O *ContextBridge* recebe os contextos de *XletContextExporter* e *BundleContextExporter* assim que o sistema é inicializado, disponibilizando esses contextos para outros componentes do sistema.

- ***WrappedProxyObject*** - Um componente que protege a implementação da interface do serviço OSGi ou da funcionalidade Ginga-J. Através de invocação indireta de métodos usando a *API Reflection*, este componente (que implementa uma interface com métodos padrão, *WrappedInterface*) é registrado de fato no *IXCRegistry* ou no OSGi *Service Registry*. Sempre um *WrappedProxyObject* esterá associado a um objeto do OSGi ou do Ginga-J. Toda vez que um *xlet* precisar invocar um serviço OSGi, ele o estará invocando indiretamente através do *WrappedProxyObject*. Da mesma forma, sempre que um bundle precisar utilizar uma funcionalidade de *xlet* ele a utilizará na forma de *WrappedProxyObject*.

Esses componentes são a base do modelo GingaJ-OSGi. Eles operam de modo a garantir dois mecanismos de interação entre as plataformas: (1) *serviços OSGi devem ser acessados por aplicações Ginga-J* e (2) *aplicações OSGi devem acessar funcionalidades exportadas pelo Ginga-J*. Os dois mecanismos são descritos em detalhes a seguir.

5.2.1 Acessando serviços OSGi através de aplicações Ginga-J

No cenário de acesso a serviços OSGi por aplicações Ginga-J, ilustrado na Figura 5.3, estão presentes *XletContextExporter*, *BundleContextExporter*, *ContextBridge* e *OSGi-GingaJRegister*. Uma vez inicializado o sistema, o gerenciador de aplicações define um mecanismo baseado em *Java ClassLoaders* para ativar objetos *XletContextExporter* e *BundleContextExporter*. Ao inicializar o *xlet* e o *bundle* seus contextos são passados ao objeto *ContextBridge* que permanece ativo até que o *framework* seja desativado.

Uma vez inicializado o componente ponte, o *OSGi-GingaJRegister* é ativado e permanece na espera pela notificação de registro de algum serviço no OSGi. Quando um *bundle* registra seu serviço no *Service Registry* (passo 1), uma referência (objeto *ServiceReference*) é recuperada pelo *OSGi-GingaJRegister* (passo 2). Por meio do *ServiceReference* é possível acessar algumas propriedades do serviço utilizando uma consulta expressa em linguagem LDAP (*Lightweight Directory Access Protocol*) (Howes, 1997). Essas propriedades servirão para que posteriormente o *xlet* consiga encontrar o objeto correpondente no IXC. O componente *OSGi-GingaJRegister* possui um método

chave que passa o objeto de serviço OSGi a uma instância de *WrappedProxyObject* (passo 3). Nesse ponto o *OSGi-GingaJRegister* acessa um contexto de *xlet* armazenado no *ContextBridge* para exportar o *WrappedProxyObject* ao *IXCRegistry* (passo 4). Dessa forma, todo acesso de um *xlet* Ginga-J ao objeto de serviço é protegido pelo objeto *WrappedProxyObject*. Por fim, o *xlet*, após recuperar o objeto do *IXCRegistry* (passo 5), precisa apenas saber o nome e os tipos dos parâmetros que devem ser passados na chamada ao método e invocação do serviço.

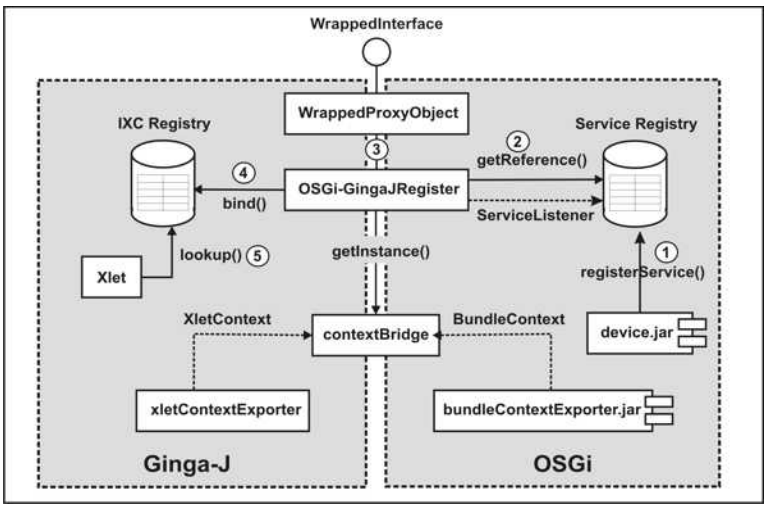


Figura 5.3: Exportando serviços OSGi para o domínio Ginga-J.
FONTE: Do Autor.

5.2.2 Acessando funcionalidades Ginga-J através de aplicações OSGi

O segundo mecanismo, esquematizado na Figura 5.4, consiste em registrar funcionalidades de *xlet* no *OSGi Service Registry*, permitindo que *bundles* OSGi utilizem funções de *xlets* Ginga-J.

Uma vez ativados os componentes *XletContextExporter*, *BundleContextExporter* e *ContextBridge* um *xlet* deve registrar o objeto de uma interface no *IXCRegistry* (passo 1). Nesse ponto um componente *GingaJ-OSGiRegister* implementa um *Listener* para os eventos sobre o *IXCRegistry*. Esses eventos podem ser de três tipos: BINDED, REBINDED e UNBINDED que notificam se uma interface foi registrada, atualizada

ou removida do *IXC Registry*, respectivamente (passo 2). A partir daí o processo é semelhante ao do tópico anterior: o *GingaJ-OSGiRegister* ativa um *WrappedProxyObject* para proteger os métodos do objeto registrado no *IXCRegistry* (passo 3). Após isso o *GingaJ-OSGiRegister* utiliza um contexto de *bundle* armazenado no *ContextBridge* para acessar um serviço do *GingaJ-OSGiBundle* que permite registrar um *WrappedProxyObject* como um serviço OSGi (passo 4).

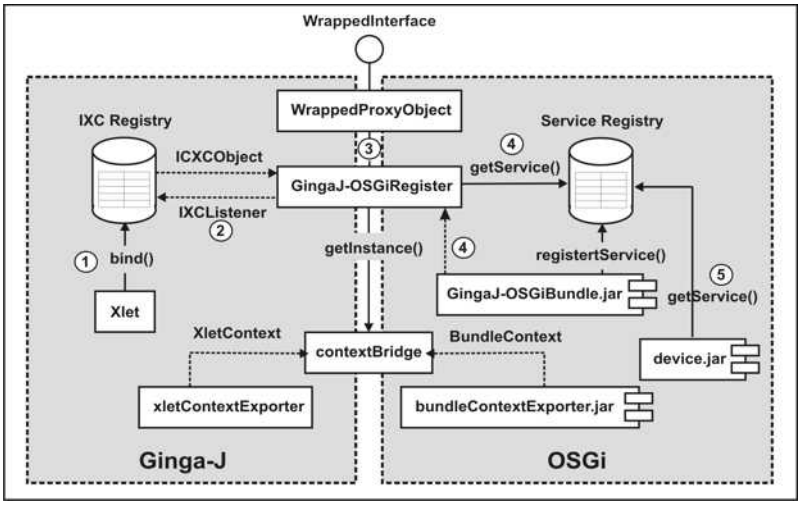


Figura 5.4: Exportando funcionalidades Ginga-J como serviços OSGi.
FONTE: Do Autor.

Como o *GingaJ-OSGiBundle* fornece apenas um serviço para todos os objetos *WrappedProxyObject* provenientes do ambiente Ginga-J, é necessário um mecanismo para gerenciar esses objetos. Para cada objeto *WrappedProxyObject* que chega (correpondente a um evento de BIND no *IXCRegistry*), o *GingaJ-OSGiBundle* o adiciona num repositório de objetos no OSGi. Da mesma forma, sempre que um objeto é removido do *IXCRegistry*, o *GingaJ-OSGiBundle* busca a referência correspondente no repositório de objetos e também a remove. Isso garante a sincronia no registro/remoção de serviços entre os dois domínios.

Na última etapa, um *bundle* OSGi, para acessar o serviço correspondente à funcionalidade Ginga-J, precisa saber como obter um *ServiceReference* para o *WrappedProxyObject* (passo 5). Para acessar um método do objeto do *IXCRegistry* o *bundle* passa o nome e o tipo para o *WrappedProxyObject* que o invoca, via *Java Reflection*.

5.3 Modelo GingaNCL-OSGi

Assim como o modelo procedural, a camada de software sob a plataforma GingaNCL-OSGi apresenta componentes que habilitam o fluxo de objetos entre os dois domínios. A arquitetura desta abordagem é ilustrada na Figura 5.5. Nela pode-se observar o ambiente comum STB-HG baseado no *middleware* declarativo Ginga-NCL. Logo acima da camada da Máquina Virtual Java estão os componentes de software do Ginga-NCL que serão modificados para a criação da plataforma. Esses componentes se encontram no núcleo comum do Ginga (Ginga-CC). Na última camada estão as aplicações. Estas são constituídas de scripts NCL ou aplicativos OSGi, que serão gerenciados pelos componentes do ambiente GingaNCL-OSGi.

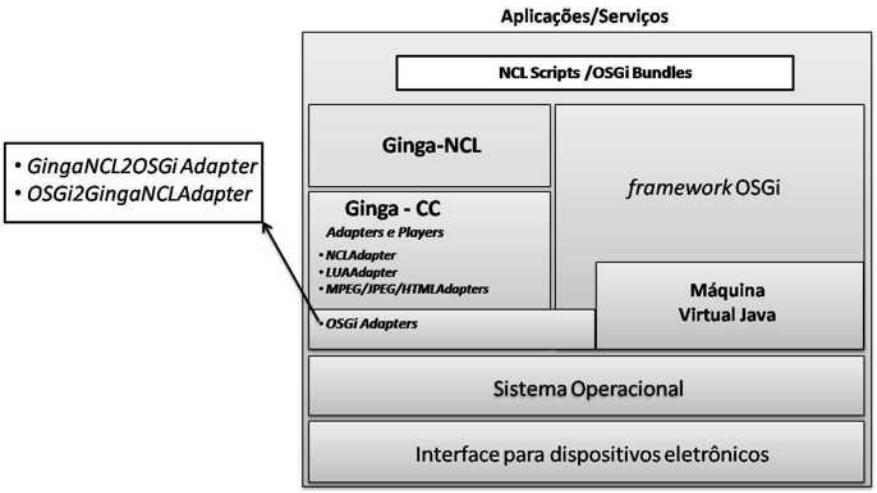


Figura 5.5: Arquitetura do ambiente comum STB-HG baseado no *middleware* declarativo Ginga-NCL.

FONTE: Do Autor.

Nesse modelo, dois *Adapters* Ginga-NCL fazem o papel de componentes-ponte. Na arquitetura do Ginga-NCL, um *Adapter* é um componente que funciona como um container (uma região na qual serão adicionados outros objetos) para a apresentação de mídias no documento audiovisual interativo. Cada *Adapter* possui uma região específica da tela onde será exibida a mídia, sendo associado a um componente *Player*, que define as propriedades da mídia. O par (*Adapter*, *Player*) é a base para a apresentação de mídias no documento NCL. Por exemplo, para a interpretação de scripts NCL que realizarão

alguma tarefa no sistema, existem *Adapters* que traduzem as instruções no documento para ações no middleware; para a apresentação de mídias como texto, áudio e vídeo, existem *Adapters* e *Players* correspondentes; para a integração com conteúdo procedural, sejam scripts LUA ou aplicações *xlet* do domínio Ginga-J, existem *Adapters* e *Players* para a interpretação dos códigos.

Os *Adapters* criados para dar suporte ao registro e busca de serviços OSGi seguem os mesmo padrão que os demais *Adapters* para as mídias comuns. Cada *Adapter* implementa mecanismos para encontrar serviços específicos no OSGi, bem como para registrar funções do Ginga-NCL (como uma área da tela da TVDI) como serviços no OSGi. Para tal, os *Adapters* fazem uso de outros componentes adicionais que garantem a execução completa do modelo. Ao todo o modelo possui cinco componentes. São eles:

- ***GingaNCL2OSGiAdapter*** - Esse componente gerencia o acesso a serviços OSGi, apresentando as informações dos dispositivos na tela da TVDI. Para isso o *GingaNCL2OSGiAdapter* recupera um contexto de *bundle* armazenado no objeto *ContextBridge*. Pelo lado do Ginga-NCL, o *GingaNCL2OSGiAdapter* utiliza um componente *Player* do Ginga-NCL para gerenciar as informações provenientes do OSGi.

- ***OSGi2GingaNCLAdapter*** - Esse componente habilita interações entre aplicações OSGi e conteúdo declarativo NCL. O *GingaNCL2OSGiAdapter* implementa uma interface que será registrada como um serviço que será passada ao OSGi. Essa interface deverá representar o *OSGi2GingaNCLAdapter* no domínio OSGi. Entretanto, como o *OSGi2GingaNCLAdapter* não é um *bundle*, ele não está habilitado a registrar serviços no OSGi. Para isso é utilizado um serviço do *OSGi2GingaNCLBundle*.

- ***ContextBridge*** - Esse componente tem papel semelhante ao correspondente do modelo procedural: armazena contextos de objetos. Porém apenas o contexto de *bundle* é armazenado, já que não existem estruturas de contexto no domínio Ginga-NCL. Um objeto *ContextoBridge* é sempre acessado pelos *Adapters* do sistema.

- ***BundleContextExporter*** - Esse componente tem a mesma função que o correspondente procedural: exporta um contexto de *bundle* para o *ContextBridge*.

- ***OSGi2GingaNCLBundle*** - Esse componente disponibiliza um serviço no ambiente que permite o registro de componentes do Ginga-NCL como serviços no OSGi. De forma semelhante ao modelo procedural, apenas um serviço do *OSGi2GingaNCLBundle* será registrado e o *bundle* deve gerenciar todos os objetos que vêm do domínio Ginga-NCL por meio do *OSGi2GingaNCLAdapter*.

Da mesma forma que no modelo procedural, os componentes da plataforma GingaNCL-OSGi operam para garantir dois mecanismos de colaboração distintos, sobre os quais poderão ser construídos alguns cenários de aplicações: (1) *serviços OSGi devem ser acessados por elementos do Ginga-NCL* e (2) *componentes do domínio Ginga-NCL devem ser acessados por bundles OSGi.* A seguir uma descrição dos dois mecanismos.

5.3.1 Acessando serviços OSGi através de componentes Ginga-NCL

Nesse mecanismo, ilustrado na Figura 5.6, estão presentes *GingaNCL2OSGiAdapter*, *ContextBridge* e *BundleContextExporter*.

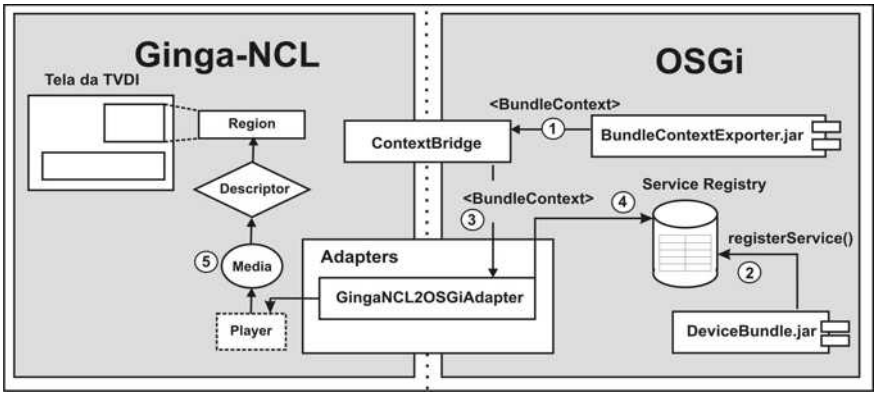


Figura 5.6: Acesso à serviços OSGi por elementos do domínio Ginga-NCL.
FONTE: Do Autor.

Uma vez inicializado o sistema, o *ContextBridge* recebe o contexto de *BundleContextExporter* (passo 1). Após isso, um *bundle* disponibiliza seu serviço no

OSGi Service Registry (passo 2). Desse modo, para obter a referência ao serviço do *bundle* registrado (um objeto *ServiceReference*) o *GingaNCL2OSGiAdapter* acessa o *ContextBridge* (passo 3), recupera o contexto e faz a busca pelo serviço (passo 4). No passo seguinte, o *GingaNCL2OSGiAdapter* utiliza o componente *Player* associado para exportar algumas propriedades do serviço para uma região da tela da TVDI (passo 5). No documento NCL, o programador apenas define uma referência ao *GingaNCL2OSGiAdapter*, especificando ao componente *Player* em que momento as informações do serviço OSGi serão apresentadas na tela.

5.3.2 Acessando componentes Ginga-NCL através de aplicações OSGi

Nesse mecanismo, ilustrado na Figura 5.7, além de *ContextBridge* e *BundleContext* estão presentes os componentes *OSGi2GingaNCLAdapter* e *OSGi2GingaNCLBundle*.

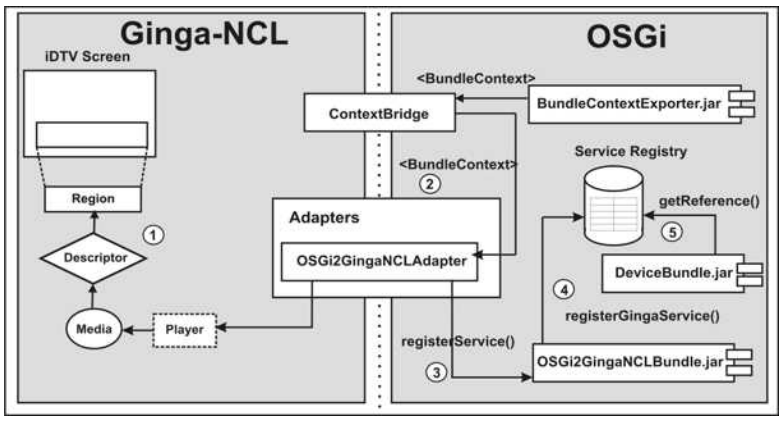


Figura 5.7: Acesso à componentes do Ginga-NCL por bundles OSGi.
FONTE: Do Autor.

O *OSGi2GingaNCLAdapter* implementa uma interface que será passada ao OSGi. Essa interface pode estar associada a alguma propriedade de um *Adapter* ou de uma região da TVDI (passo 1). Para exportar a interface para o *framework*, o *OSGi2GingaNCLAdapter* acessa o contexto do *ContextBridge* (passo 2). Dessa forma ele consegue invocar o serviço registrado pelo *OSGi2GingaNCLBundle* (passo 3). Com esse serviço é possível registrar a interface do *Adapter* no OSGi *Service Registry* (passo 4). No

último passo, um *bundle* acessa o OSGi e busca pelo serviço de *OSGi2GingaNCLBundle*. Neste serviço será encontrado o objeto proveniente do domínio Ginga-NCL (passo 5).

5.4 Mecanismos adicionais à rede OSGi

Nesta seção são apresentadas algumas soluções agregadas à plataforma de modo a facilitar a exportação de serviços OSGi para os domínio Ginga-J/Ginga-NCL. É importante ressaltar que os mecanismos não visam resolver possíveis problemas decorrentes do modelo Ginga-OSGi. São apenas estratégias que auxiliam na organização dos componentes do domínio OSGi, abrangendo questões de projeto da rede OSGi, como algumas discutidas por Gong (2001b). Ao agregar esses dois mecanismos na plataforma, espera-se obter dois resultados principais: (1) o processo de obtenção de serviços OSGi por aplicações Ginga deve ocorrer de um único modo: ao solicitar um serviço OSGi, um componente Ginga acessa apenas um elemento OSGi exportado que busca o serviço solicitado; e (2) um componente Ginga deve sempre saber quando um serviço OSGi está ou não disponível: não deve haver problemas de referência inconsistente a serviços OSGi. Os dois mecanismos são descritos a seguir.

5.4.1 Invocando automaticamente serviços OSGi

Para contemplar os dois cenários gerais Ginga-OSGi e OSGi-Ginga construiu-se uma rede OSGi sob os componentes do modelo (para cada dispositivo da rede, há um *bundle* associado com API's apropriadas para acesso às funções do dispositivo). Conforme mostra a Figura 5.8 esta rede centraliza os serviços de todos os *bundles* num componente único (*home.jar*), que gerencia o registro de todos os serviços dos *bundles* da rede.

O *home.jar* é o componente que terá seu serviço exportado ao domínio Ginga (na forma de um *WrappedProxyObject* no Ginga-J ou por meio de um *Adapter* do Ginga-NCL). O *bundle* possui um *Listener* para eventos específicos no ambiente OSGi. Quando um *bundle* qualquer registra seu serviço no *Service Registry*, o *ServiceListener* do *home.jar* consulta o *ServiceReference* para verificar as propriedades do serviço (como o tipo de serviço) por meio de uma consulta LDAP no objeto *Properties* associado ao serviço (o

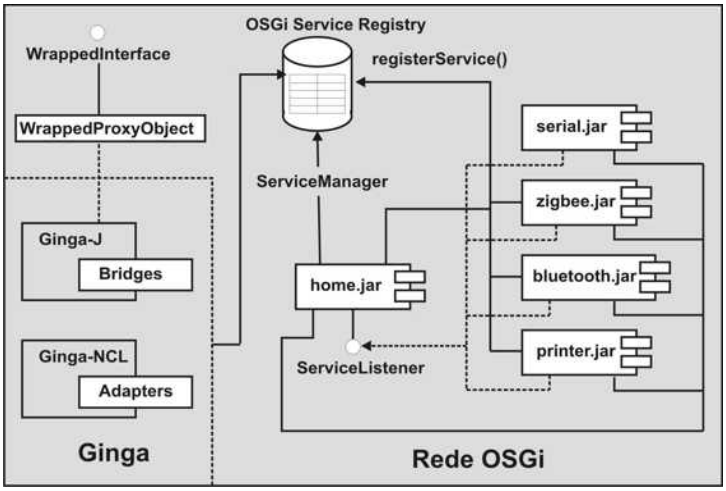


Figura 5.8: Modelo para invocação automática de serviços OSGi.

FONTE: Do Autor.

home.jar verifica a seguinte propriedade: "*serviceType* = HOME SERVICE"). Uma vez verificada a propriedade, o *home.jar* armazena todas as informações sobre o objeto de serviço (como métodos, parâmetros e tipos) em um repositório de objetos. O *home.jar* possui apenas uma interface (*ServiceManager* interface) que registra um serviço a ser exportado ao domínio Ginga.

O objeto de serviço *ServiceManger* é exportado para o *IXCRegistry* (caso o modelo utilize o Ginga-J) ou para os Adapters do Ginga-NCL. Assim, no primeiro caso, é criado apenas um objeto *WrappedProxyObject* e qualquer cliente *xlet* não precisa saber informações específicas sobre cada um dos *bundles* da rede OSGi, pois ele pode obtê-las automaticamente do *home.jar*.

5.4.2 Gerenciando automaticamente dependências entre serviços

Com o objetivo de proporcionar melhorias ao mecanismo de registro de serviços no OSGi, foi adicionado um modelo de gerenciamento automático de dependências entre os serviços OSGi. O problema de dependências entre serviços OSGi é comum em qualquer implementação da plataforma. Ele surge principalmente pela natureza do ambiente OSGi, no qual as aplicações (*bundles*) são componentes independentes uns dos outros que registram e removem seus serviços do ambiente sem se preocupar se o serviço está sendo utilizado ou não. Hoje existem algumas abordagens para lidar com esse problema.

Uma típica situação ocorre quando um *bundle* "consumidor" de um serviço não possui controle sobre o mesmo, pois o *bundle* "provedor" pode realizar operações em tempo de execução sobre o serviço que podem modificá-lo ou torná-lo indisponível, o que pode criar uma referência inconsistente. Por exemplo, suponha que na plataforma criada um *xlet* utilize um serviço de impressão. Quando o serviço torna-se indisponível por falta de papel ou problemas na impressora, deve haver um mecanismo para informar ao *xlet* da falta do serviço.

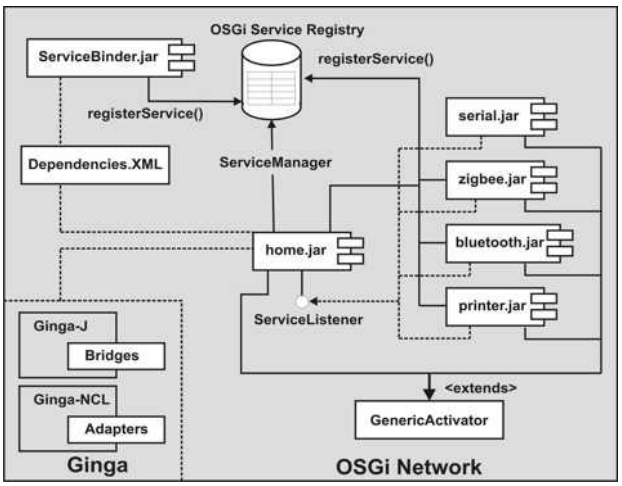


Figura 5.9: Modelo para gerenciamento de dependências entre serviços.
FONTE: Do Autor.

Gong (2001b) sugere algumas soluções para esse problema, mas todas criam uma carga grande de código repetido em cada *bundle* onde é necessário gerenciar tais dependências. A solução para o problema, adotada nesta plataforma, foi a utilização do *Dependency Manager Service Binder* (Binder, 2009), que simplifica a tarefa de escrever código de gerenciamento de dependências e reduz a complexidade para criar serviços no ambiente OSGi (Cervantes e Hall, 2003). Todas as dependências são expressas em formato XML e resolvidas pelo *Service Binder* e o *bundle* cliente (no caso *home.jar*) deve apenas estender uma classe (*GenericActivator*). A Figura 5.9 ilustra o modelo no esquema geral da plataforma.

Os detalhes da implementação dos mecanismos de invocação e gerenciamento automático de dependências serão analisados no Capítulo 6.

5.5 Resumo

Neste capítulo foram apresentados o modelo para a plataforma integrada TVDI-GR baseada no middleware brasileiro Ginga e no framework OSGi. Dois modelos distintos que refletem as duas abordagens adotadas foram criados: um que utiliza componentes do middleware procedural Ginga-J, dentro do subconjunto de componentes GEM compatível, e outro que utiliza componentes do middleware declarativo Ginga-NCL. Para cada modelo, dois mecanismos de interação entre as plataformas foram ilustrados.

Para o modelo procedural foram exploradas as principais entidades de registro de objetos de cada domínio (*IXCRegistry* para o Ginga-J e *Service Registry* para o OSGi), criando uma maneira padrão de registrar objetos de serviços nessas entidades. O modelo foi baseado em padrões de projeto como *Proxy* e *Decorator* para proteger os objetos de cada domínio, registrando apenas um componente-ponte com operações de acesso aos objetos do Ginga-J e do OSGi.

Já o modelo declarativo, dadas as características específicas do *middleware* Ginga-NCL, que oferece menos funcionalidades para agregar novos componentes à plataforma (pois trata-se de um *middleware* declarativo onde a base de programação não são componentes e sim *scripts* NCL) foi criado um modelo menos complexo. Os serviços OSGi foram tratados como mídias do *framework*, suporte este dado pelos componentes *Adapters* do Ginga-NCL. Para cada serviço OSGi o modelo define um *Adapter* correspondente. Do mesmo modo é possível exportar *Adapters* para o OSGi na forma de serviços.

Por fim, foram adicionados dois mecanismos à rede OSGi, um para tratar do problema do registro de muitos serviços na plataforma (a estratégia concentra todos os serviços num só componente) e outro para lidar com a questão das dependências em tempo de execução entre serviços.

No próximo capítulo serão descritos os processos para a construção de cada um dos componentes do modelo Ginga-OSGi, considerando as duas abordagens adotadas. Com isso espera-se oferecer nos Capítulos 5 e 6 a especificação da plataforma necessária para o projeto e construção dos cenários dos estudos de caso do Capítulo 7.

Capítulo 6 - Implementação da Plataforma

Neste capítulo serão apresentados os mecanismos para implementação dos modelos procedural e declarativo de interação do *middleware* Ginga com o *framework* OSGi analisados no capítulo anterior. São descritos ambiente de execução e as ferramentas que foram utilizadas, configuradas e modificadas para a construção dos componentes-ponte Ginga-OSGi descritos em cada modelo.

Para o modelo Ginga-J–OSGi será apresentada uma visão do ambiente comum *XleTView-Knopflerfish*, como plataforma de software para o ambiente STB-HG; são detalhados os sete componentes do modelo e representadas as principais seqüências de interação entre eles de forma a garantir o processo de colaboração GingaJ-OSGi. O código de alguns componentes será apresentado, dando destaque às principais funções de cada um.

