UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA INTEGRAÇÃO ENTRE AMBIENTES INTELIGENTES E O SISTEMA BRASILEIRO DE TV DIGITAL

Porto Alegre
(2011)

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA INTEGRAÇÃO ENTRE AMBIENTES INTELIGENTES E O SISTEMA BRASILEIRO DE TV DIGITAL

Tese de doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Doutor em Engenharia Elétrica.

Área de concentração: Controle e Automação.

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

(2011)

REINER FRANTHESCO PEROZZO

FRAMEWORK PARA INTEGRAÇÃO ENTRE AMBIENTES INTELIGENTES E O SISTEMA BRASILEIRO DE TV DIGITAL

Esta tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS.

Doutor pela Universidade de Stuttgart – Stuttgart, Alemanha.

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS.

Doutor pelo Instituto Nacional Politécnico de Grenoble - Grenoble, França.

Prof. Dr. João César Netto, UFRGS.

Doutor pela Universidade Católica de Louvain – Louvain, Bélgica.

Prof. Dr. Marcelo Götz, UFRGS.

Doutor pela Universidade de Paderborn – Paderborn, Alemanha.

Prof. Dr. Vicente Ferreira de Lucena Júnior, UFAM.

Doutor pela Universidade de Stuttgart – Stuttgart, Alemanha.

Prof. Dr. Walter Fetter Lages, UFRGS.

Doutor pelo Instituto Tecnológico de Aeronáutica – São José dos Campos, Brasil.

Coordenador do PPGEE: _____

Prof. Dr. Alexandre Sanfelice Bazanella

Porto Alegre, março de 2011.

DEDICATÓRIA

Dedico este trabalho à minha esposa Manuela, pelo carinho, pela compreensão, pela cumplicidade e pela aliança bem sucedida.

AGRADECIMENTOS

Aos meus pais, pelo apoio.

Ao meu irmão, pela parceria.

Aos meus colegas de laboratório, pelo companheirismo.

Ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), pela oportunidade.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo provimento da bolsa no início do meu curso de doutorado.

Em especial, agradeço ao meu orientador Prof. Dr. Carlos Eduardo Pereira, com quem tive a felicidade de conviver ao longo desses anos. Além de ser um professor extraordinário, é uma pessoa que serve como exemplo a ser seguido. Muitíssimo obrigado por confiar em meu trabalho e por se tornar um grande amigo.

RESUMO

Desde dezembro de 2007, o Brasil está implantando o Sistema Brasileiro de TV Digital (SBTVD). Além de esse novo sistema proporcionar imagens em alta definição e permitir a mobilidade da TV, ele oferece a interatividade, através do *middlewar*e Ginga, o qual está sendo disponibilizado em receptores de TV digital para permitir que os telespectadores possam interagir com as aplicações que são transmitidas - juntas com o sinal de áudio e de vídeo - pelas emissoras de TV. Esses receptores de TV digital, ou *Set-top Boxes* (STBs), como são conhecidos, estão se tornando cada vez mais presentes nas residências, o que possibilita a sua integração com os Ambientes Inteligentes (AmIs), que são cenários compostos por sistemas de automação predial/residencial capazes de se adaptar à presença dos usuários, oferecer serviços e permitir uma interação multimodal com o ambiente. Assim, este trabalho propõe um *framework* para integração entre AmIs e o SBTVD, a fim de permitir que os serviços e os dispositivos de automação presentes nas residências possam ser gerenciados pelos receptores de TV digital compatíveis com o *middleware* de interatividade Ginga. O *framework* proposto define uma arquitetura que é implementada para atender às seguintes funcionalidades: (i) mapeamento dos dispositivos físicos de automação presentes no AmI para o mundo computacional; (ii) suporte ao desenvolvimento de aplicações interativas para acesso aos serviços e aos dispositivos de automação do AmI; (iii) criação de cenários de automação independentes da plataforma de *hardware* em que serão executados; (iv) reutilização de projetos para otimização do tempo de desenvolvimento de novas aplicações interativas; (v) geração automática de código, em que são construídas aplicações baseadas no perfil do *hardware* da plataforma-alvo e no perfil da linguagem de programação suportada pelo *middleware* de interatividade do SBTVD. Além disso, este trabalho é validado através de três estudos de casos que utilizam os conceitos e as ferramentas computacionais propostas no âmbito desta tese.

Palavras-chave: Ambientes Inteligentes, Automação Residencial, *Middleware* de Interatividade, TV Digital.

ABSTRACT

Since December 2007, Brazil has been introducing the Digital TV Brazilian System (*SBTVD*). This new system not only provides high definition images and TV mobility, but also offers interactivity by means of middleware Ginga, which has become available in Digital TV receivers in order to grant that viewers are able to interact with the transmitted applications – along with the video and audio signal – by broadcasting stations. Such Digital TV receivers, or the well-known Set-top Boxes (STBs), have ranked high in homes, a factor that enables its integration with Intelligent Environments (*AmIs*), which are sceneries composed by home automation systems, and may adapt to the presence of users, offer services and allow multimodal interaction with the environment. Thus, this thesis presents a framework for integration between AmIs and *SBTVD* in order to grant that services and automation devices located in homes can be managed by Digital TV receivers with Ginga interactivity middleware. The referred framework defines an architecture that is implemented with the aim of meeting the following functionalities: (i) physical devices mapping of automation in *AmIs* to the computational world; (ii) support to the development of interactive applications to access the *AmI* services and automation devices; (iii) creation of automation sceneries which do not depend on the hardware platform they will be run in; (iv) reuse of projects to optimize the development time of new interactive applications; (v) code automatic generation, in which applications will be constructed based on the profile of the target platform hardware and on the programming language profile supported by the *SBTVD* interactivity middleware. Besides, this work is validated due to three case studies that utilize concepts and computational tools underlying this thesis.

Keywords: Digital TV, Home Automation, Intelligent Environments, Interactive TV Middleware.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACAP: Advanced Common Application Platform

ADL: Architecture Description Language

AM: Adaptation Manager

AmI: Ambiente Inteligente

API: Application Programming Interface

ARIB: Association of Radio Industries and Businesses

ATSC: Advanced Television System Committee

BML: Broadcast Markup Language

BPEL: Business Process Execution Language

CD: Compact Disc

CM: Context Manager

DASE: Digital Television Application Software Environment

DPWS: Device Profile for Web Services

DVB: Digital Video Broadcasting

EAD: Ensino à Distância

ECA: Evento/Condição/Ação

EIB: European Installation Bus

FCDP: Format-Independent Change

FINEP: Financiadora de Estudos e Projetos

GCAR: Grupo de Controle, Automação e Robótica

GEM: Globally Executable Middleware

HD: High Definition

HSNET: HomeSystems Network

HTML: Hypertext Markup Language

HTTP: HyperText Transfer Protocol

IHM: Interface Homem-Máquina

IP: Internet Protocol

ISDBT: Integrated Services Digital Broadcasting

ITU: International Telecommunication Union

IXC: Inter-Xlet Communication

JSP: Java Server Pages

LAN: Local Area Network

MB: Megabytes

MDS: Mobile Data Server

MEADL: Mobile Enterprise Architecture Description Language

MEM: Mobile Environment Manager

MHP: Multimedia Home Platform

Mhz: Megahertz

NCL: Nested Context Language

OCAP: Open Cable Application Platform

OMG: Object Management Group

OSGi: Open Services Gateway initiative

PC: Personal Computer

PDA: Personal Digital Assistants

PiP: Picture-in-Picture

PLC: Power Line Communication

PM: Policy Manager

RBS: Rede Brasil Sul de Comunicação

RFID: Radio-Frequency IDentification

SBTVD: Sistema Brasileiro de TV Digital

SMTP: Simple Mail Transfer Protocol

SOA: Service-Oriented Architecture

SSL: Secure Sockets Layer

STB: Set-top Box

TAS: Terminals Adaptation System

TCP: Transmission Control Protocol

TLS: Transport Layer Security

TV: Televisão

UHF: Ultra High Frequency

UML: Unified Modeling Language

UPnP: Universal Plug and Play

USB: Universal Serial Bus

VXML: Voice Extensible Markup Language

WLAN: Wireless Local Area Network

WML: Wireless Markup Language

WSDL: Web Services Description Language

XHTML: eXtensible Hypertext Markup Language

XMI: XML Metadata Interchange

XML: eXtensible Markup Language

XSL: eXtensible Stylesheet Language

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Nos últimos anos, a automação predial/residencial tem se tornado um assunto bastante comentado e recorrente. Parte desse sucesso está relacionado com a grande difusão e oferta de aparelhos eletrônicos portáteis, com variado poder de computação, baixo consumo de energia e, principalmente, alto grau de conectividade. Juntamente com a utilização da automação predial/residencial, é definindo o termo "Ambientes Inteligentes" (AmIs), que traz uma mudança de paradigma: deixa-se de lado o fato de apenas controlar dispositivos e parte-se para um mundo onde os dispositivos se autogerenciam, estando cientes (*aware*) de tudo que os cercam (ARK & SELKER, 1999; DUCATEL, 2001). Com isso, começa a se tornar realidade a computação ubíqua, vislumbrada por (WEISER, 1991), e definida como sendo um estágio em que sistemas computacionais disponibilizam informações e serviços às pessoas, em qualquer lugar e em qualquer momento. A computação ubíqua e pervasiva podem ser vistas como áreas que integram o AmI (ANASTASOPOULOS *et. a.l.*, 2005).

No AmI, tem-se a visão de um mundo cercado por uma grande quantidade de dispositivos que oferecem suporte inteligente às atividades diárias dos usuários. O AmI consiste em um paradigma de informações tecnológicas em que os objetos informatizados estão introduzidos em um determinado ambiente físico que se adapta às diferentes necessidades e situações dos usuários (KIRSTE, 2005; ARTS, 2002), tendo autonomia para agir (LINDWER, 2003) e sendo programado para reconhecer ou, até mesmo, aprender o comportamento do usuário que vive nesse ambiente (LIU, 2004; HAGRAS, 2004).

O AmI está relacionado com a automação predial/residencial, uma vez que o desenvolvimento de projetos nessa área necessita de espaços físicos automatizados, incluindo sensores, atuadores e, principalmente, sistemas inteligentes para o gerenciamento e

otimização de tarefas (NAZARI, 2008; EDWARDS, 2006). Uma ampla variedade de serviços em áreas como segurança (controle de acesso, identificação de usuários), conforto (controle de temperatura e umidade, iluminação) e entretenimento está surgindo diariamente, trazendo consigo muitos problemas a serem resolvidos, como o aumento do consumo de energia elétrica nesses ambientes, que pode ser otimizado através da utilização de sistemas computacionais inteligentes que tratam do gerenciamento de energia. Outros problemas estão relacionados com as questões de mobilidade, adaptabilidade e heterogeneidade nesses ambientes, devido ao grande número de soluções existentes tanto em termos de *hardware* quanto em termos de *software*.

Em um AmI, pode ser encontrada uma grande variedade de redes de dispositivos inteligentes, permitindo a integração entre equipamentos eletroeletrônicos e pessoas, fornecendo informação, comunicação, serviços e entretenimento (ARTS, 2004). As redes de comunicação de dados nesses ambientes é um dos fatores essenciais para a construção de projetos, pois elas podem permitir desde a troca de informações entre os próprios eletrodomésticos, por meio da rede elétrica, até o gerenciamento do AmI, através das redes sem fios utilizadas por dispositivos móveis, como telefones celulares, *Personal Digital Assistants* (PDAs) e *smartphones*. Dessa forma, é necessário haver mecanismos adaptativos para tratamento e disponibilização de informações e serviços aos usuários, bem como a elaboração de estratégias que permitam ao usuário acessar e interagir com o AmI.

Além disso, as técnicas de interação entre homem e ambientes automatizados vêm recebendo, há algum tempo, uma atenção especial tanto no meio acadêmico quanto na indústria. Um exemplo está no centro de pesquisas da Philips (PHILIPS RESEARCH, 2008), a qual acredita que, no ano de 2020, as pessoas irão se relacionar com os dispositivos eletroeletrônicos de forma mais natural e amigável, uma vez que os AmIs e seus dispositivos

terão a capacidade de responder às necessidades e adaptar seu comportamento em função da presença dos usuários.

A área dos AmIs está cercada por uma diversidade de subáreas, exploradas com o claro objetivo de oferecer aos usuários maior conforto, praticidade, segurança ou, simplesmente, que os ambientes possam oferecer suporte inteligente a atividades diárias do ser humano. Como exemplos de projetos que tentam atingir o nível tecnológico previsto pela Philips, é possível citar o projeto Amigo (AMIGO, 2008), que envolve um esforço coletivo entre algumas das principais empresas e instituições de pesquisa da Europa, nos quesitos de telefonia celular, desenvolvimento de *software*, redes residenciais, e diversos outros segmentos com o objetivo de construir um *middleware* aberto e padronizado para suportar as mais variadas aplicações existentes em AmIs. Outra tentativa é liderada pela empresa Microsoft, através do projeto *Easy Living* (EASY LIVING, 2008), a qual busca o desenvolvimento de novas tecnologias, também para oferecer suporte na criação dos AmIs. No *Easy Living*, o foco está mais direcionado aos ramos de visão computacional, rastreabilidade (*tracking*), redes de sensores e interoperabilidade de dispositivos.

Além desses projetos, os quais possuem grandes recursos financeiros e equipes compostas por pesquisadores de diversas áreas do conhecimento, existem outras iniciativas de empresas que veem nos AmIs um grande potencial de mercado, como é o caso da empresa Homesystems (HOMESYSTEMS, 2010), cujo modelo de negócio está no desenvolvimento e na implantação de uma infraestrutura com vistas a projetos de automação predial/residencial, a qual é utilizada para a criação de cenários de AmIs desenvolvidos pela própria empresa. Existem, ainda, outras importantes insituições que atuam fortemente na área de desenvolvimento de produtos e sistemas para automação residencial, com soluções para controle de iluminação, segurança, gerenciamento de energia, irrigação de jardins, sistemas de áudio e vídeo e sistemas de interação. (ECHELON, 2010; HAI, 2010; INSTEON, 2010).

Além disso, o conceito dos AmIs está cada vez mais presente no cotidiano das pessoas, trazendo consigo muitas soluções e também vários desafios que devem ser tratados para se atingir o estágio tecnológico citado pela Philips Research, em que se destacam a busca por ambientes mais autônomos, sensíveis ao contexto e à presença dos usuários, facilmente adaptáveis e gerenciáveis através de diversos sistemas de interface homem-máquina (IHM).

Sobre esse aspecto, pode-se tirar proveito das tecnologias e conceitos utilizados em televisão (TV) digital, especialmente quando se trata do Sistema Brasileiro de TV digital (SBTVD) que, não só oferece imagens em alta definição e mobilidade, mas também permite interatividade, ou seja, aplicações interativas que são transmitidas (juntamente com o sinal de áudio e vídeo) pelas emissoras de TV e executadas nos receptores de TV digital, resultando na interação entre o telespectador e a programação da emissora. Neste caso, o conceito de interação entre telespectador e emissora de televisão pode ser utilizada como mais uma forma de interação no contexto dos AmIs, em que os receptores de TV digital, que são essencialmente projetados para decodificar o sinal de áudio e vídeo, podem agregar novas funcionalidades a fim de permitir que os telespectadores utilizem essas plataformas tanto para assistir à televisão quanto para gerenciar a automação de seus AmIs. Um exemplo desse conceito é apresentado na Figura 1.1, em que o telespectador interage com os dispositivos de automação do AmI através do televisor. Nesse caso, existem dois domínios de aplicação: o SBTVD e o AmI, sendo que a integração entre eles é a principal contribuição abordada por este trabalho.

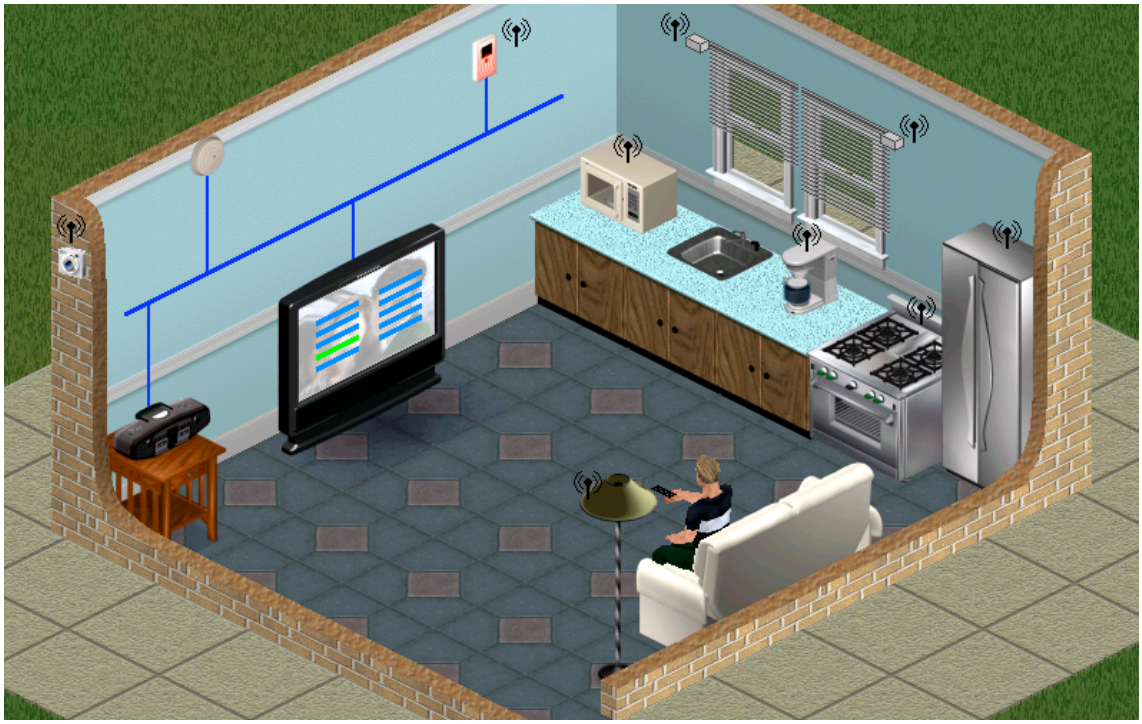

Figura 1.1 Gerenciamento de um AmI através de receptores de TV digital.

1.2 MOTIVAÇÃO

Em um AmI, o usuário pode dispor de uma variedade de plataformas computacionais que permitem a interação com os dispositivos e serviços automatizados, os quais oferecem maior praticidade, comodidade, suporte em atividades cotidianas ou, simplesmente, entretenimento. Da mesma forma como ocorre em AmIs, o conceito de interação também é encontrado em sistemas de TV digital, suportado por *middlewares* que oferecem mecanismos de interação entre telespectadores e radiodifusores. Um exemplo é o próprio SBTVD, o qual possui a especificação de um *middleware* de interatividade que permite aos telespectadores interagir com a programação que está sendo transmitida. Com a interatividade na TV, estão sendo criadas propostas em que a partir de um receptor de TV digital, o telespectador pode ter acesso a serviços bancários, governamentais, de comércio eletrônico, e diversos outros que visam à inclusão social e digital através da TV digital brasileira. Essa interatividade permite que, além dos serviços citados, possam ser criados outros tipos de aplicações, a fim de explorar esses recursos disponíveis no SBTVD.

Outro ponto relevante é o fato de que diversos outros países estão adotando o padrão brasileiro de TV digital (FÓRUM SBTVD, 2010), cuja disseminação, juntamente com os recursos de interatividade, torna importante a definição de uma estratégia para construir essas aplicações interativas. Uma vez que a especificação para o *middleware* do SBTVD é recente, não há ferramentas de autoria para a construção de aplicações interativas e muito menos para integração entre diferentes sistemas computacionais ou domínios de aplicação.

Uma das principais motivações deste trabalho está, justamente, no desafio de integrar a automação dos AmIs com o sistema de interatividade da TV digital, a fim de permitir que os serviços automatizados presentes nos AmIs possam ser acessados pelos receptores de TV. Isso pode resultar em mais uma modalidade de interação entre os usuários e os AmIs, o que pode ser atraente tanto do ponto de vista dos usuários, que teriam uma nova forma de interação com o ambiente, quanto do ponto de vista dos fabricantes de receptores e radiodifusores, que poderiam explorar o desenvolvimento de aplicações interativas para o mercado de automação predial/residencial.

Além disso, existem muitos desafios científicos que vêm sendo abordados pela comunidade acadêmica, contemplados em projetos de pesquisa, dissertações e teses. Dentre esses desafios, além da integração entre diferentes domínios de aplicação que se mantem em aberto diante de novas propostas que surgem constantemente, existe a busca por diferentes estratégias de interação humano-computador, que se aproxime cada vez mais das interações cotidianas entre as pessoas. Com a popularização das redes sociais e da Internet, há um avanço em pesquisa e desenvolvimento de plataformas convergentes, multifuncionais e que permitem ao usuário estar conectado o tempo todo, como é o caso dos novos modelos de televisores, cujas funções para navegação na *web* são muito parecidas com a de um computador. Tal exemplo mostra como a TV está além do conceito de somente assistir ao conteúdo da emissora ou a um determinado vídeo. Diante disso, existem outros trabalhos

acadêmicos que buscam solucionar os desafios relacionados à usabilidade da TV em que o controle remoto pode não ser, necessariamente, a melhor forma de interação para uma determinada tarefa. Outros trabalhos tratam dos desafios quanto à segurança dos dados que trafegam sobre as redes de comunicação, e utilizam essas plataformas como terminal de acesso.

Enfim, existem diversos desafios a serem explorados, tanto no meio acadêmico quanto na indústria, para oferecer aos usuários diferentes soluções de interação e de acesso aos novos serviços que surgem constantemente.

1.3 OBJETIVOS

O objetivo deste trabalho é a integração entre AmIs e o SBTVD, através da proposta de um *framework* capaz de permitir que os serviços e os dispositivos de automação presentes no AmI possam ser acessados por aplicações que são executadas sobre o *middleware* de interatividade do padrão brasileiro de TV digital. Essa estratégia busca agregar maior valor aos receptores de TV digital, permitindo que estes possam, além de decodificar o sinal da TV digital, se tornar plataformas computacionais de gerenciamento do AmI, oferecendo aos usuários uma nova modalidade de interação com o AmI. Assim, o *framework* proposto possui as seguintes funcionalidades: (i) mapear os dispositivos físicos de automação presentes no AmI para o mundo computacional, através de modelos orientados a objetos ou da descrição das classes de dispositivos; (ii) auxiliar na construção de aplicações interativas para acesso aos serviços e aos dispositivos de automação existentes nos AmIs; (iii) permitir a criação de cenários de automação independentes da plataforma de *hardware* em que serão executados; (iv) possibilitar a reutilização de projetos, otimizando o tempo de desenvolvimento de novas aplicações; (v) permitir a geração automática de código, criando aplicações baseadas no perfil

do *hardware* da plataforma-alvo e no perfil da linguagem de programação suportada pelo *middleware* de interatividade do SBTVD.

1.4 ORGANIZAÇÃO DO TEXTO

O presente texto está organizado da seguinte forma: o próximo capítulo apresenta uma breve fundamentação teórica, em que são abordados os principais conceitos que servem como base para a elaboração do trabalho. São tratados os temas relacionados com AmIs, TV digital e *middlewares* de interatividade. O capítulo 3 relaciona o estado da arte, sendo destacadas arquiteturas, *middlewares*, *frameworks* e estratégias que tratam desde o gerenciamento de serviços em AmIs até os sistemas de interação multimodal e as aplicações para TV digital interativa. No capítulo 4 é apresentada a proposta da tese, a qual está dividida em duas partes: conceitual e implementação. O capítulo 5 contém a validação do trabalho, sendo apresentados três estudos de casos que utilizam os conceitos e as implementações definidas no capítulo anterior. Por fim, são expostas as conclusões e os direcionamentos para a elaboração de trabalhos futuros que poderão ser realizados a partir desta tese.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 AMBIENTES INTELIGENTES (AMIS)

Os AmIs podem ser definidos como um conjunto de cenários que são construídos para oferecer suporte inteligentes nas atividades diárias dos usuários (BOLSANI, 2004). Esses cenários são suportados por uma infraestrutura de automação predial/residencial, o que envolve uma grande variedade de sensores, atuadores e controladores. Porém, nos AmIs, o nível de abstração é maior que na automação predial/residencial, pois o usuário não, necessariamente, está interessado em saber se um determinado dispositivo de automação está ou não presente na residência. Ele necessita saber se o AmI pode lhe oferecer um determinado serviço, independente de qual dispositivo irá provê-lo (PEROZZO & PEREIRA, 2008).

Com base nessa premissa, é necessária a existência de sistemas computacionais inteligentes para gerenciamento da automação predial/residencial presente nos AmIs. Esses sistemas inteligentes podem auxiliar nas mais variadas tarefas e realizar verificação de consistência das informações, tal como: “fechar as janelas quando elas estiverem abertas se nenhum usuário estiver em casa e a previsão do tempo for chuva”. Existem algumas técnicas utilizadas na tentativa de tornar um sistema apto a tomar esse tipo de decisão. Uma delas é a criação de regras específicas, para cada possível evento, que são definidas *a priori* (HOMESYSTEMS, 2010). Outra técnica é utilizar a inteligência artificial, com algoritmos de autoaprendizagem, para que os sistemas de gerenciamento aprendam o comportamento do usuário e evoluam conforme a dinâmica do ambiente. Ambas as técnicas possuem alguma limitação, por exemplo: na primeira, existe a necessidade de uma constante inserção de novas regras, as quais não tenham sido previstas no início do projeto. Na segunda técnica, um algoritmo de autoaprendizagem teria muita dificuldade para mapear todas as possíveis situações que ocorreriam no ambiente (BOLSANI, 2004), e quanto maior a complexidade dos

algoritmos para tomar decisões, menor será a transparência para o usuário (EDWARDS & GRINTER, 2001).

Existem, ainda, outras formas para tomadas de decisão em AmIs (BOLSANI, 2004), tais como: (i) sistemas não coercitivos, que sinalizam os eventos e deixam o usuário tomar a decisão, (ii) sistemas de caráter geométrico, que limitam a área de atuação dos dispositivos de automação, e (iii) sistemas de rastreamento e percepção, capazes de identificar a localização dos usuários e tentar prever o seu comportamento (EMBASSI, 2008).

2.1.1 Automação Predial/Residencial

A automação predial e residencial teve a sua origem na industrial, herdando além dos conceitos, uma variedade de dispositivos e redes de comunicação. Porém, o contexto da automação em residências difere dos existentes em fábricas, por exemplo: em uma residência podem não ser necessários complexos dispositivos e lógicas para controle de processos produtivos, contudo, é possível encontrar uma infraestrutura de comunicação e diversos equipamentos que geram tráfego de dados, sendo que os usuários podem não ter um conhecimento técnico sobre o assunto (BOLSANI, 2004). Desse modo, ao contrário da indústria, os projetos de automação predial/residencial devem considerar a inexistência de um engenheiro de automação, ou seja: a automação nesses ambientes deve ser onipresente para os usuários (WEISER, 1991).

Apesar de a automação predial e a residencial estarem relacionadas e utilizarem recursos similares, elas divergem em alguns aspectos conceituais: a automação predial está baseada no conceito de integração de sistemas eletroeletrônicos e eletromecânicos, cujo objetivo é obter maior eficiência dos recursos disponíveis. Normalmente, a automação direcionada para esse domínio de aplicação adota como premissa a existência de um usuário padrão, na tentativa de planejar e criar um cenário de automação que permita atender à

maioria dos usuários. Além disso, no âmbito predial, a automação nem sempre é transparente para o usuário, o que dificulta a sua interação com o ambiente. Talvez essas sejam as principais características que a diferenciem da automação residencial, em que o usuário está em constante interação com o sistema. Nesse caso, a premissa está em customizar o ambiente para cada ocupante, ou seja, é considerada a criação de diferentes perfis de utilização e interação com o ambiente (BOLSANI, 2004).

Entretanto, um ponto de convergência está associado com o consumo de energia elétrica em ambientes automatizados, o qual pode ser reduzido através da utilização de sistemas para gerenciamento de energia. Mesmo parecendo contraditório reduzir o consumo de energia através da instalação de novos dispositivos eletrônicos, essa abordagem pode ser eficiente (PEROZZO *et al.*, 2008a). A utilização de *softwares* de gerenciamento em conjunto com sensores e atuadores podem otimizar os recursos disponíveis no ambiente, permitindo um uso mais racional e inteligente da energia. Controladores de portas, janelas e persianas possibilitam uma maior utilização da luz natural, mas balanceando a incidência direta do sol para não sobrecarregar o ar condicionado. Um exemplo sobre redução de gastos com consumo de energia seria evidente caso a concessionária de energia elétrica realizasse a cobrança de forma horo-sazonal, isto é, levando em consideração a estação e o horário em que a energia foi utilizada, de modo similar ao que ocorre com o sistema telefônico. Nessa situação, seria vantajoso que os equipamentos do ambiente pudessem negociar com a concessionária o melhor horário para entrar em funcionamento, obtendo, assim, menores taxas (PEROZZO *et al.*, 2008b). Outro exemplo é durante um período de não utilização do ambiente, em que o sistema de gerenciamento poderia ser capaz de alterar a iluminação e a climatização para situações como “noturno”, “férias”, ou “não funcionamento”. Durante o período de utilização, o sistema seria capaz de conservar energia e otimizar o uso dos

equipamentos, seguindo sempre os limites pré-definidos de conforto ou segurança. (BOLSANI, 2004).

2.1.2 Redes Prediais/Residenciais

A evolução das redes prediais/residenciais e, por consequência, a automação predial/ residencial, está baseada no fato de ela permitir a comunicação entre dispositivos, oferecendo aos AmIs acesso tanto aos seus recursos internos quanto aos recursos e serviços disponíveis no mundo externo (HELAL, 2005). Nos AmIs, existe uma série de subsistemas que oferecem algum tipo de funcionalidade, tais como: (i) telefonia: sistema telefônico, intercomunicadores, porteiros eletrônicos; (ii) informática: rede residencial para comunicação de dados, acesso compartilhado, serviços via Internet; (iii) rede elétrica: controle de cargas, sistema de distribuição de energia, monitoramento de falhas, sistema de emergência, rede de dados; (iv) iluminação: emergência, decorativa, externa, cenários; (v) climatização: condicionador de ar, ventilação, abertura e fechamento de janelas, cortinas e persianas.

Com todas essas funcionalidades oferecidas pelos AmIs, um dos grandes desafios está na integração dos diferentes dispositivos de automação presentes nas residências. Normalmente, nesses casos, é comum encontrar o papel de um sistema computacional integrador, responsável por estabelecer a comunicação e a cooperação entre os dispositivos do AmI. A Figura 2.1 ilustra como as camadas de uma rede residencial podem trabalhar em conjunto. No centro, o integrador de sistemas residenciais é o responsável pela harmonia e interoperabilidade de todo o conjunto.

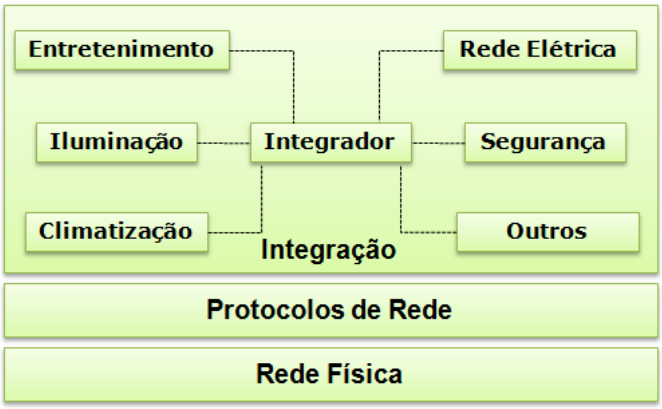

Figura 2.1 Integração entre serviços - Adaptado de (BOLSANI, 2004).

Nem toda a forma de comunicação encontrada em AmI é centralizada e/ou depende de um integrador, pois essa estratégia pode tornar o sistema hierarquicamente dependente. Existe, também, a visão de sistemas inteligentes e distribuídos, de forma que a comunicação é realizada diretamente entre dispositivos (OTSUKA, 2006).

O avanço tecnológico permite que, cada vez mais, seja possível integrar dispositivos de automação, de áudio, de vídeo e outros eletroeletrônicos com computadores e, consequentemente, com o AmI. De acordo com (BOLSANI, 2004), a convergência dos sistemas reduz os preços, permite a conexão entre equipamentos e viabiliza a construção dos AmIs. Na Figura 2.2, é apresentado um modelo de redes residenciais, contendo um extenso mercado de dispositivos e *softwares* que agrega maior valor aos ambientes e motiva uma cadeia de produtos e serviços para AmIs.

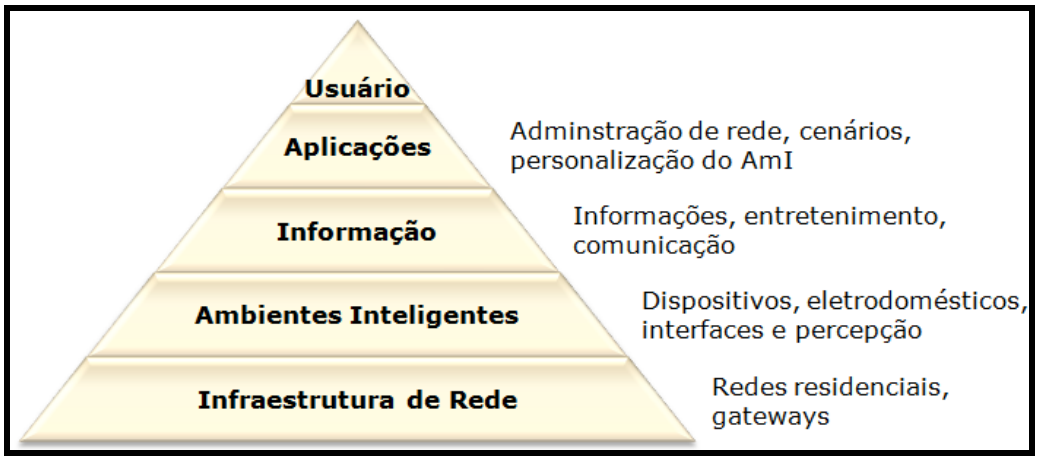

Figura 2.2 Pirâmide de dispositivos e serviços - Adaptado de (BOLSANI, 2004).

A pirâmide de dispositivos e serviços ilustrada na Figura 2.2 é dividida em cinco níveis: (i) Infraestrutura de Rede: corresponde à transmissão de dados do sistema e provê a troca de informações entre os dispositivos do AmI, bem como permite aos usuários acesso aos serviços que utilizam a Internet, por exemplo; (ii) Ambientes Inteligentes: camada responsável por concentrar a infraestrutura de automação predial/residencial, incluindo sensores, atuadores, controladores e interfaces de interação com o AmI; (iii) Informação: corresponde aos diferentes conteúdos presentes no mundo digital, tais como: música, filmes, textos, gráficos. Nessa camada os usuários podem ter acesso a informações que vão desde bolsa de valores, previsão do tempo e rádios online, até dados estatísticos ou *logs* dos sistemas de automação predial/residencial; (iv) Aplicações: camada basicamente composta por ferramentas computacionais de administração da rede, gerenciamento e customização de cenários de automação, bem como aplicações multimodais para interação com o AmI; (v) Usuários: utilizadores dos dispositivos e serviços existentes no ambiente, podendo existir para cada usuário diferentes perfis e níveis de acesso às funcionalidades do AmI.

Em linhas gerais, um dos principais fatores a ser considerado na construção dos AmIs é o planejamento do sistema de informação, buscando identificar aspectos fundamentais, como: a integração, a interoperabilidade, a infraestrutura de automação predial/residencial, além dos sistemas de gerenciamento e interação.

2.2 TV DIGITAL

Uma das principais características da TV digital está associada com a qualidade de imagem, proporcionando aos telespectadores a alta definição (1920 x 1080 pixels) e uma recepção livre de fantasmas ou chuviscos, propriedades que fazem parte das transmissões de TV analógica. Essa qualidade de imagem pode ser perceptível em televisores compatíveis com a tecnologia digital, chamados de televisores *Full High Definition* (*Full* HD), que

possuem alta resolução e um formato de aspecto 16:9, similar às telas de cinema. Além disso, o áudio deixa de ser dois canais (estéreo) e passa a ser seis canais (*surround*), o que permite uma maior imersão no conteúdo que está sendo transmitido pela emissora (TV GLOBO, 2008; RBS TV, 2010).

Outro aspecto importante da TV digital é a mobilidade, em que os telespectadores podem receber o sinal da TV digital em telefones celulares, TVs portáteis e outros aparelhos eletrônicos que ofereçam essa funcionalidade. O que diferencia a mobilidade da recepção fixa é, justamente, a capacidade de se assistir ao conteúdo televisivo em movimento e de forma gratuita. Além disso, existe a interatividade (GINGA, 2010), uma tentativa de oferecer novas funcionalidades ao telespectador. No SBTVD, além do áudio e do vídeo, são transmitidas pelo ar aplicações interativas, que podem ou não estar relacionadas com o conteúdo que a emissora está transmitindo, oferecendo ao telespectador a possibilidade de interagir com essas aplicações (DTV, 2010).

A interatividade é dividida em dois modos: (i) sem canal de retorno: o telespectador interage localmente com a aplicação recebida, não podendo acessar o mundo externo - aplicação e/ou receptor de TV sem conexão à Internet; (ii) com canal de retorno: além de interagir com a aplicação, o telespectador tem acesso ao mundo externo, podendo acessar serviços baseados na *web*, tais como enquetes e portais de interatividade - aplicação e receptor com conexão à Internet (ABNT NBR 15607-1, 2008). Um dos motivadores da interatividade está relacionado com a inclusão social, cuja ideia é permitir que o cidadão consiga, através do seu televisor, ter acesso aos mais variados tipos de serviços, tais como: governo (*t-Government*), bancário (*t-Banking*), comércio eletrônico (*t-Commerce*) e educação à distância (*t-Learning*) (DOU, 2003).

O SBTVD é considerado como o sistema de TV digital terrestre mais avançado do mundo, por utilizar tecnologias inovadoras e apresentar vantagens em relação aos outros

padrões, como, por exemplo, o *middleware* de interatividade, com código aberto e livre de *royalties* (SET, 2008).

2.2.1 Middleware de Interatividade

O *middleware* de interatividade é uma camada de *software* localizada entre o sistema operacional dos receptores de TV digital e as aplicações interativas (SOARES, 2008). Essa camada tem o objetivo de tornar os aplicativos independentes da plataforma de *hardware* e de *software* dos fabricantes de receptores. Com isso, as aplicações passam a ser desenvolvidas com base em um *middleware* padronizado, agilizando o processo de criação e garantindo a compatibilidade das aplicações com todos os receptores que utilizam a implementação desse *middleware* padrão. (BARBOSA & SOARES, 2008).

Uma das principais funções de um *middleware* é oferecer suporte às aplicações, através de uma interface de programação de aplicativos - *Application Programming Interface* (API), cuja funcionalidade está atrelada aos requisitos das aplicações. Além disso, as aplicações de TV digital são, normalmente, desenvolvidas com base em dois modos de programação: declarativo e imperativo. As linguagens de programação declarativas são consideradas de mais alto nível de abstração, em que o programador fornece apenas o conjunto de tarefas a serem realizadas, não se preocupando com os detalhes de como o interpretador ou compilador implementará essas tarefas. Entre as linguagens declarativas mais comuns estão a *Nested Context Language* (NCL) (ABNT NBR 15606-2, 2009) e a *eXtensible Hypertext Markup Language* (XHTML) (W3C, 2002). Já no modo imperativo, ou procedural, deve ser informado cada passo a ser executado, em que o programador pode estabelecer todo o fluxo de controle e execução da aplicação. Entre as linguagens procedurais mais comuns para o domínio de TV digital estão C, Java, ECMAScript e Lua. A maioria dos *middlewares*

para TV digital terrestre oferecem suporte tanto ao desenvolvimento de aplicações utilizando linguagens declarativas quanto procedurais (BARBOSA & SOARES, 2008).

2.2.2 Padrões de Middlewares

Os principais sistemas de TV digital utilizam o padrão de interatividade *Globally Executable Middleware* (GEM), recomendado pela (ITU-T, 2001). Cada *middleware* possui suas próprias características, mas a recomendação é utilizada como referência, na tentativa de evitar uma proliferação de padrões de *middleware* (DAMASCENO, 2008).

2.2.2.1 MHP – Multimedia Home Platform

O *Multimedia Home Platform* (MHP, 2010) é o *middleware* do padrão europeu e busca oferecer um ambiente de TV interativa independente de plataforma, sendo aberto e interoperável para receptores. O ambiente de execução é baseado no uso de uma máquina virtual Java e um conjunto de APIs que permitem aos aplicativos, desenvolvidos em linguagem Java, o acesso padronizado a recursos e funcionalidades do receptor de TV digital.

Além da API Java, o MHP 1.1 possui a linguagem de programação semelhante ao *Hypertext Markup Language* (HTML), denominada *Digital Video Broadcasting* (DVB)-HTML. O MHP permite, entre outras funções, (i) carregar aplicações através do canal de interatividade; (ii) armazenar aplicações em memória persistente, (iii) acessar leitores de *smart cards*; e (iv) controlar aplicações baseadas na *web*, incluindo a leitura de e-mails.

2.2.2.2 DASE – DTV Application Software Environment

O *Digital Television Application Software Environment* (DASE) (ATSC, 2010) é o padrão norte-americano para a camada de *middleware* dos receptores de TV digital. Assim como no padrão MHP, o DASE utiliza uma máquina virtual Java para a execução de aplicações interativas. Esse padrão permite o uso de linguagens declarativas utilizadas para desenvolvimento de sistemas *web*, tais como HTML e JavaScript. Apesar de terem como base uma máquina virtual, os padrões MHP e DASE não são compatíveis, ou seja, um aplicativo

desenvolvido para o MHP não poderá ser utilizado no *middleware* DASE. Esse padrão está sendo substituído nos Estados Unidos pelo *Advanced Common Application Platform* (ACAP) e pelo *Open Cable Application Platform* (OCAP), visando a uma melhor compatibilidade com os demais padrões em uso. O ACAP é o resultado da harmonização dos padrões de *middleware* OCAP e DASE, do *Advanced Television System Committee* (ATSC). Essa harmonização garante a compatibilidade entre as transmissões por cabo e terrestre, permitindo que as aplicações interativas possam ser executadas em qualquer sistema de TV dos Estados Unidos.

2.2.2.3 ARIB – Association of Radio Industries and Businesses

O *Association of Radio Industries and Businesses* (ARIB, 2010) é o *middleware* japonês presente no *Integrated Services Digital Broadcasting* (ISDB). Esse *middleware* é composto por algumas especificações, como a ARIB STD-B24 (*Data Coding and Transmission Specification for Digital Broadcasting*) a qual define uma linguagem de programação declarativa, chamada de *Broadcast Markup Language* (BML) (ARIB B-24, 2004). Essa linguagem é baseada na *eXtensible Markup Language* (XML) e é utilizada para a especificação de serviços multimídia de TV digital.

Outra especificação desse padrão é a ARIB-STD B23 (*Application Execution Engine Platform for Digital Broadcasting*), baseada na especificação DVB-MHP. Essa especificação é uma tentativa para estabelecer um núcleo comum entre o seu padrão de *middleware*, o MHP e o DASE.

2.2.2.4 GEM – Globally Executable MHP

O GEM (GEM, 2009) é uma especificação padrão para terminais de acesso, em que as implementações de *middlewares* devem se adaptar para obter uma conformidade que garanta a execução global de aplicações. Essa especificação foi proposta para que as

aplicações desenvolvidas para o MHP pudessem ser utilizadas, também, em outras plataformas com outros *middlewares*, tais como o OCAP e o ARIB.

Além disso, o padrão define um conjunto de APIs, protocolos e formatos de conteúdos em que as aplicações interoperáveis podem utilizar para a construção de serviços interativos. Uma das vantagens da especificação GEM é a harmonização dos diferentes *middlewares* de interatividade existentes, criando padrões e tentando garantir a interoperabilidade entre as plataformas.

2.2.2.5 Ginga

Ginga (GINGA, 2010) é a especificação do *middleware* para o padrão brasileiro de TV digital terrestre. Uma de suas principais características está na sua utilização em diferentes sistemas de recepção, tais como: TVs, *Set-top Boxes* (STBs) e telefones celulares (SOARES, 2008). As aplicações executadas sobre Ginga são classificadas em duas categorias: (i) procedurais, escritas usando a linguagem Java e (ii) declarativas, escritas usando linguagem NCL. Esse suporte a diferentes paradigmas de programação só é possível graças à arquitetura do *middleware* Ginga, dividida em três núcleos principais: Ginga- NCL, Ginga-J e Ginga-CC. O Ginga-NCL foi desenvolvido para prover uma infraestrutura de apresentação de aplicações baseadas em documentos hipermídia. O Ginga-NCL utiliza uma linguagem declarativa, baseada em XML e, também, possui um ambiente procedural, baseado na linguagem de *scripts* LUA. Uma das características da linguagem declarativa é que ela descreve *o que* e não *como* seus procedimentos funcionam, ou seja, descrevem propriedades da solução desejada, não especificando como o algoritmo, em si, deve agir. O Ginga-J tem como principal funcionalidade processar as aplicações procedurais (*Xlets* Java). Um dos componentes mais importantes do ambiente procedural é o mecanismo de execução do conteúdo, que tem como base uma máquina virtual Java. (ABNT 15606-4). Por outro lado, o Ginga-CC oferece o suporte básico tanto para o ambiente declarativo quanto para o procedural. Como uma de suas

atribuições está o tratamento de exibição dos objetos de mídia. Além disso, ele controla o acesso ao canal de interatividade, que é o módulo responsável por gerenciar o acesso à camada de rede. O Ginga-CC também é responsável por obter conteúdos, através de procedimentos e decodificadores que podem atuar tanto no fluxo do *transport stream* quanto no canal de comunicação com a Internet, também conhecido como canal de interatividade. (BRACKMANN, 2010), (ABNT NBR 15606-5). A Figura 2.3 apresenta uma visão de alto nível da arquitetura Ginga.

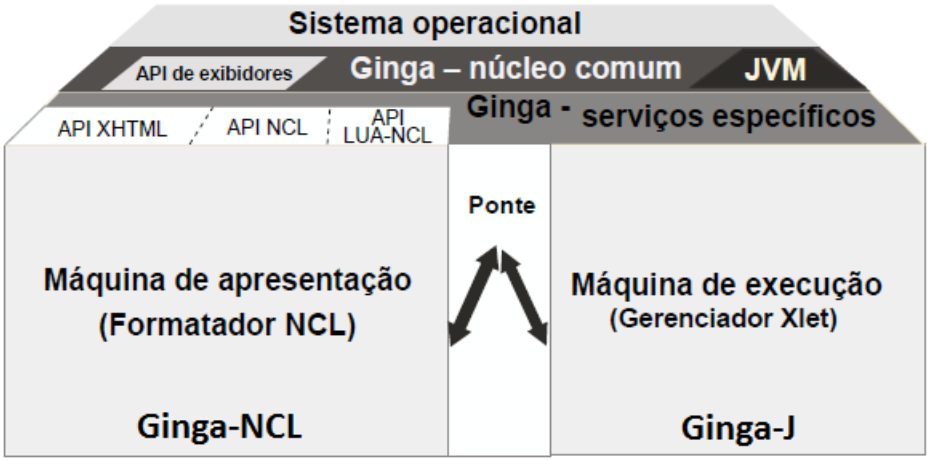

Figura 2.3 Arquitetura alto nível do Ginga (Adaptado de ABNT NBR 15606-5).

2.2.2.6 Comparação entre o Ginga e os demais middlewares de interatividade

A especificação do *middleware* Ginga apresenta uma série de inovações em relação aos outros *middlewares* existentes ao redor do mundo (SOARES, 2008). Apesar de a interatividade ser o objetivo de todos os *middlewares* de TV digital, o Ginga possui duas grandes vantagens: a primeira está relacionada com inclusão social/digital, uma vez que a especificação foi criada para atender à realidade brasileira, permitindo levar ao cidadão todos os meios para que ele tenha acesso à informação, educação e serviços sociais através do uso da televisão, o que inclui o acesso aos serviços de governo, bancários, educacionais e de âmbito geral (BRACKMANN, 2010). A segunda vantagem está associada com a tecnologia, em que uma das principais inovações está no uso da linguagem NCL, capaz de prover

mecanismos de adaptabilidade e sincronização de forma declarativa, sem a necessidade de se utilizarem *scripts* procedurais.

Conforme (DAMASCENO, 2008) os *middlewares* declarativos dos sistemas de TV digital europeu (DVB-HTML), americano (ACAP-X) e japonês (BML) possuem a implementação de uma máquina de apresentação para controle de aplicações declarativas. Mais especificamente, essas máquinas de apresentação são navegadores XHTML, em que é privilegiada a interatividade em detrimento da sincronização e adaptabilidade de conteúdo. A sincronização e adaptabilidade são realizadas através de *scripts* procedurais (ECMA*Scripts*). Além disso, o Ginga foi desenvolvido para permitir a interatividade em qualquer receptor compatível com o sistema brasileiro de TV digital, isso significa que o telespectador poderá utilizar a interatividade em qualquer receptor fixo ou portátil, devido aos diferentes perfis especificados para esse padrão (ABNT NBR 15606-2, 2009). A Tabela 1 elenca uma comparação entre os diferentes padrões de *middlewares* e suas respectivas tecnologias envolvidas.

Tabela 1 Padrões e tecnologias - Adaptado de (BARBOSA & SOARES, 2008).

Middleware	Padrão de TV	Ambiente Declarativo	Ambiente Procedural
DASE	Americano/ATSC	ACAP-X (ATSC A-101, 205) XHTML like e ECMAScript	ACAP-J (ATSC A-101, 2005) Java
MHP	Europeu/DVB-T	DVB-HTML (ETSI TS 102 812 V1.2.2, 2006) XHTML like e ECMAScript	MHP (ETSI TS 102 812 V1.2.2, 2006) Java
ARIB-BML	Japonês/ISDB-T	ARIB - BML (ARIB B-24, 2004) XHTML like e ECMAScript	Opcional (GEM - (ETSI TS 102 819 V1.2.1, 2005) like) não implementado
Ginga	Brasileiro/SBTVD	Ginga-NCL (ABNT NBR 15606-5, 2009) NCL e LUA	Ginga-J (ABNT NBR 15606-5, 2010) Java

Atualmente, além da adoção do padrão brasileiro por diversos outros países (TELECO, 2010), o Ginga foi reconhecido como padrão de interatividade pela *International Telecommunication Union* (ITU) (ITU H.761, 2009) o que impulsiona ainda mais a sua utilização e cria uma expectativa de convergência mundial ao Ginga (DE LUCA, 2009).

3 ANÁLISE DO ESTADO DA ARTE

3.1 GERENCIAMENTO DE SERVIÇOS EM AMBIENTES INTELIGENTES

Muitos sistemas de computação pervasiva desenvolvidos no passado, atualmente apresentam uma falta de habilidade para tratar tanto das novas tecnologias que surgem como das alterações de requisitos e de necessidades dos usuários em diversos domínios de aplicação. Há um esforço coletivo em grupos de pesquisa de universidades e de indústrias para se desenvolverem protótipos de sistemas que demonstrem os benefícios da computação pervasiva nos mais diversos segmentos (INDULSKA, 2001; HSU, 2007; PARK, 2006; OTSUKA, 2006; XU, 2007). Os projetos, basicamente, estão focados na integração de sistemas e no gerenciamento dos serviços disponíveis no AmI, o que inclui interconexão, sensores, atuadores, computadores, interfaces de interação e outros dispositivos presentes no ambiente (PEROZZO & PEREIRA, 2008).

A proposta Gator Tech (HELAL, 2005) para ambientes pervasivos programáveis é um projeto vinculado ao laboratório de computação móvel e pervasiva da Universidade da Flórida, cujo objetivo é oferecer um modo de desenvolvimento de tecnologias inteligentes. O projeto considera a existência de ambientes de execução e bibliotecas de *software* para o desenvolvimento de AmIs. A descoberta de serviços e os protocolos de *gateways* integram, automaticamente, componentes de sistemas utilizando um *middleware* genérico que suporta uma definição de serviço para cada sensor e atuador no ambiente. Dentro da abordagem proposta no projeto Gator Tech, estão destacados mecanismos que variam desde o tratamento de mensagens eletrônicas, com percepção e notificação aos usuários, passando pela identificação de usuários e controle de acesso ao ambiente através de *tags* e leitores de *Radio-Frequency IDentification* (RFID), e chegando ao conceito de banheiros com sistemas para exibição de mensagens importantes no espelho, além de possuírem sensores de fluxo de água

e de temperatura. Outras características da proposta envolvem os refrigeradores inteligentes e o monitoramento e segurança residencial, com chamadas de emergência e diversos outros serviços que auxiliam os usuários, contemplam o domínio do ambiente pervasivo programável proposto em (HELAL, 2005).

Para criar a casa inteligente Gator Tech, foi desenvolvida uma arquitetura de referência que pode ser utilizada na construção de outros AmIs. Conforme ilustra a Figura 3.1, a arquitetura é composta por seis camadas: física, plataforma de sensores, serviços, conhecimento, gerenciamento de contexto e aplicação.

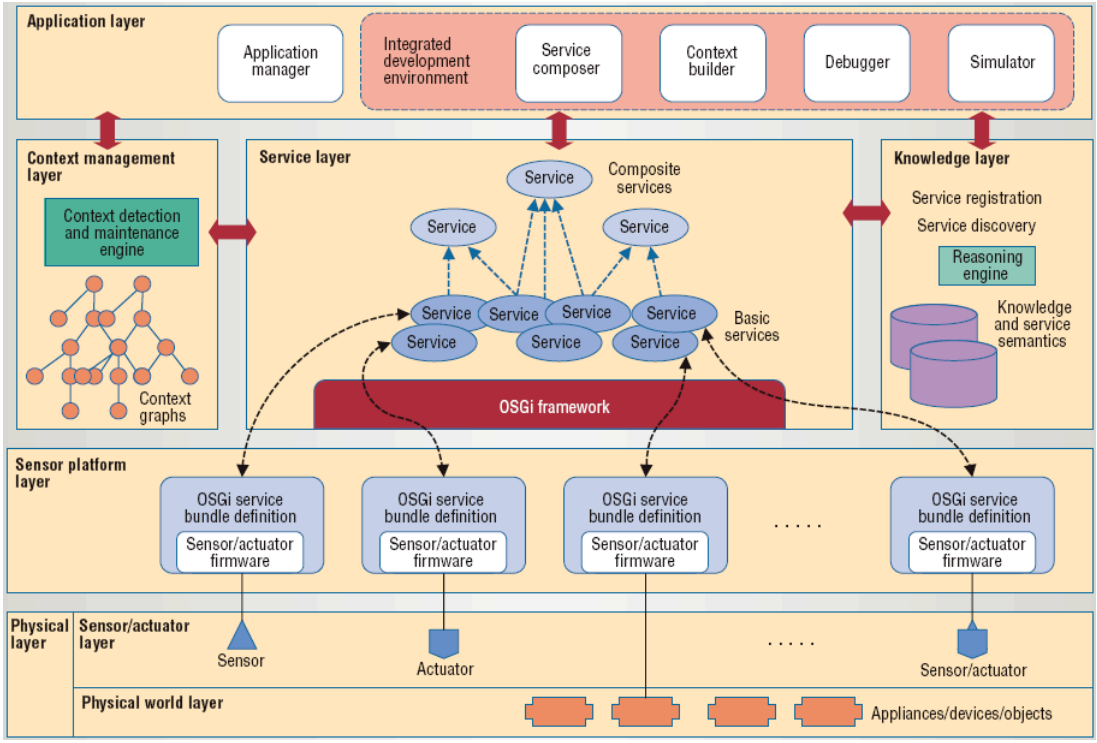

Figura 3.1 Arquitetura de referência genérica (HELAL, 2005).

A camada física contém os dispositivos que são utilizados pelos usuários, tais como: lâmpadas, televisores, STBs, aparelhos de ar condicionado, termostatos e outros. Na camada de plataforma de sensores, existe a comunicação com os dispositivos de automação, tais como: sensores e atuadores. Nesse caso, é realizada uma conversão de sensores e atuadores da camada física para serviços de *software* que podem ser programados ou compostos por outros serviços. Uma vantagem é que os desenvolvedores podem definir serviços sem ter que,

necessariamente, entender o mundo físico dos dispositivos. Já a camada de serviço é baseada no *framework Open Services Gateway initiative* (OSGi) (OSGi, 2011), em que serviços básicos representam o mundo físico por meio da plataforma de sensores, consistindo em *bundles* (elementos ou aplicações de serviços que podem ser instalados no *framework* OSGi) de serviços para cada sensor e atuador representado no *framework* OSGi. A vantagem, nesse caso, é a de que os desenvolvedores de aplicações podem criar serviços compostos utilizando um protocolo de descoberta de serviços para encontrar os serviços que, posteriormente, serão utilizados. Na camada de conhecimento está a informação sobre os vários serviços, dispositivos e aplicações domésticas conectadas ao sistema. Isso cria a possibilidade de identificação sobre os serviços, como, por exemplo: o sistema deve converter a saída de um sensor de temperatura antes de realizar outro serviço. A *engine* de raciocínio determina se algum serviço composto está disponível em um dado momento. A camada de gerenciamento de contexto permite que os desenvolvedores de aplicação criem e registrem os contextos de interesse. Cada contexto é implementado como um serviço OSGi e pode definir ou restringir a ativação de serviços para várias aplicações e necessidades. A *engine* de contexto é responsável por detectar e recuperar os estados.

Finalmente, é na camada de aplicação que se torna possível ativar e desativar serviços e interagir com o ambiente. Um componente interessante nessa camada é o *context builder*, o qual permite que um desenvolvedor construa visualmente um gráfico e relacione o seu comportamento com o contexto. Os desenvolvedores podem, ainda, utilizar um agregador de serviços para procurar, descobrir e registrar um novo serviço.

Quanto aos módulos implementados no projeto Gator Tech, vale salientar a existência de um mecanismo chamado *smart plugs*, que oferece um modo inteligente de percepção quanto aos dispositivos elétricos/eletrônicos que são instalados no ambiente. Cada tomada de energia elétrica do ambiente é equipada com um leitor RFID conectado a um computador

principal. Já os dispositivos que utilizam a eletricidade (lâmpadas, aparelhos de ar condicionado, dentre outros) possuem uma *tag* RFID anexada ao conector de alimentação do dispositivo que contém informações individuais. A ideia é: quando o usuário conectar um dispositivo elétrico à tomada, a *tag* do dispositivo é lida e as informações são enviadas ao computador central, através do leitor de RFID instalado em cada local. Isso permite identificar dispositivos e controlá-los de forma transparente por meio dos serviços oferecidos pelo ambiente.

Outra proposta é apresentada em (REDONDO, 2007), cujo objetivo é encapsular as informações lógicas de composição de serviços dentro de uma aplicação OSGi, sendo que tal composição é registrada em um *framework*. Quando o serviço composto é instanciado, uma *engine* de execução interpreta a descrição da composição para instanciar e gerenciar os processos que estão em execução. A proposta tenta ser transparente tanto para os serviços de composição solicitados como para o serviço de registro OSGi. São definidos os chamados "*virtual bundles*" que especificam um serviço composto utilizando a *Business Process Execution Language* (BPEL) (BPEL, 2011) e a BPEL *Engine* para composição OSGi. A BPEL *Engine* (servidor de aplicação responsável por executar as composições BPEL) é inserida na arquitetura com um conjunto de serviços OSGi.

Como ilustrado na Figura 3.2, os componentes de execução virtual (*virtual bundles*), são componentes (*bundles*) que não oferecem propriamente a execução de serviço, mas sim a especificação de um serviço composto. Um "*virtual bundle*", assim como qualquer *bundle*, é empacotado em um arquivo Java (JAR) que inclui: (i) um arquivo com a definição BPEL do serviço composto; (ii) um *Bundle Activator* responsável por registrar o serviço composto e notificar o BPEL *Engine*. Para isso, a sequência de eventos segue como: o "*Bundle Activator*" obtém a localização da BPEL *Engine* (1) e notifica a BPEL *Engine* sobre sua existência, "*AddBPELProcess*" (2). Quando a BPEL *Engine* é notificada sobre um novo serviço BPEL

OSGi (3), é criado um componente de execução para um "*virtual bundle*" que registra o serviço composto no *framework* OSGi (4). O componente de execução é um processo BPEL responsável por gerenciar um serviço composto quando instanciado.

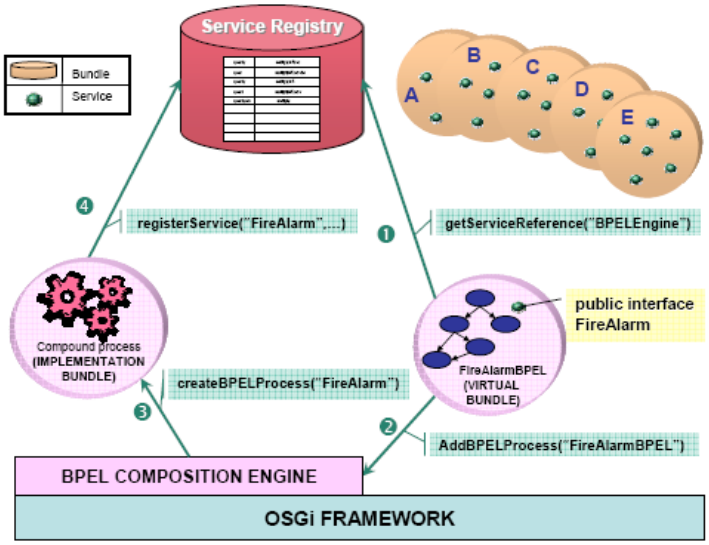

Figura 3.2 Registrando uma composição de serviços OSGi (REDONDO, 2007).

A proposta é implementada considerando como exemplo um cenário de AmI, em que são instalados quatro componentes de execução (B, C, D, E) que fornecem os serviços para definir a lógica de segurança (Figura 3.3). Para o exemplo, é descrito um código BPEL para o serviço "*trigAlarm*" na interface "*FireAlarm*", da seguinte forma:

```
<process name="trigAlarn">
 <partnerLinks>
  <partnerLink name="actvAlarm"/>
  <partnerLink name="unlockDoors"/>
    …
 </partnerLinks>
 <variables>
    <variable name="BOutput"/>
    <variable name="DInput"/>
    ...
 </variables>
  ...
```

```
...
<sequence>
 ...
 <assign><copy>
    <from variable="BOutput"/>
    <to variable="DInput"/>
 </copy></assign>
 <assign><copy>
 ...
 </copy></assign>
 ...
```

Figura 3.3 Código BPEL para um serviço (REDONDO, 2007).

Na primeira parte da especificação, são detalhados os serviços atômicos OSGi, (*actvAlarm*) e os nomes das variáveis de entrada e saída desses serviços. Na outra parte da especificação são definidas as associações entre as variáveis, em que a saída de um serviço pode ser a entrada de outro. Já, a sequência das instanciações é ilustrada na Figura 3.4.

```
...
<flow>
  <sequence>
      <invoke name="invokeB" partnerLink="actvAlarm"/>
      <receive name="rcvinvokeB" partnerLink="actvAlarm"
               variable="BOutput"/>
  </sequence>
  <sequence>
      <invoke name="invokeD" partnerLink="unlockDoors"
               inputVariable="BOutput"/>
      <receive name="rcvinvokeD" partnerLink="unlockDoors"
               variable="DOutput"/>
  </sequence>
  ...
</flow>
...
```

Figura 3.4 Sequência de instanciações (REDONDO, 2007).

Quando um componente de execução A precisa usar o serviço "*FireAlarm*", ele obtém a referência para o componente de execução do "*Service Registry*" através do comando "*getServiceReference*" e utiliza o método "*trigAlarm*" da interface "*FireAlarm*", conforme ilustra a Figura 3.5.

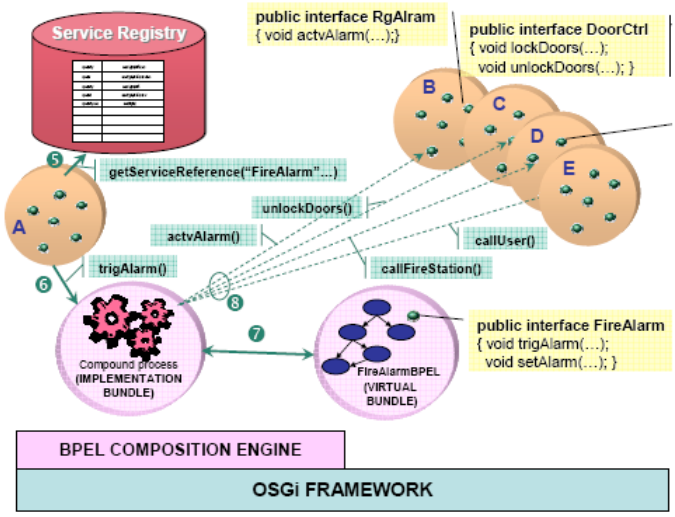

Figura 3.5 Instanciando um serviço OSGi composto (REDONDO, 2007).

Na instanciação de serviços o componente de execução lê a informação BPEL e solicita o serviço apropriado (componentes B, C, D e E) na ordem especificada pela descrição BPEL. Portanto, do ponto de vista de um componente de execução solicitante, não há diferença entre utilizar um serviço OSGi simples ou composto, pois ambos são considerados serviços no *framework* OSGi.

DAFNE (STAVROULAKI, 2006), é uma proposta de *framework* de gerenciamento distribuído para serviços reconfiguráveis em AmIs. Foi desenvolvido com o objetivo de permitir o gerenciamento de redes heterogêneas, serviços e dispositivos computacionais dentro do contexto de AmIs. O modelo conceitual do *framework* DAFNE é caracterizado por abstrair a heterogeneidade da infraestrutura de rede no AmI. Assim, no contexto dos AmIs, são identificadas e especificadas as seguintes entidades: (i) Provedor de infraestrutura de rede no ambiente: responsável por instalar e configurar a infraestrutura do AmI; (ii) Gerador de componentes de percepção: responsável por instalar e configurar componentes de percepção, podendo ter entrada de áudio, de imagem, de movimento e indicadores de temperatura; (iii) Gerador de modelo: responsável por combinar contexto disponível fornecido pelos componentes de percepção, identificando os mais altos níveis dos estados de contexto; (iv) Provedor de serviço/aplicação: responsável pelo desenvolvimento das aplicações do AmI. Esse componente requer acesso aos modelos da aplicação-alvo, criados pelo "Gerador de modelo".

A arquitetura definida para o *framework* DAFNE permite que haja interconexões e troca de serviços personalizados entre as entidades. Na Figura 3.6, é apresentada uma visão alto nível das interações entre as principais entidades com o *framework* DAFNE. Essa interação é realizada em várias etapas no desenvolvimento do AmI.

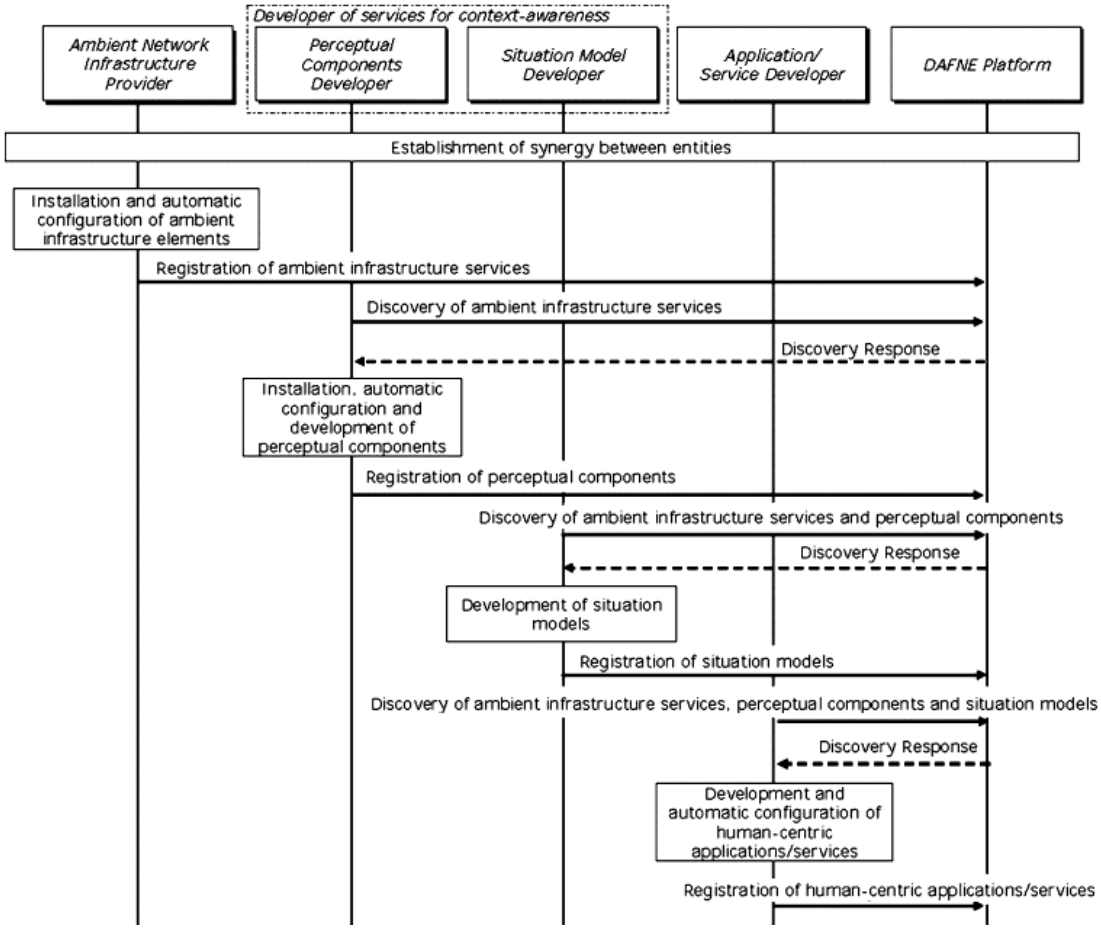

Figura 3.6 Visão alto nível das interações com o DAFNE (STAVROULAKI, 2006).

Na interação existente, a infraestrutura de rede instala e configura os elementos relacionados com a infraestrutura do ambiente, sendo registrados no DAFNE. Os componentes de percepção descobrem, através do *framework* DAFNE, os elementos de infraestrutura disponíveis, instalando e configurando os componentes de percepção. O gerador de modelo descobre os elementos de infraestrutura e os componentes de percepção que foram instalados. Por fim, o gerador de aplicações descobre os modelos de contexto disponíveis e define serviços conscientes ao contexto, que também são registrados no *framework*. Em resumo, o DAFNE provê cada uma das entidades, identificando e permitindo que sejam registrados os seus serviços. O DAFNE também oferece mecanismos para disponibilizar elementos de infraestrutura, componentes de percepção, modelos de contexto e aplicações. A arquitetura do *framework* é orientada a serviços, sendo capaz de atender às seguintes funcionalidades: (i) permitir a integração de componentes e subsistemas para AmIs, que inclui

redes heterogêneas e dispositivos computacionais, bem como o gerenciamento desses ambientes; (ii) disponibilizar serviços sensíveis ao contexto, personalizados, e interfaces de usuários; (iii) permitir o desenvolvimento das aplicações para AmIs centradas no usuário. E, por isso, foi definida a arquitetura genérica (ilustrada na Figura 3.7), com a intenção de atender às três funcionalidades descritas anteriormente.

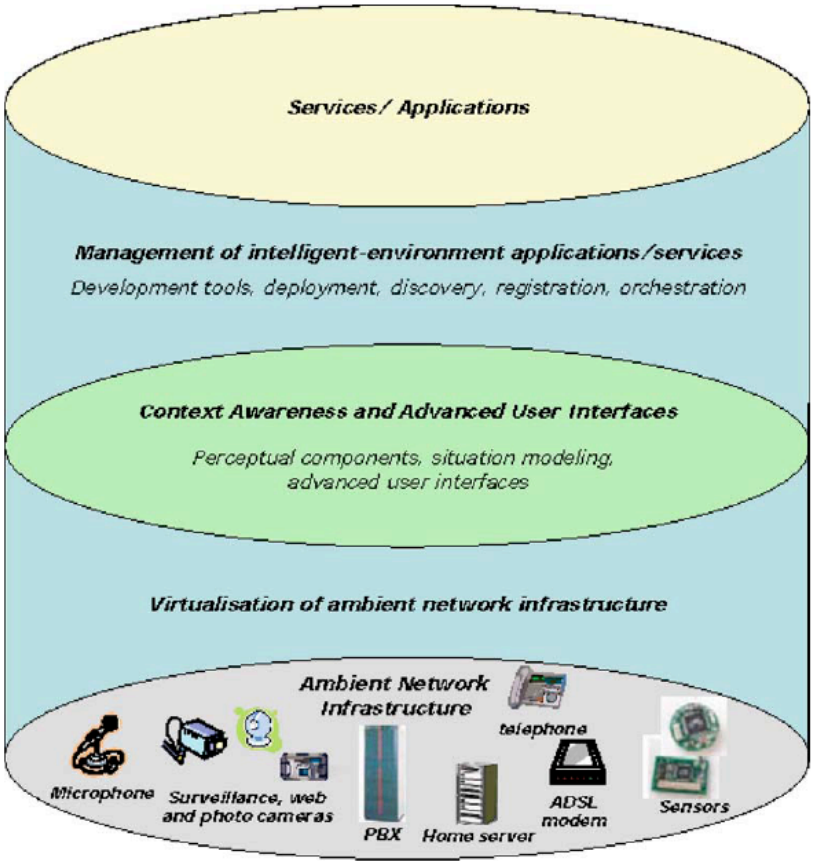

Figura 3.7 Arquitetura DAFNE (STAVROULAKI, 2006).

Outra proposta para gerenciamento de serviços é apresentada por (INDULSKA, 2001), cuja abordagem inclui, também, o gerenciamento de contexto, capturando as descrições dos usuários, dos dispositivos e das aplicações. Como pode ser observado na Figura 3.8, a arquitetura da proposta está dividida em três camadas: (i) Camada de Coordenação, que consiste em um gerenciador que executa *scripts* escritos em *Mobile Enterprise Architecture Description Language* (MEADL); (ii) Gerenciador Dedicado, que consiste nos gerenciadores

de contexto, de adaptação e de políticas, presentes do ambiente pervasivo; (iii) Camada de Serviço Distribuído, que abrange os serviços de notificação, segurança e protocolos de rede.

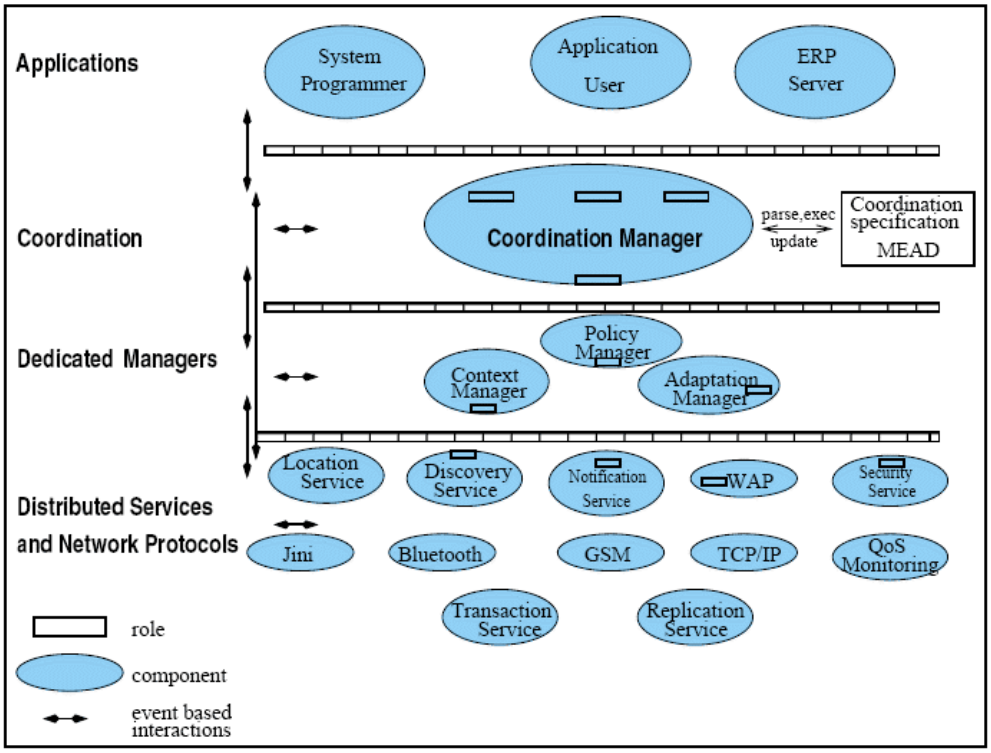

Figura 3.8 Arquitetura de gerenciamento (INDULSKA, 2001).

A Camada de Coordenação está inserida na arquitetura com o intuito de aumentar a produtividade do programador e proporcionar um desenvolvimento mais rápido e dinâmico dentro de sistemas complexos, sendo a principal motivação que precede o desenvolvimento das linguagens e modelos de coordenação. Os métodos baseados em coordenação oferecem uma separação bem definida entre componentes de *software* e suas interações dentro da organização. Nesse ponto, existe a *Architecture Description Language* (ADL) que modela notações para o desenvolvimento baseado em arquiteturas. A Camada de Coordenação é o centro da arquitetura, e está focada nos conceitos necessários para gerenciar as implicações da mobilidade para as atividades. A Coordenação consiste em especificar com a MEADL a interação entre funcionalidades e executar os *scripts* de MEADL através do gerenciador de coordenação. Os *scripts* MEADL são transformados em XML e, assim, o gerenciador coordena os eventos (enviar/receber) com base na especificação XML gerada.

A Camada do Gerenciador Dedicado é composta por três outros gerenciadores, que oferecem as três principais funcionalidades da arquitetura. O primeiro é o "*Context Manager*" (CM) que é sensível ao contexto do ambiente e se comunica com o "*Adaptation Manager*" (AM), o qual realiza adaptações de acordo com o contexto da informação. O "*Policy Manager*" (PM) assegura que as funcionalidades envolvidas no processo serão suportadas.

A Camada de Serviço é definida pelas características dinâmicas que estão presentes nos ambientes, como a inclusão de novos serviços e de novos protocolos de comunicação. Essa camada oferece a notificação e a descoberta de serviços, que informa o restante dos componentes da arquitetura, bem como o seu ambiente de execução. Outro serviço dessa camada é a interface universal para as camadas superiores, a qual ativa o uso dos protocolos de comunicação, tais como: *Simple Mail Transfer Protocol* (SMTP), *HyperText Transfer Protocol* (HTTP), *Transmission Control Protocol* (TCP) / *Internet Protocol* (IP) ou protocolos seguros.

Para elaboração da arquitetura, foram considerados alguns elementos essenciais para aplicações de computação pervasiva. O uso de componentes baseados em funções e a MEADL fornecem suporte *plug and play* e o serviço de notificação habilita um amplo formato de integração de componentes, permitindo reconfigurações da arquitetura.

3.2 ADAPTAÇÃO E CUSTOMIZAÇÃO DE CENÁRIOS

Dos diversos recursos e atrativos oferecidos em AmIs está presente, também, o acesso às informações pelos dispositivos com conexões à redes. Com isso, surgem algumas propostas com a finalidade de tratar os desafios de segurança e acesso aos dados em AmIs no domínio predial/residencial. Uma das alternativas para se tratar da segurança em AmIs é o gerenciamento de perfil, que busca aumentar a segurança no ambiente através da identificação de seus usuários, suas necessidades e ações que esses usuários podem realizar. Nesse sentido,

é proposto em (ZIEGLER, 2005), um *middleware* capaz de integrar bases de dados de perfis diversos com mecanismos genéricos de autenticação que permitem o gerenciamento no *framework* OSGi. Os perfis de usuários são utilizados para capturar informações sobre preferências e habilidades básicas destes, enquanto que os perfis de dispositivos fornecem as principais características e descrevem as atuais situações de cada dispositivo. Existem também as definições dos tipos de acesso seguro em cada ambiente, contendo informações confidenciais de identificação que devem ser preservadas. Por esse motivo, a proposta do *middleware* possui uma arquitetura de segurança para AmIs com o propósito de garantir que somente usuários autorizados executem operações em perfis.

Para a elaboração da arquitetura, são considerados alguns cenários, tais como: considerar que um usuário, ao chegar em casa, queira abrir a porta da frente e, em função disso, a autenticação é necessária. Assim, o usuário passa por um processo de autenticação biométrica. Após entrar no ambiente, o usuário quer as últimas notícias e liga a TV. Entretanto, as aplicações da TV necessitam realizar uma leitura no servidor de perfis, em que é identificado o perfil deste usuário. Com isso, uma lista de canais favoritos é exibida, não sendo necessário considerar como uma aplicação crítica de segurança o fato de o usuário querer “assistir à TV”. Já para aqueles usuários que desejam realizar tarefas críticas de segurança, como editar dados privados de configuração do ambiente ou executar transações financeiras através de *websites* bancários, é necessária uma segurança mais reforçada. Existem diferentes níveis de segurança para diferentes serviços ou usuários. A proposta de (ZIEGLER, 2005) define uma arquitetura distribuída, em que o componente de comunicação pode utilizar o protocolo TCP/IP sobre tecnologias com *Wireless Local Area Network* (WLAN - IEEE 802.11) e *Bluetooth* (IEEE 802.15) ou, ainda, usando a comunicação sobre a rede elétrica (*Power Line Communication* - PLC) e *European Installation Bus* (EIB). A fim de garantir a segurança nos sistemas distribuídos é utilizada uma comunicação baseada em

Secure Sockets Layer (SSL) / *Transport Layer Security* (TLS) com certificados X.509. Uma visão geral da arquitetura proposta pode ser observada na Figura 3.9.

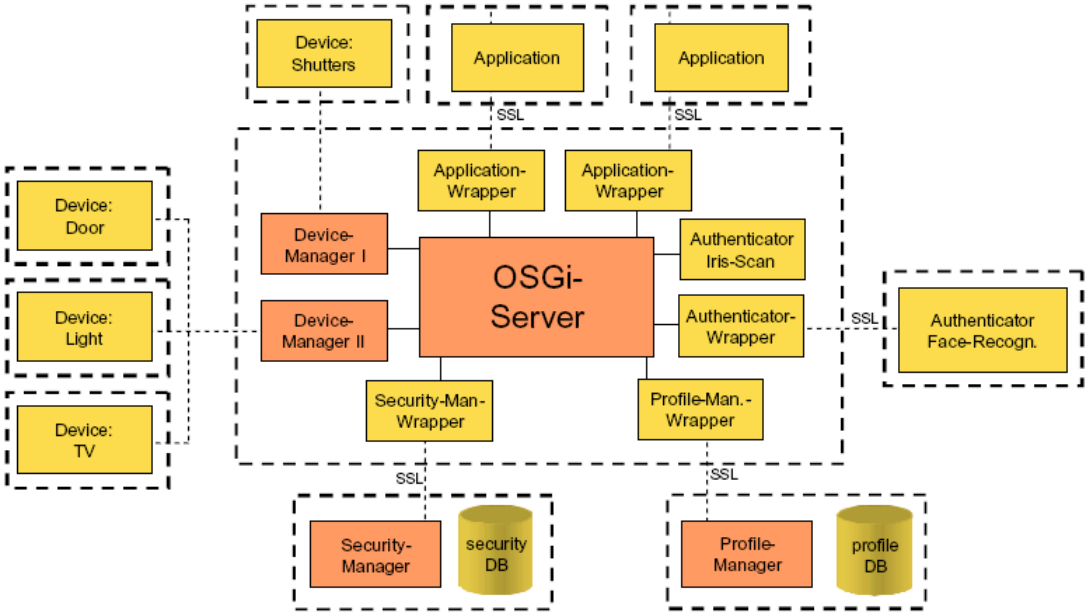

Figura 3.9 Visão geral da arquitetura (ZIEGLER, 2005).

Todos os componentes da arquitetura precisam ser registrados, executando o servidor principal, chamado de "*OSGi-Server*". Em linhas gerais, o sistema possui cinco componentes principais que interagem com um conjunto de aplicações, definidos como:

- *Applications*: que fornecem serviços executáveis aos usuários, tanto em computadores pessoais, com um sistema de reconhecimento de voz, quanto em plataformas móveis, como PDAs e telefones celulares;
- *Security Manager*: proporciona mecanismos de segurança em todo o sistema. Para uma aplicação acessar um serviço, ela tem que registrar uma entrada que transfere direitos de acesso seguro para uma aplicação em nome de um usuário específico. Uma entrada é assinada pelo "*Security Manager*" usando um esquema de segurança para evitar falsificação;
- *Profile Manager*: todos os dados de perfil são armazenados em um banco de dados, o qual somente é acessado através do "*Profile Manager*" e que apenas permite ler ou escrever os dados se a aplicação solicitante tiver a capacidade de

oferecer direitos de acesso. Para aumentar a segurança dos dados, os perfis de entrada na base de dados são criptografados também pelo "*Profile Manager*";

- *Device Manager*: responsável por monitorar todos os serviços que são fornecidos pelos dispositivos dentro do ambiente, tais como: portas, luzes, TVs e *Compact Disc* (CD) *Players*. Um exemplo de serviço poderia ser "abrir a porta da frente", "trocar o canal da TV ", "ajustar a intensidade da luz". Assim como ocorre com o "*Profile Manager*", o "*Device Manager*" somente permite acesso aos serviços após receber uma entrada válida para direitos de acesso de uma aplicação;
- *Authenticators*: no ambiente, pode existir um ou mais pontos de autenticação e, dessa forma, os autenticadores são os responsáveis pelo serviço de identificação para diversos níveis de segurança, que variam desde uma simples verificação de senhas até um reconhecimento de face;
- *OSGi Server*: o servidor principal é baseado em OSGi. Todos os componentes do sistema devem ser registrados no "*OSGi Server*" para fornecer ou utilizar algum serviço.

O sistema implementa diferentes métodos de autenticação, dependendo dos níveis de segurança que são solicitados. Esses métodos são: autenticações por senha, *smartcard*, reconhecimento de face, impressão digital e escaneamento de retina, sendo que para cada método existe um escore de pontos associados, que variam de 25 até 100. O método de senha possui 25 pontos, o *smartcard* possui 45 pontos, reconhecimento de face possui 70 pontos, impressão digital possui 80 pontos e escaneamento da retina possui 100 pontos, ou seja, variações do método menos seguro (25 pontos) para o mais seguro (100 pontos). Com isso, dependendo do nível de segurança exigido em certo momento ou em alguma aplicação, pode haver uma combinação entre métodos de autenticação para atender a um requisito de

segurança. Por exemplo: para um determinado serviço a ser executado, o nível de segurança exigido é médio (ou seja, 50 pontos), um usuário pode ser autenticado através da combinação do método de senha (25 pontos) com o método de *smartcard* (45) pontos, resultando em 70 pontos e atingindo o nível mínimo de segurança exigido para aquele serviço.

O usuário poderia utilizar, ainda, qualquer outro método de autenticação com pontuação de segurança maior do que a exigida, por exemplo: o reconhecimento de face (70 pontos), a impressão digital (80) pontos, ou o escaneamento da retina (100 pontos). Uma vez que o usuário foi identificado, ele tem um perfil para direitos de acesso, do tipo (*accessMode* = R, W, X) que diz respeito às ações que este usuário pode realizar com aquele perfil, nesse caso, R = *Read*, W = *Write* e X = *Execute*.

Um exemplo de solicitação para o perfil de entrada pode ser observado na Figura 3.10, que oferece um exemplo aplicável com similaridade em diversas situações e serviços, como "abrir portas", "escutar música" e "assistir à TV".

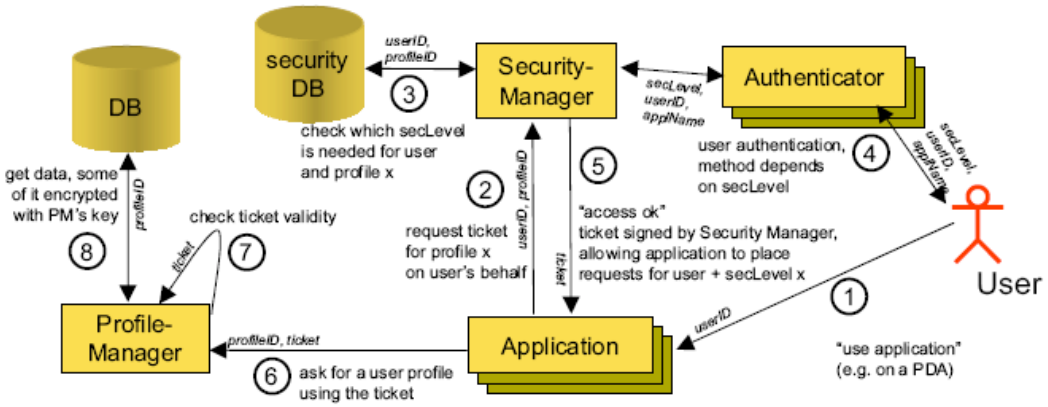

Figura 3.10 Solicitando um perfil de entrada (ZIEGLER, 2005).

O exemplo considera que o usuário deseja ler o estado atual de sua conta bancária. Devido ao nível de segurança exigido, é possível aplicar criptografias e autenticações para diferentes canais de comunicação, acompanhando os seguintes passos: (1) o usuário abre uma aplicação em seu PDA e solicita o estado atual da sua conta bancária; (2) quando a aplicação não possui uma entrada válida para o usuário atual e perfil, é solicitada uma entrada ao "*Security Manager*" que reside no "*OSGi Server*"; (3) o "*Security Manager*" verifica qual

nível de segurança é necessário para acesso de leitura daquele *profileID* e *userID*; (4) dependendo do nível de segurança exigido, um autenticador confiável ou uma combinação de autenticadores são escolhidos pelo "*Security Manager*" e, então, é solicitada ao usuário a sua identificação; (5) se a autenticação for bem sucedida, o "*Security Manager*" fornece uma entrada que permite à aplicação localizar solicitações de serviços em nome do usuário; (6) com uma entrada correta para acesso, a aplicação solicita ao "*Profile Manager*" o estado da conta bancária para aquele perfil de entrada; (7) então, o "*Profile Manager*" verifica a validade de entrada; (8) se os passos anteriores tiverem sido realizados com sucesso, o "*Profile Manager*" lê a entrada do perfil solicitado, realiza, caso necessário, uma decodificação usando uma chave secreta do banco de dados e, finalmente, retorna os dados para a aplicação solicitante.

Os perfis de usuários também podem fazer parte do domínio de requsitos nos AmIs, como é abordado por (GROPPE & MUELLER, 2005), em que a proposta busca permitir a realização de customizações nos AmIs de acordo com as situações cotidianas, com as preferências dos usuários e com os recursos computacionais presentes nos dispositivos. Basicamente, em (GROPPE & MUELLER, 2005), é considerado o perfil como sendo uma coleção de dados estruturados que descrevem as propriedades de um objeto, que são requisitados por necessidades específicas, abrangendo dois aspectos: (i) a descrição do perfil e (ii) a computação para a criação e modificação das instâncias de perfil. O primeiro é utilizado para descrever perfis, incluindo a modelagem de objetos, a captura de propriedades, a especificação de vocabulário e o codificador de dados. Já para o segundo aspecto, a computação de perfil refere-se ao processo de especificar valores e gerar instâncias de perfis para os objetos, contando com três pontos importantes: a aquisição de dados, o processamento de dados e a avaliação dos perfis.

O *framework* proposto em (GROPPE & MUELLER, 2005), está concentrado no processamento de perfil e na avaliação para customizações de um AmI e, assim, assume-se que esse AmI integra múltiplos dispositivos móveis distribuídos, que podem ser, automaticamente, monitorados e controlados. A Figura 3.11 evidencia uma arquitetura com a ideia geral do sistema de customização.

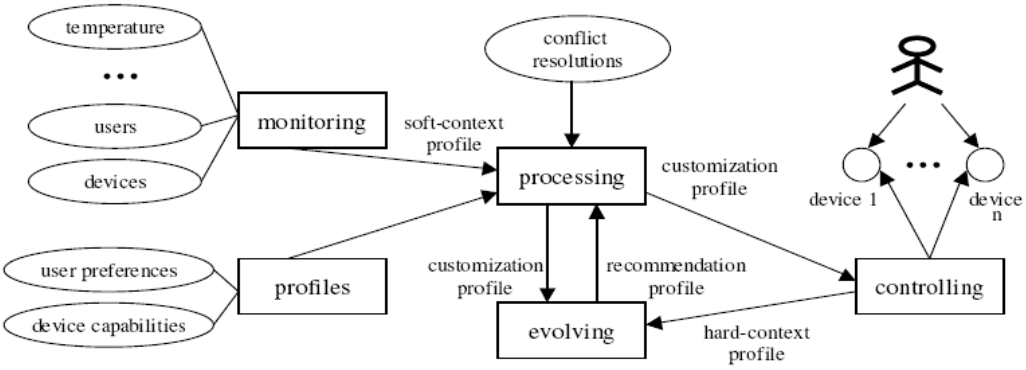

Figura 3.11 Arquitetura de customização em AmI (GROPPE & MUELLER, 2005).

Em linhas gerais, existem dois grupos de perfis: (i) pré-definidos e (ii) em demanda. O primeiro grupo é composto por dois outros tipos de perfis: "*user profiles*", para descrever as preferências e as características dos usuários e os "*device profiles*", que contêm as informações relacionadas com a capacidade dos dispositivos, tanto de *hardware* quanto de *software*. O segundo grupo, em demanda, é composto por outros quatro tipos de perfis: "s*oft-context profile*", que consiste da informação de um sistema de monitoração, incluindo temperatura, data, hora, identificação e localização de pessoas e dispositivos; "*hard-context profile*", que informa o atual estado dos dispositivos, e é capturado pelo controle do sistema; "*recommendation profile*", que inclui informação de uma preferência, e "*customization profile*", que dispõe das informações para customizar o ambiente preferido.

Nessa arquitetura, um conjunto pré-definido de perfis fornece informações sobre as preferências dos usuários e a capacidade dos dispositivos. O sistema de monitoramento captura as informações do ambiente, detecta usuários, dispositivos e gera um "*soft-context profile*". O sistema de controle utiliza um perfil de customização para acionar os dispositivos

de modo a estabelecer um ambiente customizado. Além disso, é criado um "*hard-context profile*" com o status dos dispositivos. Os usuários podem adaptar a customização, manualmente, caso ela não esteja de acordo com a preferência daquele usuário específico. A adaptação é capturada pelo sistema de controle e é refletida no "*hard-context profile*". Comparando o perfil de customização com o "*hard-context perfil*", o sistema de evolução pode filtrar as novas preferências e, automaticamente, atualizar os perfis dos usuários.

Além das propostas que visam à customização de diferentes cenários em residências, onde o AmI é privado, há também propostas que tratam da mesma questão para os ambientes públicos, como, por exemplo, estações de metrô, aeroportos e restaurantes, ou seja, a possibilidade de se ter acesso às informações pessoais e/ou privadas em ambientes de domínio público, como é o caso da proposta apresentada em (RÖCKER, 2006), através de um sistema chamado SPIROS. Nesse sistema, a principal contribuição está na possibilidade de se ter um espaço privado, dentro de espaços públicos, em que as informações que estão disponíveis aos usuários podem ser controladas de acordo com as preferências de cada um dos usuários do sistema. Esse mecanismo de adaptação e representação de informações é dado quando o indivíduo acessa o seu espaço privado, sendo automaticamente detectado e identificado para apresentação de seus dados customizados na tela de exibição pública.

O espaço privado dos usuários no SPIROS busca o conceito do mundo real, que é adaptado para o domínio virtual. O termo *espaço privado* refere-se a um ambiente em que todas as interações são visíveis apenas para o usuário que está interagindo naquele momento. Ou seja, outras pessoas não podem ver a interação que um determinado indivíduo está realizando. Uma ideia desse conceito pode ser observada na Figura 3.12a, que ilustra um usuário interagindo com seu espaço privado numa tela de exibição pública. O semicírculo em frente à tela de exibição apresenta o espaço pessoal do usuário, representando exatamente o espaço físico que é necessário para continuar livre de interrupções ou intrusões. Como esse

espaço não é invadido por outros usuários, todo o conteúdo pode ser exibido normalmente. As áreas com a cor preta exibidas na tela representam aplicações ou documentos que contêm informações confidenciais. Já as áreas de coloração cinza indicam aplicações com informações de domínio público. No momento em que outra pessoa entra no espaço privado de um usuário (Figura 3.12b), as informações privadas (áreas de coloração preta) que estavam sendo exibidas, são temporariamente canceladas, garantindo a privacidade naquele espaço.

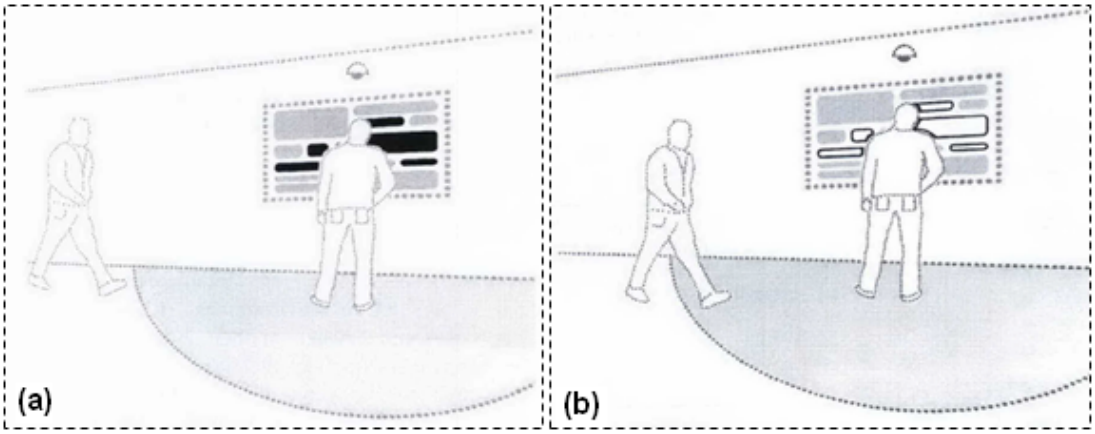

Figura 3.12 Espaço privado em domínio público (RÖCKER, 2006).

A preocupação com a privacidade das informações em AmIs também é observada na proposta de (HELAL & HAMMER, 2004), que propõem uma arquitetura, chamada UbiData, para acesso de dados ubíquos**,** projetada com a intenção de tratar alguns desafios originados pela proliferação de dispositivos e aplicações móveis. Dentre os desafios a serem tratados pela arquitetura, destacam-se: (i) acesso a dados em qualquer momento e em qualquer lugar; (ii) acesso a dados independentemente do dispositivo, que permite aos usuários alternar entre diferentes dispositivos móveis com diferentes capacidades de acesso; (iii) acesso e atualização de dados independentemente da aplicação, que possibilita aos usuários modificar documentos e arquivos.

Na arquitetura UbiData (ilustrada na Figura 3.13), são considerados clientes fixos e móveis que são conectados, através da Internet, a um servidor UbiData, o qual consiste de um *Mobile Data Server* (MDS) e um componente de *middleware* chamado *Fixed Mobile*

Environment Manager (Fixed-MEM ou F-MEM). Cada um dos dispositivos clientes habilitados com a Ubidata tem uma cópia do MEM chamada *Client*-MEM ou, simplesmente C-MEM, que se refere ao componente MEM tanto para clientes fixos quanto para clientes móveis. O C-MEM e o F-MEM são conectados à Internet ou à Intranet através de uma rede cabeada ou sem fio. Juntos, o C-MEM e o F-MEM eliminam a sincronização manual de dados entre os dispositivos fixos e móveis que são utilizados pelo usuário.

Por outro lado, o MDS consiste de um repositório de metadados, responsável por gerenciar informações sobre os usuários da UbiData, seus respectivos dados e os dispositivos com que os usuários gerenciam os seus próprios dados. Em geral, o repositório de metadados armazena para cada usuário um perfil, que contém informações sobre as preferências, as informações de autorização, as senhas e a lista de dispositivos dos usuários com informações de configuração e capacidade, bem como as informações relacionadas com os arquivos de dados que estão armazenados nos diversos dispositivos do usuário. Além dos perfis dos usuários, o repositório de metadados também contém as regras e os filtros usados pelo *Format-Independent Change Detection* (FCDP) para conversão de dados em diversos formatos, com o objetivo de oferecer interoperabilidade entre diferentes dispositivos e plataformas. Todos os metadados são representados e armazenados como dados no formato XML, sendo consultados e atualizados por meio do *Galax query processor*.