De forma semelhante, para o modelo declarativo será descrito o processo de configuração da plataforma de software STB-HG baseada na implementação de referência do Ginga-NCL (*Ginga-NCL Emulator*) e no *Knopflerfish*. Serão descritos os componentes e os dois processos de interação entre eles, sob o ponto de vista dos principais mecanismos de implementação criados.

Por fim, os mecanismos adicionais à rede OSGi (*Invocação automática de serviços* e *Gerenciamento automático de dependências entre serviços*) também serão apresentados por meio de seus principais componentes e a maneira como eles são integrados à plataforma Ginga-OSGi.

6.1 Implementação Procedural

Tanto para a implementação procedural quanto para a declarativa o protótipo foi construído sob uma plataforma baseada em PC, com as seguintes configurações: processador Intel Pentium 1.3GHz, 512MB RAM, 80GB HD, com as interfaces WiFi, Serial RS-232 e USB. O Sistema Operacional utilizado foi *Windows XP SP2*, com o *Java 2 Platform Standard Edition*, versões 1.4.2_17, 1.5.0, e 1.6.0_07. Como ferramenta de suporte à programação Java foi utilizado o Eclipse IDE (Eclipse, 2009). Como ferramenta Ginga-J GEM compatível adotou-se o XleTView 0.3.6 (XleTView, 2009), no qual foram adicionadas as seguintes funcionalidades principais, descritas nos próximos tópicos: (1) suporte à *Inter-Xlet Communication*, através da IR *Java CDC Personal Profile* (Sun, 2009b) e um modelo para *Listeners* de eventos sobre o IXC. Como ambiente para gerenciamento de serviços OSGi utilizou-se a implementação de referência Knopflerfish 3.3.6 (Knopflerfish, 2009).

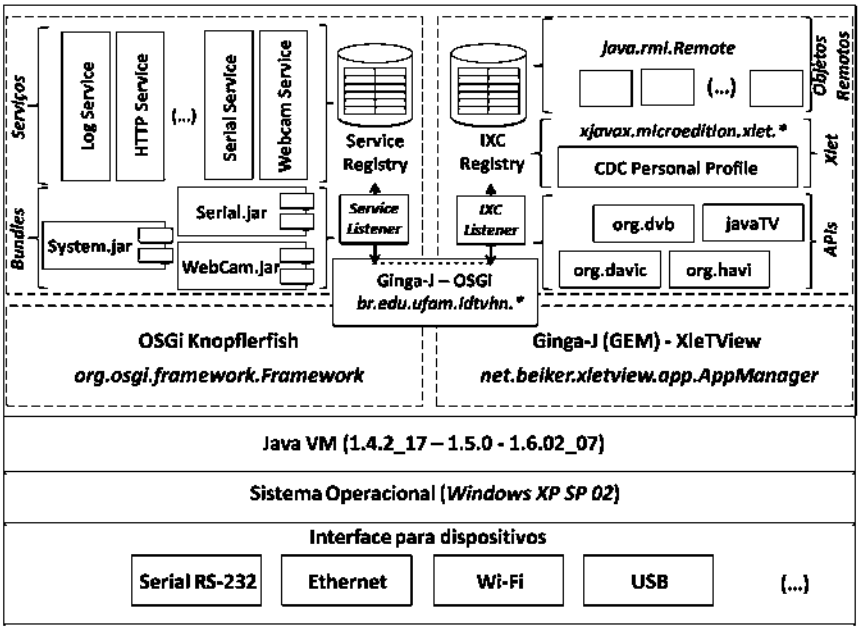


Figura 6.1: Plataforma de Suporte ao Ambiente Ginga-J–OSGi e tecnologias.
FONTE: Do Autor.

A Figura 6.1 ilustra um modelo que agrupa as tecnologias utilizadas na plataforma STB-HG Ginga-J–OSGi. Na figura são identificados os componentes centrais de suporte à plataforma. Da perspectiva do OSGi, tem-se o ambiente Knopflerfish,

cujo framework (em *org.osgi.framework.Framework*) controla o ciclo de vida dos *bundles* e gerencia o resgitro/remoção de serviços no *Service Registry* (representado pelo componente *ServiceRegistration*, em *org.osgi.framework*). Uma vez registrados no OSGi, os serviços serão exportados como objetos do domínio Ginga-J (utilizando os componentes da plataforma, no pacote *br.ufam.tvdihn.**).

Ainda na Figura 6.1, no ambiente Ginga-J, representado pelo XleTView, o gerenciador de aplicações (em *net.beiker.xletview.app.AppManager*) utiliza API's do sistema (*havi*, *javatv*, *cdc personal profile*, *nanoxml*) para controlar a execução dos *xlets* (em *xjavax.microedition.xlet.**), os quais podem registrar objetos *java.rmi.Remote* no *IXCRegistry* (em *com.sun.xlet.ixc.IXCRegistry*) que repassa esses objetos à plataforma, para serem exportados ao *OSGi Service Registry*. A seguir serão apresentados os procedimentos adotados para permitir tal integração Ginga-J–OSGi.

6.1.1 Configuração do Ambiente

Ambiente OSGi Knopflerfish

O Knopflerfish 3.3.6 é uma implementação de referência OSGi R3 (*Release 3*) que oferece um ambiente para a execução de pacotes de aplicativos (*bundles*) e o compartilhamento de serviços entre eles (Knopflerfish, 2009). A Figura 6.2 ilustra a interface do Knopflerfish. Todos os componentes são tratados como *bundles* ou serviços. Para o próprio *framework*, há um *bundle* para cada funcionalidade. Por exemplo, o *SystemBundle* é o componente principal do *framework* que ativa todos os outros *bundles* da plataforma; o *ConsoleBundle* é o *bundle* que permite utilizar recursos de linha de comando (o *prompt*) para instalar/remover outros *bundles* e o *Remote Shell GUI Bundle* é o *bundle* responsável pela criação da interface gráfica que permite a visualização das propriedades de todos os *bundles*/serviços instalados na plataforma.

Como implementação de referência, o Knopflerfish deve prover algumas funções padrão exigidas pelo OSGi. Para a implementação R3, além de funções básicas para gerenciamento do sistema (componentes do grupo funcional *org.osgi.framework*) o *framework* provê funções (na forma de serviços) para: (1) gerenciamento de

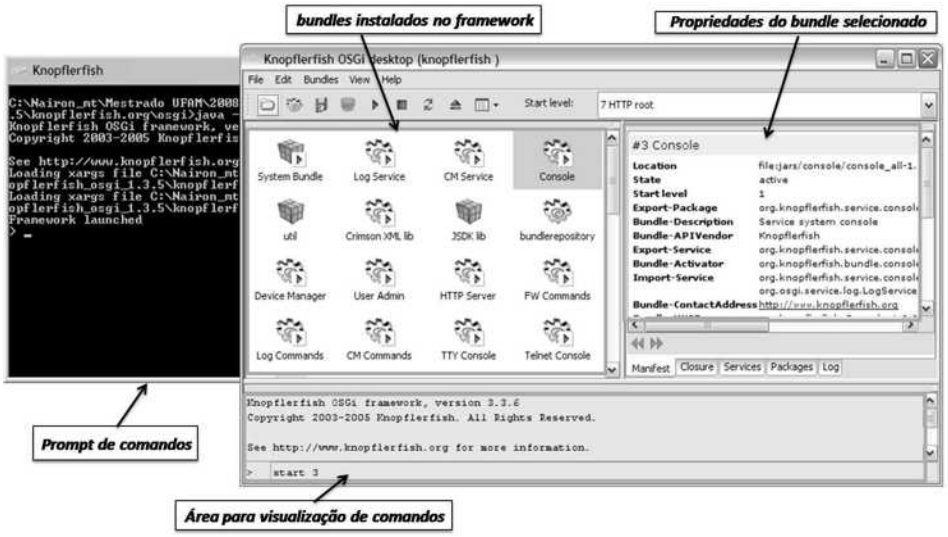


Figura 6.2: Interface do OSGi Knopflerfish.

FONTE: Do Autor.

pacotes que serão exportados ao ambiente (grupo *org.osgi.service.packageadmin*); (2) gerenciamento de permissões (em *org.osgi.service.permissionadmin*); (3) tratamento de URLs (em *org.osgi.service.url*) e monitoramento/rastreamento de serviços (em *org.osgi.util.tracker*). Alguns serviços adicionais como serviços declarativos são exigidos apenas em implementações OSGi R4 ou superiores (OSGi, 2009a) e portanto não estão presentes no Knopflerfish.

A Figura 6.3 ilustra a estrutura de projeto do Knpflerfish, destacando os pacotes OSGi R3, além de grupos de componentes específicos do framework (em *org.knopflerfish.framework*) que apresentam funções para carregamento de classes, instalação de *bundles*, registro de serviços, gerenciamento de arquivos e memória, entre outras. Entre os principais componentes desse grupo, destaca-se *org.knopflerfish.framework.Main*, que representa a classe principal do ambiente. A partir dela, o *framework* é iniciado e os *bundles* e serviços tanto de sistema como os já instalados na plataforma são ativados.

Ambiente Ginga-J XleTView

A implementação de referência (IR) XleTView 0.3.6 é o ambiente para a simulação dos componentes Ginga-J do modelo. Originalmente, o XleTView é uma IR incompleta do MHP, apresentando funções para gerenciamento de recursos e ciclo de vida de aplicações,

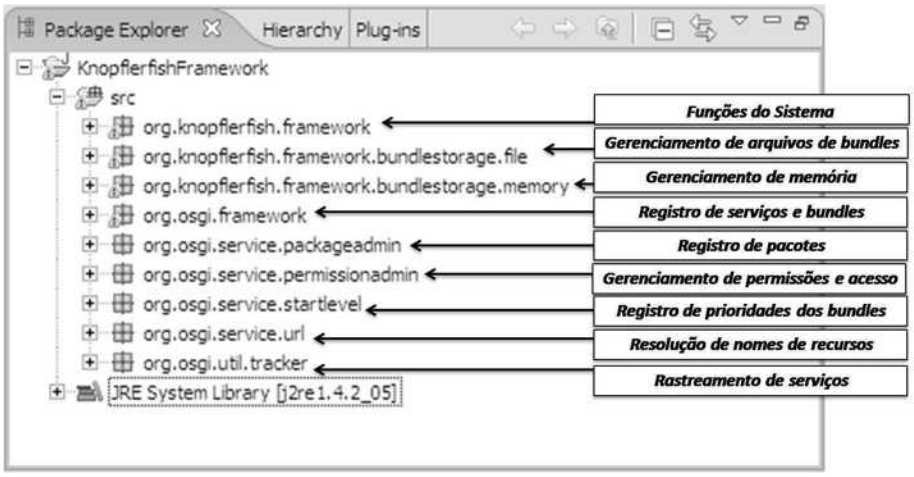


Figura 6.3: Estrutura do Projeto Knopflerfish. Principais grupos de componentes.

FONTE: Do Autor.

gerenciamento de componentes gráficos, suporte à integração com vídeo (via *Java Media Framework*) e parser XML (via *nanoxml*). Entretanto o emulador não apresenta funções importantes do MHP como suporte ao tratamento de informações de serviços (informações no *Transport Stream*) e mecanismo para comunicação entre aplicativos *xlet*, o *Inter-Xlet Communication*, além de apresentar uma implementação incompleta de alguns componentes gráficos (alguns componentes HAVi e DAVIC não presentes). A Figura 6.4 ilustra a interface principal do emulador, com uma área para a apresentação da interface gráfica do *xlet*, uma área para interação do usuário através do controle remoto virtual e a área de *prompt*.

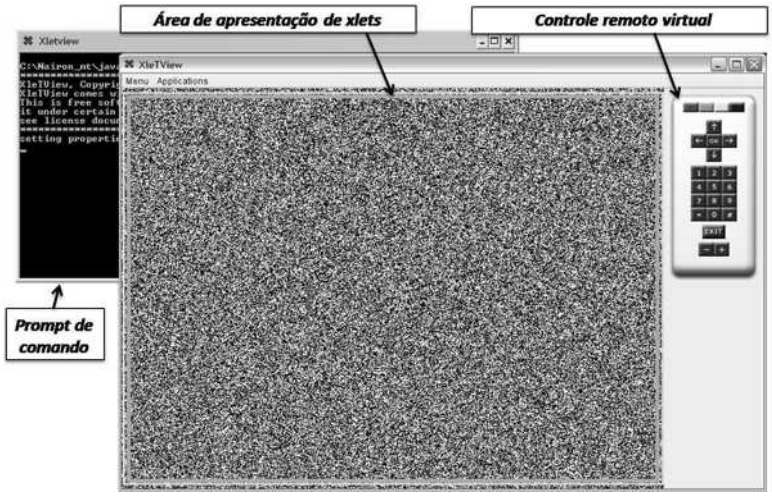


Figura 6.4: Interface do emulador XleTView.

FONTE: Do Autor.

Embora as limitações do emulador não prejudiquem as funcionalidades básicas da plataforma (visualização de *xlets* e integração com algumas API's Java), para o modelo de integração Ginga-J–OSGi o XleTView foi modificado. A Figura 6.5 ilustra os grupos funcionais da implementação modificada do XleTView, compostos por funções padrão de tratamento de arquivos (*net.beiker.xletview.io*), gerenciamento de mídias (*net.beiker.xletview.media*), carga de classes do sistema (*net.beiker.xletview.classloader*) e interface gráfica (*org.davic.** e *org.havi.**).

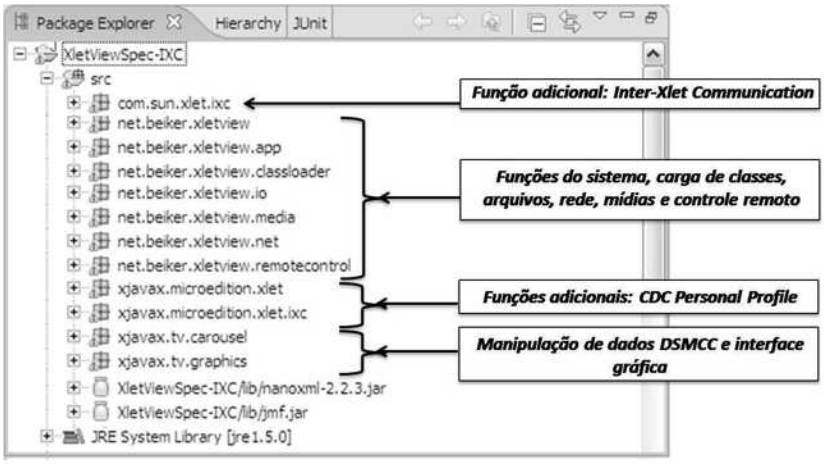


Figura 6.5: Estrutura do Projeto XleTView. Principais grupos funcionais e grupos adicionais. FONTE: Do Autor.

Para dar suporte ao modelo Ginga-J–OSGi, foi necessária a adição de componentes da implementação de referência *Java CDC Personal Profile* (CDC PP). O CDC PP especifica uma plataforma Java que pode executar em dispositivos de consumo como set-top boxes (Sun, 2009b). Do CDC PP dois grupos foram importados para o XleTView: *xjavax.microedition.** e *com.sun.xlet*. O primeiro apresenta alguns dos componentes javaTV modificados, além de componentes adicionais; o segundo carrega as classes necessárias para garantir a implementação do modelo de IXC, permitindo aos xlets de *xjavax.microedition.** registrar objetos remotos (objetos que estendem a inplementação de *java.rmi.Remote*) na entidade padão de registro do MHP-GEM, o *IXCRegistry*. Com esse grupo funcional adicionado à plataforma é possível a exportação de objetos do Ginga-J para o OSGi. O processo de integração entre essas plataformas que garante tal exportação é descrito a seguir.

Integração das Plataformas

Um dos pontos chave na construção da plataforma é o mecanismo de integração entre os ambientes Ginga-J e OSGi. Após uma longa análise nos modelos dos trabalhos relacionados, os mecanismos de Cabrer et al. (2006); Redondo et al. (2007); Yang et al. (2007); e Lin et al. (2008) foram estudados e parcialmente implementados. Percebeu-se que, embora utilizassem implementações de referência do MHP e OSGi, seus mecanismos de integração apresentavam algumas divergências. Um dos principais pontos estudados foi como os componentes de um ambiente são ativados pelo outro e como os objetos de contexto (*XletContext* para o MHP e *BundleContext* para o OSGi) são ativados e compartilhados. Os procedimentos variam desde a utilização de *ClassLoaders Java*, até a invocação remota de métodos (pela exploração do *Java Remote Method Invocation*).

Nos trabalhos de Cabrer et al. (2006) e Redondo et al. (2007), os ambientes XleTView e OSCAR (OSCAR, 2009) executam sob JVMs separadas e a comunicação entre eles se dá via *Java RMI*. A primeira inicialização é a do XleTView, que por sua vez ativa o ambiente OSCAR numa outra JVM. A Figura 6.6 esquematiza esse processo. A principal desvantagem desse método é a necessidade de gerenciamento da estrutura ponte *XBundLET* criada que possui ciclos de vida de *xlet* e *bundle* num único componente. Além do complexo mecanismo para gerenciamento dos contextos dessa estrutura, já que conforme os autores ela pode se comportar ora como *xlet* ora como *bundle*.

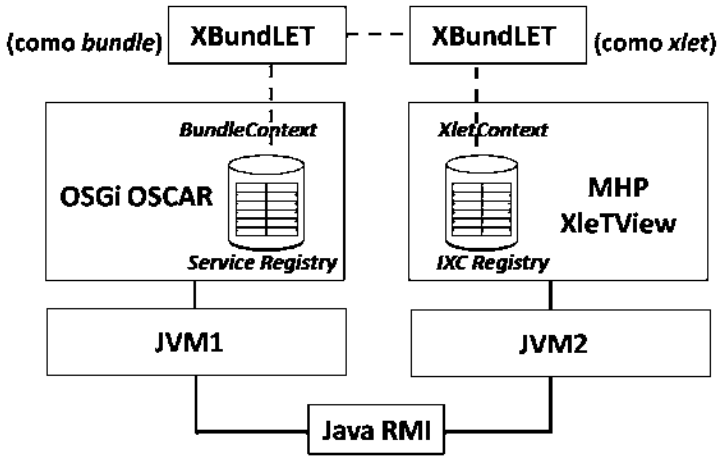


Figura 6.6: Ilustração do mecanismo de integração OSCAR-XleTView de Cabrer et al. (2006) e Redondo et al. (2007).

FONTE: Do Autor.

No trabalho de Yang et al. (2007) o ambiente Knopflerfish-XleTView é montado sob uma mesma Máquina Virtual. Entretanto a principal dificuldade continua sendo como permitir um ambiente acessar o outro. Para isso, as características dos *Java ClassLoaders* foram exploradas: os parâmetros de contexto (*XletContext* e *BundleContext*) são alocados fora do ambiente comum MHP-OSGi, num diretório de extensão do *Java Runtime Environment* (JRE), em [JAVAHOME/LIB/EXT]. O mecanismo de *Classloaders* garante a carga dessas classes e o acesso de *xlets* ao domínio OSGi, ao mesmo tempo que permite a *bundles* acessarem o ambiente MHP. A Figura 6.7 esquematiza esse processo.

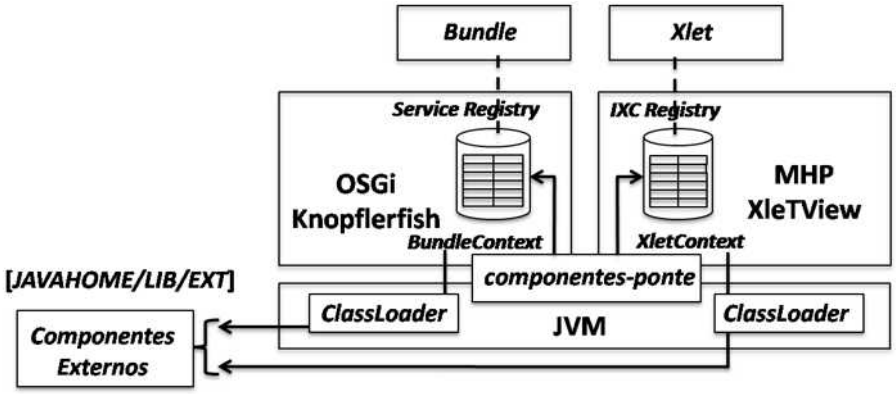


Figura 6.7: Ilustração do mecanismo de integração Knopflerfish-XleTView de Yang et al. (2007).

FONTE: Do Autor.

Já o mecanismo adotado por Lin et al. (2008) difere dos dois anteriores. As funcionalidades do ambiente MHP XleTView, como o *Application Manager* são agrupadas num *bundle* instalado no ambiente Knopflerfish. Dessa forma os contextos de *xlet* que são criados pelo *Application Manager* podem ser acessados diretamente pelo OSGi: ao invés de serem criados no ambiente XleTView, os contextos e os *xlets* são criados dentro do *bundle MHPAppManagerBundle* o que os permite também acessar outros contextos do ambiente OSGi. Embora essa implementação pareça resolver o problema, ela acaba por modificar todo o modelo MHP, ao extrair a funcionalidade principal de gerenciamento de aplicações e instalá-la como *bundle* OSGi. A Figura 6.8 ilustra esse processo.

A abordagem de intergação XleTview-Knopflerfish do ambiente Ginga-J–OSGi procurou extrair as principais características das abordagens anteriores. Os ambientes foram alocados sob uma mesma JVM, e foram exploradas as características dos *Java*

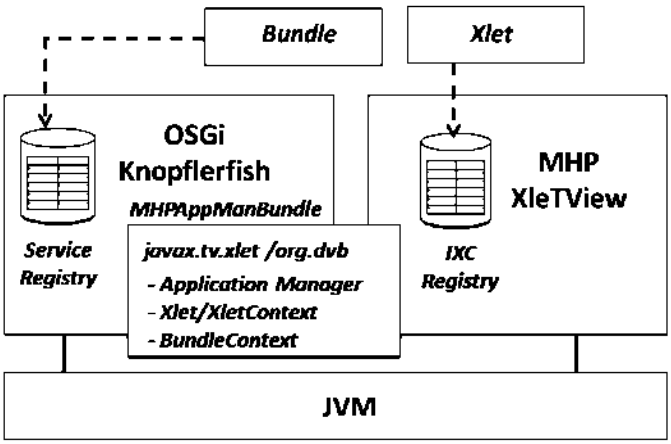


Figura 6.8: Ilustração do mecanismo de integração Knopflerfish-XleTView de Lin et al. (2008).

FONTE: Do Autor.

ClassLoaders, embora numa forma diferente da de Yang et al. (2007), o que não exigiu a criação de uma estrutura complexa como o *XBundLET* de Cabrer et al. (2006) e Redondo et al. (2007) tampouco a exportação de parte do modelo MHP-GEM para o ambiente OSGi como em Lin et al. (2008).

Para a integração XleTView-Knopflerfish foi criado um mecanismo adicional de carregamento de classes do ambiente OSGi pelo XleTView. Este possui um mecanismo particular de carga de classes, localizado no grupo *net.beiker.xletview.classloader*. Cada *xlet* do sistema possui seu próprio *ClassLoader*, o *XletClassLoader*. O XleTview possui ainda dois componentes nesse grupo: o *MainClassLoader* e o *EmulatorClassLoader*. O primeiro é o *Class Loader* mais externo na hierarquia de *Class loaders*, o último antes do *URLClassLoader* de *java.net.**. Já o segundo é responsável pela carga de todas as outras classes do sistema.

A estratégia de integração utiliza o componente *EmulatorClassLoader* para ativar o ambiente Knopflerfish. Para isso, as classes compiladas do *framework* foram adicionadas ao projeto do XleTView. Isso garante que apenas o código-fonte do XleTView seja modificado. Na inicialização do emulador (*net.beiker.xletview.Main*) o *URLClassLoader* ativa o *MainClassLoader* que por sua vez inicializa o *EmulatorClassLoader*. Este procura no sistema a classe inicial do Knopflerfish (*org.knopflerfish.framework.Main*) invocando dinamicamente seu método *main*. Uma vez feito isso, todas as classes do Knopflerfish serão

acessíveis pelo ambiente XleTView. Esse mecanismo permite a exportação dos parâmetros de contexto entre os dois domínios. A listagem da Figura 6.9 ilustra o processo descrito. Por fim, na Figura 6.10 pode ser observado o ambiente comum XleTView-Knopflerfish.

```
URLClassLoader systemLoader = (URLClassLoader)CopyOfMain.class.getClassLoader();
MainClassLoader loader = new MainClassLoader(systemLoader.getURLs());

Object emulatorClassLoader = null;
try {
    Class dynamicClass = loader.loadClass(
            "net.beiker.xletview.classloader.EmulatorClassLoader");

    Class[] constructorArgumentTypes = { URL[].class };
    Constructor classConstructor =
        dynamicClass.getConstructor(constructorArgumentTypes);
    Object[] constructorArgs = { loader.getURLs() };
    emulatorClassLoader = classConstructor.newInstance(constructorArgs);
} catch (Exception e) {
    e.printStackTrace();
}

// Inicializar o Knopflerfish com o mesmo classLoader do XleTView
// Para isso usa-se o componente EmulatorClassLoader
try {
    System.out.println("Inicializando Knopflerfish ...");
    Class dynamicClass_ = Class.forName("org.knopflerfish.framework.Main",
            false, (ClassLoader)emulatorClassLoader);

    Object[] constructorArgs = { args };

    // invocando metodo 'main'
    Method main = dynamicClass_.getMethod("main",
            new Class[]{String[].class});

    main.invoke(null, constructorArgs);

} catch (Exception e) {
    e.printStackTrace();
}
```

Configurar URLClassLoader e MainClassLoader

Criar um EmulatorClassLoader para o Knopflerfish

Configurar inicialização do Knopflerfish

Inicializar o Framework

Figura 6.9: Listagem do procedimento de carga do *framework* Knopflerfish a partir do ambiente XleTView.

FONTE: Do Autor.

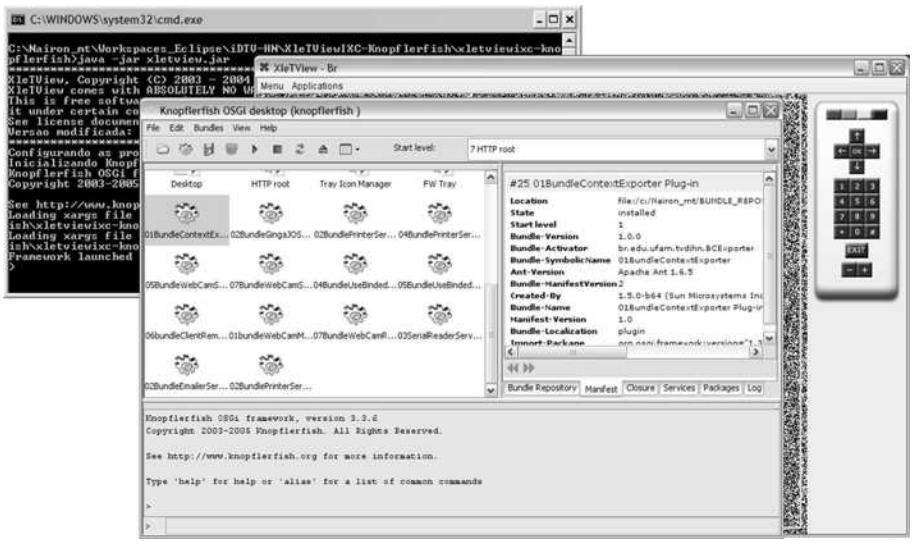


Figura 6.10: Ambiente integrado XleTView-Knopflerfish.

FONTE: Do Autor.

6.1.2 Configuração dos Componentes

Esta seção apresenta algumas funções importantes dos componentes do modelo Ginga-J–OSGi.

Componentes Comuns

Conforme mencionado, os componentes da plataforma encontram-se distribuídos no grupo *br.edu.ufam.tvdihn*. O grupo principal apresenta o componente ponte *ContextBridge* que serve como exportador de objetos de contexto para o ambiente. Ao acessar os contextos em *ContextBridge* um *xlet* pode buscar um serviço no *OSGi Service Registry* e um *bundle* pode buscar um objeto remoto no *MHP IXCRegistry*. O *ContexBridge* é um objeto que obedece ao padrão *Singleton*, ou seja, apenas uma instância desse conponente é permitida no sistema. O *XletContextExporter* é um *xlet* instalado na plataforma que, ao inicializar passa seu contexto ao *ContextBridge*; já o *BundleContextExporter* é um *bundle* instalado no ambiente OSGi que também repassa seu contexto à ponte. A inicialização do objeto *ContextBridge*, se dá, nos métodos *startXlet* e *start*, do *XletContextExporter* e *BundleContextExporter*.

No grupo funcional *br.edu.ufam.tvdihn.wrapped* estão os componentes centrais do modelo. O *WrappedProxyObject* e o *WrappedInterface*. O *WrappedProxyObject* armazena informações de um *objeto de serviço* que aqui é definido como o objeto a ser exportado de uma plataforma para a outra (o serviço OSGi ou o objeto remoto Ginga-J IXC). A principal função do *WrappedProxyObject* é proteger o acesso direto ao objeto que ele representa. Dessa forma, ele é uma implementação dos padrões de projeto *Wrapped* e *Proxy* descritos em Gamma et al. (1995).

Quando um objeto de serviço é associado ao *WrappedProxyObject* ele armazena cada um dos métodos do objeto. Para isso ele utiliza três componentes adicionais: *java.lang.Object object* (referente ao objeto de serviço), *java.reflect.Method method* (um objeto que representa o método a ser invocado do objeto de serviço) e *java.lang.object.Object methodObject* (que representa o objeto resultado da invocação do método *method*). Esses componentes auxiliam na proteção do objeto de serviço.

A título de exemplo se por acaso o objeto de serviço passado ao *WrappedProxyObject* fosse um serviço OSGi de impressão, representado pelo objeto *Printer* com o método *java.lang.String print(String fileName)*. No *WrappedProxyObject* o componente *object* será associado a *Printer*; *method* será associado a *String print(String fileName)* e *methodObject* será o objeto de retorno *String*.

A invocação do método do objeto de serviço se dará indiretamente no *WrappedProxyObject* por meio do método estático *invoke* que exige uma instância de *WrappedInterface*, o nome do método, a classe dos parâmetros e a lista dos parâmetros. A listagem da Figura 6.11 ilustra o processo no método *invoke*: no passo (1) é ativado o *initMethod*, método que inicializa o objeto *method*; no passo (2) o método inicializado é invocado e o objeto de retorno é armazenado em *methodObject*; por fim, no passo (3) o objeto *methodObject* é passado como retorno da chamada do método *invoke*.

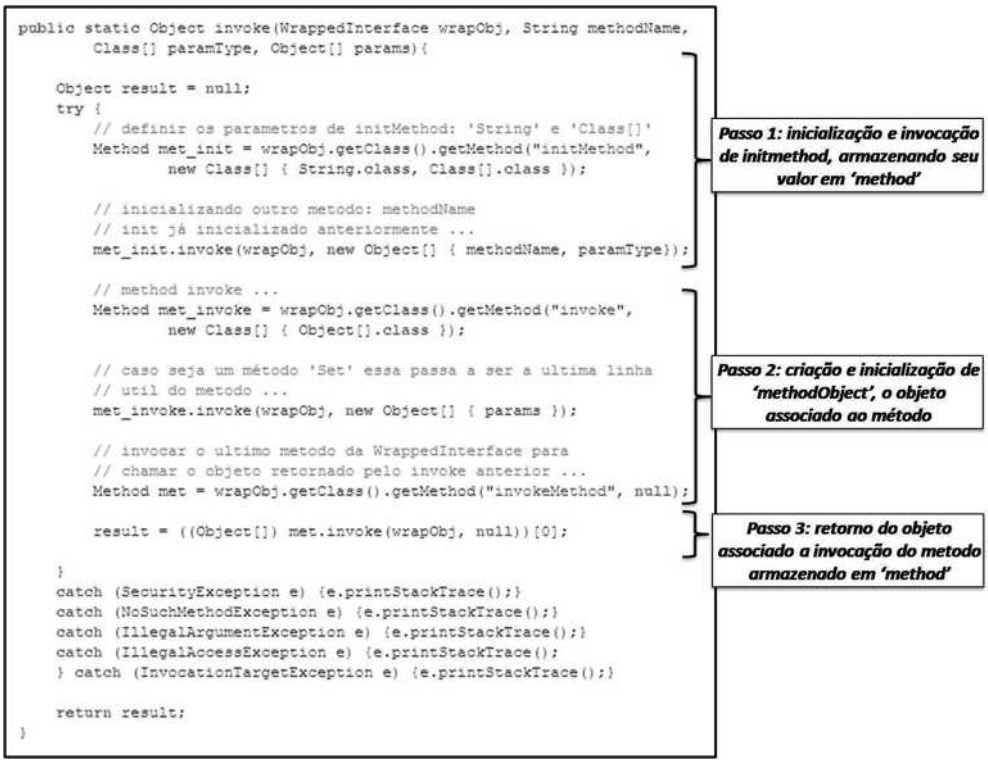


Figura 6.11: Listagem do processo de invocação automática do método no objeto de serviço em *WrappedProxyObject.invoke*.

FONTE: Do Autor.

O último grupo funcional de componentes comuns é *br.ufam.tvdihn.listeners* que apresenta classes para a criação do mecanismo de *Listeners* para eventos no IXC. Esse

mecanismo originalmente não está presente no XleTView. Ele é importante para o modelo pois garante que objetos remotos do Ginga-J registrados no *IXCRegistry* serão automaticamente passados ao domínio OSGi. As classes que compõem este grupo são: *IXCEvent* que representa um evento no IXC, que pode ser *BINDED* (quando um *xlet* registra um objeto remoto), *UNBINDED* (quando um *xlet* remove um objeto remoto previamente registrado) ou *REBINDED* (quando um *xlet* atualiza um objeto remoto previamente registrado); *IXCListener*, que de fato captura os eventos no *IXCRegistry*; e *IXCListeners* que armazena todos os objetos *IXCListener* para eventos de IXC que são criados.

Modelo Ginga-J–OSGi

Aqui é apresentado o componente diretamente associado ao processo de exportação de serviços OSGi para o domínio Ginga-J: *OSGiGingaJ-Register*, representado pela classe *OSGiGingaJRegister* no grupo *br.edu.ufam.tvdihn*.

O *OSGiGingaJRegister* implementa um mecanismo de registrar serviços OSGi como funções do Ginga-J. Para isso é implementado um *Listener* para registro de serviços OSGi (através do componente *org.osgi.framework.ServiceListener*). Uma vez detectado, o serviço é exportado ao Giga-J utilizando principalmente os componentes *ContextBridge* e *WrappedProxyObject*. De uma forma geral, a classe 'converte' um serviço OSGi num objeto remoto do IXC usando os mecanismos do *WrappedProxyObject* descritos anteriormente.

O *OSGiGingaJRegister* possui três funções: capturar objetos de serviço OSGi, registrar esses objetos no IXC e remover os mesmos do IXC. A primeira função é realizada através de um componente padrão do OSGi, o *ServiceEvent*, associado a um evento no *Service Registry*. Os eventos podem ser dos tipos *ServiceEvent.REGISTERED* (indica que um serviço OSGi foi registrado), *ServiceEvent.UNREGISTERING* (um serviço OSGi está sendo removido) e *ServiceEvent.MODIFIED* (indica que as propriedades de um serviço foram atualizadas).

Ao recuperar um objeto de serviço o *OSGiGingaJRegister* verifica as propriedades do serviço, fazendo uma consulta no padrão da linguagem LDAP, buscando duas propriedades principais: *SERVICE_ALIAS = "HOME_NETWORK_ALIAS"* e

SERVICE_TYPE = "HOME_NETWORK_SERVICE". Caso essas propriedades não sejam satisfeitas, o serviço OSGi não será considerado um serviço a ser exportado no *IXCRegistry*. Caso satisfeitas, o *OSGiGingaJRegister* executará sua segunda função: registrar o componente no IXC. Para isso o objeto de serviço é inicializado (utilizando um contexto de *bundle* armazenado no *ContextBridge*), passado ao *WrappedProxyObject* e o *IXCRegistry* é invocado (usando um contexto de *xlet* armazenado no *ContextBridge*) para registrar o objeto remoto (através do método *IXCRegistry.bind*). O *Alias* (identificação do objeto) para o objeto remoto do IXC também é recuperado do objeto de serviço OSGi, utilizando a propriedade *SERVICE_IXC_ALIAS*. A listagem da Figura 6.12 ilustra esse processo.

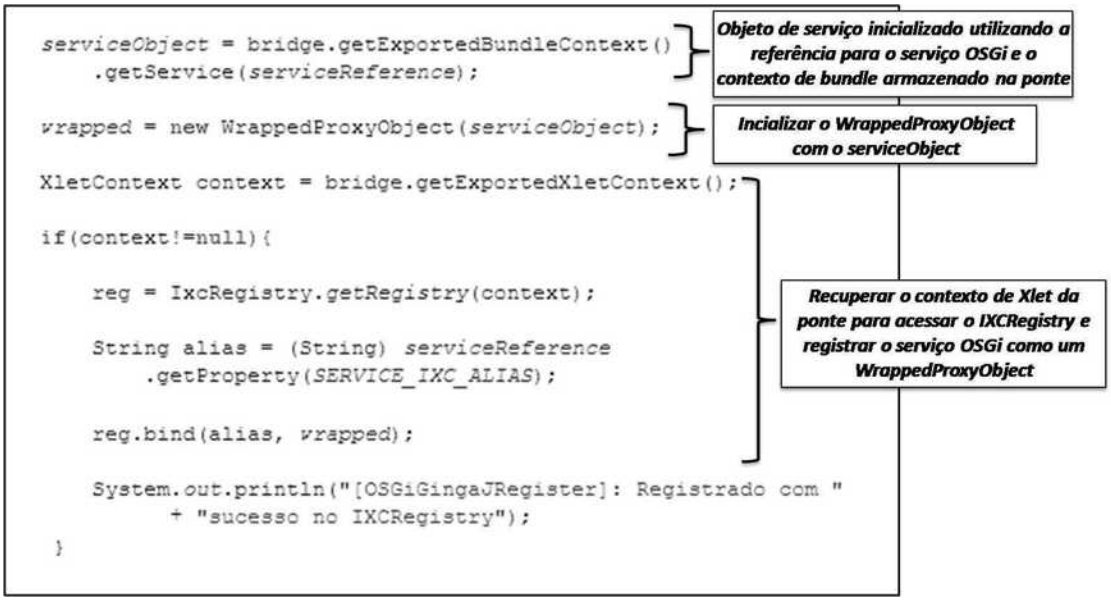


Figura 6.12: *OSGiGingaJRegister*: registro do objeto de serviço do OSGi como objeto remoto do IXC.

FONTE: Do Autor.

A terceira função do *OSGiGingaJRegister* é o processo inverso: remover a funcionalidade do IXC. Quando um *bundle* remove seu serviço no OSGi um evento *ServiceEvent.UNREGISTERING* é disparado e capturado pelo *OSGiGingaJRegister*, que verifica o *Alias* para o objeto correspondente no *IXCRegistry* e o remove através de uma operação *IXCRegistry.unbind*.

Modelo OSGi–Ginga-J

Aqui são apresentados os componentes que dão suporte ao processo de registrar objetos remotos do *IXCRegistry* (no formato *WrappedProxyObject*) no *OSGi Service Registry*. Os componentes são *GingaJ-OSGiRegister* e *GingaJ-OSGiBundle*, ambos configurados em um único *bundle*, o *BundleGingaJOSGiRegister*, no grupo *br.edu.ufam.tvdihn*.

O componente *GingaJOSGiRegisterActivator* é responsável pela ativação do *BundleGingaJOSGiRegister*. Este implementa um *Listener* para eventos sobre o *IXCRegistry* (por meio do *IXCListener* presente *em br.ufam.tvdihn.listeners*). As principais funções do *GingaJOSGiRegisterActivator* são capturar o objeto registrado no IXC, registrar e remover o objeto no OSGi *Service Registry*. Diferentemente do *OSGiGingaJRegister* que sempre registra um novo objeto remoto para cada serviço do OSGi, o *GingaJOSGiRegisterActivator* registra apenas um único serviço no OSGi para todos os objetos provenientes do Ginga-J, o *GingaJOSGiService*. Esse serviço será acessado por *bundles* que necessitem utilizar funcionalidades do domínio Ginga-J. O *GingaJOSGiService* é associado ao objeto *WrappedProxyObject* do ambiente Ginga-J.

O fato de registrar apenas um serviço por meio do *GingaJOSGiService* exige um mecanismo para armazenar as referências aos objetos *WrappedProxyObject* vindas do Ginga-J. Caso contrário, sempre que um novo serviço for registrado no *IXC Registry*, seu correspondente registro no OSGi por meio de um único serviço do *GingaJOSGiService* irá sobrepor o registro anterior. Para contornar essa dificuldade foi adicionado um repositório de objetos ao *BundleGingaJOSGiRegister*, de forma que sempre que um objeto novo do IXC chegar ao *BundleGingaJOSGiRegister* ele será adicionado ao repositório.

O diagrama da Figura 6.13 ilustra o processo descrito até aqui. Nele podem ser observados dois objetos do IXC que são capturados pelo *Listener* do *GingaJOSGiRegisterActivator* e passados ao *GingaJOSGiService* que os armazena num repositório, utilizando os componentes *ServiceRefRegPair* de *br.edu.ufam.tvdihn.service.repository*. Esses componentes armazenam objetos *ServiceReference* do OSGi que por sua vez armazenam informações de propriedades de serviços. Com essas propriedades será possível

recuperar o objeto de remoto do IXC desejado do repositório, ao acessar o serviço de *OSGiGingaJService*.

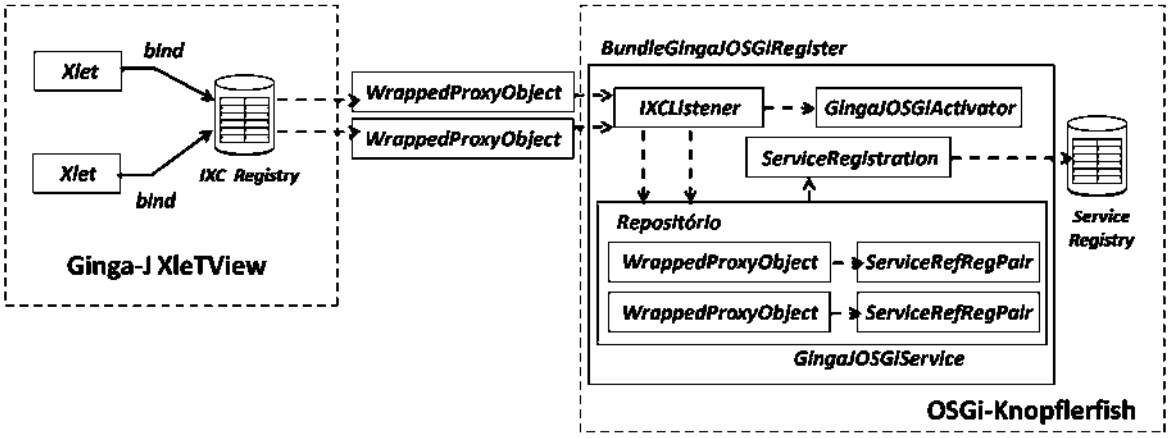


Figura 6.13: Componentes do *BundleGingaJOSGiRegister* em atuação na plataforma.
FONTE: Do Autor.

O código para o mecanismo descrito na Figura 6.13 é ilustrado na Figura 6.14. O processo começa quando o *GingaJOSGiRegisterActivator* detecta um evento do tipo *IXCListener.BINDED*. Após isso a instância do objeto remoto do IXC é recuperada (usando o componente *IXCEvent*) e passada para a inicialização de um *WrappedProxyObject*. Em seguida uma instância de *GigaJOSGiService* é criada para armazenar o *WrappedProxyObject*. Após isso o *Alias* do objeto do IXC recuperado do *IXCEvent* é configurado como uma propriedade do *GingajOSGiService*. Dessa forma um *bundle* saberá como acessar o objeto utilizando esse *Alias*. Em seguida o *GingajOSGiService* é registrado com o objeto *WrappedProxyObject*. Por fim, o *GingajOSGiService* é novamente recuperado e utilizado para inicializar dois objetos: um *ServiceReference* e um *ServiceRegistration* (ambos em *org.osgi.framework*). O par <*ServiceReference, ServiceRegistration*> garante a identificação única do objeto do IXC e é adicionado ao repositório de objetos de *ServiceRefRegPair*.

O outro evento que pode ocorrer sobre o *GingaJOSGiRegisterActivator* é do tipo *IXCListener.UNBINDED* que indica que um objeto foi removido do IXC e seu correspondente no OSGi também precisa ser removido. O processo inverso ao descrito anteriormente é executado, encontrando o par <*ServiceReference, ServiceRegistration*> com o *Alias* do objeto do IXC e removendo-o do repositório.

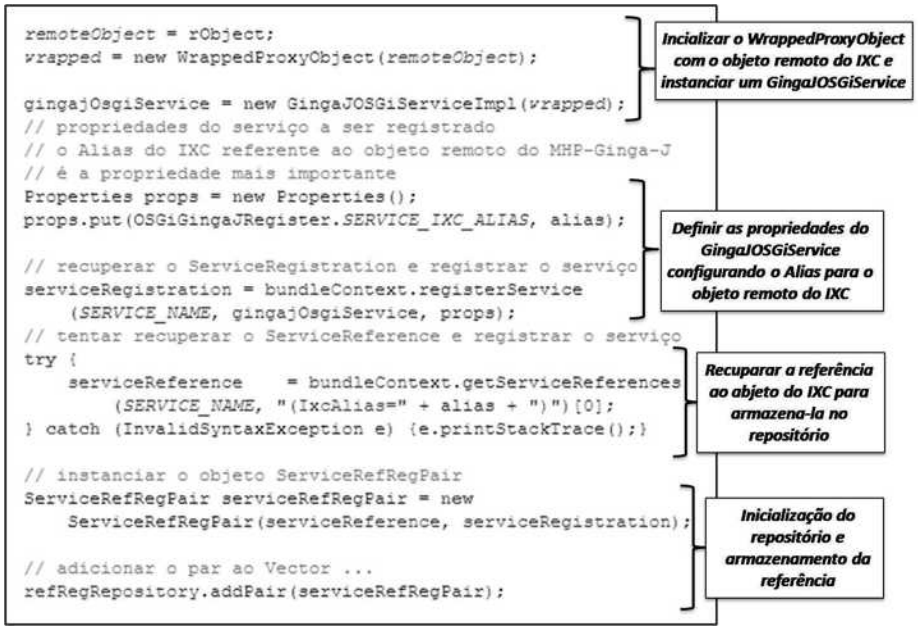


Figura 6.14: Listagem para o processo de registro do objeto do IXC como serviço do OSGi. FONTE: Do Autor.

6.1.3 Componentes em Ação

Nesta seção é apresentado um processo para a construção de aplicações no ambiente integrado XleTView-Knopflerfish, para cada mecanismo de interação. As etapas do processo, desde o instante em que um objeto de serviço é criado em um ambiente até o momento em que ele pode ser utilizado no outro ambiente, são apresentadas a seguir.

Esquema 1: Ginga-J–OSGi

Para esse cenário, considera-se o seguinte problema: exportar um objeto de serviço referente a um serviço de impressão simplificado (Printer), do OSGi para o Giga-J permitindo assim que *xlets* modifiquem as propriedades do objeto exportado. Para tal cenário o processo consiste nas seguintes etapas:

Passo 1 - Configuração do bundle e do serviço: nessa etapa é criado o *bundle* e os objetos de serviço são definidos e configurados para serem registrados no momento da inicialização do bundle no *framework*. Uma estrutura típica de projeto de bundle ilustrada na Figura 6.15, contém os seguintes grupos: *grupo do activator*, onde estará a classe que implementa *org.osgi.framework.Activator*, que contém os estados para o

gerenciamento do *bundle* (na figura, o pacote *br.ufam.osgi.printer*); *grupo do serviço*, que contém a interface que será registrada no *OSGi Service Registry*; e *grupo de implementação* que contém a implementação real do serviço. Além destes, estão também presentes bibliotecas padrão do JRE (*JRE System Library*) e bibliotecas do ambiente Ginga-J–OSGi (*knopflerfish-xletviewixc_lib*, que contêm as bibliotecas *xletview.jar* e *framework.jar*).

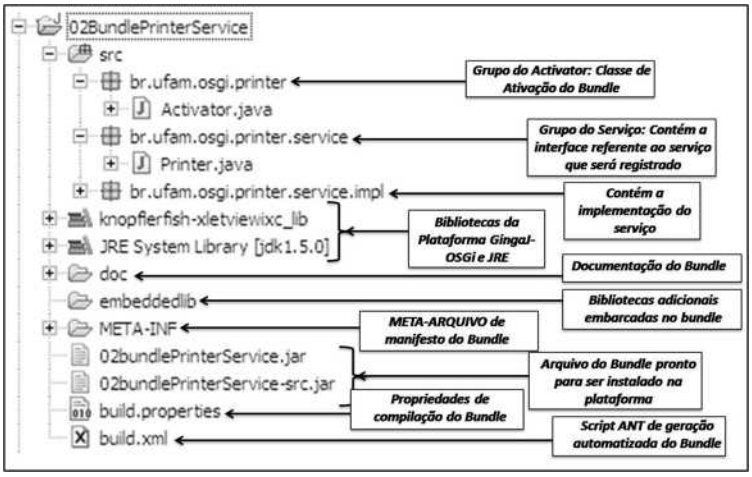


Figura 6.15: Típica estrutura de um projeto de um *bundle* para a plataforma Ginga-J-OSGi. FONTE: Do Autor.

Na estrutura do *bundle* existem ainda alguns diretórios e arquivos padrões, como: */doc* que armazena a documentação HTML das classes do *bundle*; */embeddeedlib* que armazena API's adicionais integradas ao *bundle*; */META-INF* que contém o arquivo de manifesto do *bundle* (*MANIFEST.MF*), um meta-arquivo que será interpretado no *framework* indicando quais serviços possui o *bundle*, de quais serviços o *bundle* depende, quais grupos de pacotes serão exportados, qual a classe *Activator* do *bundle*, etc.; *build.properties*, arquivo que armazena informações de compilação das classes do *bundle*; e por fim, o arquivo *build.xml*, um *script Ant build* que contém comandos para a compilação automática e geração do *bundle* empacotado com todos os componentes. *Ant build* é uma tecnologia da *Apache Software Foundation* (Apache, 2009) que oferece um mecanismo automatizado baseado em XML para a execução de comandos sobre a JVM.

Passo 2 - Instalação do bundle na plataforma: o serviço exportado (*br.ufam.osgi.printer.service.Printer*) é então passado ao *Activator* do *bundle*. Para que o registro do objeto *Printer* no OSGi possa ser reconhecido como um registro a

ser exportado ao Ginga-J algumas propriedades do serviço devem ser definidas, tais como *OSGiGingaJRegister.SERVICE_ALIAS* e *OSGiGingaJRegister.SERVICE_TYPE* explicadas anteriormente. Também deve ser definido um *Alias* para o objeto remoto do IXC correspondente ao serviço OSGi. Isso é feito definindo mais um valor para uma propriedade, *OSGiGingaJRegister.SERVICE_ALIAS*. Todo esse processo de configuração acontece no método *start* do *bundle* do serviço de impressão. Ao ser instalado no *framework* esse método é executado e os componentes da plataforma entram em ação.

Passo 3 - Uso dos componentes-ponte para exportar o serviço ao Ginga-J: no momento em que o serviço *Printer* é registrado ele é detectado pelo *Listener* do *OSGiGingaJRegister*, que realiza o processo descrito na seção 6.1.2. A Figura 6.16 ilustra esse processo, no ambiente integrado XleTView-Knopflerfish. Primeiramente, o *OSGiGingaJRegister* detecta o objeto de serviço *Printer* e o registra no *IXCRegistry* do Ginga-J. Logo após, o *GingaJOSGiRegisterBundle* detecta que um objeto foi registrado no IXC, mas que, por não ser um objeto remoto do Ginga-J, não precisa ser exportado ao domínio OSGi. Nesse ponto o ciclo está completo: o serviço *Printer* passa a estar disponível tanto para *xlets* do Ginga-J quanto para *bundles* do OSGi.

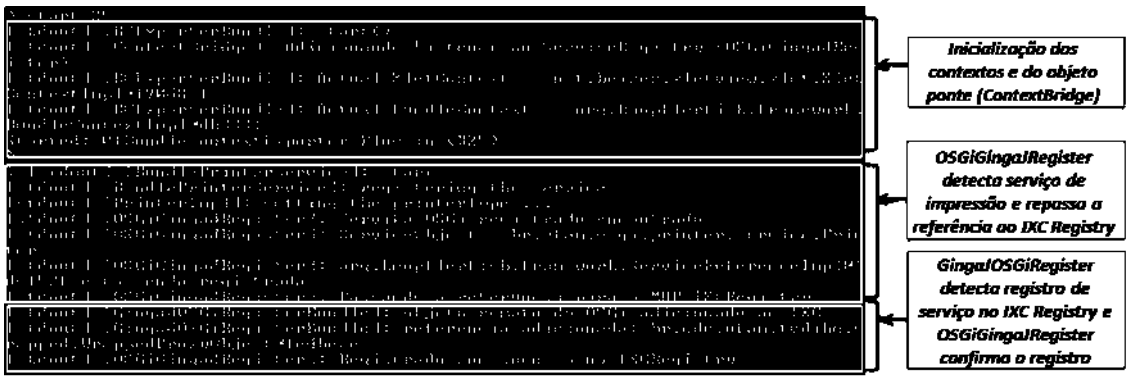


Figura 6.16: Exportando o serviço *Printer* do OSGi para o Ginga-J. Execução no *prompt*.
FONTE: Do Autor.

Passo 4 - Configuração do xlet para utilizar o serviço do bundle: para um *xlet* do Ginga-J, não há diferença entre um objeto de serviço do OSGi e um objeto remoto do próprio Ginga-J, já que para obtê-los ele simplesmente acessa o *IXCRegistry*. Para acessar os dois métodos disponibilizados pelo serviço *Printer*, o *xlet* segue os procedimentos ilustrados na Figura 6.17: primeiro, ele recupera o objeto remoto no

formato de *WrappedProxyObject* do *IXCRegistry* utilizando o *Alias* definido no registro do serviço; logo após ele utiliza o método *invoke* do *WrappedProxyObject* para invocar indiretamente o método *getPrinterType* passado no parâmetro. O retorno do método fornece o tipo da impressora configurada no momento do registro do serviço no OSGi; o último passo é modificar o tipo da impressora, acessando *setPrinterType* também através de *invoke*, dessa vez passando os parâmetros do novo tipo configurado. Dessa forma o *xlet* consegue modificar as propriedades do objeto de serviço do OSGi.

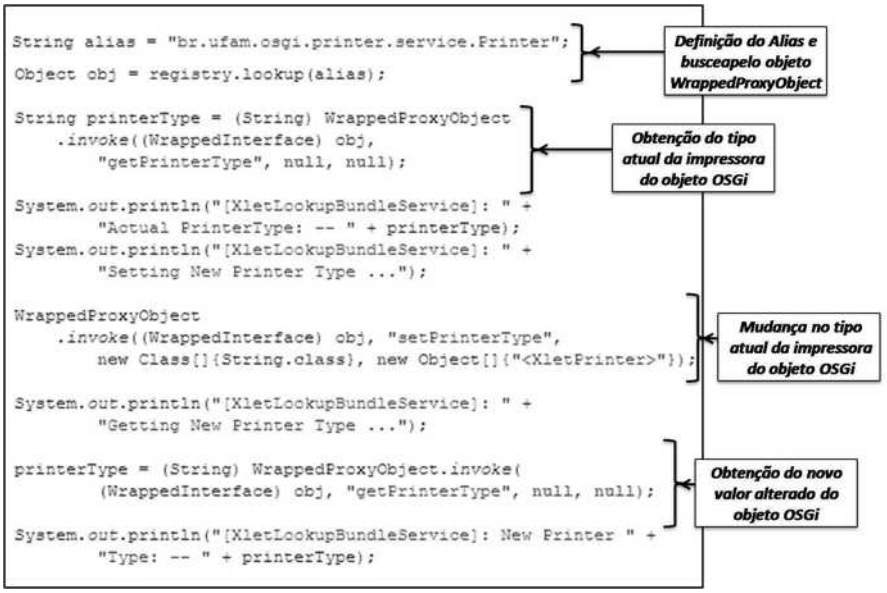


Figura 6.17: Listagem: Acesso às informações do serviço de impressão pelo *xlet*.
FONTE: Do Autor.