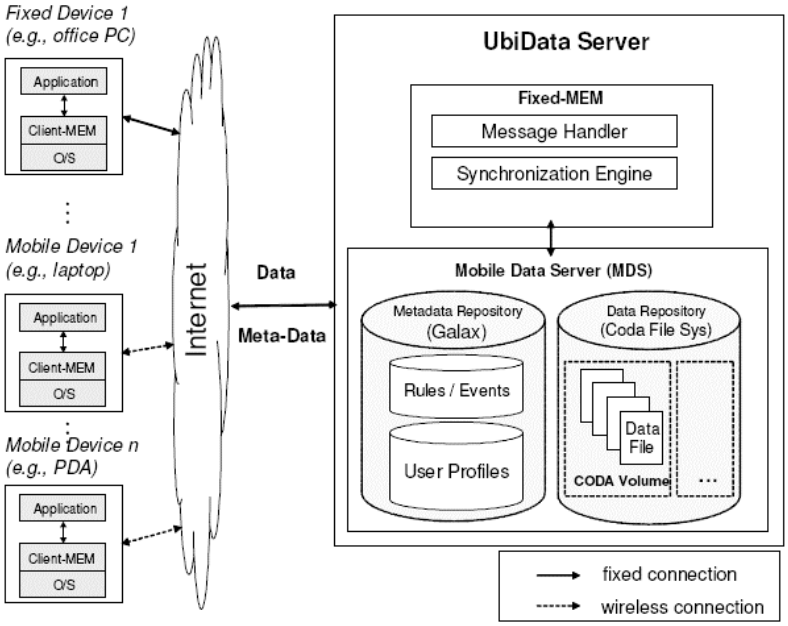

Figura 3.13 Arquitetura UbiData (HELAL & HAMMER, 2004).

As operações na arquitetura UbiData são suportadas pelo MEM da seguinte forma: em cada dispositivo móvel, o C-MEM monitora continuamente o padrão de acesso aos dados por parte dos usuários, com o objetivo de definir um conjunto de trabalho que será utilizado pelo F-MEM. A arquitetura UbiData foi implementada em um protótipo que está sendo utilizado no laboratório da Harris, na Universidade da Flórida. O protótipo consiste de um servidor UbiData e de diversos clientes com sincronização de arquivos de dados nestes clientes. O protótipo suporta, ainda, o gerenciamento básico dos perfis de usuários.

Além da privacidade das informações contidas em AmIs há, ainda, a preocupação em determinar o modo como a informação será apresentada ou acessada pelos usuários. Nos AmIs, há uma diversidade de dispositivos de automação e sistemas heterogêneos compondo esse ambiente, em que os requisitos podem sofrer alterações constantes e o ambiente, juntamente com as suas aplicações, deve estar preparado para responder a essa nova mudança de contexto.

Existem muitas propostas que tratam de adaptação de aplicações frente aos dispositivos nos quais são executadas e frente à variação de requisitos e troca de contextos (LEE, 2006; SOUZA & GARLAN, 2002). Um exemplo disso é o FlexXML, proposto por

(KAPLAN & LUNN, 2001), que consiste de uma extensão para XML/*eXtensible Stylesheet Language* (XSL) a fim de permitir que aplicações baseadas na *web* possam se adaptar às variações de comunicação e aos diferentes dispositivos de *hardware* (PDAs, telefones celulares, *Personal Computers* (PCs)) que acessam essas aplicações. Nesse caso, usando XML, um desenvolvedor cria uma especificação simples com conteúdo para uma aplicação *web* e diversos estilos XSL para apresentação dessa mesma aplicação. Por exemplo, uma especificação XSL poderia ser usada na criação de uma apresentação HTML para um determinado dispositivo, com uma exibição baseada em gráficos. Outro estilo XSL poderia criar uma nova apresentação usando *Wireless Markup Language* (WML) para um dispositivo com menos recurso computacional, baseada em texto. Com as informações dos dispositivos de cada usuário que pretende acessar a aplicação *web*, o *framework* FlexXML seleciona automaticamente o estilo XSL, aplica-o à especificação XML e transmite o conteúdo apropriado. Assim, os desenvolvedores não necessitam manter múltiplas versões de uma mesma aplicação. O FlexXML também permite que o usuário customize seu ambiente de visualização, podendo indicar os tipos de informações que ele espera da aplicação ou definir suas conexões de rede atuais, sendo utilizada essa informação para exibir um conteúdo que melhor se adapte ao dispositivo em questão. A Figura 3.14 alude a uma visão conceitual do FlexXML e sua relação com os serviços *web*. O *framework* comunica-se com os quatro seguintes componentes: um *proxy* de aplicação, um servidor *web proxy*, um *parser* XML e um processador XSL.

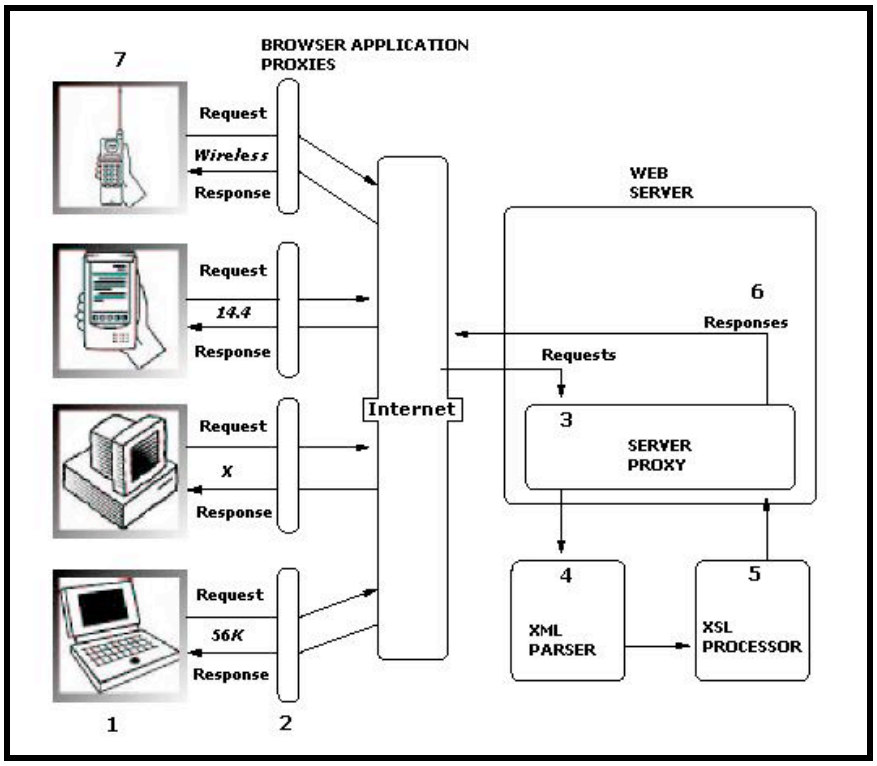

Figura 3.14 O framework FlexXML (KAPLAN & LUNN, 2001).

Como ilustrado na Figura 3.14, o *proxy* do servidor *web* situa-se entre o navegador do usuário e um servidor *web*. Seu objetivo é interceptar e processar os cabeçalhos HTTP que são adicionados pelo *proxy* de aplicação do navegador. O *proxy* do servidor *web* utiliza essa informação para selecionar um estilo XSL que melhor satisfaz as características do navegador do usuário. O *proxy* do servidor encaminha um documento XML ao XML *parser*, gerando uma representação de árvore do documento XML, enviando-a ao processador XSL. Usando o estilo XSL selecionado pelo *proxy*, o processador XSL percorre a árvore e gera a representação apropriada para o navegador, seguindo as regras de tradução dadas no estilo XSL. Por exemplo, um cabeçalho HTTP criado pelo FlexXML poderia indicar uma requisição para documentos somente de texto, enquanto outro cabeçalho poderia indicar uma solicitação para documentos gráficos. Usando essa informação, o *proxy* do servidor *web* seleciona, automaticamente, um estilo XSL que melhor se enquadra na transformação de um documento XSL para uma representação desejada.

Outra proposta de adaptação é apresentada em (DI NITTO, 2003), através de uma infraestrutura de acesso universal e adaptação de conteúdos baseados na *web*, chamada *@Terminals Adaptation System* (@TAS). Nesse caso, a infraestrutura atua como um mediador entre terminais de usuários, provedores de serviços e seleção de conteúdos. Na Figura 3.15, é caracterizada uma visão geral dessa infraestrutura e de seus principais componentes: *Terminal Manager*, responsável por realizar a comunicação física, reconhecendo os terminais e capturando o contexto de entrega correspondente; *Session Manager*, responsável por gerenciar a interação entre os usuários e os serviços, rastreando as sessões de navegação. Esse componente também é responsável por gerenciar a troca de terminais durante a sessão de navegação, permitindo aos usuários suspender a sessão e retomá-la em um terminal diferente; *Service Adapter*, cuja principal atribuição é executar a ação de adaptação, selecionando os filtros disponíveis para adaptar o conteúdo ao contexto de entrega atual; e o componente *Service Integrator*, que implementa uma interface pública na qual os provedores de serviços podem utilizá-la para acessar informações atualizadas do contexto de entrega, da sessão de navegação e dos dados sobre os usuários.

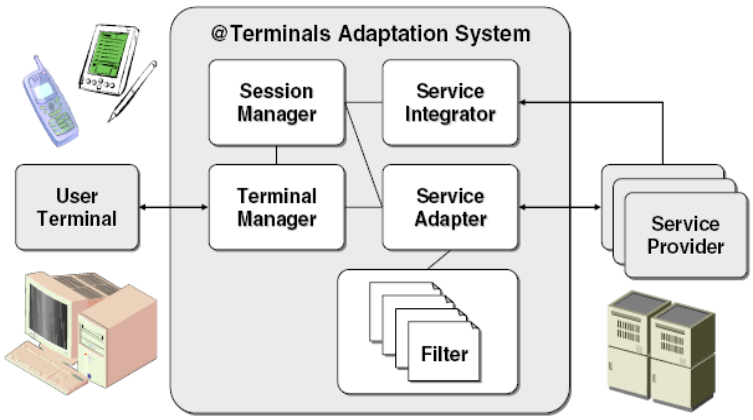

Figura 3.15 A Arquitetura @TAS (DI NITTO, 2003).

A adaptação de conteúdo realizada pelo @TAS é dinâmica: a decisão sobre o modo e o conteúdo que deve ser adaptado para a entrega é obtida somente quando o conteúdo é solicitado, pois é apenas nesse momento que se identifica quem originou tal solicitação. A

infraestrutura pode realizar adaptações genéricas em conteúdos da *web* como, por exemplo, redimensionamento de imagem para possibilitar a sua exibição em vários tamanhos de telas e eliminação de um tipo específico de formatos não suportados. A adaptação de conteúdo é o resultado da composição e execução de filtros. Os filtros são tratados pelo @TAS como uma caixa preta que implementa uma interface descrita por seu perfil. Este caracteriza-se por descrições XML que especificam o tipo de transformação que o filtro pode realizar, o tipo de entrada e saída e as condições de ativação.

Tanto na proposta de (KAPLAN & LUNN, 2001) quanto na de (DI NITTO, 2003), a customização das informações a serem apresentadas ao usuário e a adaptação da interface gráfica de interação podem ser ampliadas, no sentido de abranger outros sistemas de interação, como, por exemplo, a utilização de TVs e STBs em vez de PCs ou PDAs.

3.3 SISTEMAS DE INTERAÇÃO MULTIMODAL

A crescente busca pela interação entre homens e máquinas tem permitido o desenvolvimento de novas aplicações compostas por mais de uma técnica de interação para a realização de uma mesma tarefa. Nesse sentido, existem diversas abordagens que pretendem aumentar as possibilidades de os usuários interagirem com os AmIs através da multimodalidade (TSOURAKIS, 2006). Comumente, encontram-se na área de AmIs propostas como a de pesquisadores da Universidade da Flórida (MANN & HELLAL, 2002), que buscam, através da criação de soluções para dispositivos móveis, aumentar a qualidade de vida e reduzir a dependência de pessoas idosas ou com necessidades especiais em AmIs. A proposta de (MANN & HELAL, 2002) consiste na utilização de *smartphones* como mecanismo para oferecer suporte aos idosos em AmI. A ideia é transformar os *smartphones* em uma espécie de "varinha mágica", que oferece, entre outras coisas, funcionalidades de um controle remoto universal para os dispositivos de automação presentes no AmI.

A utilização dos *smartphones* em AmIs surge como uma forma de oferecer diversos serviços aos idosos, sem que estes precisem se locomover pelo ambiente para realizar algumas ações, ou solicitar a realização de uma tarefa a outra pessoa presente no mesmo local. Por exemplo: com um *smartphone*, o idoso poderia ligar ou desligar uma lâmpada, aumentar ou reduzir a temperatura do ar condicionado, verificar o estado atual das portas e janelas de sua casa (como uma questão de segurança patrimonial). Poderia, ainda, trocar o canal da TV ou ouvir uma música em especial, tudo remotamente. Assim, a arquitetura proposta foi projetada em torno de algumas tecnologias, ilustradas na Figura 3.16, e que são tratadas como a conectividade para o sistema de AmIs que utilizam *smartphones*.

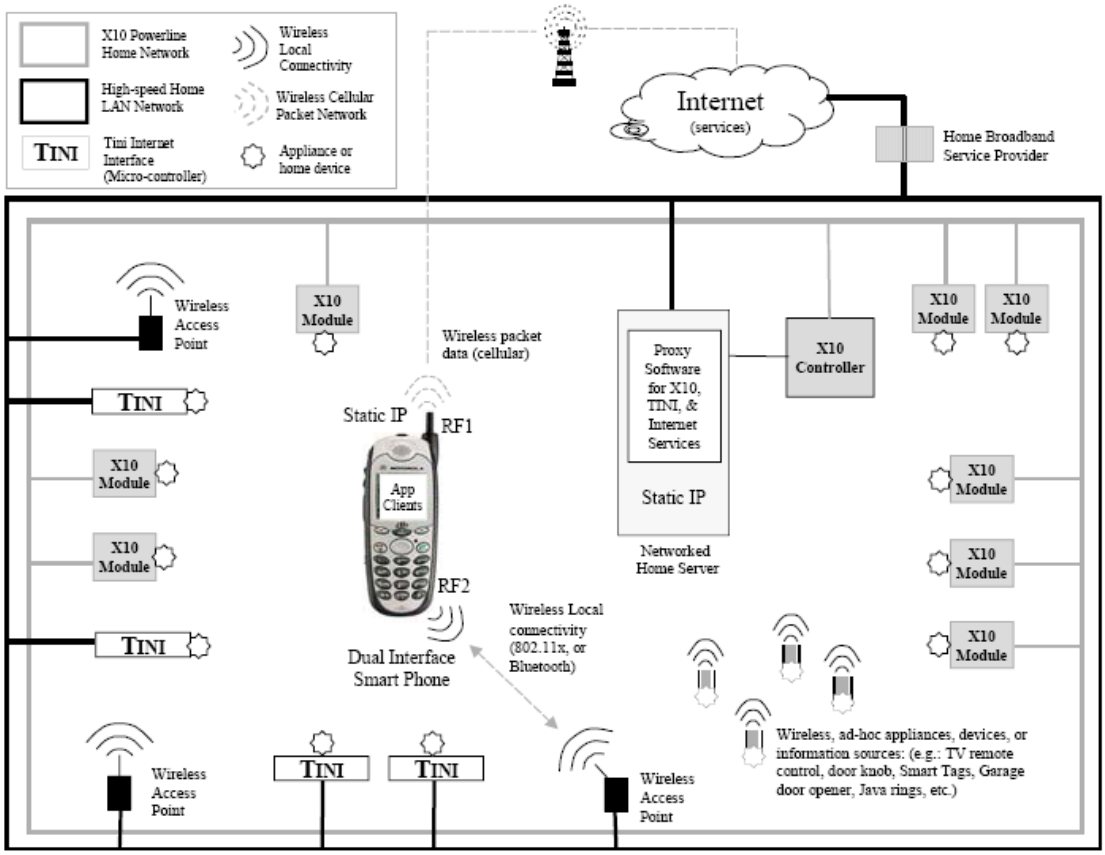

Figura 3.16 Conectividade do sistema (MANN & HELAL, 2002).

Para existir a conectividade do sistema, é indispensável a integração entre os componentes da arquitetura, o que envolve desde a necessidade de se ter um *smartphone* com suporte à linguagem Java, até controladores e *softwares* X10 - que é um protocolo de comunicação para aplicações de automação predial/residencial e que utiliza a rede elétrica

como meio de comunicação (O'REILLY, 2008). Outra característica da proposta é que ela se concentra em torno do servidor residencial e do *smartphone*, sendo direcionada a eventos, em que as ações são realizadas em função dos acontecimentos e das condições pré-definidas. As regras de Evento/Condição/Ação (ECA) são especificadas e armazenadas em uma base de dados para regras específicas de ECA. Quando ocorre algum evento, uma *engine* ECA avalia as regras na base de dados e realiza uma ação apropriada, ou seja, todo o gerenciamento previsto para o ambiente é feito em tempo de projeto, limitando a interação entre usuário e AmI e a inserção dinâmica de novos serviços ou dispositivos no ambiente, uma vez que a aplicação de interação e as regras ECA são estabelecidas *offline*.

Uma solução semelhante é proposta pela empresa Homesystems (HOMESYSTEMS, 2010), que utiliza as plataformas computacionais móveis iPod e iPhone como um controle remoto universal integrado aos seus próprios dispositivos de automação residencial. Da mesma forma como ocorre na proposta de (MANN & HELAL, 2002), a abordagem da Homesystems permite ao usuário, por exemplo, ligar as lâmpadas, trocar os canais da TV, fechar as persianas, ligar e ajustar o aparelho de ar condicionado. A diferença básica entre as propostas está em que uma utiliza os módulos X10 para a automação residencial, enquanto a outra desenvolve seus próprios módulos.

Outra abordagem de interação faz uso da realidade virtual como forma de facilitar a vida das pessoas em cenários de AmI, como é o caso da AR Phone (ASSAD, 2003), uma proposta de pesquisadores da Universidade de Sidnei, que busca a integração entre a realidade aumentada e a computação móvel. Além disso, em AmIs são encontrados cada vez mais sistemas computacionais que tratam tanto do gerenciamento de funcionalidades quanto da configuração de diferentes cenários através do comando de voz (GÁRATE, 2005). Esse mecanismo de interação permite, por exemplo, que o usuário solicite alguma tarefa do ambiente sem, necessariamente, ter que pressionar alguma tecla para realizar a mesma função.

Normalmente, em AmIs, são encontradas alternativas de interação multimodal, as quais possibilitam ao usuário optar por uma ou outra forma de interação que mais lhe agrada. Eventualmente, a interação com o ambiente pode ser realizada através da combinação entre duas ou mais modalidades.

Um mecanismo de interação natural entre usuários e AmIs pode conter os mais variados recursos: o usuário ligar o condicionador de ar simplesmente chamando pelo nome do dispositivo e, em seguida, indicando a tarefa a ser realizada, como, por exemplo: "*Ar Condicionado...*", "*...ligar*". Da mesma forma, nos serviços que estão relacionados ao entretenimento, há o mesmo princípio básico de funcionamento: o usuário poderia querer ouvir alguma música e solicitar ao ambiente uma listagem dos arquivos de áudio disponíveis no sistema e, então, ouvi-los. Outras aplicações por comando de voz poderiam considerar o fato de o usuário querer ler seus e-mails ou verificar se recebeu novas mensagens eletrônicas sem ter que interagir com o computador através de mouse e teclado: bastaria, por exemplo, solicitar ao ambiente que projetasse suas mensagens na parede da sala. Em relação às questões de segurança, o usuário também poderia solicitar ao ambiente a projeção de vídeo das câmeras de monitoramento distribuídas fisicamente no AmI. É com base em aplicações desse segmento que são observados esforços e propostas de trabalho envolvendo reconhecimento e interação por voz entre usuários e AmIs (SHI *et al*., 2003), (KLUS & RAUSCH, 2006).

Na visão de (GÁRATE, 2005), um AmI poderia reconhecer os usuários e ser sensível à presença humana, tendo um comportamento baseado no conhecimento do ambiente, dos moradores e das atividades realizadas diariamente. Mecanismos para oferecer entretenimento, segurança e conforto poderiam, ainda, melhorar a qualidade de vida das pessoas que vivem no ambiente, sendo que um dos focos em AmIs é a interação mais natural do usuário com o local que o circunda. Com base nesse princípio é que a empresa Fagor (FAGOR, 2007) vem

trabalhando no desenvolvimento de uma rede de comunicação na qual sensores, atuadores, sistemas de segurança e sistemas de aquecimento são conectados e gerenciados por um controlador central chamado *Maior-Domo*.

No trabalho proposto em (GÁRATE, 2005), é apresentada uma aplicação de AmI, que demonstra e construção de uma cozinha, em que os usuários podem comandar o local falando com o ambiente de maneira natural. Dentre as atividades que o usuário pode comandar pela voz, destacam-se a leitura de *e-mails*, a programação da máquina de lavar roupas, a ativação da máquina de lavar louças, a audição de músicas e a visualização de vídeos e fotos. O objetivo principal da proposta consiste no diálogo entre usuário e sua residência, em que o controlador, chamado *Maior-Domo*, tem uma representação humana (um avatar exibido em uma TV) com quem o usuário pode interagir. Quando o usuário fala, o *Maior-Domo* extrai os comandos e controla os dispositivos da residência. As informações ou eventos dos dispositivos podem ser transmitidos ao usuário através de locuções. Na interação existente, cada usuário possui um microfone sem fio preso à camisa, que permite a captura de sua voz e todos os sons ao seu redor. O som capturado é, então, emitido para um computador onde são executados *softwares* para reconhecimento de voz e conversores de texto para fala - desenvolvidos com a utilização de recursos como: *Voice Extensible Markup Language* (VXML), JAVA, *Java Server Pages* (JSP) e *JavaBeans* - além de uma aplicação para comandos e controle da residência.

De um modo geral, os sistemas de interação se propõem a facilitar e viabilizar a comunicação homem-máquina, e essa interação pode ser dada em diversos espaços físicos em que os usuários se encontram, seja na residência, no carro ou em algum terminal de acesso público. Essa diversidade de ambientes com interação é o foco do projeto Embassi (EMBASSI, 2008), financiado pelo Ministério da Educação e Pesquisa da Alemanha, cujo objetivo é o desenvolvimento de novos paradigmas e arquiteturas para interação homem-

máquina com dispositivos eletrônicos. A principal característica do projeto Embassi está na interação multimodal, com a proposta de uma arquitetura que forneça suporte inteligente e diversas interfaces de acesso ao sistema dentro de um *framework* desenvolvido para essa finalidade. Na proposta (EMBASSI, 2008), há uma mudança de conceitos, em que sistemas orientados a funções e centrados nos usuários são substituídos por sistemas com interação baseada no objetivo a ser atingido, em que são utilizadas técnicas de inteligência artificial (IA) para aquisição de conhecimento sobre esses sistemas. A ideia é que o usuário consiga interagir com os dispositivos de automação através de comandos naturais ou intuitivos.

3.4 TV DIGITAL INTERATIVA

O desenvolvimento de aplicações para TV digital está sendo bastante explorado no Brasil e, principalmente, em países onde a TV interativa já está presente há mais tempo (DOLAN, 2001), (HOPKINS, 2009), sendo possível encontrar diversos grupos de pesquisa trabalhando em projetos cujo propósito está focado na construção de aplicações interativas (PETA5, 2010), (FILGUEIRAS & GIANNOTO, 2009). Dentre esses trabalhos, destaca-se a proposta de (SIMIONI & ROESLER, 2006) que apresenta um *framework* para criação de aplicações de TV digital com o objetivo de facilitar e agilizar o processo de desenvolvimento desse tipo de aplicação. O *framework* é baseado na estrutura GEM – MHP. (MHP, 2010). Em linhas gerais, (SIMIONI & ROESLER, 2006) concentram a proposta no desenvolvimento de aplicações interativas voltadas ao comércio eletrônico e ao acompanhamento de informações referentes ao mercado financeiro. O *framework* é responsável por gerenciar as informações recebidas pelo STB e está dividido em três camadas: i) Catálogo: responsável pelo gerenciamento da base de dados; ii) Painel: responsável por montar a interface da aplicação; iii) Aplicação: utiliza as informações geradas pelas outras camadas. A camada Catálogo irá manipular os arquivos recebidos pelo carrossel de dados e fornecer à camada Painel os itens

solicitados. Esta, por sua vez, disponibiliza uma interface para a Aplicação. Basicamente, as aplicações que utilizam esse *framework* recebem dados de um arquivo XML que serve para a atualização das interfaces de interação. De acordo com o conteúdo da informação recebida, a aplicação recria a interface, substituindo um determinado item (Figura 3.17).

Figura 3.17 Seleção de itens no aplicativo comércio eletrônico (SIMIONI & ROESLER, 2006).

Outra abordagem (CABRER, 2006) propõe a criação de um componente de *software* para integração entre plataformas de serviços e de TV, utilizando o MHP como *middleware* para TV digital e OSGi como uma plataforma para configurações de *gateways* residenciais. Com os avanços tecnológicos que estão ocorrendo nos últimos anos, os *gateways* residenciais são fundamentais para a criação de pontes de comunicação entre os AmIs, seus dispositivos e o mundo externo. Nessa proposta, o objetivo é permitir ao usuário acessar, através do televisor, as aplicações e os serviços existentes no AmI. Por isso, a necessidade em se utilizar o MHP e o OSGi, sendo que o primeiro é orientado a funções, e o segundo orientado a serviços. Como as duas tecnologias possuem arquiteturas diferentes, existe o desafio de integração entre elas. Essa questão é resolvida com a criação do *XbundLET*, uma aplicação que permite a comunicação entre o MHP e o OSGi. Um *XbundLET* não apenas define uma ponte de comunicação entre essas plataformas, como também constitui um elemento de *software* híbrido que pode ser executado em ambas as arquiteturas.

Na Figura 3.18a, é possível observar a comparação entre as diferentes arquiteturas, MHP e OSGi, respectivamente. Já na Figura 3.18b, é ilustrada a solução de integração entre as diferentes plataformas, através do componente *XbundLET*.

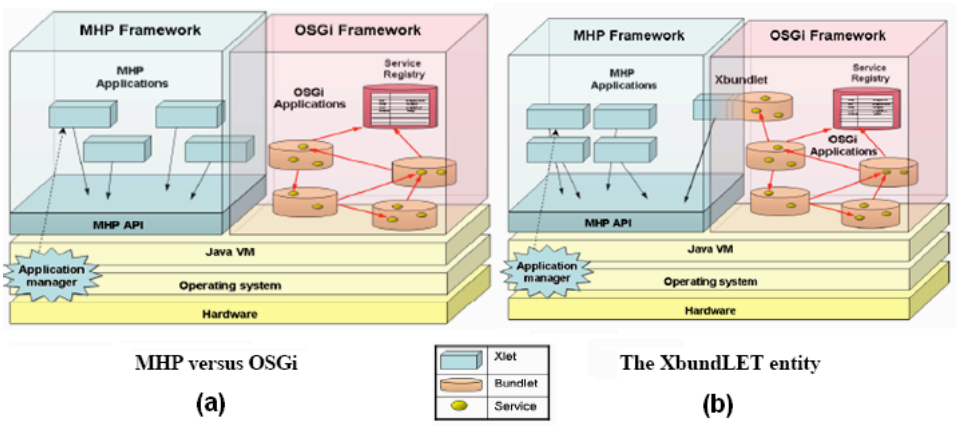

Figura 3.18 Arquiteturas e integração com XbundLET (CABRER, 2006).

A especificação MHP é um modelo com múltiplas camadas, que define a comunicação entre o sistema e as aplicações MHP. As aplicações MHP (*Xlets*) são um conjunto de arquivos de classes Java, escritas para utilizar um conjunto de bibliotecas na API MHP. Um *Xlet* pode, também, ser totalmente controlado por um gerenciador de aplicações contido nos STBs, os quais são responsáveis por monitorar, iniciar ou parar um *Xlet*. Logo, para a integração, foi desenvolvido o *XbundLET*, como uma aplicação híbrida e que pode integrar-se tanto com *Xlets* quanto com *bundles*.

O ciclo de vida de um *XbundLET* é definido em quatro estados: (i) iniciado: estado em que o *XbundLET* não está pronto para executar um *Xlet* ou um *bundle*, ele é apenas iniciado; (ii) inativo: estado em que o *XbundLE*T está pronto para executar um *Xlet* ou um *bundle*; (iii) ativo: estado em que o *XBundLET* é ativado; (iv) finalizado: estado para determinar que a execução do *XbundLET* foi finalizada.

Em relação ao método de como as aplicações MHP utilizam os serviços OSGi, ele pode ser visualizado (ver Figura 3.19a) e definido da seguinte forma: (1) os *bundles* para controle de dispositivos da residência registram seus serviços no OSGi *Service Registry*. Para informar o ambiente MHP da existência sobre um novo serviço OSGi, um novo componente

chamado "*Master XbundLET*" está constantemente em modo de "escuta" para detectar alguma mudança na plataforma OSGi; (2) O "*Service Registry*" informa ao "*Master XbundLET*" sobre a presença de um novo serviço, através do componente "*Service Listener*"; (3) logo após, o "*Master XbundLET*" envia uma mensagem atualizada ao *Inter-Xlet Communication* (IXC) *Registry*, responsável por permitir a comunicação entre *Xlets*. Por essa razão, um *Xlet* do *framework* MHP pode solicitar ao "*IXC Registry*" para que procure os serviços OSGi disponíveis até o momento.

Para o método inverso, ou seja, de como as aplicações OSGi utilizam o MHP, está ilustrado, na Figura 3.19b, sendo definido como: o "*Application Manager*" exporta as funcionalidades da API MHP para o ambiente OSGi sob um conjunto de serviços gerenciados pelo "*OSGI2MHP XbundLETs*" que registra suas características dentro do *"OSGI Service Registry*". Após isso, no contexto do *framework* OSGi, as características fornecidas pela API MHP podem ser utilizadas como qualquer outro serviço OSGi. Para os *bundles* não há diferença entre usar os serviços oferecidos por outros *bundles* ou oferecidos pela plataforma MHP com os "*OSGi2MHP XbundLETs*".

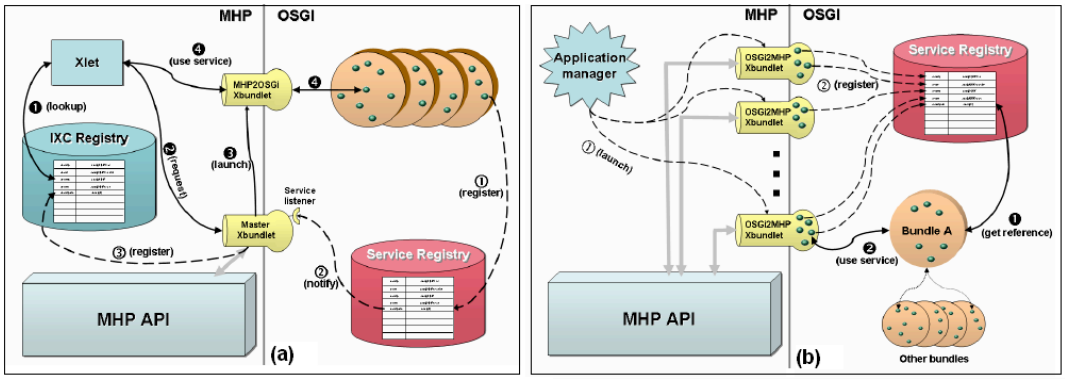

Figura 3.19 (a) MHP com OSGi (b) OSGi com MHP (CABRER, 2006).

A integração entre diferentes plataformas e utilização de TV para interação também é apresentada na proposta de (VIANA *et al.*, 2009), cujo objetivo é a criação de uma interface de acesso chamada Ginga OSGi *Bridge*, capaz de oferecer às aplicações Java e NCL uma ponte de acesso transparente ao *framework* OSGi. Nesse caso, as aplicações desenvolvidas

poderiam verificar estados de sensores, acionar dispositivos e interagir com os serviços residenciais que estivessem disponíveis no ambiente e, consequentemente, no *framework* OSGi. Por outro lado, a proposta de (VIANA *et al.*, 2009) necessitaria que a especificação do *middleware* Ginga fosse ampliada para suportar essa comunicação transparente entre as aplicações Ginga e o *framework* OSGi, uma vez que a integração é dada em nível de *middleware.* Outra abordagem que utiliza os recursos da TV digital é proposta por (OLIVEIRA *et al.*, 2009), em que o foco está em oferecer aos usuários diversos serviços de monitoração da residência e de monitoração pessoal. Para isso, são definidos quatro componentes que fazem parte do projeto *Digital Automation in Monitoring and control using GINGA technology* (Diga), são eles: (i) Saúde: responsável por integrar os dispositivos de monitoração dos sinais vitais e saúde pessoal dos usuários; (ii) Casa: integra os dispositivos de monitoração da residência, tais como: sensores de temperatura e de presença; (iii) Mundo: realiza a ampliação do componente Saúde, agregando uma estrutura para permitir que o usuário possa solicitar um pedido de resgate caso alguma emergência ocorra. (iv) Aqui: é um conjunto de APIs que permitem ao usuário criar e acessar comunidades de interesse comum, com o objetivo de obter informações sobre doenças e tratamentos. Os componentes da estrutura Diga são integrados com módulos de comunicação sem fio (SUNSPOT, 2010) com o intuito de permitir uma interface para troca de dados com PCs ou receptores de TV digital. Na abordagem de (OLIVEIRA *et al.*, 2009), foram implementados dois protótipos, sendo um para monitoração da residência e outro para monitoração de pacientes, ambos utilizando a TV digital como meio de acesso a esses serviços. A Figura 3.20 ilustra a arquitetura de comunicação para monitoração de pacientes.

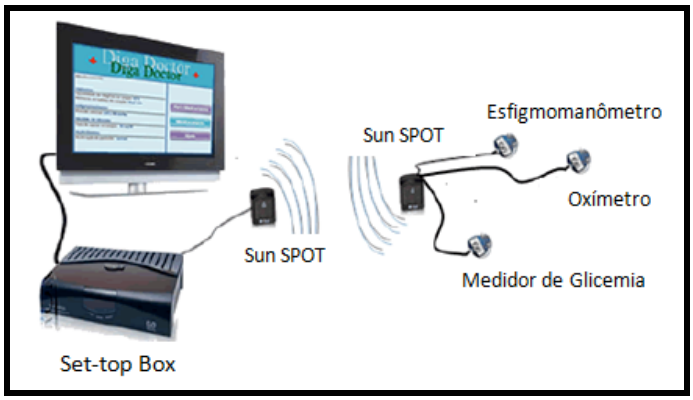

Figura 3.20 Integração de dispositivos no Diga (OLIVEIRA *et al.*, 2009).

Um dos principais aspectos dessa abordagem é que, a exemplo de (VIANA *et al.*, 2009), ela necessitaria estender a especificação do *middleware* Ginga para prover uma integração com o *hardware* utilizado no projeto.

Já na dissertação de mestrado (ANDREATA, 2006), é proposto um portal para aplicações colaborativas de TV digital, chamado de InteraTV. A abordagem foi desenvolvida para o *middleware* de interatividade do sistema europeu e o objetivo do trabalho é apoiar as áreas de educação com ferramentas para ensino à distância, baseadas em TV digital interativa. Em outra dissertação de mestrado (BORGES, 2007), o objeto de estudo está na criação de uma arquitetura para descoberta de serviços em ambientes de TV digital interativa. O autor propõe uma estratégia para utilizar serviços distribuídos e fazer também o balanceamento de carga para acesso a um determinado serviço, como por exemplo: a emissora de TV transmite um aplicativo que utiliza o canal de retorno para acesso a um determinado serviço, e todos os telespectadores resolvem acessá-lo no mesmo momento, podendo ocorrer uma sobrecarga no sistema. Nesse caso, a solução buscaria distribuir os clientes para acessarem o mesmo serviço em provedores distintos. Na arquitetura proposta por (BORGES, 2007), existe um camada abaixo da do *middleware* de interatividade, que ficaria disposta entre esta e a camada do sistema operacional, sendo responsável pela descoberta de serviços e pelo agrupamento dos protocolos de descoberta de serviços (Figura 3.21).

Figura 3.21 Camadas da arquitetura para descoberta de derviços (BORGES, 2007).

A proposta de (BORGES, 2007) sugere mais uma camada na arquitetura de interatividade, só que nesse caso, o trabalho utilizou para validação aplicações que são executadas sobre o *middleware* de interatividade e que utilizam APIs para descoberta de serviços já existentes. Nesse caso não seria necessária uma camada adicional na arquitetura para tentar abstrair a complexidade no desenvolvimento das aplicações, uma vez que essas APIs já possuem essa funcionalidade.

3.4.1 Modelos de Negócio

Além das propostas e projetos de pesquisas que estão surgindo no meio acadêmico, a utilização de recursos da TV digital interativa recebe a atenção do mercado que busca obter vantagens comerciais dessa área. As aplicações interativas no Brasil vêm sendo desenvolvidas e transmitidas pelas principais emissoras de TV aberta (IDGNOW, 2010) as quais acreditam que, além de poder inserir informações adicionais à programação, poderão ter uma nova fonte de receita em publicidade, uma vez que as aplicações interativas podem ser patrocinadas, levando na interface da aplicação a logomarca do anunciante. Essas emissoras estão formando grupos de profissionais para trabalhar em cooperação com empresas de desenvolvimento de aplicações interativas para TV digital já estabelecidas no mercado (HXD, 2010), (TQTVD, 2010). Um exemplo dessa evolução é a TV Globo (GLOBO, 2010), que já desenvolveu e

transmitiu vinte e cinco aplicações interativas desde 2006 (FRIAS, 2010). Dentro desse portfólio, destacam-se as aplicações interativas para esportes e entretenimento.

Mesmo as aplicações tendo um enfoque que leva informações complementares ao vídeo a que o telespectador está assistindo, as emissoras possuem estratégias e visões diferentes em relação à interatividade. Algumas apostam em aplicações customizadas para cada programa de TV (GLOBO, 2010; RECORD, 2010; BAND, 2010), outras apostam em aplicações de estilo portal (SBT, 2010), em que a mesma aplicação está disponível durante toda a programação, variando apenas o conteúdo. A Figura 3.22 ilustra uma série de aplicativos que estão sendo desenvolvidos pelas emissoras de TV.

Figura 3.22 Exemplos de aplicativos que as emissoras de TV estão desenvolvendo.

Além das aplicações para interação do público com as emissoras de TV, outras aplicações interativas vêm sendo desenvolvidas com o objetivo de explorar diferentes áreas do conhecimento, identificar nichos de mercado ou oferecer algum tipo de serviço aos usuários. Um exemplo disso é o projeto Escola1.TV (RESOLVE, 2010) que é uma solução para Ensino à Distância (EAD) utilizando a interatividade da TV digital. Esse projeto é apoiado pela Financiadora de Estudos e Projetos (FINEP) e consiste, basicamente, de uma proposta que utiliza toda uma infraestrutura de ensino já existente para aprendizagem na *web*

e adiciona uma nova interface de acesso voltada à TV digital. Um aspecto importante nesse projeto é que ele leva em consideração a usabilidade na TV digital, que é objeto de estudo em dissertações de mestrado (BRACKMANN, 2010), uma vez que a interface de acesso para o EAD disponível para PCs é totalmente diferente dos recursos disponíveis nas TVs e STBs. Vários outros aplicativos vêm sendo propostos (EUROITV, 2009), abrangendo desde aqueles voltados ao acesso de serviços bancários (*t-banking*) quanto de comércio eletrônico (*t-commerce*) disponibilizados, à medida que a TV digital vai conquistando espaço (STICKERCENTER, 2010).

3.5 CONSIDERAÇÕES GERAIS

Na literatura, são encontrados diversos trabalhos direcionados à área de AmIs, que vão desde propostas de infraestruturas de rede e automação até aplicações para interação com esses AmIs. Neste capítulo, foram selecionadas algumas das principais propostas que estão relacionadas com o presente trabalho. Dentro dessa pesquisa, é possível identificar os pontos positivos e negativos de cada abordagem, analisando as contribuições e os pontos de interesse, como, por exemplo: na proposta de (HELAL, 2005) a contribuição está no *middleware* e na arquitetura genérica que serve como referência para construção de projetos na área de AmI. A utilização do *framework* OSGi como base para a descoberta e a composição de serviços também merece destaque por este ser um padrão que vem se consolidando a cada dia e se tornando um *gateway* bastante utilizado em AmI. Os autores destacam as camadas de conhecimento e de aplicação como sendo as áreas que ainda há muito trabalho a ser feito. Isso reflete a busca intensa por uma interação cada vez mais natural e transparente entre usuários e AmI – ou humanos e sistemas computacionais de um modo geral. A identificação dos dispositivos que estão instalados no ambiente também é um ponto

importante na abordagem. Nesse caso, uma das dificuldades estaria na necessidade de se ter uma série de leitores e *tags* de RFID espalhados pelo ambiente.

No *framework* proposto em (REDONDO, 2007), o fator positivo está na possibilidade de que novos serviços possam ser instalados e reconfigurados apenas trocando as especificações BPEL, que consistem em um arquivo XML. Embora o *framework* BPEL/OSGi aceite serviços OSGi compostos, isso não resolve todos os problemas relacionados à descoberta OSGi, pois o cliente OSGi é obrigado, também, a conhecer as informações sobre os serviços. Uma desvantagem da proposta está, justamente, no fato de que a composição dos serviços somente é possível desde que a especificação BPEL seja criada ou alterada em tempo de projeto, o que implica no desenvolvimento de aplicações estáticas (*offline*). Parece ser interessante, do ponto de vista do usuário e do conceito de AmI (que se adapta às necessidades dos usuários), ter uma aplicação que seja capaz de descobrir e gerenciar serviços em tempo de execução. Além disso, é possível observar que outras propostas (STAVROULAKI, 2006; INDULSKA, 2001) buscam abstrair a complexidade do gerenciamento de serviços nos AmIs.

Outra contribuição importante está na proposta apresentada em (ZIEGLER, 2005), que integra diversas bases de dados e possui um sistema de autenticação, baseada em níveis de segurança que podem aumentar de acordo com o tipo de aplicação que se deseja utilizar no AmI. Porém, a arquitetura do *middleware* proposta está condicionada ao *framework* OSGi. Já na abordagem de (GROPPE & MUELLER, 2005), ocorre um conflito entre preferências e dispositivos quando um dispositivo especificado não está disponível por alguma razão, desencadeando um problema. Uma solução poderia ser a utilização de outro dispositivo com a mesma característica ou capacidade. Caso o dispositivo especificado esteja bloqueado por outra preferência e nenhuma alternativa seja possível, isso pode ser tratado pelo componente "*conflict resoluction*". Outra solução seria utilizar um dispositivo de outra parte física do

ambiente. Na proposta de (RÖCKER, 2006), mesmo permitindo-se o acesso às informações privadas dos usuários em locais públicos, um dos grandes problemas está em como integrar todos os AmIs públicos e como gerenciar o perfil dos usuários, uma vez que o SPIROS utiliza RFID para a identificação das pessoas. Isso implica em, além de cada usuário ter que carregar consigo uma *tag* RFID, há a necessidade de um ou vários sistemas com grande capacidade de armazenamento dos perfis. Além disso, seria necessário identificar a estratégia de comunicação a ser adotada, considerando, por exemplo, as arquiteturas de sistemas distribuídos e centralizados.

Na proposta de (KAPLAN & LUNN, 2001), a vantagem está na possibilidade de se visualizar dinamicamente um mesmo conteúdo em múltiplos dispositivos, em que são considerados seus recursos computacionais. Isso poderia facilitar, por exemplo, a forma como o usuário interage com o AmI e como este disponibiliza as mesmas informações de interação para múltiplos pontos de acesso. Por outro lado, o FlexXML necessita que sejam criadas diversas especificações XSL de apresentação, basicamente uma especificação para cada terminal que irá acessar o sistema. Ou seja, de um lado há a vantagem de se criar e gerenciar um único conteúdo, mas, por outro lado há a necessidade de se manter múltiplos arquivos de apresentação desse conteúdo. Semelhante ao que acorre na proposta do FlexXML, a infraestrutura @TAS necessita de múltiplas versões de filtros para adaptar um conteúdo a determinados dispositivos. Além disso, a @TAS não prevê como esses filtros são implementados e quais são os requisitos, uma vez que são tratados como uma caixa preta. Novamente, no contexto dos AmIs, a manutenção e a evolução dos sistemas baseados nessas abordagens poderiam estar comprometidas, devido à necessidade de reprojeto caso exista a inserção de novos serviços e dispositivos no sistema. Mesmo com a adaptação dinâmica do conteúdo a ser exibido, poderia ser necessária a criação de novos filtros e a descrição de novas regras para filtragens a cada atualização no cenário do AmI.

No que se refere à interação multimodal, a proposta de (MANN & HELAL, 2002) apresenta um controle remoto universal para AmIs, em que é necessário existir a conectividade de todo o sistema, através da integração entre os componentes da arquitetura, que envolve desde a necessidade de se ter um *smartphone* com suporte à linguagem Java, até controladores e *softwares* X10. Da mesma forma, as soluções apresentadas em (HOMESYSTEMS, 2010) possuem problemas semelhantes aos existentes na abordagem de (MANN & HELAL, 2002): as aplicações são projetadas com um comportamento estático, em que todas as regras e condições para interação são previstas em tempo de projeto. Novamente, neste caso há uma limitação no sistema para reconhecer ou se adaptar aos novos serviços e dispositivos de automação que são inseridos no AmI após o desenvolvimento das aplicações que estão em execução. Já a interação por voz entre usuário e AmI apresentada em (GÁRATE, 2005) tem como requisito a necessidade de o usuário carregar consigo um microfone sem fio. Essa é uma solução que pode ser interessante quando consideradas as opções de multimodalidade, inclusas no projeto (EMBASSI, 2008) e que tenta abranger diferentes domínios de aplicação (residências, automóveis, espaços públicos, etc.). Por outro lado, existe a dificuldade em tratar os requisitos de projetos, que podem ter características totalmente diferentes em cada cenário ou domínio.

Quanto ao uso de TV digital Interativa em AmIs, observam-se algumas contribuições importantes, como na proposta de (SIMIONI & ROESLER, 2006) em que é definido um *framework* para auxiliar na construção de aplicações interativas. A proposta está mais direcionada a aplicações de comércio eletrônico e utiliza o *middleware* do padrão europeu como base para o desenvolvimento de aplicações. Já na proposta de (CABRER, 2006), a contribuição está na possibilidade de permitir que os usuários acessem os serviços disponíveis no AmI, o que envolve o acionamento de dispositivos de automação instalados no ambiente. A base para o desenvolvimento da proposta está, além da utilização do OSGi e do MHP, na

criação do componente híbrido *XbundLET*, que realiza a função de integração entre essas diferentes arquiteturas, servindo como uma ponte de conexão entre as camadas da arquitetura MHP e os serviços OSGi. Assim, a integração entre essas tecnologias proporciona aos usuários a possibilidade de acesso aos dispositivos e serviços do AmI. Por outro lado, a proposta está atrelada ao uso das duas tecnologias, MHP e OSGi, resultando, como uma das consequências, a necessidade de se desenvolver aplicações em linguagem Java, utilizando as APIs para MHP e OSGi. Além disso, os serviços existentes no AmI devem, obrigatoriamente, ser visualizados e acessados exclusivamente pelo televisor. Uma alternativa seria permitir aos usuários o acesso a serviços e a interação com o AmI de qualquer terminal, ou dispositivo computacional, que estivesse no ambiente, e que oferecesse suporte a essas aplicações, como, por exemplo, telefones celulares com o *middleware* de interatividade. Nesse caso, a adoção de *Web Services* poderia, também, resolver o problema de interoperabilidade entre dispositivos e a forma como estes acessam os serviços do AmI, em que um mesmo serviço ou dispositivo de automação poderia estar disponível a diversas plataformas computacionais sem, necessariamente, fazer uso do MHP e do OSGI. Outra possibilidade seria a separação da interface de aplicação com a execução dos serviços, similarmente ao FlexXML (KAPLAN & LUNN, 2001), em que o conteúdo de uma aplicação exibido em um dispositivo computacional pode ser disponibilizado para múltiplas plataformas apenas adaptando à sua forma de representação. Ainda, outra característica não considerada por (CABRER, 2006) é a composição de serviços, oferecendo ao usuário a possibilidade de criar novos serviços na própria aplicação de visualização e interação com o AmI, ou seja, em tempo de execução do sistema, o usuário poderia compor novos serviços a partir de serviços já existentes, similar a abordagem para composição de serviços proposta em (REDONDO, 2007).

Em outras propostas (VIANA *et al*., 2009; OLIVEIRA *et al*., 2009; BORGES, 2007) identifica-se a necessidade de extensão da especificação atual do *middleware* Ginga para que

seja possível utilizar os recursos apresentados. Porém, do ponto de vista comercial, diversas soluções já estão sendo implementadas e disponibilizadas por emissoras de TV em todo o país. As aplicações interativas estão sendo desenvolvidas para as áreas de esportes, meteorologia, notícias, novelas, dentre outras. A principal contribuição da TV digital interativa para os radiodifusores é a possibilidade que as aplicações oferecem para agregar informação ao conteúdo que está sendo transmitido e, também, a capacidade de obter uma nova fonte de receita, uma vez que os aplicativos podem ser patrocinados por anunciantes, tornando a interatividade na TV um novo modelo de negócio.

3.6 DESAFIOS

Inúmeras são as propostas e, por consequência, os desafios a serem tratados em AmIs. Alinhado com a visão de (HELAL, 2005), um dos principais desafios em AmIs está concentrado nas camadas de conhecimento, de aplicação e de serviços, que envolve desde a sua descoberta até a sua composição. Não necessariamente a descoberta de um serviço (disponível em um AmI) por parte do usuário signifique que ele o conheça ou saiba quais os impactos que esse serviço irá causar no AmI. Além disso, há o desafio relacionado à questão de como descobrir e compor novos serviços e se essa tarefa será realizada em tempo de execução das aplicações ou, ainda, em tempo de projeto, como proposto em (REDONDO, 2007), numa tentativa de aumentar o suporte da composição de serviços OSGi presente em ambientes inteligentes, através de um *framework* OSGi/BPEL.

Além disso, percebe-se que a maioria das propostas apresentadas para o tema de gerenciamento de serviços estão vinculadas ao *framework* OSGi. Essa pode ser uma das alternativas a ser utilizada, entretanto, o desafio está relacionado também ao fato se tentar desenvolver uma arquitetura independente de tecnologia e que seja capaz de explorar outros

mecanismos para gerenciamento de serviços que não, necessariamente, utilizem ou dependam da existência de uma tecnologia específica.