Esquema 2: OSGi–Ginga-J

Para esse cenário, considera-se o seguinte problema: exportar um objeto de remoto do *IXC Registry* do Ginga-J para o ambiente OSGi. Como ilustração será considerada a seguinte situação: um *xlet* do Ginga-J precisa exportar uma área correspondente à tela da TV Digital para o OSGi, de forma que seja possível receber imagens do domínio OSGi. Isso pode acontecer por exemplo, quando um *bundle* que gerencia uma câmera do sistema de segurança precisa enviar uma imagem para a tela da TVDI. Além da área de imagem o *xlet* deve também exportar um objeto para receber informações de texto do OSGi. Para tal cenário o processo consiste nas seguintes etapas:

Passo 1 - Definir os objetos remotos do IXC: da mesma forma que um serviço OSGi, um objeto remoto do IXC é uma interface Java que obedece a certos padrões. Para o IXC, objetos remotos devem sempre estender de *java.rmi.Remote* e todos os métodos devem ativar uma exceção do tipo *java.rmi.RemoteException*. O projeto do *xlet* que exportará um objeto ao IXC tem como grupo principal *br.edu.ufam.idtvhn.remoteixc*, que contém o objeto remoto (*RemoteObj*) e sua implementação (*RemoteObjImpl*). No objeto estão presentes métodos para receber a imagem e o texto provenientes do ambiente OSGi, ou do próprio Ginga-J, caso a invocação do objeto venha de outro *xlet*.

Passo 2 - Registrar os objetos no IXC Registry: no método *startXet* é realizado o registro do objeto remoto. Um *Alias* para o objeto é definido e utilizado na operação *bind* do *IXCRegistry*. Nesse ponto os componentes do modelo entrarão em ação.

Passo 3 - Uso dos componentes-ponte para exportar o objeto remoto ao ambiente OSGi: no momento em que o objeto *RemoteObj* é registrado ele é detectado pelo *Listener* do *GingaJOSGiRegisterActivator* no *BundleGingaJOSGiRegister*. A Figura 6.18 ilustra esse processo, no ambiente integrado XleTView-Knopflerfish, em execução no *prompt* de comando. Primeiramente, o *GingaJOSGiRegisterActivator* detecta o *RemoteObj* e o registra no *Service Registry* do OSGi como um serviço *GingaJOSGiService* (conforme descrito na seção 6.1.2). Ao registrá-lo no *Service Registry* o *OSGiGingaJRegister* detecta o registro, mas não realiza operação alguma, pois não se trata de um serviço do OSGi e sim de um *OSGiGingaJService* referente a um objeto do IXC. Por fim, ao registrar o *RemoteObj*, o repositório de objetos do *GingsJOSGiService* é incrementado em mais uma referência (conforme na seção 6.1.2).

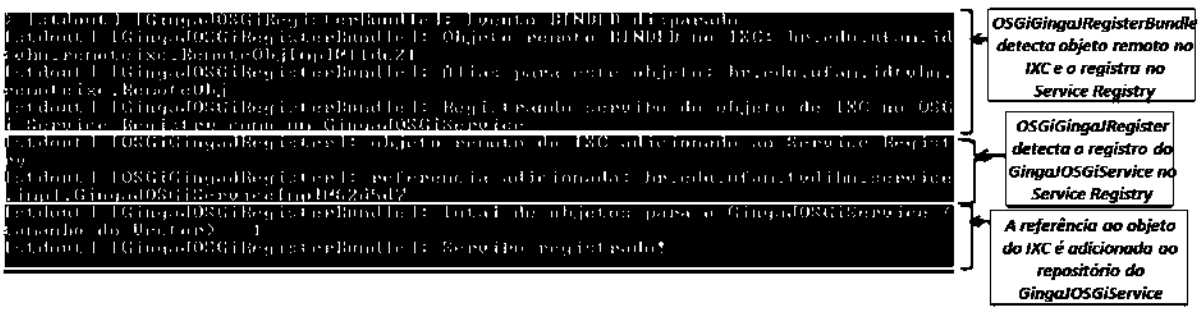


Figura 6.18: Exportando o *RemoteObj* do Ginga-J para o OSGi. Execução no *prompt*.
FONTE: Do Autor.

Passo 4 - Configuração do bundle para usar os objetos remotos do Ginga-J: para que um *bundle* possa acessar os métodos do *RemoteObj*, exportando texto e imagem à tela do *xlet*, é necessário primeiramente encontrar o serviço *GingaJOSGiService*. Após isso, o *bundle* precisa localizar, dentre os objetos remotos armazenados no repositório de objetos, o objeto remoto apropriado. Para isso, o *bundle* busca nas propriedades do *GingaJOSGiService* somente o objeto que satisfaz a condição de registro inicial indexada pela propriedade *IXCAlias*, o mesmo *Alias* do objeto remoto registrado previamente no *IXCRegistry*. Como só é permitido registrar apenas um *Alias* por objeto remoto no IXC, o objeto encontrado será exatamente o desejado. Esse objeto virá na forma de *WrappedProxyObject* e seus métodos serão invocados indiretamente, como no processo anterior.

A Figura 6.19 ilustra a interface final do ambiente XleTView-Knopflerfish para esse cenário, dando destaque às interfaces do *xlet* e do *bundle* que trocam objetos de texto e imagem através do objeto remoto do IXC. Para enviar uma imagem ou texto ao XleTView, um *bundle* do Knopflerfish chama o método *invoke* do *WrappedProxyObject* recuperado no repositório.

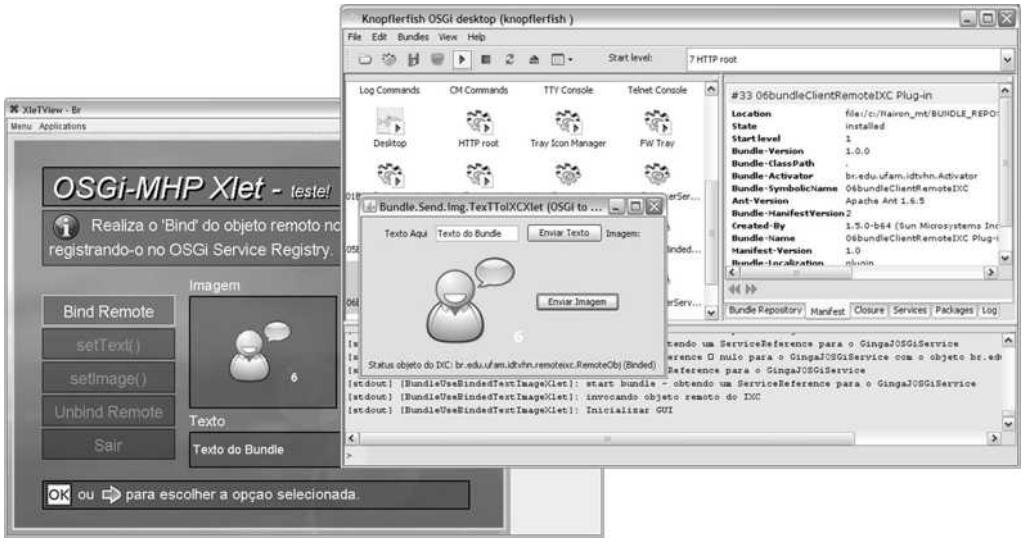


Figura 6.19: Interface final para o cenário de intergração Ginga-J–OSGi baseado no objeto remoto do IXC.

FONTE: Do Autor.

6.2 Implementação Declarativa

O modelo que define a implementação declarativa STB-HG utiliza tecnologias agregadas ao middleware declarativo brasileiro, o Ginga-NCL, e construídas sobre a implementação de referência Ginga-NCL Emulator (Ginga-NCL, 2009a), um ambiente para a interpretação de *scripts* baseados em NCL, apresentação de mídias (áudio, vídeo, texto, html, etc) e execução de conteúdo procedural basedo na linguagem Lua. Até o período em que se finalizou este trabalho, a versão disponível do emulador era a 1.1.1. Assim como no modelo procedural, a implementação do OSGi utilizada foi o Knopflerfish 3.3.6.

A Figura 6.20 ilustra o modelo que agrupa os principais componentes das tecnologias utilizadas no ambiente Ginga-NCL–OSGi. No que se refere ao OSGi, os mesmos mecanismos básicos do ambiente procedural foram utilizados; já no ambiente da TVDI, a arquitetura do sistema foi projetada de modo a permitir ao programador de scripts NCL buscar e interagir com serviços OSGi na rede. Dessa forma, os componentes-ponte devem ser criados de modo a permitir a troca entre objetos de serviços OSGi e objetos de serviços Ginga-NCL (os *Adapters*, discutidos na seção 6.2.1).

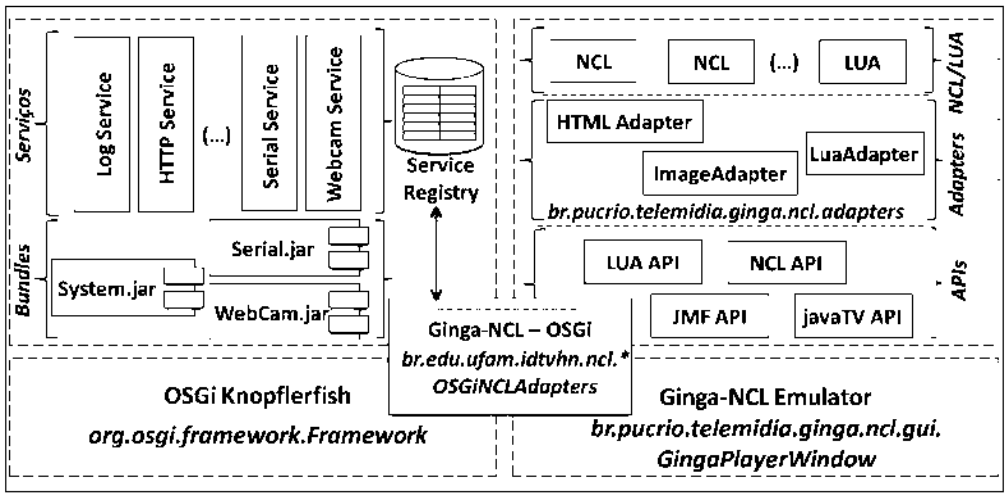


Figura 6.20: Plataforma de suporte ao ambiente Ginga-NCL–OSGi e tecnologias.
FONTE: Do Autor.

Dadas às carcaterísticas do middleware declarativo, que se limita a uma máquina de apresentação de documentos, os recursos de programação se resumem apenas à funções

para o tratamento deste tipo de conteúdo. No ambiente procedural, ao contrário, o programador tem mais liberdade para definir suas próprias funções e mecanismos. Como exemplos comparativos pode-se citar as características do ambiente Ginga-NCL Emulator quando comparadas ao emulador XleTView; no primeiro, todas os procedimentos estão limitados à sintaxe NCL com alguma dinâmica em LUA; já no segundo, há a possibilidade de integração com outras API's como havi, davic ou o CDC Personal Profile, o que permitiu a definição do mecanismo de IXC, por exemplo. Essas carcaterísticas podem ser utilizadas para justificar o fato de o mecanismo declarativo ser, do ponto de vista arquitetural, menos complexo que o procedural. A seguir serão apresentados quais os mecanismos que permitiram a execução do ambiente Ginga-NCL–OSGi.

6.2.1 Configuração do Ambiente

Aqui são apresentadas as principais características do ambiente declarativo, implementado no Ginga-NCL Emulator, que foram exploradas de modo a possibilitar a integração com o OSGi.

Ambiente Ginga-NCL Emulator

Atualmente existem IRs do Ginga-NCL construídas para diversas plataformas, usando ferramentas e linguagens de programação diferentes. Em (Ginga-NCL, 2009a) podem ser verificadas versões do Ginga-NCL para ambientes Windows, Linux e MacOS, e implementações baseadas em Java e C++. O fato de ter sido escolhido uma implementação em Java do middleware é justificado pela facilidade de integração com o ambiente OSGi, já que este, representado pelo Knopflerfish, executa sobre uma JVM.

A estrutura do projeto do Ginga-NCL emulator apresenta grupos funcionais que dividem parte da interpretação do conteúdo declarativo e integração com conteúdo procedural LUA, além de grupos que tratam com as mídias do sistema. Dessa forma, dentre os principais grupos funcionais do ambiente podem ser citados: componentes do núcleo do sistema (em *br.org.ginga.core* e *br.pucrio.telemidia.ginga.core.io*, que executam as mídias suportadas pelo sistema); componentes de suporte ao tratamento

de scripts NCL segundo o modelo NCM (em *br.org.ginga.ncl*); e componentes que permitem a integração de mídias e conteúdo procedural com o middleware (em *br.pucrio.telemidia.ginga.ncl.adapters*).

No último grupo funcional citado está a base para a apresentação de conteúdos multimídia no Ginga-NCL, onde estão localizados os *Adapters*. Um *Adapter* é um componente responsável pela garantia de execução das mídias no ambiente. Ele é associado a um objeto *Player*, que representa diretamente a mídia. Dessa forma, para cada tipo de conteúdo que executa no *middleware*, há um par <*Adapter*, *Player*> correspondente. Como exemplos pode-se citar o *HTMLPlayerAdapter* e o *HTMLPlayer*, *Adapter* e *Player*, respectivamente, associados a apresentação de conteúdo HTML no middleware. Outro exemplo importante de par (*Adapter*, *Player*) são *LuaPlayer* e *LuaPlayerAdapter*, que interpretam e executam conteúdo procedural escrito em LUA.

Para associar objetos *Adapters* e *Players* às mídias que executarão no sistema, o componente *PlayerAdapterManager* (*br.pucrio.telemidia.ginga.ncl.adapters*) mantém em dois arquivos de propriedades valores que associam cada adapter a um tipo de objeto de mídia que será posteriormente invocado no arquivo NCL. No diretório *./gingaNclConfig.players* o arquivo *ctrldefs.ini* relaciona um nome para cada classe relativa a um *Adapter* do sistema. Já no arquivo *mimidefs.ini* são definidos os identificadores que poderão ser usados no documento NCL, para referenciar um *Adapter*. A Figura 6.21 ilustra um esquema que associa as mídias (arquivos LUA e HTML) aos *Adapters* e como são configuradas no documento NCL.

Integração das Plataformas

Da mesma forma que no modelo procedural, o ambiente Ginga-NCL–OSGi foi integrado numa mesma JVM, utilizando os componentes padrão de inicialização de um ambiente para ativar os elementos de inicalização do outro ambiente. No Ginga-NCL Emulator, o grupo *br.pucrio.telemidia.ginga.ncl.gui* contém o componente de ativação *GingaPlayerWindow* que inicializa as classes do sistema por invocação direta, sem utilizar o mecanismo de *ClassLoaders* como no XleTView, no modelo explicado anteriormente.

No momento da ativação do Ginga-NCL Emulator é inicializado o componente

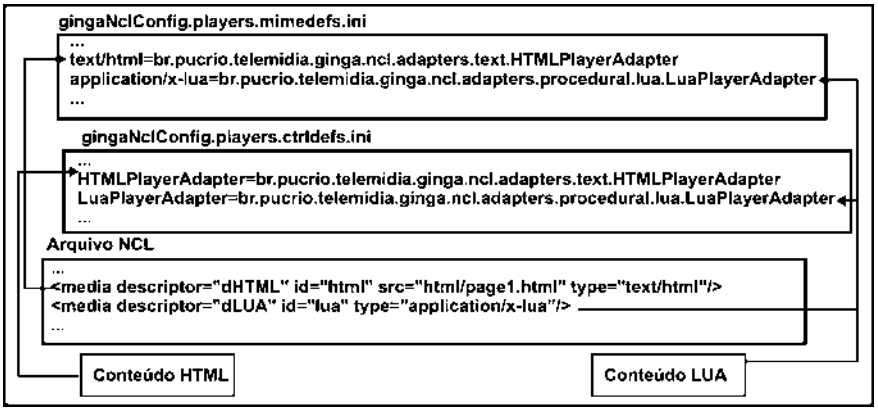


Figura 6.21: Esquema de configuração dos *Adapters* para cada tipo de mídia do sistema.
FONTE: Do Autor.

GingaPlayerWindow. A cargo dele fica a tarefa de carregar o ambiente OSGi. Para tanto, a implementação de referência do Knopflerfish foi adicionada ao projeto do Ginga-NCL Emulator, para serem compiladas conjuntamente, permitindo dessa forma que componentes do Ginga-NCL acessem funções padrão do OSGi e vice-versa. Com essa estratégia, o *GingaPlayerWindow* pode acessar diretamente o componente *org.knopflerfish.framework.Main*, ativando o Knopflerfish.

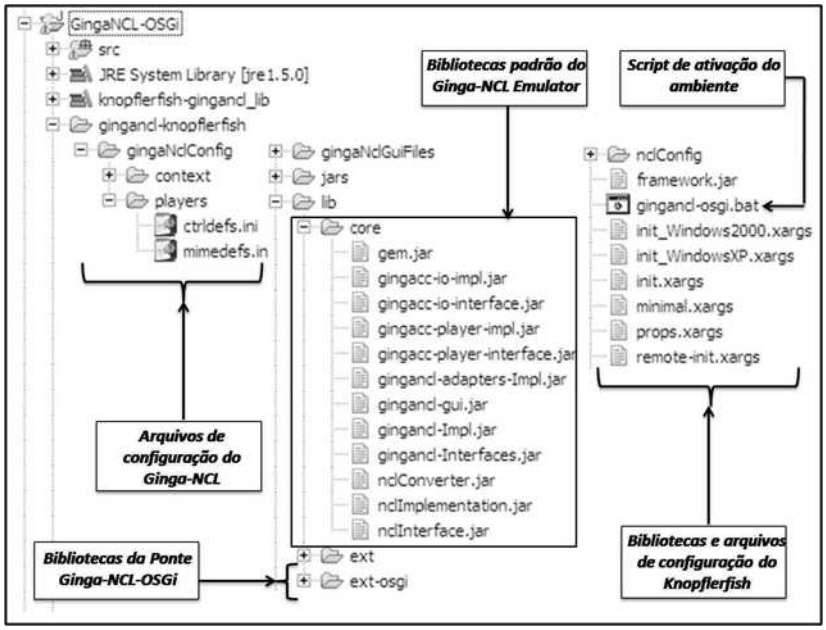


Figura 6.22: Esquema do Projeto do ambiente integrado Ginga-NCL–Knopflerfish.
FONTE: Do Autor.

A Figura 6.22 ilustra o projeto do ambiente Ginga-NCL-Knopflerfish. A plataforma contém referências para as bibliotecas *framework.jar* do Knopflerfish, além de

referências para as bibliotecas geradas da compilação dos grupos do Ginga-NCL (*gem.jar*, *gingacc-io-impl.jar*, *gingacc-player-interface.jar*, *gingancl-gui.jar*, etc). O ambiente é gerado automaticamente através de um *script Apache Ant*, que trata de compilar em conjunto as classes da plataforma e alocar os arquivos de sistema usados pelos dois ambientes em seus respectivos diretórios (por exemplo, o *script ant* é responsável por alocar os componentes-ponte do modelo no diretório */lib/ext-osgi*). Uma vez executado, o ambiente Ginga-NCL–OSGi apresenta interface como a ilustrada na Figura 6.23.

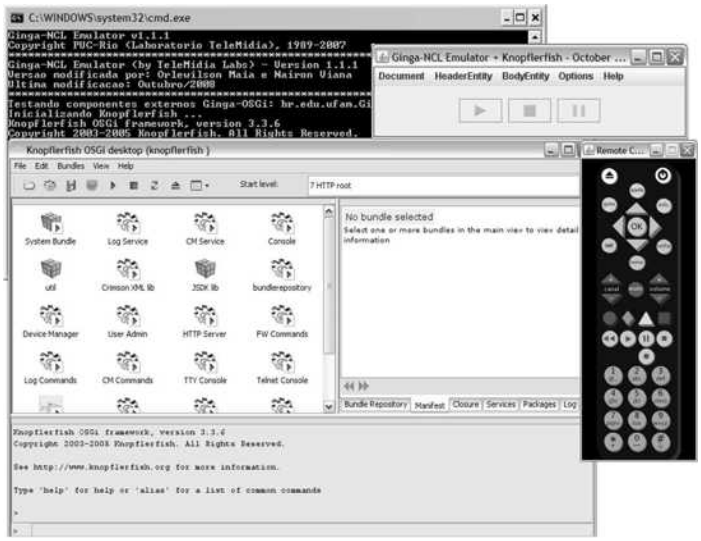


Figura 6.23: Interface do ambiente integrado Ginga-NCL–OSGi.
FONTE: Do Autor.

6.2.2 Configuração dos Componentes e Interação na Plataforma

Conforme citado no Capítulo 5 são cinco os principais componentes do modelo Ginga-NCL–OSGi: 2 *Adapters* (*GingaNCL2OSGiAdapter* e *OSGi2GingaNCLAdapter*), uma ponte (*ContextBridge*) e dois *bundles* (*BundleContextExporter* e *OSGi2GingaNCLBundle*). Os componentes *bundle* são visíveis por toda a plataforma, pois são instalados diretamente no *Knopflerfish*. Já os componentes *Adapter* e a ponte foram organizados de outro modo para que pudessem ser acessíveis pelos ambientes Ginga-NCL Emulator e Knopflerfish. Para isso, esses componentes foram alocados em uma biblioteca (*GingaNCLOSGiExt.jar*) instalada no diretório padrão *./lib/ext-osgi*. Dessa forma, a ponte e os *Adapters* são compilados junto com os outros componentes, fazendo

parte do ambiente integrado Ginga-NCL-Knopflerfish. As funções implementadas em cada um desses componentes são analisadas a seguir.

Componentes Comuns

No grupo *br.ufam.idtvhn.ncl* está o componente *ContextBridge*. Sua função é semelhante à do *ContextBridge* procedural: armazenar contextos. Entretanto, como apenas o ambiente Knopflerfish requer contexto para o acesso, somente objetos *BundleContext* são armazenados. Sempre que um objeto *GingaNCL2OSGiAdapter* ou *OSGi2GingaNCLAdapter* precisar acessar o Knopflerfish ele recupera um contexto de *bundle* no *ContextBridge*. Já *BundleContextExpoter* é o *bundle* que detem o contexto padrão para acesso ao domínio OSGi-Knopflerfish que é passado ao *ContextBridge* durante sua ativação (método *start* do *BundleContextExporter*) no *framework*.

Modelo Ginga-NCL–OSGi

Na biblioteca de componentes-ponte (*GingaNCLOSGiExt.jar*), no grupo funcional *br.ufam.idtvhn.ncl.gingancl2osgi* estão os componenets que permitem configurar o acesso aos serviços OSGi usando scripts NCL. Os componentes *GingaNCL2OSGiAdapter*, *GingaNCL2OSGiPlayer* e *ContextBridge* dão suporte ao processo.

O *Adapter* do modelo possui funções para a inicialização do objeto *GingaNCL2OSGiPlayer* que irá gerenciar a apresentação da mídia. Para isso, o *GingaNCL2OSGiPlayer*, ao estender a interface *DefaultPlayerImplementation* (em *br.pucrio.telemidia.ginga.core.player*) possui como principais funções: (1) configuração do container para a exibição da mídia relacionada às informações do serviço OSGi; (2) gerenciamento da apresentação da mídia (iniciar, parar, pausar); (3) tratamento de eventos sobre a mídia (por exemplo, caso alguma tecla do controle remoto seja pressionada enquanto as informações do serviço OSGi estiverem sendo apresentadas) o que permite o controle interativo sobre os serviços OSGi; e (4) funções para acesso e invocação do serviço OSGi (usando o contexto em *ContextBridge*). Numa típica sequência de interação desses componentes, partindo do *middleware* Ginga-NCL para acessar *framework* OSGi, tem-se as seguintes etapas:

Etapa 1 - Configuração do bundle e do serviço: essa etapa é idêntica à correspondente procedural. O *bundle* é configurado e instalado na plataforma Knopflerfish. Uma vantagem disso é que o mesmo serviço OSGi pode ser utilizado tanto por aplicações *xlets* Ginga-J quanto por *scripts* Ginga-NCL. Nesse caso, para efeito ilustrativo, pode-se considerar o mesmo serviço de impressão usado no modelo procedural.

Etapa 2 - Configuração dos componentes-ponte: aqui atuarão *GingaNCL2OSGiAdapter* e *GingaNCL2OSGiPlayer*. Este último, através do método *play()*, invoca o serviço OSGi e exibe as informações. A listagem da Figura 6.24 ilustra o processo: primeiro o container de exibição da mídia é configurado; em seguida o serviço OSGi é acessado e por fim o container apresenta o conteúdo do objeto de serviço OSGi (no exemplo, o nome do objeto *br.ufam.osgi.printer.Printer*).

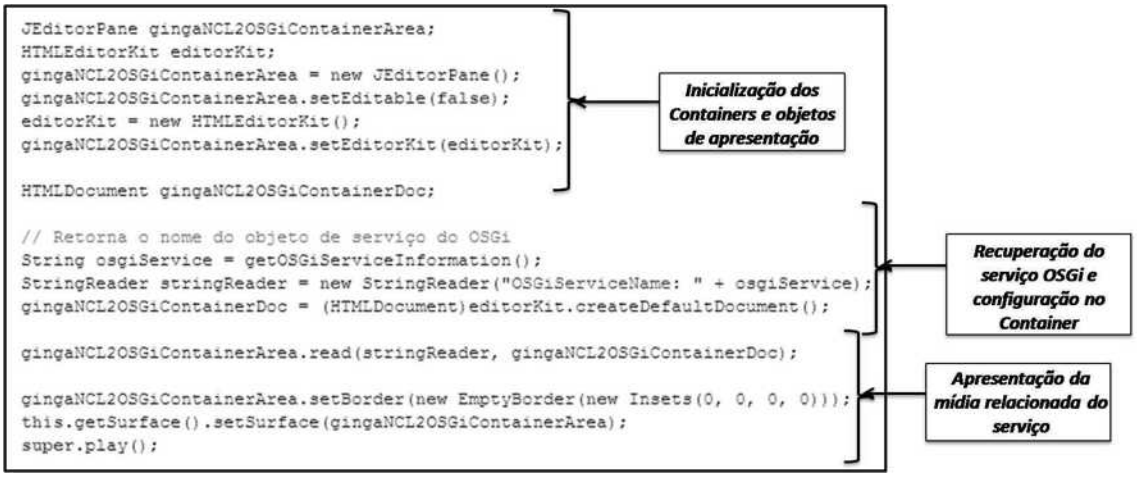


Figura 6.24: Listagem para a apresentação da mídia do serviço OSGi no *GingaNCL2OSGiPlayer*.

FONTE: Do Autor.

Etapa 3 - Configuração do script NCL para acesso a midia do objeto OSGi: no arquivo NCL interpretado pelo middleware, é criada uma região e definido o tempo para a apresentação da mídia do serviço OSGi. Para que a mídia do *GingaNCN2OSGiPlayer* seja reconhecida no NCL a mesma configuração apresentada na Figura 6.21 deve ser seguida, ou seja, os arquivos *ctrldefs.ini* e *mimedefs.ini* em *./gingaNclConfig.players/players* devem ter referência aos *Adapters* da ponte. Para isso foi criado um tipo associado ao *GingaNCL2OSGiAdapter*, o *application/x-ginga-OSGiadapter* (da mesma forma foi criado o correspondente ao *OSGi2GingaNCLAdapter*,

o *application/x-OSGi-gingaadapter*). A Figura 6.25 ilustra a interface da aplicação desse modelo, no ambiente GingaNCL Emulator–Knopflerfish onde são exibidas duas mídias, uma para o serviço OSGi e uma mídia de vídeo.

Figura 6.25: Exibição da mídia relacionada do serviço OSGi no ambiente GigaNCL-Knopflerfish.

FONTE: Do Autor.

Modelo OSGi–Ginga-NCL

Os componentes do grupo funcional *br.ufam.idtvhn.ncl.osgi2gingancl* da biblioteca de componentes-ponte (*GingaNCLOSGiExt.jar*), habilitam a exportação de uma interface do domínio Ginga-NCL para o OSGi. Esta funcionalidade é útil quando, por exemplo, um *bundle* OSGi necessita exibir informações diretamente na tela do Gingan-NCL: para isso ele acessa o objeto exportado que contém funções para o envio de texto ou imagens para a tela da TVDI. Além dos componentes do GingaNCLOSGiExt (*OSGi2GingaNCLAdapter*, *OSGi2GingaNCLPlayerAdapter*), *ServiceObject* e *ServiceObjectImpl*) também dá suporte ao modelo o *OSGi2GingaNCLBundle*, instalado no domínio OSGi.

Como exemplo de um cenário que ilustra a utilização desses componentes pode-se propor a seguinte situação: uma aplicação NCL apresenta três componentes de mídia (vídeo, representado por *JMFVideoPlayerAdapter*; html, associado a *HTMLPlayerAdapter*; e imagem, associado a um objeto *ImagePlayerAdapter*) e um

bundle do domínio OSGi precisa obter informações sobre essas mídias. A Figura 6.26 ilustra o processo de interação entre os componentes, composto dos seguintes passos:

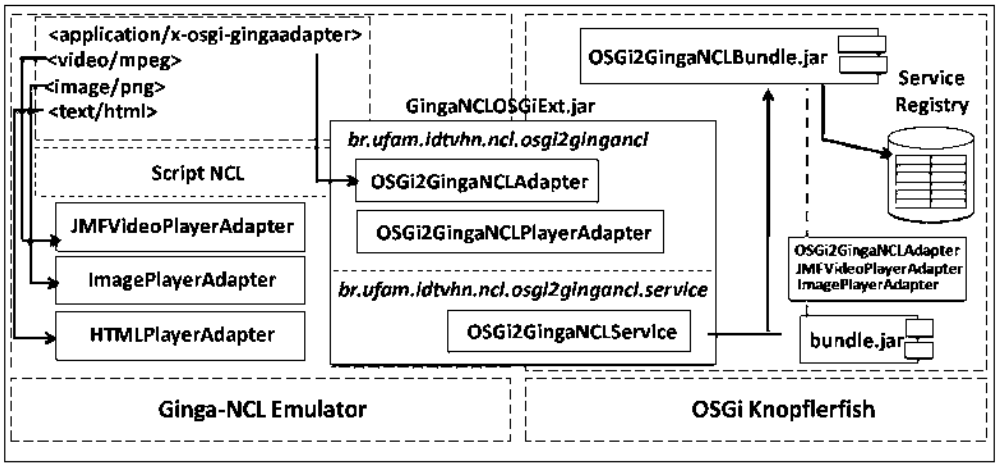


Figura 6.26: Componentes criados interagindo na plataforma Ginga-NCL–OSGi. Exportando um objeto do Ginga-NCL para o OSGi.

FONTE: Do Autor.

Etapa 1 - Configuração do script NCL: essa etapa consiste na criação do arquivo NCL fazendo referência aos objetos de mídia que serão exibidos na tela. Para tanto serão definidas, para cada mídia, uma região (tag *<region/>*), um descritor (tag *<descriptor/>*) e um nó de mídia (tag *<media/>*). A região define as propriedades da tela de apresentação da mídia como dimensões e transparência; o descritor é a relação entre a mídia e a região, definindo por exemplo por quanto tempo a mídia será apresentada; já ós nós de mídia fazem referência para os *Adapters*, de cada mídia: para vídeo tem-se o valor *video/mpeg*; para imagem, *image/png*; para o texto, *text/html* e para o objeto NCL exportado ao OSGi tem-se *application/x-osgi-gingaadapter*.

Etapa 2 - Configuração dos componentes-ponte no Ginga-NCL e no OSGi: aqui é criado uma interface (*OSGi2GingaNCLService*) que contém métodos para acessar os objetos *JMFVideoPlayerAdapter*, *ImagePlayerAdapter* e *HTMLPlayerAdapter*. Uma vez configurado o *OSGi2GingaNCLService*, o par *<OSGi2GingaAdapter*, *OSGi2GingaPlayerAdapter>* entra em ação: quando a mídia definida em *application/x-osgi-gingaadapter* é iniciada, o *OSGi2GingaPlayerAdapter* (em seu método principal *play*) acessa o *ContextBridge* para recuperar um serviço de *OSGi2GingaNCLBundle*. Com esse serviço o *OSGi2GingaPlayerAdapter* repassa o

OSGi2GingaNCLService ao *OSGi2GingaNCLBundle*, que trata a funcionalidade Ginga-NCL de *OSGi2GingaNCLService* como um serviço do OSGi.

Etapa 3 - Configuração do bundle para acesso às informações do OSGi2GingaNCLService: nessa última etapa, um *bundle* instalado no Knopflerfish acessa o serviço de *OSGi2GingaNCLBundle*, onde obtém funções para acesso direto ao objeto *OSGi2GingaNCLService* do Ginga-NCL. Assim, ao invocar as funções do objeto *OSGi2GingaNCLService*, o *bundle* poderá acessar os objetos de imagem, vídeo e texto que são exibidos no Ginga-NCL. A Figura 6.27 ilustra a interface final do aplicativo construído sobre as etapas apresentadas.

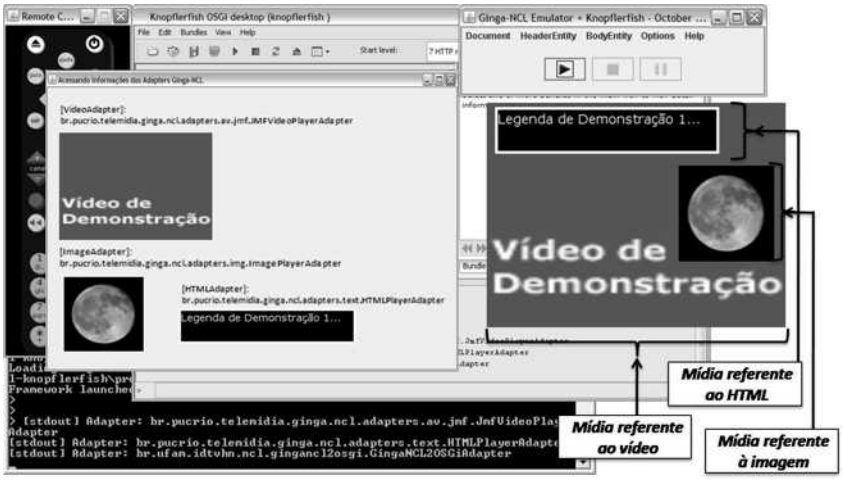


Figura 6.27: *Bundle* OSGi acessando objetos de mídia no ambiente Ginga-NCL-Knopflerfish. FONTE: Do Autor.

A seguir serão descritos as tecnologias associadas aos mecanismos de melhoria adicionados à rede OSGi e como foram integrados às plataformas Ginga-J–OSGi e Ginga-NCL–OSGi.

6.3 Mecanismos de melhoria

Os mecanismos apresentados aqui não foram criados com o objetivo de resolver possíveis problemas da solução de integração Ginga-OSGi. Ao contrário, são mecanismos adicionais que incrementam o modelo e oferecem mais funcionalidades a serem exploradas pelo programador de *xlets* Ginga-J ou de *scripts* Ginga-NCL.

6.3.1 Esquema de invocação de serviços

Uma questão importante que surge para os modelos GingaJ/NCL-OSGi é a invocação de métodos dos objetos de serviço OSGi. No domínio OSGi todos os serviços têm seus métodos e nomes específicos que serão acessados por *xlets* do Ginga-J ou *Adapters* do Ginga-NCL. Dentro dessa situação, a questão que pode ser vista como um possível problema é: como o programador de aplicações saberá os nomes dos identificadores para os serviços que estão disponíveis na plataforma? Mais ainda: para cada serviço, como o programador de TVDI saberá quais seus métodos, parâmetros e como devem ser invocados? No esquema geral do modelo apresentado no capítulo anterior, o programador de *xlets* e o provedor de serviços OSGi não são necessariamente a mesma entidade, ou seja, nem sempre o programador saberá essas informações. No modelo implementado até aqui para que um *xlet* utilize um serviço OSGi é necessário:

- *O Alias para o objeto correspondente no IXCRegsitry*: ao realizar a operação *bind* no IXC, para a recuperação de um objeto remoto referente ao *WrappedProxyObject*, o *Alias* do objeto do IXC deve ter o mesmo valor que o definido na propriedade *OSGiGingaJRegister.IXC_ALIAS*, configurada no objeto *java.util.Properties* associado ao serviço.

- *O nome do método a ser invocado*: uma vez passado o *Alias* do objeto, o método *invoke* de *WrappedProxyObject* exige que se passe o nome do método e uma lista com os tipos e objetos de seus parâmetros. Como pode haver vários serviços na rede OSGi que podem ser acessados por aplicações (*xlets*, *scripts* NCL, ou *bundles*), para cada um deles há a necessidade de ter previamente essas informações.

Para contornar essas dificuldades criou-se um mecanismo de centralização e invocação indireta de serviços OSGi. O esquema apresenta uma entidade central, denominada *Service Manager* que mantém um repositório de referências a todos os serviços registrados no OSGi. O *Service Manager* exporta um único serviço com funções para acessar os demais serviços da rede OSGi. Conhecendo apenas o *Service Manager*, o programador de *xlets* ou *scripts* NCL acessará seus métodos padrão para invocar

indiretamente qualquer outro serviço da rede OSGi. O *Service Manager* apresenta as seguintes funcionalidades:

- Um *Alias* padrão, que deverá ser reconhecido pelo programador de *xlets/scripts* relativo ao objeto de serviço *Service Manager*.
- Um *Listener* para eventos sobre o OSGi *Service Registry*, que identificará se o serviço registrado deve ser armazenado no repositório do *Service Manager* para ser disponibilizado aos aplicativos de TVDI.
- Métodos padrão para: listar os serviços disponíveis no repositório, obter nomes dos métodos e tipos dos parâmetros do serviço e executar um método associado a um serviço.

Para um aplicativo de TVDI acessar um serviço da rede do *Service Manager*, primeiro é encontrada a referência ao objeto de serviço *ServiceManager*. Em seguida a lista de serviços disponíveis é obtida e, para um dado serviço é solicitado ao *Service Manager* que execute um dado método, com os parâmetros especificados. Assim é realizada a invocação indireta.

6.3.2 Esquema de gerenciamento de dependências

O mecanismo do *Service Binder*, proposto por Cervantes e Hall (2003) consiste de um bloco adicional de componentes (representados por um *bundle* do *framework* OSGi) que simplificam o desenvolvimento de serviços OSGi e automatizam o gerenciamento de dependências entre serviços.

O ambiente OSGi, como implementação da arquitetura orientada à serviços (ou SOA, *Service Oriented Architecture*) requer que as aplicações desenvolvidas sejam entregues à plataforma na forma de componentes responsáveis pela publicação, descoberta e utilização de serviços, além da adaptação à disponibilidade dos serviços que estão sendo usados em tempo de execução. Essa adaptabilidade requer um mecanismo para o gerenciamento de dependências.

O *Service Binder* concentra toda a lógica de gerenciamento de dependências em um único *bundle* instalado sobre o OSGi. Para utilizar o mecanismo, o modelo define alguns componentes que devem ser configurados de modo a garantir a criação e remoção das dependências entre os serviços. Os principais componentes do modelo são:

Instância de serviço: um objeto que representa a relação de dependência. Sempre que uma dependência é estabelecida essa instância é criada e, dependendo da relação entre as entidades da dependência (o componente que *usa* e o componente que *provê* o serviço) a instância pode assumir três estados: *válido*, *inválido* e *destruído*.

Descritor da instância: um arquivo XML que descreve algumas propriedades das instâncias a serem criadas na relação de dependências. Para cada *bundle* que deseja provê ou utilizar um serviço por meio do *Service Binder*, um arquivo XML deve ser criado especificando quais componentes serão *publicados* (disponibilizados como serviços) e quais serão requeridos. Para isso é criado no arquivo descritor do *bundle* (*MANIFEST.MF*) um novo campo (*Metadata-Location*) que especifica a localização do descritor da instância.

```
<?xml version="1.0" encoding="UTF-8"?>
<bundle>
    <instance class="org.foo.impl.SpellCheckServiceImpl">
        <service interface="org.foo.service.SpellCheckService"/>
        <property name="version" value="1.0" type="string"/>
        <requires service="org.foo.service.DictionaryService"
            filter="(Language=*)"
            cardinality="1..n"
            policy="dynamic"
            bind-method="addDictionary"
            unbind-method="removeDictionary"/>
    </instance>
</bundle>
```

Figura 6.28: Esquema de configuração do XML para o descritor de instância do *Service Binder*.

FONTE: Adaptado de Cervantes e Hall (2003).

O exemplo da Figura 6.28 ilustra um descritor de instância para um serviço de verificação ortográfica. Entre as tags que podem ser notadas, merecem destaque: *instance*, a instância do serviço, que é representada pela interface (*org.foo.impl.SpellCheckServiceImpl*); *service* que representa a interface do serviço; *requires* que indicará de qual serviço o *SpellCheckService* dependerá (nesse caso,

do *DictionaryService*); *filter*, que indica o mecanismo LDAP para recuperar o *DictionaryService*; *cardinality*, que representa quantas instâncias de *DictionaryService* podem ser usadas por um *SpellCheckService*; *policy*, que indica se a instância de serviço *SpellCheckService* será destruída (*policy=static*) ou mantida (*policy=static*) quando da indisponibilidade de *DictionaryService*; e por fim os métodos de *SpellCheckServiceImpl* para instalar (*bind-method*) ou remover (*unbind-method*) a instância de serviço.

Gerenciador de instâncias: o *instance manager* (IM) é responsável pela criação das instâncias e manutenção de seus estados. O IM, dependendo das especificações do descritor da instância e das mudanças do ambiente, pode validar, invalidar ou destruir a instância.

No modelo o *Service Binder* foi adicionado da seguinte forma: as dependências foram estabelecidas entre o *home.jar* e os demais bundles da rede (*printer.jar*, *webcam.jar*, *lamp.jar*), sendo criados arquivos descritores para cada um dos *bundles*. Como o modelo da plataforma já garante a exportação do serviço de *home.jar* ao ambiente Ginga, nenhuma modificação nesse ambiente foi necessária.

6.4 Resumo

Neste capítulo foi construída a plataforma de integração Ginga-OSGi modelada no capítulo anterior. Tanto para a abordagem procedural, como para a declarativa, foi apresentado o processo que vai desde a escolha das plataformas, o esquema de integração e exemplos de casos de teste que ilustram o compartilhamento de objetos de serviços, sejam serviços OSGi, ou objetos de serviço de *xlet* Ginga-J ou *Adapter* Ginga-NCL.

No mecanismo procedural Ginga-J-OSGi foram apresentadas as ferramentas XleTView e Knopflerfish, bem como o processo de modificação de seus componentes e adição de novas tecnologias de forma a permitir que os dois emuladores executem num ambiente comum. O mecanismo de integração XleTView-Knopflerfish foi apresentado como um resultado da análise dos mecanismos empregados pelas principais referências base. Por fim, foram analisadas as implementações dos componentes do núcleo Ginga-J-OSGi e como interagem no ambiente.

Já para o mecanismo declarativo foram introduzidos alguns conceitos sobre o *Ginga-NCL Emulator*, principalmente relacionados à como o ambiente identifica e interpreta cada mídia. Foi construído o ambiente comum Ginga-NCL-OSGi, baseado nos *Adapters* Ginga-NCL, para o qual os componentes núcleo foram testados sob os dois meios de interação Ginga-NCL-OSGi através de alguns exemplos.

Ao final do capítulo, a construção dos dois mecanismos de melhoria da rede OSGi foi detalhada, com a apresentação dos elementos principais que constituem os dois modelos, quais as funções de cada, e como foram adicionados no modelo geral Ginga-OSGi.

No capítulo seguinte será construído o ambiente de teste sobre o qual foram realizados os cenários que validam as plataformas Ginga-J-OSGi e Ginga-NCL-OSGi. Materiais, dispositivos eletrônicos, procedimentos e alguns resultados serão analisados em seguida.

Capítulo 7 - Estudos de Caso

Este capítulo apresenta os estudos de caso que foram criados como provas de conceito para a plataforma de integração TVDI-HG desenvolvida. Os cenários servem como ponto de partida para projetos mais complexos sobre a plataforma, podendo ser a base para a construção de um protótipo inicial de ambiente de *Home Network* que, em longo prazo, pode se tornar em um produto e contribuir para a melhoria da qualidade de vida do usuário doméstico da TV Digital Interativa.

É importante ressaltar que, como a plataforma criada se trata de uma API implementada sob o ambiente comum Ginga-OSGi, os cenários construídos exigem mais do que a plataforma, ou seja, há a necessidade de *hardware* (microcontroladores, placas, dispositivos como impressoras, câmeras, celulares, etc.) e *software* (APIs que implementam alguns protocolos de acesso à hardware como *Serial*, *WiFi*, *Bluetooth*, ou APIs de acesso aos dados da rede TCP/IP, etc) específicos. Para cada cenário, a plataforma Ginga-OSGi serve para compartilhar os objetos de serviços de um domínio para o outro, sejam referentes à conexões bluetooth (num cenário de interação TV-Celular), serial (num cenário TV-impressora) ou *Ethernet* (num cenário de monitoramento remoto, por exemplo). A lógica principal dos cenários estará, portanto, em como serão utilizadas essas APIs e tecnologias adicionais em conjunto com o ambiente TVDI-HG baseado no *middleware* Ginga.

Seguindo a estratégia traçada desde o Capítulo 5, que divide o problema em duas abordagens distintas (procedural Ginga-J–OSGi e declarativa Ginga-NCL–OSGi), foram criados dois grupos para os cenários de teste. O primeiro utiliza a máquina de execução Ginga-J e o segundo, a máquina de apresentação Ginga-NCL. Para cada um deles, um modelo geral será apresentado, seguido de um detalhamento de cada componente

e apresentação em etapas de todo o processo que leva desde a definição das tecnologias, integração com a plataforma até a execução e obtenção dos resultados. Para os cenários procedurais, foram escolhidos: (1) monitoramento remoto de câmeras; (2) controle de acesso, com câmeras e dispositivo embarcado e (3) monitoramento de temperatura. Já para o ambiente declarativo os dois mecanismos de interação foram testados num único ceário de impressão de documento sincronizada com a mídia de vídeo.

Com este capítulo espera-se testar os ambientes Ginga-J–OSGi e Ginga-NCL–OSGi. Com relação aos mecanismos de melhoria, estes podem também ser implementados para cada cenário, pois, assim como a plataforma, tratam apenas dos objetos de serviço que são trocados entre os dois domínios. Assim, o *ServiceManager* da invocação indireta de serviços e o *ServiceBinder* do gerenciamento automático de dependências podem estar presentes em cada um dos cenários de teste, sem interferir na execução correta do processo.

7.1 Cenários Ginga-J–OSGi

Aqui serão descritos os elementos de hardware e software que fazem parte dos cenários que utilizam a parte Ginga-J do modelo de convergência com o OSGi. É importante ressaltar que para estes e para os cenários declarativos, foi utilizada a infra-estrutura (máquinas, dispositivos, software, rede, etc.) dos laboratórios do Centro de P&D em Tecnologia Eletrônica e da Informação (CETELI), vinculado ao Mestrado em Engenharia Elétrica da Universidade Federal do Amazonas.

7.1.1 Cenário 1: Monitoramento remoto de câmeras

Visão Geral

O primeiro cenário procedural utiliza dispositivos de imagens para criar um ambiente de monitoramento remoto de câmeras de uma rede doméstica centrado na TVDI. Segundo o modelo desse cenário, ilustrado na Figura 7.1, um usuário, ao interagir com a interface da TVDI pode acessar serviços remotos de dispositivos OSGi localizados em diferentes regiões da residência. A plataforma comum Ginga-J–OSGi dá suporte para

aplicações de TVDI acessarem serviços de dispositivos de imagem na rede TCP/IP e por sua vez realizarem operações remotas sobre cada dispositivo (conectar, ativar, pausar, desativar e obter imagem).

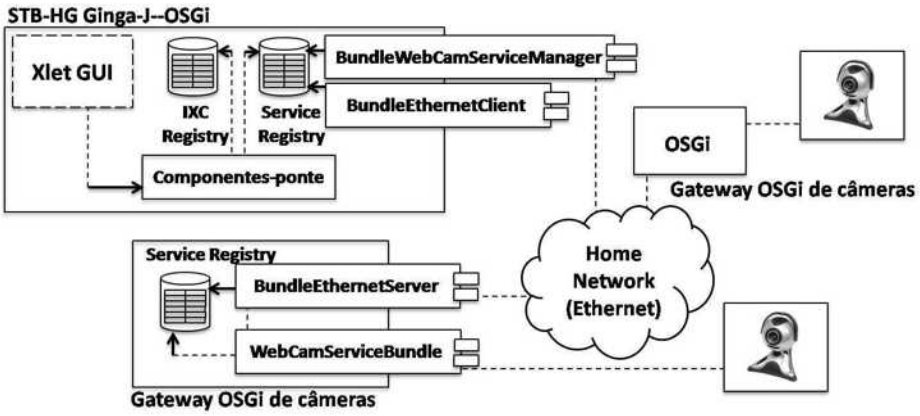


Figura 7.1: Visão geral do cenário de monitoramento de câmeras.
FONTE: Do Autor.

No modelo desse cenário, do ponto de vista do OSGi, figuram dois serviços principais: *EthernetService*, que permitirá a conexão do STB-HG com outros dispositivos da rede e *WebCamService* que está associado diretamente ao dispositivo de captura de imagens. O primeiro é instalado no Ginga-J–OSGi; já para o segundo, deve haver uma implementação de referência do OSGi disponível (que pode representar um gateway de acesso a câmeras, por exemplo). Em resumo, o sistema tem uma base cliente-servidor com vários servidores OSGi de dispositivos de imagens, com um único cliente Ginga-J–OSGi.

Do ponto de vista da TVDI, uma interface gráfica baseada em HAVi é construída de modo a permitir que o usuário "navegue" pelas câmeras da rede, podendo realizar operações sobre o objeto registrado no domínio Ginga-J (no *IXC Registry*) que refletirão diretamente sobre o objeto remoto de serviço OSGi. Para isso, no Ginga-J deve estar instalado um serviço, *WebCamServiceManager*, que utiliza o serviço de conexões TCP/IP de *EthernetService* para acessar os dispositivos. Para cada função provida pelo *WebCamServiceManager*, há uma função correspondente mapeada na interface da TVDI, que permitirá ao usuário realizar operações de gerenciamento remoto.

Recursos de Hardware

Como recursos de hardware foram utilizados:

- quatro computadores (três foram utilizados como gateway para o dispositivo de imagem e um para a plataforma STB-HG) com as seguintes configurações: Dell Desktop Dimension, 120GB HD, 1GB RAM, Intel Pentium 4, interfaces USB, Serial e de rede.

- três dispositivos de captura de imagens (um para cada gateway remoto) com as seguintes configurações: resolução 1.3Mp, conexão USB, 24 bits de cores e taxa de captura 30 quadros por segundo.

O cenário foi construído usando a infra-estrutura de rede local *Ethernet* do CETELI. Os três gateways para dispositivos de imagens foram alocados em salas diferentes do prédio, sendo o gateway STB-HG instalado no laboratório de TV Digital do centro.

Recursos de Software

Como descrito no capítulo anterior, no STB-HG estão instalados além das versões da JVM, implementações de referência do MHP (XleTView) e OSGi (Knopflerfish). A plataforma utiliza recursos das APIs de rede da JVM para estabelecer conexões TCP/IP com os outros dispositivos. Usando funções do pacote *java.net.** o gerenciador de serviços do STB-HG consegue realizar conexões TCP ou UDP com servidores da rede, enviar e receber pacotes.

Já do lado dos servidores de dispositivos de imagens, estão instaladas versões do Knopflerfish para o gerenciamento dos serviços. Os mesmos serviços padrão estão presentes (Log, HTTP, System, etc.); porém existem dois serviços essenciais para que o computador funcione como gateway de imagens: um serviço de conexão na rede *Ethernet* e um serviço de acesso ao dispositivo *Web Cam*. O primeiro é implementado com recursos de *java.net.**; o segundo utiliza as funções de acesso à mídias da API JMF *Java Media Framework* 2.1.1e (JMF, 2009).

O JMF fornece recursos para a apresentação de mídias no ambiente da JVM. Através da API é possível gerenciar a exibição de áudio e vídeo em alguns formatos (.avi, .mpeg, .au, .wave, .mp3, etc.), podendo ser realizado controle temporal da apresentação (com informações de tempo e taxa de transmissão), formatação e sincronização entre mídias. A API é fornecida junto com o ambiente *JMF Studio*, uma ferramenta que permite testar algumas funcionalidades do JMF.

Os principais grupos funcionais do JMF utilizados no projeto foram: *javax.media.format.** para formatação e suporte a tipos de mídia; *javax.media.control* para o controle da apresentação temporal da mídia; e *com.sun.media.protocol.vfw.** que oferece funções para a captura e armazenamento de imagens de dispositivos (Web Cams), como interface para acesso a drivers e gerenciamento de fluxo do conteúdo.

Descrição do Processo

Essa etapa subdivide-se em duas outras: primeiro, a modelagem do sistema, onde são identificadas algumas funções e representadas em diagramas de casos de uso UML (*Unified Modeling Language*). Os casos de uso identificados são implementados na segunda etapa, na qual são definidos os componentes (diagramas de classes), suas configurações e como interagem no ambiente Ginga-OSGi (diagramas de seqüência).

A Figura 7.2 resume as principais funções do sistema em quatro casos de uso. Os dois primeiros, *ativar contexto de bundle* e *ativar contexto de xlet*, são realizados pelo framework OSGi e middleware Ginga-J, respectivamente, e consistem em inicializar o objeto ponte do modelo com os contextos. O terceiro caso de uso (*cadastrar dispositivos*) refere-se ao cadastro dos dispositivos da rede acessíveis ao STB-HG e o último (*obter informações do dispositivo*) está associado ao processo de utilização das funções dos dispositivos da rede, que envolve desde a configuração dos serviços OSGi no servidor, exportação do serviço correspondente no cliente STB-HG, e por fim a conexão e gerenciamento remoto dos dispositivos.

As funções dos casos de uso foram agrupadas em etapas nas quais são apresentadas as classes que compõem o cenário. Sendo assim, para um aplicativo de TVDI acessar o serviço de imagens na *Home Network*, são necessárias as seguintes etapas:

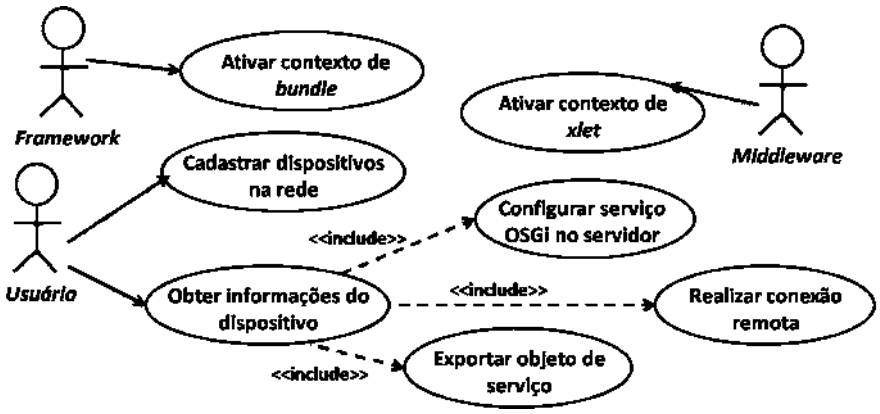


Figura 7.2: Casos de uso para o cenário de monitoramento de câmeras.
FONTE: Do Autor.

Passo 1: Configuração dos bundles no servidor OSGi: essa etapa é semelhante à descrita na seção 6.1.3. Em cada gateway de dispositivos de imagens estão presentes dois *bundles*. O primeiro, *EthernetServerBundle* contém recursos de *java.net.** que o permite receber conexões TCP/IP em uma porta específica. O *EthernetServer* está associado ao segundo *bundle*, o *WebCamServiceBundle*, que encapsula operações para acesso ao dispositivo de imagens utilizando funções da API JMF. Uma vez servindo a um cliente na rede TCP/IP, o *EthernetServer* pode receber comandos e repassá-los ao dispositivo utilizando o serviço de *WebCamServiceBundle*.

Passo 2: Cadastro de dispositivos na rede: uma vez realizada a etapa 1 em cada servidor da rede, a etapa seguinte consiste no cadastro de informações sobre cada servidor. No domínio OSGi, o *BundleWebCamServiceManager* provê uma interface ao operador do gateway OSGi que permite obter/salvar em XML dados do tipo: identificação do servidor, ip e porta. Além destas, o *bundle* também permite utilizar os parâmetros numa conexão de teste, por meio do serviço de um *EthernetClientBundle*, que representa o cliente para as conexões TCP/IP com um *EthernetServerBundle*. Essas informações serão exibidas ao usuário de TVDI e utilizadas para o estabelecimento das conexões entre o cliente STB-HG e os servidores.

Passo 3: Exportar serviço OSGi ao Ginga-J: o *BundleWebCamService-Manager* provê o serviço que será exportado (*WebCamServiceManager*) ao ambiente Ginga-J como um *WrappedProxyObject*, segundo o processo discutido nos capítulos

anteriores. O serviço será usado pelo *xlet*, onde para cada comando selecionado pelo usuário, comandos correspondentes em *WrappedProxyObject*, *WebCamServiceManager*, *EthernetClientBundle*, *EthernetServerBundle* e *WebCamServiceBundle* serão executados.

Passo 4: Configurar a interface do xlet: nessa etapa, componentes HAVi, AWT, DAVIC e JavaTV são usados para criar a interface com o usuário, com botões que permitirão a navegação entre os servidores, além de uma área da tela que permitirá a visualização do objeto de imagem obtido de uma solicitação a um desses servidores remotos.