Relacionados à customização de AmIs, existem, pelo menos, dois grandes desafios a serem tratados: o primeiro está ligado à localização dos dispositivos de automação, que poderiam ser automaticamente reconhecidos pelo sistema sempre que um novo dispositivo fosse inserido no AmI. O segundo desafio está no critério a ser utilizado para atribuir as prioridades de acesso dos usuários frente aos serviços e aos dispositivos de automação, ou seja, definir qual usuário tem prioridade no acesso a uma funcionalidade caso ocorra um solicitação de acesso simultânea, ou caso um serviço já esteja sendo utilizado por outro usuário (GROPPE & MULLER, 2005). Outras propostas para customização de sistemas que são baseados na *web* também poderiam ser utilizadas no domínio dos AmIs. Para isso, o desafio seria a criação de componentes para uma arquitetura, capaz de explorar as funcionalidades de adaptação necessárias para essa área, as quais incluem o ajuste automático do conteúdo de visualização, das aplicações de interação e do AmI, considerando: (i) as diferentes plataformas computacionais de acesso ao AmI; (ii) a infraestrutura de comunicação de dados existente no ambiente; (iii) os usuários e seus requisitos; (iv) os dispositivos de automação e os serviços disponíveis no AmI; (v) os mecanismos de segurança e controle de acesso e; (vi) a evolução do AmI.

Apesar das propostas de interação que vêm sendo apresentadas nos últimos anos, ainda persiste o desafio de como se desenvolverem aplicações de interação para múltiplas plataformas sem existir o reprojeto. Além disso, há o desafio de como o usuário pode gerenciar o seu próprio AmI sem a necessidade de se ter, ou ser um administrador de sistemas de automação. Outro fator a ser considerado é que, cada vez mais, há necessidade e busca por soluções de interação mais simples, amigáveis e intuitivas entre homens e máquinas. Nesse caso a interação multimodal vem ganhando destaque, por permitir que o usuário possa

interagir com um mesmo equipamento ou serviço de várias maneiras. Uma dessas formas poderia ser utilizar o avanço da TV digital como plataforma de interação com o AmI - nesse caso, o desafio estaria em oferecer mecanismos para auxiliar o desenvolvimento de aplicações interativas, integradas com sistemas de automação predial/residencial e que fossem executadas em receptores de TV digital dotados de um *middleware* de interatividade padronizado, como, por exemplo, o Ginga.

Atualmente, ainda não há ferramentas de autoria para construção de aplicações interativas para o *middleware* Ginga. Todas as aplicações que são desenvolvidas, tanto na área acadêmica quanto comercial, necessitam de programadores com conhecimento técnico em linguagens NCL e/ou Java. Nesse caso, o desafio estaria em propor uma solução que auxiliasse na construção dessas aplicações e que pudesse gerar código automaticamente. Os *frameworks* poderiam contribuir significativamente para esse processo, otimizando o tempo de desenvolvimento de aplicações interativas e permitindo a reutilização desses projetos.

Diante dos desafios encontrados em AmIs, este trabalho é proposto na tentativa de oferecer uma solução para facilitar e agilizar o processo de construção de aplicações interativas direcionadas ao gerenciamento de AmIs através da TV digital, atacando principalmente a descoberta de dispositivos e de serviços automatizados, a criação de cenários para gerenciamento de AmIs e a geração automática de código para aplicações interativas.

4 PROPOSTA DA TESE

Este capítulo apresenta a proposta de um *framework* para a integração entre AmIs e o *Middleware* do Sistema Brasileiro de TV Digital Terrestre. O *framework* proposto é especificado através da *Unified Modeling Language* (UML) (OMG, 2010), uma linguagem visual, padronizada, amplamente difundida e utilizada para modelar sistemas computacionais através da orientação a objetos. Essa linguagem tornou-se padrão para modelagem de *software* e é adotada internacionalmente pela indústria de engenharia de *software*, pois auxilia na definição das características do sistema, tais como seus requisitos, seu comportamento, sua estrutura lógica e a dinâmica de seus processos (GUEDES, 2005).

4.1 MODELO CONCEITUAL

A principal contribuição deste trabalho está na especificação de um *framework* que possibilite a integração entre duas áreas: (i) Ambientes Inteligentes: cenários de automação predial/residencial compostos por sensores, controladores, atuadores e infraestrutura de comunicação de dados; (ii) Sistema Brasileiro de TV Digital Terrestre: que além de oferecer imagens em alta definição e permitir a mobilidade da TV, é composto pelo *middleware* de interatividade, o qual permite que aplicações computacionais interativas possam ser executadas sobre os receptores de TV digital.

A proposta oferece mecanismos para que, além de decodificar o sinal da TV digital, os receptores possam se tornar plataformas computacionais para gerenciamento do AmI, agregando a esses dispositivos uma nova funcionalidade e oferecendo aos usuários a possibilidade de uma nova interface para acesso aos serviços e aos dispositivos de automação existentes nos AmIs. A Figura 4.1 ilustra a contribuição da proposta através da integração dessas duas áreas.

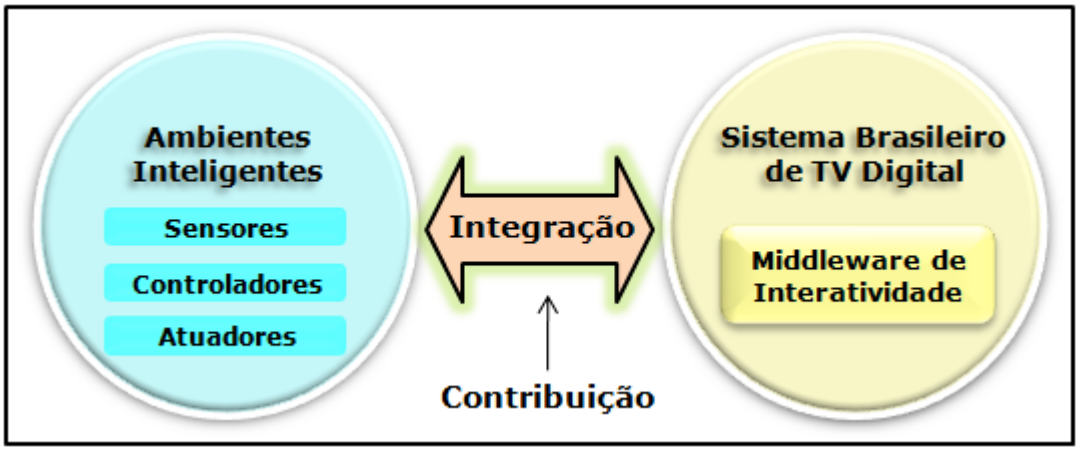

Figura 4.1 Contribuição da proposta.

A integração, ilustrada na Figura 4.1, é realizada através de um *framework* que será especificado nas próximas seções. Na literatura existem algumas definições para o conceito de *frameworks* (ROGERS, 1997; SHALLOWAY & TRORR, 2004). Entretando, neste trabalho a definição adotada é a mesma proposta por (GAMMA *et al.,* 2000): "um conjunto de classes cooperantes para a construção de projetos reutilizáveis". A utilização de um *framework* se justifica por ele implementar uma arquitetura que pode ser utilizada de modo a tornar o desenvolvimento de sistemas computacionais mais rápidos e produtivos, facilitando a captura de decisões de projetos que são comuns ao domínio de uma aplicação e permitindo uma redução do tempo utilizado com o desenvolvimento de novas soluções. A principal contribuição de um *framework* é a definição de sua arquitetura, projetada para suportar todas as aplicações de um determinado domínio, de maneira flexível e extensível (GAMMA *et al*., 2000). Dessa forma, o *framework* proposto nesse trabalho possui algumas funcionalidades, ilustradas no diagrama de casos de uso da Figura 4.2, sendo destacadas principalmente as seguintes:

- Mapear os dispositivos físicos de automação presentes no AmI para o mundo computacional, através de modelos orientados a objetos ou da descrição das classes de dispositivos;

- Auxiliar na construção de aplicações interativas para acesso aos serviços e aos dispositivos de automação existentes nos AmIs;

- Permitir a criação de cenários de automação independentes da plataforma de *hardware* em que serão executados;

- Possibilitar a reutilização de projetos, otimizando o tempo de desenvolvimento de novas aplicações;

- Permitir a geração automática de código, criando aplicações baseadas no perfil do *hardware* da plataforma-alvo e no perfil da linguagem de programação suportada pelo *middleware* de interatividade do SBTVD.

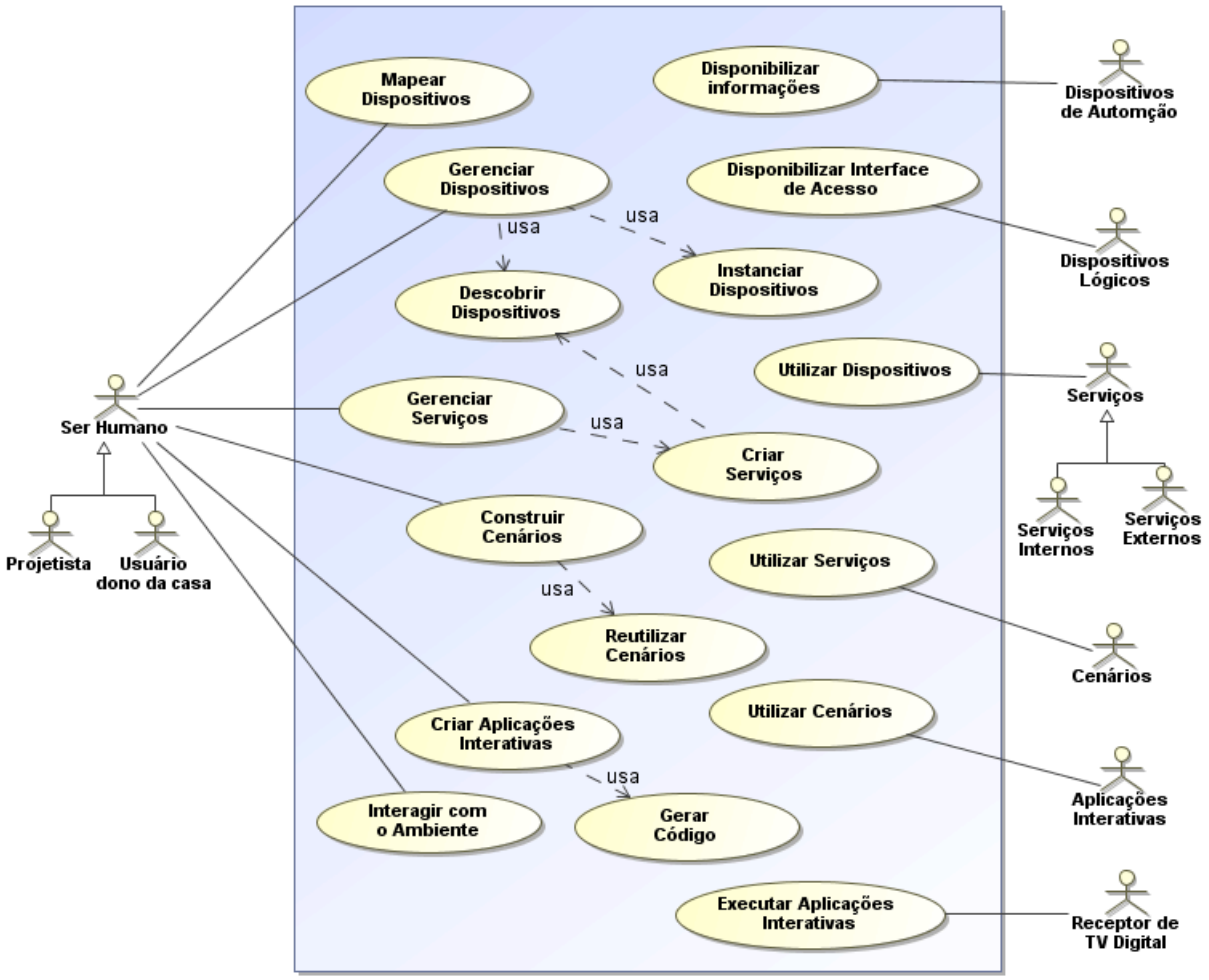

Figura 4.2 Principais funcionalidades da proposta.

As funcionalidades que são abordadas pelo *framework* proposto neste trabalho podem estar disponíveis em diversas plataformas computacionais, dentre elas os receptores de TV digital, os quais podem ter a sua estrutura representada por cinco camadas, ilustradas na Figura 4.3 (BARBOSA & SOARES, 2008).

Figura 4.3 Estrutura em camadas de um receptor de TV Digital (BARBOSA & SOARES, 2008).

Como um dos objetivos deste trabalho é oferecer recursos para que a as aplicações interativas de gerenciamento de AmIs sejam executadas sobre o *middleware* de interatividade dos receptores de TV digital, é ilustrada na Figura 4.4 uma representação em camadas que compõem a estrutura do *framework* proposto.

Figura 4.4 Estrutura em camadas do *framework*.

A estrutura do *framework* proposto está dividida em sete camadas: (i) dispositivos de automação: representa os dispositivos físicos encontrados no ambiente de automação predial/residencial, tais como: sensores, controladores e atuadores; (ii) modelagem:

corresponde ao processo de mapeamento dos dispositivos de automação através de modelos orientados a objetos; (iii) dispositivos lógicos: corresponde à representação computacional dos dispositivos de automação encontrados no mundo real; (iv) serviços: representam as funcionalidades e recursos disponibilizados pelos dispositivos de automação; (v) cenários: representa um conjunto de serviços disponíveis para a interação com o AmI; (vi) geração de código: é o processo de transformar as informações de um cenário e gerar código em alguma linguagem de programação que seja suportada pelo *middleware* de interatividade; (vii) aplicações interativas: são os cenários já codificados e serão executadas nos receptores de TV digital para gerenciarem o AmI.

Comparando as Figuras 4.3 e 4.4, observa-se que a camada "Aplicações Interativas" está representada em ambas as estruturas. Essa representação serve para identificar o ponto de intersecção entre as duas áreas que serão integradas. Ou seja, a integração entre os AmIs e o SBTVD é realizada neste trabalho através das aplicações interativas, construídas com base na proposta de um *framework* e executadas sobre o *middleware* de interatividade do SBTVD. Com a união das Figuras 4.3 e 4.4 tem-se uma visão de todas as camadas envolvidas no processo de integração entre AmIs e os receptores de TV digital.

A integração entre os AmIs e o SBTVD pode ser realizada através de diferentes estratégias, como, por exemplo, em nível de *middleware* (VIANA *et al.*, 2009). Porém, no escopo deste trabalho, a proposta de integração está em nível de aplicação, pelo fato de essa abordagem não necessitar de ampliação no que se refere à especificação do *middleware* de interatividade e permitir que diferentes aplicações possam ser exploradas em muitos segmentos, tais como: (i) emissoras de TV interessadas em desenvolver e comercializar aplicações de automação residencial; (ii) empresas de *software* buscando em plataformas de TV um novo nicho de mercado e; (iii) empresas de automação oferecendo novas interfaces de acesso aos seus sistemas.

Desse modo, é apresentada na Figura 4.5 a arquitetura de alto nível do *framework* proposto neste trabalho, cujos componentes especificados conseguem atender ao escopo da proposta. A estrutura completa da arquitetura e esses componentes serão descritos nas seções seguintes (de 4.1.1 a 4.1.7).

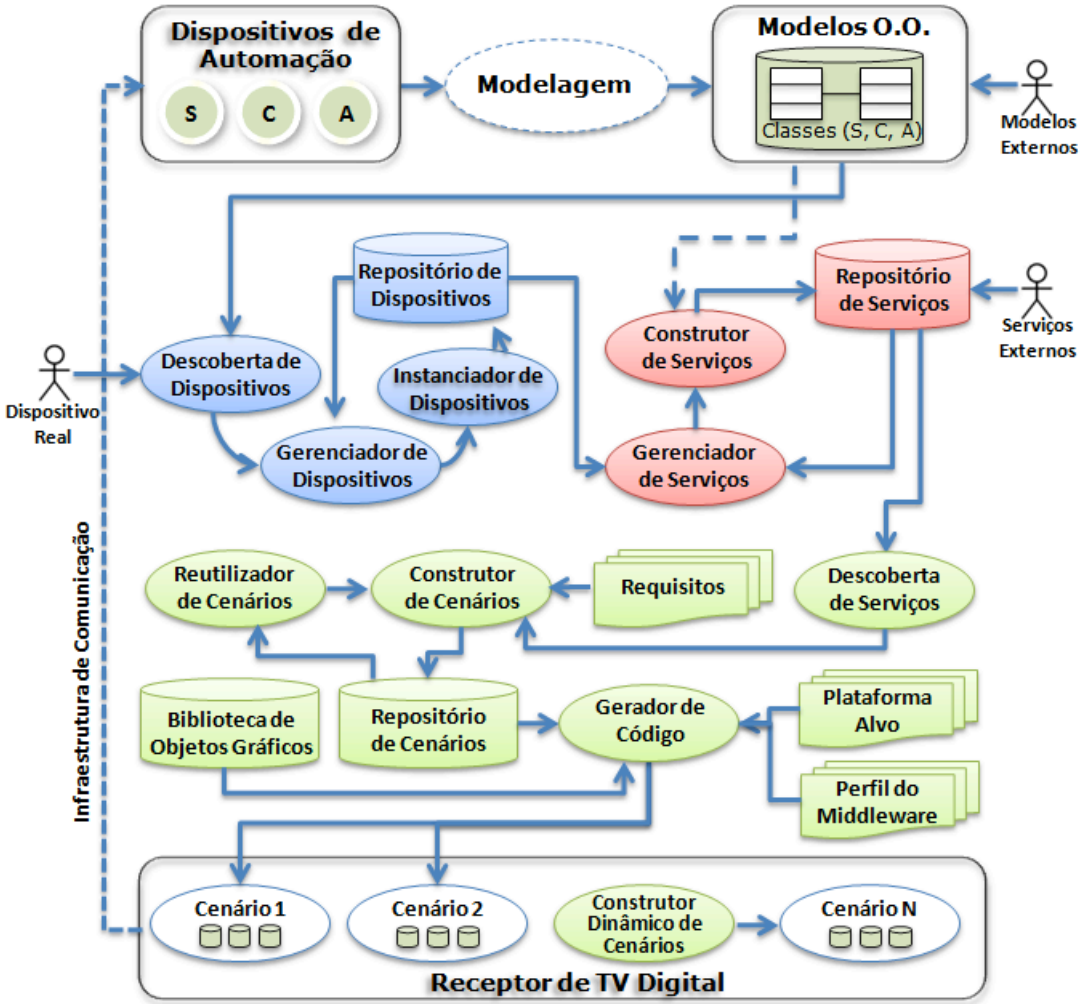

Figura 4.5 Arquitetura proposta.

A sequência de ações seguida pela arquitetura do *framework* é iniciada com a identificação dos dispositivos de automação que estão instalados no AmI. Baseado nessas informações é realizada a modelagem desses dispositivos, gerando uma representação computacional de todas as classes dos dispositivos de automação que podem ser utilizados. A partir desse momento, são criados os objetos, ou seja, são identificados os atributos e os métodos de cada classe de dispositivos, sendo então estas classes instanciadas. Com base nas instâncias de dispositivos, é possível criar os serviços responsáveis pela realização de uma

determinada tarefa no AmI e utilizam, para isso, os métodos disponibilizados pelos dispositivos instanciados (objetos). Além disso, é possível ter um serviço gerado a partir de uma classe de serviços. Para a construção dos cenários, é possível selecionar quais dos serviços criados serão utilizados, levando em consideração os requisitos do projeto, os quais podem conter o perfil que um usuário deverá ter para utilizar um serviço ou, ainda, quais serviços poderão ser executados por quais usuários. Os cenários são construídos de modo a independer da plataforma de *hardware* em que serão executados, visando à reutilização de cenários, ou seja, eles correspondem a um conjunto de serviços que estão disponíveis para uma configuração criada. A partir de um cenário é possível, então, gerar-se código em uma linguagem de programação suportada pelo *middleware* de interatividade, transformando o cenário em uma aplicação interativa que será executada na plataforma-alvo (receptor de TV digital).

4.1.1 Dispositivos de Automação

Nos AmIs pode existir uma grande variedade de dispositivos e sistemas de automação, cada um responsável pela execução de determinadas tarefas. Nem sempre as soluções de automação presentes nesses ambientes são desenvolvidas por um mesmo fabricante ou os protocolos de comunicação utilizados podem ser incompatíveis, ou seja, podem existir diferentes subsistemas de automação que não estejam, necessariamente, interligados. Nesse caso, a busca pela integração e pela interoperabilidade entre fabricantes vem sendo fortemente explorada nos últimos anos, através da criação de consórcios e tentativas para especificação de protocolos de comunicação padrozinados (ECHELON, 2010; BLUETOOTH, 2010; ZIGBEE, 2010).

Neste trabalho, o *framework* proposto considera como dispositivos de automação todos os sensores, controladores e atuadores que estão disponíveis no AmI e que oferecem

uma interface computacional de acesso às suas funcionalidades. É com base numa interface de comunicação conhecida que os serviços e os cenários são construídos e as aplicações poderão gerenciar o AmI. Como pode ser observado na Figura 4.6, um dispositivo de automação possui pelo menos uma interface de comunicação e oferece ao menos uma funcionalidade. Os dispositivos de automação podem ser especializados, basicamente, em três classes: (i) Sensores: responsáveis pela aquisição de dados do AmI; (ii) Controladores: responsáveis pelo processamento dos dados e tomada de decisão; (iii) Atuadores: responsáveis pela execução das ações no AmI. Em alguns casos, um mesmo dispositivo de automação pode agregar mais de um papel, por exemplo: um sensor poderia coletar os dados do AmI, executar os algoritmos para tomada de decisão e realizar o acionamento desejado, implementando o conceito de dispositivos inteligentes (BOLZANI, 2004, p. 23). Entretanto, o mais comum ainda é a existência de dispositivos dedicados à realização de uma única tarefa.

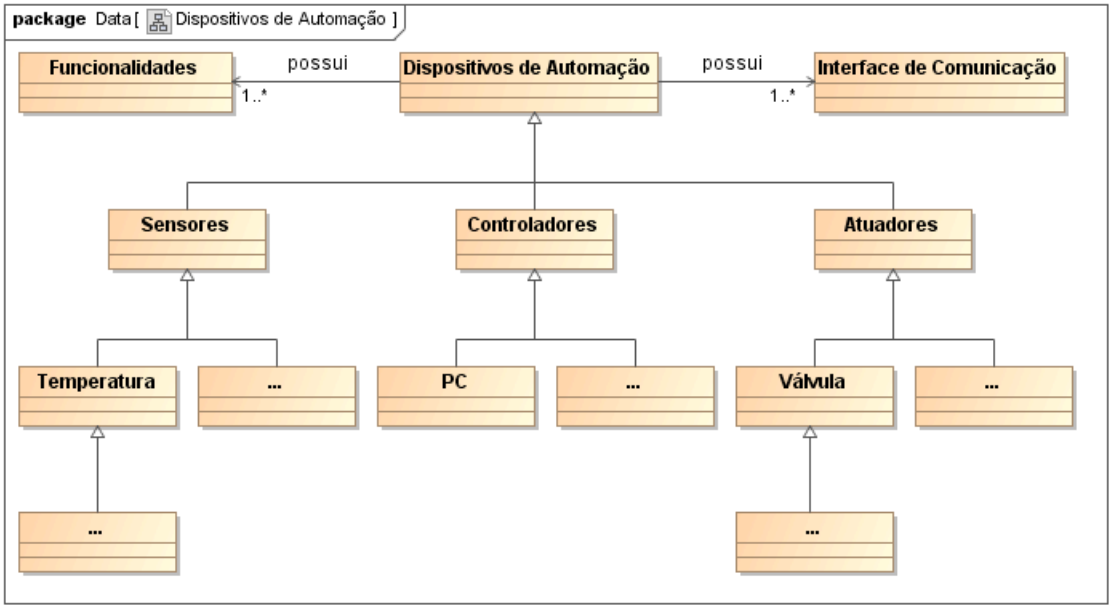

Figura 4.6 Dispositivos de automação existentes em AmIs.

Os dispositivos de automação disponíveis no AmI podem estar organizados e classificados hierarquicamente, em que classes genéricas de dispositivos podem ser especializadas em vários níveis. Por exemplo: se um atuador é um dispositivo de automação e uma válvula é um tipo de atuador, logo, uma válvula também é um tipo de dispositivo de

automação, porém, com algumas características que a diferenciam de outros dispositivos da classe de sensores ou da classe de controladores. Os conceitos de herança e generalização da orientação a objetos permitem, entre outras possibilidades, a construção desse tipo de associação, oferecendo uma melhor compreensão da estrutura e distribuição dos dispositivos de automação que compõem um determinado ambiente.

4.1.2 Modelagem

A modelagem baseada no paradigma da orientação a objetos é uma técnica que serve para construção de modelos visuais capazes de representar computacionalmente, através de diagramas, a complexidade do mundo real, ou seja, a modelagem é o mapeamento computacional da realidade. A modelagem é extremamente importante neste trabalho pelo fato de ela ser a responsável por trazer os dispositivos de automação presentes no AmI para dentro da estrutura computacional do *framework,* através da identificação e do mapeamento desses dispositivos. No processo de modelagem, é necessário conhecer o dispositivo de automação a fim de especificar os atributos e os métodos de cada um, ou seja, saber quais são as características (atributos) do dispositivo e quais são as funcionalidades (métodos) que ele oferece. Um exemplo de mapeamento computacional pode ser observado na Figura 4.7, em que os dispositivos físicos de automação de um determinado ambiente (a) são modelados e passam a ser representados como classes de dispositivos (b) que irão compor o *framework*.

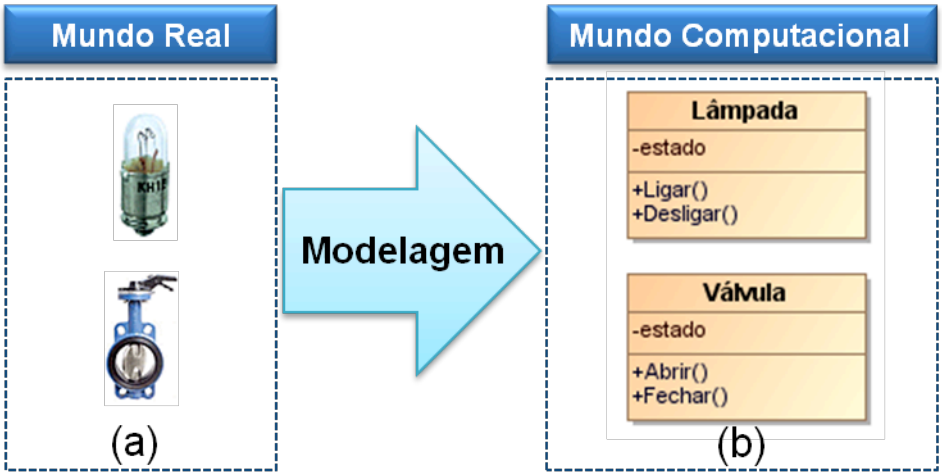

Figura 4.7 Modelagem de dispositivos.

Através do mapeamento computacional é possível, então, identificar quais são as classes de dispositivos que compõem o AmI. No exemplo ilustrado na Figura 4.7, é possível observar que os dispositivos de automação lâmpada e válvula foram representados computacionalmente por classes compostas dos seus respectivos atributos e métodos. A partir desse mapeamento, as classes dos dispositivos de automação presentes no AmI ficam disponíveis para serem utilizadas, isto é, os outros componentes do *framework* poderão instanciar essas classes, criando os dispositivos lógicos.

4.1.3 Dispositivos Lógicos

Com a modelagem orientada a objetos é possível construir um conjunto de classes que representam computacionalmente os dispositivos de automação presentes no mundo real. Essas classes podem ser instanciadas, criando os objetos que são tratados neste trabalho como dispositivos lógicos. Um dispositivo lógico corresponde a um componente com funcionalidades lógicas, os quais podem representar todo um sistema de automação, através de conexões e interações entre esses dispositivos (ARAUJO, 2005). Quando um dispositivo lógico é criado ele recebe todos os parâmetros de configuração necessários para a comunicação com o seu respectivo dispositivo de automação do AmI. De modo geral, o dispositivo lógico implementa uma interface conhecida de acesso ao dispositivo de

automação. No nível de dispositivos lógicos, a Figura 4.8 apresenta as classes referentes a essa camada da arquitetura do *framework*, sendo destacados os quatro componentes principais: (i) **Descoberta de Dispositivos**: responsável por consultar os modelos orientados a objetos construídos, identificando todas as classes de dispositivos de automação que foram mapeadas ou descritas e estão disponíveis no AmI. Além disso, esse componente permite que alguns dispositivos de automação possam ser descobertos sem a necessidade de estarem representados computacionalmente pelos modelos. Nesse caso, o componente poderia implementar diferentes protocolos utilizados para a descoberta de dispositivos; (ii) **Gerenciador de Dispositivos**: responsável por orquestrar os demais componentes da camada de dispositivos lógicos, oferecendo ao usuário uma interface de gerenciamento. Ele disponibiliza uma visão das classes de dispositivos que foram obtidas no processo de descoberta e uma visão dos dispositivos lógicos que foram instanciados. (iii) **Instanciador de Dispositivos**: responsável pela criação dos objetos, ou seja: os atributos e métodos de cada classe dos dispositivos de automação são parametrizados, gerando os dispositivos lógicos; (iv) **Repositório de Dispositivos**: componente responsável pelo armazenamento dos dispositivos lógicos que são criados, disponibilizando-os para serem consultados e utilizados pelos serviços que serão construídos.

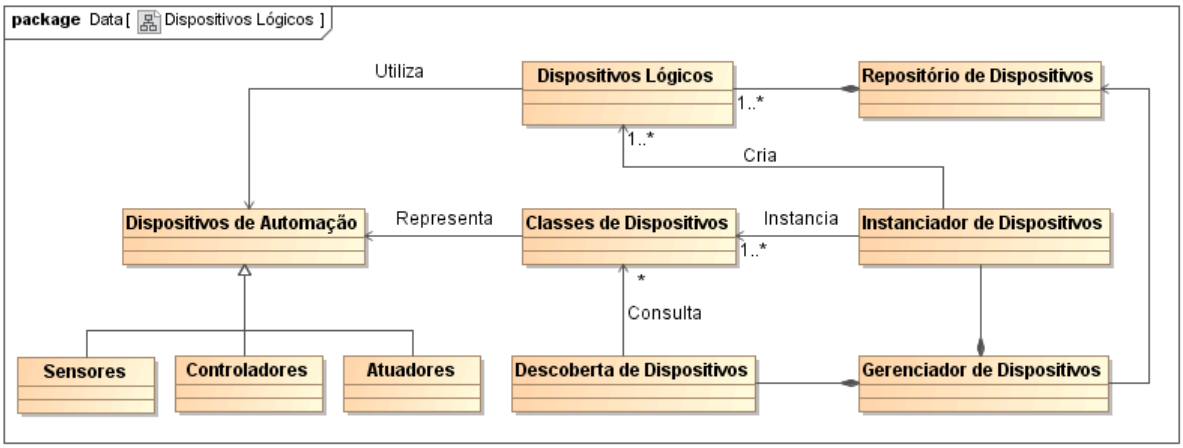

Figura 4.8 Dispositivos lógicos.

Em complemento à Figura 4.8, é apresentado na Figura 4.9 um diagrama de sequência UML para o gerenciamento de dispositivos. As classes de dispositivos representam, de forma computacional, os dispositivos de automação presentes no AmI. Essas classes de dispositivos são descobertas e instanciadas criando, então, os dispositivos lógicos que implementam a interface de acesso aos dispositivos de automação, os quais ficam disponíveis em um repositório de dispositivos. Na instanciação, os métodos e atributos de cada classe são parametrizados. Por exemplo: o serviço "*iluminar o ambiente*" utiliza o método *Ligar*() que é parametrizado com o valor correspondente ao *setpoint* de ligar o dispositivo de automação existente no AmI. Nesse caso o parâmetro poderia ser o valor um (1). Para o método *Desligar()*, o valor poderia ser zero (0). Esse exemplo serve para demonstrar como pode ser realizada a parametrização dos atributos e métodos de cada classe instanciada. Outros exemplos poderiam ser parâmetros do tipo *on*/*off*, ou um valor de dimerização da lâmpada, por exemplo: 30%, 75%, 100%. Por isso, os parâmetros de configuração de cada dispositivo lógico a ser instanciado dependem das características do sistema de automação e dos dispositivos de automação que compõe o ambiente. Todos os parâmetros definidos são armazenados dentro do respectivo dispositivo lógico que é instanciado. Caso algum dispositivo precise ser substituído, o serviço pode ser mantido, uma vez que o novo dispositivo lógico criado pode assumir os parâmetros do anterior.

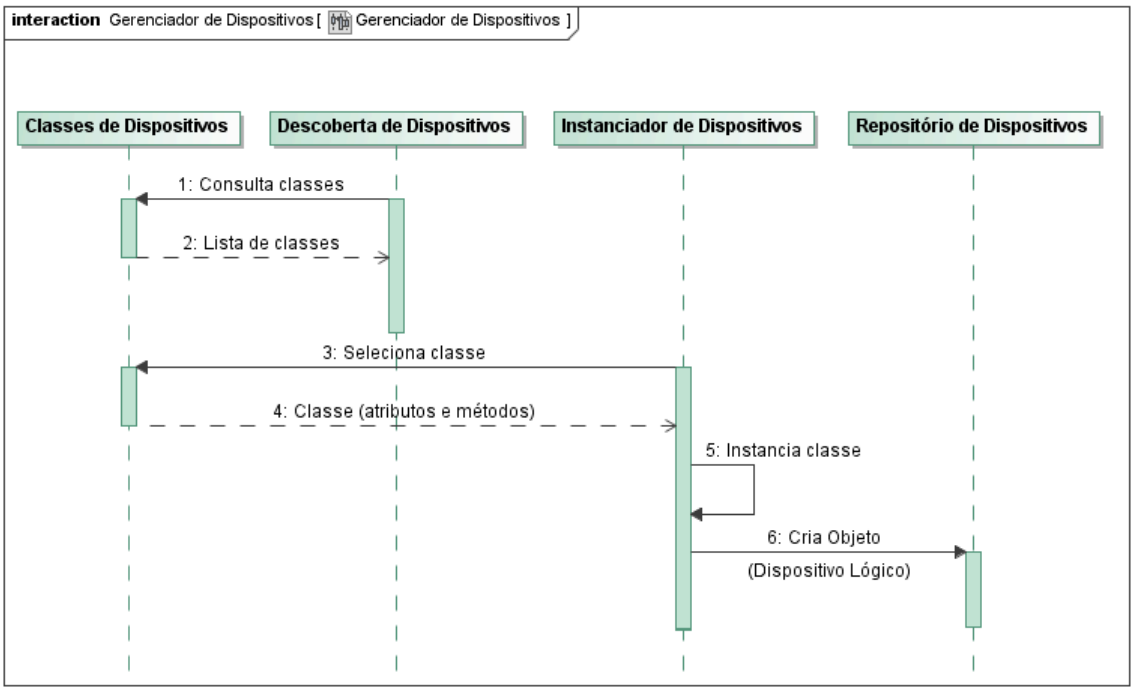

Figura 4.9 Gerenciamento de dispositivos.

O conceito de polimorfismo, existente na orientação a objetos, também pode ser adotado nessa abordagem, uma vez que a sobrecarga de métodos permite que um mesmo método existente na classe mãe seja herdado pelas classes filhas, porém com implementações diferentes, por exemplo: tanto a classe de dispositivo "*lâmpada convencional*" quanto "*lâmpada dimerizada*" herdam da classe mãe "*lâmpada*" o método "*ligar()*". Ambas irão implementar esse método, porém, a classe "*lâmpada dimerizada*" além de ligar deverá definir na implementação do método qual será o valor de dimerização desejado.

4.1.4 Serviços

Em AmIs os usuários não, necessariamente, estão interessados em saber quais dispositivos de automação estão instalados no local, eles normalmente desejam saber quais serviços estão disponíveis para utilização. Por exemplo: um serviço para "ventilar o ambiente" poderia ser oferecido tanto por um dispositivo "ventilador" quanto por um "ar condicionado". Nesse caso, o que o usuário deseja é ventilar o ambiente, independente de qual dispositivo irá executar essa ação. Dentro do contexto dos AmIs, a opção por executar a

tarefa através de um ou outro dispositivo, pode estar associada com economia de energia, em que o AmI poderá tomar a decisão baseando-se na premissa do dispositivo que ofereça menor consumo. Por outro lado, a criação dos serviços que compõem o AmI pode ser pré-definida, através da criação de um serviço que esteja associado com o dispositivo desejado. Nesse caso, quem tomaria a decisão de qual dispositivo utilizar seria o usuário (em tempo de projeto) e não o ambiente (em tempo de execução). Eventualmente, essa decisão pode ser colaborativa: o usuário decide, porém o AmI pode se adaptar, fornecendo ao usuário opções de configuração que irão atender à mesma tarefa com um menor consumo ou substituindo um equipamento falho.

Existem diversas estratégias relacionadas à utilização de serviços em AmIs e como os dispositivos de automação podem ser explorados de modo a oferecer algum benefício cooperando com outros dispositivos para atender a uma solicitação de serviço. Nesta abordagem a proposta é que os serviços possam ser criados pelo usuário, o qual possui total controle sobre as associações entre serviços e dispositivos de automação, ou seja: o usuário poderá criar os serviços e definir quais dispositivos serão utilizados para atender a esse serviço.

Conforme pode ser observado na Figura 4.10, os serviços são os componentes que utilizam os atributos e os métodos definidos pelas classes dos dispositivos de automação e parametrizados pelos dispositivos lógicos.

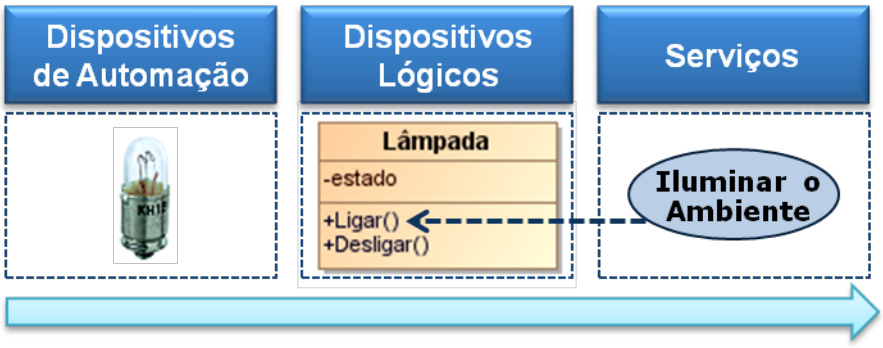

Figura 4.10 Criação de serviços.

A criação de um serviço pode exigir mais de um dispositivo de automação. Nesse caso, um mesmo serviço pode utilizar vários métodos de vários dispositivos distribuídos no AmI, por exemplo: a construção de um serviço chamado “Assistir à TV” pode ser composto pelos dispositivos TV, Persiana e Lâmpada, em que os métodos utilizados para executar essa tarefa seriam: “TV.Ligar()”, “Persianas.Fechar()” e “Lâmpada.Desligar()”. Ou seja: são três dispositivos, com três métodos independentes, porém combinados em um único serviço. Na Figura 4.11 é apresentado um diagrama ilustrando as classes e os relacionamentos existentes na camada de serviços.

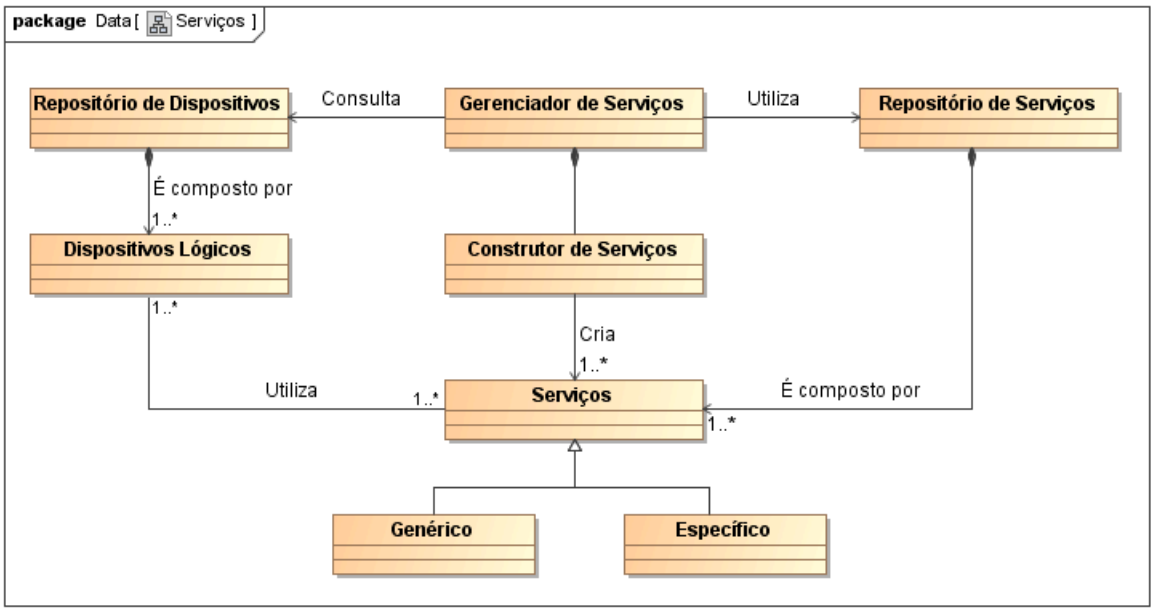

Figura 4.11 Classes da camada serviços.

A camada de serviços é composta por três componentes: (i) **Gerenciador de Serviços**: responsável por descobrir quais são os dispositivos lógicos que estão disponíveis para utilização, através de consultas que são realizadas em um repositório de dispositivos. Além disso, esse componente oferece um sistema de gerenciamento que possibilita identificar quais serviços já foram criados, acessando o repositório de serviços; (ii) **Construtor de Serviços**: é o principal componente dessa camada, pois ele permite a criação de um serviço através da seleção de um ou mais dispositivos de automação que estão disponíveis. No processo de criação do serviço, é possível escolher quais serão os dispositivos utilizados para realização

do serviço e quais métodos (funcionalidades) desses dispositivos serão utilizados. Além disso, o serviço pode ter uma descrição e uma classificação, por exemplo, um serviço com o nome *"iluminar o ambiente"* pode ser muito abrangente se existir mais de um ambiente com essa funcionalidade. Nesse caso, a descrição é importante para identificar o que realmente o serviço faz ou onde faz. No caso da classificação, é possível associar um serviço criado a um grupo de serviços, como entretenimento, segurança, conforto, emergência, etc. Isso pode facilitar a identificação dos serviços, através da utilização de filtros por grupos de serviços; Outro aspecto importante seria a criação de um serviço genérico para um dispositivo genérico, em que novamente utilizando polimorfismo seria possível ter várias instâncias desse serviço, um para cada ambiente, por exemplo: o serviço genérico *"iluminar o ambiente"* poderia ser utilizado em todas as instâncias da classe *"lâmpada"*. A diferença entre um serviço genérico e um serviço específico é que o primeiro não possui um dispositivo alocado, ele pode utilizar qualquer dispositivo que realize uma determinada tarefa, diferentemente do segundo, o qual é construído com base em um dispositivo; (iii) **Repositório de Serviços**: é responsável pelo armazenamento dos serviços que são criados e pela disponibilização de suas informações ao Gerenciador de Serviços. Esse repositório pode ser composto por dois tipos de serviços: (a) serviços internos: criados com o auxílio dos componentes que fazem parte do *framework* e acessam os dispositivos de automação do AmI, ou (b) serviços externos: referentes aos serviços que podem estar disponíveis remotamente e que não, necessariamente, estão associados com dispositivos de automação. Exemplos de serviços externos podem ser: consulta à previsão do tempo, informações do mercado financeiro, busca por novas mensagens em redes sociais, acesso a *web sites*, etc.

O padrão de interação e a troca de mensagens entre os componentes da camada de serviços podem ser observados na Figura 4.12, a qual ilustra a inicialização do processo de gerenciamento de serviços através de consultas que são realizadas aos repositórios de

dispositivos lógicos e de serviços. O resultado da primeira consulta disponibiliza a lista dos dispositivos que poderão ser utilizados na criação de serviços. Já o resultado da segunda consulta apresenta as lista dos serviços que já foram criados. Essa criação é dada através da seleção dos métodos de um ou mais dispositivos lógicos necessários para a execução de um serviço.

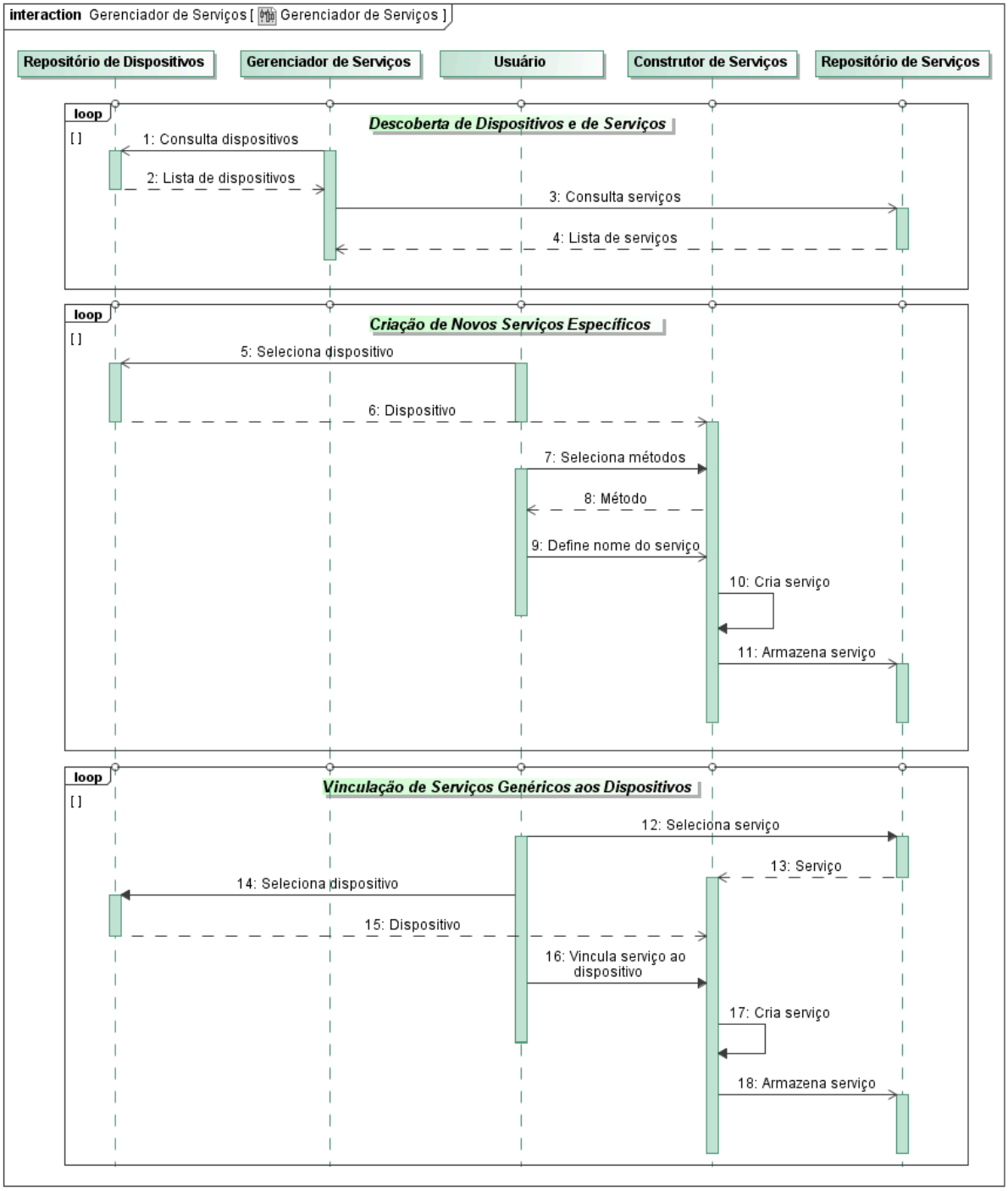

Figura 4.12 Gerenciamento de serviços.

4.1.5 Cenários

Em AmIs é comum a existência de diversos cenários que são construídos para atender a uma determinada tarefa. Os cenários podem ser definidos como uma sequência de ações (PETRIU, 2004), executadas de acordo com condições pré-definidas (ARAUJO, 2005). Como os AmIs podem estar em constante evolução, tanto no que se refere à infraestrutura de automação quanto na inserção de novas regras para o ambiente, é necessária a criação de novos cenários ou a atualização de cenários já existentes, buscando adequá-los aos novos requisitos que são apresentados. Os cenários correspondem às configurações de um ambiente e são construídos para atender a uma demanda, estando relacionados com os serviços automatizados do AmI. Esses cenários podem estar associados a serviços de segurança, conforto, entretenimento, entre outros. Além disso, a construção de um cenário pode agregar vários serviços e combiná-los, a fim de atender a uma demanda específica. Por exemplo, um cenário poderia ser criado para a necessidade de um usuário realizar leituras na sala de estar. Nesse caso, o cenário "Leitura" poderia ser composto por diversos serviços que estejam relacionados com a configuração daquele ambiente, o qual poderia *ajustar o nível de iluminação da sala*, *climatizar o ambiente*, *desligar o sistema de som* (caso estivesse ligado), além de diversos outros serviços.

Um cenário pode lembrar um serviço composto (PEROZZO & PEREIRA, 2008), entretanto, a ideia de cenário é a possibilidade de haver inúmeros serviços (simples ou compostos) agregados, com uma sequência lógica de execução pré-estabelecida, visando a atender aos desejos ou as necessidades de um usuário. Uma das vantagens na utilização de cenários é a possibilidade de configurar o ambiente através de um nível mais alto de abstração, em que o usuário não precisa, necessariamente, se preocupar com quais dispositivos de automação irão executar uma tarefa ou de que forma os serviços utilizaram esses dispositivos. Uma vez criados os serviços, qualquer nova configuração no AmI pode ser

realizada em nível de cenários, sem a necessidade de alteração na camada de serviços. A Figura 4.13 apresenta os cinco componentes que compõem a camada de cenários: (i) **Construtor de Cenários**: é o principal componente da camada, sendo o responsável pela construção dos cenários de automação que irão gerenciar o AmI. Esse componente permite que o usuário possa criar os seus cenários de acordo com os serviços disponíveis no AmI, com os seus desejos e com os requisitos do projeto; (ii) **Descoberta de Serviços**: é o componente responsável por identificar quais são os serviços que foram criados e estão disponíveis para utilização; (iii) **Requisitos**: corresponde ao conjunto de informações que são necessárias para a criação de um cenário de automação, o que pode incluir a identificação de perfis de usuários que utilizarão um cenário com determinados serviços, a atribuição de horários em que esses cenários estarão disponíveis para utilização, quais serviços serão acessados por quais interfaces de comando e regras condicionais baseadas no retorno de serviços como, por exemplo, ligar o aparelho de ar condicionado caso o sensor de temperatura apresente um valor superior a 24ºC, ou ligar a iluminação externa da casa após às 20 horas; (iv) **Repositório de Cenários**: é o componente responsável pelo armazenamento dos cenários que são construídos; (v) **Reutilizador de Cenários**: esse componente permite que todos os cenários que são criados possam ser reutilizados, ou seja, é possível criar um novo cenário através do reaproveitamento de algum cenário já criado.