A interação entre os componentes do sistema, de modo a garantir a realização do cenário, é representada no diagrama de seqüencia da Figura 7.3. O diagrama representa o caso de uso da utilização das funções do dispositivo na rede. No diagrama são representadas duas ações principais: uma solicitação de conexão por parte do *xlet* e uma solicitação da imagem do dispositivo conectado.

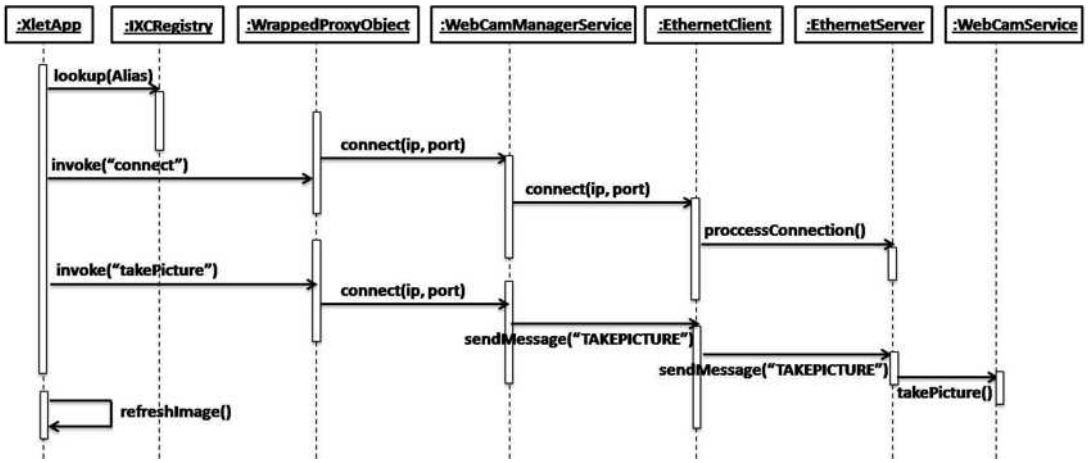
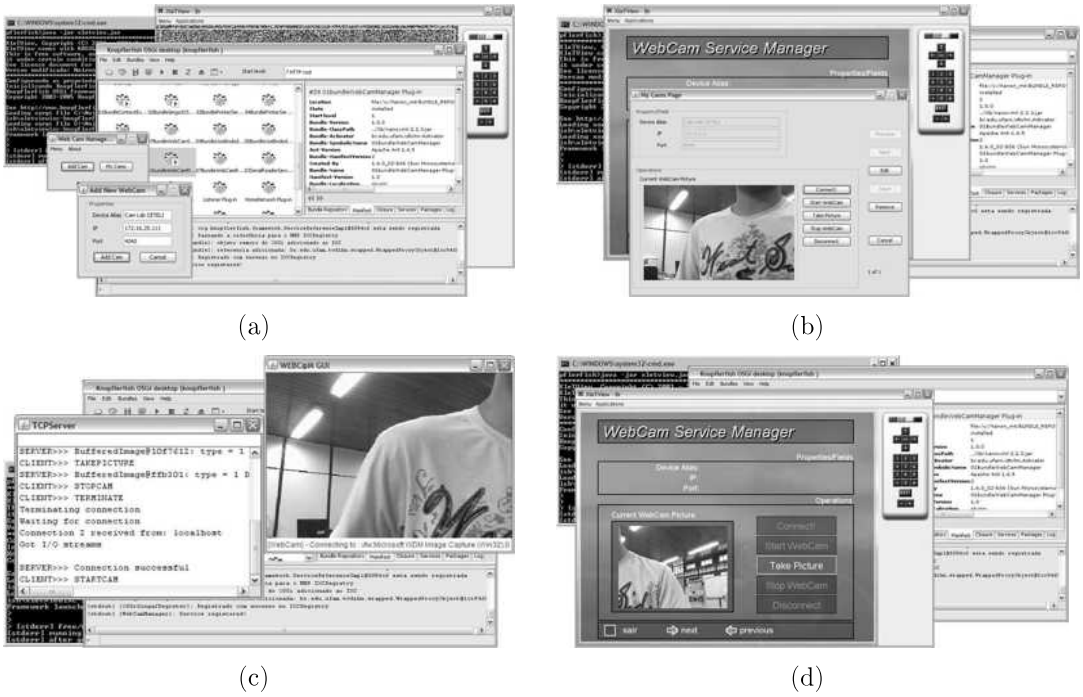


Figura 7.3: Diagrama de seqüência para o cenário de monitoramento de câmeras.
FONTE: Do Autor.

Na primeira ação, um *xlet* acessa o *IXCRegistry* para recuperar o objeto de serviço *WrappedProxyObject*. Em seguida, é chamado do objeto o método para conexão que é repassado ao *WebCamManagerService* e deste para o *EthernetClienteBundle*. Este *bundle* repassa a chamada ao *EthernetServerBundle* que invoca o método para a conexão. A segunda ação ocorre de forma semelhante, com a diferença dos métodos que são chamados entre os objetos (ao invés do método *connect* é chamado o *takePicture*).

Resultados

As figuras abaixo ilustram algumas telas do sistema para as respectivas funções: cadastro de dispositivos no sistema (Figura 7.4(a)); configuração da conexão do cliente com o servidor do dispositivo de images (Figura 7.4(b)); processamento do comando para a inicialização do dispositivo do servidor de imagens (Figura 7.4(c)); e obtenção da imagem do servidor e atualização no aplicativo *xlet* (Figura 7.4(d)).



(a) (b)

(c) (d)

Figura 7.4: Cenário de monitoramento de câmeras. Telas do sistema.
FONTE: Do Autor.

7.1.2 Cenário 2: Controle de acesso

Visão Geral

O segundo cenário procedural ilustra um sistema de controle de acesso centralizado no STB-HG. Segundo o modelo, um usuário de TVDI pode receber (enquanto assistindo ou interagindo com outra aplicação) uma solicitação de um outro usuário que deseja, por exemplo, entrar na residência. Este último interage com três dispositivos: um computador, que representa um gateway de segurança da *Home Network*; um dispositivo

embarcado, com um teclado para receber entradas do usuário e um display LCD para mostrar o andamento da solicitação; e uma WebCam, que é utilizada para enviar ao STB-HG, uma imagem do usuário que faz a solicitação. Ao receber a mensagem na tela, o usuário da TVDI pode responder à solicitação, enviando uma mensagem de retorno ao segundo usuário, que é mostrada no dispositivo embarcado. Esse cenário, ilustrado na Figura 7.5, foi apresentado no Capítulo 1.

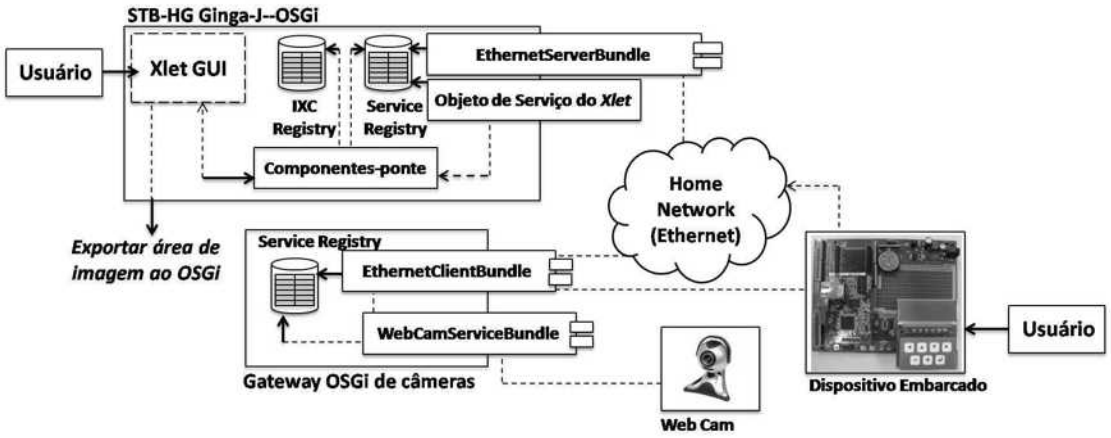


Figura 7.5: Visão geral do cenário de controle de acesso.
FONTE: Do Autor.

Assim como o cenário anterior, este representa um modelo cliente-servidor. Entretanto, o STB-HG é o servidor para as conexões feitas pelo gateway de segurança. No STB-HG, os dois mecanismos de interação descritos na seção 5.2 são utilizados: um *xlet* exporta sua funcionalidade ao OSGi, de modo a garantir que uma área da tela da TVDI possa ser usada por *bundles* do OSGi para atualizar as informações da solicitação; ao mesmo tempo, um *bundle* (*EthernetServerBundle*) deve exportar seus serviços ao domínio Ginga-J de modo viabilizar o envio da mensagem de retorno do *xlet* para o dispositivo embarcado conectado ao gateway de segurança.

Já no gateway de segurança, na implementação OSGi devem figurar três serviços: o primeiro (em *EthernetClientBundle*) permite a conexão do gateway à rede; o segundo (*WebCamService*) permite a conexão ao dispositivo de imagens; e o último permite a interação com o dispositivo embarcado. Os dois primeiros serviços são configurados da mesma forma que no cenário anterior (usando APIs *java.net* e JMF), já no último uma

conexão de rede é estabelecida entre o gateway de segurança e o dispositivo, no qual é carregado um programa em C que configura as interfaces de teclado, LCD e rede.

Recursos de Hardware

Foram utilizados dois computadores (um como gateway de segurança e outro como STB-HG) e uma WebCam com as mesmas configurações anteriores. Como dispositivo embarcado foi utilizado o RabbitCore RCM 3200 Kit (Rabbit, 2009), um sistema baseado em módulos que permite o rápido desenvolvimento de sistemas embarcados provendo uma série de funções que permitem ao programador criar aplicações utilizando recursos da Serial, Ethernet, WiFi, IrDA, teclado, display LCD, entre outros.

Algumas das características do Kit Rabbit: microprocessador Rabbit 3000 44.2MHz; 52 linhas de I/O 5V: 44 configuráveis para I/O, 4 entradas fixas e 4 saídas fixas; duas entradas adicionais, uma analógica e outra digital; 10 timers de 8 bits e um timer de 10 bits com dois registradores; memória flash 512K, 512K SRAM para execução de programas e 256K SRAM para dados; clock de tempo real; portas Ethernet 10/100Base-T RJ-45; seis portas serial, taxa de transmissão máxima 5.5Mbps; e suporte a IrDa 1.15 Mbps.

Recursos de Software

Os únicos recursos de software diferentes com relação ao cenário anterior são os relacionados ao módulo embarcado Rabbit. Tanto no STB-HG quanto no gateway de segurança são usadas APIs java para conexão TCP/IP e controle dos dispositivos de imagens. Para o módulo embarcado, foi utilizada a linguagem de programação C, fornecida com o kit *Dynamic C Development Kit*, uma IDE que permite a escrita de código em *Dynamic C* (uma variante de C para sistemas embarcados com suporte à compilação rápida e depuração interativa, ou seja, o software que executa no dispositivo pode ser monitorado pelo programador através da IDE).

O kit *Dynamic C* utiliza uma conexão serial para a carga e depuração dos programas. A criação de software para a plataforma não é complexa, considerando que que existem uma série de programas padrão para teste de cada funcionalidade do dispositivo.

A execução de um programa pode ser configurada de três modos na IDE, usando as opções do menu *Options->Project Options*: (1) o programa é codificado para executar somente na placa (*Code and BIOS in FLASH*); (2) o programa executa somente no computador (*Code and BIOS in RAM*); e (3) o programa executa na placa e no computador (depuração dinâmica, opção *Code and BIOS in FLASH, run in RAM*).

Os principais grupos utilizados do ambiente *Dynamic C* foram funções do módulo Ethernet, para a criação de um cliente para conexões com o gateway de segurança (utilizando conexões socket TCP/IP da biblioteca TCPCONFIG); e funções para apresentação de texto e tratamento das entradas do usuário, no teclado e display LCD (em KEYPAD7.LIB e KEYPAD9.LIB).

Descrição do Processo

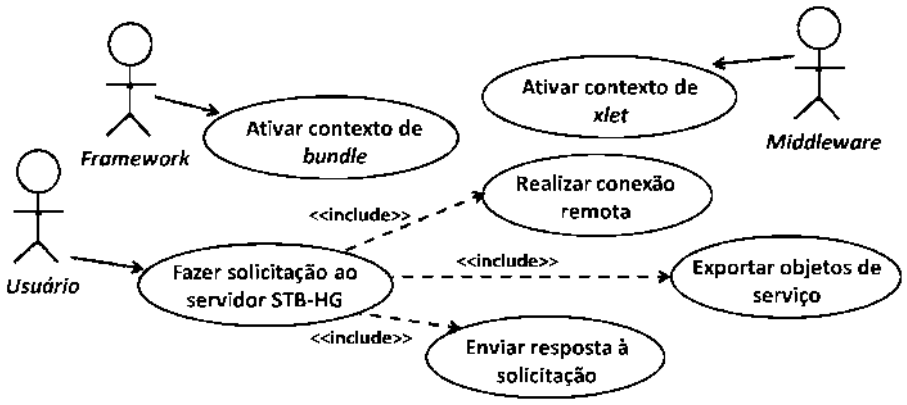


Figura 7.6: Casos de uso para o cenário de controle de acesso.
FONTE: Do Autor.

No diagrama de casos de uso da Figura 7.6, além dos casos padrão (*ativar contexto do bundle* e *ativar contexto do xlet*), presentes em todos os outros cenários, o caso de uso *fazer solicitação ao servidor STB-HG* representa o processo central do cenário. Este é realizado utilizando três outros casos: *realizar conexão remota*, *exportar objetos de serviço* e *enviar resposta à solicitação.*

As funções dos casos de uso foram agrupadas em etapas nas quais são apresentadas as classes que compõem o cenário. Sendo assim, para que o dispositivo embarcado consiga enviar a solicitação ao STB-HG e receber uma resposta, são necessárias

as seguintes a seguir. Na Figura 7.7, os componentes de cada etapa que fazem parte do caso de uso principal são ilustrados.

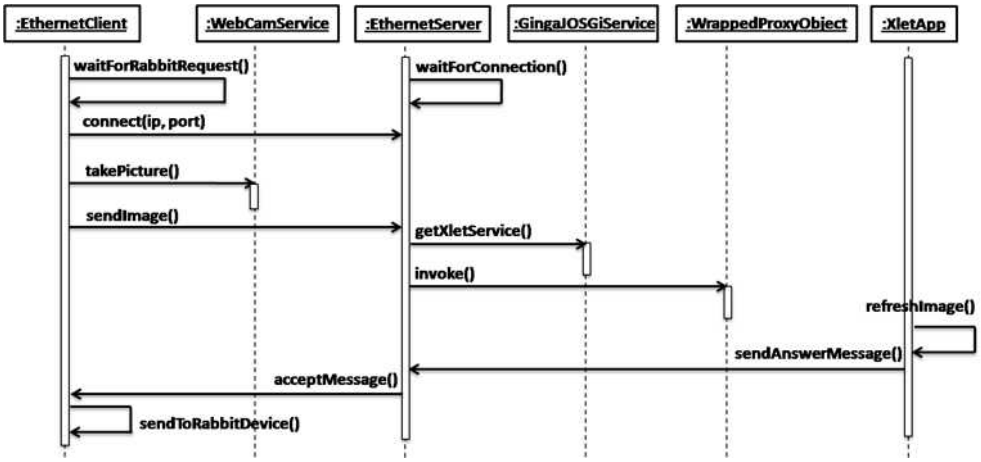


Figura 7.7: Diagrama de seqüência para o caso de uso *fazer solicitação ao servidor STB-HG*. FONTE: Do Autor.

Passo 1: Configurar bundles e serviços OSGi: no gateway de segurança, o componente *EthernetClientBundle* é um servidor para conexões TCP/IP com o dispositivo Rabbit e cliente do STB-HG. Uma vez instalado o serviço deste *bundle* na plataforma, o componente espera requisições do dispositivo. Ao receber uma solicitação, é buscado o serviço de imagem de *WebCamBundle* para a captura da tela do usuário que faz a requisição. Do lado do STB-HG, um *EthernetServerBundle* aguarda conexões e objetos de imagem do *EthernetClientBundle*.

Passo 2: Configurar conexão do dispositivo com o gateway de segurança: essa etapa é realizada no dispositivo embarcado, onde é desenvolvida a aplicação em *Dynamic C*, que utiliza recursos do módulo TCP/IP do kit Rabbit. No aplicativo são configurados IP e porta para conexão com o gateway de segurança, além de um menu interativo que permitirá ao usuário saber do andamento da solicitação.

Passo 3: Exportar objetos de serviço no STB-HG: uma vez recebido o objeto de imagem pelo *EthernetServerBundle*, é necessário acessar uma área da tela da TVDI para mostrá-la juntamente com a aplicação de requisição. Para isso o *EthernetServerBundle* utiliza uma funcionalidade do aplicativo *xlet* do Ginga-J como um serviço OSGi, conforme em 6.1.3.

Passo 4: Configurar a interface do aplicativo xlet e a resposta ao dispositivo: nessa etapa o usuário deve estar apto a responder à solicitação. Para isso, é configurada uma interface Havi e AWT que permite ao usuário da TVDI escolher uma opção de resposta e enviar uma mensagem personalizada ao dispositivo embarcado. Uma vez selecionada a opção de resposta, o *xlet* acessa um serviço OSGi de *EthernetServerBundle* (segundo o mecanismo em 6.1.3) e envia a resposta ao gateway de segurança que repassa ao dispositivo Rabbit.

Na seqüência principal de interação entre os componentes descritos, conforme na Figura 7.7, um objeto *EthernetClient* de *EthernetClientBundle* aguarda a requisição TC/IP do dispositivo embarcado. Após receber a solicitação, é aberta uma conexão *socket* com o objeto *EthernetServer* de *EthernetServerBundle* do STB-HG. O *EthernetClient* então recupera o serviço de imagens e envia pela rede um objeto referente à imagem do usuário que está fazendo a requisição no dispositivo embarcado. Ao receber esses dados, o *EthernetServer* busca um *GingaJOSGiService* para recuperar um *WrappedProxyObject* relativo à uma área da tela do aplicativo *xlet*. Este repassa a resposta do usuário ao *EthernetServer* e deste ao *EthernetClient* que a envia ao dispositivo Rabbit.

Resultados

A Figura 7.8 ilustra algumas telas do sistema para as respectivas funções: dispositivo Rabbit aguarda requisição do usuário (figura 7.8(a)); gateway de segurança recebe conexão do dispositivo e ativa a Web Cam (figura 7.8(b)); mensagem da requisição é recebida na TVDI e usuário escreve resposta (figura 7.8(c)); e dispositivo recebe resposta à solicitação (figura 7.8(d)).

7.1.3 Cenário 3: Monitoramento de temperatura

Visão Geral

O terceiro cenário Ginga-J–OSGi representa um esquema de monitoramento de temperatura. O esquema, ilustrado na Figura 7.9, permite ao usuário de TVDI acessar informações de sensores de temperatura presentes na residência. No STB-HG é conectado

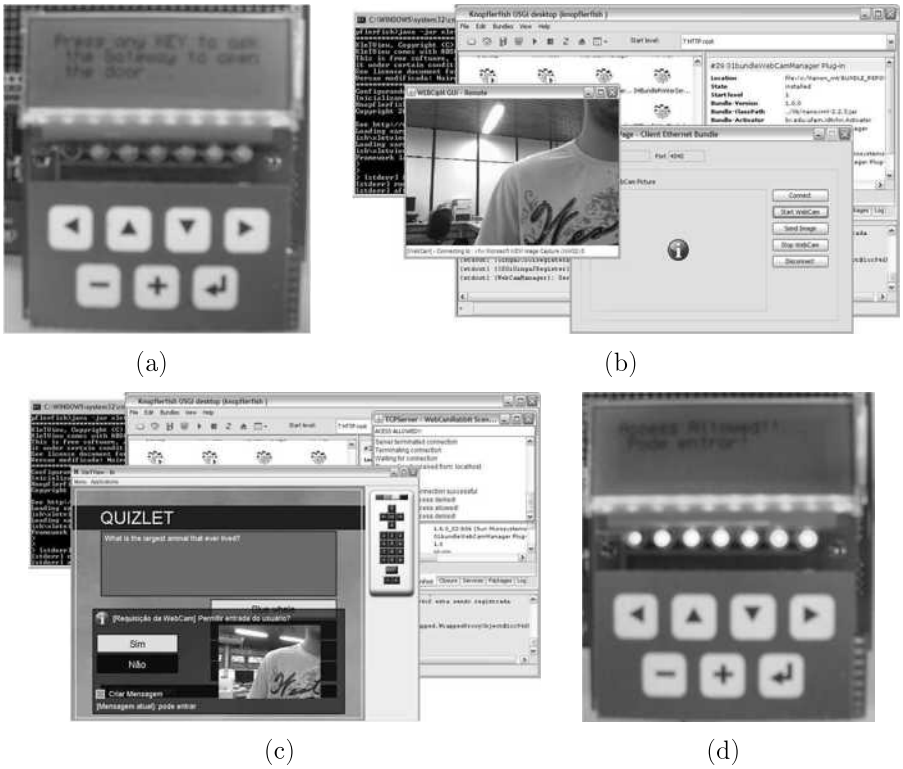


Figura 7.8: Cenário de controle de acesso. Telas do sistema.

FONTE: Do Autor.

um dispositivo embarcado que representa o nó principal da rede de sensores. Aplicações instaladas nesse dispositivo e nos demais nós periféricos (presentes em outros pontos da residência) são responsáveis pela comunicação entre os nós e obtenção dos valores dos sensores de temperatura de cada dispositivo. O nó principal repassa ao STB-HG por meio da interface RS-232, os valores de temperatura obtidos.

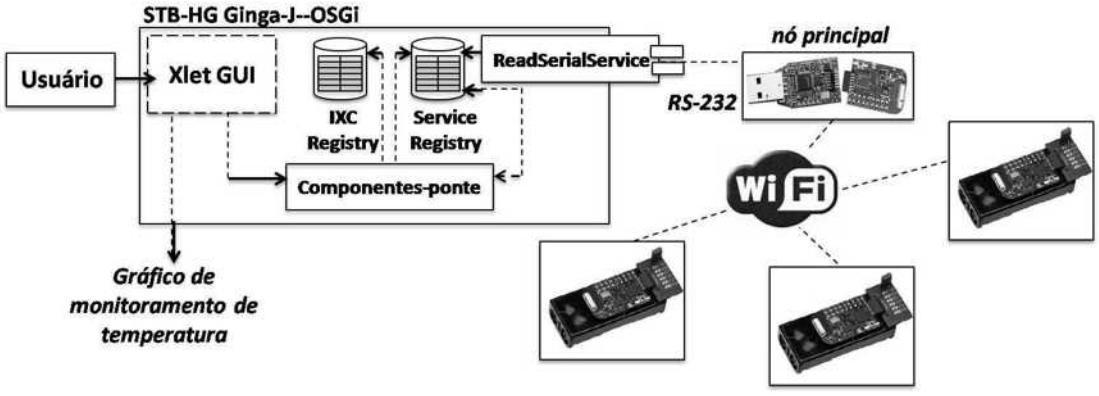


Figura 7.9: Visão geral do cenário de monitoramento de temperatura.

FONTE: Do Autor.

Na plataforma integrada Ginga-J–OSGi o *bundle ReadSerialBundle* é responsável pela configuração do software para acessar as informações do nó principal. Ele define um protocolo onde são tratados alguns parâmetros da conexão serial (taxa de transmissão e bits de paridade, porta, buffer, etc) e o esquema de formatação das informações obtidas. O *ReadSerialBundle* provê ao ambiente OSGi um serviço por meio do qual os dados do sensor são exportados ao domínio Ginga-J e mostrados graficamente na tela da TVDI pelo aplicativo *xlet*.

Recursos de Hardware

Foi utilizado um computador como STB-HG com as mesmas configurações anteriores. Como dispositivos embarcados foram utilizados cinco módulos de dispositivos Texas MSP430, do kit MSP430 EZ430-RF2500 (Texas, 2009) (um módulo nó sensor e quatro nós periféricos). O kit oferece uma plataforma para desenvolvimento de aplicações *wireless* provendo hardware e software para o microcontrolador MSP430F2274. A plataforma apresenta uma interface que permite a comunicação serial com o PC pelo uso da porta USB.

Algumas das características do MSP430 EZ430-RF2500: microcontrolador MSP430F2274 com 32KB Flash, 1KB RAM, 1 USCI (UART, (2) SPI, I2C, IrDA), 200Ksps 10bit-ADC, 2 Amp. Op., Timer de 16bits; transceptor cc2500 de 2.4GHz *multichannel*, 1LEDs, 1 botão *reset*, placa USB removível, interface USB com protocolo *MSP 430 Application UART* para acesso a serial, módulo integrado de sensor de medição de temperatura, voltagem e taxa de sinal.

Recursos de Software

Os recursos de software utilizadas dão suporte à duas funções: carregamento do aplicativo no módulo MSP430 e interface do dispositivo com o ambiente Ginga-J–OSGi. Para a primeira função foram utilizadas IDEs que acompanham o kit MSP430 EZ430. Para a segunda, a API de comunicação serial da SUN, *Javacomm, Java Communications API* (JCA, 2009), foi integrada ao OSGi.

Para a carga do aplicativo embarcado, duas ferramentas foi utilizado o *Code Composer Essentials for MSP430 v2.0.4.0*, uma extensão da IDE Eclipse com *plugins* específicos para suporte à programação C em plataformas MSP430. Ao fornecer uma série de funcionalidades ao programador (depuração de código, navegação, ajuda integrada à web, simulação do programa, etc.) a ferramenta permite o rápido desenvolvimento de aplicações embarcadas.

Para acessar informações do dispositivo emuladas na serial, alguns recursos da API *Javacomm* foram utilizados. No OSGi, o *SerialReaderBundle* apresenta uma interface que oferece funções para a configuração do acesso à serial, além de implementar um protocolo específico para extrair as informações relevantes do *buffer* do STB-HG. O grupo funcional do *Javacomm* incorporado ao *bundle* (*javax.comm.**) dá suporte à: enumeração das portas seriais disponíveis, configuração da conexão, inicialização do buffer de leitura, configuração de eventos sobre a porta serial, entre outras funções.

Descrição do Processo

Além dos casos de uso padrão, o sistema tem como principal função repassar os dados obtidos de um nó da rede de sensores à parte OSGi do STB-HG e desta, por meio dos componentes-ponte Ginga-J–OSGi, para a interface do aplicativo *xlet*. O diagrama de seqüência da Figura 7.10 ilustra os elementos que compõem esse processo. As seguintes etapas descrevem a configuração dos grupos de componentes envolvidos no processo.

Passo 1: Configuração do aplicativo embarcado: nessa etapa, no *Code Conposer* é criado o aplicativo em C que executará no nó sensor principal e nos periféricos. São configuradas opções de acesso ao microcontrolador e ao módulo de sensor de temperatura, além de se definir o protocolo através do qual as informações serão repassadas ao STB-HG. O aplicativo embarcado é uma versão do código original disponível com o kit MSP430 EZ430-RF2500 customizada para o sistema.

Passo 2: Definição dos bundles/serviços OSGi e integração com Javacomm: nessa etapa, o *SerialReaderBundle* fornece um serviço (*SerialReaderService*) que integra as funções do *Javacomm* ao ambiente XleTView-Knopflerfish. A interface

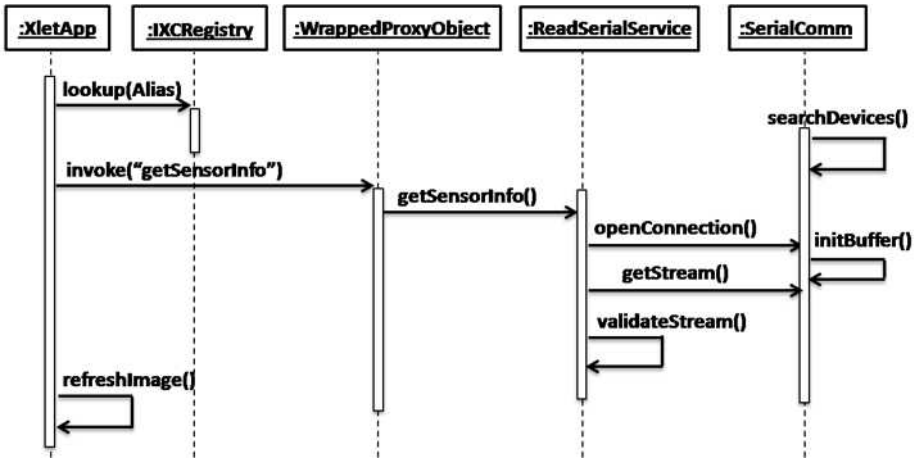


Figura 7.10: Diagrama de seqüência para o esquema geral do cenário de monitoramento de temperatura.