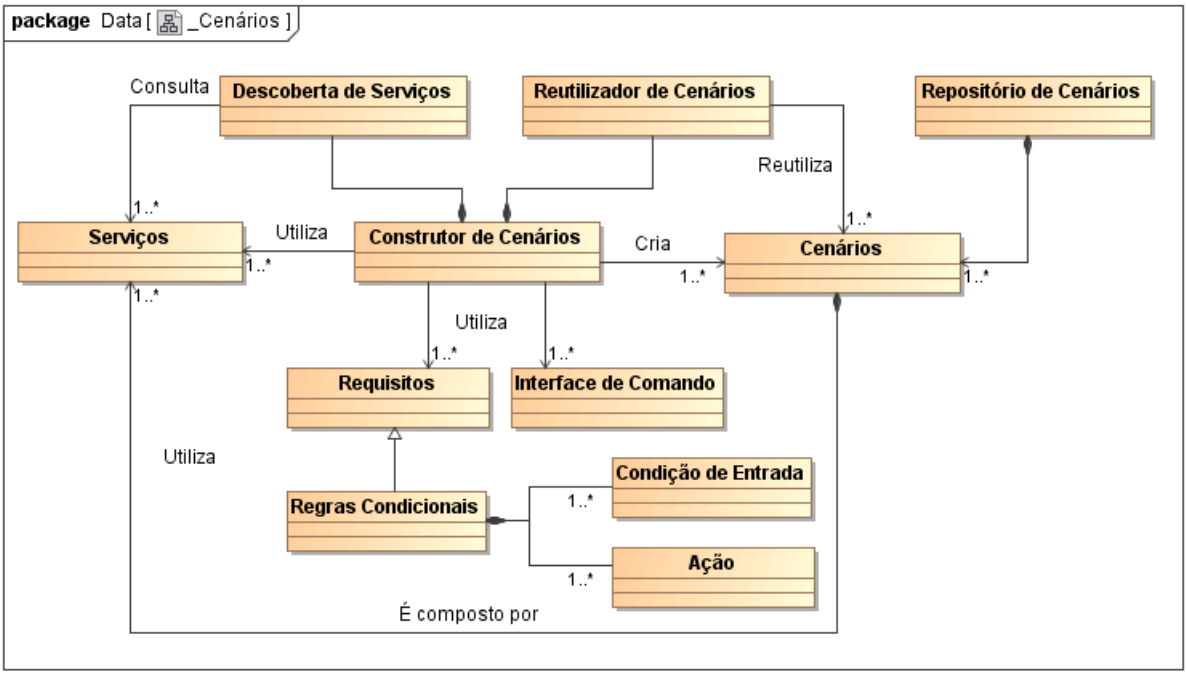

Figura 4.13 Diagrama de classes da "Camada Cenários".

A construção de um cenário, ilustrada na Figura 4.14a, é realizada através da descoberta dos serviços que estão disponíveis no AmI. No Construtor de Cenários, é possível selecionar quais serviços serão utilizados, considerando os requisitos do projeto, e verificar quais foram os cenários já criados. Cada serviço escolhido para compor o cenário é associado com uma interface de comando que faz parte dos requisitos, por exemplo: o serviço "Ligar Iluminação" poderia ser executado através da tecla de cor amarela do controle remoto da TV, ou, através de um comando de voz, captado por um microfone instalado no AmI. A interface de comando depende de quais dispositivos de interação existem no ambiente e quais poderão ser utilizados para essa finalidade. Após um cenário ter sido criado, ele é disponibilizado ao Repositório de Cenários, para ser utilizado pelos outros componentes da arquitetura do *framework*, os quais irão transformar os cenários em aplicações, codificando-os em uma linguagem de programação compatível com a plataforma-alvo escolhida para gerenciamento do AmI. Similar ao que ocorre na construção de cenários, o processo de reutilização, ilustrado na Figura 4.14b, identifica os cenários construídos e seleciona um para a edição. A edição permite que os serviços possam ser alterados ou excluídos do cenário. Após esse

procedimento, o cenário editado é disponibilizado ao Construtor de Cenários para que possam ser adicionados novos serviços ao cenário original, gerando um novo cenário.

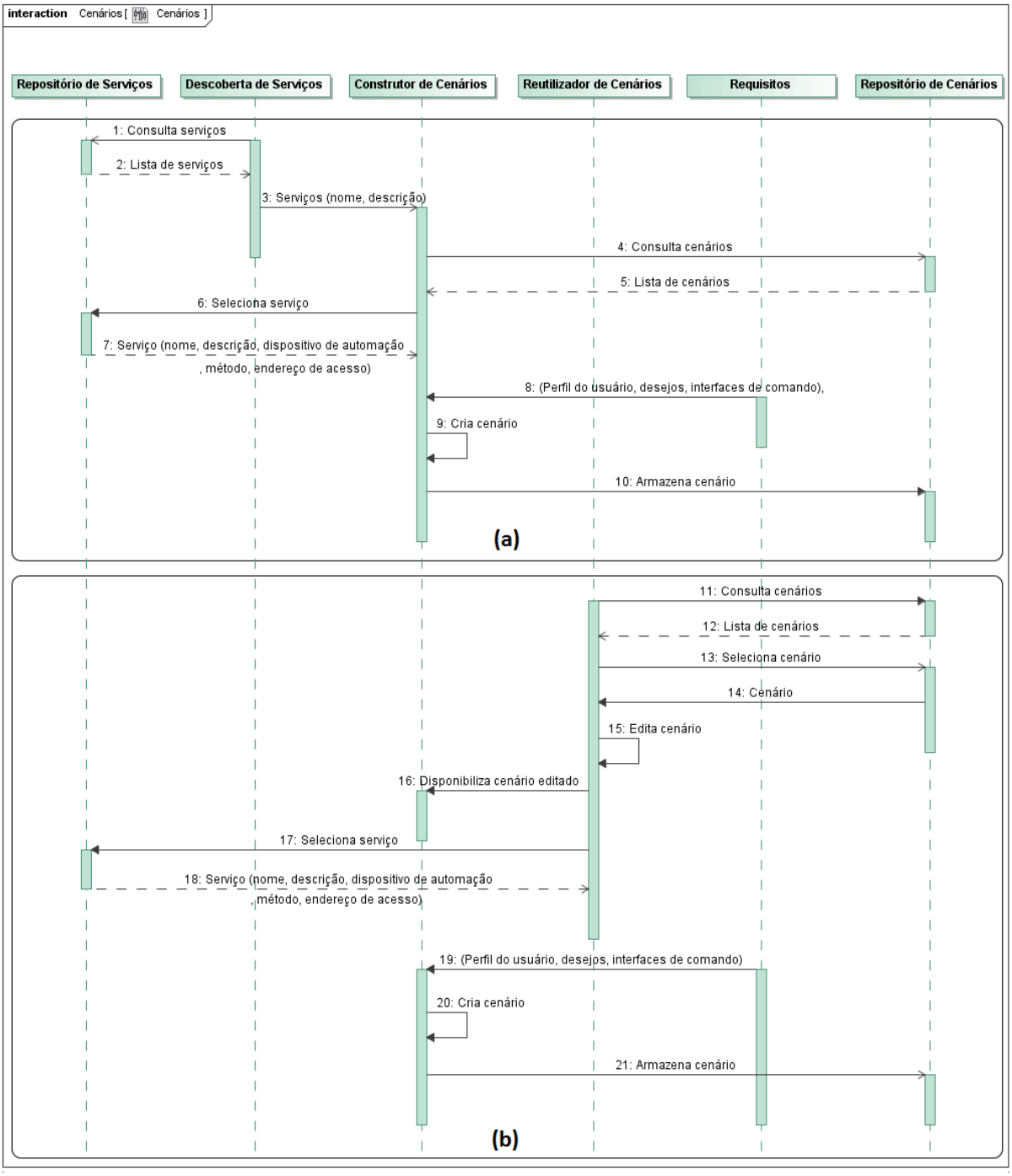

Figura 4.14 Construção de cenários.

A reutilização de cenários permite agilizar a construção de novos cenários, principalmente, quando o cenário a ser criado é muito parecido com algum já existente. Por exemplo, na Figura 4.15, há um cenário (A) construído para o ambiente da sala de estar,

sendo composto pelos serviços “Desligar TV”, “Desligar Iluminação” e “Ouvir Música”. Porém, o usuário deseja criar um novo cenário (B), muito similar ao existente, cuja única diferença seria substituir o serviço “Ouvir Música” pelo serviço “Climatizar Ambiente”. Nesse caso, o cenário B será o resultado da edição do cenário A, somado a um novo serviço escolhido no Repositório de Serviços.

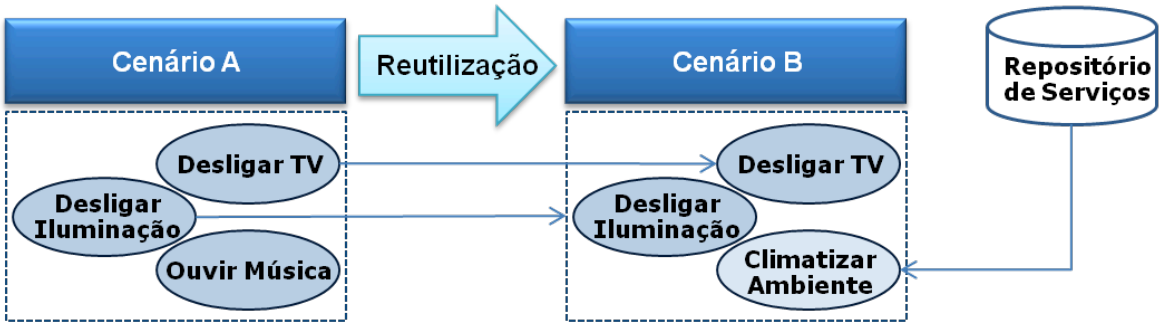

Figura 4.15 Reutilização de cenários.

Uma das vantagens obtidas nessa abordagem é a capacidade de serem construídos cenários independentes de plataformas de *software* e de *hardware*, o que possibilita maior flexibilidade e heterogeneidade na arquitetura proposta neste trabalho. Os cenários são considerados pelo *framework* como um conjunto de regras a serem executadas, ou seja: um cenário contém as informações de quais serviços ele utiliza, quais são as descrições desses serviços e a que grupo eles pertencem, além das informações sobre os endereços de acesso aos serviços e as suas respectivas interfaces de comando para utilização no AmI. Com base nessas informações, é possível, codificar um mesmo cenário em diversas linguagens de programação, que sejam suportadas por diferentes plataformas-alvo de gerenciamento do AmI, que não, necessariamente, sejam televisores ou receptores de TV digital. Ou seja, até esse momento, a arquitetura do *framework* proposto pode ser utilizada para criar cenários de gerenciamento de AmIs para qualquer plataforma computacional. Entretanto, nas próximas seções são apresentados os componentes que oferecem os recursos para permitir que os cenários de gerenciamento da automação do AmI possam ser integrados com as plataformas do SBTVD.

4.1.6 Geração de Código

Uma das principais camadas da arquitetura do *framework* proposto refere-se à Geração de Código, responsável por transformar em código de programação os cenários de automação construídos na camada anterior. O processo de codificação de um cenário envolve alguns desafios, como, por exemplo, (i) obter as informações contidas nos cenários, transformando-os em aplicações através da codificação desses cenários em alguma linguagem de programação e (ii) identificar os perfis do *hardware* e do *middleware* de interatividade compatível com o SBTVD, a fim de permitir a aderência entre a aplicação gerada e a plataforma-alvo escolhida para gerenciamento do AmI.

Na camada de Geração de Código existem quatro componentes, ilustrados em azul na Figura 4.16, que são os responsáveis pelo processo de criar aplicações interativas de gerenciamento do ambiente, são eles: (i) **Gerador de Código**: responsável por codificar os cenários em uma linguagem de programação, criando as aplicações para gerenciamento do AmI. Esse componente recebe as informações sobre o perfil da plataforma-alvo que executará a aplicação, bem como o perfil do *middleware* de interatividade presente nessa plataforma. Uma das características desse componente é o fato de ele utilizar uma biblioteca de códigos-fonte que contém funções escritas em uma linguagem compatível com o *middleware* de interatividade e que serão parametrizadas de acordo com as configurações de cada cenário. Para a construção de uma aplicação, o Gerador de Código oferece, no momento da geração, a possibilidade de escolher qual será o cenário codificado, em qual linguagem de programação e para qual plataforma-alvo. Além disso, é possível associar a cada cenário uma interface com objetos gráficos que representam ou correspondem ao cenário escolhido; (ii) **Perfil do *Middleware***: responsável por prover, ao Gerador de Código, as informações necessárias sobre o *middleware* de interatividade para o qual as aplicações são geradas. No SBTVD há dois perfis de *middlware:* um para aplicações declarativas (Ginga-NCL) e outro para aplicações

procedurais (Ginga-J). Dessa forma, esse componente armazena e disponibiliza a biblioteca de códigos-fonte que é utilizada na geração das aplicações interativas para gerenciamento do AmI. De um modo geral, a biblioteca de códigos-fonte é composta por dois *templates* de códigos (um para a linguagem procedural e outra para a declarativa) que recebem os parâmetros de configuração do cenário a ser gerado; (iii) **Plataforma Alvo**: responsável por fornecer as informações sobre o *hardware* que irá executar a aplicação interativa. No Brasil, a interatividade na TV pode ser obtida tanto em receptores fixos (TVs e STBs) quanto móveis (telefones celulares). As normas brasileiras (GINGA, 2010) definem que todos os receptores fixos que são fabricados deverão, obrigatoriamente, ser compostos pelos dois tipos de perfis (Ginga-NCL e Ginga-J). Já para os receptores portáteis, a obrigatoriedade está associada a apenas um perfil, o Ginga-NCL. Ou seja, um mesmo cenário pode ser gerado para um dos perfis ou ambos. Além das informações de qual perfil de *middleware* está presente na plataforma-alvo que receberá a aplicação, o Gerador de Código necessita identificar outras informações do *hardware*, como, por exemplo, a resolução da tela. Essas informações são extremamente importantes, pois elas permitem a parametrização correta dos *templates* de código fonte que darão origem à aplicação. Por exemplo, a resolução de tela de um televisor é totalmente diferente da resolução de um telefone celular. Mesmo os dois receptores em questão sendo compatíveis com o perfil Ginga-NCL e a geração de código ser a mesma, os parâmetros de largura e altura da interface gráfica para exibição na tela do receptor são diferentes. Ou seja, a aplicação gerada para um STB ficará com a interface gráfica maior do que se fosse gerada para um telefone celular. Resguardadas as devidas proporções da interface gráfica, a mesma aplicação poderia ser executada em ambos os receptores, pois a codificação por trás da interface é idêntica, devido à compatibilidade dos *templates* de código fonte com as linguagens presentes no SBTVD; (iv) **Biblioteca de Objetos Gráficos**: esse componente corresponde ao repositório de objetos que compõem a interface gráfica da aplicação de

gerenciamento do AmI. A interface oferece as informações visuais dos serviços e dos seus respectivos comandos de acionamento para cada cenário.

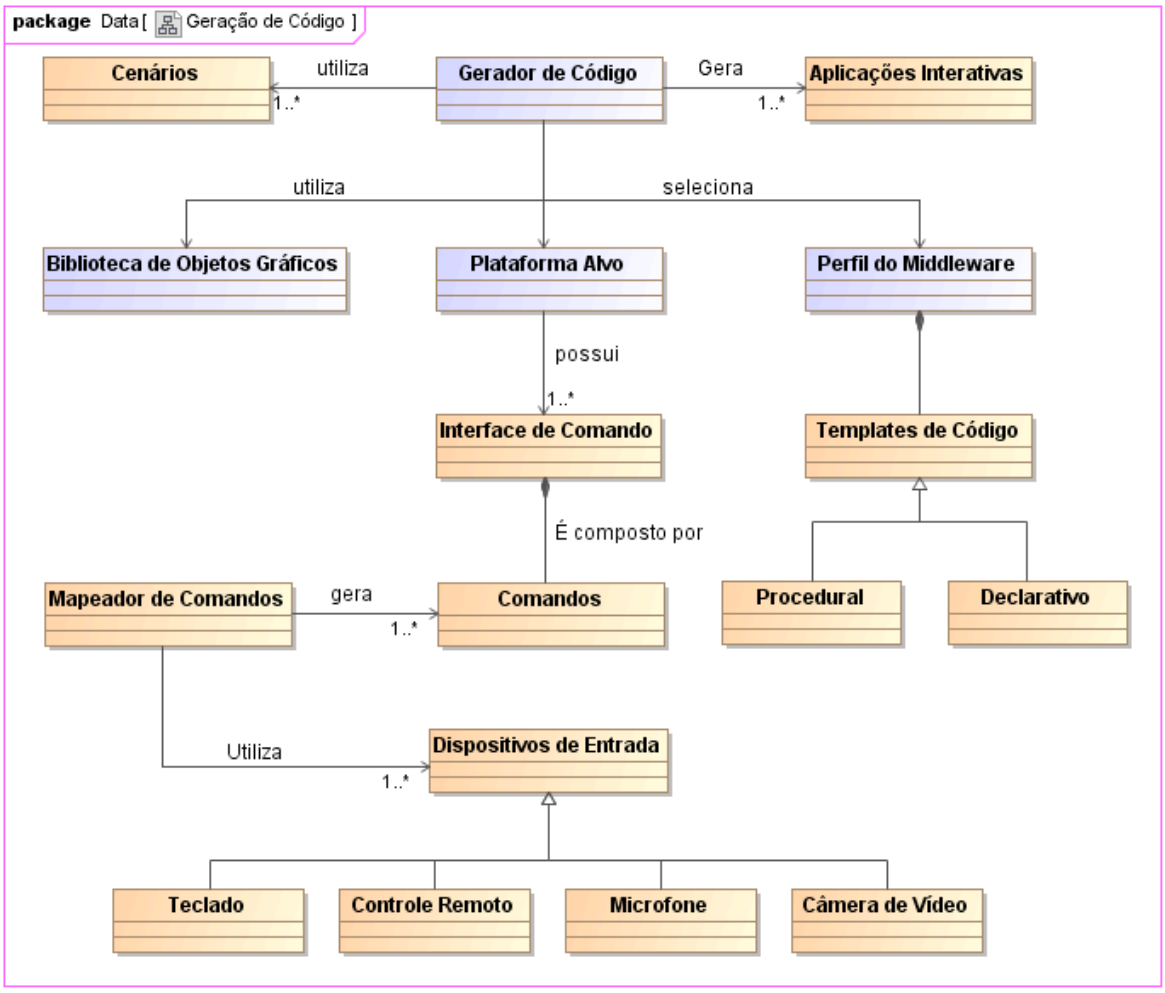

Figura 4.16 Componentes para geração de código.

As representações gráficas que fazem parte da interface da aplicação podem ser criadas pelo próprio usuário, utilizando um editor de imagens e gerando figuras que correspondam às funcionalidades oferecidas pelo cenário, ou imagens criadas por terceiros e que estejam disponíveis para utilização na biblioteca de objetos gráficos.

Em relação à plataforma-alvo, esta pode ser especializada em outras classes e subclasses, tais como: “plataforma-alvo fixa”, com a subclasse “STB” e “plataforma móvel”, com a subclasse “telefone celular”, sendo que cada uma dessas classes e subclasses possuem características diferentes uma da outra. Esse refinamento pode ser realizado por diversas vezes até ser especializada uma classe que mais se aproxima das características físicas do

dispositivo presente no mundo real. Além disso, conforme apresentado na Figura 4.16, a plataforma-alvo possui uma interface de comando composta por todos os comandos disponíveis em seus dispositivos de entrada e mapeados para serem utilizados como interface de interação entre o usuário e o receptor de TV digital. Por exemplo, o serviço de "iluminar o ambiente" presente em um cenário de automação pode ser executado em um STB através do pressionar de uma tecla do controle remoto ou, através de um comando de voz que foi capturado pelo microfone do ambiente e mapeado para o STB. Obviamente, os dispositivos de entrada a serem utilizados como interface de comando devem ser conhecidos pela plataforma-alvo. Os dispositivos de entrada mais comuns encontrados para interação com receptores de TV são os controles remotos e os teclados.

Entretanto, a utilização de reconhecimento de voz e de gestos depende do nível de integração entre essas áreas. Provavelmente, seria necessário incluir no AmI um sistema computacional capaz de mapear esses comandos e criar uma interface com os receptores de TV digital, ou adicionando a estes tais funcionalidades. De um modo geral, ou a interface de comando é suportada pela plataforma-alvo, ou ela pode ser suportada por outra plataforma que seja capaz de oferecer essas funcionalidades, as quais devem ser processadas e mapeadas de uma forma que o receptor de TV digital consiga interpretar o comando e executar o serviço. Isso poderia ser considerado como uma espécie de colaboração distribuída entre os dispositivos do AmI.

Para ilustrar a troca de mensagens entre os componentes da camada de geração, é possível observar na Figura 4.17 que a sequência de eventos é iniciada pelo componente Gerador de Código, o qual seleciona um cenário a ser codificado. Em seguida, é escolhido o perfil do *middleware* de interatividade a ser utilizado. O resultado dessa escolha é um *template* de código que corresponde à linguagem declarativa ou procedural desejada para codificação. No evento seguinte são obtidas as informações da plataforma-alvo, ou seja, as

características do *hardware* que irá executar a aplicação. Por fim, são selecionados os objetos gráficos que irão compor a interface da aplicação e, então, é gerada a codificação do cenário, transformando-o em uma aplicação interativa que irá gerenciar o AmI.

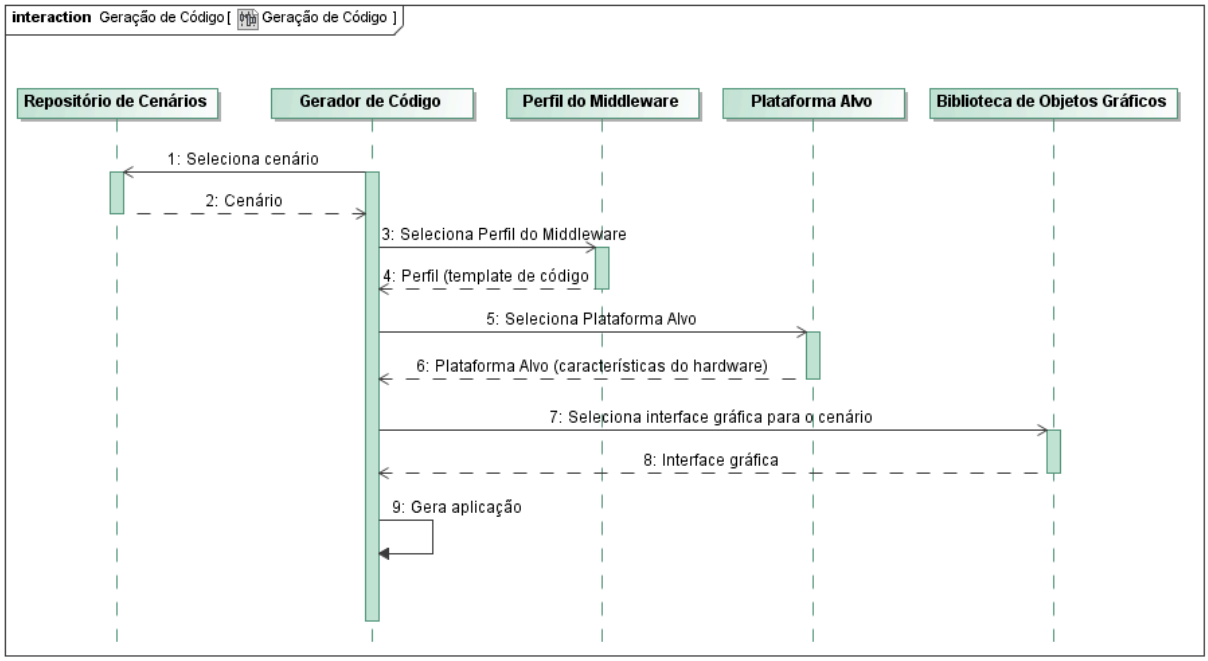

Figura 4.17 Sequência de eventos para construção de aplicações.

4.1.7 Aplicações Interativas

A interatividade da TV digital brasileira é, atualmente, o padrão recomendado pelo ITU (ITU H.761, 2009), o que pode alavancar ainda mais a adoção do SBTVD em outros países fora da América Latina. Talvez, a interatividade seja uma das principais contribuições no padrão brasileiro frente aos demais sistemas existentes, podendo pesar favoravelmente no momento em que outros países forem escolher um sistema de TV digital. Como já foi mencionado anteriormente, os benefícios da interatividade na TV são muitos, pois além da decodificação do sinal de TV, os receptores podem executar aplicações interativas que são criadas para as mais diversas áreas, desde aplicações com serviços do governo para inclusão social e digital até aplicações customizadas para realização de um determinado serviço. Essa possibilidade é fortemente explorada na proposta deste trabalho, pois a última camada da

estrutura do *framework* corresponde às aplicações interativas. Essa camada é a responsável pelas aplicações que irão gerenciar o AmI, através dos receptores de TV digital, o que inclui TVs, STBs e telefones celulares compatíveis com o SBTVD.

As aplicações interativas são os resultados dos cenários que foram criados anteriormente somados ao processo de codificação, ou seja, uma aplicação interativa de gerenciamento do AmI nada mais é do que um cenário codificado em uma linguagem de programação suportada pelo *middleware* de interatividade da TV digital brasileira.

Quando a aplicação interativa é criada, ela pode ser automaticamente carregada e executada pelo receptor de TV digital. Existem, basicamente, três formas de disponibilizar a aplicação em um receptor: (i) instalada de fábrica; (ii) carregada através de uma memória externa (*pen drive*) e (iii) transmitida pela emissora de TV através do ar. Mesmo sendo a opção de memória externa mais acessível para carregar aplicações customizadas, como é o caso de aplicações para AmI, em que cada residência possui as suas próprias particularidades e tipos de sistemas de automação, nada impede que as emissoras de TV ou fabricantes disponibilizem essas mesmas aplicações pelos seus meios, seja na fabricação dos receptores ou na transmissão do sinal.

Considerando que aplicação interativa estará disponível nos receptores de TV digital, os usuários poderão iniciar o gerenciamento do seu AmI de acordo com cada cenário que foi construído para essa finalidade. Os receptores podem conter inúmeras aplicações interativas/ cenários, voltados para atender a uma determinada tarefa ou demanda do AmI. A quantidade de aplicações disponíveis no receptor depende, única e exclusivamente, de sua capacidade de armazenamento.

Além das aplicações interativas de automação que são criadas *a priori*, o *framework* proposto neste trabalho poderia permitir que os cenários que dão origem a essas aplicações pudessem ser criados em tempo de execução, ou seja, o componente **Construtor Dinâmico**

de Cenários (ilustrado na Figura 4.5, que apresenta a arquitetura da proposta) permitiria que novos cenários pudessem ser criados dinamicamente pelo próprio receptor de TV, através da descoberta de novos serviços e da geração de novas aplicações interativas. Nesse caso, a ideia seria levar o construtor de cenários do tempo de projeto para o tempo de execução. Esse construtor dinâmico de cenários seria uma aplicação interativa com a capacidade de gerar outras aplicações interativas. O desafio nesse caso se daria no mérito de os receptores de TV digital terem a capacidade computacional e os *softwares* necessários para executar os componentes propostos neste *framework*.

Seja em um computador ou em um receptor de TV, a criação de uma aplicação interativa para gerenciamento do AmI depende das camadas anteriores e é dada em fases, conforme ilustra a Figura 4.18:

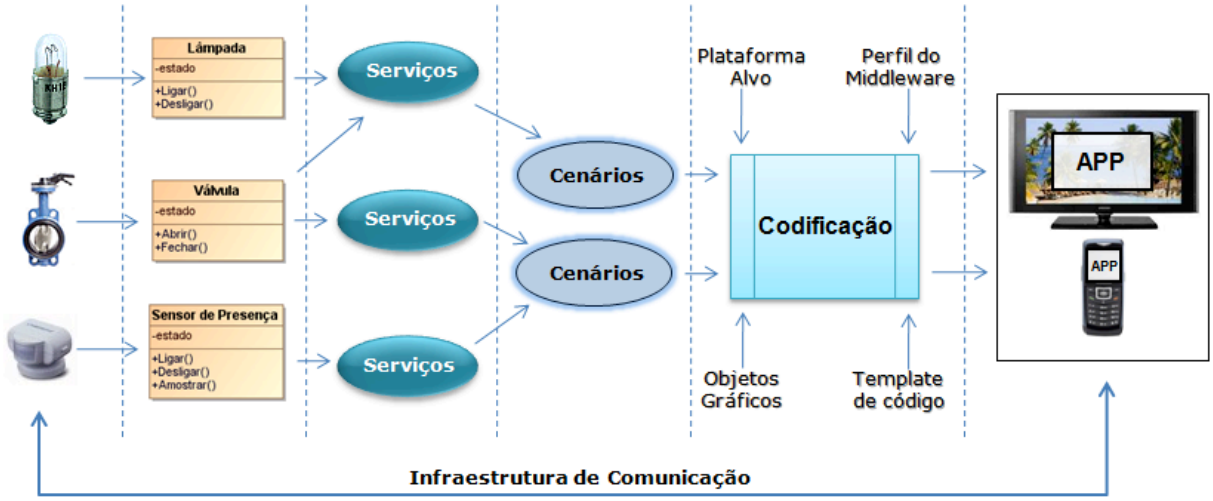

Figura 4.18 Fases até a criação de uma aplicação interativa.

Conforme ilustra a Figura 4.18, existem diversas fases até a construção de uma aplicação interativa, iniciando no processo de modelagem dos dispositivos de automação que fazem parte do ambiente, passando pela criação de serviços e construção de cenários, até chegar ao processo de codificação e de execução da aplicação. Além da distribuição dos componentes do *framework* - os quais podem fazer estar alocados em uma ou em muitas plataformas computacionais criando uma estrutura de comunicação distribuída de uma

estrutura local ou remota – a infraestrutura de comunicação de dados também é extremamente importante nesta abordagem pelo fato de ela permitir a comunicação entre os receptores de TV digital e os dispositivos de automação do AmI.

4.1.7.1 Infraestrutura de Comunicação e Canal de Interatividade

As aplicações interativas se beneficiam da interface de comunicação de dados presente nos receptores de TV digital e da infraestrutura de comunicação existente no AmI. Normalmente, os AmIs possuem algumas redes de comunicação heterogêneas, as quais podem ser cabeadas, sem fio ou ambas. Em alguns casos, é comum serem encontrados sistemas de comunicação que utilizam a rede elétrica para transmissão de dados e acesso à Internet (REBOUÇAS, 2011). Assim como ocorre com os PCs, telefones celulares e câmeras de monitoramento que podem estar constantemente conectados às redes, os receptores de TV seguem o mesmo princípio, ainda mais com os avanços tecnológicos relacionados com as TVs conectadas ou *broadband* TVs, identificada como um nicho de mercado em potencial e explorada por grandes fabricantes de TVs (TERRA TECNOLOGIA, 2010) e empresas de tecnologia da informação, como é o caso da Google, que apresenta um equipamento (GOOGLE TV, 2010) com uma única proposta: realizar a convergência entre computador e TV. Especificamente para o SBTVD, os receptores fixos de TV comercializados atualmente possuem uma interface *Ethernet* que pode ser conectada à rede local de um AmI e, consequentemente, à Internet. Já os receptores móveis, utilizam a rede de dados 3G disponibilizada pelas operadoras de telecom. É por essa rede 3G que pode ser realizada a interatividade em telefones celulares que recebam o sinal da TV digital e possuam o *middleware* Ginga.

A interatividade no SBTVD é classificada de duas formas: (i) parcial: quando o receptor de TV digital possui o *middleware* de interatividade, executa a aplicação interativa e, por algum motivo, não é utilizada uma rede de comunicação para troca de dados, seja pelo

fato de a aplicação não utilizar canal de retorno ou o receptor não ter essa funcionalidade, por exemplo: a emissora de TV transmite o aplicativo de interatividade pelo ar, juntamente com o sinal de áudio e vídeo, e o telespectador apenas interage com a aplicação localmente, navegando e selecionando opções que já estão na aplicação e; (ii) plena: quando o receptor possui o *middleware*, executa a aplicação e troca dados com outro equipamento, em uma rede local ou na Internet, como, por exemplo, servidores para receberem a votação de uma enquete esportiva. A emissora de TV transmite o aplicativo pelo ar e o telespectador retorna o seu voto através da Internet.

A forma de interatividade utilizada neste trabalho é a plena, pois as aplicações interativas de gerenciamento do AmI, que são executadas nos receptores de TV digital, necessitam se comunicar com os dispositivos de automação que estão presentes no ambiente. Dessa forma, os receptores devem estar interligados e possuir acesso à rede do sistema de automação que se deseja gerenciar. Muitos dos sistemas de automação predial/residencial comercializados atualmente oferecem suporte à comunicação TCP/IP, protocolo que é utilizado pelos receptores de TV digital para comunicação de dados.

4.2 IMPLEMENTAÇÃO

Nessa subseção é apresentada a implementação dos componentes de *software* que fazem parte da arquitetura do *framework* proposta na subseção anterior. No modelo conceitual, foram discutidos os aspectos relacionados com o que deveria ser feito a fim de integrar os AmIs com o SBTVD. Sob a ótica da implementação, o importante é destacar como essa integração é realizada e quais tecnologias podem ser utilizadas para atingir esse objetivo. Assim, nas próximas subseções são apresentados a arquitetura de implementação do modelo conceitual e os componentes de *software* desenvolvidos, os quais estão disponíveis para *download* no *website* oficial do trabalho: **http://sourceforge.net/projects/dtvi4ha**.

4.2.1 Arquitetura de Implementação

A arquitetura de implementação tem por objetivo apresentar quais tecnologias são utilizadas no trabalho para validar experimentalmente o modelo conceitual. Em linhas gerais, a arquitetura de implementação é similar à arquitetura conceitual. A diferença básica entre elas é que a arquitetura de implementação busca definir, para cada componente teórico apresentado no modelo conceitual, (i) quais são as linguagem de programação utilizadas na implementação dos componentes de *software*, (ii) quais são os sistemas para armazenamento de informações, e (iii) que padrões e tecnologias são especificados para troca de dados entre os componentes da arquitetura.

Quando se trata da implementação de uma arquitetura computacional composta, basicamente, por componentes de *software,* podem existir diversas formas, técnicas e tecnologias para implementar um componente. Por exemplo, considerando que em uma arquitetura de *software* exista um componente responsável por realizar uma conexão com outro componente e transmitir um valor referente ao estado atual de um sensor, utilizando *sockets*, esse componente poderia ser implementado tanto em linguagem C++, sobre uma plataforma Linux, quanto em Java, sobre uma plataforma Mac. O que define quais linguagens e tecnologias devem ser utilizadas para a implementação de uma arquitetura de *software* é, fundamentalmente, a composição de alguns fatores, tais como os requisitos do projeto, os recursos oferecidos pela tecnologia escolhida, a familiaridade do desenvolvedor com a linguagem de programação, entre outros.

Para a implementação deste trabalho, foram escolhidas algumas tecnologias, selecionadas de acordo com a necessidade de cada componente da arquitetura, por exemplo: o processo de modelagem do AmI é realizado através da linguagem UML, responsável por

mapear para o mundo computacional os dispositivos de automação presentes no mundo real. Essa escolha se justifica pelo fato de a UML ser um padrão adotado internacionalmente para modelagem de sistemas e pela familiaridade do grupo de pesquisa com essa tecnologia. Quanto aos componentes de *software*, eles foram implementados em linguagem Java, pois ela oferece portabilidade do código gerado. Isso permite que os componentes presentes na arquitetura do *framework* proposto tenham interoperabilidade entre diferentes plataformas computacionais. No que se refere ao armazenamento e compartilhamento de informações entre os componentes da arquitetura, são utilizadas as tecnologias XML e XML *Metadata Interchange* (XMI), ambas criadas pela *Object Management Group* (OMG, 2010). A XML foi criada com o objetivo de ser uma linguagem lida por outros *softwares* e passível de integração com outras linguagens de programação. Uma de suas principais características é a flexibilidade na separação entre o conteúdo e a apresentação desse conteúdo, sendo utilizada também para armazenamento de informações. No caso da XMI, ela permite a troca de informações entre ferramentas de modelagem e é baseada na linguagem XML. O seu objetivo é permitir a troca de modelos UML entre as ferramentas de modelagem (PENDER, 2004). Entretanto, um arquivo XMI gerado por uma ferramenta UML pode ser utilizado por outros *softwares* para compartilhar informações que sejam de interesse comum.

Dessa forma, a Figura 4.19 ilustra a arquitetura de implementação da proposta.

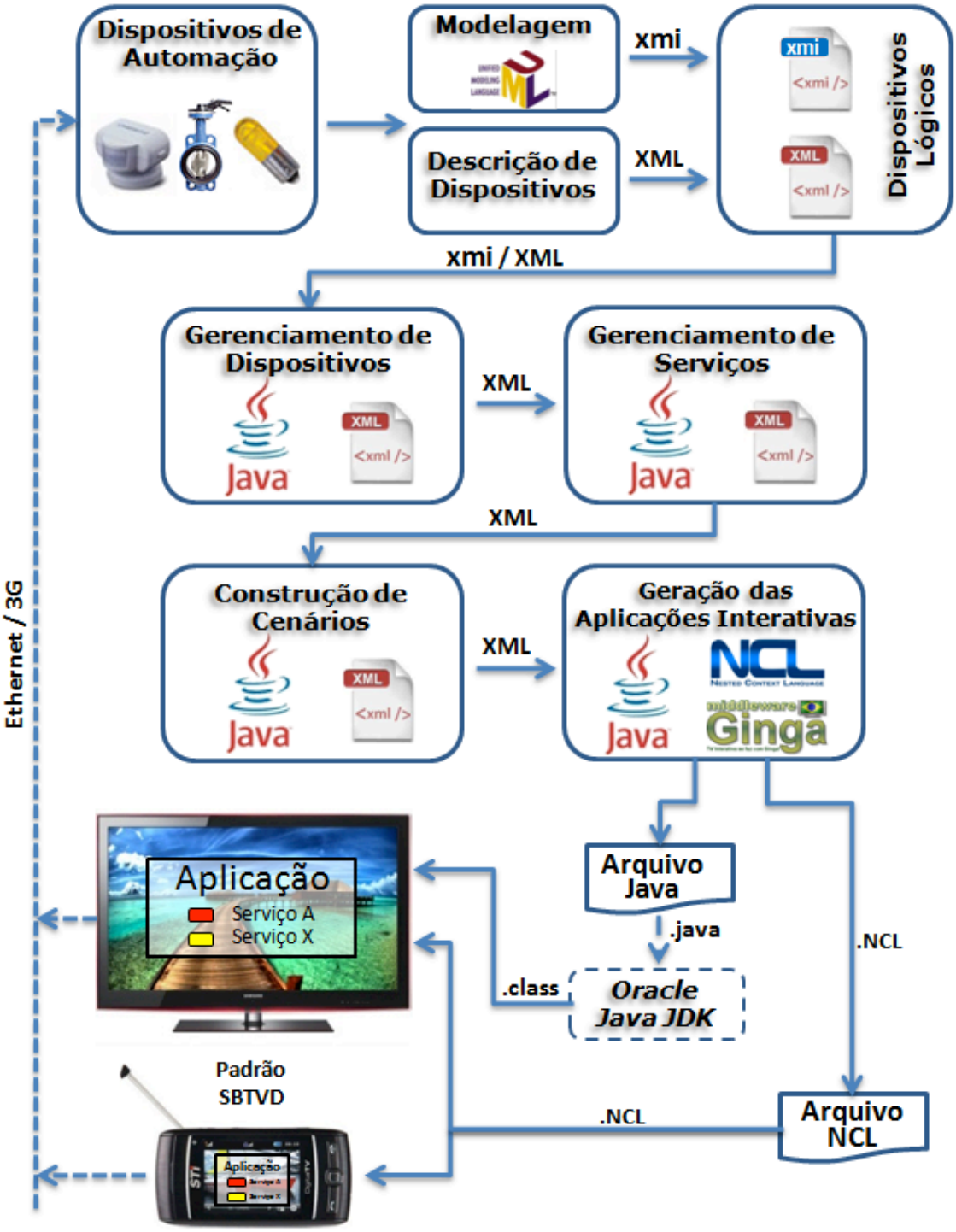

Figura 4.19 Arquitetura de implementação do *framework*.

4.2.2 Módulos Desenvolvidos

4.2.2.1 Mapeamento Computacional do AmI

O mapeamento computacional é, conforme discutido no modelo conceitual, a forma de se representarem computacionalmente as características e funcionalidades dos dispositivos de

automação presentes no AmI. Neste trabalho, o mapeamento pode ser realizado de duas formas: (i) utilizando uma ferramenta de modelagem UML para representar, através de modelos orientados a objetos, todos os dispositivos que serão utilizados pelo *framework*; (ii) utilizando uma editor de texto para descrever em linguagem XML esses dispositivos. A primeira opção permite que sejam desenhadas as classes correspondentes a cada um dos dispositivos de automação e gerados os arquivos XMI, os quais contêm as informações dos desenhos criados pela ferramenta UML, porém disponível num formato textual, baseado em uma árvore XML. A própria ferramenta de modelagem UML (MAGICDRAW, 2010) utilizada pelo *framework* permite exportar os modelos orientados a objetos para um formato XML. A segunda opção permite que seja realizado o mapeamento computacional dos dispositivos de automação sem, necessariamente, utilizar uma ferramenta de modelagem. Nesse caso, o *framework* recebe as informações dos dispositivos que serão utilizados através da edição de arquivos XML, os quais contêm a descrição das classes, atributos e métodos de cada dispositivo. Um exemplo dessa abordagem pode ser observado nas Figuras 4.20 e 4.21. Na primeira figura, um dispositivo real de automação é mapeado para o mundo computacional com o auxílio de uma ferramenta de modelagem, a qual permite que seja criada uma classe de dispositivo chamada “Lâmpada” a qual é exportada para um formato de arquivo XML que ficará disponível para os demais componentes do *framework.* Já na segunda figura, o mapeamento computacional é realizado diretamente por um editor de textos, responsável também pela criação de um arquivo XML que contém as informações sobre as propriedades do dispositivo de automação que se está mapeando.

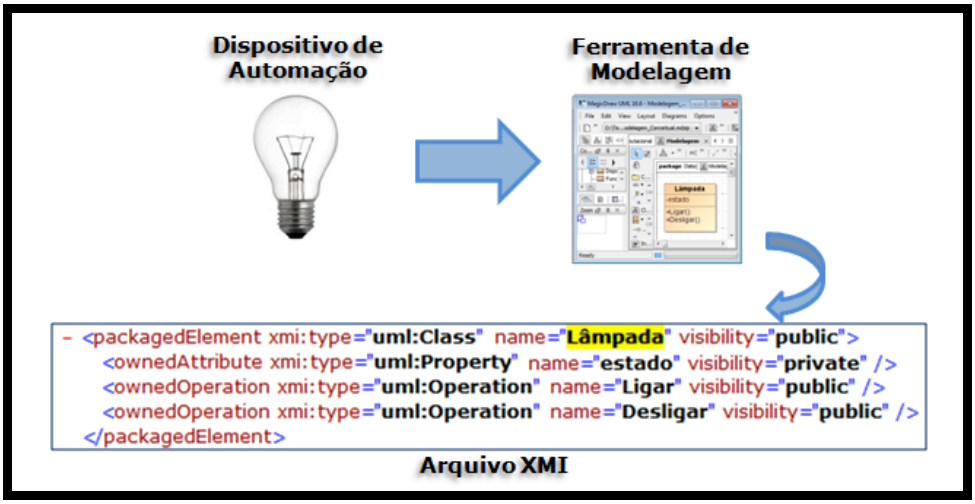

Figura 4.20 Mapeamento através de uma ferramenta de modelagem UML.

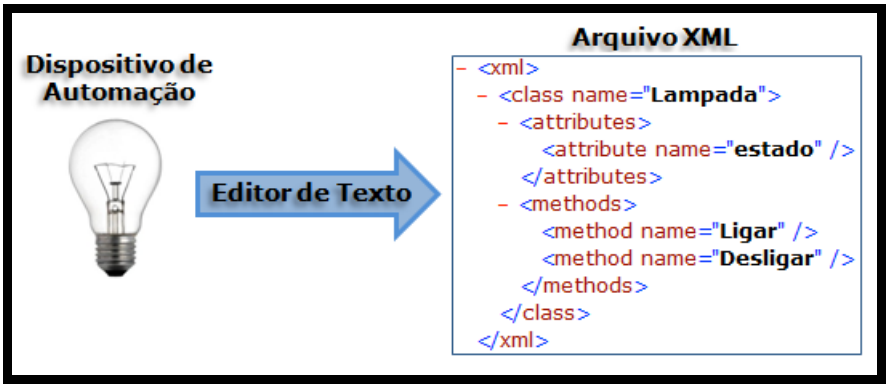

Figura 4.21 Mapeamento através de um editor de textos.

Ambas as soluções consideradas irão originar um arquivo XML para troca de dados com os outros componentes do *framework*. Entretanto, o mapeamento através de um editor de textos deve ser realizado seguindo o padrão especificado para o arquivo, sendo preenchidos os elementos *<class name>*, *<attributes>* e *<methods>*, pois são esses os elementos que serão lidos pelo componente de descoberta de dispositivos, no módulo **Gerenciamento de Dispositivos**. No caso de ser utilizada uma ferramenta de modelagem, o componente de descoberta de dispositivos também é capaz de identificar os elementos *uml:Class*, *uml:Property* e *uml:Operation* presentes no arquivo XMI que é exportado pela ferramenta UML.

4.2.2.2 Gerenciamento de Dispositivos

O módulo desenvolvido para o gerenciamento dos dispositivos de automação que fazem parte do AmI é formado por quatro componentes: descoberta, gerenciador, instanciador e repositório. A ferramenta de *software* desenvolvida para esse módulo agrupa os quatro componentes e permite que sejam realizadas as ações de cada um dos componentes. A implementação do componente **Descoberta de Dispositivos** permite que sejam obtidas e identificadas todas as classes dos dispositivos de automação presentes no AmI e que foram mapeados para o mundo computacional, através da ferramenta de modelagem ou da descrição textual dessas classes. Inicialmente, esse componente realiza buscas no repositório em que se encontram os arquivos XML resultantes do processo de mapeamento. Com base nessa busca, são identificados os nomes, os atributos e os métodos de cada classe de dispositivo de automação disponíveis para utilização. A descoberta dos dispositivos é realizada nessa implementação através da leitura de arquivos XML, entretanto, esse componente poderia utilizar outros mecanismos para descoberta, tais como: (i) utilizar um protocolo específico para descoberta de dispositivos, como é o caso do *HomeSystems Network* (HSNET) - um protocolo proprietário desenvolvido por uma empresa de automação residencial (HOMESYSTEMS, 2010) e que utiliza como base uma rede RS-485; (ii) utilizar o *Device Profile for Web Services* (DPWS) que é uma abordagem similar ao *Universal Plug and Play* (UPnP) e define um conjunto de implementações para permitir a integração entre serviços de dispositivos. Seguindo a ideia de (GAMMA *et al.*, 2000) a arquitetura de um *framework* tem que ser flexível e extensível para suportar diversas aplicações de um domínio. Dessa forma, o componente de descoberta de dispositivos poderia utilizar diferentes padrões de descoberta, necessitando apenas implementar as funções para cada tecnologia escolhida. Isso permite que os componentes do *framework* possam se adequar a novos requisitos e realizar as funcionalidades necessárias conforme o AmI vai evoluindo ao passar do tempo. Na Figura

4.22, é ilustrada a busca pelas classes de dispositivos em um repositório de arquivos XML (implementado) e a possibilidade de se utilizar outros padrões para descoberta de dispositivos sem, necessariamente, terem passado pelo processo de mapeamento computacional (extensível).

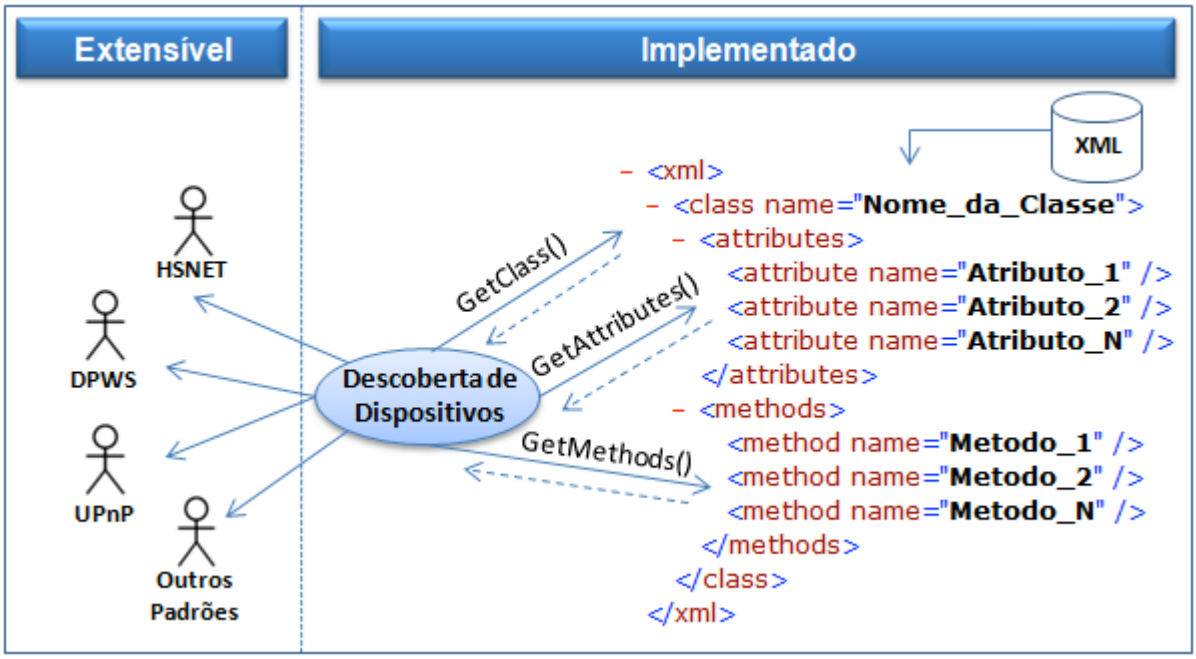

Figura 4.22 Descoberta das classes de dispositivos.

O resultado da descoberta permite que sejam identificados quais dispositivos de automação poderão ser utilizados e quais são as características de cada uma dessas classes de dispositivos. Para serem utilizados os dispositivos de automação, é necessário instanciar as classes descobertas, criando os objetos (dispositivos lógicos). Para isso, é implementado o componente **Instanciador de Dispositivos**, o qual recebe as informações dos dispositivos que foram descobertos e permite que os objetos sejam criados com base nas características de cada classe de dispositivos. Esse componente possui uma interface em que é possível parametrizar o nome do objeto, os atributos e os métodos que serão definidos para aquele dispositivo lógico. O resultado dessa instanciação é um dispositivo lógico que implementa e conhece a interface de acesso ao dispositivo real de automação que está instalado no AmI. Na Figura 4.23, é apresentada a interface do módulo para gerenciamento dos dispositivos de

automação, em que é possível observar os campos utilizados para instanciação e criação dos dispositivos lógicos.

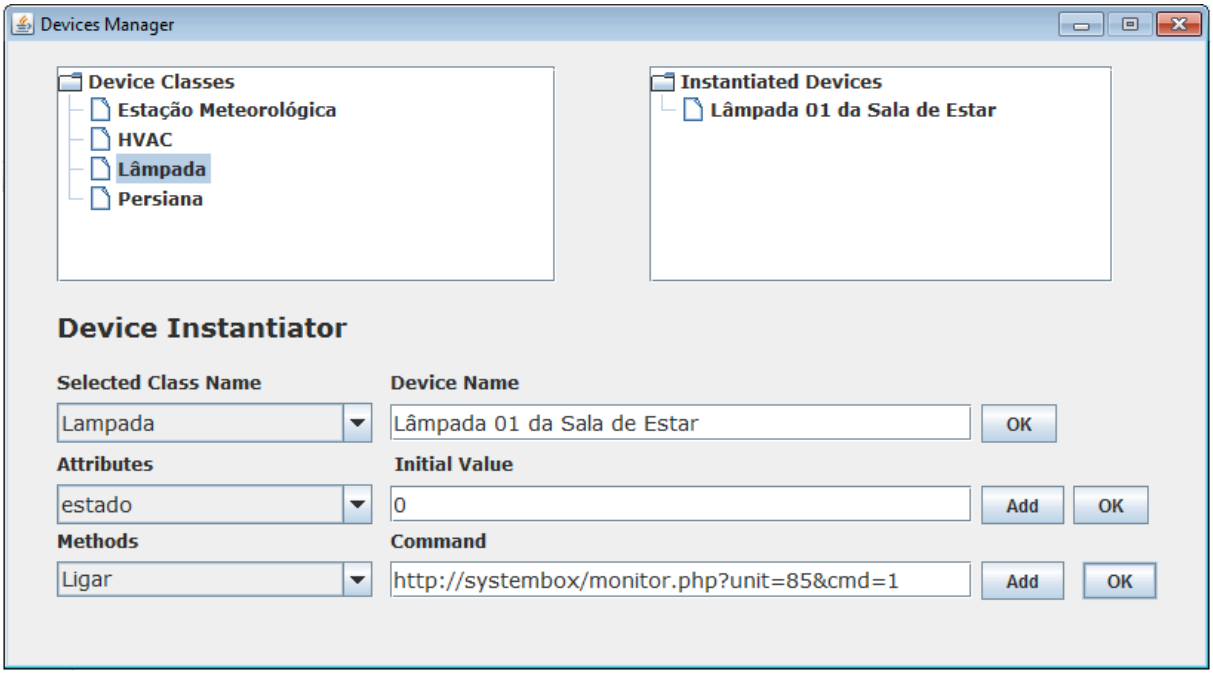

Figura 4.23 Interface de gerenciamento dos dispositivos de automação.

A interface de gerenciamento dos dispositivos de automação permite que sejam visualizadas as classes de dispositivos descobertas, bem como os objetos (dispositivos lógicos) que são instanciados. Ainda, na Figura 4.23, é possível observar um exemplo de instanciação, em que é selecionada uma classe chamada "Lâmpada". Essa classe é instanciada, através da parametrização que irá gerar um dispositivo lógico chamado (Lâmpada 01 da Sala de Estar) o qual irá herdar todas as características da classe "Lâmpada", ou seja, o atributo (estado) e os métodos (Ligar, Desligar e Ler Estado) existentes na classe "Lâmpada" serão utilizados pelo objeto que está sendo criado. Outros dispositivos lógicos baseados na classe "Lâmpada" podem ser instanciados igualmente, alterando apenas os parâmetros já definidos na criação do primeiro objeto. Todos os atributos e métodos de uma classe podem ser parametrizados pelo Instanciador de Dispositivos utilizando os botões de adicionar novo atributo ou método (ADD). Entretanto, nem todos os atributos e métodos de uma classe precisam ser utilizados na criação de um objeto (dispositivo lógico). É possível escolher

apenas os que interessam para uma determinada situação. Depois de instanciadas as classes dos dispositivos de automação e criados os dispositivos lógicos, estes são armazenados no **Repositório de Dispositivos** e ficam disponíveis para serem utilizados pelos serviços que serão criados. Os dispositivos lógicos ficam disponíveis, também, em arquivos com formato XML e seguem a estrutura apresentada na Figura 4.24.

```
<xml>
    <instantiated_device>
        <selected_class_name>Lâmpada</selected_class_name>
        <device_name>Lâmpada 01 da Sala de Estar</device_name>
        <attributes>
                <attribute name="estado" value="0" />
        </attributes>
        <methods>
                <![CDATA[ <method name = "Ligar"
                command = "http://systembox/monitor.php?unit=85&cmd=1"/>]]>
                <![CDATA[<method name = "Desligar"
                command = "http://systembox/monitor.php?unit=85&cmd=0"/>]]>
                <![CDATA[<method name = "Ler Estado"
                command = "http://systembox/monitor.php?unit=85&cmd=get"/>]]>
        </methods>
    </instantiated_device>
</xml>
```

Figura 4.24 Dispositivos lógicos representados em XML.

Na estrutura do arquivo XML são atribuídos os parâmetros do dispositivo lógico que está sendo instanciado, sendo que esses parâmetros correspondem às informações de acesso ao dispositivo real de automação que está instalado no AmI. Na Figura 4.25, é apresentado, através de um diagrama de atividades, o algoritmo utilizado na implementação do módulo de gerenciamento dos dispositivos de automação.

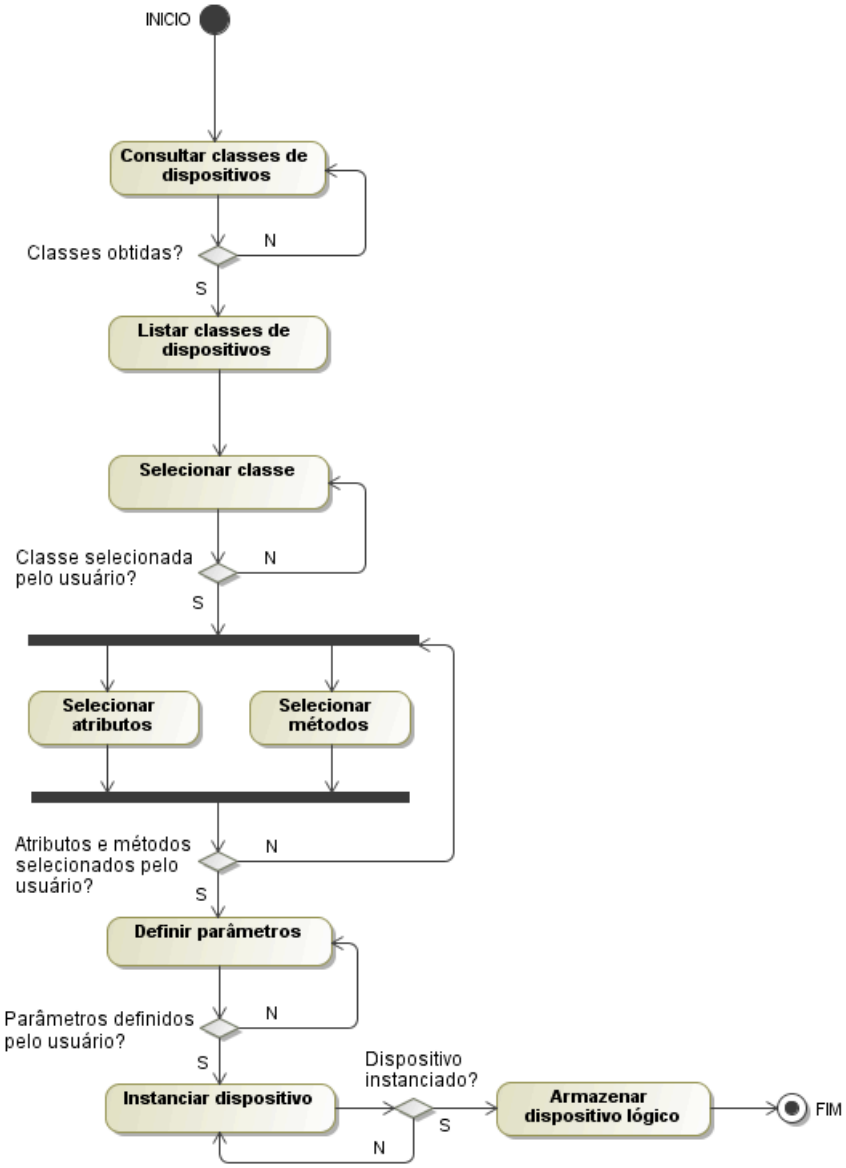

Figura 4.25 Algoritmo de gerenciamento dos dispositivos de automação.

4.2.2.3 Gerenciamento de Serviços

O módulo implementado para o gerenciamento de serviços é formado por três componentes: gerenciador, construtor e repositório. Assim como no módulo de gerenciamento de dispositivos, a ferramenta de *software* desenvolvida para o gerenciamento de serviços agrupa os três componentes e oferece uma interface de administração desse módulo. O fluxo de comunicação entre os componentes da ferramenta é iniciado pelo **Gerenciador de Serviços**, o qual acessa o Repositório de Dispositivos e realiza uma busca pelos dispositivos lógicos que estão disponíveis para utilização. A técnica de busca utilizada pelo gerenciador

consiste em uma varredura de arquivos XML correspondentes aos dispositivos lógicos de automação que foram instanciados no módulo de gerenciamento de dispositivos (ver Figura 4.24). Essa varredura identifica os elementos de cada arquivo XML, que são listados em uma janela de dispositivos lógicos e ficam disponíveis para a criação de serviços. Apesar de a varredura obter todos os elementos da estrutura do arquivo, a criação de um serviço considera, basicamente, os elementos <*device_name*> e <*method name*>, pelo fato de eles contemplarem as respostas de qual dispositivo de automação será utilizado e qual funcionalidade desse dispositivo será utilizada. A criação de um serviço é realizada pelo **Construtor de Serviços**, o qual oferece uma interface para que sejam definidos o nome e a descrição do serviço que está sendo criado, além de realizar a associação entre esse serviço e um grupo de serviços já existentes (segurança, emergência, entretenimento, etc.). O Construtor de Serviços permite, ainda, que para cada serviço possa ser associado mais de um dispositivo de automação, ou seja, a criação de um serviço chamado "*Iluminar o Ambiente*" pode ser composto pelos dispositivos "Lâmpada" e "Persiana", sendo que o método utilizado do dispositivo "Lâmpada" é o "Ligar" e o do dispositivo "Persiana" é "Abrir". Nesse caso, a interface de gerenciamento de serviços não precisaria listar todas as informações de cada dispositivo. Seria necessário listar apenas os nomes dos dispositivos de automação e dos seus respectivos métodos que seriam utilizados. Na Figura 4.26, é exibida a interface do módulo de gerenciamento de serviços.

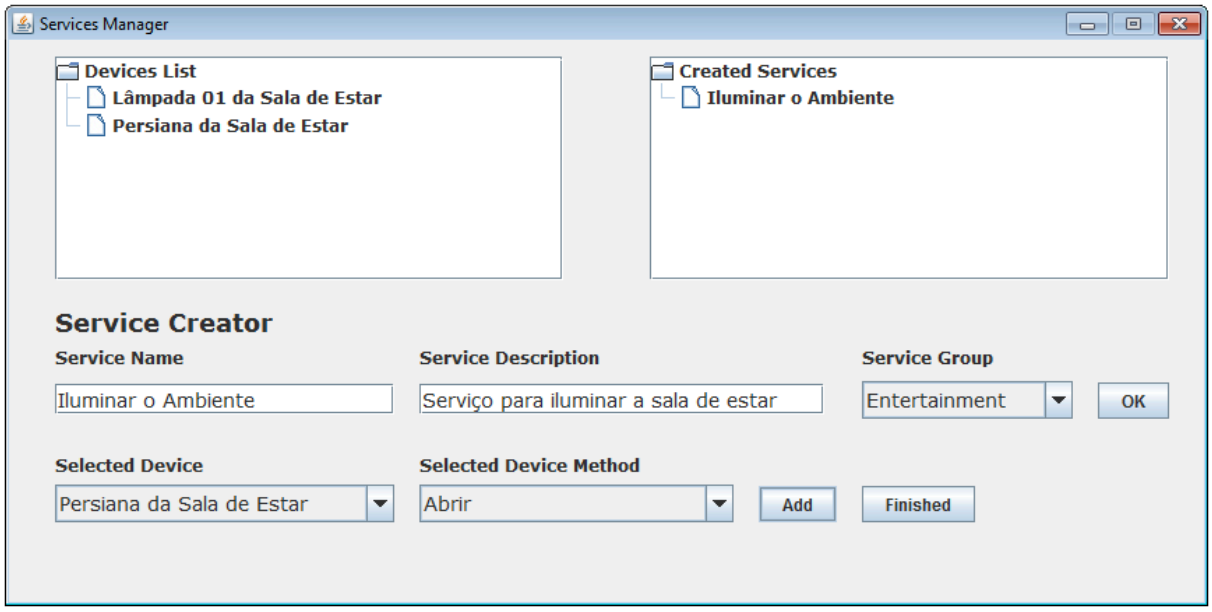

Figura 4.26 Interface de gerenciamento de serviços.

Conforme pode ser observado na Figura 4.26, a interface de gerenciamento disponibiliza duas listagens, uma contendo os dispositivos lógicos que foram instanciados e estão aptos para utilização, e outra contendo os serviços que já foram ou estão sendo criados. A ferramenta desenvolvida permite que, rapidamente, sejam escolhidos os dispositivos de automação que serão utilizados, sejam selecionados os seus métodos e criados os serviços. Esses serviços são criados no formato XML, conforme o padrão apresentado na Figura 4.27, e disponibilizados no **Repositório de Serviços**, o qual será utilizado na construção de cenários.

```
<xml>
    <service>
        <created_service name="Iluminar o Ambiente" group="Entertainment" />
        <service_description>Serviço para iluminar a sala de estar</service_description>
        <devices>
                <device name="Lampada 01 da Sala de Estar" method="Ligar" />
                <![CDATA[ <command>http://systembox/monitor.php?unit=85&cmd=1</command> ]]>
                <device name="Persiana da Sala de Estar" method="Abrir" />
                <![CDATA[ <command>http://systembox/monitor.php?unit=34&cmd=1</command> ]]>
        </devices>
    </service>
</xml>
```

Figura 4.27 Serviços representados em XML.

O algoritmo utilizado na implementação do módulo de gerenciamento de serviços é apresentado na Figura 4.28:

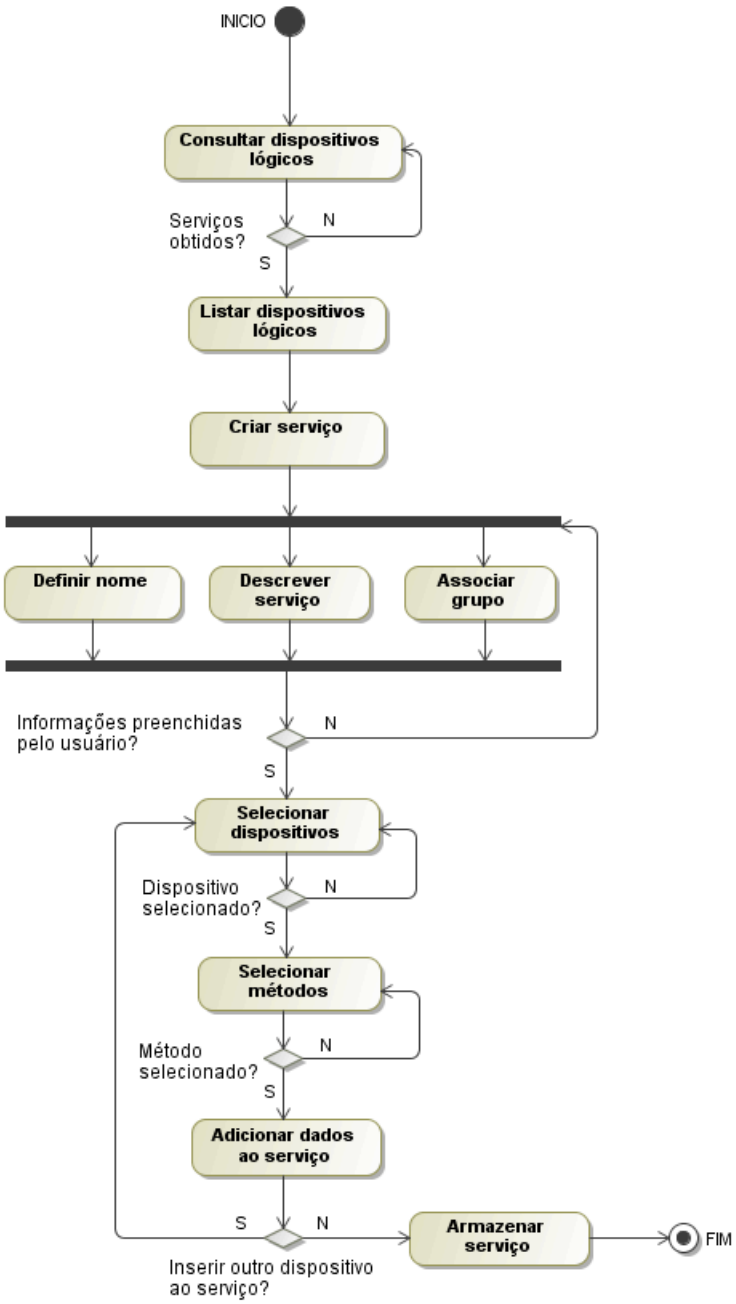

Figura 4.28 Algoritmo de gerenciamento de serviços.

4.2.2.4 Construção de Cenários

A ferramenta desenvolvida para realizar a construção de cenários que irão gerenciar o AmI é combinada por quatro componentes: descoberta de serviços, construtor de cenários, reutilizador e repositório. A ferramenta de *software* criada para esse módulo possui um fluxo de comunicação entre os seus componentes, que é iniciado através da **Descoberta de**

Serviços. Essa descoberta consiste na realização de consultas ao Repositório de Serviços que foi implementado no módulo de Gerenciamento de Serviços. Como resultado da busca, são listados todos os serviços que foram criados anteriormente e estão disponíveis para utilização por parte dos cenários, ou seja, um cenário pode ser composto por um ou muitos serviços. Existem diversas tecnologias para descoberta de serviços que são utilizadas em *Service-Oriented Architecture* (SOA) e que podem ser aproveitadas por esse componente, tais como: OSGi, Jini, *WebServices*, ontologias e outros (SILVA & GONÇALVES, 2010). Entretanto, mesmo o componente de descoberta de serviços podendo implementar essas tecnologias, neste trabalho a descoberta de serviços é realizada através da leitura de arquivos XML, os quais contêm a descrição e todas as informações do serviço criado no módulo de Gerenciamento de Serviços. Na Figura 4.29, é ilustrada a busca pelos serviços que estão armazenados no repositório e descritos em linguagem XML. O componente de Descoberta de Serviços realiza uma varredura no repositório de serviços, extraindo as informações de cada serviço existente e disponibilizando-as para a construção dos cenários. Essas informações consistem em identificar o nome do serviço descoberto, quais dispositivos de automação ele utiliza, quais são os métodos utilizados a partir desses dispositivos e qual a interface de acesso a esse dispositivo.

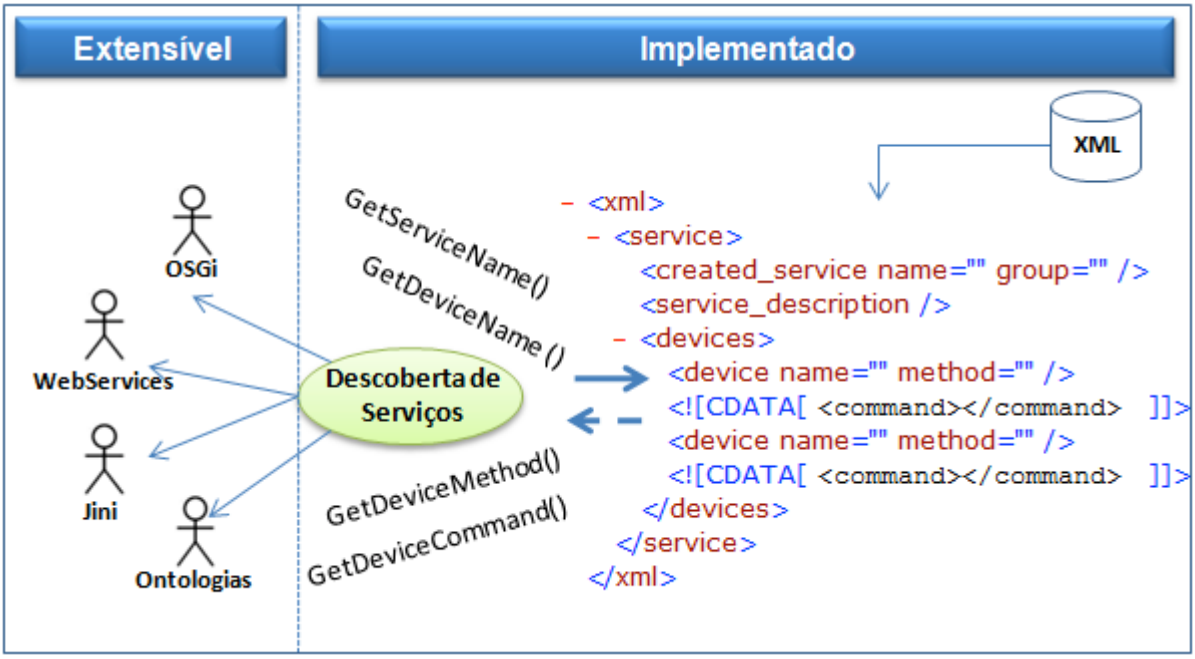

Figura 4.29 Descoberta de serviços.

Na Figura 4.27, apresentada anteriormente, é possível visualizar como um serviço é representado utilizando a estrutura XML, ilustrada na Figura 4.29. Essa estrutura de arquivo permite que haja uma compatibilidade na troca de informações entre os componentes do *framework*, pois o componente de Descoberta de Serviços sabe exatamente quais são as informações a serem extraídas e de quais elementos do arquivo elas devem ser retiradas, por exemplo: o elemento <*created service name*> possui a informação referente ao nome do serviço que foi criado e o elemento <*device name*> contém o nome do dispositivo de automação que é utilizado pelo serviço. Esse é o mecanismo de descoberta de serviços implementado neste trabalho. Caso o dispositivo de automação seja modificado durante a operação, as informações do serviço podem ser atualizadas sem que haja, necessariamente, uma redescoberta, uma vez que o serviço já é conhecido. Quando se deseja utilizar um determinado serviço externo, ou seja, que não foi criado utilizando o *framework* é necessário que o componente de Descoberta de Serviços conheça e implemente a interface de acesso a esses serviços. Por exemplo: a criação de um cenário chamado "Despesas do Dia" poderia ser composto pelos serviços "*verificar o consumo atual de energia da casa*" e "*verificar o saldo*

da conta bancária". O primeiro serviço poderia estar descrito de acordo com a estrutura de um arquivo XML criado para os serviços e armazenado no Repositório de Serviços local (implementado). Já o segundo serviço poderia estar descrito na linguagem padrão utilizada para *WebServices* chamada *Web Services Description Language* (WSDL) e armazenado no Repositório de Serviços do banco em que o usuário tem uma conta (extensível). Nesse caso, o componente de Descoberta de Serviços faria uma busca pelos serviços disponíveis no repositório local e no remoto. A única particularidade desse componente seria implementar a interface de acesso para as tecnologias de descoberta de serviços que se deseja utilizar.

Após ter sido realizada a descoberta dos serviços, a ferramenta construída para esse módulo realiza uma listagem com o resultado de todos os serviços que foram descobertos e estão disponíveis para utilização no AmI. Nesse momento, os cenários já podem ser criados pelo componente **Construtor de Cenários**, o qual oferece os recursos para que sejam selecionados quais dos serviços descobertos e listados serão utilizados na construção do cenário. Conforme pode ser observado na Figura 4.30, a interface gráfica da ferramenta desenvolvida permite que seja criado um cenário e, inseridos nesse cenário, diversos serviços. Por exemplo, a criação de um cenário chamado "Chegar em Casa" é composto pelos serviços "*Despesas do Dia*", "*Iluminar o Ambiente*" e "*Verificar a Hora*". Cada serviço é responsável por uma ou mais atividades, entretanto, a combinação desses serviços resulta em um cenário que foi projetado de acordo com os **Requisitos** ou preferências do usuário. Esses requisitos podem envolver desde premissas de quais grupos de serviços um usuário pode ou não utilizar, até desejos e preferências dos usuários. Além disso, é possível observar que a ferramenta mostra também a descrição de cada serviço selecionado. Isso facilita a identificação do que realmente o serviço escolhido faz, pois, como a criação dos nomes de serviços e de cenários pode ser muito abstrata ou diferente de um usuário para outro, essa descrição é extremamente

importante quando há muitos serviços no repositório e/ou o sistema é utilizado por muitas pessoas.

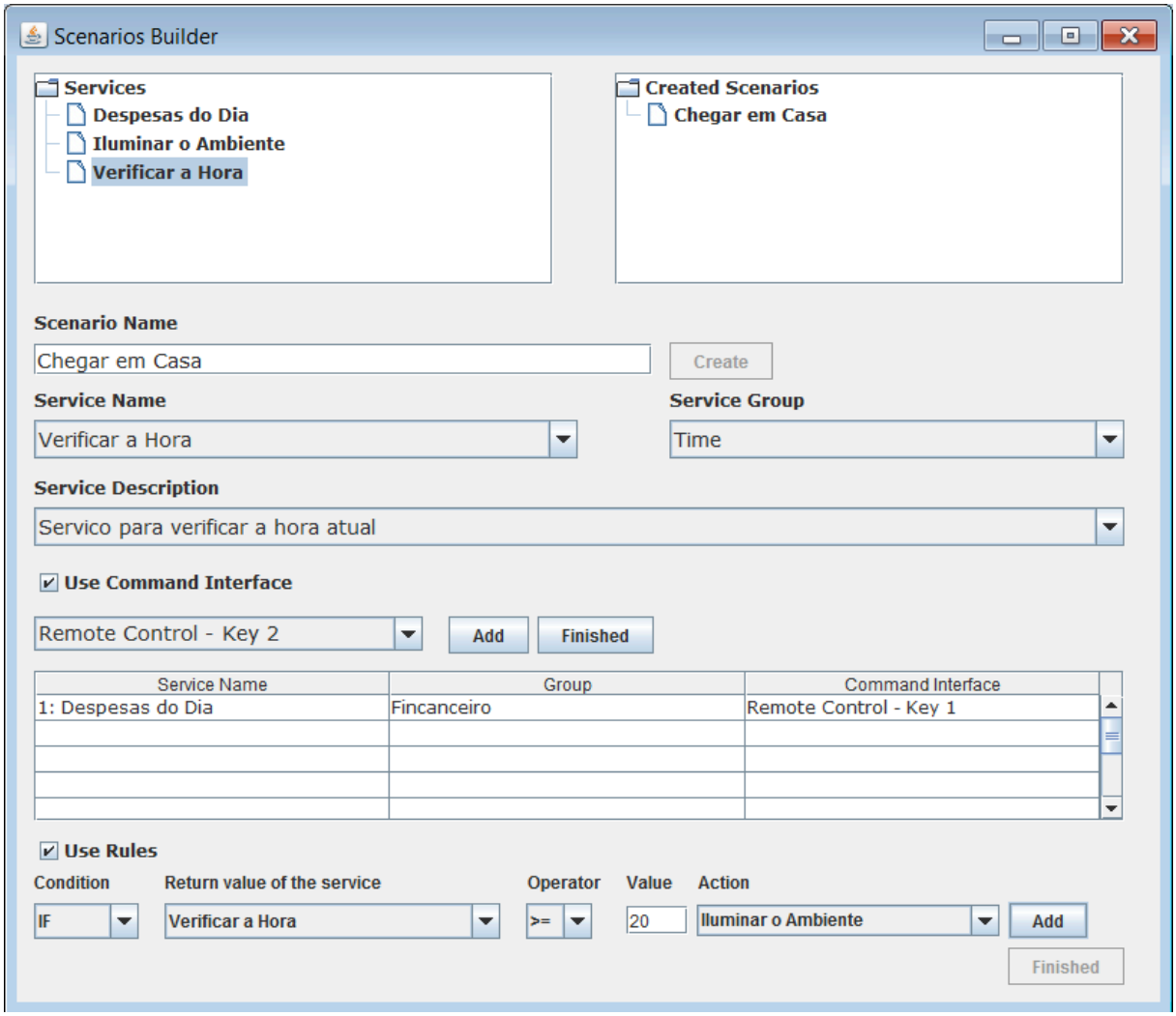

Figura 4.30 Interface Gráfica para construção de cenários.

Após a seleção e identificação dos serviços que serão utilizados pelo cenário, a ferramenta permite que seja realizada uma associação entre o serviço escolhido e uma interface de acesso a esse serviço. As interfaces de acesso a um serviço podem ser as mais variadas, desde o comando de voz, captado por um microfone e enviado para uma unidade de reconhecimento e execução do comando, até um simples clique de mouse, se for uma interface de comando baseado em computador. Neste trabalho, a intenção é que os serviços automatizados e, por consequência, o AmI, possa ser gerenciado por TVs, STBs ou telefones celulares com suporte ao *middleware* de interatividade Ginga. Nesse caso, a interface de

comando escolhida é o teclado do controle remoto do dispositivo. Dessa forma, a ferramenta permite que, para cada serviço selecionado para compor um cenário, seja associada uma tecla do controle remoto ou uma regra condicional baseada em informações temporizadas e/ou baseadas nos valores de retorno de algum serviço. Dessa forma, conforme ilustra o exemplo apresentado na Figura 4.30, o cenário "Chegar em Casa" é composto pelo serviço "*Despesas do Dia*" (que pode ser acessado pela tecla 1 do controle remoto) e pelo serviço "*Iluminar o Ambiente"* (que somente é executado a partir das 20 horas). Outro exemplo para execuções automáticas de serviços dependentes de entrada poderia ser o acionamento do aparelho de ar condicionado caso a temperatura do ambiente estivesse acima dos 24ºC. Nesse caso, o serviço "*Ligar HVAC"* seria executado de acordo com a comparação entre a temperatura desejada e o valor informado pelo dispositivo "sensor de temperatura", através do serviço *"Verificar Temperatura"*. Eventualmente, um terceiro serviço poderia ser associado à tecla 2 ou a outra tecla qualquer. Também poderia ser selecionada a mesma tecla para execução de mais de um serviço. Enfim, a interface de comando pode evoluir de acordo com os requisitos, desejos dos usuários ou evolução do AmI.

Quando a construção de um cenário é finalizada, é gerado um arquivo XML contendo toda a configuração escolhida para o cenário (ver Figura 4.31), o qual é armazenado no **Repositório de Cenários**. Todos os cenários que são criados são arquivos de configuração e não código em uma linguagem de programação. Isso permite que um mesmo cenário criado possa ser gerado para várias plataformas de *hardware* e em diversas linguagens de programação. Além disso, separar a configuração de um cenário da sua aplicação propriamente dita torna a arquitetura do *framework* mais flexível e modular, em que a alteração ou inserção de alguns componentes não afetem a estrutura básica da arquitetura.

```
<xml>
    <created_scenario>
        <scenario name="Chegar em Casa" />
        <used_services>
            <service name="Despesas do Dia" group="Financeiro" />
            <description>Serviço para verificar os gastos com o consumo de energia</description>
```

```
            <command_interface>Remote Control - Key 1</command_interface>
            <![CDATA[ <command>http://systembox/monitor.php?unit=34&cmd=getinfo</command>]]>
            <service name="Verificar a Hora" group="Time" />
            <description>Serviço para verificar a hora atual</description>
            <![CDATA[ <rule> CONDITION: IF Verificar a Hora.returnValue >=20 ACTION = Iluminar
                o Ambiente </rule>]]>
        </used_services>
    </created_scenario>
</xml>
```

Figura 4.31 Cenários representados em XML.

Outro fator importante é que os cenários podem ser reutilizados e, através do componente **Reutilizador de Cenários**, é possível a criação de novos cenários com base nos cenários já existentes. Isso pode tornar mais ágil o processo de desenvolvimento de novas aplicações de gerenciamento do AmI. Além disso, poderia ser considerada a utilização de *templates* de cenários, os quais poderiam ser parametrizados no momento da geração de código. Essa abordagem pode ser útil no caso de existir um cenário genérico que necessita ser especializado diversas vezes, por exemplo: em um hotel, o cenário para "Gerenciamento do Ambiente" pode ser idêntico em todos os quartos, alterando apenas o endereço dos dispositivos de automação do quarto específico. Com isso existiria um cenário padrão instanciado diversas vezes. Na Figura 4.32, é caracterizado o diagrama de atividades referente ao algoritmo implementado para a construção de cenários.

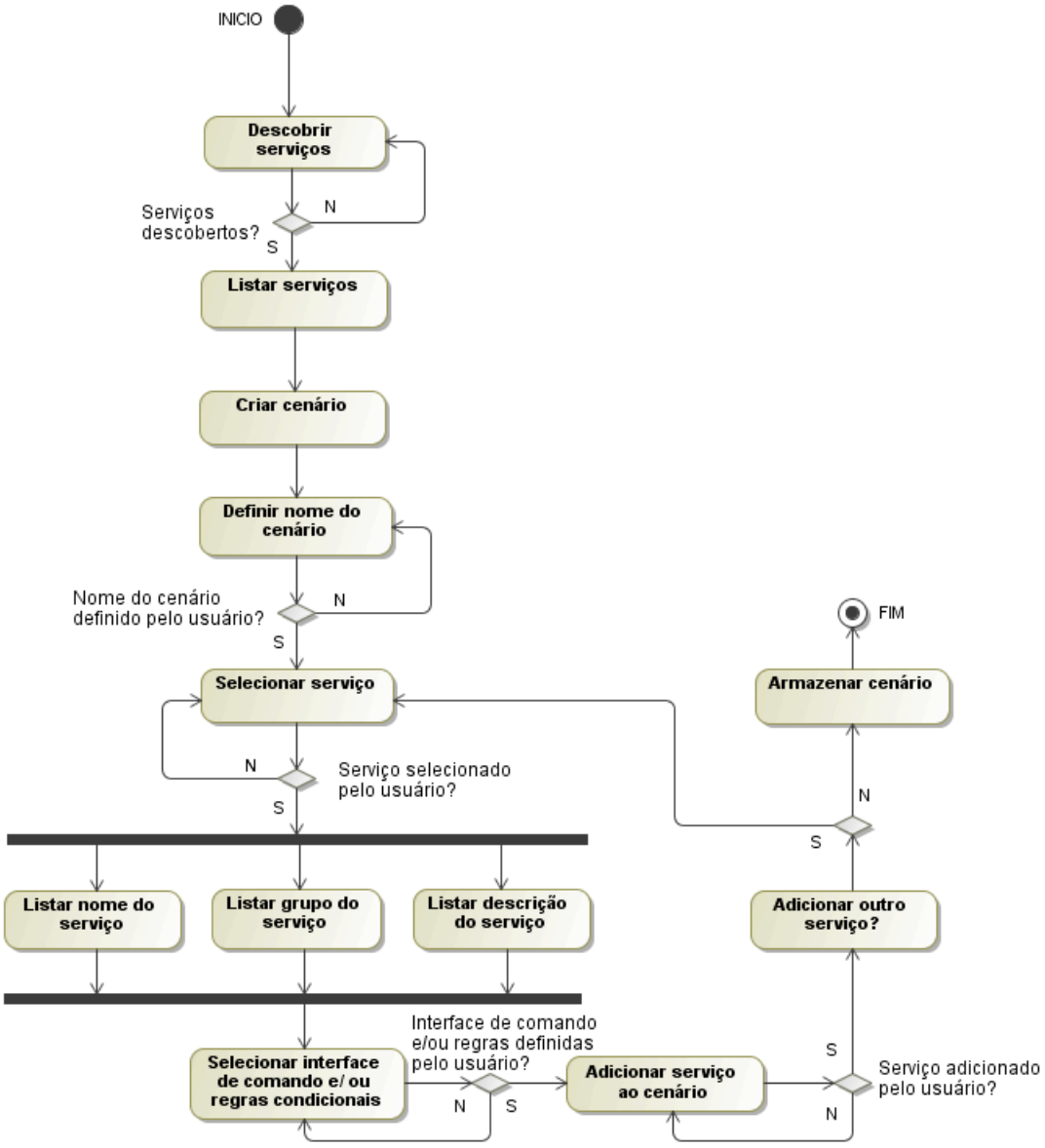

Figura 4.32 Algoritmo para construção de cenários.

4.2.2.5 Geração das Aplicações Interativas

O módulo para geração das aplicações interativas é o responsável por transformar os cenários de automação em código de programação. Esse processo é um dos mais importantes da proposta do trabalho, pois permite que os cenários construídos possam, de fato, ser executados e explorados pelos usuários através dos receptores de TV digital. A ferramenta de *software* implementada para esse módulo possui quatro componentes. O primeiro é o **Gerador de Código**, o qual possui a função de consultar no Repositório de Cenários todos os

cenários que foram construídos e que estão disponíveis para codificação. O resultado da consulta permite que seja possível selecionar qual cenário se deseja transformar em uma aplicação, podendo ser escolhido o tipo da plataforma-alvo que irá executar a aplicação, qual o perfil do *middleware* de interatividade a ser utilizado e qual a interface gráfica da aplicação. Essas informações são extremamente importantes, pois são elas que fornecem a base para a geração das aplicações interativas. O fluxo de comunicação entre os componentes desse módulo é iniciado a partir da consulta ao Repositório de Cenários, que é realizada pelo Gerador de Código. Conforme ilustra a Figura 4.33, esse componente acessa os arquivos XML correspondentes aos cenários de automação e realiza uma busca pelos elementos <*scenario name*>, <*service name*>, <*command_interface*> e <*command*> de cada cenário. O primeiro elemento traz a informação referente ao nome do cenário que foi construído; o segundo contém o nome dos serviços que são utilizados pelo cenário; os terceiro e quarto elementos trazem as informações de qual interface de comando foi utilizada para cada serviço e qual o endereço ou comando para acesso a esse.

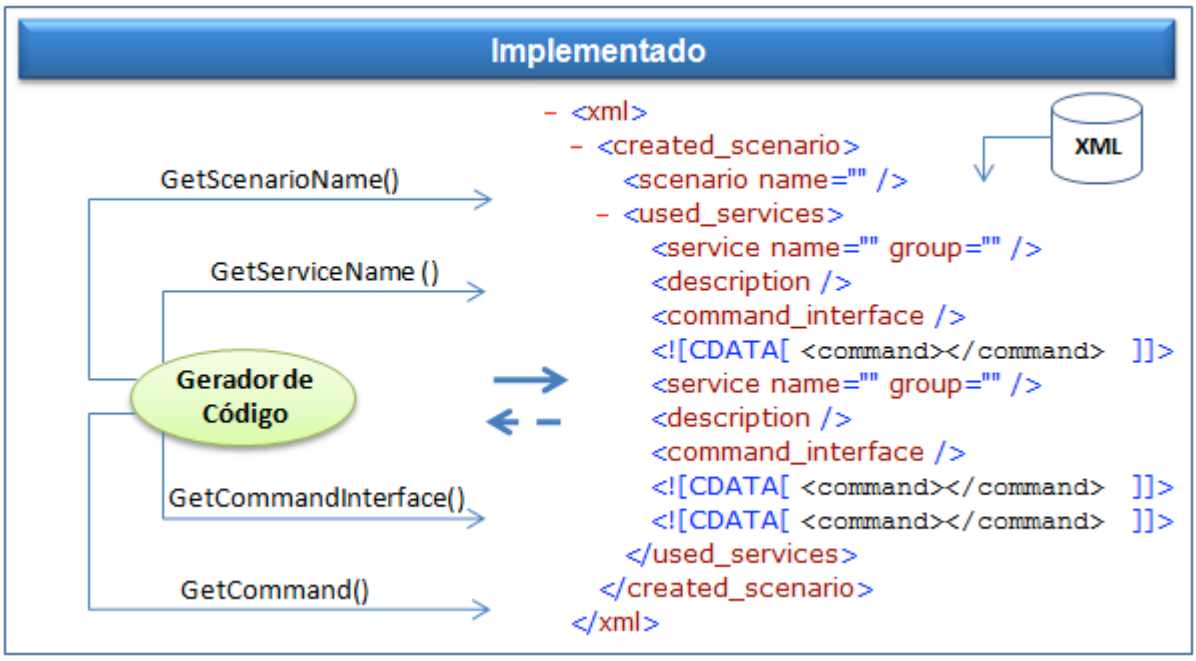

Figura 4.33 Consulta aos cenários de automação.

Para complementar a ilustração da Figura 4.33, é possível voltar à Figura 4.31 e observar a representação de um cenário que foi criado, em que todas as informações de cada elemento estão preenchidas. Após a identificação dos cenários de automação, a ferramenta de *software* desenvolvida para esse módulo, apresentada na Figura 4.34, permite que seja selecionado o cenário que será codificado.

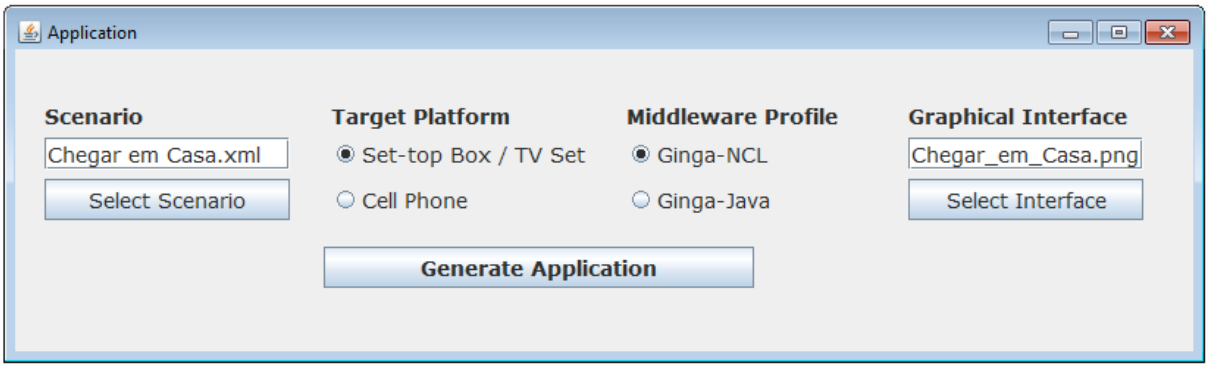

Figura 4.34 Interface gráfica para geração das aplicações interativas.

Com a escolha do cenário a ser codificado, é necessário definir em qual **Plataforma-Alvo** a aplicação será executada. Esse componente irá configurar os parâmetros de visualização da aplicação interativa, tais como resolução e nível de transparência. Como o tamanho da tela de visualização em uma TV é diferente de um telefone celular, esse componente permite ajustar a interface gráfica da aplicação de acordo com a plataforma-alvo selecionada. Já o nível de transparência é um parâmetro que irá definir o percentual de transparência de vídeo que será exibido no fundo da aplicação, uma vez que a aplicação interativa e o vídeo que está sendo transmitido pela emissora de TV são apresentados em camadas diferentes, sendo que a camada da aplicação sobrepõe-se à camada de vídeo, e por isso a importância de definir o quão transparente será a aplicação se o usuário desejar também assistir ao vídeo. Outras estratégias para visualização das aplicações podem conter o enquadramento de vídeo, em que a aplicação cria uma janela dentro dela mesma para colocar o vídeo que está sendo exibido, seguindo o exemplo das funções *Picture-in-Picture* (PiP) presentes em alguns televisores. Entretanto, na implementação desse módulo o componente

Plataforma-Alvo aborda apenas a resolução e o nível de transparência desejado para a aplicação. A partir da definição de qual plataforma-alvo irá executar a aplicação, é necessário escolher o **Perfil do Middleware** que se deseja utilizar para codificar o cenário, ou seja: qual linguagem será utilizada nesse processo. A especificação do *middleware* Ginga prevê o suporte tanto para aplicações desenvolvidas em JavaDTV quanto NCL / LUA. Ou seja, teoricamente um receptor de TV digital irá executar ambos os tipos de aplicações. Entretanto, o governo federal definiu que o suporte ao JavaDTV deve estar presente obrigatoriamente apenas em receptores fixos (TVs e STBs), sendo dispensado o seu uso em receptores móveis (telefones celulares). Já o suporte ao NCL / LUA deve ser obrigatório em ambas as plataformas, tanto fixas quanto móveis. Além disso, a linguagem JavaDTV possui disponível para a comunidade científica apenas a especificação da linguagem, não existindo qualquer implementação oficial dessa API por parte da Oracle SUN. Existem algumas implementações dessa API que estão sendo desenvolvidas por fabricantes de televisores e empresas de desenvolvimento de *software* responsáveis por embarcar o *middleware* nos receptores de TV digital. Quanto à linguagem NCL / LUA, ela está amplamente difundida e consolidada há alguns anos, sendo utilizada também para desenvolvimento de jogos (GAMEDEV, 2010). Dessa forma, a codificação dos cenários de automação utilizada neste trabalho é realizada utilizando a linguagem NCL / LUA. Essa opção se deve ao fato da não disponibilidade, até o momento, de uma API JavaDTV para construção de aplicações interativas e, principalmente, pelo fato de os cenários gerados em linguagem NCL terem a capacidade de ser executados tanto em receptores fixos quanto móveis, diferentemente do JavaDTV.

O último componente desse módulo refere-se à **Interface Gráfica**, a qual é utilizada pela aplicação para apresentar aos usuários todas as informações do cenário. É na interface gráfica que o usuário poderá visualizar os serviços contidos no cenário e poderá executar alguns desses serviços a partir da respectiva tecla do controle remoto. A interface gráfica da

aplicação poderia ser gerada automaticamente com base na leitura do arquivo XML referente ao cenário escolhido para codificação. Entretanto, neste trabalho optou-se por incluir o papel de um editor de imagens, em que este visualiza as informações que estão contidas no cenário e gera a interface gráfica, através de uma ferramenta para edição de imagens. Ou seja, para cada cenário de automação construído, será associada uma imagem que irá representar graficamente as informações dos serviços e modos de interação que o cenário oferece. A Figura 4.35 ilustra o fluxo para a criação dessa interface gráfica, tomando como exemplo o cenário apresentado na Figura 4.31.

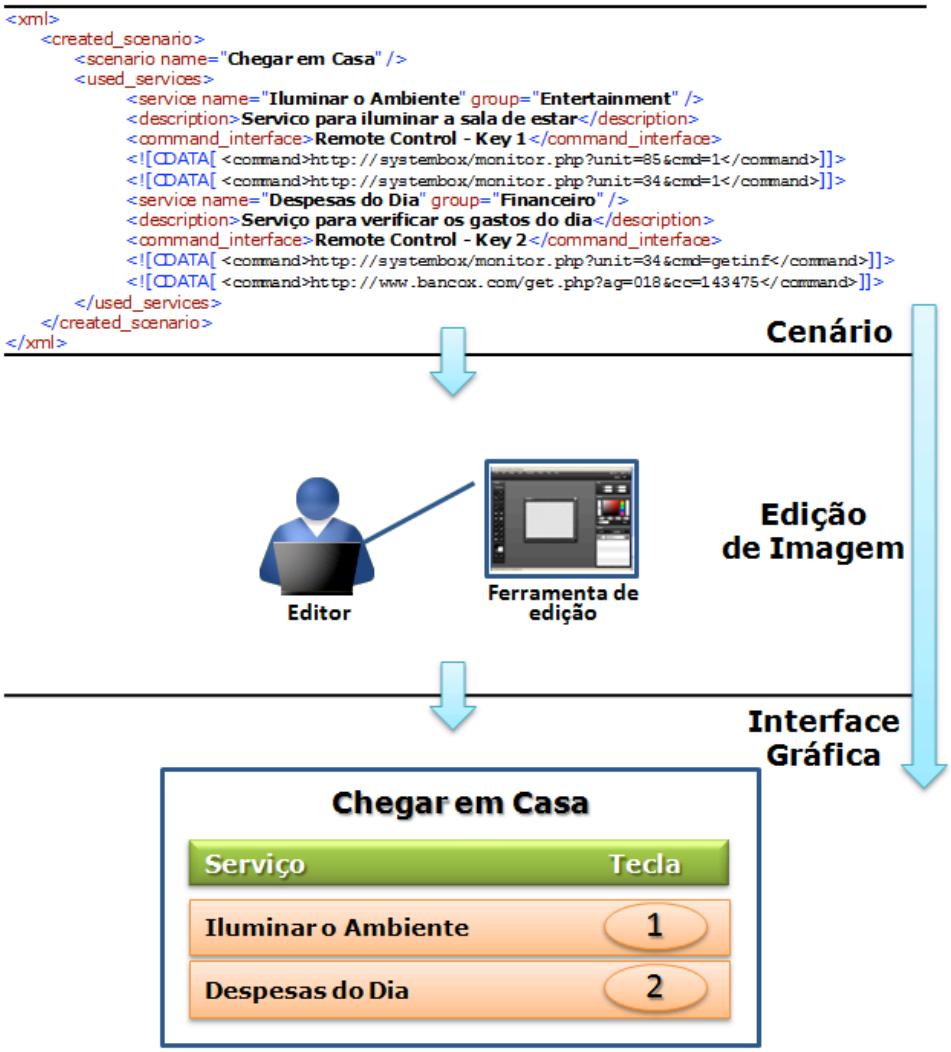

Figura 4.35 Processo para criação de interfaces gráficas.

A partir da criação da interface gráfica é possível, através da ferramenta construída, associá-la ao cenário e iniciar a codificação. Para isso, foram criados neste trabalho os

templates de código em NCL / LUA, os quais são parametrizados de acordo com as configurações presentes nos cenários e com as opções escolhidas na interface para geração das aplicações. Um exemplo desses *templates* de código pode ser observado no trecho ilustrado na Figura 4.36 e na Figura 4.37, onde são parametrizados parte do arquivo NCL e parte do arquivo LUA, respectivamente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <ncl id="APP_Interatividade" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
        <head>
        <regionBase>
            <region width="60%" height="80%" left="10%" top="10%" id="rgInteratividade"/>
            <region width="1071" height="598" left="22.0%" top="22.0%" id="rgInicial"/>
        </regionBase>
        <descriptorBase>
            <descriptor id="dInteratividade" region="rgInteratividade">
                <descriptorParam name="transparency" value="85%"/>
            </descriptor>
            <descriptor id="dInicial" region="rgInicial">
                <descriptorParam name="transparency" value="85%"/>
            </descriptor>
        </descriptorBase>
        </head>
        <body>
            <port id="pontoEntradaC" component="Interatividade"/>
            <port id="pontoEntradaB" component="Inicial"/>
            <media id="Interatividade" src="Chegar em Casa.lua" descriptor="dInteratividade"/>
            <media id="Inicial" src="Chegar em Casa.png" descriptor="dInicial"/>
        </body>
    </ncl>
```

Figura 4.36 Parametrização do template de código NCL.

```
local function start_tcp()
        request = { class="tcp", type="connect", host="systembox", port="80" }
        event.post("out",request)
end

local function keyHandler (evt)

-- EVENTO: TECLADO
        if evt.class == 'key' and evt.type == 'press' then
        local key = evt.key

        if (key == '1') then
                service_address1 = '/monitor.php?unit=85&cmd=1'
                service_address2 = '/monitor.php?unit=34&cmd=1'
                start_tcp()
                MSG_RETORNO = 'Accessing the service 1'
                redraw()
```

Figura 4.37 Parametrização do template de código LUA.

De uma forma resumida, o código NCL organiza a chamada e a apresentação dos arquivos de mídia utilizados no cenário, e o código LUA contém os *scripts* para acesso e execução dos serviços. A vantagem de se utilizar *templates* de código é que eles já possuem a implementação das principais funções utilizadas em aplicações interativas, tais como: comunicação TCP/IP, leitura de teclas que são pressionadas no controle remoto e exibição de textos e imagens na tela. Nesse caso, é muito comum os cenários conterem serviços que realizam chamadas aos dispositivos de automação através do protocolo TCP/IP. Assim, independente de qual endereço de dispositivo de automação o serviço utilize, a função é quase sempre a mesma, variando apenas o endereço e o tipo do comando, os quais são, então, parametrizados nos *templates,* gerando uma nova aplicação. Pode-se dizer que uma aplicação interativa é um cenário específico criado com base em um *template* genérico de códigos NCL / LUA.

O resultado do processo de codificação é um conjunto com três arquivos: cenário.ncl, cenário.lua e cenário.png. Esses arquivos são inseridos no receptor de TV digital através de uma memória externa (*pen drive*). Eventualmente, esses mesmos arquivos podem ser transmitidos pela emissora de TV, juntamente com o áudio e o vídeo da programação. Existem, ainda, outras possibilidades de transmissão pelo ar desses arquivos, como, por exemplo, através de um computador com uma placa para modulação e transmissão de sinais de TV digital (DEKTEC, 2010), que pode ser utilizado em ambientes de testes. Além disso, no caso dos telefones celulares, esses arquivos poderiam ser carregados tanto pelo ar quanto por *SD Cards.* Neste trabalho, a disponibilização das aplicações criadas é realizada por um *pen drive*, o qual é inserido diretamente na porta *Universal Serial Bus* (USB) dos receptores de TV digital, conforme ilustra a Figura 4.38. Além disso, na mesma figura é possível observar o fluxo de informações, desde a codificação do cenário até a interação do usuário

com a aplicação interativa de gerenciamento do AmI. Já na Figura 4.39, é apresentado o algoritmo construído para o módulo de geração das aplicações interativas.

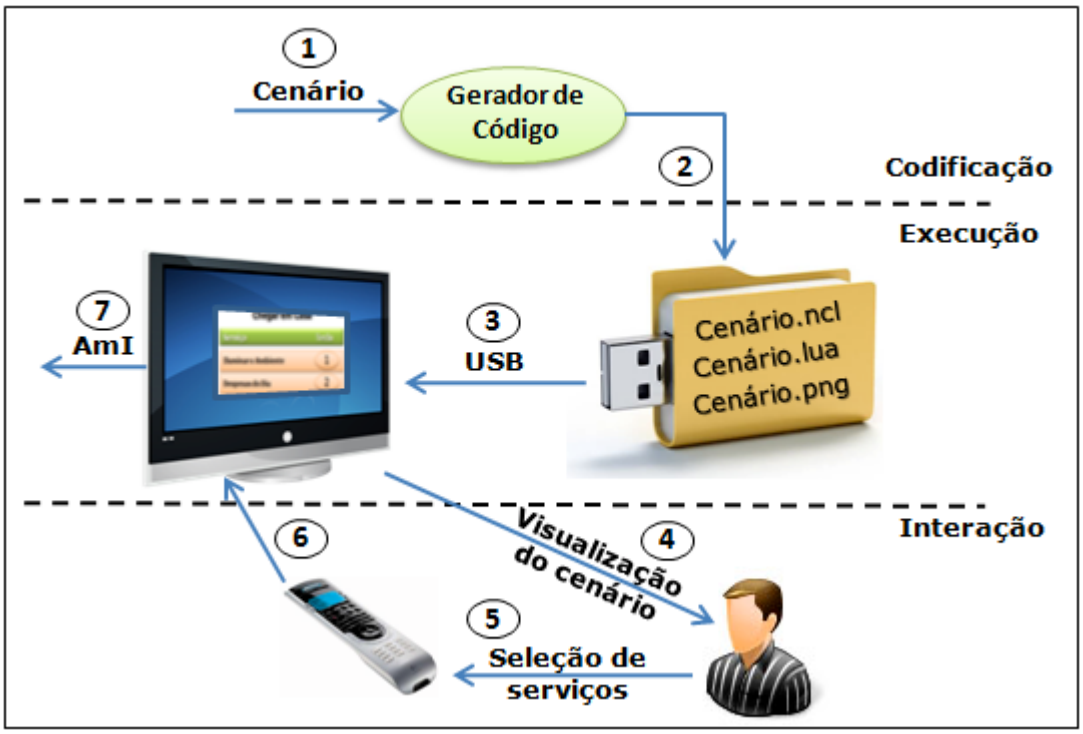

Figura 4.38 Disponibilização da aplicação e fluxo de informações.

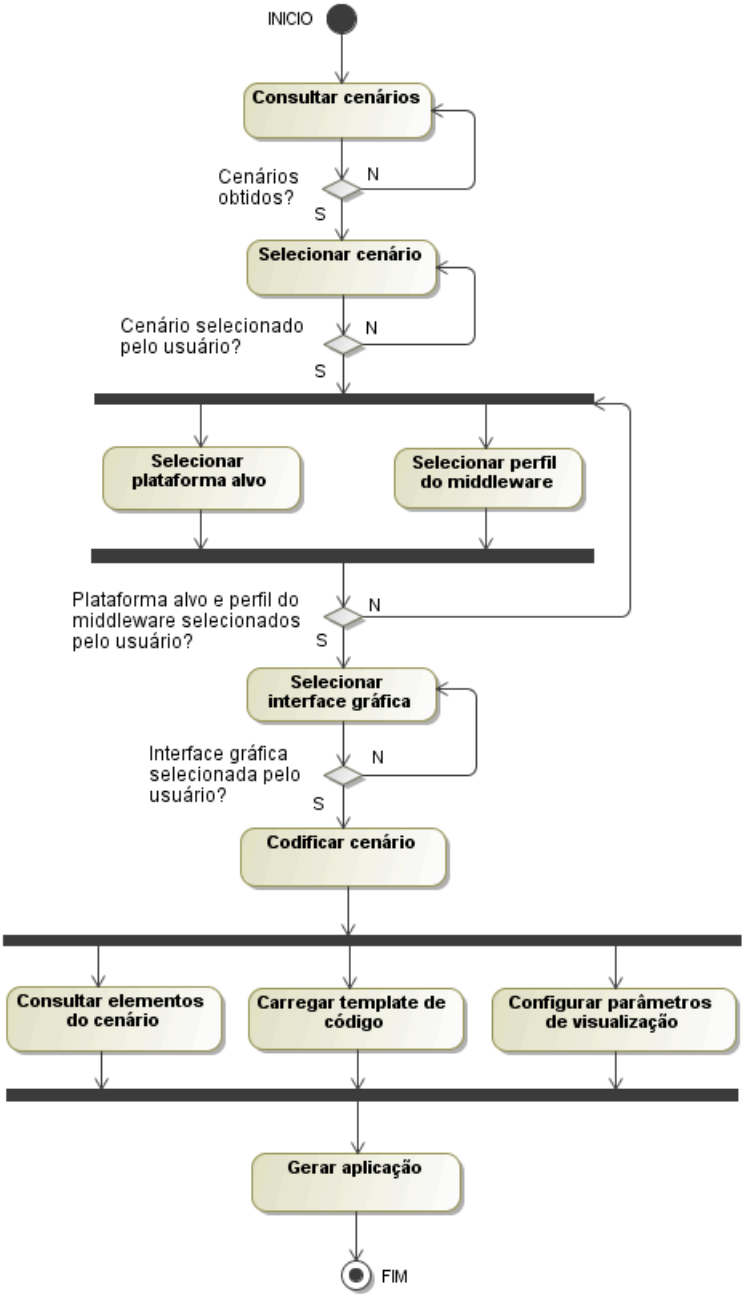

Figura 4.39 Algoritmo para geração das aplicações interativas.

4.2.3 Considerações sobre a Implementação

A arquitetura de implementação proposta neste trabalho, bem como os seus componentes e as ferramentas de *software*, foram desenvolvidas para serem executadas em PCs. Entretanto, a arquitetura poderia ser implementada para que os componentes e as ferramentas fossem executadas diretamente nos receptores de TV digital. Optou-se, nesse

momento, por utilizar esses receptores apenas para executar as aplicações de gerenciamento do AmI geradas pelas ferramentas. Isso se deve ao fato de que, até o momento, os receptores de TV digital apresentam algumas limitações em relação à capacidade computacional, como, por exemplo, baixa capacidade de processamento (≈ 300 *Megahertz* (Mhz)) e poucos recursos de memória RAM (≈ 64 *Megabytes* (MB)), se comparados com os PCs atuais. Entretanto, com o futuro avanço computacional dessas plataformas de *hardware* para decodificação do sinal da TV digital, essa tarefa de migração da arquitetura de implementação, que hoje é executada em um PC, poderá ser realizada para TVs, STBs e telefones celulares que ofereçam suporte ao *middleware* Ginga.

5 VALIDAÇÃO DO TRABALHO

Este capítulo apresenta a validação do trabalho proposto, sendo abordados três estudos de casos que utilizam os conceitos e as implementações definidas no capítulo anterior. No primeiro estudo de caso, são demonstrados todos os passos de utilização das ferramentas construídas, que vão desde o mapeamento computacional dos dispositivos de automação até a geração das aplicações interativas de gerenciamento do AmI. Entretanto, como o processo de utilização de conceitos e ferramentas em todos os estudos de casos é similar, os dois últimos estudos de casos estão mais direcionados aos resultados dessa utilização, uma vez que o processo detalhado é apresentado no primeiro estudo de caso.

5.1 GERENCIAMENTO DE RESIDÊNCIAS AUTOMATIZADAS VIA TV DIGITAL

O objetivo deste estudo de caso é demonstrar como é possível realizar o gerenciamento de eletrodomésticos existentes nas residências, utilizando os receptores de TV digital com suporte ao *middleware* Ginga. Esse estudo de caso é o resultado de um projeto desenvolvido pelo Laboratório de Sistemas de Controle, Automação e Robótica (LASCAR - UFRGS) que foi apresentado em um concurso de inovação da Whirlpool S.A. (empresa multinacional fabricante de eletrodomésticos da linha branca). O escopo da proposta consiste na montagem de uma infraestrutura de comunicação, na instrumentação de um eletrodoméstico e na utilização do *framework* proposto neste trabalho para permitir o gerenciamento do eletrodoméstico via TV digital. Para o estudo de caso, é utilizado um refrigerador como eletrodoméstico alvo de gerenciamento. As principais funcionalidades deste projeto estão classificadas em: (i) Implementação de sensores e atuadores no refrigerador: consiste na instrumentação do eletrodoméstico, para que ele seja capaz de ajustar *setpoints* de operação, obter e enviar imagens do conteúdo interno do equipamento e mensurar a temperatura. Nesse caso, a ideia está em agregar valor ao refrigerador e permitir que estes

possam ser integrados no AmIs e gerenciados remotamente; (ii) Gerenciamento de eletrodomésticos via TV digital: através de um sistema de comunicação sobre a rede elétrica (PLC), são obtidas informações do refrigerador (temperatura, alarme porta aberta, imagem do conteúdo interno do equipamento). Estas informações podem ser acessadas por meio de uma aplicação interativa que é executada no STB e pode ser visualizada na TV; (iii) Gerenciamento e otimização do consumo de energia para os equipamentos presentes no ambiente: o objetivo está na supervisão dos estados de operações dos equipamentos, permitindo eventuais ajustes de configuração para redução no consumo de energia.

5.1.1 Arquitetura do Estudo de Caso

O processo de automação do refrigerador buscou criar mecanismos para que este pudesse ser inserido no contexto dos AmIs, oferecendo serviços e permitindo ser gerenciado por receptores de TV. Para isso, foi utilizado um microprocessador MCF8282 (FREESCALE, 2010) que pertence à família de processadores de 32 *bits* MCF528X baseados na arquitetura *ColdFire*. Além disso, foi criada uma placa de instrumentação para monitorar e controlar a temperatura do refrigerador, bem como realizar o acionamento do compressor. Também foi inserido no refrigerador um medidor multivariável de grandezas elétricas, modelo UPD200, fabricado pela empresa Ciber do Brasil (CBR, 2009). Esse medidor permite, entre outras funções, monitorar o consumo de energia de um determinado equipamento. Também foi inserido no refrigerador um transmissor de vídeo sem fio que, acoplado a uma fonte geradora de áudio e/ou vídeo, transmite o sinal para os televisores. No caso do projeto, a fonte geradora de vídeo é a câmera instalada dentro do refrigerador. A distância de transmissão varia de acordo com a potência do transmissor. Nesse caso específico, o equipamento utilizado possui um raio de transmissão de aproximadamente cinquenta metros e transmite as imagens no canal dezoito *Ultra High Frequency* (UHF). O receptor de TV digital utilizado foi um STB

Proview modelo XPS-1000 com o *middleware* Ginga implementado pela empresa (RCASOFT, 2010). A arquitetura desse estudo de caso pode ser observada na Figura 5.1.

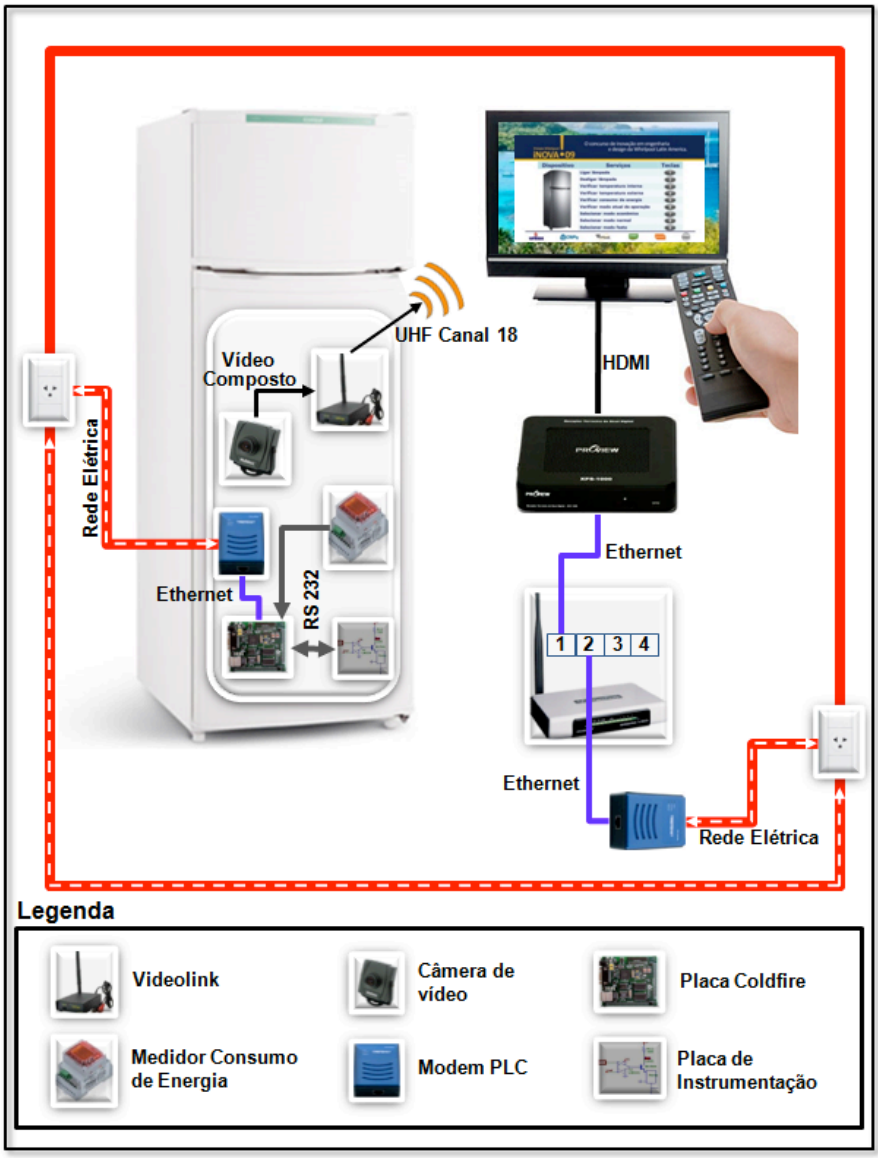

Figura 5.1 Arquitetura do estudo de caso.

Como o foco deste estudo de caso não está na instrumentação e na automatização do refrigerador, e sim na utilização de dispositivos que já ofereçam serviços e estejam integrados aos AmIs através de sistemas de automação, a partir desse momento é considerado apenas o processo de utilização do *framework* para permitir que o refrigerador possa ser gerenciado pelos receptores de TV digital. Ou seja, como é dado o processo desde o mapeamento computacional do dispositivo até a aplicação que o gerencia pela TV. O relatório completo

desse projeto, que inclui o processo de instrumentação, pode ser obtido no *website* oficial do trabalho: **http://sourceforge.net/projects/dtvi4ha**.

5.1.2 Utilização do *Framework*

A utilização do *framework* no estudo de caso é iniciada pelo processo de mapeamento computacional do eletrodoméstico, em que são identificados e descritos todos os atributos e os métodos da classe de dispositivos chamada “Refrigerador”, conforme ilustra a Figura 5.2.

```
<xml>
    <class name="Refrigerador">
        <attributes>
            <attribute name="estadoLâmpada" />
            <attribute name="consumo" />
            <attribute name="temperatura" />
            <attribute name="modoOperação" />
        </attributes>
        <methods>
            <method name="Ligar Lâmpada" />
            <method name="Desligar Lâmpada" />
            <method name="Verificar Temperatura Interna" />
            <method name="Verificar Temperatura Externa" />
            <method name="Verificar Consumo de Energia" />
            <method name="Verificar Modo Atual de Operação" />
            <method name="Selecionar Modo Econômico" />
            <method name="Selecionar Modo Normal" />
            <method name="Selecionar Modo Festa" />
        </methods>
    </class>
</xml>
```

Figura 5.2 Mapeamento computacional do dispositivo real.

Conforme apresentado no modelo conceitual, o processo de mapeamento computacional pode ser realizado tanto por uma ferramenta de modelagem UML, a qual irá gerar um arquivo XMI referente à classe do dispositivo, quanto por um editor de textos, gerando diretamente um arquivo XML correspondente. Neste estudo de caso, optou-se por utilizar diretamente um editor de textos.

Com a classe “Refrigerador” devidamente mapeada, o próximo passo consiste em instanciar essa classe e criar o dispositivo lógico. Para isso, é utilizada a ferramenta para gerenciamento dos dispositivos de automação. Conforme ilustrada na Figura 5.3, a ferramenta realiza a descoberta de uma nova classe de dispositivos e permite que esta possa ser instanciada.

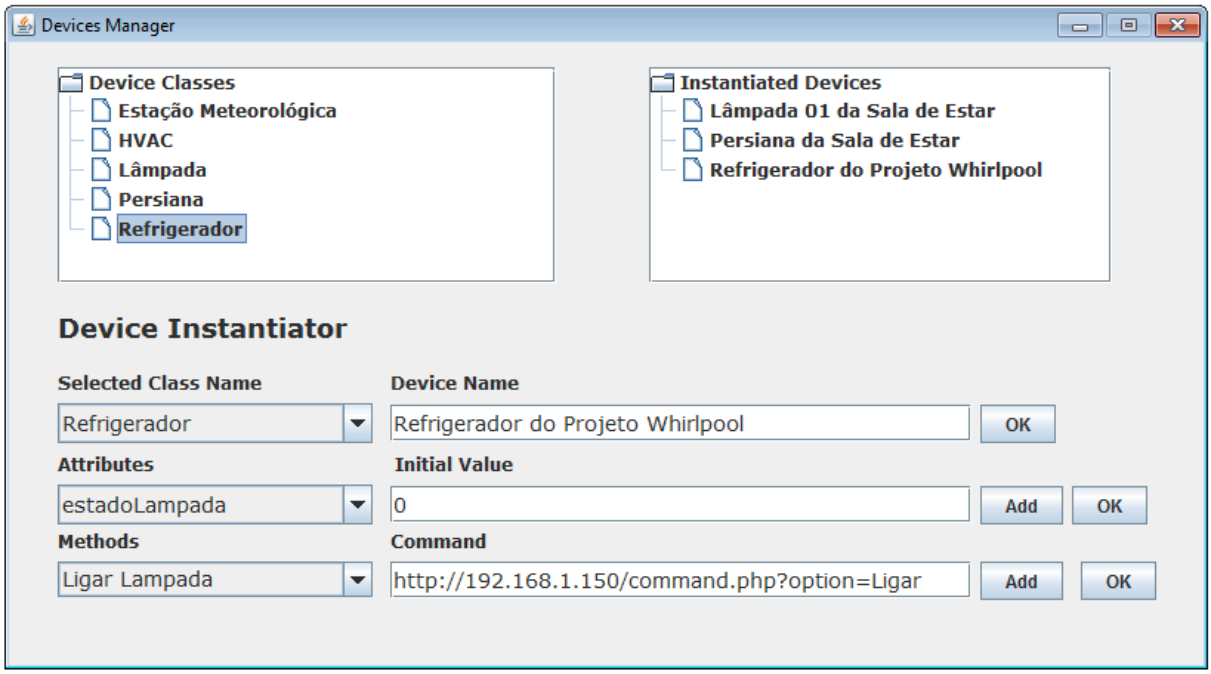

Figura 5.3 Descoberta e instanciação da classe refrigerador.

A partir da descoberta e da classe "Refrigerador", a ferramenta de gerenciamento dos dispositivos de automação permite que todos os atributos e métodos dessa classe possam ser parametrizados de acordo com as suas características ou com as necessidades do projeto. O resultado desse processo é a criação de um dispositivo lógico, chamado "Refrigerador do Projeto Whirlpool", instanciado conforme ilustra a Figura 5.4.

```
<xml>
    <instantiated_device>
        <selected_class_name>Refrigerador</selected_class_name>
        <device_name>Refrigerador do Projeto Whirlpool</device_name>
        <attributes>
            <attribute name="estadoLâmpada" value="0" />
            <attribute name="consumo" value="0" />
            <attribute name="temperatura" value="0" />
            <attribute name="modoOperação" value="Normal" />
        </attributes>
        <methods>
            <![CDATA[
                <method name = "Ligar Lâmpada" command =
                "http://192.168.1.150/command.php?option=Ligar"/>
             ]]>
            <![CDATA[
                <method name = "Desligar Lâmpada" command =
                "http://192.168.1.150/command.php?option=Desligar"/>
             ]]>
            <![CDATA[
                <method name = "Verificar Temperatura Interna" command =
                "http://192.168.1.150/command.php?option=getTempI"/>
             ]]>
            <![CDATA[
                <method name = "Verificar Temperatura Externa" command =
                "http://192.168.1.150/command.php?option=getTempE"/>
             ]]>
            <![CDATA[
```

```
            <method name = "Verificar Consumo de Energia" command =
            "http://192.168.1.150/command.php?option=getConsumo"/>
        ]]>
        <![CDATA[
            <method name = "Verificar Modo Atual de Operação" command =
            "http://192.168.1.150/command.php?option=getMode"/>
        ]]>
        <![CDATA[
            <method name = "Selecionar Modo Econômico" command =
            "http://192.168.1.150/command.php?option=setEco"/>
        ]]>
        <![CDATA[
            <method name = "Selecionar Modo Normal" command =
            "http://192.168.1.150/command.php?option=setNormal"/>
        ]]>
        <![CDATA[
            <method name = "Selecionar Modo Festa" command =
            "http://192.168.1.150/command.php?option=setFesta"/>
        ]]>

    </methods>
  </instantiated_device>
</xml>
```

Figura 5.4 Dispositivo lógico refrigerador.

Após a criação do dispositivo lógico “Refrigerador do Projeto Whirlpool”, é possível construir os serviços para acesso a esse dispositivo, por meio da ferramenta de gerenciamento de serviços. Essa ferramenta irá consultar o “Repositório de Dispositivos” e listar todos os dispositivos que estão disponíveis para ser utilizados pelos serviços. Conforme discutido no modelo conceitual, é possível criar serviços baseados em mais de um dispositivo de automação. Entretanto, como nesse estudo de caso o escopo do projeto consiste apenas em acessar os serviços exclusivamente do refrigerador, então todos os métodos presentes nesse dispositivo lógico são transformados em serviços, ou seja, um método chamado “Selecionar Modo Festa” passa a ser mapeado por um serviço chamado “Selecionar Modo Festa”. Por outro lado, seria perfeitamente possível criar um serviço chamado “Modo Festa” que utiliza os métodos “Selecionar Modo Festa” do refrigerador e “Desligar Iluminação” referente às lâmpadas da sala. Dessa forma, a Figura 5.5 apresenta a utilização da ferramenta para gerenciamento de serviços, em que estes são criados com base nos dispositivos lógicos de automação, no caso, o refrigerador instanciado pela ferramenta de gerenciamento de dispositivos.

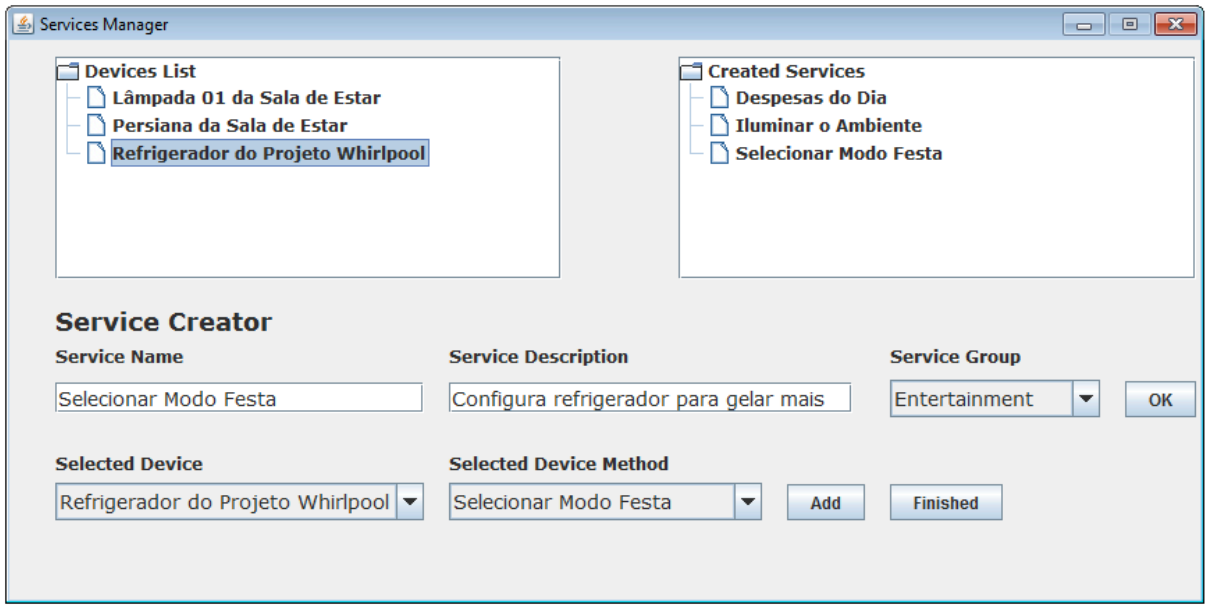

Figura 5.5 Criação de serviços baseados em dispositivos.

O resultado da criação do serviço "Selecionar Modo Festa" é o arquivo XML ilustrado na Figura 5.6. Esse serviço fica disponível para ser utilizado na construção dos cenários de automação do AmI.

```
<xml>
    <service>
        <created_service name="Selecionar Modo Festa" group="Entertainment" />
        <service_description>Configura refrigerador para gelar mais</service_description>
        <devices>
            <device name="Refrigerador do Projeto Whirlpool" method="Selecionar Modo Festa" />
            <![CDATA[ <command>http://192.168.1.150/command.php?option=setFesta</command>]]>
        </devices>
    </service>
</xml>
```

Figura 5.6 Serviço "Selecionar Modo Festa".

Com a criação dos serviços, estes ficam disponíveis no "Repositório de Serviços" e podem ser utilizados na construção de cenários de automação. Em um cenário de automação podem existir diversos dispositivos que oferecem inúmeros serviços, como é o caso do Refrigerador, em que é possível acessar uma variedade de funções. A criação desses cenários é realizada pela ferramenta de construção de cenários, ilustrada na Figura 5.7.

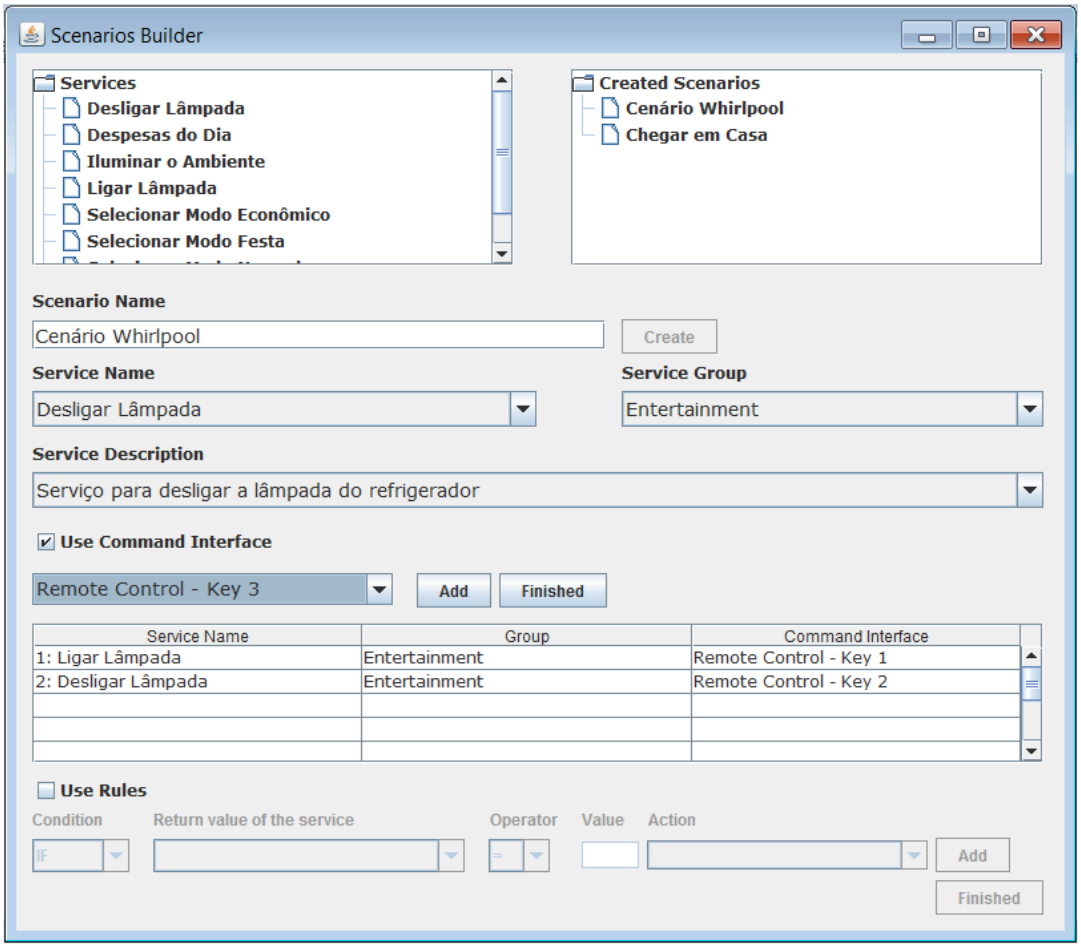

Figura 5.7 Construindo o cenário Whirlpool.

A ferramenta para construção do cenário "Whirlpool" resulta em um arquivo XML de configuração de cenários, sendo que a Figura 5.8 apresenta um trecho desta configuração. Esse arquivo de cenário é utilizado para codificação e geração da aplicação interativa que é executada no receptor de TV digital.

```
<xml>
    <created_scenario>
    <scenario name="Cenário Whilrpool" />
        <used_services>
            <service name="Ligar Lâmpada" group="Entertainment" />
            <description>Serviço para ligar a lâmpada interna do refrigerador</description>
            <command_interface>Remote Control - Key 1</command_interface>
                <![CDATA[ <command>http://192.168.1.150/command.php?option=Ligar</command>]]>
             <service name="Desligar Lâmpada" group="Entertainment" />
             <description>Serviço para desligar a lâmpada interna do refrigerador</description>
             <command_interface>Remote Control - Key 2</command_interface>
                <![CDATA[ <command>http://192.168.1.150/command.php?option=Desligar</command>]]>
                ....
                ....
        </used_services>
  </created_scenario>
 </xml>
```

Figura 5.8 Trecho de configuração do cenário Whirlpool.

Baseado no arquivo XML de configuração do Cenário Whirlpool, é possível criar a interface gráfica correspondente. Nesse caso, é utilizado um editor de imagens para criar a tela, apresentada na Figura 5.9, que será inserida na aplicação interativa para gerenciamento do refrigerador.

Figura 5.9 Interface gráfica da aplicação interativa para gerenciamento do refrigerador.

O último passo é a geração de código, ou seja, transformar o Cenário Whirlpool construído em uma aplicação interativa que será executada no receptor de TV digital. Para isso, é utilizada a ferramenta de geração de código, ilustrada na Figura 5.10.

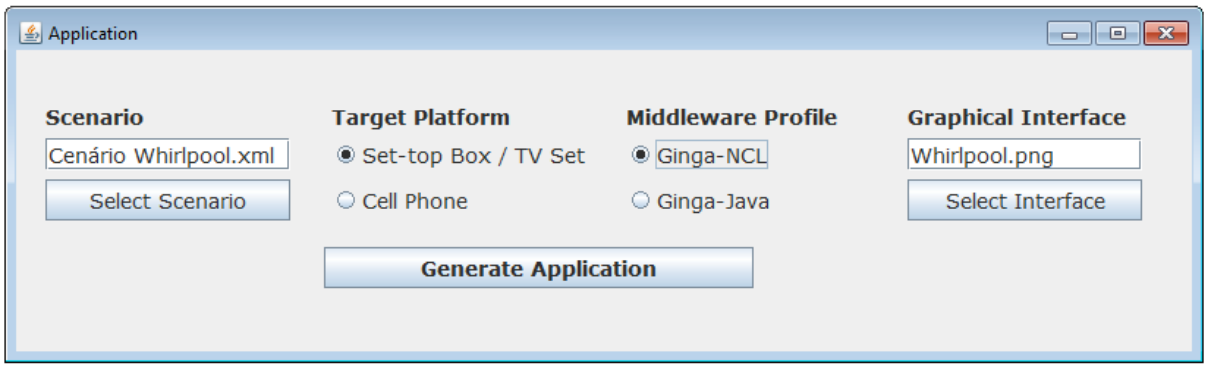

Figura 5.10 Geração da aplicação interativa.

O resultado do processo de geração da aplicação interativa são os arquivos “Cenário Whirlpool.ncl” e “Cenário Whirlpool.lua”, que é o cenário criado originalmente e que foi codificado para a linguagem NCL / LUA para ser executado sobre o *middleware* de

interatividade Ginga e, consequentemente, sobre os receptores de TV digital do padrão SBTVD. O trecho da codificação gerada pode ser observada na Figura 5.11.

```
local function start_tcp()
    request = { class="tcp", type="connect", host="192.168.1.150",port="80" }
    event.post("out",request)
end
local function keyHandler (evt)
-- EVENTO: TECLADO
    ifevt.class == 'key' and evt.type == 'press' then
        local key = evt.key
    if (key == '1') then
    service_address = '/command.php?option=Ligar'
    start_tcp()
            MSG_RETORNO = ''Accessing the service 1'
    redraw()
```

Figura 5.11 Trecho do cenário codificado em linguagem NCL / LUA.

Com a geração dos arquivos NCL e LUA, estes são copiados para uma memória externa (*pen drive*) e inseridos no receptor de TV digital. Neste estudo de caso, tanto o Refrigerador quanto o receptor de TV digital estavam conectados à mesma *Local Area Network* (LAN). Em decorrência da utilização do *framework*, este estudo de caso permitiu que o Refrigerador pudesse ser gerenciado por um receptor de TV digital, conforme ilustra a Figura 5.12:

Figura 5.12 Aplicação interativa para gerenciamento do refrigerador.

Um vídeo de demonstração desse estudo de caso em funcionamento pode ser obtido no *website* oficial do trabalho: **http://sourceforge.net/projects/dtvi4ha.**

5.2 Integração com Sistemas de Automação Predial/Residencial

Esse estudo de caso tem por objetivo demonstrar como é possível realizar a integração entre o SBTVD e os sistemas de automação. Existem atualmente no mercado diversos fabricantes e integradores de sistemas de automação predial/residencial, entretanto, neste trabalho, optou-se por utilizar os sistemas da empresa Homesystems, uma empresa gaúcha, com sede em Porto Alegre e que ao longo dos anos vem desenvolvendo projetos de pesquisa aplicada em parceria com o Grupo de Controle, Automação e Robótica (GCAR) desta universidade.

O conjunto de soluções projetado pela empresa Homesystems possui uma arquitetura com diferentes módulos que são conectados em rede. Esses módulos podem ser autônomos ou receberem comandos de um módulo controlador central, chamado *Systembox*. O *Systembox* é

um computador com sistema operacional Linux, responsável por controlar a rede HSNET com o protocolo proprietário que opera sobre a camada física RS-485 e executar funcionalidades, oferecendo comunicação remota com todos os módulos da arquitetura, através de uma interface *web*. A arquitetura Homesystems disponibiliza, ainda, um conjunto de dispositivos que podem ser utilizados em sistemas de iluminação, ar condicionado e sistemas de segurança.

Nesse estudo de caso, foi escolhida a Sala 301-A, do Programa de Pós-Graduação em Engenharia Elétrica desta Universidade. Essa sala possui um sistema Homesystems e foi projetada para automatizar alguns cenários de aula, por exemplo: quando um professor chega ao local para lecionar, a sala o reconhece, através de um leitor biométrico que fica na porta de entrada, e configura todo o ambiente de acordo com as preferências de cada professor, tais como: temperatura do ar condicionado, intensidade da iluminação, escolha de quais lâmpadas serão ligadas, entre outras. Além disso, caso o professor deseje alterar alguma configuração do ambiente, ele pode utilizar um computador e acessar a interface *web* do controlador central (*Systembox*) ou, ainda, dirigir-se até um teclado que está fixado na parede e pressionar as teclas referentes a cada novo cenário desejado. Outra funcionalidade importante do AmI é que a sala pode identificar a ausência de pessoas no local e, automaticamente, desligar os dispositivos que, eventualmente, tenham sido esquecidos ligados. Sob o ponto de vista da redução no consumo de energia, essa pode ser uma estratégia a ser utilizada em outros ambientes automatizados. Na Figura 5.13, é ilustrada a planta baixa da sala 301-A, seus dispositivos de automação e a visão da integração entre o receptor de TV digital e o sistema controlador de automação do ambiente, alvo desse estudo de caso.

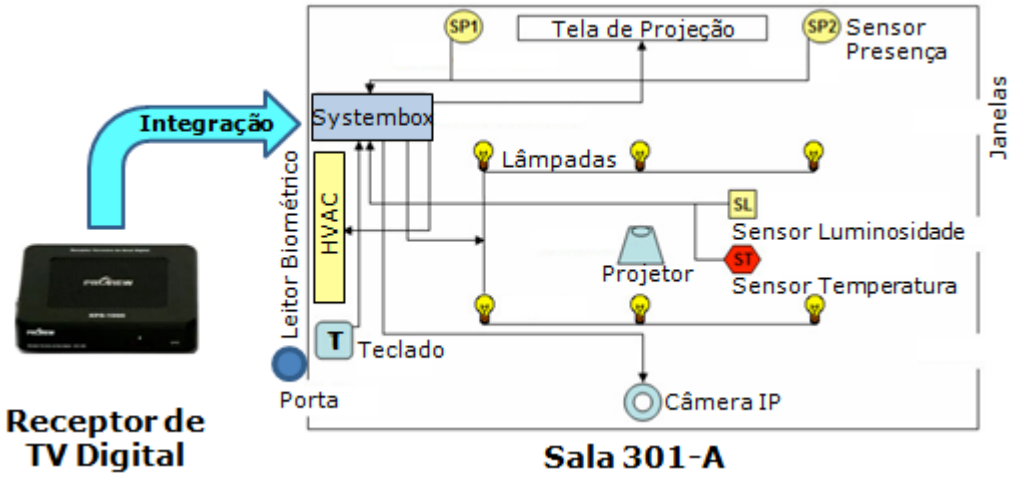

Figura 5.13 Planta baixa da sala 301-A e a integração com a TV digital.

5.2.1 Definição do Escopo e Utilização do *Framework*

Diferentes cenários para gerenciamento da sala 301-A podem ser definidos usando-se o sistema de automação predial/residencial instalado. Entretanto, para esse estudo de caso, de todos os dispositivos de automação presentes na sala foram selecionados quatro para utilização: (i) lâmpadas; (i) aparelho de ar condicionado; (iii) projetor e (iv) tela de projeção motorizada. Com o mapeamento computacional, esses dispositivos são transformados nas classes de dispositivos de automação, conforme ilustra a Figura 5.14.

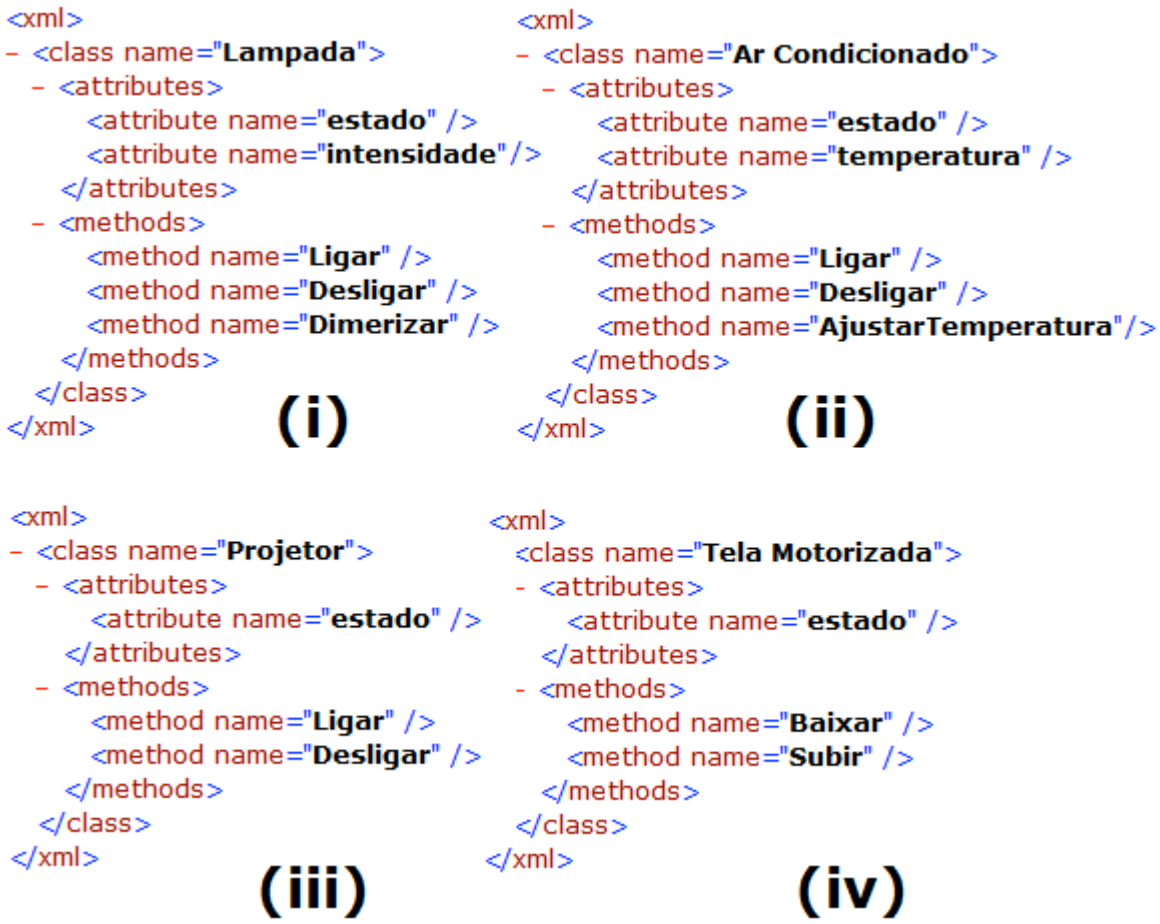

Figura 5.14 Classes de dispositivos.

As classes de dispositivos mapeadas são disponibilizadas à ferramenta de Gerenciamento de Dispositivos, a qual permite que essas classes possam ser instanciadas. Como na Sala-301-A existem duas fileiras de lâmpadas (frente e fundos), são instanciados pela ferramenta dois dispositivos lógicos: “Lâmpadas da Frente” e “Lâmpadas do Fundo”. Para não haver repetições desnecessárias, é apresentado, na Figura 5.15, apenas um exemplo dessa instanciação, pois a criação tanto para as Lâmpadas quanto para os demais dispositivos lógicos seguem a mesma estrutura.

```
<xml>
    <instantiated_device>
        <selected_class_name>Lâmpada</selected_class_name>
        <device_name>Lâmpadas da Frente</device_name>
        <attributes>
                <attribute name="estado" value="0" />
                <attribute name="intensidade" value="0" />
        </attributes>
        <methods>
                <![CDATA[ <method name = "Ligar" command =
                "http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=119&newvalue=1"/>]]>
```

```
                <![CDATA[ <method name = "Desligar" command =
                "http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=119&newvalue=0"/>]]>
                <![CDATA[ <method name = "Dimerizar" command =
                "http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=119&newvalue=50%"/>]]>
        </methods>
    </instantiated_device>
</xml>
```

Figura 5.15 Criação do dispositivo lógico "Lâmpadas da Frente".

Seguindo a mesma estrutura apresentada na Figura 5.15, também foram criados os dispositivos lógicos "Lâmpadas do Fundo", "Ar Condicionado", "Projetor" e "Tela Motorizada". Com base nesses dispositivos lógicos, são criados os serviços que serão utilizados no AmI. Nesse estudo de caso, optou-se por definir a criação de quatro serviços, cujas funcionalidades e dispositivos que os compões são apresentados na Tabela 2.

Tabela 2 Definição dos serviços.

Serviços	Funcionalidades/Requisitos	Dispositivos Utilizados
Projetar Apresentação	Ligar o projetor Baixar a tela de projeção Desligar as lâmpadas da frente Ligar o ar condicionado	Projetor Tela Motorizada Lâmpadas da Frente Ar Condicionado
Escrever no Quadro	Subir a tela de projeção Ligar as lâmpadas da frente	Tela Motorizada Lâmpadas da Frente
Retomar Apresentação	Baixar a tela de projeção Desligar as lâmpadas da frente	Tela de Motorizada Lâmpadas da Frente
Desligar Equipamentos	Desligar o projetor Subir a tela de projeção Desligar as lâmpadas Desligar ar condicionado	Projetor Tela Motorizada Lâmpadas da Frente Lâmpadas do Fundo Ar Condicionado

A partir da definição de quais serviços serão criados e quais funcionalidades de cada dispositivo serão utilizadas, a ferramenta de serviços permite gerar cada um dos serviços apresentados na Tabela 2. Seguindo o princípio para não repetir o mesmo conceito, a Figura 5.16 apresenta o resultado do processo de criação de apenas um dos serviços que foram

definidos. Assim como na fase de instanciação das classes, o processo de criação de serviços é idêntico para todos.

```
<xml>
  <service>
    <created_service name="Projetar Apresentação" group="Aula" />
    <service_description>Ministrar aula com slides</service_description>
    <devices>
        <device name="Projetor" method="Ligar" />
        <![CDATA[
         <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=114
        &newvalue=1</command>
        ]]>
        <device name="Tela Motorizada" method="Baixar" />
        <![CDATA[
       <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=116
        &newvalue=1</command>
        ]]>
        <device name="Lâmpadas da Frente" method="Desligar" />
        <![CDATA[
        <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=119
        &newvalue=0</command>
        ]]>
        <device name="Ar Condicionado" method="Ligar" />
        <![CDATA[
        <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=112
        &newvalue=1</command>
        ]]>
     </devices>
   </service>
</xml>
```

Figura 5.16 Criação do serviço “Projetar Apresentação”.

Além do serviço “Projetar Apresentação” ilustrado na Figura 5.16, foram criados também os demais serviços: “Escrever no Quadro”, “Retomar Apresentação” e “Desligar Equipamentos”. Esses serviços ficam disponíveis, no Repositório de Serviços, para serem utilizados na construção dos cenários de automação. Nesse estudo de caso, foi definido um único cenário, chamado “Gerenciar o Ambiente”, criado com o auxílio da ferramenta *construção de cenários* e sendo composto pelos quatro serviços descritos na Tabela 2. Na Figura 5.17, é apresentado um trecho do arquivo XML referente ao cenário “Gerenciar o Ambiente”.

```
<xml>
    <created_scenario>
    <scenario name="Gerenciar o Ambiente" />
        <used_services>
            <service name="Projetar Apresentação" group="Aula" />
            <description>Ministrar aula com slides</description>
            <command_interface>Remote Control - Key 1</command_interface>
                <![CDATA[
                <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=114
                &newvalue=1</command>
```

```
                ]]>
                <![CDATA[
                 <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=116
                &newvalue=1</command>
                ]]>
                <![CDATA[
                <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=119
                &newvalue=0</command>
                ]]>
                <![CDATA[
                <command>http://10.1.7.30/monitor/monitor.cgi?ref_page=cmd&unit=112
                &newvalue=1</command>
                ]]>
            <service name="Escrever no Quadro" group="Aula" />
            <description>Serviço para utilizar interromper os slides e utilizar o quadro</description>
            <command_interface>Remote Control - Key 2</command_interface>
                ....
                ....
                ...
                ...
        </used_services>
    </created_scenario>
</xml>
```

Figura 5.17 Trecho de configuração do cenário Whirlpool.

Com base nas informações contidas no arquivo de configuração do cenário "Gerenciar o Ambiente", é construída a interface gráfica da aplicação interativa para gerenciamento da sala, apresentada na Figura 5.18.

Figura 5.18 Interface gráfica da aplicação interativa para gerenciamento da Sala 301-A.

Por fim, a próxima tarefa a ser executada utilizando o *framework* é a construção da aplicação interativa, ou seja, utilizar a ferramenta para geração de código. A exemplo das Figuras 5.10 e 5.11 apresentadas no estudo de caso anterior, são criados para esse estudo de

caso os arquivos “Gerenciar o Ambiente.ncl” e “Gerenciar o Ambiente.lua”, os quais são disponibilizados ao receptor de TV digital através do *pen drive*. O resultado dessa série de passos é a execução da aplicação correspondente ao cenário de acesso aos serviços que foram criados para o gerenciamento da sala 301-A. Na Figura 5.19, é apresentada a aplicação interativa em execução no STB (XPS-1000) e sendo visualizada na TV. Nesse estudo de caso, tanto o receptor de TV digital quanto o controlador central de automação da Homesystems (*Systembox*) estavam interligados na mesma rede local (LAN), sendo que a comunicação entre eles é realizada através do protocolo TCP/IP.

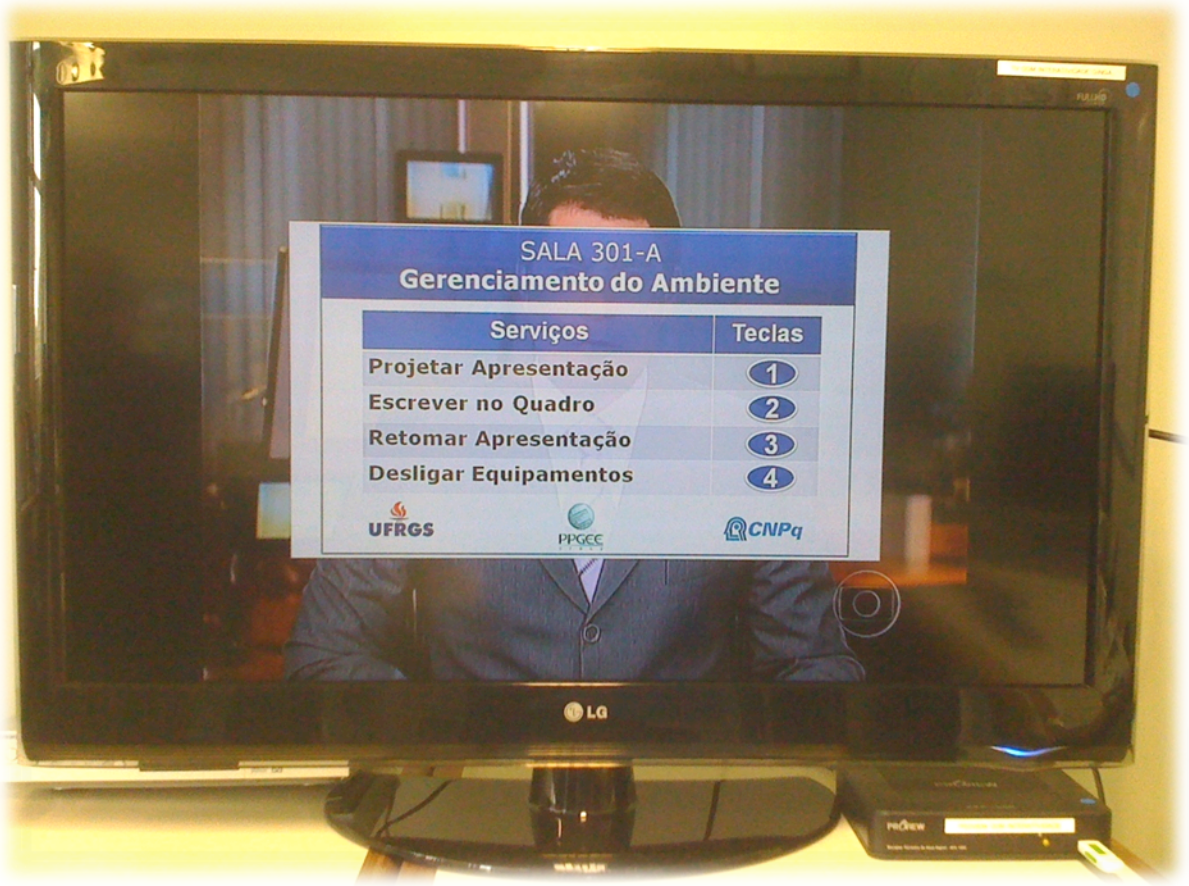

Figura 5.19 Aplicação de gerenciamento da Sala 301-A sendo executada.

5.3 Acesso a Serviços Externos: Estação Meteorológica RBS TV

Uma das principais características de um AmI é a possibilidade de ele agregar novos serviços e poder evoluir de acordo com as necessidades dos usuários. Entretanto, esses

serviços não precisam estar, necessariamente, relacionados com os dispositivos de automação locais. Um exemplo disso é o acesso via TV digital aos serviços bancários, como consultas e movimentações financeiras utilizando aplicações interativas que são disponibilizadas pelos bancos (STICKERCENTER, 2010). Outros serviços, baseados na *web*, que são utilizados com frequência, referem-se aos noticiários e informações meteorológicas. Tomando como exemplo este último, o presente estudo de caso demonstra como o AmI pode acessar, por meio da TV digital, serviços meteorológicos que são disponibilizados por uma estação meteorológica remota. Em outras palavras, nesse estudo de caso é utilizado o *framework* para permitir que seja criado um cenário para busca de informações climáticas, integrando os receptores de TV digital com dispositivos de automação remotos que oferecem serviços pela Internet.

5.3.1 Definição do Escopo

O dispositivo de automação utilizado nesse estudo de caso é uma estação meteorológica, modelo Vantage Pro 2, fabricada pela empresa Davis (DAVIS, 2010) e instalada na emissora de televisão Rede Brasil Sul de Comunicação (RBS TV), com sede em Porto Alegre. Essa estação tem a capacidade de oferecer diversos serviços, tais como temperatura, umidade do ar, velocidade do vento, direção do vento e outros. Dentre esses serviços, no estudo de caso são utilizados apenas os três primeiros citados. A RBS TV possui inúmeros serviços que podem ser acessados através da Internet, entretanto, esse estudo de caso permitiu que as informações climáticas pudessem ser acessadas pelas aplicações interativas que são executadas nos receptores de TV digital. Dessa forma, o *framework* proposto neste trabalho, foi utilizado a fim de permitir a criação de um cenário dentro do contexto de AmI e que acessa serviços externos de uma estação meteorológica remota. A

arquitetura de comunicação utilizada pela emissora de TV para disponibilizar, via Internet, os dados da estação meteorológica é apresentada na Figura 5.20.

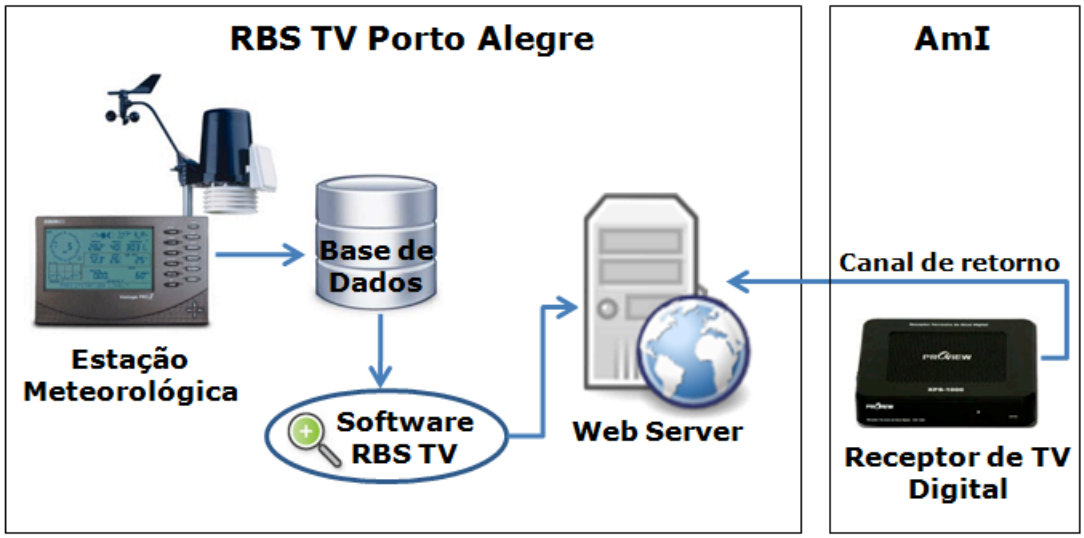

Figura 5.20 Arquitetura de comunicação para acesso aos serviços meteorológicos.

A estação meteorológica possui um sistema de coleta e armazenamento de informações disponibilizado pela própria fabricante e que acompanha o equipamento. Adicionalmente foi desenvolvido, pelo Departamento de Tecnologia da RBS TV, um *software* que acessa a base de dados da estação meteorológica, realiza consultas, extrai as informações e as disponibiliza em um servidor *web,* o qual fica disponível para ser acessado pelos receptores de TV digital.

5.3.2 Requisitos para Acesso aos Serviços

Para acesso aos serviços de informações meteorológicas que foram selecionados (temperatura, umidade e velocidade do vento) o *Web Server* da RBS TV disponibiliza endereços individuais para cada solicitação. Por exemplo, os parâmetros utilizados no endereço de acesso ao serviço *temperatura* são diferentes dos utilizados no serviço *umidade*. Isso permite que os serviços possam ser utilizados individualmente ou em conjunto por um determinado cenário.

5.3.3 Utilização do *Framework*

No mapeamento computacional, é criada uma classe chamada "Estação Meteorológica", com os seus respectivos atributos e métodos ilustrados na Figura 5.21.

```
<xml>
    <class name="Estação Meteorológica">
        <attributes>
            <attribute name="temperatura" />
            <attribute name="umidade" />
            <attribute name="velocidade_vento" />
        </attributes>
        <methods>
            <method name="Informar temperatura" />
            <method name="Informar umidade" />
           <method name="Informar velocidade_vento" />
        </methods>
    </class>
</xml>
```

Figura 5.21 Mapeamento computacional da estação meteorológica.

Na Figura 5.22, é apresentada a instanciação da classe "Estação Meteorológica", ou seja, é criado o dispositivo lógico "Estação Meteorológica RBS TV", o qual herda os atributos e métodos da classe utilizada.

```
<xml>
    <instantiated_device>
        <selected_class_name>Estação Meteorológica</selected_class_name>
        <device_name>Estação Meteorológica RBS TV</device_name>
        <attributes>
                <attribute name="temperatura" value="0" />
                <attribute name="umidade" value="0" />
                <attribute name="velocidade_vento" value="0" />
       </attributes>
       <methods>
                <![CDATA[ <method name = "Informar temperatura" command =
               "http://clictempo.clicrbs.com.br/estacao/temp.php"/>]]>
               <![CDATA[ <method name = "Informar umidade" command = "
               http://clictempo.clicrbs.com.br/estacao/umidade.php"/>]]>
               <![CDATA[ <method name = "Informar velocidade_vento" command =
               "http://clictempo.clicrbs.com.br/estacao/vento.php"/>]]>
        </methods>
    </instantiated_device>
</xml>
```

Figura 5.22 Criação do dispositivo lógico "Estação Meteorológica RBS TV".

Com o dispositivo lógico "Estação Meteorológica RBS TV" instanciado, todos os métodos desse dispositivo são utilizados na criação dos serviços, definidos como: (i) Verificar temperatura: serviço que utiliza o método "Informar temperatura" do dispositivo; (ii) Verificar umidade: utiliza o método "Informar umidade" e (iii) Obter a velocidade do vento: utiliza o serviço "Informar velocidade_vento". Esses serviços são criados da mesma forma

como nos estudos de casos anteriores, utilizando a ferramenta de criação. Após a construção dos serviços, é definido o cenário composto pelos três serviços criados anteriormente. A interface gráfica da aplicação de interação referente a esse cenário pode ser visualizada na Figura 5.23. Já o resultado final da integração entre os receptores de TV digital e a estação meteorológica instalada na RBS TV, em que são acessados os serviços externos, pode ser observada na Figura 5.24, a qual ilustra a aplicação em execução no STB.

Figura 5.23 Interface gráfica da aplicação para acesso aos serviços meteorológicos.

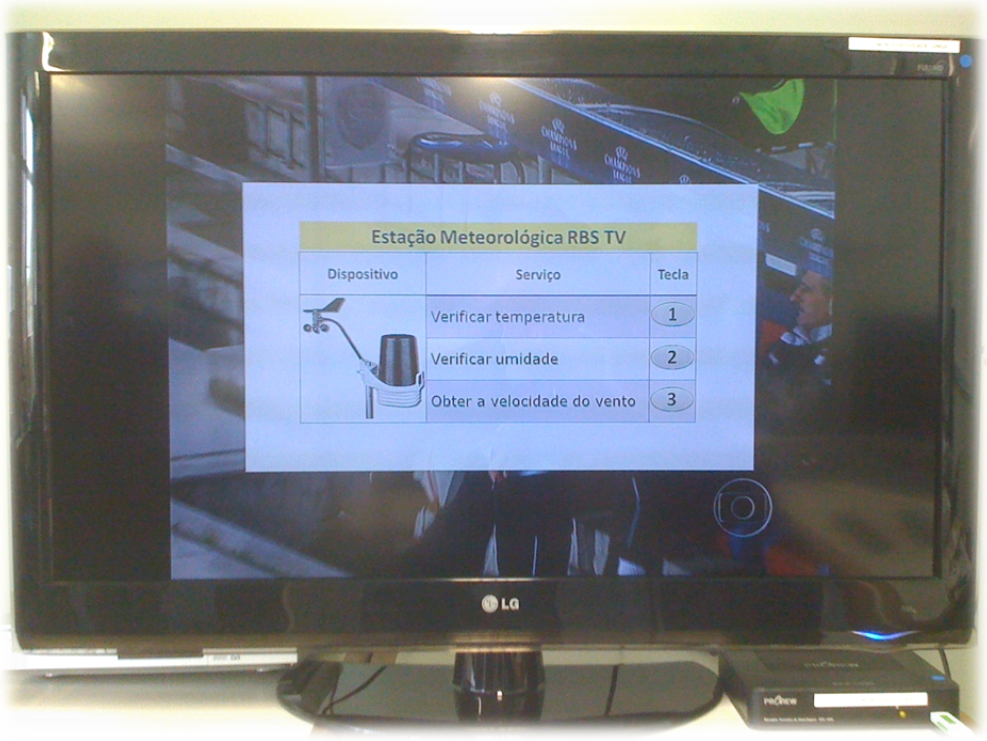

Figura 5.24 Aplicação de acesso aos aerviços meteorológicos em execução.

5.4 CONSIDERAÇÕES SOBRE OS ESTUDOS DE CASOS

Os estudos de casos foram definidos para permitir que a proposta deste trabalho pudesse ser demonstrada, através de exemplos, e validada. No primeiro estudo de caso, foi apresentado o resultado de um projeto real, implementado para a participação de um concurso de inovação em ciência e tecnologia, chamado INOVA! e promovido pela empresa Whirlpool S.A. Nesse estudo de caso, foi destacado como os eletrodomésticos podem ser inseridos no contexto dos AmIs, sendo conectados a outros dispositivos, compartilhando serviços e podendo ser integrado ao SBTVD, através de aplicações de gerenciamento que são executadas sobre o *middleware* de interatividade presente nos receptores de TV digital. No segundo estudo caso, foi apresentada a utilização do *framework* para integrar grande parte de um AmI com o SBTVD. Nesse estudo de caso, foram utilizados alguns dispositivos de automação de uma sala de aula com o objetivo de permitir a criação de cenários automatizados com gerenciamento, também, por receptores de TV digital. O estudo de caso demonstrou a possibilidade de comunicação e integração entre as aplicações interativas de TV e os sistemas de automação predial/residencial. No último estudo de caso, o foco estava em demonstrar a capacidade de um AmI acessar serviços externos de outros dispositivos através da Internet. Nesse estudo de caso, foi criado um cenário específico para meteorologia, em que as aplicações interativas utilizavam os serviços de acesso e consulta a informações como: temperatura, umidade e velocidade do vento.

Em linhas gerais, a utilização do *framework* proposto neste trabalho permite que sejam geradas aplicações interativas de gerenciamento de AmIs. Entretanto, as ferramentas implementadas podem ser utilizadas na criação de outras aplicações, específicas para emissoras de televisão, as quais podem se beneficiar da arquitetura do *framework* para construir toda a aplicação e, então, transmiti-la pelo ar junto com o sinal de áudio e vídeo da programação, resultando no acesso aos serviços que são oferecidos na aplicação. Esse acesso,

por parte do usuário/telespectador, pode ser realizado através do canal de retorno, ou canal de interatividade, como é conhecida a infraestrutura de comunicação de dados que interliga, via Internet, os receptores de TV digital das residências com os computadores ou equipamentos eletrônicos das emissoras de TV. Dessa forma, o *framework* proposto neste trabalho permite que sejam criados cenários para gerenciamento de AmIs compostos tanto por serviços locais quanto externos, integrando o SBTVD com dispositivos e sistemas de automação predial/residencial presentes nos AmIs.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi proposto um *framework* para integração entre AmIs e o SBTVD, sendo destacada a interatividade da TV digital como uma das formas de interação multimodal entre usuários e cenários de automação predial/residencial. Nos últimos anos os AmIs vêm sendo bastante comentados por oferecer serviços e suporte inteligente aos usuários, em que conforto, praticidade e otimização de energia não são, somente, os únicos atrativos. Os AmIs podem oferecer todo um suporte personalizado para pessoas idosas ou portadoras de necessidades especiais. Além disso, os AmIs vão além de residências automatizadas, pois eles oferecem serviços com maior grau de abstração, capazes de se adaptar à presença e aos requisitos dos usuários, além de oferecer a multimodalidade de interação entre humanos e ambiente, a qual pode ser realizada por comandos de voz, reconhecimento de gestos, leitores de radiofrequência, *smartphones* e até mesmo através das TVs conectadas. Inicialmente as TVs tinham como principal característica oferecer entretenimento aos telespectadores, em que o aparelho realizava a decodificação do sinal de áudio e de vídeo transmitido pelos radiodifusores. Atualmente, esses aparelhos possuem recursos computacionais que permitem, além da execução de funções tradicionais de uma TV, conexão com a Internet, realização de videoconferências, acesso a serviços bancários, integração com redes sociais e diversas outras funcionalidades. Essa série de serviços agregados aos televisores é realizada através da interatividade presente em aplicativos como os *widgets* ou em *middlewares* que são instalados nos receptores de TV digital e oferecem suporte aos mais variados domínios de aplicação.

Neste trabalho, a principal contribuição foi a definição de uma arquitetura computacional que permitisse a integração entre essas duas áreas, sendo um dos principais objetivos oferecer mecanismos que pudessem tornar um AmI totalmente gerenciável por receptores de TV digital compatíveis com o *middleware* de interatividade presente no SBTVD, o Ginga. Isso foi realizado através da especificação de um *framework* em que seus

componentes permitiram que todo o processo de integração fosse contemplado. Esse processo vai desde a identificação e o mapeamento dos dispositivos de automação que estão presentes nos AmIs, passando pela criação de serviços e cenários de automação desejados, até chegar ao processo de geração de código. Isto é, a partir do mapeamento computacional de um AmI, faz-se possível criar aplicações interativas para gerenciamento do ambiente. Para isso, foram reunidos diversos trabalhos relacionados e identificados os desafios comumente encontrados em AmIs, como o mapeamento de dispositivos, a descoberta de serviços, a integração aberta entre diferentes tecnologias e a interação multimodal nesses ambientes (MANN & HELAL, 2002). Essa pesquisa serviu como base para a proposta de integração entre os AmIs e o SBTVD, sendo definida uma arquitetura conceitual que foi apresentada e discutida neste trabalho. Além disso, foi criada uma arquitetura de implementação, cujo objetivo era criar subsídios para a validação dos conceitos propostos. Foram definidos e implementados componentes de *software* e ferramentas computacionais para serem utilizados em todas as fases do processo de integração. A arquitetura modular proposta neste trabalho ofereceu flexibilidade ao *framework*, uma vez que foi definido um conjunto de componentes com padrão de interação entre eles. Essa arquitetura permitiu que os dispositivos físicos de automação presentes no AmI fossem mapeados para o mundo computacional, através de modelos orientados a objetos e através da descrição das classes de dispositivos. Essas classes de dispositivos puderam ser instanciadas, resultando nos dispositivos lógicos que eram utilizados na criação de serviços. Outro resultado importante alcançado com uma arquitetura foi a criação de cenários independentes da plataforma de *hardware* em que são executados. Esses cenários podem ser compostos por diversos serviços e podem ser totalmente reaproveitados em qualquer plataforma-alvo do SBTVD. Essa reutilização leva a uma otimização no tempo de construção de novos projetos, pois os componentes de *software* e as ferramentas desenvolvidas permitem que os serviços contidos nos cenários possam ser

removidos, inseridos ou substituídos. Com base nos cenários foi possível gerar, automaticamente, as aplicações interativas para gerenciamento do AmI. Essas aplicações interativas nada mais são do que os cenários de automação, codificados em uma linguagem de programação compatível com o Ginga e configurado para ser executado na plataforma-alvo, em que são considerados os aspectos do *hardware*, como, por exemplo, resolução de tela. Isso se deve ao fato de os receptores de TV digital terem características computacionais diferentes uns dos outros, principalmente no que se refere a visualização de informações, diferentes entre um telefone celular e um televisor.

Na validação do trabalho, foram apresentados três estudos de casos que serviram para exemplificar a utilização do *framework* proposto neste trabalho. Um destaque pode ser dado ao estudo de caso de gerenciamento de residências automatizadas via TV digital, resultante de um projeto em parceria entre uma fabricante multinacional de eletrodomésticos e esta universidade. Nesse estudo de caso, todas as fases de desenvolvimento do projeto utilizando o *framework* foram contempladas e ilustradas através do uso das ferramentas que foram desenvolvidas. Além disso, nos outros estudos de casos foi possível observar como os conceitos e as ferramentas propostas são utilizadas na presente tese.

O fato de a arquitetura do *framework* ser flexível permite que os seus componentes de *software* implementem outras funcionalidades, sendo ampliados de acordo com a evolução do AmI. Um exemplo disso são os componentes para descoberta de dispositivos e de serviços, os quais podem, além de realizar as funções criadas para validar os conceitos de integração propostos neste trabalho, implementar novas funcionalidades para comunicação com outras tecnologias amplamente difundidas, tais como OSGi, *Web Services* ou ontologias.

A inserção de uma nova modalidade de interação nos AmIs pode ser bastante atraente, uma vez que, tanto em telefones celulares quanto em TVs com suporte ao Ginga, o usuário pode acessar os serviços e os dispositivos de automação existentes em cada cenário e

gerenciar a sua residência. Com o avanço mundial do padrão brasileiro de TV digital, sendo adotado por diversos países, novos modelos de negócios podem surgir com o objetivo de explorar, entre outras áreas, o mercado de automação predial/residencial. Os radiodifusores poderiam oferecer esse tipo de aplicação em programações diferenciadas ou, até mesmo, empresas de automação desenvolver essas aplicações compatíveis com o seu portfólio e as disponibilizarem em repositórios na *web*. Outra possibilidade seria os próprios fabricantes de receptores de TV digital embarcar essas aplicações em parceria com as empresas de automação.

Pelo fato de a especificação do *middleware* de interatividade do SBTVD ser recente não há, até o momento, ferramentas de autoria que permitam a criação das aplicações interativas que são executas sobre esse *middleware*. Existem apenas alguns *plugins* que auxiliam na codificação, porém esse auxílio é direcionado aos especialistas em programação. Dessa forma, as ferramentas desenvolvidas no âmbito desta tese permitem que os cenários construídos sejam automaticamente transcritos para a linguagem do *middleware*. Essa é uma importante contribuição, pois permite a geração automática de código em linguagens NCL / Lua e JavaDTV. Essas ferramentas foram implementadas para serem executadas em PCs, e o resultado da sua utilização gera as aplicações interativas que são executadas nos receptores de TV digital. Entretanto, todos esses componentes e ferramentas poderão ser executadas diretamente nos receptores desde que, no futuro, estes ofereçam maior capacidade de processamento e recursos de memória, que hoje são limitados se comparados aos atuais computadores pessoais.

Além disso, a proposta desta tese concentra o seu ponto de integração na camada de aplicação, estando aderente às normas do padrão brasileiro de TV digital e não necessitando de ampliação da especificação do *middleware* de interatividade para que exista essa integração. Outro fator importante é que o *framework* oferece todos os recursos para que a

proposta de integração entre AmIs e o SBTVD apresentada neste trabalho seja, de fato, realizada.

Como sugestão para trabalhos futuros, são destacadas as possibilidades de migração de todos os componentes do *framework* para os receptores de TV digital, o que permitiria que tanto as fases de projeto quanto de execução fossem realizadas em uma única plataforma, aproveitando uma possível evolução da arquitetura de *hardware* dos receptores. Outra possibilidade, seria a criação de aplicações interativas de gerenciamento do AmI capazes de descobrir, em tempo de execução, novos serviços e dispositivos de automação que estariam disponíveis no AmI. Isso resultaria em cenários mais flexíveis e poderia otimizar o processo de geração de novos cenários. Também poderia ser explorada a questão dos requisitos temporais para a execução de determinados serviços, a fim de atender a alguma necessidade. Além disso, poderia ser importante uma proposta para geração automática de interfaces gráficas, em que um componente poderia obter as informações de configuração dos cenários de automação gerados pelo *framework* e criar uma interface gráfica para a aplicação interativa correspondente.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT) **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 de aplicação XML para codificação de aplicações. Rio de Janerio, 2009.

_____. **NBR 15606-4**: televisão digital terrestre: codificação de dados e especificações de transmissão para radiodifusão digital: Parte 4: ginga-J – ambiente para a execução de aplicações procedurais, Rio de Janeiro, 2010.

_____. **NBR 15606-5**: televisão digital terreste: codificação de dados e especificações de transmissão para radiodifusão digital: Parte 5: ginga-NCL para receptores portáteis - linguagem de aplicação XML para codificação de aplicações. Rio de Janeiro, 2009.

_____. **NBR 15607-1**: televisão digital terrestre: canal de interatividade: Parte 1: protocolos, interfaces físicas e interfaces de software. Rio de Janeiro, 2008.

AMIGO PROJECT. **Ambient intelligence for the networked home environment**. Disponível em: <http://www.hitech-projects.com/euprojects/amigo>. Acesso em: 30 out. 2008.

ANASTASOPOULOS, M. et al. Towards a reference middleware architecture for ambient intelligent systems. In: WORKSHOP FOR BUILDING SOFTWARE FOR PERVASIVE COMPUTING, 2005, San Diego. **Procedings…** OOPSLA, 2005.

ANDREATA, J. M. **InteraTV:** um portal para aplicações colaborativas em tv digital interativa utilizando a plataforma. 2006. 110 f. Dissertação (mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2006.

ARAUJO, J. J. **Framework orientado a objetos para desenvolvimento de aplicações de automação predial e residencial**. 2005. 116 f. Dissertação (mestrado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005.

ASSOCIATION OF RADIO INDUSTRIES AND BUSINESSES (ARIB). **Overview of standards and technical reports.** Disponível em: <http://www.arib.or.jp/english>. Acesso em: 20 set. 2010.

ASSOCIATION OF RADIO INDUSTRIES AND BUSINESSES (ARIB). **B-24**: terrestrial integrated services digital broadcasting: XML-based multimedia coding scheme: ARIB standard b-24 data coding and transmission specifications for digital broadcasting, version 4.0. Tokyo, 2004.

ARK, W. S.; SELKER, T. A look at human interaction with pervasive computers. **IBM Systems Journal**, Riverton, v. 38, n. 9, p. 504 – 507, 1999.

ARTS, E. Ambient intelligence: a multimedia perspective. **IEEE Multimedia**, Los Alamitos, v. 11, n. 1, p. 12 – 19, Jan./Mar, 2004.

ARTS, E.; HARWING, R., SCHUURMANS, M. **Ambient intelligence**. New York: McGraw-Hill, 2002.

ASSAD, M. et al. AR phone: accessible augmented reality in the intelligent environment. In: AUSTRALIASIAN COMPUTER HUMAN INTERACTION CONFERENCE**,** OZCHI 2003. **Proceedings…** Brisbane: University of Queensland, 2003, p. 232 – 235.

ADVANCED TELEVISION SYSTEMS COMMITTEE (ATSC). **DTV applications software environment (DASE).** Disponível em: <http://atsc.org>. Acesso em: 20 set. 2010.

BARBOSA, S. D. J, SOARES, L. F. G. **TV digital interativa no Brasil se faz com ginga**: fundamentos, padrões, autoria declarativa e usabilidade, 2008. Disponível em: <http://www.tvdi.inf.br/upload/artigos/jai2008-artigo.pdf>. Acesso em: 13 set. 2010.

BLUETOOTH. **The official bluetooth**: technology info site. Disponível em: <http://www.bluetooth.com/>. Acesso em: 10 Ago. 2010.

BOLSANI, Caio, A., M. **Residências inteligentes**. São Paulo: Livraria da Física, 2004. 332 p.

BORGES, R. C. **Uma arquitetura para descoberta de serviços em ambientes de TV digital interativa**. 2007. 63 f. Dissertação (mestrado) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Goiás, Goiânia, 2007.

BUSINESS PROCESS EXECUTION LANGUAGE (BPEL). **Oracle BPEL**. Disponível em: <http://www.oracle.com/technetwork/middleware/bpel/overview/index.html>. Acesso em: 14 jan. 2011.

BRACKMANN, C. P. **Usabilidade em TV digital**. 2010. 199 f. Dissertação (mestrado) – Programa de Pós-Graduação em Ciência da Computação, Universidade Católica de Pelotas, Pelotas, 2010.

BRASIL. Decreto-lei 4.901, de 26 de novembro de 2003. Institui o sistema brasileiro de televisão digital: SBTVD. **Diário Oficial da União**, Brasília, DF, Seção 1, p. 7.

CABRER, M. R. et al. Controlling the smart home from tv. In: INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS, 2006, Las Vegas: **Proceedings…** New York: IEEE, 2006, p. 421 - 429.

CIBER DO BRASIL (CBR). Disponível em: <http://www.ciberdobrasil.com.br/>. Acesso em: 18 nov. 2009.

DAMASCENO, J. R. **Middleware ginga**: interatividade no SBTVD. 2008. Disponível em: <http://www.midiacom.uff.br/~debora/fsmm/trab-2008-2/middleware.pdf>. Acesso em: 20 set. 2010.

DAVIS. **Wireless & cabled professional & home weather stations by Davis**. Disponível em: <http://www.davisnet.com/weather/products/vantagepro.asp>. Acesso em: 30 nov. 2010

DE LUCA, C. **TV Digital**: ginga-NCL é agora recomendação H.761 da UIT, 2009. Disponível em: <http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=18718&sid=54> . Acesso em: 20 set. 2010.

DEKTEC. **DTA-115** - Multi-Standard VHF/UHF Modulator for PCI Bus. Disponível em: <http://www.dektec.com/Products/PCI/DTA-115/index.asp>. Acesso em: 19 nov. 2010.

DI NITTO, E.; SASSAROLI, G.; ZUCCALÀ, M. Adaptation of web contents and services to terminals capabilities: the @terminals approach. In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS, 23-26 Mar. 2003, Dallas-Fort Worth. **Proceedings…** New York: IEEE Computer Society, 2003, p. 433 – 440.

DOLAN, M. A. **Report on television data applications**, 2001. Disponível em: <http://www.itl.nist.gov/div897/staff/barkley/tv-data-apps-mdolan.pdf>. Acesso em: 04 out. 2010.

DTV. **Site oficial da TV Digital Brasileira**. Disponível em: <http://www.dtv.org.br>. Acesso em: 20 set. 2010.

DUCATEL, K. et al. **Scenarios for ambient intelligence** (ISTAG Report). Seville: Institute for Prospective Technological Studies (European Commission), 2001.

EASY LIVING. **Microsoft easy living project**. Disponível em <http://research.microsoft.com/easyliving>. Acesso em: 30 out. 2008.

ECHELON. **LonWorks control networking**. Disponível em: <http://www.echelon.com/products/lonworks_control_networking.htm>. Acesso em: 10 Ago. 2010.

EDWARDS, W.; GRINTER, R. **At home with ubiquitous computing**: seven challenges. Berlin: Springer-Verlag, 2001.

EDWARDS, W. K. Discovery Systems in Ubiquitous Computing. **IEEE pervasive computing**, New York, v. 5, n. 2, p. 70-77, Apr./June 2006.

EMBASSI. **The embassi project**: multimodal assistance for infotainment and service infrastructures. Disponível em <http://www.embassi.de/estart.html>. Acesso em: 21 out. 2008.

ESPECIAL TV digital. **Revista da Set**, Rio de Janeiro, n. 105, 2009. Disponível em: <http://www.set.com.br/artigos/indice.htm#ed105>. Acesso em: 23 set. 2010.

EUROPEAN INTERACTIVE TV CONFERENCE (EUROITV). **Interactive television**. Disponível em: <http://euroitv2009.org/proceedings.html>. Acesso em: 27 set. 2010.

FAGOR. **Maior-Domo**. Disponível em <http://www.fagorelectronica.com/trata/domogestor.html>. Acesso em: 06 maio 2007.

FILGUEIRAS, L. V. L.; GIANNOTO, E. C. Estudo de aplicações interativas na TV usando rastreamento do olhar. In: SIMPÓSIO INTERNACIONAL DE TELEVISÃO DIGITAL (SIMTVD), Bauru. **Anais**... Bauru: UNESP, 2009.

FÓRUM SBTVD. **Fórum do Sistema Brasileiro de TV Digital Terrestre**. Disponível em: <http://www.forumsbtvd.org.br/materias.asp?id=454>. Acesso em: 30 nov. 2010.

FREESCALE. **Freescale semiconductor**. Disponível em: <http://www.freescale.com/>. Acesso em: 02 fev. 2010.

FRIAS, L. **Rede Globo**: aplicações interativas para TV digital. Disponível em: <http://www.set.com.br/sistema/upload/upload/UploadFolder/leonardo%20frias.pdf>. Acesso em: 27 set. 2010.

GAMEDEV. **GameDev.net**: an introduction to Lua by Ash Matheson. Disponível em: <http://www.gamedev.net/reference/programming/features/lua>. Acesso em: 19 nov. 2010.

GAMMA, E. et al. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

GÁRATE, A.; HERRASTI, N.; LÓPEZ, A. GENIO: an ambient intelligence application in home automation and entertainment environment. In: CONFERENCE ON SMART OBJECTS AND AMBIENT INTELLIGENCE, 2005, Grenoble. **Proceedings…** New York: ACM Press, 2005, p. 241 - 245.

GLOBALLY EXECUTABLE MHP (GEM). **Specification 1.2.2 (including IPTV), DVB Document A 139 r4, 2009**. Disponível em: <http://www.mhp.org/specs/a139_GEM_122_r4.pdf>. Acesso em: 20 set. 2010.

GINGA. **Brazilian digital TV middleware specification**. Disponível em: <http://http://www.ginga.org.br>. Acesso em: 20 set. 2010.

GLOBO. **Site oficial da TV Globo**. Disponível em: <http://redeglobo.globo.com/>. Acesso em: 27 set. 2010.

GOOGLE TV. **Introducing Google TV**: TV, apps, search, and the web...together at last. Disponível em: <http://www.google.com/tv>. Acesso em: 08 nov. 2010.

GROPPE, J.; MUELLER, W. Profile management technology for smart customizations in private home applications. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 2005, Copenhagen: **Proceedings…** New York: IEEE Computer Society, 2005, p. 226 - 230.

GUEDES, G. T. A. **UML 2 guia de consulta rápida**. São Paulo: Novatec, 2005.

HAGRAS H. et al. Creating an ambient-intelligence environment using embedded agents. **IEEE Intelligent Systems**, Los Alamitos, v. 19, n. 6, p. 12 – 20, Nov./Dec. 2004.

HOME AUTOMATION INC (HAI). **Automation Simplified**: apllying technology for comfort, convenience and safety. Disponível em: <http://www.homeauto.com/main.asp>. Acesso em: 31 out. 2008.

HELAL, A.; HAMMER, J. UbiData: requirements and architecture for ubiquitous data access. In: **SIGMOD Rec Journal…** New York: ACM. V. 33, n. 4, p. 71 – 76, Dec. 2004.

HELAL, S. et al. The Gator tech smart house: a programmable pervasive space. **Computer,** Long Beach, v. 38, n. 3, p. 50 – 60, Mar. 2005.

HOMESYSTEMS. **Ambientes inteligentes, automação e segurança**. Disponível em <http://www.homesystems.com.br>. Acesso em: 27 de set. 2010.

HOPKINS, B. **Creating interactive TV applications with the Tru2way platform and OCAP**. 2009. Disponível em: <http://java.sun.com/developer/technicalArticles/javame/iptv-tru2way>. Acesso em: 04 de out. 2010.

HSU, J. M.; WU, W. J.; CHANG, I. R. Ubiquitous multimedia information delivering Service for smart home. In: INTERNATIONAL CONFERENCE ON MULTIMEDIA AND UBIQUITOUS ENGINEERING, 2007, Seoul: **Proceedings…** New York: IEEE Computer Society, 2007, p. 341 – 346.

HXD. **HXD Interactive Television**. Disponível em: <http://www.hxd.com.br/>. Acesso em: 27 set. 2010.

IDGNOW. **TV Digital**: Globo começa a transmitir aplicação interativa do Brasileirão. Disponível em: <http://idgnow.uol.com.br/blog/circuito/2010/09/09/tv-digital-globo-comeca-a-transmitir-aplicacao-interativa-do-brasileirao/>. Acesso em: 27 set. 2010.

INDULSKA, J. et al. An open architecture for pervasive systems. In: INTERNACIONAL WORKING CONFERENCE ON NEW DEVELOPMENTS IN DISTRIBUTED APPLICATIONS AND INTEROPERABLE SYSTEMS, 2001, Krakow. **Proceedings…** Norwell: IEEE/IFIP, 2001, p. 175 – 188.

INSTEON. **Wireless home control solutions for lighting, security, HVAC and A/V systems**. Disponível em: <http://www.insteon.net>. Acesso em: 31 out. 2008.

INTERNATIONAL TELECOMMUNICATION UNION (ITU) **H.761**: nested context language (NCL) and ginga-NCL for IPTV (2009). Disponível em: <http://www.itu.int/itu-t/aap/AAPRecDetails.aspx?AAPSeqNo=1894>. Acesso em: 20 set. 2010.

INTERNATIONAL TELECOMMUNICATION UNION (ITU-T). **X.509**: | **ISO/IEC 9594-8**: 2001: information technology: open systems interconnection: Geneva: public-key and attribute certificate frameworks, 2001.

KAPLAN, A.; LUNN, J. FlexXML: Engineering a more flexible and adaptable web. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: coding and computing, Apr. 2001. Las Vegas. **Proceedings...** New York: IEEE Computer Society, 2001, p. 405 – 410.

KIRSTE T. Smart environments and self-organizing appliance ensembles. In: AARTS E, ENCARNAÇÃO J. L.: **True Visions**, Germany: Springer, 2005.

KLUS, H.; RAUSCH, A. A general architecture for self-adaptive AmI components applied in speech recognition. In: INTERNACIONAL WORKSHOP ON SELF-ADAPTATION AND

SELF-MANAGING SYSTEMS - SEAMS, 2006, Shanghai. **Proceedings...** New York: ACM Press, 2006, p. 72 - 78.

LEE, C.; HELAL, S.; NORDSTEDT, D. The µJini proxy architecture for impromptu mobile services. In: INTERNATIONAL SYMPOSIUM ON APPLICATIONS AND THE INTERNET WORKSHOPS, 2006, Phoenix. **Proceedings…** New York: IEEE Computer Society, 2006, p. 23 – 27.

LIU, R.; YANG, H.; PAN, W. An evolutionary system development approach in a pervasive computing environment. In: INTERNATIONAL CONFERENCE ON CYBERWORLDS, 2004, Tokyo. **Proceedings…** New York: IEEE Computer Society, 2004, p. 194 - 199.

MAGICDRAW. **UML modeling and design tool**: architecture made simple. Disponível em:<http://www.magicdraw.com/>. Acesso em: 11 nov. 2010.

MANN, W.; HELAL, S. Smartphones for the elders: boosting the intelligence of smart homes. In: WORKSHOP AUTOMATION AS CAREGIVER: THE ROLE OF INTELLIGENT TECHNOLOGY IN ELDER CARE, 2002, Edmonton: **Proceedings…** Menlo Park: AAAI Press, 2002, p. 74 - 79.

MULTIMEDIA HOME PLATFORM (MHP). **Official website for DVB-MHP and DVB-GEM**: open middleware for interactive TV. Disponível em: <http://http://www.mhp.org>. Acesso em: 20 set. 2010.

NAZARI, A. A. S.; KLAR, F.; KIRSTE, T. **3DSim**: rapid prototyping ambient intelligence. Disponível em: < http://www.igd.fhg.de/igd-a1/projects/amilab/index.html>. Acesso em: 15 jul. 2008.

OLIVEIRA, M. Implementing home care application in brazilian digital TV. In: **Information Infrastructure Symposium**, 2009. GIIS '09, June 2009, Hammemet, p. 1 – 7.

OBJECT MANAGEMENT GROUP (OMG). Disponível em: <http://www.omg.org>. Acesso em: 09 nov. 2010.

OPEN SERVICE GATEWAY INITIATIVE (OSGi). **OSGi Alliance.** Disponível em: <http://www.osgi.org/Main/HomePage>. Acesso em: 14 jan. 2011.

O'REILLY. **Introduction to X10 home automation technology**. Disponível em: <http://www.oreillynet.com/pub/a/network/2005/01/10/x10_hmhck.html>. Acesso em: 28 nov. 2008.

OTSUKA, Y.; SHIMIZU, N.; YAGIU, R. Home Network Remote Controller Using The appliance integrated script engine. In: INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS, 2006, Las Vegas: **Proceedings…** New York: IEEE, 2006, p. 249 - 250.

PARK, S. K.; KIM, D. C.; KANG, B. S. Context-aware middleware architecture for intelligent service in mobile environment. In: INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY, 2006, Bhubaneswar. **Proceedings…** New York: IEEE Computer Society, 2006, p. 240 – 244.

PENDER, T. **UML a bíblia.** Rio de Janeiro: Elsevier, 2004. 711 p.

PEROZZO, R. F.; PEREIRA, C. E. Management of services in intelligent environments for mobile devices. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS, 2008, Seattle. **Proceedings…** Hertfordshire: Institution of Engineering and Technology, 2008, v. 1. p. 1 - 6.

PEROZZO, R. F. et al. Inteligência premiada. **Jornal Zero Hora**. Porto Alegre, v. 45, n. 15668, p. 8, 23 jul. 2008a.

_____. Eletrodomésticos high-tech. **Correio do Povo**. Porto Alegre, v. 113, n. 289, p. 6, 15 jul. 2008b.

PETA5. **Aplicações para TV digital interativa**. Disponível em: <http://www.peta5.com.br>. Acesso em: 04 out. 2010.

PETRIU, D. C; WOODSIDE, C. M. Performance analyses with UML. In: **UML for Real**: design of embedded real-time systems. Boston: Kluwer Academic Publishers, 2004. p.220-239.

PHILIPS RESEARCH. **What is ambient intelligence**. Disponível em <http://www.research.philips.com/technologies/syst_softw/ami/index.html> Acesso em: 28 out. 2008.

RBS TV. **Chuviscos são coisas do passado**. Disponível em: <www.rbstvdigital.com.br>. Acesso em: 20 set. 2010.

RCASOFT. **Soluções em TV digital com qualidade**. Disponivel em: <http://www.rcasoft.com.br/midd_recep.php>. Acesso em: 23 nov. 2010.

REBOUÇAS, F. **PLC internet pela rede elétrica**. Disponível em: <http://www.infoescola.com/internet/plc-power-line-communications/>. Acesso em: 12 jan. 2011.

RECORD. **Site oficial da TV Record**. Disponível em: <http://rederecord.r7.com/>. Acesso em: 27 set. 2010.

REDE BANDEIRANTES DE TELEVISÃO (BAND**). Portal de notícias do Grupo Bandeirantes**. Disponível em: <http://www.band.com.br/>. Acesso em: 27 set. 2010.

REDONDO, R. P. D. et al. Enhancing residential gateways: OSGi services composition. In: INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS, 2007, Las Vegas: **Proceedings…** New York: IEEE, 2007, p. 1 – 2.

RESOLVE. **Serviços de EAD para TV digital e internet**. Disponível em: <http://www.resolveinfo.com.br/pages/servicos/tvdigital.htm>. Acesso em: 27 set. 2010.

RÖCKER, C.; HINSKE, S.; MAGERKURTH, C. SPIROS: a system for privacy enhanced information representation in smart home environments. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS, 2006, Athens: **Proceedings…** New York: IEEE, 2006, p. 267 – 274.

ROGERS, G. F. **Framework-based software development in C++**, New Jersey: Prentice-Hall, 1997. 382 p.

SISTEMA BRASILEIRO DE TELEVISÃO (SBT). Disponível em: <http://www.sbt.com.br/>. Acesso em: 27 set. 2010.

SHALLOWAY, A.; TROTT, J. R. **Explicando padrões de projeto**: uma nova perspectiva em projeto orientado a objeto. Porto Alegre: Bookman, 2004. 328 p.

SHI, Y. et al. The smart classroom: merging technologies for seamless tele-education. In: **IEEE Pervasive Computing**, New York, v. 2, n. 2, p. 47 - 55, Apr./June 2003.

SILVA, R. A.; GONÇALVES, P. A. **Um estudo sobre a aplicação da arquitetura orientada a serviços em casas inteligentes**. Universidade Federal de Pernambuco. Disponível em: <http://www.unibratec.com.br/jornadacientifica/diretorio/UFPERAS.pdf>. Acesso em: 17 nov. 2010.

SIMIONI, A., ROESLER, V. Um framework para o desenvolvimento de aplicações interativas para a Televisão Digital. In: **ERRC** - Escola Regional de Redes de Computadores, Passo Fundo, v. 1. p. 10-16, 2006.

SOARES, L. F. G. TV interativa se faz com Ginga. **Revista da Sociedade Brasileira de Engenharia de Televisão**, São Paulo. n. 105, p. 30-35, 2008.

SOUZA, J. P.; GARLAN, D. Aura: an architectural framework for user mobility in ubiquitous computing environments. In: WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE: SYSTEM DESIGN, DEVELOPMENT AND MAINTENANCE, 2002, Montreal. **Proceedings…** Norwell: IEEE/IFIP, 2002, p. 29 - 43.

STAVROULAKI, V. et al. Distributed web-based management framework for ambient reconfigurable services in the intelligent environment. **Mobile Networks and Applications**. Hingham: ACM-Springer, 2006, p. 889 – 900.

STICKERCENTER. **Site oficial do repositório de aplicativos sticker center**. Disponível em: <https://www.stickercenter.com.br/StickerWeb/pt_BR/index.html>. Acesso em: 27 set. 2010.

SUNSPOT. **Sun Spot World**. Disponível em: <http://www.sunspotworld.com/>. Acesso em: 29 set. 2010.

TELECO. **Implantação TV digital no mundo**. Disponível em: <http://www.teleco.com.br/tvdigital_mundo.asp>. Acesso em: 20 set. 2010.

TERRA TECNOLOGIA. **TV conectada da LG traz conteúdo do Terra integrado**. Disponível em:<http://tecnologia.terra.com.br/noticias/0,,OI4415362-EI12882,00-TV+conectada+da+LG+traz+conteudo+do+Terra+integrado.html>. Acesso em: 08 nov. 2010.

TQTVD. **ByYou digital**: serviços e soluções para tv digital interativa. Disponível em: <http://www.tqtvd.com.br/br/index.html>. Acesso em: 27 set. 2010.

TSOURAKIS, N. et al. An architecture for multimodal applications over wireless data networks. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS, 2006, Athens: **Proceedings…** New York: IEEE, 2006, p. 221 – 227.

TV GLOBO. **Tudo sobre a TV digital em alta definição**. 2008. Disponível em <http://www.tvglobodigital.com>. Acesso em: 20 set. 2010.

VIANA, N. S. et al. A convergence proposal between the brazilian middleware for iDTV and home network platforms. In: IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'09), Las Vegas. **Proceedings…** New York: IEEE, 2009, p. 1-5.

WORLD WIDE WEB CONSORTIUM (W3C). **REC-xhtml1-20020801**. XHTML 1.0 the txtensible hypertext markup language, Cambridge, 2002.

WEISER, M. The computer for the 21st century. **Scientific American**, New York, v. 265, n. 3, p. 94 - 104, Sept. 1991.

XU, W.; XIN, Y.; LU, G. A system architecture for pervasive computing. In: INTERNACIONAL CONFERENCE ON NATURAL COMPUTATION, 2007, Haikou. **Proceedings…** New York: IEEE Computer Society, 2007, p. 772 - 776.

ZIEGLER, M. et al. Secure Profile Management in Smart Home Networks. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 2005, Copenhagen: **Proceedings…** New York: IEEE Computer Society, 2005, p. 209 – 213.

ZIGBEE ALLIANCE. **ZigBee is the global wireless language connecting dramatically differente devices to work together and enhance everyday life**. Disponível em: <http://www.zigbee.org>. Acesso em: 10 ago. 2010.