FONTE: Do Autor.

utiliza do protocolo desenvolvido, segundo o mesmo padrão de organização dos dados estabelecido no código C, e 'publica' métodos no OSGi para acesso aos dados.

Passo 3: Exportação do serviço de temperatura ao Ginga-J: uma vez disponíveis o *bundle* e o serviço, este último (*SerialReaderService*) é exportado ao domínio Ginga-J utilizando o processo já conhecido. Assim, o *WrappedProxyObject* associado ao *SerialReaderService* fornece aos aplicativos *xlets* meios de acessar as informações do sensor de temperatura.

Passo 4: Configuração da interface do aplicativo xlet: na última etapa, o aplicativo *xlet*, ao buscar o serviço no *IXC Registry*, configura usando componentes HAVi e AWT, a interface com um gráfico mostrando informações temporais sobre os valores de temperatura obtidos dos nós da rede de sensores. A interface permite ainda ao usuário monitorar cada sensor individualmente e alternar a escala de visualização de temperatura (graus Celsius ou Fahrenheit).

Conforme no diagrama de seqüência para este cenário (Figura 7.10), a interação começa quando o aplicativo *xlet* aciona os componentes ponte, por meio da invocação no *IXCRegistry* de um *WrappedProxyObejct*, Este acessa o serviço OSGi (*ReadSerialService*) que possui funções para inicializar a conexão com a serial usando o objeto *SerialComm* da API *Javacomm*. Uma vez obtidos os dados do sensor, o *xlet* os atualiza na tela.

Resultados

Como resultado tem-se o sistema de monitoramento de temperatura centrado na TVDI, ilustrado em execução na Figura 7.11.

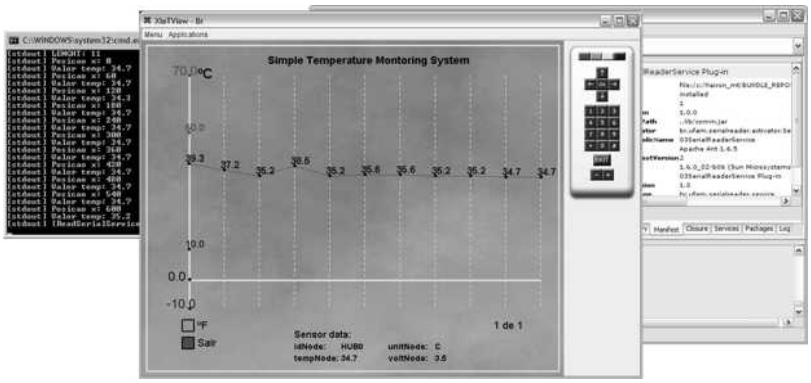


Figura 7.11: Cenário de monitoramento de temperatura. Tela do sistema.
FONTE: Do Autor.

7.2 Cenário Ginga-NCL–OSGi

Nessa seção é apresentado o cenário construído sobre o ambiente Ginga-NCL–OSGi, ferramentas de hardware e software utilizadas, bem como o processo de interação entre os componentes e os resultados obtidos.

7.2.1 Sincronização de vídeo com impressão de documento

Visão Geral

O primeiro cenário declarativo utiliza recursos do Ginga-NCL para sincronizar um aplicativo de impressão de documento com a apresentação da mídia de vídeo. Essa situação é ilustrada no primeiro cenário de exemplo do Capítulo 1. A principal característica do cenário é a possibilidade de definir o exato momento em que o aplicativo será exibido (quando, por exemplo, no vídeo em exibição um apresentador sugere que o usuário imprima o documento).

A Figura 7.12 ilustra o cenário. Na plataforma STB-HG, um *bundle* instalado é responsável por gerenciar o serviço de impressão. Do ponto de vista do aplicativo Ginga-

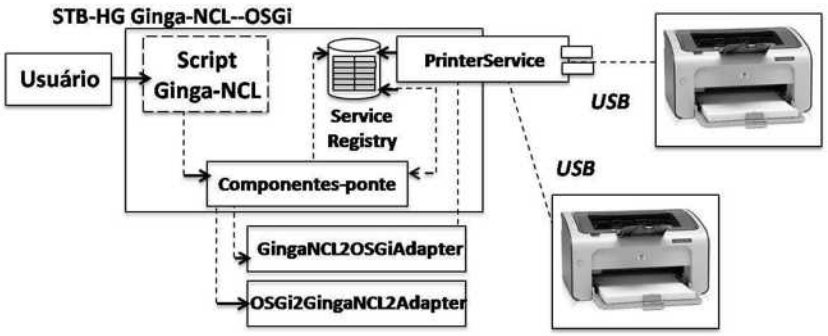


Figura 7.12: Cenário de impressão no Ginga-NCL. Visão Geral.
FONTE: Do Autor.

NCL, ao processar o *script*, a máquina de apresentação irá localizar o *Adapter* responsável pela busca do serviço. Ao encontrar o serviço, o *Adapter* oferece funções para o usuário em cada item de menu da aplicação (configurar impressão, imprimir documento, selecionar impressora, etc.). Uma vez selecionado um comando de menu, a função correspondente ao serviço OSGi será ativada. Caso algum erro ocorra durante o processo, o serviço OSGi notifica o usuário pela tela da TVDI. Enquanto a mídia do *Adapter* está sendo exibida o usuário pode interagir com a aplicação.

Recursos de Hardware

Como recursos de hardware foram utilizados: o computador STB-HG com o Ginga-NCL e uma impressora do tipo HP LaserJet 1100.

Recursos de Software

Recursos da API padrão Java para impressão (do grupo funcional *javax.printer.**) foram integrados ao ambiente para garantir o acesso às funções de impressão como configuração do *buffer*, definição do tipo de documento, procura por impressoras disponíveis, modificação das propriedades da impressora, entre outras. Essas funções foram agrupadas no serviço OSGi do *bundle BundlePrinter*.

Descrição do Processo

O processo que envolve o cenário é dividido em três etapas: primeiro a representação das principais funções em casos de uso; em seguida a identificação/descrição do caso de uso principal em etapas; por fim a interação entre os componentes do sistema é modelada num diagrama de seqüência.

A Figura 7.13 ilustra os casos de uso do sistema. Os dois primeiros (*inicializar contexto* e *inicializar adapters*) referem-se à inicialização do contexto do *BundleContextExporter* na ponte *ContextBridge* e à ativação dos *Adapters* Ginga-NCL, respectivamente. O segundo representa a solicitação de impressão feita pelo usuário, que ativa mais três casos de uso: *buscar serviço de impressão*, onde será recuperado o serviço no OSGi, *acessar dispositivo*, no qual serão usadas as funções da API Java de impressão e *responder à solicitação*, onde será enviada a mensagem de retorno à tela da TVDI.

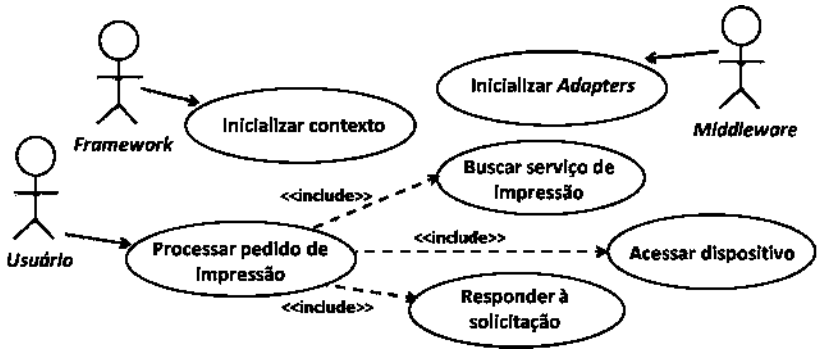


Figura 7.13: Casos de uso para o cenário de impressão.
FONTE: Do Autor.

O caso de uso principal (*buscar serviço de impressão*) é executado configurando-se as seguintes etapas:

Passo 1: configuração dos bundles e do serviço: nessa etapa são criados o *bundle* e o serviço de impressão (*BundlePrinter* e *BundlePrinterService*, respectivamente). O *BundlePrinterService* é uma interface com métodos que acessam diretamente os componentes da API Java de impressão. O *BundlePrinter* habilitará o *Adapter* Ginga-NCL (*GingaNCL2OSGiAdapter*) acessar o serviço do OSGi de acordo com o processo descrito na seção 6.2.2. Para o esquema inverso, ou seja, para garantir que a resposta à solicitação será mostrada na tela da TVDI, será configurado um objeto

OSGi2GingaNCLAdapter e repassado ao *OSGi2GingaNCLBundle* também conforme em 6.2.2.

Passo 2: utilização dos componentes ponte para acessar o serviço de impressão: nessa etapa os componentes *GingaNCL2OSGiAdapter* e *GingaNCL2OSGiPlayer* são associados ao serviço *BundlePrinterService*. No momento em que o *GingaNCL2OSGiAdapter* for invocado na aplicação ele buscará o serviço de *BundlePrinter* e executará a ação correspondente à opção de menu da tela selecionada pelo usuário. Para exibir a resposta à solicitação na tela da TVDI, o mesmo *bundle* deve utilizar o serviço do *OSGi2GingaNCLBundle* para encontrar o *OSGi2GingaNCLAdapter* que gerencia a tela e repassar o texto a ser mostrado.

Passo 3: definição da sincronização da aplicação com a mídia, no arquivo NCL: na etapa final é configurada a interface da aplicação no arquivo NCL. O menu do sistema possui duas opções (configurar impressão e imprimir), todas relacionadas ao serviço *BundlePrinterService* gerenciado pelo *GingaNCL2OSGiAdapter*. Para receber a resposta à requisição, no NCL é configurada a área da região onde será exibido o texto gerenciado pelo *OSGi2GingaNCLAdapter* e *OSGi2GingaNCLBundle*. Os dois adapters são referenciados no NCL com os identificadores padrão *application/x-ginga-OSGiadapter* e *application/x-osgi-gingaadapter* conforme na seção 6.2.2.

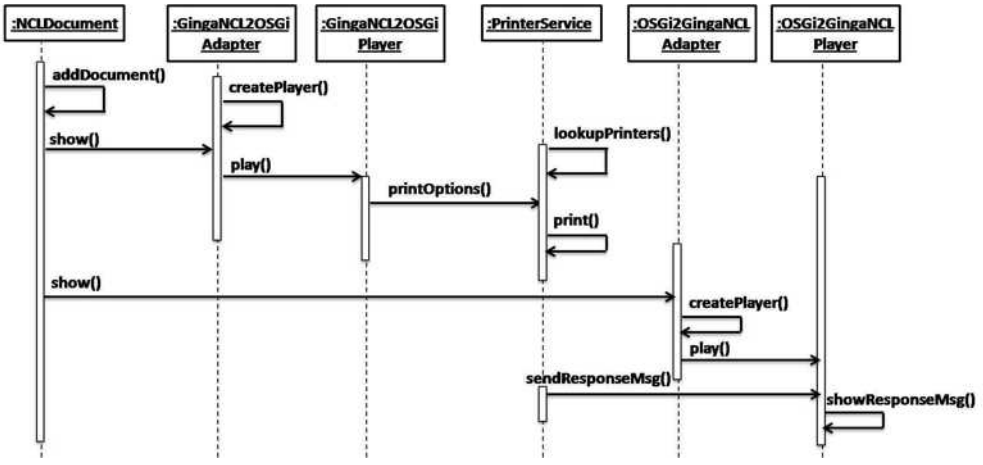


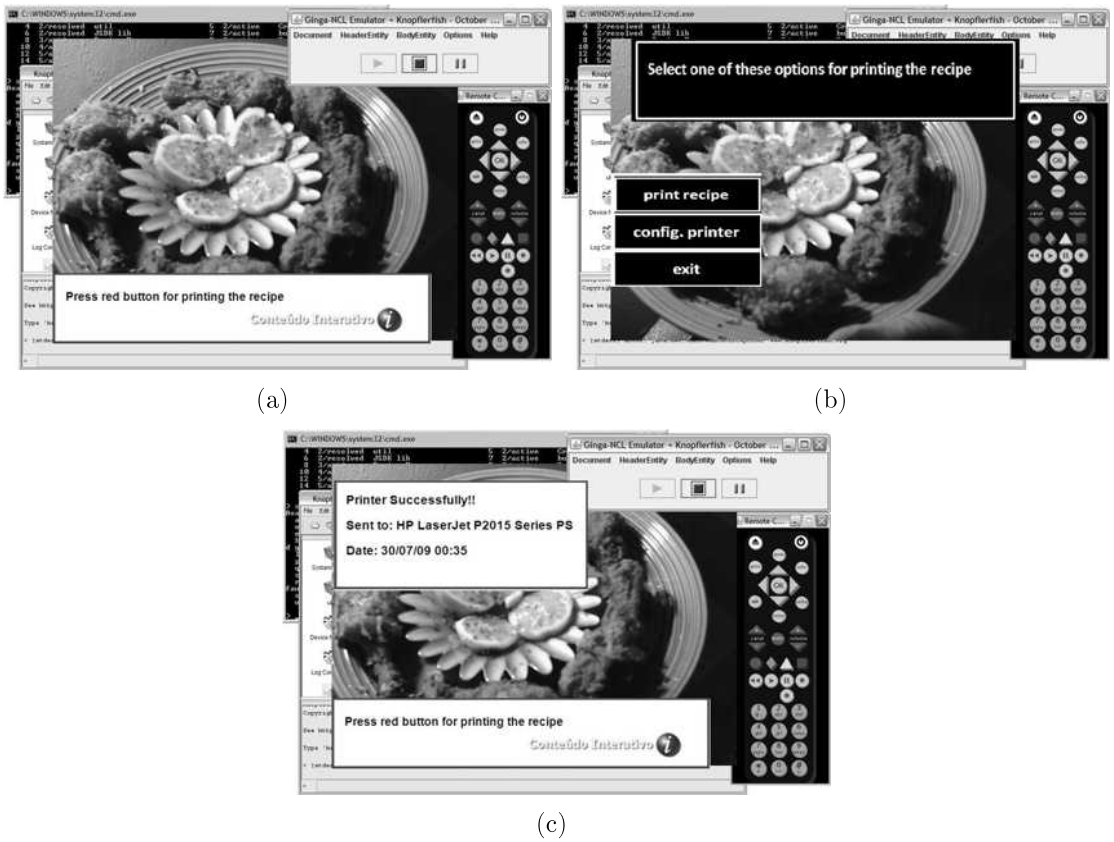
Figura 7.14: Diagrama de seqüência para o caso de uso principal do cenário de impressão.
FONTE: Do Autor.

O diagrama de seqüencia da Figura 7.14 ilustra a interação entre os componentes configuradas nas três etapas descritas. A seqüencia se inicia quando um objeto associado

ao arquivo NCL (*NCLDocument*) ativa um *GingaNCL2OSGiAdapter* que por sua vez solicita ao *GingaNCL2OSGiPlayer* que invoque o serviço OSGi. Uma vez invocado, o serviço acessa componentes da API de impressão e obtém um retorno relativo à solicitação (uma mensagem de status). Esse retorno é repassado ao *OSGi2GingaNCLPlayer* e deste para o *OSGi2GingaNCLApdater* que atualiza o status na tela e retorna a mensagem ao documento NCL.

Resultados

A figura 7.15 ilustra algumas telas do sistema para as respectivas funções: interface com o *Adapter* exibindo a opção de interatividade (Figura 7.15(a)); interface inicial do sistema (Figura 7.15(b)); e confirmação da impressão (Figura 7.15(c)).



(a) (b)

(c)

Figura 7.15: Cenário de impressão no Ginga-NCL. Telas do sistema.

FONTE: Do Autor.

7.3 Resumo

Neste capítulo foram apresentados os cenários construídos sobre a plataforma Ginga-OSGi. Os cenários servem como meios de teste das funcionalidades dos componentes, na medida em que utilizam dos mecanismos de integração do middleware OSGi com o Ginga (tanto a parte Ginga-J como a Ginga-NCL) discutida nos capítulos anteriores. Cada cenário é analisado como um processo, onde são identificadas as funções principais e uma vez modeladas, são implementadas nos componentes da plataforma. O processo começa com uma descrição geral, recursos de hardware e software utilizados e termina com as etapas de configuração dos componentes para o cenário.

Para teste da plataforma Ginga-J–OSGi escolheu-se três cenários: monitoramento de câmeras, que integra a plataforma com funções da JMF; simulação de controle de acesso, que usa funções de acesso à rede TCP/IP para estabelecer a comunicação do STB-HG com um dispositivo embarcado (Rabbit RCM 3200); e monitoramento de temperatura, que centraliza os dados obtidos de uma rede WiFi de dispositivos (Texas MSP430) no STB-HG. Para teste da plataforma Ginga-NCL–OSGi, escolheu-se o cenário de impressão sincronizada com o documento NCL, onde foi verificada a participação dos *Adapters* que constituem os dois mecanismos de interação (de Ginga-NCL para OSGi e de OSGi para Ginga-NCL).

Com este capítulo encerra-se a discussão sobre a implementação dos componentes Ginga-OSGi, abordada desde o Capítulo 5. No próximo capítulo serão levantadas algumas conclusões acerca da solução proposta, algumas dificuldades encontradas ao longo da pesquisa e sugestões para futuras extensões da plataforma.

Capítulo 8 - Considerações Finais

Este capítulo apresenta algumas considerações finais acerca do que foi desenvolvido no trabalho desta Dissertação, tanto relativo ao resultado em si (plataforma integrada Ginga-OSGi) como ao processo que envolveu o estudo sobre o tema, coleta de referências, estado da arte, testes, etc. Como resultados são enumeradas as vantagens que a plataforma possui com relação às propostas dos trabalhos relacionados, destacando principalmente o uso de componentes da especificação Ginga. Será feita uma análise dos objetivos definidos na introdução. Também serão analisadas a experiência proporcionada pelo estudo de tecnologias recentes como TV Digital e *Home Network*, as dificuldades na busca por referências ou especificações abertas na literatura (principalmente relativas à TV Digital brasileira), bem como limitações, problemas e pontos de melhoria sobre a plataforma e os cenários construídos. Por fim, serão apresentadas sugestões para a realização de trabalhos futuros, seguido do comentário final sobre o tema desta Dissertação.

8.1 Contribuições

O escopo de contribuições deste trabalho pode ser avaliado em cada um dos capítulos da Dissertação. A organização dos capítulos reflete o objetivo principal de apresentar uma solução de integração STB-HG baseada na TV Digital brasileira e no *framework* OSGi. Para chegar a este resultado, a pesquisa seguiu um processo que partiu de uma análise geral dos dois grandes temas (TVDI e *Home Network*), até apresentar o mecanismo de interação proposto.

Considerando os capítulos de conceitos iniciais, a visão geral de tecnologias no Capítulo 2 oferece uma importante base de conhecimento, com referências sobre estado da

arte em TV Digital e *Home Networks*. Já o Capítulo 3 oferece uma análise das plataformas de software para STBs e *Home Gateways*, permitindo identificar arquitetura e modelos de *middlewares* (MHP/GEM e Ginga) e *frameworks* (HAVi, OSGi, Jini), expondo suas potencialidades e limitações, e apontando quais os ambientes mais adequados para uma integração STB-HG na TVDI brasileira.

O Capítulo 4 apresenta o estado da arte das pesquisas alinhadas com esta Dissertação, sendo analisados os principais modelos de colaboração entre *middleware* de TVDI e *framework* de *Home Network* que existem na literatura. Ao analisar os mecanismos de interação STB-HG dos trabalhos relacionados, o capítulo permite a identificação de características que podem vir a ser usadas em um modelo STB-HG baseado em Ginga.

Os capítulos 5, 6 e 7 constituem o núcleo da contribuição do trabalho. Os dois primeiros apresentam modelo e implementação enquanto que o último analisa a utilização dos componentes em cenários de uso. Nesses três capítulos procura-se alcançar os principais objetivos definidos no Capítulo 1 para a plataforma.

O objetivo principal, a plataforma de integração que permite exportação de serviços e funções entre ambientes de software de TV Digital e *Home Network*, foi alcançado. Apesar de a estratégia ter dividido o mecanismo nas abordagens procedural (Ginga-J–OSGi) e declarativa (Ginga-NCL–OSGi), em cada uma delas operam componentes que promovem o compartilhamento de serviços.

De uma forma geral, a plataforma STB-HG desta Dissertação apresenta ainda outros pontos positivos, como o fato de usar as duas plataformas de software num mesmo ambiente, o que permite uma comunicação direta entre os componentes. Isso é uma vantagem considerável em relação aos trabalhos que utilizam a rede TCP/IP como meio de comunicação.

Outra característica importante é o fato de os componentes criados encapsularem as principais funções de acesso ao Ginga e ao OSGi, criando uma estrutura ponte que exporta objetos de um ambiente para o outro sem modificar o modelo básico das duas plataformas. Isso exclui do programador de aplicações (*xlets* Ginga-J, ou *bundles* OSGi) a necessidade de conhecer os componentes do núcleo da plataforma, à medida que ele pode

trabalhar com componentes Ginga-J (*IXC Regsitry*) e OSGi (*Service Registry*). No caso da plataforma Ginga-NCL-OSGi, o programador de *scripts* NCL precisa apenas conhecer os nomes dos *Adapters* para o serviço OSGi que deseja utilizar.

Dois pontos que também merecem destaque são os mecanismos de melhoria. O gerenciamento automático de dependências e a descoberta de serviços facilitam o uso da rede OSGi por aplicativos de TV Digital. *Xlets* ou *scripts* NCL não "precisam conhecer" os serviços instalados na plataforma, pois são obtidos dinamicamente.

Com relação ao modelo baseado no Ginga-NCL, a possibilidade de sincronizar a apresentação de serviços de dispositivos em *Home Networks* com outras mídias como áudio e vídeo, é um ponto importante. Uma das principais características do *middleware* declarativo é a fácil sincronização entre mídias. Ao definir um serviço OSGi como uma mídia do Ginga-NCL (através do *Adapter* correspondente) a aplicação interativa pode ser diretamente sincronizada com o dispositivo eletrônico associado ao serviço.

No caso do modelo baseado no Ginga-J, um fator que merece destaque é a natureza dos componentes criados. Pela análise feita no Capítulo 6 o processo de integração Ginga-J–OSGi é o menos complexo em relação aos mecanismos dos trabalhos relacionados. Além disso, há ainda três vantagens: o componente *WrappedProxyObject* é menos complexo que o *XBundLET* de Cabrer et al. (2006) ou Redondo et al. (2007); não há a necessidade de alocar componentes ponte fora do ambiente GingaJ-OSGi, como em Yang et al. (2007); e por fim, o *Application Manager* do Ginga-J permanece no seu domínio, não sendo exportado para o OSGi, como ocorre em Lin et al. (2008).

8.2 Experiência

O desenvolvimento de aplicações para TV Digital ainda é uma área nova no Brasil. Da mesma forma, serviços e aplicações em *Home Network* ainda são pouco estudados e a maioria dos trabalhos encontrados sobre o assunto são internacionais (ver levantamento de referências do Capítulo 4). Cabe às universidades e aos centros de pesquisa oferecerem à comunidade a base para a inserção nesta nova área do conhecimento. Ao investigar essas novas áreas este trabalho enfrenta um grande desafio: fornecer à comunidade uma

visão do atual estado da arte ao mesmo tempo em que enfrenta as dificuldades em coletar referências adequadas e confiáveis, buscar especificações abertas, compreender novos modelos e arquiteturas, e outros entraves quando se estuda essas áreas.

Os ambientes de desenvolvimento do projeto desta Dissertação foram as dependências do CETELI-UFAM, principalmente o laboratório de pesquisa e desenvolvimento de aplicações em TVDI, que conta com equipamentos e um acervo de livros sobre a área. Em contrapartida, não existe um laboratório para desenvolvimento ou simulação de *Home Networks*, e grande parte das referências constituem-se de artigos.

Para atingir o objetivo principal, a platforma STB-HG, foi feita uma análise dos recursos de hardware/software disponíveis considerando questões de custos, viabilidade e tempo. De início, procurou-se explorar os recursos oferecidos pelo Set-Top Box do laboratório de TVDI, porém dadas as suas limitações e inflexibilidade quanto à possibilidade de modificar o *middleware* disponível, optou-se por descartá-lo.

A solução seguinte foi a busca por uma plataforma flexível de STB que permitisse ao programador instalar os ambientes comuns Ginga-OSGi. Essa solução foi descartada considerando que por si só já constituia um esforço além do escopo desta Dissertação (modificar *middleware/framework*, projetar uma nova arquitetura, buscar ou criar drivers para acesso às interfaces com dispositivos, etc.). Após isso, optou-se por utilizar as ferramentas de emulação em PC disponíveis.

Sobre a plataforma, embora ela atenda ao objetivo principal da Dissertação, as limitações das ferramentas de emulação e a indisponibilidade de um ambiente completo da implementação do Ginga acabaram por contribuir para a separação da implementação nas duas abordagens estudadas, o que demandou o estudo separado da integração Ginga-J–OSGi e Ginga-NCL–OSGi. A primeira, pelo fato de o *middleware* procedural fornecer um maior número de funcionalidades, constituiu-se de um mecanismo mais complexo que a segunda. Em contrapartida, a integração de aplicações de TVDI com serviços OSGi na plataforma usando *scripts* NCL é mais simples quando comparada ao desenvolvimento de aplicativos *xlet*.

Sobre os cenários, conforme apresentado no Capítulo 7, constituem-se de provas de funcionalidades dos componentes, o que justifica o não detalhamento de questões

específicas como análise de desempenho da rede ou consumo de energia do dispositivo. Os equipamentos utilizados foram disponibilizados pelo laboratório de Automação Industrial da Universidade Federal do Amazonas. Os dispositivos, embora não constituam produtos para uma *Home Network* foram úteis na ilustração de quão usável pode ser a plataforma desta Dissertação.

8.3 Trabalhos Futuros

A plataforma desenvolvida nesta Dissertação abre possibilidades de exploração de novos serviços em TV Digital, à medida que permite a construção de cenários de integração da TVDI com dispositivos eletrônicos. Possíveis trabalhos futuros decorrentes desta pesquisa podem ser de dois tipos: trabalhos que estendem a plataforma para cenários específicos (como os estudos de caso do Capítulo 7) ou trabalhos que modificam a estrutura da plataforma para atender a outros propósitos.

Um trabalho interessante é a utilização de outras especificações para *Home Network*, integradas ao OSGi, como o Jini ou UPnP. Isso pode ser útil em redes heterogêneas, considerando-se que numa *Home Network* comum existem uma série de padrões concorrentes. O trabalho poderia investigar uma forma de integrar *frameworks* OSGi, Jini, UPnP e HAVi de maneira transparente às aplicações de TVDI, de modo que um *xlet* Ginga-J e um *script* Ginga-NCL acessem serviços Jini ou OSGi de uma mesma forma.

Uma proposta que certamente exigirá a modificação dos componentes núcleo da plataforma é a aplicação do modelo STB-HG às plataformas virtuais disponíveis para os ambientes declarativo (Ginga-NCL Virtual STB) (Ginga-NCL, 2009b) e procedural(OpenGinga) (OpenGinga, 2009). Isso exigirá a integração do OSGi com plataformas não-Java como, por exemplo, a implementação C++ do Ginga-NCL Virtual STB.

Outro trabalho de extensão do sistema que pode ser seguido é o projeto de uma arquitetura real STB-HG para a execução da plataforma Ginga-OSGi. Para isso serão necessários tanto recursos de hardware como interfaces do STB (moduladores,

decodificadores, demultiplexadores, *smart card*, etc) e HG (*Bluetooth*, *WiFi*, *GPRS*, *Zigbee*, etc), além da instalação de Sistema Operacional compatível e JVM. Com toda essa infra-estrutura montada a plataforma poderia ser avaliada num ambiente real.

Um estudo detalhado focado na construção de um cenário específico sobre a plataforma também poderia ser realizado. Nesse estudo seriam verificados não somente os componentes de software, mas também questões de desempenho, taxas de comunicação (quando da utilização de dispositivos *WiFi*, *Zigbee* ou *Bluetooth*) ou consumo de energia.

Por fim, um desafio de extensão desse trabalho é a utilização dos componentes criados para a construção de uma ferramenta de autoria que permita o desenvolvimento de aplicações de TVDI no ambiente de *Home Network* de forma mais intuitiva para o programador, sem usar código do aplicativo *xlet* ou *scripts* NCL.

8.4 Conclusão Final

A evolução das tecnologias de comunicação e de dispositivos eletrônicos de consumo permitiu a integração e distribuição de serviços em redes desde a internet até às *Home Networks*. A expansão das características da TV Digital e seu recente uso como *Home Gateway*, a torna elemento chave para o provimento desses serviços. Diante disso, este trabalho pretende contribui para a consolidação de um modelo de colaboração em software STB-HG permitindo o surgimento de cenários úteis para a melhoria da qualidade de vida do usuário. O trabalho investiga modelos já existentes de convergência entre plataformas de software para TV Digital e *Home Network* e propõe um novo modelo adequado às características do middleware brasileiro de TVDI. Além do modelo baseado na especificação procedural, a novidade do trabalho apresentado encontra-se na exploração das funcionalidades do middleware declarativo brasileiro (Ginga-NCL), provendo um mecanismo de gerenciamento de serviços OSGi pelo Ginga-NCL. Isso estende consideravelmente o escopo de cenários de convergência que podem ser verificados.

Espera-se que o trabalho venha a contribuir na literatura sobre desenvolvimento de aplicações para a TV Digital brasileira e novos serviços, com sua possível utilização em trabalhos de graduação, artigos e dissertações.

Referências Bibliográficas

6POWER. *Identification of IPv6-Enabled Devices to be Used in Home Automation. IST-2001-37613 Technical Report. 6Power Information Society Technologies. Deliverable D4.8.* Disponível em: <www.6power.org/open/6power_pu_d4_8_v2_5.pdf>. Acesso em: Agosto de 2009.

ABNT. *Associação Brasileira de Normas Técnicas. Norma Brasileira NBR 15606-1. Televisão Digital Terrestre - Codificação de Dados e Especificações de Transmissão para Radiodifusão Digital. Parte 1: Codificação de Dados.* Disponível em: <http://www.dtv.org.br/download/pt-br/ABNTNBR15601_2007Vc_2008.pdf>. Acesso em: Agosto de 2009.

ABNT. *Associação Brasileira de Normas Técnicas. Norma Brasileira NBR 15606-2. Televisão Digital Terrestre - Codificação de Dados e Especificações de Transmissão para Radiodifusão Digital. Parte 2: Ginga-NCL para Receptores Fixos e Móveis - Linguagem XML para Codificação de Aplicações.* Disponível em: <http://www.dtv.org.br/download/pt-br/ABNTNBR15602_D3_2007Vc_2008.pdf>. Acesso em: Agosto de 2009.

ACM. *ACM Digital Library.* Disponível em: <http://portal.acm.org>. Acesso em: Agosto de 2009.

ALAM, M. ; PRASAD, N. R. . Convergence Transforms Digital Home: Techno-Economic Impact. In: *Wireless Personal Communications Journal*, v. 44, n. 1, p. 75–93, 2008. ISSN 0929-6212. doi: http://dx.doi.org/10.1007/s11277-007-9380-2.

APACHE. *Apache Ant Welcome Page*. Disponível em: <http://ant.apache.org/>. Acesso em: Agosto de 2009.

ATSC. *Advanced Television Systems Committee*. Disponível em: <http://www.atsc.org>. Acesso em: Agosto de 2009.

BAE, Yu-Seok ; OH, Bong-Jin ; MOON, Kyeong-Deok ; KIM, Sang-Wook . Architecture for Interoperability of Services Between an ACAP Receiver and Home Networked Devices. In: *IEEE Transactions on Consumer Electronics*, v. 52, n. 1, p. 123–128, Fev. 2006. ISSN 0098-3063. doi: 10.1109/TCE.2006.1605036.

BARBOSA, S. D. J ; SOARES, L. F. G. . TV Digital Interativa no Brasil se faz com Ginga: Fundamentos, Padrões, Autoria Declarativa e Usabilidade. In: *Jornada de Atualização em Informática - JAI 2008, Anais, Belém-PA*, v. 1, n. 1, p. 105–174, 2008.

BINDER. *Service Binder: Simplifying application development on the OSGi services platform*. Disponível em: <http://gravity.sourceforge.net/servicebinder/>. Acesso em: Agosto de 2009.

BISHAJ, B. . *Comparison of Service Discovery Protocols.* pages 1–7. TKK T-110.51990 Technical Report. Helsinki University of Technology. Telecommunications Software and Multimedia Laboratory. Seminar on Interworking. Disponível em: <www.tml.tkk.fi/Publications/C/23/papers/Bishaj_final.pdf>. Acesso em: Agosto de 2009.

BOUCOUVALAS, A.C. . Future evolution of network technologies. In: *Proceedings of the 24th International Conference on Information Technology Interfaces, 2002. ITI 2002.*, v. 1, p. 3–6, 2002. ISSN 1330-1012. doi: 10.1109/ITI.2002.1024644.

BRASIL. Decreto-lei n. 4.901, de 26 de Novembro de 2003. *Institui o Sistema Brasileiro de TV Digital e dá outras providências*. Diário Oficial da República Federativa do Brasil. Brasília, 27/11/03, Seção 1, Pag. 07.

BURGER, R. A. ; IACOVANNI, G. ; READER, C. ; XIAMING, F. ; XIAODONG, Y. ; WANG, H. . A Survey of Digital TV Standards China. In: *Second International*

Conference on Communications and Networking in China, CHINACOM'07, v. 1, n. 1, p. 687–696, 2007. ISSN 0018-9219. doi: 10.1109/JPROC.2005.861000.

CABLELABS. *Cablelabs. Cable Technologies.* Disponível em: <http://www.cablelabs.com>. Acesso em: Agosto de 2009.

CABRER, M. R. ; REDONDO, R. P. D. ; VILAS, A. F. ; ARIAS, J. J. P. ; DUQUE, J. G. . Controlling the Smart Home from TV. In: *Consumer Electronics, IEEE Transactions on*, v. 52, n. 2, p. 421–429, 2006. ISSN 0098-3063. doi: 10.1109/TCE.2006.1649659.

CALDER, B. ; COURTNEY, J. ; FOOTE, B. ; KYRNITSZKE, L. ; RIVAS, D. ; SAITO, C. ; VANLOO, J. ; YE, T. . *JavaTV API Technical Overview: The JavaTV API Whitepaper.* Sun Microsystems. Version 1.0. Disponível em: <http://java.sun.com/javame/technology/javatv/docs/jtv-1_0-spec_overview.pdf>. Acesso em: Agosto de 2009.

CEBUS. *Consumer Electronics Bus.* Disponível em: <http://www.cebus.org>. Acesso em: Agosto de 2009.

CERVANTES, H. ; HALL, R. S. . Automating service dependency management in a service-oriented component model. In *6th ICSE Workshop on Component-Based Software Engineering.* Disponível em: <http://www.humbertocervantes.net/papers/CBSE2003.pdf>. Acesso em: Agosto de 2009.

CITESEER. *Scientific Literature Digital Library.* Disponível em: <http://citeseer.ist.psu.edu>. Acesso em: Agosto de 2009.

CRINON, R. J. ; BHAT, D. ; CATAPANO, D. ; THOMAS, G. ; VAN LOO, J. T. ; BANG, G. . Data Broadcasting and Interactive Television. In: *Proceedings of the IEEE*, v. 94, n. 1, p. 102–118, Jan. 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2005.861020.

DAVIC. *DAVIC: Digital Audio Visual Council.* Disponível em: <http://www.davic.org>. Acesso em: Agosto de 2009.

DECINA, M. ; TRECORDI, V. . Convergence of Telecommunications and Computing to Networking Models For Integrated Services and Applications. In: *Proceedings of the IEEE*, v. 85, n. 12, p. 1887–1914, Dez 1997. ISSN 0018-9219. doi: 10.1109/5.650174.

HARTOG, F.T.H. den; BALM, M. ; JONG, C.M. de; KWAAITAAI, J.J.B. . Convergence of Residential Gateway Technology. In: *IEEE Communications Magazine*, v. 42, n. 5, p. 138–143, Mai 2004. ISSN 0163-6804. doi: 10.1109/MCOM.2004.1299358.

DIXIT, S. ; PRASAD, R. . *Technologies for Home Networking*. New York: John Wiley & Sons, 2008. 218 p.

DLNA. *Digital Living Network Alliance*. Disponível em: <http://www.dlna.org>. Acesso em: Agosto de 2009.

DVB. *Digital Video Broadcasting*. Disponível em: <http://www.dvb.org>. Acesso em: Agosto de 2009.

ECLIPSE. *Eclipse.org Home*. Disponível em: <http://www.eclipse.org/>. Acesso em: Agosto de 2009.

EMERALD. *Emerald Group Publishing Limited*. Disponível em: <http://www.emeraldinsight.com>. Acesso em: Agosto de 2009.

ETSI. *ETSI TS 102 819 v1.3.1 (2005-10). Digital Video Broadcasting (DVB) Globally Executable MHP version 1.0.2 (GEM 1.0.2) Technical Specification. European Broadcast Union (EBU)*. Disponível em: <www.mhp.org/specs/ts_102819v010301p.pdf>. Acesso em: Agosto de 2009.

ETSI. *European Telecommunication Standards Institute*. Disponível em: <http://ww.etsi.org>. Acesso em: Agosto de 2009.

FCMA. *Fixed-Mobile Convergence Alliance*. Disponível em: <http://www.thefmca.com>. Acesso em: Agosto de 2009.

FERNANDES, J. ; LEMOS, G. ; SILVEIRA, G. . Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. In: *Jornada de Atualização em Informática - JAI 2004, Anais, Salvador-BA*, v. 1, p. 1–56, 2004.

FRIDAY, A. ; DAVIES, N. ; WALLBANK, N. ; CATTERALL, E. ; PINK, S. . Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments. In: *Wireless Network Journal*, v. 10, n. 6, p. 631–641, 2004. ISSN 1022-0038. doi: http://dx.doi.org/10.1023/B:WINE.0000044024.54833.cb.

GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J. . *Design patterns: elements of reusable object-oriented software.* New Jersey: Addison-Wesley, 1995. 416 p.

GE, Chengchen ; LI, Ying ; ZHI, Xiaoli ; TONG, Weiqin . The Intelligent STB - Implementation of Next Generation of Residential Gateway in Digital Home. In: *2nd International Conference on Pervasive Computing and Applications, 2007. ICPCA 2007.*, v. 1, p. 256–261, Jul 2007. doi: 10.1109/ICPCA.2007.4365450.

GERASIMENKO, S. . *The Evolution of Wireless Home Networking.* pages 1–7. TKK T-110.51990 Technical Report. Helsinki University of Technology. Telecommunications Software and Multimedia Laboratory. Seminar on Interworking. Disponível em: <www.tml.tkk.fi/Publications/C/23/papers/Gerasimenko_final.pdf>. Acesso em: Agosto de 2009.

GINGA-NCL. *Ginga-NCL. Declarative DTV Middleware. Ginga-NCL Emulator v1.1.1.* Disponível em: <http://www.gingancl.org.br/ferramentas.html>. Acesso em: Agosto de 2009.

GINGA-NCL. *Ginga-NCL Virtual Set-Top Box v. 0.10.1.* Disponível em: <http://www.ncl.org.br/ferramentas/fedora-fc7-ginga-i386.zip>. Acesso em: Agosto de 2009.

GONG, L. . A Software Architecture for Open Service Gateways. In: *Internet Computing, IEEE*, v. 5, n. 1, p. 64–70, 2001a. ISSN 1089-7801. doi: 10.1109/4236.895144.

GONG, L. . *Programming Open Service Gateway with Java Embedded Server Technology*. New Jersey: Addison-Wesley, 2001b. 456 p.

GUPTA, Satish . *Home Gateway. Technical Whitepaper*. In Wipro Technologies. Disponível em: <http://www.wipro.com/webforms/>. Acesso em: Agosto de 2009.

HAVI. *HAVi: The Home Audio Vídeo Interoperability*. Disponível em: <http://www.havi.org>. Acesso em: Agosto de 2009.

HOANG, L. N. . *Middlewares for Home Monitoring and Control*. pages 1–7. TKK T-110.51990 Technical Report. Helsinki University of Technology. Telecommunications Software and Multimedia Laboratory. Seminar on Interworking. Disponível em: <www.tml.tkk.fi/Publications/C/23/papers/NguyenHoang_final.pdf>. Acesso em: Agosto de 2009.

HOMEPNA. *Home Phoneline Network Alliance*. Disponível em: <http://www.homepna.org>. Acesso em: Agosto de 2009.

HOWES, T. . *The String Representation of LDAP Search Filters*. RFC Editor. Disponível em: <http://www.ietf.org/rfc/rfc2254.txt>. Acesso em: Agosto de 2009, Estados Unidos.

IEEE. IEEE Standard for a High Performance Serial Bus. In: *IEEE Std 1394-1995. Microprocessor and Microcomputer Standards Committee of the IEEE Computer Society*, v. 1, p. 1–392, Ago 1996.

IEEE. *IEEE Xplore: Guest Home Page*. Disponível em: <http://ieeexplore.ieee.org>. Acesso em: Agosto de 2009.

ISDB. *Integrated Services Digital Broadcasting*. Disponível em: <http://www.dibeg.org>. Acesso em: Agosto de 2009.

ISDTV. *International Standard for Digital Television. Fórum do Sistema Brasileiro de Televisão Digital (SBTVD)*. Disponível em: <http://www.forumsbtvd.org.br>. Acesso em: Agosto de 2009.

ISO. Iso. international organization for standardization. *ISO/IEC 13818-1:2000. Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Systems.* Disponível em: <http://www.iso.org>. Acesso em: Agosto de 2009.

ISO. Iso. international organization for standardization. *ISO/IEC 13818-6:1998/Cor2:2002. Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Extensions for DSMCC.* Disponível em: <http://www.iso.org>. Acesso em: Agosto de 2009.

ISSARNY, V. ; CAPORUSCIO, M. ; GEORGANTAS, N. . A Perspective on the Future of Middleware-based Software Engineering. In *Future of Software Engineering, 2007. FOSE '07*, pages 244–258, 2007.

ITU. *International Telecommunication Union.* Disponível em: <http://www.itu.int>. Acesso em: Agosto de 2009.

JCA. *Sun Core Java. Java Communications API 3.0.* Disponível em: <http://java.sun.com/products/javacomm/>. Acesso em: Agosto de 2009.

JINI. *Sun Microsystems. Jini Architecture Specification, version 1.2.* Disponível em <http://www.sun.com/software/jini/specs/jini1.2html/jini-title.html>. Acesso em: Agosto de 2009.

JINI. *The Jini Network Technology. Sun Microsystems.* Disponível em: <http://www.sun.com/software/jini/>. Acesso em: Agosto de 2009.

JMF. *Java SE Desktop Technologies - Java Media Framework (JMF).* Disponível em: <http://java.sun.com/javase/technologies/desktop/media/jmf/>. Acesso em: Agosto de 2009.

KIM, G. W. ; KIM, Doo-Woo ; LEE, Jun-Ho . Home Network-Specific RBAC Model. In *Proceedings of World Academy of Science, Engineering and Technology*, pages 232–234, 2005.

KNOPFLERFISH. *Knopflerfish project.* Disponível em: <http://www.knopflerfish.org>. Acesso em: Agosto de 2009.

LAGALLY, Michael ; PATZOLD, Jens . *New TV Standard for Digital TV in Brazil.* Apresentação no Java Mobile, Media & eMbedded Developer Days. Sun Microsystems. Disponível em: <http://download.java.net/mobileembedded/developerdays/2009/TS-5-v2.pdf>. Acesso em: Agosto de 2009.

LEA, R. ; GIBBS, S. ; DARA-ABRAMS, A. ; EYTCHISON, E. . Networking Home Entertainment Devices with HAVi. In: *IEEE Computer*, v. 33, n. 9, p. 35–43, Set 2000. ISSN 0018-9162. doi: 10.1109/2.868695.

LIN, Cheng-Liang ; WANG, Pang-Chieh ; HOU, Ting-Wei . A Wrapper and Broker Model for Collaboration Between a Set-Top Box and Home Service Gateway. In: *Consumer Electronics, IEEE Transactions on*, v. 54, n. 3, p. 1123–1129, Ago 2008. ISSN 0098-3063. doi: 10.1109/TCE.2008.4637597.

LONWORKS. *Echelon Corporation. The Next Generation of Smart Networks.* Disponível em: <http://www.echelon.com/products/lonworks/default.htm>. Acesso em: Agosto de 2009.

MARPLES, D. ; KRIENS, P. . The Open Services Gateway Initiative: an Introductory Overview. In: *Communications Magazine, IEEE*, v. 39, n. 12, p. 110–114, Dez 2001. ISSN 0163-6804. doi: 10.1109/35.968820.

MATSUBARA, F.M. ; MIURA, S. ; IMAI, S. ; AKATSU, S. . DTV Architecture Design for Multimedia Network Environments. In: *Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on*, v. , p. 147–148, 2005. doi: 10.1109/ICCE.2005.1429760.

MONTEIRO, M. S. . *Arquitetura de Referência SBTVD.* Funtel. Sistema Brasileiro de Televisão Digital (SBTVD). Relatório Técnico. Disponível em: http://tiny.cc/bxEol. Acesso em: Agosto de 2009.

MONTEZ, C. ; BECKER, V. . *TV Digital Interativa: Conceitos, Desafios e Perspectivas para o Brasil*. Florianópolis: Editora da UFSC, 2005. 160 p.

MORRIS, S. ; SMITH-CHAIGNEAU, A. . *Interactive TV Standards. A Guide to MHP, OCAP and JavaTV*. Burlington USA: Focal Press, 2004. 611 p.

O'DRISCOLL, G. . *The Essential Guide to Digital Set-top Boxes and Interactive TV*. New Jersey: Prentice Hall PTR, 2000. 295 p.

OPENGINGA. . Disponível em: <http://www.openginga.org>. Acesso em: Agosto de 2009.

OPENMHP. *OpenMHP Emulator*. Disponível em: <http://www.openmhp.org>. Acesso em: Agosto de 2009.

OSCAR. *OSCAR OSGi R3 implementation*. Disponível em: <http://forge.objectweb.org/projects/oscar>. Acesso em: Agosto de 2009.

OSGI. *OSGi Service Platform Core Specification*. release 4, v4.1. Disponível em: <http://www.osgi.org/Download/Release4V41>. Acesso em: Agosto de 2009.

OSGI. Osgi alliance. *About OSGi Service Platform*. Disponível em: <www.osgi.org/wiki/uploads/Links/OSGiTechnicalWhitePaper.pdf>. Acesso em: Agosto de 2009.

OSGI-ALLIANCE. *OSGi Alliance. OSGi: The Dinamyc Module System for Java*. Disponível em: <http://www.osgi.org>. Acesso em: Agosto de 2009.

PICCIONE, C. A. . *Modelo e Implementação de um Serviço de Datacasting para Televisão Digital*. Dissertação de Mestrado em Engenharia Elétrica, Editora Universidade Federal de Santa Catarina (UFSC).

PIESING, J. . The DVB Multimedia Home Platform (MHP) and Related Specifications. In: *Proceedings of the IEEE*, v. 94, n. 1, p. 237–247, Jan. 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2005.860997.

PORTELLI, G. ; MALNATI, G. ; FORNO, F. . HoNey: leveraging the MHP to provide HOme NEtwork interoperability. In: *Pervasive Computing and Communications, International Journal on*, v. 4, n. 1, p. 77–91, 2008. ISSN 1742-7371. doi: 10.1108/17427370810873129.

RABBIT. *Rabbit Product Manual. RabbitCore RCM3200. C Programmable Module with Ethernet. User's Manual.* Disponível em: <http://tinyurl.com/ma4lyb>. Acesso em: Agosto de 2009.

REDONDO, Rebeca P. Díaz ; VILAS, Ana Fernández ; CABRER, Manuel Ramos ; ARIAS, José J. Pazos . Exploiting OSGi Capabilities from MHP Applications. In: *Journal of Virtual Reality and Broadcasting. JVRTB EuroITV 2006 Special Issue*, v. 4, n. 16, , jul 2007. URL urn:nbn:de:0009-6-11100. ISSN 1860-2037.

SCHANTZ, R. E. ; SCHMIDT, D.C. . *Middleware for Distributed Systems: Envolving the Common Structure for Network-centric Applications.* In *Encyclopedia of Software Engineering* (J. Marciniak e G. Telecki, eds). New York: Wiley & Sons, 2002. 15 p.

SCHMIDT, D.C. ; BUSCHMANN, F. . Patterns, Frameworks, and Middleware: Their Synergistic Rgelationships. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 694–704, 2003.

SCHWALB, E. M. . *iTV Handbook Technologies and Standards.* New Jersey: Prentice Hall, 2004. 725 p.

SLING. *SLING MEDIA. Home of The Sling Box.* Disponível em: <http://www.slingmedia.com>. Acesso em: Agosto de 2009.

SOARES, L. F. G. ; BARBOSA, S. D. J. ; NETO, C. S. . *Desenvolvimento de Aplicações Declarativas para TV Digital Interativa.* Minicurso: Anais do XII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia2006), Natal-RN.

SOARES, L. F. G. ; RODRIGUES, R. F. . *Nested Context Language 3.0. Part 8: NCL Digital TV Profiles.* Monografias em Ciência da Computação. Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ).

SOARES, L. F. G. ; RODRIGUES, R. F. ; BARBOSA, S. D. J. . *Manual de Construção de Programas Audiovisuais Interativos Utilizando a NCL 2.3 - Perfil Básico*. Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ).

SOARES, Rodrigues R. F. ; MORENO, M. F. . Ginga-NCL: The Declarative Environment of the Brazilian Digital TV System. In *Journal of the Brazilian Computer Society*, pages 37–46, 2007.

SOUZA FILHO, G. L. ; LEITE, E. C. ; BATISTA, C. E. F . Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. In *Journal of the Brazilian Computer Society*, pages 47–56, 2007.

SUN. *Java ME Technology. The JavaTV API.* Disponível em: <http://java.sun.com/products/javatv>. Acesso em: Agosto de 2009.

SUN. *SUN Java CDC Personal Profile 1.0.* Disponível em: <http://java.sun.com/products/personalprofile>. Acesso em: Agosto de 2009.

TAVARES, Andre L.C. ; VALENTE, Marco Tulio . A Gentle Introduction to OSGi. In: *SIGSOFT Softw. Eng. Notes*, v. 33, n. 5, p. 1–5, 2008. ISSN 0163-5948. doi: http://doi.acm.org/10.1145/1402521.1402526.

TELEBRASIL. *Debates sobre o Decreto que define o Sistema Brasileiro de TV Digital Terrestre*. Disponível em: <http://www.telebrasil.org.br>. Acesso em: Agosto de 2009.

TEXAS. *Texas Instruments. MSP430 Wireless Development Tool. EZ430-RF2500.* Disponível em: <http://tinyurl.com/ny35d6>. Acesso em: Agosto de 2009.

TKACHENKO, D. ; KORNET, N. ; DODSON, A. ; LI, Luyang ; KHANDELWAL, R. . A Framework Supporting Interaction of iDTV Applications and CE Devices in Home Network. In: *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*, v. 1, p. 605–607, Jan. 2005. doi: 10.1109/CCNC.2005.1405249.

TKACHENKO, D. ; KORNET, N. ; KAPLAN, A. . Convergence of iDTV and Home Network Platforms. In: *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, v. 1, p. 624–626, Jan. 2004.

TKACHENKO, D. ; KORNET, N. ; LAGUNOV, A. ; KRAVTSOV, D. ; KURBANOV, A. ; KHANDELWAL, R. ; LI, Luyang . A Possible Extension for iDTV Platform to Support Interactions with Home Appliances. In: *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, v. 1, p. 228–232, Jan. 2006.

UPNP. *UPnP Device Architecture.* Disponível em: <http://www.upnp.org/download/UPnPDA10_20000613.html>. Acesso em: Agosto de 2009.

UPNPFORUM. *Universal Plug and Play Forum.* Disponível em: <http://www.upnp.org>. Acesso em: Agosto de 2009.

WALDO, J. . The Jini Architecture for Network-centric Computing. In: *ACM Communications*, v. 42, n. 7, p. 76–82, 1999. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/306549.306582.

X10. *SmartHome. Home Automation, X10, Remote Control, Lighting, Wireless Security.* Disponível em: <http://www.x10.org>. Acesso em: Agosto de 2009.

XLETVIEW. *An MHP Emulator for viewing Xlets on PC.* sourceforge project. Disponível em: <http://xletview.sourceforge.net>. Acesso em: Agosto de 2009.

YANG, Ming-Chien ; SHENG, Norman ; HUANG, Brandon ; TU, Jethro . Collaboration of Set-Top Box and Residential Gateway Platforms. In: *Consumer Electronics, IEEE Transactions on*, v. 53, n. 3, p. 905–910, 2007. ISSN 0098-3063. doi: 10.1109/TCE.2007.4341564.

ZHANG, D. L. H. ; ZHENG, S. . *Open Service Residential Gateway for Smart Homes.* In *Scientific Literature Digital Library (CiteSeer.IST), 2003.* Disponível em: <http://citeseer.ist.psu.edu/636307.html>. Acesso em: Agosto de 2009.

Apêndice A - Publicações

Abaixo são listadas as publicações que foram resultados da pesquisa desenvolvida neste trabalho de Dissertação, em ordem cronológica de publicação.

A.1 Publicação 1

Lucena Jr, V. F.; SILVA Jr, W. S.; VIANA, N. S.; MAIA, O. B.. ***Proposta de Convergência entre o Middleware de Referência Brasileiro de TV Digital e Plataformas de Redes Residenciais***. In: *Fórum de Oportunidades em Televisão Digital Interativa (TVDI'08)*, 09 Mai 2008, Poços de Caldas - MG, Brasil. TVDI'08: Biblioteca da Pontifícia Universidade Católica de Minas Gerais, 2008. v. único. p. 1-8. ISBN: 85-7669-096-9.

A.2 Publicação 2

VIANA, N. S.; de Lucena Jr., V. F.. ***Middleware Baseado em Modelos de Convergência entre Plataformas de Software para TV Digital e Redes Residenciais***. In: *IV Feira Internacional da Amazônia (IV FIAM)*, 10-13 Nov 2008, Manaus-AM. II Mostra de Trabalhos Técnico-Científicos Apoiados pela SUFRAMA. Editora da Superintendência da Zona Franca de Manaus (SUFRAMA), 2008. v. 1. p. 175-177. Link: http://www.suframa.gov.br/download/fiam/anais_ii_mostra_tecnico_cientifica_vr2.pdf ISBN: 978-85-60602-04-9.

A.3 Publicação 3

VIANA, Nairon S.; MAIA, O. B.; de Lucena Jr, V. F.; Pinto, L. C.. ***A Convergence Proposal Between the Brazilian Middleware for iDTV and Home Network Platforms***. In: *5th IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'09)*, 10-13 Jan 2009, Las Vegas - Nevada, EUA. NIME Workshop. *6th IEEE Consumer Communications Networking Conference (CCNC 2009)*, Las Vegas, Nevada, 2009. DOI: http://dx.doi.org/10.1109/CCNC.2009.4784976.

A.4 Publicação 4

VIANA, N. S.; MAIA, O. B.; de Lucena Jr., V. F.; SILVA Jr, W. S.. ***Infra-estrutura de Convergência entre o Middleware Brasileiro de TVDI e Plataformas de Home Network***. In: *XXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 25-29 Mai 2009, Recife. Anais do XXVII SBRC 2009.

A.5 Publicação 5

VIANA, N. S.; Lucena Jr, V. F.. ***A Software Model Supporting the Management of Home Network Services through the Brazilian iDTV***. In: *7th ACM European Interactive TV Conference, EuroITV2009*, Proceedings of the seventh EuroITV 2009, 3-5 June, Leuven-Belgium. DOI: http://doi.acm.org/10.1145/1542084.1542102. New York, USA: Association for Computing Machine Press, 2009. v. I. p. 101-110. ISBN: 978-1-60558-631-1.

A.6 Publicação 6

MAIA, O. B. ; VIANA, N. S.; Lucena Jr, V. F.. ***Using the iDTV for Managing Services in the Ubiquitous Computing Environment***.

In: *6th IEEE International Conference on Ubiquitous Intelligence and Computing, UIC'09. International Symposium on UbiCom Frontiers - Innovative Research, Systems and Technologies, UFirst'09*, 7-9 Jul 2009, Brisbane, Australia. DOI: http://dx.doi.org/10.1109/UIC-ATC.2009.61. ISBN: 978-0-7695-3737-5. Proceedings of the 6th IEEE UIC'09, 2009.

A.7 Publicação 7

VIANA, N. S.; MAIA, O. B., Lucena Jr, V. F.. ***Modelo de Convergência entre o Middleware Brasileiro de TVDI e Plataformas de Software para Home Networking***. In: *XV Simpósio Brasileiro de Sistemas Multimídia e Web, Webmedia'09*, 05-07 Out, Fortaleza-CE, Brasil (A ser publicado).

A.8 Publicação 8

Lucena Jr, V. F.; CHAVES Filho, J. E.; VIANA, N. S.; MAIA, O. B.. ***A Home Automation Proposal Built on the Ginga Digital TV Middleware and the OSGi Framework***. In: *IEEE Transactions on Consumer Electronics*, Vol. 55, N.o 3, ISSN 0098-3063. Agosto, 2009 (A ser publicado).

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo