JOÃO PAULO DELGADO PRETI

**Arquitetura Orientada a Serviço P2P dirigida a eventos para realização de
transições *ad hoc* em Interfaces de Usuário Distribuídas**

São Paulo

2014

JOÃO PAULO DELGADO PRETI

**Arquitetura Orientada a Serviço P2P dirigida a eventos para realização de
transições *ad hoc* em Interfaces de Usuário Distribuídas**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências.

Orientadora: Prof^a. Dra. Lucia Vilela Leite
Filgueiras

São Paulo
2014

JOÃO PAULO DELGADO PRETI

**Arquitetura Orientada a Serviço P2P dirigida a eventos para realização de
transições *ad hoc* em Interfaces de Usuário Distribuídas**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Ciências.

Área de Concentração: Sistemas Digitais

Orientadora: Prof^a. Dra. Lucia Vilela Leite
Filgueiras

São Paulo

2014

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 12 de junho de 2014.

Assinatura do autor _____

Assinatura do orientador _____

Catologação-na-publicação

Preti, João Paulo Delgado

Arquitetura orientada a serviço P2P dirigida a eventos para realização de transições ad hoc em interfaces de usuário distribuídas / J.P.D. Preti. -- versão corr. -- São Paulo, 2014.

182 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

**1.Arquitetura orientada a serviços 2.Interface multimídia
3.Interação homem-máquina I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.**

*Dedico este trabalho a minha
amada esposa Maruza.*

AGRADECIMENTOS

O Doutorado é uma jornada, portanto várias pessoas estiveram presentes, ajudando das mais variadas formas, contribuindo presencialmente ou a distância, motivando e se alegrando a cada passo conquistado. De alguma forma, tenho de organizar as menções, por isso uns virão à frente de outros; no entanto, agradeço a todos com o mesmo sentimento.

Agradeço a minha família, pai, madrasta, irmãos e irmãs, de sangue e de coração, pela motivação e por proporcionar momentos de alegria e descontração. Agradeço a Maruza, minha amada esposa, por sempre se preocupar comigo e estar ao meu lado em todos os momentos dessa jornada. A minha mãe, que mesmo não estando mais entre nós, nos acompanha e protege.

Agradeço ao criador, pela vida, pela presença de cada dia, por acalmar meu coração em momentos de angústia e por me ajudar nessa jornada.

Um agradecimento especial à minha orientadora Lucia Filgueiras, que contribuiu com questões teóricas e práticas, em São Paulo e em Cuiabá, nos almoços e caminhadas. Obrigado pelas revisões minuciosas do texto e pela dedicação ao trabalho. Obrigado pela sua amizade.

Agradeço aos coordenadores do DINTER que sempre me apoiaram na realização deste doutorado. Aos professores da USP que se deslocaram até Cuiabá e apoiaram este trabalho por meio das disciplinas ministradas. Às organizações USP, IFMT, UFMT, CAPES e FAPEMAT por tornarem possível esse sonho e me apoiarem nas atividades do doutorado.

Aos membros da banca examinadora pela disposição e por toda contribuição que certamente deixarão para a continuidade da pesquisa.

Finalmente, um obrigado aos amigos e colegas do IFMT e da UFMT que sempre apoiaram a realização deste trabalho.

Deixo meu sincero agradecimento a todos e um pedido de desculpa a aqueles que não mencionei. A jornada é longa e demasiados seriam os nomes a compor este agradecimento em palavras, que não é capaz de demonstrar o real sentimento de gratidão que tenho com todos. A todos, um grande abraço.

RESUMO

O número de dispositivos conectados e presentes na vida dos usuários é expressivo e a realização de atividades utilizando múltiplos dispositivos já é uma realidade. Entretanto, o uso integrado e sinérgico desses recursos, a que se denomina interação distribuída, tem encontrado empecilhos nas limitações dos sistemas em prover e atender interações dos usuários que atravessem as fronteiras do dispositivo. A diversidade de plataformas, bem como das áreas de pesquisa envolvidas são também forças contrárias à interação distribuída. O objetivo desta pesquisa, portanto, está em definir uma arquitetura que facilite a construção de aplicações observando aspectos de uma interação distribuída, na qual usuário e dispositivo não se acoplem unicamente no modelo de interação 1 usuário:1 dispositivo. Definiu-se uma arquitetura orientada a serviço, após pesquisa exploratória e sistemática acerca de *frameworks* e *middlewares* que promovem a integração entre dispositivos, e verificou-se sua viabilidade por provas de conceito. Como resultados, obteve-se um padrão de transições necessários para a construção de diversos cenários de uso e uma arquitetura orientada a serviços P2P dirigida a eventos que permite a implementação e execução desses cenários.

Palavras-Chave: Arquitetura Orientada a Serviços. Interfaces de Usuário Distribuídas. Mídias Cruzadas. Transição da Interação.

ABSTRACT

The number of connected devices and present in the lives of users is currently expressive and performing activities using multiple devices is already a reality. However, the integrated and synergistic use of these resources, which is called distributed interaction, has found obstacles in the limitations of current systems to provide and attend user interactions that cross the boundaries of the device. A variety of platforms, also involved research areas are contrary forces to the distributed interaction too. The aim of this research is therefore to define an architecture that facilitates the building of applications observing aspects of a distributed interaction, where user and device does not only bind in the interaction model 1 user:1 device. A service-oriented architecture has been defined, after exploratory and systematic research about frameworks and middlewares that promote integration among devices, and its feasibility has been verified by proofs of concept. As a result, we have obtained a pattern of transitions necessary for the construction of various usage scenarios and a P2P event-driven service-oriented architecture that allows implementation and execution of these scenarios.

Keywords: Service Oriented Architecture. Distributed User Interfaces. Crossmedia. Interaction Transition.

LISTA DE ILUSTRAÇÕES

FIGURA 1 - ARQUITETURA DO <i>FRAMEWORK</i> AXMEDIS.....	36
FIGURA 2 - MODELO RSL QUE PERMITE RELACIONAR ELEMENTOS DE MÍDIA AGREGANDO NOVOS METADADOS A RELAÇÃO	38
FIGURA 3 - ARQUITETURA DO <i>FRAMEWORK</i> X-GOV.....	40
FIGURA 4 - PASSOS PARA A REALIZAÇÃO DA MIGRAÇÃO DA INTERFACE GRÁFICA.....	43
FIGURA 5 – ÁREAS DE CONCENTRAÇÃO DA PESQUISA EM DUI NOS 3 <i>WORKSHOPS</i> E RESPECTIVOS PESQUISADORES	58
FIGURA 6 – MAPA MENTAL DE 16 ESTUDOS DE CASO EM DUI AGRUPADOS POR ÁREA DE APLICAÇÃO E RESPECTIVOS PESQUISADORES	59
FIGURA 7 – FERRAMENTAS QUE AUXILIAM NO DESENVOLVIMENTO DE APLICAÇÕES DE INTERAÇÃO DISTRIBUÍDA.	62
FIGURA 8 – MODELO DE COMUNICAÇÃO INDIRETA	81
FIGURA 9 – MODELO DE COMUNICAÇÃO DIRETA	81
FIGURA 10 – PILHA DE PROTOCOLOS DA ESPECIFICAÇÃO DPWS.	84
FIGURA 11 – ARQUITETURA DA ESPECIFICAÇÃO <i>WS-EVENTING</i> E SUAS OPERAÇÕES	87
FIGURA 12 – SERVIÇOS QUE DEVEM SER COMPOSTOS SOBRE A PILHA DPWS PARA OCORRÊNCIA DA TRANSIÇÃO DE DISTRIBUIÇÃO DA INTERAÇÃO	91
FIGURA 13 – SERVIÇOS QUE FORNECEM ACESSO AOS RECURSOS DO DISPOSITIVO	92
FIGURA 14 – SERVIÇOS DE RECURSOS DE INTERAÇÃO	92
FIGURA 15 – ESTRUTURA COMUM A TODOS OS SERVIÇOS QUE ESTÃO SOB A PILHA DPWS	93
FIGURA 16 - DIAGRAMA DE ESTADO PARA COMPONENTES DE INTERFACE GRÁFICA QUE SUPORTAM AS TRANSIÇÕES DE INTERAÇÃO DO QUADRO 7	97
FIGURA 17 – REPRESENTAÇÃO DO SERVIÇO PARA O SUPORTE DE TRANSIÇÕES DE INTERAÇÃO	98
FIGURA 18 – DIAGRAMA DE SEQUÊNCIA MOSTRANDO UM EXEMPLO DE USO DO SERVIÇO DE TRANSIÇÃO DA INTERAÇÃO	99
FIGURA 19 – MODELO MVC PARA TRANSIÇÕES DE INTERAÇÃO EM COMPONENTES GUI	100
FIGURA 20 – CAIXA DE TEXTO RECEBENDO INTERAÇÕES NO FORMATO <i>N:MLD</i>	102
FIGURA 21 – LISTA DE OPÇÕES RECEBENDO INTERAÇÕES NO FORMATO <i>N:MFD</i>	103
FIGURA 22 – MODELO <i>WS-DISCOVERY</i> DE MENSAGENS PARA DESCOBERTA DE DISPOSITIVOS EM REDE	104
FIGURA 23 – CLASSE <i>DPWSCClient</i> PARA LOCALIZAÇÃO DE DISPOSITIVOS NA REDE	105
FIGURA 24 – ARQUITETURA DIRIGIDA A REQUISIÇÃO	106
FIGURA 25 – ARQUITETURA DIRIGIDA A EVENTOS	106
FIGURA 26 – PUSH VERSUS PULL.....	107
FIGURA 27 – ARQUITETURA DO <i>FRAMEWORK</i> JMEDS	110
FIGURA 28 – PILHA DE SERVIÇOS DA ARQUITETURA IMPLEMENTADA E RESPECTIVAS APLICAÇÕES (<i>XDevice</i> <i>API</i> , <i>XDevice Android</i> e <i>XDevice Desktop</i>)	111
FIGURA 29 – ESTRUTURA DE PACOTES DO APLICATIVO <i>XDevice</i>	112
FIGURA 30 – ORGANIZAÇÃO DOS SERVIÇOS DE RECURSOS E DE INTERAÇÃO DO DISPOSITIVO <i>Android</i>	114
FIGURA 31 – RECURSOS NECESSÁRIOS NO DISPOSITIVO <i>Android</i> PARA EXECUÇÃO DE TODOS OS SERVIÇOS IMPLEMENTADOS	115
FIGURA 32 – INTERFACE GRÁFICA DO APLICATIVO <i>Android XDevice</i>	116
FIGURA 33 – <i>Alice_{PC}</i> TRANSFERINDO A INTERAÇÃO DE UM COMPONENTE DE TEXTO PARA <i>Bob_{SP}</i>	117
FIGURA 34 – <i>XSchema</i> CONTENDO INFORMAÇÕES DOS DISPOSITIVOS DISPONÍVEIS NA REDE	118
FIGURA 35 – OPÇÕES DA CAIXA DE TEXTO PARA A ENTRADA REMOTA.....	119
FIGURA 36 – NOTIFICAÇÃO DE TRANSIÇÃO <i>CROSS-DEVICE</i>	119
FIGURA 37 – RESPOSTA PARA A REQUISIÇÃO <i>CROSS-DEVICE</i>	120
FIGURA 38 – COMPONENTE JSF PARA LISTAR DISPOSITIVOS, SERVIÇOS E OPERAÇÕES.....	121
FIGURA 39 – COMPONENTE JSF DE MAPA PARA EXIBIR A LOCALIZAÇÃO DISPOSITIVO DPWS	121
FIGURA 40 – MODELO MVC DO <i>JavaFX</i>	123
FIGURA 41 – CAMADA DE CONTROLE PARA OS 4 COMPONENTES	124
FIGURA 42 – CAMADA <i>Skin</i> PARA OS 3 COMPONENTES	124
FIGURA 43 – <i>View</i> PADRÃO PARA TODOS OS COMPONENTES	125
FIGURA 44 – COMPONENTE <i>XTextField</i>	125
FIGURA 45 – COMPONENTE <i>XRadioButton</i>	126

FIGURA 46 – COMPONENTE XHYPERLINK.....	126
FIGURA 47 – CAMADA XBEHAVIOR PARA OS 3 COMPONENTES.....	127
FIGURA 48 – DIAGRAMA DE SEQUÊNCIA SOBRE O COMPORTAMENTO DE XTEXTFIELD.....	128
FIGURA 49 – INTERFACE DO COMPONENTE DE <i>MOUSE</i> REMOTO.....	129
FIGURA 50 – TRANSIÇÃO DE INTERAÇÃO DO COMPONENTE XMOUSE.....	130
FIGURA 51 – COMPONENTE XAUDIO.....	132
FIGURA 52 – NOTIFICAÇÃO DE CAPTURA DO ÁUDIO.....	132
FIGURA 53 – SERVIÇOS QUE COMPÕEM O DISS.....	135
FIGURA 54 – EXEMPLO DO PAPEL EXECUTADO PELO IMMS.....	142
FIGURA 55 – MODELO DE CLASSES DO IMMS.....	143
FIGURA 56 – DOCUMENTO XML CONTENDO MAPEAMENTO DAS MODALIDADES DE INTERAÇÃO E SERVIÇOS..	145
FIGURA 57 – DIAGRAMA DE SEQUÊNCIA MOSTRANDO O USO DO IMMS POR EVENTOS.....	146
FIGURA 58 – DIAGRAMA DE SEQUÊNCIA MOSTRANDO O USO DO IMMS POR <i>STREAM</i>	147
FIGURA 59 – MODELO DE CLASSES DO DMS E DO UMS.....	148
FIGURA 60 – DIAGRAMA DE SEQUÊNCIA PARA ADIÇÃO DE NOVOS DISPOSITIVOS DESCOBERTOS.....	149
FIGURA 61 – DUAS APLICAÇÕES COM SEUS RESPECTIVOS COORDENADORES.....	152

LISTA DE QUADROS

QUADRO 1 – SUPORTE DAS FERRAMENTAS ÀS CARACTERÍSTICAS DE MÍDIA CRUZADA E MODELO DE ARQUITETURA.....	47
QUADRO 2 – FERRAMENTAS QUE PROMOVEM ALGUMA FORMA DE INTERAÇÃO DISTRIBUÍDA CLASSIFICADAS DE ACORDO COM A OBSERVAÇÃO DE (BLUMENDORF, ROSCHER E ALBAYRAK, 2011)	63
QUADRO 3 – REQUISITOS DE UM <i>FRAMEWORK</i> DE TRANSIÇÕES ENTRE MÍDIAS	66
QUADRO 4 – REQUISITOS COMPLEMENTARES AOS DO QUADRO 3.....	68
QUADRO 5 – FERRAMENTAS QUE PROMOVEM ALGUMA FORMA DE INTERAÇÃO DISTRIBUÍDA, BASES TECNOLÓGICAS UTILIZADAS, ARQUITETURA E CONTEXTO DE APLICAÇÃO	75
QUADRO 6 - COMPARAÇÃO DE TECNOLOGIAS PARA INTEGRAÇÃO DE DISPOSITIVOS.....	82
QUADRO 7 – CENÁRIOS DE INTERAÇÃO DISTRIBUÍDA (DIÁLOGOS SOLICITAÇÃO (S) / RESPOSTA (R)).....	94
QUADRO 8 – FERRAMENTAS DE APOIO AO DESENVOLVIMENTO DAS PROVAS DE CONCEITO	108
QUADRO 9 – INTERAÇÃO ENTRE <i>SMARTPHONE</i> E TV	142
QUADRO 10 – 14 OBSTÁCULOS E OPORTUNIDADES DE PESQUISA	157

LISTA DE ABREVIATURAS E SIGLAS

3GPP	3rd Generation Partnership Project
B2B	Business to Business
C4A	Content for All
CC/PP	Composite Capabilities/Preference Profiles
CLDC	Connected Limited Device Configuration
CMS	Content Management System
CCS	Content Conversion Service
CSS	Context Support Service
DaaS	Device as a Service
DDD	Data Description Framework
DISS	Distributed Interaction Support Service
DLNA	Digital Living Network Alliance
DMS	Device Management Service
DNS	Domain Name System
DPWS	Devices Profile for Web Services
DRM	Digital Rights Management
DUI	Distributed User Interface
EPL	Eclipse Public License
ESB	Enterprise Service Bus
GPS	Global Position System
GUI	Graphics User Interface
GUP	Generic User Profile
HS	Home Server
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
IHC	Interação Humano-Computador
IMMS	Interaction Modality Mapping Service
IP	Internet Protocol
JAAS	Java Authentication and Authorization Service
JavaME	Java Micro Edition
JMEDS	Java Multi Edition DPWS Stack

MARIA	Model-based IAnguage foR Interactive Applications
OASIS	Organization for the Advancement of Structured Information Standards
OMS	Object Management System
OSGi	Open Service Gateway Initiative Alliance
P2P	Peer to Peer
PaaS	Platform as a Service
PC	Personal Computer
PCP	Personal Cloud Platform
PDA	Personal Digital Assistant
QoS	Quality of Service
RDF	Resource Description Framework
RFID	Radio Frequency Identification
RSL	Resource-Selector-Link
SaaS	Software as a Service
SEI	Software Engineering Institute
SFE	Service Front-End
SIP	Session Initiation Protocol
SLA	Security Level Agreement
SMS	Short Message Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
STB	Set-Top Box
TIC	Tecnologia de Informação e Comunicação
TSS	Transition Support Service
TVDi	TV Digital Interativa
UAProf	User Agent Profile
UDP	User Datagram Protocol
UI	User Interface
UIDL	User Interface Description Language
UML	Unified Modeling Language
UMS	User Management Service
UPnP	Universal Plug and Play
XHTML	eXtensible HyperText Markup Language

XML	eXtensible Markup Language
XTP	XWeb Transport Protocol
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WIMP	Window, Icon, Menu, Pointing device
WS	Web Service
WS4D	Web Service for Devices
WSDL	Web Service Description Language
WSRP	Web Services for Remote Portlets
WURFL	Wireless Universal Resource File
WWW	World Wide Web

SUMÁRIO

1	INTRODUÇÃO	17
1.1	MOTIVAÇÃO	17
1.2	OBJETIVOS DO TRABALHO	22
1.3	CONTRIBUIÇÃO	23
1.4	METODOLOGIA DE PESQUISA	23
1.5	ORGANIZAÇÃO DO TRABALHO	26
2	INTERAÇÃO DISTRIBUÍDA.....	28
2.1	MÍDIA CRUZADA, CROSSMEDIA, TRANSMEDIA	28
2.1.1	<i>Conceitos</i>	29
2.1.2	<i>O que se distribui.....</i>	31
2.1.3	<i>Suporte Tecnológico para Mídias Cruzadas</i>	34
2.1.4	<i>Capacidade de suporte a corroboração, complementação e concomitância nas arquiteturas estudadas</i>	47
2.2	COMPUTAÇÃO UBÍQUA.....	48
2.2.1	<i>Conceitos</i>	49
2.2.2	<i>O que se distribui.....</i>	50
2.2.3	<i>Como se faz?.....</i>	50
2.3	DISTRIBUTED USER INTERFACES.....	54
2.4	CONSIDERAÇÕES FINAIS DO CAPÍTULO	59
3	ARQUITETURAS PARA DUI	61
3.1	EXEMPLOS DE ARQUITETURAS PARA INTERAÇÃO DISTRIBUÍDA	61
3.2	REQUISITOS PARA UMA ARQUITETURA PARA INTERAÇÃO DISTRIBUÍDA	65
3.3	AS TRANSIÇÕES NA INTERAÇÃO DISTRIBUÍDA	72
3.4	SUPORTE TECNOLÓGICO	73
3.5	DEFINIÇÃO DA ARQUITETURA PROPOSTA	77
3.5.1	<i>A escolha de SOA</i>	77
3.5.2	<i>Modelo de comunicação.....</i>	80
3.5.3	<i>DPWS</i>	82
3.5.4	<i>WS-Discovery (WS-DISCOVERY, 2009).....</i>	85
3.5.5	<i>WS-Addressing (WS-ADDRESSING, 2004)</i>	86
3.5.6	<i>WS-MetadataExchange (WS-METADATAEXCHANGE, 2011).....</i>	86
3.5.7	<i>WS-Eventing (WS-EVENTING, 2006).....</i>	87
3.6	CONSIDERAÇÕES FINAIS DO CAPÍTULO	88
4	UM MODELO DE SERVIÇOS PARA TRANSIÇÕES DE DISTRIBUIÇÃO DA INTERAÇÃO (TSS)	89
4.1	ARQUITETURA DO SERVIÇO DE APOIO A TRANSIÇÃO (TSS)	90
4.2	DESCOBERTA DE DISPOSITIVOS NA REDE	103
5	IMPLEMENTAÇÃO DO TSS	106
5.1	TROCA DE MENSAGENS	106
5.2	PROVAS DE CONCEITO	107
5.3	AMBIENTE DE DESENVOLVIMENTO	108
5.4	CONTEXTO ANDROID	111
5.5	EXPERIMENTOS DE COMUNICAÇÃO NO CONTEXTO <i>WEB</i>	116
5.5.1	<i>DISCUSSÃO DAS PROVAS DE CONCEITO.....</i>	120
5.6	CONTEXTO <i>DESKTOP</i>	122
5.7	CONSIDERAÇÕES FINAIS DO CAPÍTULO	132

6	UMA ARQUITETURA MAIS AMPLA	134
6.1.1	<i>Serviço de mapeamento de modalidade de interação (IMMS)</i>	<i>141</i>
6.1.2	<i>Serviço de Gerenciamento de usuários e dispositivos (UMS e DMS).....</i>	<i>147</i>
6.1.3	<i>Context Support Service (CSS).....</i>	<i>149</i>
6.1.4	<i>DISS Application Program Interface (API)</i>	<i>152</i>
7	REFLEXÕES SOBRE DUI.....	154
8	CONCLUSÕES	159
8.1	OBJETIVOS E CONTRIBUIÇÕES.....	159
8.2	TRABALHOS FUTUROS.....	160
8.3	CONCLUSÃO	161
	REFERÊNCIAS	163
	APÊNDICE A – PROTOCOLO DE REVISÃO SISTEMÁTICA	173
	APÊNDICE B – ARTIGO SUBMETIDO AO DUI 2011.....	176

1 INTRODUÇÃO

Esta pesquisa insere-se na área de interação distribuída, na qual o ambiente de interação é complexo, composto por múltiplos dispositivos, múltiplos usuários e múltiplas intenções. O foco desta pesquisa são as arquiteturas que atuam como infraestrutura desses ambientes.

Nesta introdução, mostra-se a importância desses ambientes para a evolução da Interação Humano Computador (IHC), definem-se os objetivos da pesquisa e delinea-se a metodologia de trabalho.

1.1 MOTIVAÇÃO

Com o advento das redes, a computação pessoal evoluiu para a computação distribuída e a partir do momento em que os computadores tornaram-se conectados, eles começaram a compartilhar recursos através da rede. A computação distribuída marcou o passo seguinte em direção à computação ubíqua, abrindo o acesso contínuo a recursos de informação e comunicação remota com tolerância a falhas, alta disponibilidade e segurança.

Embora a *World Wide Web* (WWW) não tenha sido projetada para ser uma infraestrutura de computação distribuída, sua onipresença tornou-a uma opção atraente para a experimentação de conceitos de computação distribuída. A natureza *ad hoc* de crescimento da *Web* provou que é possível distribuir recursos de computação em grande escala. Os mecanismos simples para associação dos recursos proporcionaram um meio para integrar bases de informações distribuídas em uma única estrutura. Mais importante, a *Web* foi pioneira na criação de uma infraestrutura de informação e comunicação quase onipresente.

Apesar de a *Web* não permear o mundo real de entidades físicas, é no entanto um potencial ponto de partida para a computação ubíqua. A computação ubíqua, também chamada de computação pervasiva, representa a visão de um ambiente repleto de capacidades de computação e comunicação, integrado naturalmente com usuários humanos (SATYANARAYANAN, 2001).

As tecnologias de informação e comunicação se tornaram elementos essenciais para a realização da maioria das tarefas da vida moderna. Pessoas precisam lidar com diferentes dispositivos eletrônicos para obter dados e acessar

diferentes serviços. A área de IHC (Interação Humano-Computador) pode não ser mais sobre como interagimos com dispositivos específicos. Conforme o ambiente ao redor das pessoas se torna "o computador", a área de IHC pode se tornar também uma área sobre como as pessoas interagem com múltiplos computadores e sistemas. Ao invés de competir, aplicações estão sendo tecidas em conjuntos, tornando a interação do usuário mais rica e complexa.

Dispositivos interativos podem permitir que a interação ocorra de mais de uma forma, com flexibilidade, o que se convencionou chamar de interação multimodal. Um celular, por exemplo, pode permitir interação por comando de voz, teclado, tela sensível ao toque, sensores de movimento, dependendo dos recursos disponíveis no mesmo. Esses recursos interativos começam a ser estendidos para interagir com outros dispositivos. Por exemplo, pode-se utilizar o recurso de multitoc de um celular para aumentar ou diminuir o volume da TV e o sensor de movimento para realizar a troca de canal.

Para essas **pontes** que permitem que o fluxo de interação migre de um dispositivo para o outro adotou-se o termo de **transição**.

Em um trabalho anterior do nosso grupo de pesquisa, foram identificadas características das transições por meio de uma análise de 25 aplicações de mercado e acadêmicas, o que inclui filmes, séries de TV, jogos, campanhas de aviso, revistas, jornais, iTV e aplicações móveis e Web (MIYAMARU et al., 2008). Nesse trabalho constatou-se que do ponto de vista tecnológico, diversas plataformas são necessárias para apoiar as transições. Apesar de a convergência tecnológica ter simplificado a tarefa de engenharia, diferentes protocolos e serviços ainda são necessários para integrar diversos tipos de transição em uma aplicação.

Desenvolver aplicações que permitam interagir com diferentes dispositivos, em diferentes plataformas, requer um conhecimento do funcionamento de cada plataforma, das APIs de cada uma e dos protocolos utilizados. Além disso, cada plataforma pode apresentar restrições que incompatibilizam o uso de algum recurso de forma integrada. Dispositivos Apple e seus aplicativos, bem como dispositivos Android e respectivos aplicativos, se integram bem enquanto a comunicação se restringe a mesma plataforma. Quando se faz necessário a integração entre um dispositivo como AppleTV e um *smartphone* Android, as diferenças entre protocolos de comunicação e APIs disponibilizadas torna essa

atividade onerosa e desgastante cognitivamente, além de que pode ser impossibilitada por restrições, como a necessidade de ser um autêntico usuário de produtos Apple. As habilidades necessárias para realizar a integração entre dispositivos de diferentes plataformas de forma eficiente, pode apresentar uma barreira em utilizar o potencial dos sistemas de informação conectados e os vários objetos disseminados no meio.

Dada a diversidade de dispositivos presentes na vida do usuário, o conceito de computador pessoal (PC) se mostra insuficiente, visto que, o computador que ele utiliza para acessar a nuvem tornou-se irrelevante. Ele pode realizar suas atividades por meio do *desktop*, *notebook*, *tablet* ou *smartphone*. Por meio da computação em nuvem, dados e até mesmo recursos de processamento estão sendo transferidos dos dispositivos do usuário para a nuvem. O dispositivo do usuário está se tornando apenas um recurso interativo, um nó da rede a ser utilizado para capturar as ações do usuário.

Além disso, o uso dos dispositivos computacionais extrapola a dimensão de um único usuário. Dispositivos estão sendo compartilhados; um *tablet* pode ser utilizado por vários membros de uma família ou até mesmo por diversos funcionários em uma empresa. TVs já possuem tecnologia que permite que o mesmo aparelho seja utilizado por duas pessoas para assistirem diferentes canais (como o Samsung MultiView®).

A nova geração de vídeo *games* lançada no final de 2013 já permite, por exemplo, que mais de um jogador no mesmo console seja identificado simultaneamente, de forma que jogos cooperativos possam pontuar e registrar ganhos na conta respectiva de cada jogador e que esses mesmos jogadores possam aproveitar a evolução do jogo em seus respectivos consoles ao retornar para casa. Esses mesmos consoles já permitem que o jogador continue o jogo, sem interrupção, da TV para um dispositivo portátil (SONY, 2013).

Apesar dos esforços para que dispositivos se integrem e suportem múltiplos usuários, vários desafios são encontrados. Por exemplo, os sistemas operacionais (SO) para os consoles de vídeo *game* estão sendo projetados e atualizados para refletir a realidade atual de múltiplos usuários e múltiplos dispositivos, mas são fortemente acoplados e limitam essa capacidade aos dispositivos específicos de cada empresa.

Nos *tablets* Android também é possível registrar mais de um usuário, mas não para uso concorrente, de forma nativa suportada pelo sistema operacional. Quando outro usuário deseja utilizar o dispositivo, faz-se necessário que efetue o *logoff* do usuário anterior e realize um novo *login*.

Sistemas operacionais para *desktop*, por exemplo, não permitem que mais de um cursor de *mouse* possa ser ativado e identificado por usuários distintos. Sistemas operacionais como MAC OS, Linux e Windows, em suas versões mais recentes do ano de 2013, permitem a identificação de diversos dispositivos de entrada de dados, como *mouses*, por exemplo, mas todos os *mouses* controlam um único cursor.

Da mesma forma o foco em objetos da interface gráfica, os SO não permitem que vários focos estejam ativados simultaneamente por diferentes usuários. Se isso fosse possível poderíamos ter mais de um usuário preenchendo simultaneamente um formulário. Por exemplo, alunos em uma sala de aula poderiam controlar no computador do professor, que está conectado a um *datashow*, um cursor individual e identificado de forma que possam realizar apontamentos e anotações. Esse mesmo professor poderia delegar a interação para com uma questão que está sendo apresentada em tela para que os alunos se manifestem quanto à resposta. O componente de interface gráfica deve estar preparado para receber múltiplas respostas.

Dispositivos podem compartilhar serviços, capacidades de processamento e de comunicação para os mais diversos propósitos. Sensores estão sendo cada vez mais disseminados nesses dispositivos, permitindo entender melhor o contexto do usuário, facilitando atividades colaborativas e permitindo o estabelecimento de novos modelos de interação.

Esse cenário se estabelece como uma motivação para pesquisar como utilizar os diversos dispositivos conectados de forma integrada e sinérgica, com o objetivo de superar os limites impostos pelos modelos de interação de um determinado dispositivo, acreditando que assim é possível melhorar a qualidade de experiência do usuário.

A interação do usuário distribuída em diferentes dispositivos pode ser observada hoje como um fenômeno natural. Um estudo sobre o uso das tecnologias de informação e comunicação (TIC) por jovens (de 10 a 18 anos) concluiu que esses

usuários realizam atividades multitarefa, utilizando a televisão, vídeo games e a navegação na Internet simultaneamente (BRINGUÉ E SÁDABA, 2009). Outro estudo é o do Instituto Pew, que monitora o impacto da Internet sobre a sociedade norte-americana. Uma pesquisa com cerca de 850 especialistas em Internet, indica uma expectativa de que os usuários utilizem mais a nuvem do que o *desktop* em 2020. Essa mesma pesquisa vaticina que em 2020, as pessoas deixarão de trabalhar com software rodando em PCs de propósito geral para trabalhar na Internet a partir de seus dispositivos móveis (HORRIGAN, 2008).

Embora os nativos digitais incorporem uma infinidade de dispositivos de comunicação na sua vida pessoal e profissional, nota-se que o projeto, uma interação distribuída proposital, planejada, com o objetivo de aproveitar a sinergia da distribuição, começou a ser explorada apenas recentemente. Ferramentas começaram a surgir com vistas a permitir a construção de aplicações para essa nova natureza de interação que se estabelece.

Além disso, ainda é muito forte e presente a ideia de um usuário e um único dispositivo. Os sistemas operacionais ainda refletem uma realidade que pede por mudanças, na qual se deseja que a interação e os dispositivos possam ser compartilhados. Mesmo sob a ótica de um único usuário essa realidade se apresenta, visto que o usuário de hoje possui muitos dispositivos. Apesar dos avanços tecnológicos, o programador que deseja explorar o uso dos múltiplos dispositivos e múltiplas plataformas se depara com um ambiente que exige múltiplas habilidades do profissional.

A computação ubíqua desafia a noção convencional de um usuário logado em um dispositivo de computação pessoal, se é um *desktop*, um *tablet* ou *smartphone*. Quando o ambiente físico de um usuário contém centenas de dispositivos em rede, cada um dos quais podem ser utilizados para apoiar uma ou mais tarefas do usuário, o conceito de computação pessoal se torna insuficiente. Além disso, quando um grupo de usuários compartilha tal ambiente físico, novas formas de cooperação, partilha e colaboração são possíveis e os usuários móveis podem mudar constantemente os computadores com os quais eles interagem (ROMAN E CAMPBELL, 2002).

Os avanços tecnológicos necessários para construir um ambiente computacional ubíquo encontram-se em 4 grandes áreas: dispositivos, rede, *middleware* e aplicações (SAHA E MUKHERJEE, 2003).

Acredita-se que não será possível quebrar o paradigma PC, o “pessoal”, se não houver recursos para ampliar a criatividade do projetista. A criação de um ferramental que facilite a composição de interações distribuídas se faz, portanto, necessária.

1.2 OBJETIVOS DO TRABALHO

O objetivo principal desta pesquisa é a organização de interações distribuídas por meio de uma arquitetura orientada a serviço que permita realizar transições da interação do usuário em um ambiente composto por múltiplos dispositivos de forma integrada.

A originalidade do trabalho está na identificação e sistematização das transições de distribuição da interação e na composição de uma arquitetura de software que as suporte.

Para tanto, decompôs-se este objetivo principal em uma coleção de objetivos secundários para que possam contribuir a cada etapa para a concretização do objetivo principal, a citar:

- elucidação dos conceitos envolvidos, compreendendo a área de interfaces de usuário distribuídas (DUI) como originada em diversas disciplinas e se consolidando em IHC;
- constatação e comparação das soluções tecnológicas existentes;
- documentação das transições identificadas e respectivas características;
- definição dos serviços necessários para a realização de transições de interação e a construção de diversos cenários de uso;
- Verificação das transições de interação por meio de implementação computacional e realização de provas de conceito;
- Reflexão sobre agenda de pesquisa.

1.3 CONTRIBUIÇÃO

Ao longo deste documento poder-se-á observar que existem muitas pesquisas, em diversas áreas, na temática de interesse, sendo duas áreas as que mais nortearam esta pesquisa, a área de mídias cruzadas (*crossmedia*) e a de DUI. Acredita-se que o tema não esteja esgotado, pelas seguintes razões.

Em primeiro lugar, o conceito de mídia cruzada é bem investigado pela área de Comunicação Social, em especial na Propaganda e *Marketing*. As pesquisas dessa comunidade se concentram em aspectos da produção de conteúdo, da persuasão e do entretenimento, isso porque a mídia cruzada tem sido utilizada em jogos, séries de TV, filmes e propaganda. Sendo assim, as transições de mídia cruzada são apresentadas de uma forma muito particular, geralmente como o desafio de encontrar o próximo passo da narrativa. Não se discutem arquiteturas de apoio, salvo raros casos que serão apresentados neste texto.

Já na área de interfaces distribuídas, a comunidade está mais focada na distribuição da interface gráfica do usuário. Durante a pesquisa exploratória e sistemática não encontramos estudos específicos sobre aspectos de transição, que precisam ser mais bem compreendidos na área de IHC. Esta pesquisa contribui com os aspectos arquiteturais que possibilitam a exploração dos diversos dispositivos conectados e suas capacidades com vistas a aplicações de interação distribuída.

Pontualmente, tem-se como contribuição da pesquisa:

1. A compreensão de DUI como originada em diversas disciplinas e se consolidando em IHC;
2. Um conjunto de transições de distribuição de interação;
3. Arquitetura orientada a serviço, com serviços, transições e suas multiplicidades, componentes e provas de conceito;

1.4 METODOLOGIA DE PESQUISA

A pesquisa teve seu início motivada pela experiência do grupo de pesquisa acerca do uso sinérgico de múltiplas mídias. Um *framework* (intitulado X-gov) já havia sido concebido e experiências com o uso de integrado de *desktop*, *web* e TV já haviam sido realizadas. O trabalho de Miyamaru (2010) foi utilizado como ponto de partida para conhecimento dos conceitos básicos acerca de mídias cruzadas e da arquitetura do *framework* que já havia sido desenvolvida.

Foi realizada uma pesquisa exploratória sobre os temas de *crossmedia*, *transmedia*, *integrated media* e *multiple media* bem como o suporte tecnológico existente, como *frameworks* e *middlewares*. A investigação realizada sobre a literatura foi de cunho interdisciplinar, na qual foram abordados conceitos da área de comunicação social, de IHC e de computação distribuída.

A realização de alguns experimentos foi necessária no processo de investigação. O ambiente de desenvolvimento do *framework* X-gov foi instalado e utilizado para execução de aplicações que faziam uso do mesmo com vistas a obter um melhor entendimento do seu funcionamento e de suas capacidades. Como o X-gov implementa padrões de tarefa governamentais e o governo brasileiro formalizou padrões de interoperabilidade entre suas aplicações por meio da e-Ping (COMITÊ EXECUTIVO DE E-GOV, 2008), um estudo foi realizado com vistas a verificar a aderência do X-gov a esses padrões.

Tendo em vista o interesse desta pesquisa nas arquiteturas e dado o crescimento de aplicações dentro do modelo de computação em nuvem, foi realizado um estudo acerca desse modelo computacional, de seu modelo arquitetural e dos modelos de interação que se apresentam.

Esse primeiro processo de investigação teve como resultado duas publicações:

- Preti, J. P.; Nunes, E. P. S.; Filgueiras, L. **“Arquitetura Crossmedia para Integração de Serviços de Governo Eletrônico”**. In: WCGE – II Workshop de Computação Aplicada em Governo Eletrônico, XXX Congresso da Sociedade Brasileira de Computação, Belo Horizonte-MG, 2010.
- Preti, João P.; Filgueiras, Lucia. **“Interação em Nuvens”**. In: IHC - IX Simpósio de Fatores Humanos em Sistemas Computacionais, Belo Horizonte-MG, 2010.

A primeira publicação foi um estudo acerca da e-Ping e do *framework* Demoiselle utilizado pelo governo federal e como o mesmo poderia aproveitar os recursos do *framework* X-gov desenvolvido pelo nosso grupo de pesquisa.

A segunda publicação apresenta elementos pertinentes às formas de comunicação da computação em nuvem e da interação humano-nuvem. Apesar da computação em nuvem renovar a pesquisa em várias áreas da computação, muita

atenção tem sido dada ao PaaS (*Platform as a Service*), SaaS (*Software as a Service*) e ao IaaS (*Infrastructure as a Service*) que esse modelo oferece. Poucos estudos são encontrados acerca das implicações sociais desse novo modelo computacional e de como é a interação humano-nuvem.

Nesse período também houve a participação em três eventos:

- no Microsoft Research – FAPESP Latin America Faculty Summit 2010, Guarujá-SP, 10 a 14 de maio de 2010.
- no XXX Congresso da SBC, Belo Horizonte-MG, 20 a 23 de julho de 2010.
- no Simpósio de Fatores Humanos em Sistemas Computacionais (IHC 2010), Belo Horizonte-MG, 05 a 08 de outubro de 2010.

O primeiro evento foi uma participação sem publicação, mas que apresentava pesquisas de interesse, bem como o *framework* X-gov. As demais participações são referentes às publicações realizadas nesse período.

Após essa etapa foi possível realizar a qualificação em março de 2012 e estabelecer uma revisão sistemática de acordo com o protocolo do Apêndice A. A partir deste ponto o estilo da pesquisa se definiu na apresentação de uma “forma diferente de resolver um problema” (WAZLAWICK, 2009). Este tipo de pesquisa é característico de áreas emergentes.

Foi realizada a definição de um modelo que possibilitasse a concepção de um conjunto de transições entre dispositivos em um modelo orientado a serviço, o que possibilitou o mapeamento de um conjunto de serviços, transições e uma máquina de estados. A partir desse modelo foi possível realizar uma seleção de tecnologias, definir um ambiente de desenvolvimento, testar o ambiente e aparato tecnológico selecionado e implementar provas de conceito para verificar a viabilidade da arquitetura.

Como resultado duas publicações foram realizadas:

- Preti, J. P. and Filgueiras, L. “**Engineering transitions to bind distributed interaction**”. In: DUI 2013 – 3rd Workshop on Distributed User Interfaces, V Engineering Interactive Computing Systems, London, UK, 2013. Qualis B1 em Ciência da Computação.
- Preti, J. P.; Neto, O. and Filgueiras, L. “**Transitions: A Crossmedia Interaction Relevant Aspect**”. In: HICSS-47 – 47° Hawaii

International Conference on System Sciences, Collaboration Systems and Technologies, Hawaii, USA, 2014. Qualis A1 em Ciência da Computação.

A primeira publicação apresenta os experimentos de transição utilizando uma arquitetura orientada a serviço envolvendo *web*, *desktop* e dispositivos Android. A segunda publicação apresenta a importância do conceito de transição e como foram obtidos os requisitos para uma arquitetura que apoie esse conceito.

Como requisito do doutorado interinstitucional (DINTER) EPUSP/UFMT, foram escritos e apresentados artigos (na forma de resumo e pôster) sobre o andamento da pesquisa nas respectivas instituições de ensino (uma apresentação para cada ano):

- Preti, J. P.; Filgueiras, L. **“Mídias Cruzadas: caminhos de uma pesquisa”**. In: ERI – Escola Regional de Informática, Sociedade Brasileira de Computação – SBC, Mato Grosso-MT, 2010.
- Preti, J. P.; Filgueiras, L. **“Mídias Cruzadas e Interfaces Distribuídas”**. In: ERI – Escola Regional de Informática, Sociedade Brasileira de Computação – SBC, Mato Grosso-MT, 2011.
- Preti, J. P. and Filgueiras, L. **“Transition Support Services for Distributed User Interfaces”**. In: WPG-EC – I Workshop de Pós-graduação da Área de Concentração Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, São Paulo-SP, 2012.
- Preti, J. P. and Filgueiras, L. **“Transition Support Services for Distributed User Interfaces”**. In: WPG-EC – II Workshop de Pós-graduação da Área de Concentração Engenharia de Computação, Escola Politécnica da Universidade de São Paulo, São Paulo-SP, 2013.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho possui além desta Introdução, sete capítulos, a saber:

Capítulo 2 – apresenta um tutorial sobre a interação distribuída, as mídias cruzadas e seus *frameworks* e sua relação com DUI.

Capítulo 3 – apresenta as arquiteturas para DUI e o suporte tecnológico existente, bem como as escolhas tecnológicas realizadas.

Capítulo 4 – apresenta o modelo arquitetural proposto para realização de transições de distribuição da interação.

Capítulo 5 – apresenta a implementação do modelo e as provas de conceito implementadas.

Capítulo 6 – apresenta uma arquitetura mais ampla para DUI. Para os serviços apresentados neste capítulo não foram realizadas provas de conceito.

Capítulo 7 – faz reflexões da pesquisa sobre DUI, apresentando obstáculos e oportunidades de pesquisa.

Capítulo 8 – realiza as contribuições finais apresentando um resumo das contribuições do trabalho e apresenta trabalhos futuros que podem ser realizados para dar andamento na pesquisa.

Por fim, têm-se as Referências que subsidiaram o embasamento teórico desta pesquisa, além dos Apêndices que fazem parte do trabalho.

2 INTERAÇÃO DISTRIBUÍDA

Este capítulo tem como foco o conceito de interação distribuída. O capítulo é o resultado de uma pesquisa exploratória cujo objetivo foi o de compreender como o fenômeno da distribuição da interação, que se manifestava de forma ligeiramente diversa, em diversas áreas de conhecimento, era caracterizado e explorado em cada uma delas.

As áreas de conhecimento pesquisadas foram a de Computação Pervasiva e Ubíqua, a área de Comunicação e *Marketing* e a área de IHC. O conceito que se deseja explorar nesta pesquisa tem raízes nestas três áreas e resulta do imbricamento delas.

Dado que o foco desta pesquisa são as arquiteturas de suporte à interação distribuída, a leitura que se deu dos trabalhos nesta pesquisa exploratória teve como preocupação identificar “como se faz” a interação distribuída, mais do que identificar conceitos já formatados pelos autores.

Conforme mencionado na metodologia da pesquisa, a pesquisa exploratória iniciou-se pela área de Comunicação e Marketing, em função de trabalhos anteriores do grupo de pesquisa e se estendeu, posteriormente, para a área de Computação Pervasiva e Ubíqua, por conta da natureza do ambiente interativo.

Pesquisadores da área de IHC vêm trabalhando no refinamento do fenômeno de interação distribuída desde 2011, e nesta área estuda-se o tema com o nome de Interfaces de Usuário Distribuídas (DUI) (ELMQVIST, 2011a).

Como o capítulo evidencia, os conceitos são semelhantes em cada área, mas não coincidentes. São facetas de uma mesma realidade.

2.1 MÍDIA CRUZADA, CROSSMEDIA, TRANSMEDIA

A interação distribuída tem sido explorada desde 2004 na área de comunicação e de propaganda sob o nome de mídia cruzada, crossmedia ou *transmedia*.

2.1.1 Conceitos

2.1.1.1 Mídia

Para se compreender o conceito de mídia cruzada é importante esclarecer a definição de mídia. Essa palavra tem origem na palavra *media*, do latim, que significa meio. Buscam-se, na área de Comunicação, definições mais precisas do termo que sustentem as relações conceituais desta pesquisa.

“Mídia é um processo complexo que interfere diretamente na construção e na manutenção da sociedade, seja enquanto **dispositivo técnico**, seja enquanto **dispositivo de linguagem** (dispositivo de enunciação)” (CASAROLI E PERUZZOLO, 2008, p. 77).

O par tecnologia-linguagem recebe mais um componente na definição de mídia de Kozma (1991), segundo quem, a mídia pode ser definida pela sua tecnologia e simbologia e também pela capacidade de processamento. De forma sucinta, podem-se citar o livro, a televisão, o rádio, o celular, como formas de mídias baseadas em diferentes tecnologias. A tecnologia é a característica que permite a existência das outras duas, sendo também aquela que as restringe. A simbologia por sua vez está ligada ao formato como o conteúdo dessa mídia é apresentado: texto, audio, vídeo, etc. A capacidade de processamento de uma mídia pode complementar às de quem interage com ela, pode facilitar a execução de operações ou até mesmo permitir que a pessoa que está interagindo execute operações que ela não poderia com outra mídia. Ao assistir um vídeo, a existência (ou inexistência) das funções de avançar, retroceder e parar caracteriza exemplos de capacidade de processamento da mídia.

Com o crescimento das tecnologias de comunicação, estabeleceu-se uma pressão por mudanças na forma como a informação é obtida ou enviada, o que possibilita o desenvolvimento de novos modelos de interação entre o usuário e os serviços ofertados. Por exemplo, o SMS (*Short Message Service*) foi desenvolvido originalmente para que as empresas de telefonia enviassem notificações a seus usuários de celular, hoje esse serviço é utilizado para diversas finalidades.

Listam-se a seguir alguns termos associados a mídia que, por seus adjetivos qualificadores, mostram a diversidade de seu uso (DAVIDSON, 2010):

1. *mixed media*: é um termo que se refere à combinação de diversas mídias para compor uma experiência. Isso pode acontecer em

diversos níveis. Uma composição artística pode combinar fotografias, pinturas, papel, etc.

2. *hybrid media*: distribuição do mesmo conteúdo utilizando uma combinação de mídia digital e analógica;
3. *interactive media*: mídia digital que requisita ação do usuário;
4. *integrated media*: mídias são integradas em um sistema para distribuição em diferentes plataformas;
5. *multiple media*: uso de mais de uma mídia;
6. *multimedia*: informação digital com diferentes mídias integradas (texto, imagem, som) transportadas por um único meio (DVD, CD, etc).

Henry Jenkins, por sua vez, identificou a necessidade do conceito de *transmedia storytelling*:

Transmedia storytelling represents a process where integral elements of a fiction get dispersed systematically across multiple delivery channels for the purpose of creating a unified and coordinated entertainment experience. Ideally, each medium makes its own unique contribution to the unfolding of the story. (JENKINS, 2006, p. 95)

2.1.1.2 Mídia Cruzada

Partindo-se do conceito de transmídia, apresenta-se nesta seção o conceito de mídia cruzada.

O estudo de mídia cruzada pesquisa como utilizar o potencial de cada mídia de forma sinérgica, ou seja, na convergência entre os meios. O uso deste tipo de distribuição vem crescendo impulsionado pela quantidade de dispositivos e pela comunicação cada vez mais disponível e facilitada entre esses diversos dispositivos computacionais. O objetivo da mídia cruzada está em permitir às pessoas realizarem suas tarefas por meio da mídia mais adequada ao contexto de execução.

Já no termo transmídia é a narrativa que deve imperar sobre as tecnologias.

Embora na literatura os termos transmídia e mídia cruzada sejam usados como sinônimos, utiliza-se, neste trabalho, a definição de mídia cruzada de Filgueiras et al. (2008) que é baseada nas ideias de Boumans (2005), Dena (2004), De Haas (2005) e Wiberg, Jegers e Bodén (2007):

“Crossmedia is the collaborative support of multiple media to delivering a single story or theme, in which the storyline directs the receiver from one medium to

the next, according to each medium's strength to the dialogue" (FILGUEIRAS et al., 2008, p. 140).

Ou seja, mídia cruzada refere-se ao apoio colaborativo de várias mídias para a execução de um caso de uso procurando utilizar a mídia que fornece o melhor potencial de interação, aquela que proporcionará uma melhor experiência para o usuário, para a execução de uma determinada tarefa. Uma comunicação em mídia cruzada é uma integração complexa de experiências através de diversas mídias.

Nesta pesquisa, adota-se o conceito de transição como sendo as pontes que permitem que o fluxo de interação migre de um dispositivo para o outro.

2.1.2 O que se distribui

Pensando-se que as mídias digitais apoiam uma narrativa, isto é, um fluxo de comunicação entre o *designer* e o usuário, as aplicações de mídia cruzada têm a preocupação de compreender como distribuir a narrativa da melhor forma entre os diversos dispositivos e formatos (FILGUEIRAS et al., 2008).

Nesse contexto, o apoio colaborativo de várias mídias permite ampliar as possibilidades de interação. Outra consequência implícita é a possibilidade na troca de dados entre aplicações e a navegação entre plataformas distintas. Exemplos disso são páginas que mesclam diferentes tipos de dados, como texto, vídeo e som ou páginas que possuem um *hyperlink* para uma transmissão de rádio. Considerando que cada meio tem sua própria linguagem, capacidade e limitação, uma aplicação pode migrar de um meio para outro, de forma a obter as melhores características de cada um, caracterizando o que se denomina aplicações de mídias cruzadas (DE OLIVEIRA NETO, ROUSSEL E FILGUEIRAS, 2009).

Brodie (2003) afirmava que, de acordo com seus estudos, a informação proveniente de uma mídia tenderia a permanecer nessa mesma mídia, i.e., papel para papel, *e-mail* para *e-mail*, telefone para telefone. Somente ocasionalmente ocorreriam transições entre mídias, visto que essas transições consomem tempo e são desgastantes cognitivamente. Entretanto, esse estudo foi realizado há mais de uma década; percebe-se que as transições estão ficando cada vez mais frequentes e normais. Transições via NFC e QRCode já são comuns, por exemplo.

Um exemplo dos pioneiros na exploração dos recursos de mídia cruzada é a TV interativa (BARROS, BENINI E ZUFFO, 2006). Na ausência um *set top box* avançado, é comum estabelecer meios alternativos para produzir programas interativos ou momentos interativos em programas de TV. Isso significa que outro meio é utilizado como canal de retorno do usuário com a central de TV. Pode ser o telefone, o *e-mail*, a *Web*, *chat*, fax, ou SMS (JENSEN, 2005). Neste caso, o que se distribui é a capacidade de processamento da mídia.

Há diversos outros exemplos na área de propaganda e de entretenimento, em que o usuário é convidado a consumir um conteúdo que se encontra em outra mídia pelo mecanismo de “*Call to Action*”. Este mecanismo ao mesmo tempo estimula o usuário a migrar de mídia e torna viável esse processo de migração.

O trabalho artístico e tecnológico de Martha Gabriel é bastante poderoso para demonstrar o uso integrado de múltiplas mídias. Em “Sensitive Rose”¹, ela cria uma obra de arte interativa, que integra a mídia pessoal (o celular) com a coletiva (a projeção da rosa) por meio de QR Codes. Em sua tese, ela mostra como o uso integrado das mídias afeta e transforma a arte e vice-versa, concluindo que este uso não é apenas possível, mas provavelmente necessário, conforme aumenta a complexidade do ambiente que cerca o ser humano (GABRIEL, 2012).

Uma experiência de usuário integrada (*seamless*²), independente do canal de comunicação ou dispositivo, é um dos requisitos para se obter uma experiência *cross-channel*. Empresas e organizações que permitem ao usuário trocar de canais/dispositivos enquanto executam tarefas possuem uma vantagem competitiva (ESTES, 2013).

O atributo *seamless* é considerado um dos mais importantes a serem considerados no projeto de uma experiência que atravessa múltiplos canais. *Seamlessness* define o quanto é simples, automático e livre de atividades acessórias completar tarefas através de vários canais, caso haja o desejo. Por exemplo, se um usuário adiciona um item em um carrinho enquanto logado em um *site* de comércio eletrônico em um *smartphone*, esse mesmo item deve aparecer no carrinho quando o usuário acessar o mesmo *site* de um *notebook*.

¹ <http://www.martha.com.br/sensitive-rose/>

² Segundo <http://www.thefreedictionary.com/seamless>, traduz-se como perfeitamente consistente. Nesta pesquisa utiliza-se o termo para representar uma experiência integrada de forma fluida.

Hayes apud (MIYAMARU, 2010) define quatro níveis para a mídia cruzada:

- **1.0:** está relacionado a **distribuição**, em que o conteúdo da mídia principal é transformado para outros formatos e distribuído em outras mídias.
- **2.0:** está relacionado aos conteúdos **extras**, quando o conteúdo produzido extrapola a produção para o meio principal. Esse novo conteúdo é distribuído em diversas plataformas e é naturalmente diferente do conteúdo principal, não sendo necessariamente dependente temporal ou editorialmente.
- **3.0:** está relacionado ao estabelecimento de **pontes**. Nesta, a audiência é guiada através dos meios de comunicação para que continuem a “jornada”. Essas pontes narrativas seduzem o usuário a investigar conteúdo através das mídias ou mover-se para outro meio ou dispositivo.
- **4.0:** está relacionado às **experiências** do usuário, de forma que o usuário monte suas pontes dependendo de sua necessidade. Este nível apresenta atividades colaborativas e co-criativas. A audiência possui liberdade para criar suas próprias pontes com os diversos dispositivos presentes no seu dia a dia.

Outro aspecto relevante da abordagem de mídia cruzada é que ela segmenta o conteúdo em elementos de uma narrativa. Criar narrativas é um processo de construir personagens e eventos com estruturas e contextos bem estabelecidos com o objetivo de comunicar. O autor da narrativa tem a oportunidade de clarear, intensificar ou refinar as ideias ou sentimentos que ele deseja transmitir. A pesquisa de Neto e Filgueiras (2008) apresenta um estudo sobre o uso de narrativas lineares e não-lineares para o projeto de aplicações de mídia cruzada.

Esses conteúdos segmentados podem apresentar relações de (MIYAMARU et al., 2008):

- **corroboração:** quando o mesmo conteúdo aparece em formatos diferentes em diferentes meios;

- complementação: quando para se compreender a narrativa integralmente é necessário obter outras partes do conteúdo, que estão associadas a diversas mídias;
- concomitância: quando se requer que mais de uma mídia apresente seu elemento de conteúdo, ao mesmo tempo e de forma sincronizada, para a compreensão total da narrativa.

Apesar de ser um estudo bastante antigo, o trabalho de Olsen et al. (2000) levanta problemas que ainda hoje são pertinentes. Segundo os autores, o sucesso para aplicações de mídia cruzada depende da solução dos seguintes problemas.

- definir um protocolo interativo para comunicação entre cliente e o serviço;
- definir interfaces de usuário em um modelo independente de um modo particular de interação;
- adaptar a interação aos dispositivos de entrada disponíveis;
- adaptar a variação de tamanho de tela e resolução;
- definir interfaces que podem ser baseadas na fala e no áudio, bem como em monitores.

2.1.3 Suporte Tecnológico para Mídias Cruzadas

A seguir são apresentados *frameworks* e *middlewares* identificados na literatura que dão suporte ao desenvolvimento de aplicações de mídia cruzada. Para organizar uma sequência lógica de apresentação do suporte tecnológico, apresentam-se:

Primeiramente as tecnologias relacionadas apenas com a distribuição de conteúdo em diferentes dispositivos. Neste cenário encontram-se as ferramentas AXMEDIS e Content4All.

Em segundo lugar tecnologias que permitem que conteúdos sejam relacionados e não apenas distribuídos em diferentes meios de comunicação. Neste caso encontram-se as ferramentas iServer e OMSwe.

Em terceiro o *framework* X-gov, que permite que dispositivos e conteúdos sejam relacionados de forma a estabelecer a construção de narrativas em múltiplos dispositivos.

Por último apresentam-se tecnologias que permitem que interfaces gráficas sejam distribuídas em múltiplos dispositivos. Um problema reside no fato de definir interfaces que sejam efetivas a uma variedade de plataformas interativas. Neste sentido, tecnologias como PCP, MARIA, Munin, ZOIL e Marve fornecem contribuições em diferentes níveis que podem auxiliar na solução desse problema.

2.1.3.1 *Framework* AXMEDIS

O consórcio AXMEDIS (composto de líderes europeus produtores de conteúdos digitais, integradores, distribuidores e pesquisadores) teve por objetivo criar e manter o *framework* AXMEDIS e fornecer métodos e ferramentas inovadoras, à época, para acelerar e otimizar a produção e distribuição de conteúdos, para a produção sob demanda, para o lazer, entretenimento e valorização dos conteúdos digitais e exploração em geral. O *framework* foi criado para instalar e construir aplicações e serviços na área de produção de conteúdo, proteção e distribuição, considerando que havia o interesse no conceito COPE – *Create Once, Publish Everywhere* para a redução de custos de produção de conteúdo (FIORAVANTI, SPINU E CAMPANAI, 2007).

Trata-se de uma tecnologia que adota um formato aberto, capaz de integrar componentes de conteúdos digitais de diversos formatos de mídia (vídeo, imagens, animações, jogos, objetos de aprendizagem, multimídia, audiovisuais, documentos e audio) em diversos formatos digital, suportando uso de metadados, incluindo a identificação, classificação, aspectos de categorização, indexação, descritores, anotação, relacionamentos e de proteção. Permite a combinação de componentes de conteúdo e sua distribuição segura em relação às leis de direitos autorais, suportando diversas regras DRM (*Digital Rights Management*) e modelos de acordo com os conceitos de interoperabilidade entre modelos de DRM (NG et al., 2006).

É possível verificar uma série dessas características na Figura 1, fornecida pelo consórcio em seu site <http://www.axmedis.org>.

Figura 1 - Arquitetura do framework AXMEDIS

Fonte: <http://www.axmedis.org>

É possível verificar que o acesso ao conteúdo pode ser realizado por mais de um protocolo de acesso, via *Web Services* (WS), HTTP ou via *File Transfer Protocol* (FTP) e que não se restringe ao modelo cliente/servidor de comunicação. É possível a distribuição do conteúdo por meio de estratégia *Peer-to-Peer* (P2P). A distribuição pode ocorrer por meio da *web* através de requisições GET/POST, por meio de um servidor de *broadcast* ou por meio de *stream* para conteúdo sob demanda. O *framework* também permite a publicação do conteúdo em redes sociais ou de notícias, sendo que o acesso ao conteúdo pode ocorrer por dispositivos como *desktop*, *notebook*, PDA, *mobiles* de uma forma geral e DVB, IPTV, i-TV e dispositivos que suportem vídeo sob demanda (VOD).

AXMEDIS é uma ferramenta de mídia cruzada da Geração 1, visto que, o objetivo está em publicar conteúdos que possam ser acessados por diversos canais.

2.1.3.2 Content4All

O propósito geral do projeto Content4All (C4A), *Cross-platform Tool for Community Content Publishing*, é desenvolver um *framework open-source* de mídia cruzada, no qual o compartilhamento, a comunicação e o gerenciamento de conteúdo é alcançado por meio de *broadcast* em canais de comunicação com ou sem fio, utilizando uma infraestrutura P2P.

Praticamente tem objetivo semelhante ao do *framework* AXMEDIS, estando ambos focados na distribuição de conteúdo em múltiplas mídias. Por essa razão, considera-se que esta é uma iniciativa de Geração 1 de mídia cruzada. O uso pretendido desta arquitetura é o mesmo do AXMEDIS, criar um conteúdo e publicá-lo em qualquer lugar. Emissoras de TV, entidades educacionais, sistemas multimídia de aeronaves, são alguns exemplos de entidades interessadas em ferramentas dessa natureza, na qual o conteúdo pode ser distribuído em diferentes canais por meio de diferentes dispositivos e *players* (BURNIK et al., 2005).

2.1.3.3 Information Server (iServer) e OMSwe

Beat Signer e Moira Norrie (2005) apresentam uma arquitetura de serviço (iServer) baseada em tecnologia P2P que permite relacionar vários tipos de conteúdo. É uma arquitetura baseada em metamodelos de objetos e armazenamento orientado a objetos. Basicamente permite registrar um conteúdo como um objeto e estabelecer relações entre objetos (conteúdos).

A plataforma iServer permite a relação de mídia cruzada por meio de um modelo *Resource-Selector-Link* (RSL) estendido, conforme Figura 2:

Figura 2 - Modelo RSL que permite relacionar elementos de mídia agregando novos metadados a relação

Fonte: (SIGNER E NORRIE, 2009)

O tipo mais simples de entidade é chamado de *Resource* que representa um conteúdo completo ou uma unidade completa de informação. Um autor de hypermídia normalmente quer controlar a granularidade do *link*, selecionando partes específicas do texto, o que remete ao *Selector* que permite selecionar partes do *Resource*. Um *Link* é uma associação que pode ser de diversos tipos, uma associação *RefersTo* indica que um *Selector* está sempre associado com apenas um *Resource*, sendo que o inverso não é verdade. Para instanciação desse modelo faz-se necessário a implementação de *plug-ins*, os elementos *Resource* e *Selector* definem interfaces que precisam ser implementadas por *plug-ins* para suportar os tipos concretos de mídia.

A arquitetura do iServer está focada no registro de relações entre conteúdos, mas não quanto a distribuição do mesmo. Para distribuição do conteúdo o mesmo grupo de pesquisa desenvolveu o OMSwe. Da mesma forma como os *frameworks* Axmedis e Content4All, o OMSwe de Belotti e co-autores é uma infraestrutura de mídia cruzada de Geração 1, que facilita a produção e disseminação de conteúdo sobre diferentes meios, mas que em conjunto com iServer constitui uma infraestrutura de Geração 2 (BELOTTI et al., 2005).

Por meio dessa plataforma é possível não apenas distribuir conteúdo em diferentes canais mas também estabelecer relações entre conteúdos, tal qual acontece com documentos HTML por meio de *hyperlinks*. É possível demarcar partes de um texto, de uma imagem ou até mesmo de um vídeo e estabelecer *link* para outro conteúdo que pode ser de uma natureza diferente.

2.1.3.4 X-gov

O *framework* X-gov possibilita a distribuição da informação e dos serviços governamentais em múltiplas mídias, cada qual direcionando o cidadão ao próximo passo no processo de interação e ao meio mais adequado a esse passo. Trata-se de uma infraestrutura tecnológica que agrega componentes reutilizáveis que suportam interações de mídia cruzada em um modelo orientado a serviço. O *framework* X-gov possui elementos para (FILGUEIRAS et al., 2008):

- **Componentes de tarefas:** que implementam a comunicação entre cidadão e governo por meio de navegadores de *desktop*, dispositivos móveis e TVDi;
- **Transições de mídia cruzada:** que mantém o estado da transação mesmo ocorrendo mudança do meio de interação;
- **Troca de dados:** que implementam a migração de dados do ambiente do sistema legado para o ambiente X-gov.

Percebe-se que o núcleo do *framework* X-gov é composto por quatro elementos (MIYAMARU E FILGUEIRAS, 2009):

- **Gestor de componentes:** permite montar o serviço de mídia cruzada por meio da descrição XML, instanciando componentes;
- **Gestor de transição:** permite transportar dados transacionais de uma mídia para outra enquanto mantém o fluxo de trabalho;
- **Gestor de sessões cruzadas:** permite manter o estado de sessão dos usuários em um ambiente de mídia cruzada;
- **Gestor de conteúdo:** permite representar conteúdo como componente, permitindo resgatar o formato mais adequado para um determinado conteúdo dependendo da mídia interessada.

Figura 3 - Arquitetura do *framework X-gov*

Fonte: (MIYAMARU E FILGUEIRAS, 2009)

O modelo arquitetural X-gov permite abstrair as diversas mídias que irão interagir com os sistemas legados do governo por meio de um barramento de serviços (ESB), fazendo o papel de uma camada intermediária que permite orquestrar a interação entre os serviços de governo e as diversas mídias. Essa abstração em conjunto com a adoção de padrões abertos promove uma interoperabilidade não apenas em nível dos serviços, mas também das mídias envolvidas no processo interativo.

A vantagem da utilização de um ESB para uma aplicação de mídia cruzada é a separação de responsabilidades, isto é, a mediação de dados não ficará a cargo de cada plataforma de mídia ou para cada agência do governo, e sim uma camada de *middleware* que terá como responsabilidade esta coordenação e integração dos servidores e consumidores de dados.

Esse tipo de solução se enquadra na Geração 3 de mídia cruzada porque permite que pontes sejam estabelecidas entre os canais de comunicação. Essa característica permite guiar o usuário não apenas para outro conteúdo mas inclusive para outro canal de comunicação, permitindo que a interação ocorrida no canal de origem seja aproveitada pelo canal de destino.

É possível por meio do X-gov que uma atividade de matrícula escolar, por exemplo, seja realizada utilizando TV, *desktop* e celular. Em uma das provas de conceito realizadas por Miyamaru (2010), a TV era utilizada para fornecer informações iniciais da atividade de matrícula escolar, abertura de um protocolo de matrícula e acompanhamento do processo. Por meio de QRcode um *link* era estabelecido para que o processo de matrícula tivesse continuação por meio de um celular, no qual era possível realizar a busca e seleção da escola em que se desejava realizar a matrícula. Após selecionada a escola, o preenchimento do formulário de cadastro era realizado por meio de um *desktop*. O acompanhamento do andamento do processo poderia ser feito pela TV informando o número do protocolo. Ao final do processo uma mensagem SMS era enviada para o celular do interessado informando que já poderia ser realizado o agendamento para entrega dos documentos. O SMS continha um *hyperlink* para a página que permitia realizar a atividade de agendar.

2.1.3.5 *Model-based LAnguage foR Interactive Applications* (MARIA)

A representação de interfaces para atender ao requisito da responsividade, isto é, a adaptação a diferentes fatores de forma, levou às linguagens de definição de interfaces de usuário (UIDL – *User Interface Definition Language*), que são modelos de descrição traduzíveis para os diferentes dispositivos. Entre essas linguagens, apresenta-se a linguagem MARIA.

Quando em 2009, identificaram-se algumas tendências para aplicações interativas, as quais encontram um desafio e definem requisitos para:

- o acesso a aplicações encapsuladas em um número pré-existente de serviços *Web* que podem estar distribuídos em qualquer lugar;
- o uso de vários tipos de dispositivos (em particular móveis);
- o uso de vários tipos de sensores (como acelerômetros, de inclinação e bússola eletrônica), tecnologias de localização (como GPS, RFIDS) e;
- de diversas modalidades de interação (multitoque, por gestos, interação baseada em câmera).

De fato, os serviços *Web* estão sendo cada vez mais utilizados para apoiar o acesso remoto a funcionalidades das aplicações, que normalmente são

descritas por meio de arquivos WSDL (*Web Services Description Language*). Portanto, com a disponibilidade de serviços *Web*, que são definidos antes das aplicações interativas, o desafio está no reuso dessas funcionalidades, no projeto e desenvolvimento do serviço de fachada (SFE) para esses serviços e de como combinar tais funcionalidades em aplicações integradas com interfaces (UI) apropriadas aptas a se adaptar aos diversos contextos.

Segundo Paternò, Santoro e Spano (2009) essa combinação de fatores tem intensificado a necessidade de uma linguagem declarativa que seja universal, apresentando a linguagem MARIA. Trata-se de uma UIDL que provê geração dinâmica de interfaces gráficas adaptadas aos diferentes dispositivos durante o processo de transição do usuário.

A linguagem MARIA possui sintaxe XML e é decomposta em uma linguagem para uma descrição abstrata, independente de plataforma, seguida de uma série de linguagens dependentes de plataforma que especializam a definição abstrata dependendo dos recursos de interação considerados.

Estudos de Paternò, Santoro e Spano (2009) baseados em Bandelloni, Mori e Paternò (2005) têm pesquisado como manter e utilizar modelos da linguagem MARIA em tempo de execução para suportar a migração de interface do usuário. Migração de interfaces de usuário são aplicações interativas que podem se transferir entre diversos dispositivos preservando seu estado, dando a sensação de que a atividade não foi interrompida.

Na Figura 4 são apresentados os passos necessários para a realização da migração da interface do usuário para outro dispositivo.

Figura 4 - Passos para a realização da migração da interface gráfica

Fonte: (PATERNO', SANTORO E SPANO, 2009)

Basicamente são necessários 3 arquivos para descrição de uma mesma página para 2 dispositivos. O primeiro arquivo representa a descrição abstrata da página; basicamente, descreve quais são os elementos de interface mínimos independente do dispositivo e de *layout*. Outro arquivo faz a descrição concreta dessa página em ambiente *desktop* e o último faz para descrição concreta em PDA. O *Proxy Server* é responsável por saber qual a descrição concreta deve ser resgatada para o dispositivo que está solicitando (no caso o *desktop*), enquanto o *Reverse Semantic Redesign State Mapper Generator* é responsável por fornecer a descrição concreta para o dispositivo de destino já com os valores dos componentes dessa interface (no caso o PDA).

Um estudo de caso para o jogo pac-man (feito em XHTML e Javascript) foi realizado para validar os passos migrando do *desktop* para um celular iPhone.

2.1.3.6 Marve

É um *framework* focado em estender as funcionalidades dos *toolkits* de GUI, como Java Swing e .Net Forms com características de interação distribuída. Marve pode ser considerado uma arquitetura de mídia cruzada porque fornece um suporte para transportar componentes entre diferentes dispositivos em tempo de execução. O objetivo é criar um mecanismo que permita aos programadores

controlar os componentes distribuídos da interface gráfica da mesma forma como o fazem no modelo tradicional.

Assim como o MARIA ele permite descrever em XML características da interface e seus componentes mas com propósitos diferenciados. Na linguagem MARIA o objetivo está em descrever a interface de forma que possa ser reutilizada para descrição de interfaces concretas em outras plataformas. No MARVE a preocupação está em descrever os componentes da interface e suas características com relação a (FRÖBERG, ERIKSSON E BERGLUND, 2011):

- eventos distribuídos: por exemplo, um usuário ao clicar em um botão em seu *desktop* causa uma ação de mudar uma imagem no *desktop* de outro usuário;
- e distribuição do componente: permite descrever como componentes serão posicionados nos dispositivos.

Este último pode estabelecer relações entre componentes e dispositivos de três tipos:

- atômica: refere-se quando um componente pode estar visível em um único dispositivo em um determinado instante de tempo;
- espelhada: quando um componente é posicionado em dois ou mais dispositivos em qualquer instante de tempo;
- clonada: quando um componente é posicionado em dois ou mais dispositivos em qualquer instante de tempo, mas cada componente é único e não está conectado com a origem do componente.

Um ponto importante a ser observado é que no Marve os componentes são transferidos entre dispositivos serializando os mesmos no formato binário. Essa abordagem traz como desvantagem a necessidade de que a plataforma de interface gráfica dos dispositivos seja a mesma, incluindo a versão. Aplicações Marve utilizando Java Swing somente poderão ser executadas em dispositivos que suportem essa tecnologia.

2.1.3.7 Munin e ZOIL

Munin é um *middleware* P2P em Java. O *toolkit* é baseado em 2 componentes principais: uma memória replicada e compartilhada e um canal comum de eventos (para sincronização). O propósito está em permitir que desenvolvedores

possam facilmente construir espaços de visualização, no qual diversas superfícies, dispositivos, monitores e outras modalidades de entrada e saída sejam combinadas em um único e coerente espaço visual. O *middleware* não suporta nenhuma forma de visualização por *stream*; cada ponto possui um conjunto de dados (ou subconjunto) para visualizar, sendo que cada qual é responsável pela renderização da representação visual (ELMQVIST, 2011b).

ZOIL é um *framework* para construção de interfaces distribuídas, *zoomable*, WIMP (*window, icon, menu, pointing device*) e orientadas a objetos. Assim como o *middleware* Munin, ZOIL distribui a interface do usuário por meio da sincronização de um modelo de dados da área visual de trabalho, ao invés de utilizar protocolos que se baseiam no nível do pixel como o VNC (*Virtual Networking Computing*) (ZÖLLNER, JETTER E REITERER, 2011).

Nesse modelo de distribuição de interface existe a representação de um único espaço visual em um servidor. Cada cliente seleciona o que tem interesse em visualizar e é responsável por interpretar o modelo e compor a interface gráfica para o usuário. Nesse tipo de aplicação, caso dois dispositivos estejam interagindo com o mesmo objeto no servidor o estado é refletido em ambos dispositivos.

Esse tipo de solução difere da MARIA tendo em vista que no Munin e ZOIL toda interação tem efeito em um modelo dessa interface no servidor, que é compartilhado com outros dispositivos. Na MARIA, a interação e o estado dos objetos é mantido no cliente, não existe uma memória comum. Quando se deseja transferir a interface é realizada uma serialização do estado dos objetos dessa interface para serem aplicados em outra interface para outro dispositivo.

2.1.3.8 *Personal Cloud Platform* (PCP) (ARDISSONO et al., 2009)

Com a adoção de arquiteturas orientadas a serviço, o crescimento da oferta de *software* como serviço (SaaS) e dos *softwares* de gerenciamento de computação em nuvem integrando serviços remotos está se tornando realidade. Contudo, a contribuição atual para o usuário final é limitada pela falta de sincronização entre as aplicações.

Ardissono et al. (2009), atentos ao problema da sincronização de aplicações em um contexto como da nuvem, propuseram um protótipo de

middleware para tratar as diversas interfaces dos serviços de forma a permitir a sincronização utilizando estratégias do tipo *push*.

O *middleware* foi intitulado de *Personal Cloud Platform* (PCP) e o mesmo agrega um componente de *Cross-Application Context* responsável por armazenar informações compartilhadas entre os vários serviços, permitindo a propagação da mudança de estado das informações aos serviços interessados. Essa ponte intencional entre as aplicações caracteriza a arquitetura de Ardissono como Geração 3 e traz contribuições relacionadas ao estabelecimento dessas pontes em uma arquitetura orientada a serviço. Esse trabalho pode ser utilizado por ferramentas como Maria e Marve para possibilitar a construção de aplicações onde eventos são propagados de maneira mais fracamente acoplada.

2.1.3.9 Digital Living Network Alliance (DLNA)

Com vistas a permitir que serviços multimídia sejam disponibilizados no ambiente doméstico, independentemente da localização física do conteúdo digital, a *Digital Living Network Alliance* (DLNA³) criou em 2004 um guia, o *Home Networked Device Interoperability Guidelines*, para promover a interoperabilidade dos vários dispositivos de áudio e imagem no ambiente doméstico. A última versão (1.5) de suas recomendações data de outubro de 2006 (DLNA, 2006).

O escopo desse guia é focado na interoperabilidade entre dispositivos de mídia digital em ambientes domésticos sem se preocupar com métodos de transmissão e características de segurança. Outra característica é sobre o controle e a descoberta de dispositivos que é baseada na arquitetura *Universal Plug and Play* (UPnP), ou seja, o mecanismo de descoberta é baseado em *multicast* dentro dos limites da rede doméstica, contudo dispositivos DLNA não podem ser descobertos ou controlados por dispositivos conectados a um IP externo.

Pesquisas como as de Yeon-joo Oh et al. (2007) e Jung-tae kim et al. (2007) propuseram soluções para estender o alcance da DLNA para além da rede doméstica local. No trabalho de Yeon-joo Oh et al. (2007) obteve-se êxito por meio do uso de um *Home Server* (HS) utilizando mensagens no formato SIP (*Session Initiation Protocol*).

³ <http://www.dlna.org>

SIP é um protocolo da camada de aplicação que permite estabelecer configuração, modificação e desconexão de chamadas. O protocolo permite negociação das informações de sessão da mídia para uma conectividade fim-a-fim, modificação dos dados nessa seção e realizar desconexão entre os dispositivos.

Um HS adota vários padrões internacionais incluindo OSGi e DLNA. É utilizado basicamente para suportar conexão entre redes, qualidade de serviço (QoS) e segurança entre aplicações, bem como oferecer uma gama de serviços inteligentes.

A *Open Service Gateway Initiative Alliance* (OSGi) é um consórcio mundial de inovadores em tecnologia que avança no processo de criar especificações abertas que permitem a montagem modular de *software*. A plataforma de serviços OSGi facilita a componentização de módulos de *software* e aplicativos e garante a interoperabilidade de aplicações e serviços através de uma variedade de dispositivos de rede (OSGI ALLIANCE, 2011).

2.1.4 Capacidade de suporte a corroboração, complementação e concomitância nas arquiteturas estudadas

O Quadro 1 apresenta uma comparação das tecnologias apresentadas segundo aspectos das relações de mídias cruzadas. Essa análise procura verificar se na tecnologia em questão existe a preocupação em estabelecer essas relações.

Quadro 1 – Suporte das ferramentas às características de mídia cruzada e modelo de arquitetura

Ferramenta	COB	COM	CON	Geração	Arquitetura
AXMEDIS	•	o	o	1	P2P
Content4All	•	o	o	1	P2P
OMSwe	•	o	o	1	C/S
iServer	•	o	o	2	P2P
MARIA	o	•	•	3	C/S
X-gov	•	•	•	3	C/S
PCP	o	o	•	3	C/S
Marve	o	•	•	3	C/S
Munin	•	o	•	4	P2P
ZOIL	•	o	•	4	C/S
DLNA	o	o	•	4	P2P

Legenda: **COB** = Corroboração; **COM** = Complementação; **CON** = Concomitância.

Fonte: Próprio autor.

Como é possível observar no Quadro 1, a ferramenta X-gov é a única, dentre as 11 ferramentas pesquisadas, que atende as 3 relações de conteúdo de mídia cruzada, mas pertence a Geração 3, por não dar liberdade para o usuário

estabelecer as pontes de transição de forma dinâmica em tempo de execução. Já ferramentas de Geração 4 Munin e Zoil, permitem que transições entre dispositivos possam ocorrer de acordo com o interesse do usuário, mas são fortemente acopladas, não permitindo que aplicações desenvolvidas em outras plataformas sejam integradas. DLNA foi considerada como de Geração 4 por permitir que pontes sejam estabelecidas entre dispositivos de forma *ad hoc*, permitindo que conteúdos sejam compartilhados e que elementos de sincronismos possam ser estabelecidos, como utilizar o smartphone para controlar um vídeo que está sendo reproduzido na TV.

Quanto à arquitetura de rede o que se observa nas 11 ferramentas é o uso da arquitetura de rede cliente-servidor (C/S) para Geração 3. A abordagem C/S concentra as operações no servidor, requerendo assim um equipamento de hardware e um sistema operacional em profunda sintonia com os demais sistemas em execução, pois o aumento de processamentos (simultâneos) afeta diretamente o desempenho do equipamento e com isso a qualidade de retorno do aplicativo ao usuário. Já ferramentas da Geração 1 e 4 uma opção maior por redes P2P que seguem a premissa de não concentrar todas as operações em uma única estação (servidor), atuando então no modelo descentralizado, visando sanar o problema citado.

Nesta área, os principais problemas a serem resolvidos são:

- Como fazer um conteúdo ser representado em diferentes dispositivos?
- Como usar adequadamente o potencial de cada dispositivo para apresentar conteúdo?
- Como estimular o usuário a migrar de um dispositivo para outro?
- Como avaliar a capacidade de conversão do conjunto de canais?

2.2 COMPUTAÇÃO UBÍQUA

Durante a pesquisa exploratória acerca da interação distribuída, observou-se que a área de Computação Ubíqua e Pervasiva também se fez presente. A computação ubíqua defende a construção de sistemas distribuídos que ajudam a transformar espaços físicos em ambientes computacionalmente ativos e inteligentes. O projeto de sistemas e aplicações nestes ambientes deve levar em

consideração a existência de dispositivos heterogêneos, usuários móveis e contextos que mudam rapidamente (RANGANATHAN E CAMPBELL, 2003).

Nesse sentido faz-se importante um estudo acerca dessa área de pesquisa com vistas a identificar pontos de contribuição que certamente existem para esta pesquisa.

2.2.1 Conceitos

Não foi encontrada uma definição exata de interação distribuída dentro da computação ubíqua. A computação ubíqua é vista como um paradigma computacional na qual ocorre mínima intervenção do usuário, enfatizando a detecção do contexto do usuário e seu comportamento com o objetivo de melhorar a qualidade de experiência do mesmo. Nesse sentido a observação do contexto é requisito essencial para alcançar tal ubiquidade (KIANI et al., 2005).

Segundo Roman e Campbell (2002) 5 características são essenciais para aplicações ubíquas:

- **Consciência de recursos (*Resource-Awareness*):** usuários em cenários de computação ubíqua são cercados por centenas de dispositivos, incluindo sensores, monitores e CPUs. Para que os aplicativos possam explorar esses recursos, eles devem estar cientes de sua existência, as capacidades, disponibilidade e cardinalidade desses recursos.
- **Sensibilidade ao contexto (*Context-Sensitivity*):** o contexto é uma das propriedades mais importantes em computação ubíqua e afeta o comportamento das aplicações.
- **Múltiplos dispositivos (*Multi-Device*):** em um ambiente no qual usuários são cercados por centenas de dispositivos, a noção de *login* em um único dispositivo torna-se inadequada. Usuários entram no espaço ativo (implícita ou explicitamente) e podem tirar vantagem de qualquer entidade contida no espaço que estejam de acordo com as políticas de disponibilidade e segurança.
- **Centrado no usuário (*User-Centrism*):** com a sensibilidade ao contexto e os múltiplos dispositivos aparece uma terceira propriedade essencial: centrado no usuário. Para acomodar o particionamento da

aplicação em vários dispositivos, que variam ao longo do tempo, ligam-se aplicativos aos usuários e mapeam-se as aplicações para os recursos presentes no espaço.

- **Mobilidade (*Mobility*):** características de particionamento da aplicação e centrado no usuário exigem que as aplicações sejam móveis. Consideram-se dois tipos diferentes de mobilidade: mobilidade intra- e inter-espaço. Mobilidade intra-espaço está relacionada com a migração de componentes de aplicação dentro de um espaço ativo e é o resultado de particionamento da aplicação entre diferentes dispositivos. Mobilidade intra-espaço permite aos usuários e aos serviços externos mobilidade para mover componentes de aplicativos entre diferentes dispositivos. Mobilidade inter-espaço se concentra na mobilidade de aplicativos entre diferentes espaços.

2.2.2 O que se distribui

Ambientes de computação ubíqua são caracterizados por vários sensores capazes de detectar uma variedade de contextos diferentes. Os tipos de contextos incluem contextos físicos (como localização, tempo), os contextos ambientais (luz, tempo e níveis de som), contextos de informação (cotações da bolsa, resultados desportivos), os contextos pessoais (saúde, humor, cronograma de atividades), os contextos sociais (atividade em grupo, as relações sociais, outras pessoas em uma sala), contextos de aplicação (*e-mail* recebido, sites visitados) e contextos de sistema (o tráfego de rede, *status* de impressoras) (RANGANATHAN E CAMPBELL, 2003).

Assim, pode-se dizer que em um ambiente de computação ubíqua, o que é distribuído são os sensores de contexto. A experiência de usuário se forma com o contato com esses múltiplos sensores.

2.2.3 Como se faz?

Uma arquitetura sensível ao contexto deve fornecer diferentes mecanismos à disposição para lidar com o contexto. Estes mecanismos incluem mecanismos de raciocínio como regras escritas em diferentes tipos de lógica (lógica de primeira ordem, a lógica temporal, lógica difusa, etc), bem como mecanismos de

aprendizagem (como redes bayesianas, redes neurais ou reforçamento de aprendizagem) (RANGANATHAN E CAMPBELL, 2003).

Existem várias arquiteturas para construção de ambientes ubíquos, como *Context Toolkit*, Gaia, Aura, ICrafter, Solar, *Context Fabric*, M3, CAMUS, MoCA, PICO, Scooby. Muitas delas são clássicas como Context Toolkit que possui mais de 3490 citações, Gaia que possui mais de 1390, Aura e ICrafter. A seguir, apresenta-se brevemente cada uma, visto que, se observa que os estudos de IHC para essa área estão mais centrados nas interações multimodais e no projeto de *hardware* centrado no usuário. Quanto aos requisitos verifica-se uma preocupação comum em fornecer mecanismos para:

- Descoberta de dispositivos;
- Facilitar extração de dados de sensores;
- Identificação do contexto do usuário;
- Interação multimodal; e
- Gerenciamento de eventos distribuídos.

2.2.3.1 *Context Toolkit*

Context Toolkit utiliza o conceito de *widget* para obter informações contextuais a partir de sensores e transmiti-las para interpretadores ou servidores para agregação. Interpretadores ou servidores utilizam simplesmente o protocolo HTTP e XML como linguagem para descrição do contexto (DEY, 2001).

Essa arquitetura trouxe contribuições relacionadas:

1. ao baixo acoplamento entre interface gráfica e sensores;
2. ao armazenamento de contexto;
3. a distribuição de informações de contexto e;
4. a agregação de informações de contexto acerca de uma entidade (como uma pessoa, por exemplo)

2.2.3.2 Gaia

Gaia é um *middleware* muito referenciado que tem por objetivo auxiliar na criação de espaços físicos, como casas e edifícios inteligentes e suporta sensibilidade ao contexto em *smart spaces*. Adota um modelo de predicados para dados do contexto representados em ontologias DAML, utilizando lógicas distintas e

técnicas de aprendizado de máquina para apoiar a inferência do contexto. Gerencia os recursos e serviços de um espaço ativo, fornece localização, o contexto, serviços de eventos e armazena informações sobre o espaço ativo. Possui três componentes principais: o *kernel*, o quadro de aplicação, e as aplicações. A implementação de Gaia utiliza CORBA, no entanto, o sistema pode também utilizar SOAP, RMI, ou *middleware* de comunicação personalizado. Gaia oferece cinco serviços básicos: gerenciador de eventos, serviço de contexto, serviço de presença, repositório e sistema de arquivo de contexto (ROMAN et al., 2002a).

2.2.3.3 Aura

O projeto Aura se preocupa com a mobilidade do usuário. Os autores advogam que as abordagens para a mobilidade do usuário baseiam-se em uma de quatro técnicas, nenhuma das quais atinge totalmente esses objetivos. Uma abordagem é a de suportar o máximo de computação que um utilizador precisa no dispositivo móvel. A segunda abordagem é realizar o processamento, por meio de acesso remoto, em um servidor que armazena dados dos usuários e suas preferências, como terminais X fazem⁴. Uma terceira abordagem é fornecer aplicações padrão que são portadas para e instaladas em todos os ambientes. Essas aplicações são estendidas para se tornarem “cientes” da intenção e mobilidade do usuário. Uma quarta abordagem é o de proporcionar plataformas virtuais (tais como a Máquina Virtual Java) que permitem que o código seja móvel, seguindo o usuário conforme necessário. O projeto Aura é baseado no conceito de aura pessoal, que atua como um substituto para o usuário móvel que representa. Quando um usuário entra em um novo ambiente, a sua aura empacota recursos adequados para apoiar a tarefa do usuário. Além disso, uma aura captura as restrições que o contexto físico ao redor do usuário impõe sobre as tarefas que requerem fontes de informação diversas. Para ativar a ação de uma aura pessoal, a documentação do *framework* esclarece que novas funcionalidades e interfaces são necessárias a nível de sistema. O *framework* também capta a natureza das tarefas do usuário, preferências pessoais e intenções (SOUSA E GARLAN, 2002).

⁴ Um **Terminal X** é um terminal de visualização e de entrada de dados para aplicações clientes do [X Window System](http://pt.wikipedia.org/wiki/Window_System). Vide http://pt.wikipedia.org/wiki/Window_System.

2.2.3.4 ICrafter

O principal objetivo do ICrafter é permitir que os usuários possam interagir de forma flexível com os serviços em seu ambiente usando uma variedade de modalidades e dispositivos de entrada. Os pesquisadores observaram na época uma onda de interesse em serviços e estruturas de interface do usuário para ambientes de computação ubíqua (PONNEKANTI et al., 2001).

ICrafter contribui com três aspectos importantes:

- inteligência na infraestrutura para selecionar, gerar e/ou adaptar interfaces de serviço. Isso ajuda a transferir serviços e aparelhos e tem várias vantagens, tais como extensibilidade, e um melhor manuseio de serviços legados e recursos limitados dos dispositivos;
- um esquema novo agregação de serviços *on-the-fly*;
- facilita a criação de interfaces de usuário que são portáteis através de espaços de trabalho, mas ainda refletem o espaço de trabalho do contexto atual.

O *framework* apoia diversas modalidades, mas também aparelhos diferentes com a mesma modalidade. É interessante notar que os autores apontam que um desafio consiste em facilitar a concepção de interfaces de usuário que possam ser reutilizadas através dos espaços de trabalho. O *framework* deve facilitar a criação de interfaces de usuário para as combinações de serviços.

Se a união das interfaces gráficas de uma câmera fotográfica e de uma impressora for devolvida, o usuário ainda terá que solicitar a câmera para tirar uma foto, salvá-la em um local temporário, carregar a imagem salva para a impressora, e pedir para imprimir. Apenas compor interfaces gráficas em que os serviços não estão integrados não é suficiente.

O ideal é que a UI da combinação da câmera-impressora possa "reconhecer" que as duas possam ser combinadas de maneira mais útil (que a saída da câmera seja enviada para a impressora) e permitir que o usuário possa fazer isso tão facilmente como possível.

Um gerador pode ser associado com uma instância de serviço ou uma interface de serviço (por exemplo uma *PowerSwitchInterface*) ou qualquer padrão de instância de serviço ou interface. Exemplos de padrão incluem "todos os serviços que implementam a *PowerSwitchInterface*" (PONNEKANTI et al., 2001).

Há outros middlewares na literatura que não parecem ter promovido o mesmo impacto daqueles já apresentados, mas que se alinham com aqueles em termos de objetivo. É o caso de Solar que a contribuição está em um modelo baseado em grafo para agregação e disseminação do contexto (CHEN E KOTZ, 2005). Outro é o *Context Fabric* que fornece uma infraestrutura distribuída para adquirir e disponibilizar dados do contexto por meio dados em XML e utiliza XPath como linguagem de consulta (HONG, 2002). M3 que traz contribuições relacionadas a reconfiguração dinâmica do sistema ubíquo para uma variedade de paradigmas de comunicação (HAN et al., 2004). CAMUS que apresenta uma estratégia para extrair e apresentar dados de sensores com a finalidade de melhorar a descrição de ambientes (KIANI et al., 2005). MoCA (*MOBILE Collaboration Architecture*), PICO e Scooby que apoiam o desenvolvimento de serviços sensíveis ao contexto e aplicações sensíveis ao contexto para colaboração móvel (SACRAMENTO et al., 2004) (KUMAR et al., 2003) (ROBINSON, WAKEMAN E OWEN, 2004).

2.3 DISTRIBUTED USER INTERFACES

Com a computação pervasiva e os múltiplos dispositivos interativos apresenta-se a necessidade de novos modelos de interface. Por exemplo, o modelo de interface gráfica (GUI) do final dos anos 70 foi projetado para executar em um único computador. Como consequência, o desenvolvimento para múltiplos dispositivos é mais suscetível a erros e demanda maior esforço (BANG et al., 2005).

A área de interfaces de usuário distribuídas tem se tornado um novo campo de pesquisa e desenvolvimento em IHC. *Workshops* sobre DUI têm sido realizados anualmente pela comunidade de IHC a partir de 2011, tendo sua terceira edição em 2013.

O primeiro *workshop* DUI aconteceu em conjunto com o CHI'2011 e apresentou as novas demandas de pesquisa na área de IHC, pelo fato das UIs irem além do controle de um único usuário, em uma única plataforma e em um mesmo ambiente. Em 2011 o autor desta pesquisa teve um artigo aceito mas não pôde participar do evento (Apêndice B), nesse artigo, apresentam-se as investigações iniciais sobre mídias cruzadas na computação em nuvem. O DUI de 2012 também foi em conjunto com o CHI e, por sua vez, apresentou discussões acerca dos tópicos de colaboração e usabilidade na área de DUI. Já o de 2013 foi em conjunto com o

EICS e apresentou modelos, métodos e ferramentas para DUI. Neste último, houve a participação do autor desta pesquisa, *in loco*, com a publicação de parte desta pesquisa (PRETI E FILGUEIRAS, 2013).

O conceito de DUI ainda está em processo de definição. Encontramos em nossa pesquisa uma definição em 2002. Na primeira edição do *workshop*, os autores propuseram outras definições. Mesmo na segunda e terceira edição ainda é possível extrair outras. Segue lista de definições encontradas:

“Distributed user interfaces are a user interface that distributes its components among several interaction devices available in the environment” (BERGLUND E BAANG, 2002, p. 1).

“Interfaces whose different parts can be distributed in time and space on different monitors, screens, and computing platform, depending on several expressing parameters” (DEMEURE et al., 2008, p. 61).

“DUI is a UI with the ability to distribute parts or whole of its components across multiple monitors, devices, platforms, displays and/or users” (MELCHIOR et al., 2009, p. 70).

“Distributed User Interfaces (DUIs) enable end users to distribute in time and space any widget and piece of information across different contexts” (SENDÍN E LÓPEZ, 2011, p. 13).

“... UIs, that support any combination (>1) of different input and output Interaction Resources (IRs) to simultaneously interact with a single application” (BLUMENDORF, ROSCHER E ALBAYRAK, 2011, p. 26).

“A Distributed User Interface is a user interface with the ability to distribute part of its components, or all of them, across multiple screens, devices, platforms and users” (FARDOUN, LÓPEZ E VILLANUEVA, 2011, p. 42).

“A distributed user interface is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time” (ELMQVIST, 2011a, p. 7).

“The result of migrating a part or all the interface components of an existing user interface (distributed or not distributed) across multiple devices or platforms” (VILLANUEVA, TESORIERO E GALLUD, 2012, p. 40).

“Distributed User Interfaces (DUIs) can be distributed in at least three senses: across different physical locations, across different computers, and across different displays” (NACENTA, 2013, p. 8).

Nesta pesquisa adota-se a definição de Elmqvist (2011a) por estar mais bem formalizada, conseguir abstrair um maior número de dimensões da distribuição e não se restringir a aspectos de interface gráfica do usuário. O que se observa também nesse contexto é que o conceito de mídia cruzada é mais amplo, não se restringindo apenas a mídia digital, mas a qualquer tipo de mídia.

Apesar de adotar a definição de Elmqvist (2011a), outras pesquisas complementam a visão sobre DUI, como a pesquisa de Luyten e Coninx (2005) que apresentam que uma aplicação DUI pode ser classificada por 3 (três) dimensões:

- **orientada a localização:** determinada pela localização física do usuário;
- **orientada a atividade:** determinada pelas atividades que um ou mais usuários têm que executar em um período de tempo para alcançar um objetivo comum;
- **orientada ao dispositivo:** determinada pelas propriedades disponíveis dos recursos que estão acessíveis ao usuário.

Essa pesquisa já observa um aspecto importante além do espaço e da plataforma que é a atividade do usuário.

Já a pesquisa de Demeure et al. (2008), as examina de acordo com 4 dimensões: computação, comunicação, coordenação e configuração. Esse modelo é conhecido como *framework* 4C, um modelo de referência para DUIs.

O trabalho de Melchior et al. (2009), por sua vez, buscou identificar situações de uso para DUI quanto a migração de elementos de interface gráfica:

- ***multi-monitor usage*:** um usuário em uma plataforma distribui sua interface gráfica para vários monitores conectados a mesma plataforma;
- ***multi-device usage*:** um usuário utiliza diferentes dispositivos concomitantemente, utilizando, ou não, o mesmo sistema operacional;
- ***multi-platform usage*:** um usuário utiliza plataformas computacionais distintas, provavelmente em sistemas operacionais distintos. Note que o uso no formato *multi-device* implica no formato *multi-platform*, mas a

recíproca não é verdadeira: um usuário pode utilizar diversos computadores (*multi-platform*) que são similares (não sendo *multi-device*);

- ***multi-display usage***: combinação de *multi-monitor* e *multi-device*. Um usuário pode distribuir a UI através de vários monitores e dispositivos simultaneamente;
- ***multi-use***: representa uma extensão dos usos anteriores para vários usuários concorrentemente. Neste caso, um ou mais usuários podem distribuir todos ou partes de seus componentes de UI para vários monitores, dispositivos, plataformas ou telas.

Na pesquisa de Blumendorf, Roscher e Albayrak (2011) foram identificadas características principais das DUIs: seleção automática de dispositivo, sensibilidade ao contexto do usuário, controle do usuário, compartilhamento de informação, uso simultâneo (interação multimodal). Além disso, eles observam que DUIs são *multi-* e *cross-device*, *multi-* e *cross-modal*, *multi-* e *cross-user* e *multi-* e *cross-application*. "*Multi*" refere-se ao fato de que as UIs endereçam mais de um elemento por dimensão (i.e. dispositivo, modalidade, usuário, aplicação) ao mesmo tempo, enquanto "*cross*" refere-se ao fato de que um elemento pode mudar dinamicamente em tempo de execução (e.g. transitando do áudio para uma saída visual ou transferindo o controle para outro usuário). Nesse trabalho, além das dimensões já apresentadas, a dimensão do usuário é inserida e o conceito de *multi* e *cross* são bem aplicados.

No trabalho de Villanueva et al. (2009) são definidas propriedades consideradas essenciais para DUIs: portabilidade, decomposabilidade (e composabilidade), simultaneidade e continuidade. Eles apontam outras propriedades que podem ser derivadas das primeiras: multiplataforma, multimonitor e multiusuário, e definem outras, não essenciais, mas desejáveis, como consistência e flexibilidade. Este trabalho, por sua vez, apresenta um aspecto até agora não verificado que é a decomposabilidade e a composabilidade. Elementos de interface podem ser decompostos em elementos menores ou mais granulados para distribuição. O inverso também é válido, elementos de interface mais granulares podem ser compostos para formar novos elementos de interface.

Já a pesquisa de Melchior, Vanderdonckt e Van Roy (2011) contribui com a formalização de primitivas de distribuição de elementos de interface gráfica para DUI, permitindo realizar operações como copiar, colar, recortar, duplicar, mover e recuperar. Essas primitivas estão relacionadas com a migração dos componentes de interface, ou seja, serializá-los de uma lado e deserializá-los na outra ponta, como exemplo do *framework* MARVE (FRÖBERG, ERIKSSON E BERGLUND, 2011).

Nesta tese, a preocupação da distribuição da interação está em um nível bastante granular, a distribuição da interação por componente de interface gráfica, sem a necessidade de migração do componente de interface.

Uma análise realizada das publicações realizadas nas três edições do *workshop* sobre DUI, permitiu formular o mapa conceitual da Figura 5 (DUI, 2011) (DUI, 2012) (DUI, 2013).

Figura 5 – Áreas de concentração da pesquisa em DUI nos 3 workshops e respectivos pesquisadores

Fonte: Próprio autor

Verifica-se uma demanda de pesquisa em diversos segmentos, com uma concentração maior em estudos de caso, modelos de distribuição de interface gráfica, bem como na identificação de preocupações e características necessárias ao projeto de *software* para DUI. Também verifica-se que a área de DUI busca facilitar a complexidade em tratar interações do usuário em dispositivos heterogêneos em ambientes móveis de forma dinâmica.

Os estudos de caso mostram aplicações com características de DUI que podem ser aplicadas nas mais diversas áreas. Esses mesmos estudos visam também obter um maior entendimento do fenômeno da interação distribuída dentro

de determinado contexto. A Figura 6 apresenta um agrupamento desses estudos de caso por área de aplicação do estudo.

Figura 6 – Mapa mental de 16 estudos de caso em DUI agrupados por área de aplicação e respectivos pesquisadores

Fonte: Próprio autor

Observa-se uma concentração maior no uso de mesas interativas e de aplicações de DUI na educação.

Observa-se também que DUI pode ser utilizada para diversas finalidades: compartilhamento de informação entre usuários, atribuir tarefas, dividir a informação pública e privada em diferentes telas e dispositivos.

2.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Todo o levantamento realizado até a finalização deste capítulo foi proveniente de uma pesquisa exploratória, conforme descrita na metodologia da pesquisa. Esta foi necessária dada a diversidade de pesquisas nas diversas áreas que auxiliam, em algum aspecto, na concepção de uma interação distribuída em múltiplos dispositivos. Ao final constatou-se um esforço da comunidade de IHC em organizar estudos dessa natureza por meio da área de DUI.

Por DUI ser uma área nova e por entender que outras áreas podem contribuir com a pesquisa, esse levantamento foi importante para obter uma maior compreensão do fenômeno da interação distribuída e estabelecer de forma mais sistemática os critérios de busca da pesquisa. Portanto, o próximo capítulo reflete um estudo que busca identificar a arquitetura de ferramentas que auxiliam na construção de aplicações para DUI.

Há, sem dúvida, pontos de intersecção entre os dois campos de pesquisa. Resumidamente, o projeto de uma interação de mídia cruzada está associada à distribuição da narrativa em múltiplos dispositivos, explorando suas diferentes capacidades e os diferentes meios de comunicação, por outro lado, a pesquisa sobre DUI tem sido orientada para a distribuição de interface gráfica de usuário.

Neste trabalho, o interesse está no aspecto comum a ambos os campos: a interação acontecendo em mais de um dispositivo, seja sequencial ou simultaneamente. Navegar de um dispositivo para outro é um aspecto relevante para a interação distribuída que pode aproveitar os estudos realizados em aplicações de mídias cruzadas.

Neste trabalho estabelece-se a base em aplicações de mídias cruzadas e interfaces distribuídas para observar essas interações.

3 ARQUITETURAS PARA DUI

Este capítulo tem como objetivo analisar as arquiteturas para DUI encontradas na literatura e extrair um conjunto de requisitos para uma arquitetura de referência para DUI.

Com este objetivo, o capítulo estrutura-se da seguinte forma. A seção 3.1 apresenta os resultados de uma pesquisa sistemática, realizada sobre a literatura, para identificar exemplos de arquiteturas que suportam a interação distribuída. Esses exemplos são analisados, nessa seção, em termos das características de distribuição da interação.

Em seguida, a partir dessa análise e de estudos precursores, apresenta-se na seção 3.2 um conjunto de requisitos para a arquitetura de referência.

As transições destacam-se como essenciais no processo de distribuição da interação. Por essa razão, a seção 3.3 dedica-se a definir e caracterizar as transições, como requisitos essenciais da arquitetura proposta.

A seção 3.4 realiza uma análise das tecnologias de suporte à arquitetura, com base em uma leitura dos resultados

É importante ressaltar que não está no escopo deste trabalho atender a todos os requisitos que serão apresentados neste capítulo. Esses requisitos vêm sendo definidos e aperfeiçoados ao longo do tempo, dentro de linha de pesquisa do Laboratório de Tecnologia de *Software* (LTS), na qual esta pesquisa se insere. A visão de conjunto, no entanto, é necessária para contextualizar e delimitar a contribuição do trabalho.

3.1 EXEMPLOS DE ARQUITETURAS PARA INTERAÇÃO DISTRIBUÍDA

Como apresentado no capítulo anterior, diversas são as áreas de pesquisa que de alguma forma promovem a ocorrência de interações distribuídas. Nesta pesquisa encontram-se ocorrências nas áreas de mídias cruzadas, interfaces distribuídas, realidade virtual e misturada, computação ubíqua, computação pervasiva, interação multimodal, múltiplas superfícies, computação em nuvem, trabalho cooperativo apoiado por computador, interfaces migratórias e *displays* públicos.

A partir do referencial conceitual obtido com a pesquisa exploratória, realizou-se revisão sistemática da literatura, que teve por objetivo buscar referências dos últimos 5 anos que refletissem o estado-da-arte da interação distribuída, obtendo novas tecnologias não identificadas na pesquisa exploratória. O protocolo da revisão sistemática encontra-se no Apêndice A. A Figura 7 apresenta o mapa conceitual das áreas de pesquisa envolvidas e das ferramentas que apoiam no desenvolvimento de aplicações, selecionadas de acordo com o filtro do protocolo de revisão sistemática.

Figura 7 – Ferramentas que auxiliam no desenvolvimento de aplicações de interação distribuída.

Fonte: Próprio autor

Quando apresentados os conceitos de DUI na Seção 2.3, mostrou-se que Blumendorf, Roscher e Albayrak (2011) observaram que as DUI são caracterizadas por serem *multi-* e *cross-device*, *multi-* e *cross-modal*, *multi-* e *cross-user* e *multi-* e *cross-application*. Essa observação de Blumendorf et al., é pertinente ao contexto desta pesquisa porque consegue abstrair características comuns às arquiteturas encontradas nas diversas áreas. Desta forma, analisaram-se as arquiteturas identificadas pela revisão sistemática quanto ao atendimento dessas características, conforme pode ser observado no Quadro 2. Entende-se nesta pesquisa que uma arquitetura que atenda a todas essas características, representa uma arquitetura completa para DUI, no que tange os conceitos de *multi* e *cross* para essas 4

dimensões. O objetivo desta análise é identificar uma arquitetura que sirva de referência.

Quadro 2 – Ferramentas que promovem alguma forma de interação distribuída classificadas de acordo com a observação de (Blumendorf, Roscher e Albayrak, 2011)

Software	MD	XD	MM	XM	MU	XU	MA	XA	Referência
C3P	●	?	○	○	○	○	●	○	(ZHANG E DETERS, 2012)
ProjectTalk	●	○	○	○	●	●	○	○	(STEINERT et al., 2009)
i*Chameleon	●	○	●	●	○	○	○	○	(LO et al., 2012)
WiFly	●	○	○	○	●	○	○	○	(VANDERHULST E TRAPPENIERS, 2012)
DireWolf	●	●	○	○	●	○	⊖	○	(KOVACHEV et al., 2013)
Ensemble	●	?	●	○	●	○	●	○	(BRANTON et al., 2013)
ReticularSpaces	●	●	○	○	●	●	●	⊖	(BARDRAM et al., 2012)
MARIA	●	○	●	○	●	○	○	○	(MANCA E PATERNÒ, 2011) (GHIANI, PATERNÒ E SANTORO, 2010)
OPEN	●	●	○	○	●	○	○	○	(NICKELSEN et al., 2010)
INDiE	●	○	○	○	○	○	○	○	(KLOMPMAKER, SCHRAGE E REIMANN, 2009)
VIGO	●	○	●	●	○	○	○	○	(KLOKMOSE E BEAUDOUIN-LAFON, 2009)
PCP	●	○	○	○	●	○	●	○	(ARDISSONO et al., 2009)
MMM	●	○	○	○	●	○	○	○	(BIER, FREEMAN E PIER, 1992)
OpenInterface	●	○	●	○	●	○	●	○	(LAWSON et al., 2009)
Mona	●	○	●	○	●	○	○	○	(BAILLIE et al., 2005)
Roam	●	●	○	○	○	○	○	○	(CHU et al., 2004)
Toolkit	●	●	○	○	●	○	○	○	(MELCHIOR et al., 2009)
DynaMo	●	○	●	●	○	○	○	○	(AVOUAC, LALANDA E NIGAY, 2012)
Shared Substance	●	●	●	●	●	●	○	○	(GJERLUFSEN et al., 2011)
PuRe Widgets	●	○	●	●	●	○	○	○	(CARDOSO E JOSÉ, 2012)
i-LAND	●	○	●	○	●	○	○	○	(STREITZ et al., 1999)
iRos	●	○	●	○	○	○	○	○	(JOHANSON, FOX E WINOGRAD, 2002)
Proxywork	●	●	○	○	○	○	○	○	(VILLANUEVA, TESORIERO E GALLUD, 2013)
iStuff	●	?	●	○	○	○	○	○	(BALLAGAS et al., 2003)

MD:multi-device; **XD:**cross-device; **MM:**multi-modality; **XM:**cross-modality; **MU:**multi-user; **XU:**cross-user; **MA:**multi-application; **XA:**cross-application.

Legenda: bolinha cheia, positivo; bolinha vazia, negativo; bolinha com traço, parcialmente; ? não identificado.

Fonte: Próprio autor

A verificação do atendimento à característica *multi-device* (MD) teve por objetivo identificar as arquiteturas que suportam múltiplos dispositivos, sendo este um dos fatores de corte da pesquisa sistemática. Algumas arquiteturas são mais focadas em determinados tipos de dispositivos, como Shared Substance e Reticular Spaces, que foram concebidas para dispositivos como monitores, *tablets* e mesas multitoque. Na análise, o que se constatou é que os dispositivos mais comumente utilizados são o *desktop*, o *tablet* e o *smartphone*.

Foram marcadas como positivas para *cross-device* (XD), aquelas arquiteturas em que se encontrou suporte para que o usuário possa trocar de dispositivo de forma dinâmica em tempo de execução para continuar interagindo com a aplicação. Aplicações que não foram marcadas tendo este suporte, como i*Chameleon, é porque relacionam os dispositivos em tempo de compilação ou na carga da aplicação por meio de arquivo de configuração.

Para *multi-modality* (MM) foi utilizado o critério de verificar se a arquitetura permite o uso de diversas modalidades de interação e para *cross-modality* (XM) se essas modalidades podem ser intercambiadas em tempo de execução para interagir com um mesmo elemento de UI.

Para *multi-user* (MU) foi utilizado o critério de verificar se a arquitetura permite a identificação de múltiplos usuários e se estes podem utilizar a aplicação concomitantemente. Para *cross-user* (XU) foi utilizado o critério de verificar se é possível transferir um elemento de UI da aplicação para outro usuário em tempo de execução.

Para *multi-application* (MA) foi utilizado o critério de verificar se a arquitetura permite que outras aplicações se integrem a arquitetura, por meio do uso de padrões abertos. DireWolf foi marcada como parcial porque restringe a integração entre *widgets*.

Já no conceito de *cross-application* (XA) verificou-se se existe a possibilidade de elementos de UI serem transferidos para outra aplicação. Apenas ReticularSpaces atendeu parcialmente este critério, porque permite a transferência para aplicações dentro da mesma plataforma. Sob o ponto de vista de aplicações desenvolvidas em outras plataformas, nenhuma delas atende esse critério.

O que se observa é que não existe uma arquitetura de referência, tampouco um *framework* conceitual ou diretrizes (*guidelines*). Apesar de todas essas

ferramentas possuírem algum mecanismo que permita alguma forma de interação distribuída, é natural que elas possuam especificidades da área de pesquisa na qual estão inseridas (DUI, pervasiva, *crossmedia*, ...), em função das aplicações a que pretendem atender. Além disso, os aspectos por elas contemplados se complementam dentro do universo da interação distribuída. Observa-se que essas ferramentas possuem requisitos comuns de arquitetura que são implementados por meio de diferentes estratégias.

A experiência do grupo de pesquisa, a pesquisa exploratória e a revisão sistemática ajudaram na formação de uma visão acerca dos requisitos de uma arquitetura que suporte a construção de cenários de interação distribuída. A próxima seção apresenta esses requisitos.

3.2 REQUISITOS PARA UMA ARQUITETURA PARA INTERAÇÃO DISTRIBUÍDA

Com o propósito de eliciação de requisitos, diversas podem ser as fontes para a análise de um domínio, cada qual com suas vantagens e desvantagens: livros, artigos, padrões, aplicações existentes, especialistas do domínio (KANG et al., 1990).

Para identificar requisitos que direcionem a definição de uma arquitetura e permitam a construção de aplicações utilizando múltiplos dispositivos, as fontes utilizadas nesta pesquisa foram principalmente da literatura, na forma de artigos científicos e livros.

Apesar de a análise experimental de ferramentas e aplicações existentes ser uma das formas mais ricas para obtenção de requisitos, ela é custosa e muitas vezes impossibilitada pela dificuldade na aquisição, configuração e execução de cenários de teste.

Como a área de pesquisa em DUI é relativamente nova, os padrões e especialistas ainda estão em processo de formação.

Em um trabalho anterior do grupo de pesquisa já foram identificados requisitos importantes para uma arquitetura de DUI e que foram atendidos por meio do *framework* X-gov (MIYAMARU, 2010), conforme se vê no Quadro 3.

Como apresentado no Capítulo 1, transição é definida neste trabalho como as pontes que permitem que o fluxo de interação migre de um dispositivo para o outro.

Na pesquisa de Miyamaru (2010) a preocupação estava em como guiar o usuário por meio de múltiplas mídias seguindo o fluxo de uma narrativa (**transição entre mídias**). O aspecto de transição estava na migração do usuário para a mídia mais adequada no ponto da narrativa em que ele se encontrava. Os requisitos do Quadro 3 refletem essa realidade.

Quadro 3 – Requisitos de um *framework* de transições entre mídias

#Req.	Requisito
	Descrição
1	Prover mecanismos para suportar transições entre canais de comunicação mais utilizados.
	O <i>framework</i> deve apresentar ao desenvolvedor um conjunto pré-definido de transições que relacione as mídias mais utilizadas: <i>websites</i> , <i>e-mail</i> , telefone, celulares, portais de voz, SMS, mensagens instantâneas, iDTV. Esse conjunto deve ajudar os desenvolvedores a implantar aplicações de interação distribuída com reduzido esforço, bem como adaptar mídias que venham a se integrar a ele.
2	Prover interfaces reutilizáveis de componentes de transição que ocultam detalhes da implementação, enquanto permitem ao <i>designer</i> customizar a interface do usuário.
	Ocultar detalhes tecnológicos acelera a curva de aprendizado dos desenvolvedores. A customização da camada de apresentação deve facilitar a integração das transições no projeto da narrativa.
3	Prover recursos para automatizar os mecanismos de transição tanto quanto possível.
	Transições fluidas/integradas (<i>seamless</i>) reduzem o esforço na definição de parâmetros e controle dos dispositivos.
4	Apresentar dados relacionados à transição quando solicitados pelo usuário.
	URLs, endereços, números de telefone, endereços de <i>e-mail</i> são definidos como dados relacionados à aplicação e devem estar ocultos para os usuários em uma transição <i>seamless</i> . Contudo, havendo a necessidade de obter esses dados para a realização de

	uma transição manual, o <i>framework</i> deve prover recursos para informar ao usuário dados relativos à transição.
5	Prover recursos para informar o usuário de possíveis erros que impossibilitem os mecanismos de transição automatizados de operar corretamente. Retornos para o usuário devem ser fornecidos pelo <i>framework</i> com vistas a ajudar o usuário a controlar o fluxo da interação.
6	Prover os usuários com recursos para desfazer uma transição. Usuários devem ser capazes de voltar passos no fluxo da narrativa.
7	Ser flexível para suportar a adição de novos tipos de transição. Conforme o aparecimento de novos recursos de comunicação, estes devem poder ser incorporados no <i>framework</i> , sem comprometer a solução existente.
8	Manter registro da navegação do usuário através dos canais. O gerenciador da aplicação deve manter histórico da navegação do usuário. Ele ou ela devem poder descobrir quantos usuários foram expostos a uma <i>call-to-action</i> (uma chamada para agir); quantos tentaram completar a transição e quantos foram bem sucedidos. Um dos benefícios do <i>framework</i> é a habilidade de acompanhar seus usuários. Uma métrica importante para aplicações distribuídas é a taxa de conversão, que é, o número de usuários que, tendo sido expostos a um <i>call-to-action</i> , conseguiram completar a migração para outro dispositivo. Essa métrica pode ser utilizada em ambientes comerciais para verificar a efetividade da abordagem de mídia cruzada.

Fonte: (MIYAMARU, 2010)

A esses, em função do aprofundamento da pesquisa, acrescentam-se os do Quadro 4. Os requisitos complementares foram elicitados observando as novas demandas de interação distribuída. Aspectos de colaboração e de multimodalidade estão presentes e dois novos tipos de transição foram identificados (**transição entre usuários e transição de distribuição da interação**).

Para esclarecer os requisitos, usam-se neste texto a descrição de cenários de uso. Nesses cenários, para representar a comunicação entre dois

usuários A e B, optou-se por utilizar os termos Alice e Bob⁵, respectivamente. Esses são nomes comumente usados para representar personagens arquetípicos em domínios como a criptografia e física.

Para representar os tipos dos dispositivos TV, *tablet*, PC e *smartphone* são utilizados os acrônimos TV, TB, PC e SP, respectivamente. O asterisco (*) pode ser utilizado tanto para o dispositivo quanto para o usuário. Se utilizado para o dispositivo indica que tanto faz o dispositivo utilizado, pode ser qualquer um. Se utilizado para o usuário indica que a transição pode ocorrer para múltiplos usuários ou um usuário desconhecido. O símbolo → é utilizado para denotar a ocorrência da transição. Portanto, ao utilizar as notações:

- Alice_{PC}, entenda-se que se trata do PC do usuário A ou o usuário A realizando algo em seu PC;
- Alice_{PC} → Bob*, entende-se que está ocorrendo uma transição da interação de um dispositivo PC do usuário A para um dispositivo qualquer do usuário B;
- Alice_{PC} → *_{TB}, entende-se que está ocorrendo uma transição da interação de um dispositivo do usuário A do tipo PC para múltiplos usuários em seus respectivos *tablets*;

Quadro 4 – Requisitos complementares aos do Quadro 3

#Req.	Requisito
	Descrição
9	Reconhecer seus usuários.
	Manter registro da identidade dos usuários permite registrar preferências e fornecer recursos de customização.
10	Manter registro dos dispositivos do usuário.
	Manter o registro de quais dispositivos cada usuário utiliza permite identificar quais são os dispositivos de um determinado usuário. Dessa forma é possível filtrar dispositivos pelo seu uso.
11	Permitir que interações em um dispositivo sejam delegadas a usuários em outros dispositivos.
	Este requisito trata da transição de distribuição da interação . Para

⁵ http://pt.wikipedia.org/wiki/Alice_e_Bob

	<p>atividades colaborativas, faz-se necessário que seja possível transferir a interação em um dispositivo para outro usuário em outro dispositivo. Este requisito deve ocorrer no formato mais atômico possível, para que várias interações possam ocorrer concorrentemente em um mesmo dispositivo.</p> <p>Por exemplo, um cenário $Alice_{PC} \rightarrow *$, no qual Alice em seu PC, deseja delegar a interação para com um componente GUI de controle (caixas de texto, de seleção, ...) a diversos outros usuários independente do dispositivo que estejam utilizando (PC, <i>smartphone</i>, TV, ...).</p>
12	<p>Permitir que recursos do dispositivo sejam configurados para permitir ou não o acesso concorrente.</p> <p>Recursos que exigem, por exemplo, a interação do usuário, podem não ser realizados de forma concorrente. Outros dispositivos que estejam interessados em solicitar uma interação desse mesmo usuário devem aguardar pelo processo que já se encontra em execução.</p>
13	<p>Permitir que mais de um dispositivo seja utilizado pelo mesmo usuário de forma simultânea e sincronizada.</p> <p>Recursos dos dispositivos em rede devem poder ser relacionados de forma a manter uma comunicação contínua e mais imediatista possível. Por exemplo, num cenário $Alice_{PC} \rightarrow Bob_{PC}$. Alice utilizando o mouse em seu PC, causa efeito imediato no cursor de mouse de Bob em seu PC. Em outro exemplo, $Alice_{SP} \rightarrow Alice_{TV}$. Alice em seu <i>smartphone</i>, causa efeito imediato na mudança de volume em sua TV.</p>
14	<p>Permitir que mais de uma modalidade de interação seja utilizada para interagir com um determinado recurso do dispositivo.</p> <p>Entrada de dados devem poder ser convertidas para um formato que a interface do recurso do dispositivo possa consumir.</p> <p>Por exemplo, o cenário $Alice_{SP} \rightarrow Alice_{TV}$. Alice em seu <i>smartphone</i> deseja utilizar o sensor de movimento para realizar a mudança de canal em sua TV e a tela sensível ao toque para controlar o volume. Este requisito possui dependência com o requisito 13.</p>
15	<p>Permitir a restrição de acesso de um ou mais dispositivos a um determinado usuário ou contexto.</p> <p>Este requisito visa impedir que transições indesejadas ocorram,</p>

	<p>possibilitando ao usuário selecionar quais dispositivos podem participar da interação distribuída.</p> <p>Por exemplo, um cenário em que Alice está utilizando o microfone de seu dispositivo. Alice não pode receber transições de interação via modalidade de áudio enquanto não encerrar a atividade que já estava em execução.</p>
16	<p>Converter conteúdos não suportados por um determinado dispositivo.</p> <p>A arquitetura deve fornecer recursos para identificar os formatos de conteúdo (áudio, vídeo, texto, ...) suportados pelo dispositivo e convertê-los quando necessário.</p> <p>Por exemplo, Alice_{SP} deseja enviar uma imagem para Bob_{TV}, mesmo que a TV de Bob não tenha suporte ao formato de imagem enviado por Alice_{SP}.</p>
17	<p>Permitir que metadados possam ser agregados ao conjunto de informações do dispositivo.</p> <p>Deve ser possível associar novos metadados aos dispositivos. Este requisito pode auxiliar na realização de interações mais contextualizadas e personalizadas.</p> <p>Para que isso que seja possível, se faz necessário não apenas identificar os dispositivos do usuário, mas também agregar e extrair metadados dos dispositivos com vistas a registrar e obter informações da capacidade do mesmo. Deve ser possível também registrar dados contextuais do dispositivo que porventura o mesmo não ofereça, por exemplo, a localização geográfica de uma TV que não possui GPS, permitindo direcionar o conteúdo para a mídia mais adequada.</p> <p>Por exemplo, Alice_{SP} → *_{TV}, Alice em seu <i>smartphone</i>, pode querer enviar uma imagem para uma TV qualquer, que esteja próxima dela. Nesse caso, este requisito auxilia no filtro dos dispositivos encontrados de acordo com a localização de Alice.</p>
18	<p>A arquitetura deve permitir que dispositivos sejam agrupados ou classificados conforme interesse.</p> <p>É importante saber que dispositivo pertence a que usuário, se ele é compartilhado e em que contexto é utilizado. Alice e Bob podem compartilhar o mesmo, sendo que Alice o utiliza mais no trabalho enquanto Bob o utiliza em casa. Este requisito pode auxiliar na realização de</p>

	interações mais contextualizadas e personalizadas.
19	Permitir a descoberta de dispositivos de forma <i>ad hoc</i>.⁶
	Dispositivos devem poder ser descobertos e inseridos no processo interativo de forma <i>plug and play</i> , sem muitas configurações. O uso dos dispositivos deve ocorrer da forma mais simples possível para o usuário final.
20	Facilitar a realização de testes de comunicação com os dispositivos.
	Desenvolvedores devem ser capazes de realizar testes no dispositivo de forma facilitada, sem a necessidade de programar chamadas aos recursos. Uma ferramenta que permita visualizar os dispositivos na rede e interagir com seus recursos deve ser fornecida. Uma interface gráfica padrão por tipo de dispositivo deve ser associada ao dispositivo de forma a testar seus recursos. Por exemplo, por mais variadas que sejam as funcionalidades das TVs, todas elas possuem funcionalidades comuns (volume, canal, seleção de entrada, etc). A ferramenta deve possibilitar a visualização e identificação da TV, bem como seus metadados. Ao selecionar a TV deve ser possível fazer chamadas às suas funcionalidades.
21	Permitir que novas aplicações, em outras plataformas, possam ser facilmente integradas.
	Padrões abertos de comunicação devem ser utilizados e os dispositivos devem expor seus recursos para acesso externo.
22	Permitir que a comunicação entre dispositivos possa ocorrer de forma segura.
	Dispositivos devem se comunicar de forma que a integridade e a confidencialidade dos dados trocados sejam mantidas. Os dados trafegados entre os diversos dispositivos não podem ser alterados entre o envio e recebimento, bem como só podem ser lidos pelo devido destinatário. A autenticidade do remetente também deve ser garantida.

Fonte: (PRETI, MIYAMARU E FILGUEIRAS, 2014)

⁶ Este requisito faz a arquitetura ser classificada como sendo de quarta geração em mídias cruzadas.

3.3 AS TRANSIÇÕES NA INTERAÇÃO DISTRIBUÍDA

Nesta pesquisa, verifica-se a existência de dois novos tipos de transição e novas demandas que merecem discussão, tanto arquitetural, quanto reflexivas acerca dos impactos dessas transições em vista do que é proposto. Tendo em vista, a experiência do grupo de pesquisa e, de que a transição é o aspecto mais forte do conceito de *cross*, optou-se por verificar os requisitos relacionados a transição de distribuição da interação. Esses requisitos surgem com vistas a disponibilizar serviços de acesso aos recursos e aos elementos de interação dos diferentes dispositivos, pensando-se em um modelo de interação em que entrada e saída não mais se acoplem 1:1.

Fardoun, López e Villanueva (2011) apontam que uma das principais preocupações em DUI é o complexo e dinâmico gerenciamento de interação do usuário em diferentes tipos de dispositivos.

Por meio da pesquisa realizada na literatura, da reflexão acerca dos conceitos já enunciados e das ferramentas apresentadas, foi possível identificar três tipos de transição da interação.

Uma é a **transição entre mídias**, que representa a navegação entre meios de comunicação, na qual o usuário é direcionado de forma a alcançar as etapas necessárias em busca do seu objetivo. Por exemplo, um cenário no qual Alice deseja dar continuidade na interação, movendo sua tarefa Alice_{TV} → Alice_{SP}.

Outra é a **transição entre usuários**, que representa a mudança de usuário para a realização de uma tarefa já iniciada, de forma a aproveitar as etapas já realizadas na busca de um objetivo. É o caso de tarefas de natureza colaborativa. Por exemplo, Alice deseja que Bob conclua o preenchimento de um formulário, portanto a interação migra de Alice, em seu smartphone (Alice_{SP}), para Bob, usando o que quer que ele esteja (Bob*). Neste caso em particular, este tipo de transição implica também em uma **transição entre mídias**, cabendo à arquitetura fornecer meios de migrar a sessão de Alice para Bob, isto é, reconhecer qual a mídia usada por Bob (Alice não precisa saber qual é).

A terceira é a **transição de distribuição da interação**, quando um usuário deseja que outros usuários interajam com os componentes de interface de usuário (UI), enquanto o primeiro mantém o gerenciamento da tarefa. Por exemplo, um usuário requisitando a outro usuário em outro dispositivo para completar uma

entrada de texto ou selecionar um item de uma lista ou mesmo fornecer uma imagem, vídeo ou áudio.

Acredita-se que o conceito de transições está fortemente inserido na definição da interação distribuída. As transições entre mídias e de distribuição da interação estão inseridas nas dimensões de entrada, saída, plataforma, espaço e tempo presentes na definição de DUI de Elmqvist. Para navegar entre mídias ou distribuir a interação para outros dispositivos todas essas dimensões são consideradas.

Um olhar mais próximo sobre as transições, como elemento fundamental da interação distribuída, deve considerar dois aspectos. O primeiro é o do design da interação, o qual está focado nas habilidades comunicativas da transição e em seus aspectos de usabilidade: o que faz com que uma transição ocorra satisfatoriamente, o quão perceptível, persuasiva e fluida é a navegação de um dispositivo para o outro, o quão comunicativos são os símbolos que expressam as ações de transição e seus estados, quais alternativas fornecem uma melhor experiência para o usuário. Outro aspecto é a perspectiva da engenharia, a qual está preocupada com os mecanismos computacionais para implementar as transições. Este trabalho tem como foco principal a **perspectiva da engenharia**, embora se reconheça a importância dos aspectos de **design de interação**.

Não foi identificada, até data de publicação desta pesquisa, uma arquitetura de referência para essa natureza de aplicação. A próxima seção apresenta um estudo do suporte tecnológico das ferramentas de DUI do Quadro 2.

3.4 SUPORTE TECNOLÓGICO

Após elucidação dos conceitos relacionados à interação distribuída, de suas aplicações e de definir requisitos de uma arquitetura, faz-se necessário a identificação de um suporte tecnológico que auxilie no cumprimento desses requisitos.

Com vistas a verificar se as ferramentas do Quadro 2 possuem um padrão quanto ao uso de tecnologias e de arquitetura distribuída, foi realizada uma segunda análise sobre os artigos lá referenciados visando identificar características tecnológicas que estão sendo utilizadas pelos respectivos autores. O objetivo desse levantamento é verificar qual a tecnologia e o modelo de comunicação distribuída

utilizado (ponto a ponto ou cliente/servidor) para construção de soluções computacionais que auxiliam na implementação de aplicações que promovem interações distribuídas.

O Quadro 5 apresenta o resultado desta análise, identificando as seguintes características:

- bases tecnológicas,
- tipo de *software*,
- arquitetura de rede e
- aplicação.

As bases tecnológicas identificam as tecnologias utilizadas por essas ferramentas, podendo apresentar linguagens, *frameworks* e *middlewares*.

O tipo de *software* representa o modelo de uso dessa ferramenta, se como um *framework*, *middleware*, linguagem própria ou como aplicação específica, pronta para implantação e uso em domínio específico.

Sob arquitetura de rede, verifica-se qual a estrutura da aplicação distribuída utilizada, se ponto a ponto (P2P) ou cliente/servidor (C/S). Na arquitetura de rede P2P os extremos podem atuar como clientes ou servidores e têm as mesmas responsabilidades ao compartilhar recursos. Na configuração C/S um servidor pode ser alterado sem prejudicar o funcionamento dos clientes, enquanto que nos sistemas P2P é preciso atualizar cada ponto da rede, no entanto, as redes C/S podem sofrer congestionamento e sobrecarregar o servidor, enquanto os esforços nas solicitações em P2P melhoram com a quantidade de participantes. Os usuários em uma configuração C/S também podem sofrer quando um servidor vital falha, enquanto a distribuição de tarefas em P2P assegura que a rede funcionará enquanto não estiver no limite da capacidade.

O Quadro 5 apresenta a compilação completa desse levantamento.

Quadro 5 – Ferramentas que promovem alguma forma de interação distribuída, bases tecnológicas utilizadas, arquitetura e contexto de aplicação

Software	Bases Tecnológicas	Tipo	Arq.	Aplicação
C3P	RESTful WebSockets	<i>Middleware</i>	C/S	Diversas
ProjectTalk	Squeak Smalltalk Morphic GUI <i>Framework</i>	Aplicação Específica	C/S	Atividades de equipes em <i>eXtreme Programming</i>
i*Chameleon	SOAP	<i>WS Framework</i>	C/S	Diversas
WiFly	WiFi Hotspot	<i>Framework</i>	C/S	Interação social com dispositivos públicos
DireWolf	RESTful Services ROLE SDK (Shindig) XMPP Messages HTML+CSS+JS	<i>Framework</i>	C/S	Distribuir e sincronizar <i>web widgets</i>
Ensemble	Apple Bonjour OSC Messages ContentServer YAML	<i>Framework</i>	C/S	Aplicações <i>desktop</i> distribuídas com swing
ReticularSpaces	Apple Bonjour Aexo Framework Java RESTful Adobe Flex AIR	<i>Framework</i> (GUI) <i>Middleware</i> (Comunicação)	P2P	<i>Smart spaces</i> com múltiplos <i>displays</i>
MARIA	Linguagem XML	Linguagem	C/S	Diversas
OPEN	OSGi XML-RPC	<i>Middleware</i>	C/S	Jogos Business domains
INDiE	VRPN .Net	<i>Framework</i>	C/S	Realidade virtual e misturada
VIGO	Python-Twisted (Server) Cocoa (Client) PyGTK (Client)	<i>Framework</i>	C/S	Múltiplas superfícies
PCP	Java REST/SOAP GigaSpaces	<i>Middleware</i>	C/S	Integração de serviços <i>Web Web Desktop</i>
MMM	Cedar language SunOS	<i>Framework</i>	C/S	Editor de texto
OpenInterface	OpenInterface Kernel C++	<i>Middleware</i>	C/S	Diversas
Mona	SOAP HTML/XML VoiceXML	Linguagem	C/S	Quiz <i>games</i>
Roam	Java Java RMI J2ME	<i>Framework</i>	C/S	Diversas
Toolkit	Tcl/Tk Mozart Oz Programming Language	<i>Framework</i>	P2P	Diversas
DynaMo	OSGi ROSE iPOJO CILIA	<i>Framework</i>	C/S	Diversas
Shared Substance	Substance	<i>Middleware</i>	C/S	Múltiplas superfícies
PuRe Widgets	Google App Engine GWT REST	<i>Framework</i>	C/S	<i>Displays</i> públicos
iLand	Smalltalk	<i>Framework</i>	C/S	Área de trabalho interativa
iRos	XML IBM Tspaces	meta-OS	C/S	Área de trabalho interativa
Proxywork	-	<i>Framework</i>	C/S	Distribuir DIVs HTML entre dispositivos
iStuff	iROS Java	<i>Middleware</i>	C/S	Diversas

Fonte: Próprio autor

Analisando-se as **bases tecnológicas** é possível verificar grande heterogeneidade em relação às linguagem de programação, *frameworks* e *middlewares*.

Entende-se que a interoperabilidade natural dos *web services* é importante para que aplicações possam trocar dados e compartilhar ações de forma desacoplada e para compor novas estruturas.

Entende-se que alternativas como linguagem e meta-OS são fortemente acopladas a uma plataforma não sendo indicadas em um ambiente com múltiplas plataformas. *Frameworks* e *middlewares* são mais indicados para um contexto distribuído multiplataforma, mas normalmente *middlewares* tratam de aspectos de mais baixo nível de comunicação, na qual a interface das operações é bem conhecida sendo desenvolvido em camadas. Já *frameworks* normalmente são modulares e agregam uma série de recursos e padrões de projeto que podem ser estendidos e reutilizados.

Ao analisar **arquitetura de rede** foi possível constatar que 91,6% da soluções pesquisadas adota um modelo de comunicação cliente/servidor. Entende-se que esse perfil seja decorrente dos recursos limitados de processamento e memória de parte dos dispositivos que devem ser integrados, como TVs e sensores. Essa limitação faz com que seja necessário buscar em outros dispositivos recursos adicionais de memória e processamento, o que leva a centralização. Acredita-se nesta pesquisa que essa limitação já está sendo superada, já que dispositivos como celular e TV já possuem grande poder de processamento.

O estilo arquitetural da aplicação define como os diferentes componentes são organizados. Extensibilidade e flexibilidade são influenciadas pelo estilo arquitetural, classificados nesta pesquisa como modular e em camadas. Essa característica foi verificada e não inserida no quadro, visto que, não apresentava uma tendência e sim um certo equilíbrio entre soluções de mais baixo nível organizadas em camadas e as de mais alto nível organizadas de forma modular.

No modelo em camadas tem-se como vantagens o favorecimento do desenvolvimento incremental, as camadas podem ser substituídas por equivalentes (requer interfaces estáveis), mudanças em uma camada só impactam na camada superior e camadas superiores podem ser independentes de plataforma/*hardware*. Todavia, estruturar um sistema em camadas não é trivial, pode ser difícil identificar

quais os serviços elementares de cada camada, além de que muitas camadas podem comprometer o desempenho do sistema. Utilizar somente um modelo arquitetural em camadas em partes manuteníveis para grandes sistemas pode não ser suficiente. A arquitetura modular pode estender a arquitetura introduzindo outra dimensão para separar os sistemas de *software*.

O que se observa é que a arquitetura indicada para um ambiente heterogêneo composto por múltiplos dispositivos está na integração e compartilhamento dos dados trafegados entre si. Uma abordagem que não considera a troca coordenada dos dados entre as plataformas, ou seja, que não permite os dados trafegarem entre um canal e outro, cria uma arquitetura de baixa flexibilidade.

3.5 DEFINIÇÃO DA ARQUITETURA PROPOSTA

Visando atender aos requisitos do Quadro 3 e do Quadro 4, e com base na pesquisa apresentada na seção anterior, propõe-se uma arquitetura com as seguintes características:

- Orientada a serviços;
- Estruturada sobre a pilha DPWS;
- Organizada como um *Web Service Framework*;
- Utilizando uma arquitetura de rede híbrida;

As seções seguintes justificam a escolha dessas características.

3.5.1 A escolha de SOA

SOA é um estilo arquitetural para construção de aplicações, fracamente acopladas, que permitem composição. Por exemplo, a construção de um *datacenter* utilizando princípios SOA, que remete a nuvem, uma infraestrutura orientada a serviço.

Nesta pesquisa adota-se o conceito de serviços como componentes, sendo considerados blocos de construção independentes, os quais coletivamente representam um ambiente de aplicação. No entanto, diferente de componentes tradicionais, os componentes de serviços têm algumas características únicas que lhes permitem participar como parte da uma arquitetura orientada a serviços.

Uma destas características é que um serviço tem completa autonomia em relação a outros serviços. Cada serviço é responsável por seu próprio domínio, o que tipicamente significa limitar seu alcance para uma função de negócio específica (ou um grupo de funções relacionadas) (ERL, 2009).

Outra característica que distingue os componentes na orientação a serviços é a maneira pela qual a separação de preocupações é realizada e o modo como ela modela as unidades individuais de lógica. ERL (2009) apresenta 8 princípios que norteiam a concepção de serviços:

1. contrato de serviço padronizado: os serviços devem expressar seu propósito e suas capacidades por meio de um contrato de serviços. Importante para se projetar a interface técnica pública de um serviço;
2. baixo acoplamento de serviço: defende a criação de um tipo específico de relacionamento dentro e fora dos limites do serviço, com ênfase constante em reduzir as dependências entre o contrato de serviço, sua implementação e os consumidores do serviço;
3. abstração de serviço: enfatiza a necessidade de ocultar o maior número possível de detalhes subjacentes de um serviço;
4. capacidade de reuso de serviço: enfatiza o posicionamento de serviços como recursos corporativos com contextos funcionais agnósticos;
5. autonomia de serviço: a lógica do serviço precisa ter um grau significativo de controle sobre seu ambiente e recursos, de forma a realizarem suas capacidades de forma consistente e confiante;
6. independência de estado do serviço: os serviços devem manter informações de estado apenas quando estas forem necessárias, visto que, o gerenciamento de excessivas informações de estado pode comprometer a disponibilidade de um serviço e reduzir o potencial de escalabilidade;
7. visibilidade do serviço: o serviço deve ser facilmente identificado e entendido quando houver a oportunidade de reuso;
8. composição de serviços: serviços devem ser capazes de participar como membros de composição eficazes, mesmo que precisem ser imediatamente incluídos em uma composição.

Este enfoque de projeto resulta na criação de unidades isoladas de funcionalidades de negócio ligadas fracamente entre si. Isto é possível por causa da definição de uma estrutura padrão de comunicação. Devido à independência que esses serviços desfrutam dentro desta estrutura, a lógica de programação que encapsulam não tem necessidade de obedecer a nenhuma outra plataforma ou conjunto de tecnologias.

Em 2008, segundo Noll, Alam e Chowdhury (2008) a limitação dos recursos para dispositivos móveis (variação da qualidade do sinal, custo da transmissão, capacidade de processamento, protocolos de segurança e comunicação utilizando protocolos WS) era um empecilho para suportar a segunda geração de padrões para *Web Services*, i.e. WS-*⁷, mas isso vem se modificando ao longo dos anos dado o aumento da capacidade de processamento e das larguras de banda dos canais de comunicação.

O uso de SOA na arquitetura proposta é adequado porque:

1. Existe tendência de se aplicar SOA, em particular padrões de *Web Services* (SOAP, WSDL, DPWS, etc.) diretamente nos dispositivos (KARNOUSKOS et al., 2009), o que viabiliza uma arquitetura de rede P2P;
2. Existe a perspectiva de integração de aplicações, por exemplo, informações distribuídas podem atuar em colaboração com sistemas corporativos (BRONSTED, HANSEN E INGSTRUP, 2010), o que viabiliza o *cross-application*;
3. Padrões utilizados no ambiente doméstico estão tentando incorporar a tecnologia de *Web Services*, como a UPnP 2.0⁸ e a Microsoft em sua plataforma de computação invisível⁹, o que viabiliza o *cross-device*;
4. Existe uma convergência dos sistemas móveis e estacionários em que aplicações se tornam um recurso livre da restrição de dispositivo e de localidade (MARC MCLOUGHLIN, 2009).

⁷ Forma utilizada no mercado para representar coletivamente todas as especificações de *Web services*. http://en.wikipedia.org/wiki/List_of_Web_service_specifications

⁸ UPnP (*Universal Plug and Play*) é um sistema distribuído, de arquitetura de rede aberta, no qual os dispositivos são conectados diretamente uns com os outros em casa, escritório e espaços públicos. Vide [http:// en.wikipedia.org/wiki/Universal_Plug_and_Play](http://en.wikipedia.org/wiki/Universal_Plug_and_Play)

⁹ A plataforma da Microsoft de computação invisível tenta integrar o mundo da computação para pequenos dispositivos por meio da implementação de *Web Services* em *chips* de dispositivos.

Neste trabalho supõe-se que as tecnologias de *Web services* serão cada vez mais disseminadas nas plataformas de dispositivos heterogêneos sobre o protocolo Internet. Apesar disso, o uso típico de *Web Services* não é adequado, visto que, deve ser levado em consideração os recursos limitados de processamento e largura de banda de dispositivos de menor capacidade.

Outro problema reside no WSDL (*Web Service Description Language*). Esta representa a interface do serviço, fornece a descrição básica e especificações técnicas para as mensagens que são trocadas. No entanto, a descrição fornecida não vai além de simples interações entre o cliente e o serviço. Essas interações podem ser *stateless*, síncrona ou assíncrona. Essas relações são inadequadas para expressar, modelar e descrever composições complexas de múltiplos serviços. Geralmente várias mensagens são trocadas em uma ordem bem definida. Em tais composições complexas, mensagens síncronas e assíncronas podem ser combinadas, e as interações são geralmente de longa duração, muitas vezes envolvendo informações sobre o estado que tem que ser preservadas.

Um aspecto importante é também a capacidade de descrever como lidar com erros e outras situações excepcionais. Dadas as limitações da WSDL, faz-se necessário a criação de um mecanismo para descrever a composição de serviços *Web* em processos mais complexos, surgindo assim as extensões de protocolos para *Web Services* (WS-*).

3.5.2 Modelo de comunicação

A integração entre dispositivos em uma arquitetura orientada a serviço, basicamente, pode ocorrer de duas maneiras: de forma indireta (i.e., cliente/servidor) ou direta (i.e., P2P). A Figura 8 e a Figura 9 apresentam, respectivamente, os dois modelos de comunicação.

Figura 8 – Modelo de comunicação indireta

Fonte: Próprio autor

Figura 9 – Modelo de comunicação direta

Fonte: Próprio autor

Para a arquitetura proposta nesta pesquisa optou-se por um modelo de comunicação direta, tendo em vista que:

1. fornece ao dispositivo a capacidade de se conectar espontaneamente com outro para a realização de uma atividade;
2. promove uma comunicação facilitada, estilo *plug and play*, sem muitas configurações;
3. não há hierarquia entre os nós de comunicação e nem um ponto de falha comprometedor, cada dispositivo pode ser um provedor ou consumidor, não havendo dependência comum na arquitetura a um determinado dispositivo.

Este modelo arquitetural implica que todo consumidor conhece cada provedor. A descoberta de serviço acontece a todo instante, devido à constante mudança dos dispositivos que entram e saem da rede.

Por causa desse comportamento problemas de consumo de energia podem ocorrer, dada a necessidade de constante realização de buscas. A descoberta de dispositivos e serviços na arquitetura proposta utiliza uma solução que minimiza esse consumo.

Dada a variedade de especificações para integração de dispositivos, a pesquisa de Bohn, Bobek e Golatowski (2006) apresenta uma comparação das tecnologias mais referenciadas. O resultado dessa pesquisa pode ser observado no Quadro 6.

Quadro 6 - Comparação de tecnologias para integração de dispositivos

	OSGi	HANA	JINI	UPnP	WS	DPWS
Plug and Play	○	●	●	●	○	●
Suporte de dispositivos	●	●	●	●	○	●
Independência de ling. de programação	○	●	○	●	●	●
Independência de rede	○	○	●	●	●	●
Larga escalabilidade	●	○	●	○	●	●
Segurança	●	●	●	○	●	●
Grande aceitação do mercado	●	○	●	●	●	●

Legenda: bolinha cheia, atende; bolinha vazia, não atende.

Fonte: (BOHN, BOBEK E GOLATOWSKI, 2006)

O uso de SOA nos dispositivos está se tornando comum e remete ao conceito de *Device as a Service* (DaaS). Esse conceito é tratado em diversas pesquisas, como as de James, Mensch e Smit (2005), De Deugd et al. (2006), Bohn, Bobek e Golatowski (2006) e Andreas Muller et al. (2009), sendo que diversas delas apontam a especificação DPWS como promissora.

Membros do NESSI *User Service Interaction Working Group* (NESSI-USIWG) têm uma posição a respeito da próxima geração de tecnologias de interface de usuário para a Internet do futuro. Usuários terão aplicações realizando as tarefas do dia a dia compostas por *Web services*, mas a tecnologia atual não é endereçada o suficiente ainda para os usuários e os processos do dia a dia. Obviamente, SOA, como concebida originalmente, representa uma arquitetura focada fundamentalmente em um contexto *Business-to-Business* (B2B).

3.5.3 DPWS

Dada a perspectiva futura de dispositivos oferecendo recursos por meio de *Web Services* foi realizado um estudo acerca da especificação *Devices Profile for Web Services* (DPWS) por se mostrar promissora, com o objetivo de verificar se a mesma pode ser utilizada para atender os requisitos da arquitetura proposta. Apesar da especificação DPWS estar bem descrita em Zeeb et al. (2007), a próxima seção apresenta características dessa especificação que estão relacionadas ao contexto da pesquisa.

A pesquisa de Bohn, Bobek e Golatowski (2006) apresenta a especificação *Devices Profile for Web Services* (DPWS) como promissora e melhor pontuada frente a outras mais comumente utilizadas. É interessante observar que apesar desta opinião, nenhuma das ferramentas verificadas em nossa revisão sistemática fez uso da mesma.

As ferramentas analisadas que possuem uma arquitetura orientada a serviço utilizam *Web Services*, mas no formato cliente/servidor. Ao optar por uma arquitetura de rede P2P faz-se necessário delimitar um conjunto de especificações de *Web Services* passíveis de execução em diversos tipos de dispositivos. Por isso, a opção pela DPWS mostra-se interessante. Outras ferramentas construídas sobre OSGi seguem o princípio SOA, mas é restrita a plataforma Java e também segue o modelo cliente/servidor. UPnP seria uma alternativa SOA P2P para dispositivos móveis e embarcados, mas possui restrição de rede local.

Aplicações imbricam-se buscando novos modelos de interação. Esse novo paradigma requer composição de serviços no formato *ad hoc*, o que justifica o uso da DPWS.

Por esses motivos e pelo conjunto de serviços ofertados, que serão apresentados em seguida, é que optou-se pela DPWS.

A DPWS é uma especificação da OASIS para comunicação entre dispositivos, essa especificação é construída sobre uma pilha de protocolos. A pilha de protocolos DPWS é formada por (OASIS WEB SERVICES DISCOVERY AND WEB SERVICES DEVICES PROFILE, 2009):

- **WSDL**: para descrever de forma abstrata a interface dos serviços;
- **XML Schema**: para definir o formato dos dados utilizados na construção das mensagens trocadas entre os serviços;
- **SOAP**: o protocolo para transmitir as mensagens relacionadas aos serviços;
- **WS-Addressing**: permite concentrar todas as informações de endereçamento das mensagens no cabeçalho da mensagem SOAP, tornando o transporte da mensagem independente do protocolo de transporte utilizado (HTTP, SMTP, TCP, UDP, ...);
- **WS-Policy**: utilizado para definir políticas associadas ao serviço, complementando a descrição WSDL do mesmo;

- **WS-MetadataExchange**: permite a descoberta de forma dinâmica dos metadados de um serviço, fornecendo um mecanismo de reflexão;
- **WS-Security**: conjunto opcional de mecanismos para garantir integridade, confidencialidade e autenticação das mensagens trocadas;
- **WS-Discovery**: é um protocolo *plug-and-play* para descoberta de recursos disponíveis na rede. Define um protocolo *multicast* para busca e localização de dispositivos;
- **WS-Eventing**: define um protocolo que permite receber e enviar mensagens de notificação de eventos.

A Figura 10 permite visualizar melhor a organização em camadas dessa pilha de protocolos:

Figura 10 – Pilha de protocolos da especificação DPWS.

Fonte: (OASIS WEB SERVICES DISCOVERY AND WEB SERVICES DEVICES PROFILE, 2009).

A especificação DPWS define uma arquitetura na qual dispositivos executam dois tipos de serviços: *hosting services* e *hosted services*. *Hosting services* estão diretamente ligados ao dispositivo e têm um papel importante no processo de descoberta de dispositivos. *Hosted services* são mais funcionais e dependem do dispositivo no qual estão inseridos (JAMMES, MENSCH E SMIT, 2005).

Além desses serviços, a DPWS especifica um conjunto de serviços embutidos (DPWS, 2009):

- *discovery services*: utilizado pelo dispositivo conectado a rede para dar conhecimento de si mesmo a outros e descobrir outros dispositivos;
- *metadata exchange services*: provê acesso dinâmico aos serviços publicados e respectivos metadados, como o WSDL ou definição do XML Schema;
- *publish/subscribe eventing services*: permite a outros dispositivos registrarem-se para tomar conhecimento dos eventos produzidos por outro serviço.

3.5.4 WS-Discovery (WS-DISCOVERY, 2009)

A **descoberta de serviços** é necessária para que as aplicações possam dar/ter conhecimento dos serviços ofertados no meio. Esta característica é provida pela especificação *WS-Discovery*.

Essa especificação define um protocolo de descoberta para localizar serviços. Em modo de operação *ad hoc*, mensagens *probe* são enviadas para um grupo *multicast* e serviços compatíveis retornam uma resposta direta para o requerente. Para conseguir escalar para um grande número de *endpoints*¹⁰ e para estender o alcance do protocolo, a especificação define um modo de operação gerenciado e forma de supressão do comportamento de *multicast* se um *proxy* de descoberta estiver disponível na rede. Para minimizar a necessidade de *polling*¹¹, serviços que desejam ser descobertos podem enviar uma mensagem de anúncio quando entram e deixam a rede.

A especificação *WS-Discovery* permite:

- a descoberta de serviços em redes *ad hoc* com um mínimo de serviços de rede (não é necessário um DNS ou um diretório de serviços);
- reduzir o tráfego da rede e permite a descoberta de serviços em redes gerenciadas;
- mudar em tempo de execução entre o modo *ad hoc* e redes gerenciadas;

¹⁰ Um *endpoint* representa uma operação já associada de forma concreta a um protocolo de rede e a um formato de mensagem, visto que, no WSDL as operações e mensagens são descritas de forma independente do formato da mensagem ou do protocolo de rede utilizado para estabelecer a comunicação (WSDL, 2001).

¹¹ [http://en.wikipedia.org/wiki/Polling_\(computer_science\)](http://en.wikipedia.org/wiki/Polling_(computer_science))

- a descoberta de serviços em dispositivos com recursos limitados de memória e processamento;
- o *bootstrapping* para outros protocolos de *web service* bem como outros protocolos de transporte;
- a descoberta de serviços por vários critérios de busca;
- a criação de extensões para cenários não previstos.

3.5.5 WS-Addressing (WS-ADDRESSING, 2004)

A **transparência de localização** provida pela especificação WS-Addressing também é uma característica importante para que a troca de mensagens ocorra independente dos protocolos de rede utilizados no meio. O objetivo da especificação WS-Addressing é prover meios para identificar o *endpoint* de um *Web service* e uma forma de usar esse identificador nas mensagens SOAP para entrega e troca de mensagens entre *Web services*.

A especificação se faz necessária pelos seguintes motivos:

- O acesso ao serviço não pode ser limitado a um único protocolo de transporte, como o HTTP por exemplo. Mensagens SOAP podem ser transportadas por meio de outros protocolos e até mesmo vários protocolos diferentes podem ser usados para uma simples interação com o serviço. Para que uma transição de distribuição da interação possa ocorrer via *Bluetooth* essa flexibilidade se faz necessária;
- A interação com o serviço não está restrita ao modelo síncrono de requisição-resposta. O modelo assíncrono de mensagens e outras propriedades como segurança e transação são necessárias. A propriedade de transação permite que mudanças de estado realizadas nos serviços envolvidos possam ser desfeitas;
- Existem interações que podem necessitar de um período longo para ocorrer e que precisam manter o estado da conversação.

3.5.6 WS-MetadataExchange (WS-METADATAEXCHANGE, 2011)

A **vinculação dinâmica** apoiada pela especificação WS-MetadataExchange é essencial para facilitar o arranjo entre recursos computacionais em tempo de execução. Isso é importante em DUI porque permite uma comunicação

entre dispositivos na *forma ad hoc* e também porque fornece um modelo padronizado para troca de dados relacionados ao dispositivo, como:

- dados de fabricação;
- serviços oferecidos;
- identificação na rede.

Essa especificação permite a descoberta de metadados relacionados aos serviços de forma distribuída e não centralizada como ocorre no UDDI, respeitando a arquitetura de rede P2P.

3.5.7 WS-Eventing (WS-EVENTING, 2006)

O **tratamento de eventos** apoiado pela especificação *WS-Eventing* permite estabelecer uma arquitetura orientada a serviços dirigida a eventos, o que representa uma mudança de paradigma em relação a tradicional arquitetura orientada a serviço dirigida a requisição. E isso é importante em DUI porque permite que transições de distribuição da interação possam ocorrer sem a necessidade de que os dispositivos fiquem constantemente verificando se a interação já ocorreu. Cabe ao dispositivo que recebeu a interação notificar os interessados que ela já ocorreu e está pronta para ser consumida. Esse tipo de estratégia melhora o tempo de resposta, a escalabilidade da aplicação e consome menos largura de banda.

A especificação permite a *Web services* inscrever ou aceitar inscrições para mensagens de notificação de eventos. Essa especificação define um protocolo para que um *Web service* (chamado *event sink*) registre interesse (atividade chamada de *subscription*) em outro *Web service* (chamado *event source*) para receber mensagens sobre eventos (chamadas *notifications*).

Para melhorar a robustez, a inscrição é “arrendada” pelo *event source* a um *event sink*, sendo que a assinatura expira com o tempo, podendo o *event sink* renovar a inscrição. A Figura 11 ilustra a arquitetura *WS-Eventing* e suas operações.

Figura 11 – Arquitetura da especificação WS-Eventing e suas operações

Fonte: <http://crishantha.com/wp/?p=885>

A forma de entrega das mensagens de notificação do *WS-Eventing* pode utilizar estratégias *push*, *pull* e *wrapped*. O modo *wrapped* pode empacotar diversas mensagens de notificação em uma única mensagem para uma entrega mais eficiente. A especificação também permite o uso de um ponto de extensão “*Delivery*” presente na mensagem de inscrição para incorporar outras estratégias de entrega de mensagem de notificação.

3.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

A definição de uma arquitetura para DUI é complexa, dada a própria natureza da distribuição. Ela não se estabelece apenas nos níveis mais baixos de comunicação, chega a atingir a camada de apresentação e, por isso, observa-se muitas soluções em diversos níveis, utilizando variadas tecnologias.

O que se verifica é uma tendência, e até mesmo uma necessidade, de se utilizar padrões abertos nos mais diversos níveis da arquitetura. Outro ponto observado, é a necessidade dos recursos computacionais, abundantes no mundo atual, poderem ser facilmente “plugados” e utilizados de forma colaborativa.

A especificação DPWS auxilia no atendimento de requisitos importantes identificados no Quadro 4. O modelo de comunicação direta da DPWS vai ao encontro do requisito 19 (*ad hoc*), mas implica que todos os dispositivos devem implementar todos os protocolos necessários. A especificação também se mostra a mais completa para arquiteturas orientadas a serviço em dispositivos móveis, estacionários ou com recursos limitados. Essa especificação auxilia no atendimento do requisito 17 (metadados), por meio da especificação *WS-MetadataExchange*, do requisito 11, por meio da especificação *WS-Eventing*, e do requisito 21, por utilizar padrões abertos de comunicação.

Apesar de a especificação DPWS estar mais presente nos domínios de aplicações industriais, residenciais, sistemas automotivos e sistemas de telecomunicação, é possível a sua utilização em cenários de DUI conforme será apresentado no próximo capítulo.

4 UM MODELO DE SERVIÇOS PARA TRANSIÇÕES DE DISTRIBUIÇÃO DA INTERAÇÃO (TSS)

Uma arquitetura que atenda a todos os requisitos do Quadro 3 e do Quadro 4 é complexa. Dado que o conceito de transição é essencial para DUI, por permitir que a interação do usuário atravessasse (*cross*) múltiplos dispositivos, é que optou-se por tratar neste trabalho o suporte às transições de distribuição da interação.

Apesar do estabelecimento desse foco, vários requisitos da arquitetura que será apresentada já são atendidos:

- Requisito 02: Prover interfaces reutilizáveis de componentes de transição que ocultam detalhes da implementação, enquanto permitem ao *designer* customizar a interface do usuário.
- Requisito 03: Prover recursos para automatizar os mecanismos de transição tanto quanto possível.
- Requisito 05: Prover recursos para informar o usuário de possíveis erros que impossibilitem os mecanismos de transição automatizados de operar corretamente.
- Requisito 07: Ser flexível para suportar a adição de novos tipos de transição.
- Requisito 11: Permitir que interações em um dispositivo sejam delegadas a usuários em outros dispositivos.
- Requisito 12: Permitir que recursos do dispositivo sejam configurados para permitir ou não o acesso concorrente.
- Requisito 13: Permitir que mais de um dispositivo seja utilizado pelo mesmo usuário de forma simultânea e sincronizada.
- Requisito 17: Permitir que metadados possam ser agregados ao conjunto de informações do dispositivo.
- Requisito 19: Permitir a descoberta de dispositivos de forma *ad hoc*.
- Requisito 21: Permitir que novas aplicações, em outras plataformas, possam ser facilmente integradas.
- Requisito 22: Permitir que a comunicação entre dispositivos possa ocorrer de forma segura.

No capítulo 6 todos os 22 requisitos são resgatados para apresentar se foram ou não atendidos e como se realizou esse atendimento.

Este capítulo, portanto, propõe e define o modelo de arquitetura, objeto desta pesquisa, em modelo de SOA, de forma a facilitar a construção de aplicações em que a transição de distribuição da interação seja possível de forma *ad hoc* entre múltiplos dispositivos. Entende-se por *ad hoc*, a capacidade de dinamicamente realizar transições da interação entre dispositivos que surgem espontaneamente no meio.

4.1 ARQUITETURA DO SERVIÇO DE APOIO A TRANSIÇÃO (TSS)

Este serviço é responsável por realizar a **transição de distribuição da interação**. Este serviço atende, por exemplo, ao cenário Alice_{PC} → Bob*, no qual Alice em seu PC, deseja delegar a interação para com um componente GUI de controle (caixas de texto, de seleção, ...) a outro usuário independente do dispositivo que esteja utilizando (PC, *smartphone*, TV, ...).

A arquitetura do TSS segue a especificação DPWS como arquitetura base para descoberta e comunicação entre dispositivos. Todos os dispositivos envolvidos devem possuir uma implementação da pilha de protocolos da DPWS. Não se faz necessário, na verdade, que a especificação DPWS esteja disseminada nos dispositivos, mas sim que especificações como *WS-Discovery*, *WS-MetadataExchange* e *WS-Eventing* estejam neles implementadas. Como apresentado no capítulo anterior, a presença da tecnologia WS é crescente nos dispositivos. Se outra especificação de *Web Services* para dispositivos surgir e em sua pilha essas três especificações estiverem presentes, a arquitetura TSS ainda é viável. Iniciativas como da *Web Services for Devices* (WS4D¹²) já disponibilizam implementações DPWS em C, C++, Java e Android, a Microsoft¹³ também disponibiliza uma implementação em sua plataforma Microsoft .NET Micro *Framework*.

Sob a pilha DPWS faz-se necessário a implementação de serviços de dois tipos (vide Figura 12):

¹² <http://ws4d.e-technik.uni-rostock.de>

¹³ <http://msdn.microsoft.com/en-us/library/cc533006.aspx>

- **Serviços de recursos do dispositivo:** representam serviços que fornecem acesso a recursos já existentes no dispositivo. Em um *smartphone*, são exemplos de serviços de recursos os que dão acesso ao GPS, acelerômetro, bússola, emissões sonoras, etc. .
- **Serviços de recursos de interação:** representam serviços que deverão expor uma interface gráfica personalizada para o usuário. Por exemplo, apresentação de uma lista para seleção.

Figura 12 – Serviços que devem ser compostos sobre a pilha DPWS para ocorrência da transição de distribuição da interação

Fonte: Próprio autor

Como serviços de recursos do dispositivo apresentam-se, por exemplo, os da Figura 13, que representam possíveis serviços de um *smartphone*. Essa lista de serviços não é completa, podendo ser ajustada de acordo com o tipo de dispositivo que irá oferecer seus recursos na rede. Não está no escopo definir todos os serviços para cada tipo de dispositivo e sim demonstrar como a arquitetura funciona.

Os serviços de recursos do dispositivo devem ser fornecidos de fábrica, assim como já ocorre com o UPnP no ambiente doméstico. Um grande número de TVs já é compatível com UPnP, ou seja, expõem suas funcionalidades como serviço para comunicação com outros dispositivos. Isso garante um padrão quanto ao modelo de comunicação, mas não dos serviços ofertados. Cada fabricante de TV (Samsung, Sony, LG, ...) possui um padrão próprio quanto a interface dos serviços que são oferecidos.

Figura 13 – Serviços que fornecem acesso aos recursos do dispositivo

Fonte: Próprio autor.

Já para o mesmo *smartphone*, apresenta-se como exemplo de serviços de recursos de interação os apresentados na Figura 14. Basicamente dois serviços são ofertados, um relacionado à entrada de dados por meio de teclado (*KeyboardService*) e outro relacionado à escolha de itens em uma lista (*ChoiceService*). Não se vislumbra o fornecimento desses serviços de fábrica; iniciativas como da WS4D, de empresas especializadas ou desenvolvedores finais da solução distribuída devem implementar esses serviços nas diversas plataformas que irão compor o cenário distribuído.

Figura 14 – Serviços de recursos de interação

Fonte: Próprio autor

Todos os serviços (de recursos de dispositivo e de interação) devem ser especializações de *XService*, conforme Figura 15.

Figura 15 – Estrutura comum a todos os serviços que estão sob a pilha DPWS

Fonte: Próprio autor

Essa estrutura deve ser herdada para todos os serviços com vistas a permitir que:

- operações possam ser bloqueadas (*lockOperation*). Dessa forma é possível estabelecer se mais de um usuário pode interagir com a mesma operação de forma concorrente ou se deve aguardar até a finalização da atividade pelo usuário corrente. Esta atende ao requisito 12 do Quadro 4;
- um dispositivo qualquer, interessado em ter conhecimento de qualquer mudança de estado do serviço de outro dispositivo, possa fazê-lo por meio da operação *anEvent()*. Esta característica é muito benéfica porque proporciona um alto grau de desacoplamento do sistema distribuído. Essa forma de comunicação não precisa ser tão preocupada com alguns aspectos dos modelos de solicitação-resposta, tais como parâmetros de entrada e definições de interface de serviço. A única exigência é um formato de mensagem bem definido. Esta atende ao requisito 21 do Quadro 4.

Tendo os dispositivos cada qual sua implementação da pilha DPWS e os demais serviços de recursos de dispositivo e de interação, visualizam-se 10 possibilidades de comunicação. Oito dessas possibilidades representam cenários de **transição de distribuição da interação**. A notação utilizada anterior como (→) para representar a transição é agora especializada conforme Quadro 7, que apresenta as notações criadas para representar cada cenário, bem como uma descrição.

Neste quadro, são apresentados cenários de comunicação. Nela, a coluna S e R indicam, respectivamente, o número de usuários para qual é feita a solicitação e o número de respostas e respectivo tratamento:

- 1, na coluna S, indica que é estabelecida a comunicação com um único usuário e são permitidas R respostas;
- S=N significa que é estabelecida comunicação com múltiplos usuários;
- R=0 significa que não é esperado um retorno da interação do usuário com o qual se estabeleceu a comunicação;
- R=1 significa que será considerada válida uma única resposta desse usuário, possuindo tratamentos diferenciados dependendo da especialização de 1 (1f, 1l);
- R=M indica que múltiplas respostas desse usuário são esperadas ou permitidas, possuindo tratamentos diferenciados dependendo da especialização de M (Ml, Ma, Mfd, M/d).

O quadro a seguir detalha melhor esses modelos de comunicação.

Quadro 7 – Cenários de interação distribuída (diálogos solicitação (S) / resposta (R))

S	R	Cenário
1	0	<p><i>Singlecast message</i></p> <p>Envia uma única mensagem (não é uma requisição). Neste cenário, Alice deseja enviar uma mensagem para Bob. Este não é um cenário de transição de interação, mas ele deve ser considerado na implementação dos serviços para atender exemplos como:</p> <ul style="list-style-type: none"> - Alice envia uma mensagem <i>call-to-action</i> preparando Bob para uma atuação em um futuro próximo; - Alice informa a Bob que acabou de ficar online; - Alice informa a Bob uma mudança de estado de suas atividades no dispositivo; - Alice interage com um objeto e deseja que Bob obtenha informações acerca do referido objeto. Esse objeto pode ser uma imagem que quando selecionada envia a Bob informações textuais acerca do objeto.
1	1	<p><i>Singlecast first answer</i></p> <p>Caso típico, uma requisição é feita e uma resposta é esperada. Neste cenário, Alice realiza um <i>call-to-action</i> para Bob, transferindo a interação de um componente GUI para Bob, esperando uma única resposta. Ocorrida a interação de Bob, automaticamente a transição é encerrada, impedindo novas interações. Como exemplos desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice deseja saber a localização geográfica de Bob; - Alice solicita a Bob uma resposta acerca de uma pergunta.
1	M/	<p><i>Singlecast last answer</i></p> <p>Dispositivo remoto pode responder continuamente, mas para cada nova resposta enviada, a anterior é descartada pelo recipiente da mensagem. Neste cenário, Alice realiza um <i>call-to-action</i> para Bob, transferindo a</p>

		<p>interação de um componente GUI para Bob, permitindo que ele interaja de forma contínua com o componente GUI de Alice. Bob pode anular a interação anterior realizando uma nova interação com o componente. Alice decide o momento em que a transição deve ser encerrada.</p> <p>Como exemplo desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice solicita a Bob uma resposta acerca de uma pergunta. Bob pode mudar a resposta enquanto a transição estiver ativa;
1	Ma	<p><i>Singlecast multianswers</i></p> <p>Dispositivo remoto pode responder continuamente, sendo todas as respostas consideradas válidas.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para Bob, transferindo a interação de um componente GUI para Bob, permitindo que ele interaja de forma contínua com o componente GUI de Alice. Nenhuma das interações de Bob é descartada, todas as interações de Bob são apresentadas a Alice. Alice decide o momento em que a transição deve ser encerrada.</p> <p>Como exemplo desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice solicita a Bob o preenchimento de uma lista. Bob pode fornecer várias respostas que vão sendo agregadas ao componente de lista. Todas as ações de Bob são consideradas válidas e aceitas para compor a lista solicitada por Alice.;
N	0	<p><i>Multicast message</i></p> <p>Envia múltiplas instâncias da mesma mensagem (não é uma requisição). Neste cenário, Alice deseja enviar uma mensagem não somente para para Bob, mas para outros dispositivos também.</p> <p>Este não é um cenário de transição de interação, mas ele deve ser considerado na implementação dos serviços para atender exemplos como:</p> <ul style="list-style-type: none"> - Alice envia uma mensagem <i>call-to-action</i> preparando os usuários para uma atuação em um futuro próximo; - Alice informa a um grupo de usuários que acabou de ficar online; - Alice informa a um grupo de usuários uma mudança de estado de suas atividades no dispositivo; - Alice interage com um objeto e deseja que um grupo de usuários obtenha informações acerca do referido objeto. Esse objeto pode ser uma imagem que quando selecionada envia ao grupo informações textuais acerca do objeto. Por exemplo, um componente de ajuda contextual que tem atuação distribuída.
N	1f	<p><i>Multicast request, only first answer</i></p> <p>A transição é realizada para múltiplos dispositivos, sendo apenas uma resposta considerada válida, a do dispositivo que responder primeiro.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para um conjunto de dispositivos, transferindo a interação de um componente GUI para o conjunto. A primeira interação efetivada é considerada válida, encerrando automaticamente a transição para o conjunto, impedindo novas interações.</p> <p>Como exemplos desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice lança para uma turma, em uma sala de aula, uma questão valendo ponto para quem responder primeiro;
N	1l	<p><i>Multicast request, only last answer</i></p>

		<p>A transição é realizada para múltiplos dispositivos, sendo apenas uma resposta considerada válida, a última resposta recebida, do conjunto de todos os dispositivos envolvidos.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para um conjunto de dispositivos, transferindo a interação de um componente GUI para o conjunto. O conjunto pode interagir continuamente com o componente, mas a cada nova interação a anterior é descartada. Alice decide o momento em que a transição deve ser encerrada.</p> <p>Como exemplo desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice solicita a um grupo de alunos uma resposta acerca de uma pergunta. O grupo pode mudar a resposta enquanto a transição estiver ativa;
N	Mfd	<p><i>Multicast request, all first answers</i></p> <p>A transição é realizada para múltiplos dispositivos e somente a primeira resposta de cada dispositivo é válida.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para um conjunto de dispositivos, transferindo a interação de um componente GUI para o conjunto. A primeira interação efetivada em cada dispositivo é considerada válida, encerrando automaticamente a transição apenas do dispositivo que respondeu, impedindo novas interações desse dispositivo. Outros dispositivos do conjunto continuam com a transição ativa.</p> <p>Como exemplos desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice lança para uma turma, em uma sala de aula, uma enquete. Cada aluno deve fornecer uma única resposta à enquete;
N	Mld	<p><i>Multicast request, all last answers</i></p> <p>A transição é realizada para múltiplos dispositivos e somente a última resposta de cada dispositivo é válida.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para um conjunto de dispositivos, transferindo a interação de um componente GUI para o conjunto. O conjunto pode interagir continuamente com o componente, mas a cada nova interação a anterior, relacionada ao mesmo dispositivo, é descartada. Alice decide o momento em que a transição deve ser encerrada.</p> <p>Como exemplos desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice solicita, a todos os alunos de uma turma, uma resposta acerca de uma pergunta. Cada aluno pode responder e mudar a resposta enquanto a transição estiver ativa;
N	Ma	<p><i>Multicast request, all answers</i></p> <p>A transição é realizada para múltiplos dispositivos e todos os dispositivos podem submeter continuamente e todas as respostas são válidas.</p> <p>Neste cenário, Alice realiza um <i>call-to-action</i> para um conjunto de dispositivos, transferindo a interação de um componente GUI para o conjunto, permitindo interação de forma contínua do conjunto com o componente GUI de Alice. Nenhuma das interações do conjunto é descartada, todas as interações são apresentadas a Alice. Alice decide o momento em que a transição deve ser encerrada.</p> <p>Como exemplo desse cenário pode-se citar:</p> <ul style="list-style-type: none"> - Alice solicita na sala de aula que os alunos respondam quais cidades já visitaram. Todas as interações serão consideradas válidas.

O comportamento desses cenários pode ser implementado em componentes ou serviços. A Figura 16 permite abstrair na forma de diagrama de estado esses cenários para componentes ou serviços de interface gráfica que promovem uma interação distribuída por meio de transições de distribuição da interação.

Figura 16 - Diagrama de estado para componentes de interface gráfica que suportam as transições de interação do Quadro 7

Fonte: Próprio autor

O componente está pronto (*Ready*) para receber requisições depois de obter todas as informações sobre os dispositivos na rede (*Getting Available Devices and Services*) e só realiza esta operação novamente caso ocorra um erro de interpretação (*Parse Fault* - considerando a possibilidade de erro na lista de dispositivos retornada) no momento de exibir a lista de dispositivos (*Showing Devices*) ou no caso de um erro de rede (*Network Fault*), considerando que o dispositivo pode não estar mais *on-line*.

A condição de guarda `[(lastInput || allInputs) && !firstInput]` abrange a diversidade de cenários presentes no Quadro 7. Sendo verdadeira essa condição, o estado de *Listening* é mantido ativo aguardando novas interações do dispositivo remoto. Ao receber a resposta, entra-se no estado de *Working* para realizar a atualização da interface gráfica do componente com vistas a refletir o efeito da transição.

Para a realização das transições de distribuição da interação apresentam-se dois modelos de implementação. O primeiro, representado pela Figura 17, apresenta os serviços necessários para orquestração das transições. O segundo incorpora os elementos do primeiro em componentes MVC. A vantagem de se incorporar o serviço de transição em cada componente MVC é que fornece autonomia para que cada instância do componente de interface gráfica administre o

processo de transição, resultando em um menor número de chamadas, obtendo um ganho de performance e possibilitando uma maior granularidade e independência do componente. Características de composabilidade e decomposabilidade podem ser beneficiadas por esse modelo. Caso contrário, todas as instâncias de caixa de texto de uma aplicação, por exemplo, precisam delegar o processo de transição para o serviço de transição da interação, que ficará responsável em gerenciar a distribuição da interação de todas as instâncias de caixas de texto da aplicação.

Figura 17 – Representação do serviço para o suporte de transições de interação

Fonte: Próprio autor

Segundo o diagrama da Figura 17, para cada componente GUI que deseja realizar transições de interação faz-se necessário a especialização do serviço abstrato *InteractionTransitionSupportService*. Por exemplo, transições de interação relacionadas a caixa de texto são apoiadas pelo serviço *TextfieldTransitionSupportService*. Segundo esse modelo, todos os componentes instanciados de caixa de texto devem recorrer ao *TextfieldTransitionSupportService* para realização da transição da interação. Dessa forma, as chamadas dos componentes é centralizada em uma instância de serviço, que irá notificar todos os componentes de caixa de texto inscritos no serviço, sempre que receber um evento do tipo *TextfieldEventClient*, indicando a necessidade de uma atualização da interface gráfica.

Para que a transição ocorra são necessárias duas etapas. A primeira instancia a transição por meio da operação *addTransition*. Essa operação espera receber uma instância de *InteractionTransition*, que por sua vez, exige que sejam definidos o tipo de transição que irá ocorrer (*transitionType*), para quais dispositivos essa transição deve ocorrer (*devices*) e qual a mensagem de *call-to-action* (*callToActionMessage*) deve ser exibida para informar o usuário da transição e solicitar sua participação.

Após configurada a transição, inicia-se a segunda etapa por meio da invocação da operação *transitionAction()* que realiza a transição e guarda referência de cada interação ocorrida em instâncias de *InteractionEventClient*. Essa classe deve ser especializada para tratar as interações de acordo com seu tipo. Interações de caixa de texto são tratadas por *TextfieldEventClient*. A Figura 18 ilustra essas etapas e por meio de um exemplo de uso do serviço de transição para um componente de caixa de texto.

Figura 18 – Diagrama de sequência mostrando um exemplo de uso do serviço de transição da interação

Fonte: Próprio autor

Para componentes de interface gráfica apresenta-se um modelo MVC para implementação das transições de acordo com a Figura 19. Essas classes são abstratas e precisam ser especializadas por todos os componentes GUI.

Figura 19 – Modelo MVC para transições de interação em componentes GUI

Fonte: Próprio autor

As classes *Model-View-Controller* estão precedidas com o caractere “X” para remeter a palavra *cross*, o que denota a relação entre dois ou mais dispositivos. Um ponto importante está em futuramente pensar nos aspectos de design de interação para que possam ser definidos neste padrão de projeto sem alterar o comportamento das transições.

A classe *XView* representa a camada de visão, na qual se estabelece que os componentes GUI deverão possuir uma área para representação visual das informações de transição (*transitionPanel*) uma imagem que remeta ao conceito de transição para fácil identificação do propósito dessa região da interface gráfica (*image*), apresentação do conjunto de dispositivos disponíveis para se estabelecer o processo de transição (*devices*) e um componente que remeta ao momento em que a ação de transição deve ocorrer (*transitionAction*).

Adotou-se, apenas para demonstração, o símbolo de compartilhamento (↔) para representar o conceito de transição na forma de ícone. Faz-se necessário um estudo iconográfico para verificar a melhor forma de representar o tipo de transição de interação configurada para o componente.

A classe *XController*, por sua vez, representa a camada de controle na qual se estabelece que é possível configurar o componente GUI para definir o tipo de transição de interação desejada (*interactionTransitionType*) e a mensagem que deve ser apresentada no dispositivo remoto para denotar qual o objetivo da transição que está sendo realizada (*callToAction*).

Por última, a classe *XModel* representa a camada de modelo. É nessa camada que o comportamento do diagrama de estado deve ser implementado. Essa

camada define que se faz necessário conhecer o serviço com o qual o componente se relaciona (*service*) e quais os dispositivos estão disponíveis para que o serviço seja invocado (*devices*). Também está definido que se faz necessário implementar ações que permitam atualizar a lista de dispositivos disponíveis na rede (*updateDevices*) bem como o algoritmo de invocação do comportamento de transição (*transitionAction*). Como um determinado componente GUI pode interagir concorrentemente com diversos dispositivos na rede de forma assíncrona e dirigida a eventos, se faz necessário a instanciação de *EventClient* para cada operação remota invocada com vistas a monitorar possíveis eventos dos dispositivos remotos. Essa classe define que é necessário a implementação de um algoritmo para o recebimento do evento (*eventReceived*), um algoritmo para determinar o que o componente deve fazer quando a inscrição em um evento for encerrada (*subscriptionEndReceived*) e como o componente deve se comportar quando a inscrição no evento for encerrada porque o tempo limite da comunicação foi excedido (*subscriptionTimeoutReceived*).

A relação entre as classes do modelo MVC não foi estabelecida, visto que essa relação pode variar dependendo do modelo MVC implementado pela API gráfica escolhida para implementação. No mínimo, toda arquitetura MVC permite que a camada de visão enxergue a camada de controle, que por sua vez enxerga a camada de modelo. O modelo MVC proposto apresenta um conjunto mínimo de elementos necessários para que a relação *cross-device* se estabeleça para o conjunto de transições de interação previamente estabelecido.

Cada plataforma deve ter a sua própria implementação dos serviços de recursos de interação e dos componentes GUI, visto que, elementos gráficos de interação variam entre as plataformas. Por exemplo, selecionar uma opção em uma lista pode ser realizada de forma muito diferente entre um PC e um *smartphone*. Nesta pesquisa foi realizada a implementação das provas de conceito nas plataformas Android, Windows e MacOs, para dispositivos *desktop*, *laptop*, *tablet* e *smartphone*.

Apesar disso, acredita-se que o custo de desenvolvimento é menor que nas soluções *ad hoc* vistas na pesquisa de literatura, porque com o uso do DPWS houve um *empowerment* nas capacidades do dispositivo em uma estrutura de *Web Services*, que permite uma comunicação facilitada e menos propensa a erros. Além

disso, com o crescimento do padrão WS, integrações futuras são mais bem administradas, sem grandes refatoramentos no código ou adaptadores de comunicação complexos.

Espera-se que em um breve futuro, as próximas APIs gráficas contemplem a necessidade de atender uma interação distribuída. Que aproveitem a definição deste padrão de transição da interação apresentado para um conjunto mínimo de componentes, como ocorre com o HTML nos navegadores ou como ocorre com o JSF¹⁴ para interfaces gráficas MVC. Apesar do esforço inicial, essa solução pode ser adotada para inúmeras aplicações naquela plataforma. Especializar esses componentes para a plataforma Android, por exemplo, já permite criar aplicações para milhões de dispositivos. O mesmo vale para plataformas como Windows, iOS e Ginga.

É importante ressaltar também que o efeito das interações na interface gráfica do componente (*look*), de acordo com os cenários descritos no Quadro 7, é particular a cada componente.

Por exemplo, seja o cenário Alice_{PC} → N:M/d *, no qual Alice solicita a todos os alunos de uma turma, uma resposta acerca de uma pergunta. Cada aluno pode responder e mudar a resposta enquanto a transição estiver ativa. Nesse cenário, como deve se comportar o componente GUI de caixa de texto (da pergunta) que está em Alice_{PC}?

Em nossa implementação, o componente de caixa de texto se comporta como um *combo-box*. Cada opção do *combo* representa um usuário e a *string* formada segue o padrão:

<Identificação do usuário> (<tipo de dispositivo>): <valor fornecido>

A Figura 20 ilustra esse componente, em que a primeira opção do *combo* representa a interação do usuário jppreti em seu *desktop* com o valor “Fun”.

Figura 20 – Caixa de texto recebendo interações no formato N:M/d

Fonte: Próprio autor

¹⁴ Framework MVC baseado em Java para construção de GUI baseadas em componentes para web.

Outro cenário é o Alice_{PC} →_{N:Mfd} *, no qual Alice lança para uma turma, em uma sala de aula, uma pergunta que possui uma lista de opções de resposta. Cada aluno deve fornecer uma única resposta. Nesse cenário o componente foi implementado sendo um conjunto de botões *Radio* e barras de progresso. A Figura 21 ilustra esse componente.

Figura 21 – Lista de opções recebendo interações no formato N:Mfd

Fonte: Próprio autor

Um estudo de usabilidade não está no escopo deste trabalho, mas é uma pesquisa necessária. Como reforço entende-se a necessidade de um estudo de iconografia. É possível verificar que componentes GUI, que permitem uma transição de interação, podem apresentar os efeitos da interação de maneira muito particular a cada um. Um estudo relacionado pode levar a necessidade de criação de novos padrões de interface gráfica.

É bastante curioso observar que a distribuição da interação agregou uma nova dimensão aos componentes GUI que refletem o modelo 1 usuário:1 dispositivo. Essa nova dimensão das transições obriga repensar todos os componentes de interface gráfica.

4.2 DESCOBERTA DE DISPOSITIVOS NA REDE

A localização dos dispositivos na rede é realizada por meio de uma implementação da especificação *WS-Discovery*, que define um protocolo *multicast* de descoberta para localizar serviços em uma rede local. Assim como o nome sugere, a comunicação entre os nós é feita utilizando padrões de *Web Services*, mais especificamente SOAP sobre UDP. A ilustra as mensagens trocadas em modo *ad hoc*, sendo que o *Target Service* é a representação do dispositivo como serviço.

Figura 22 – Modelo *WS-Discovery* de mensagens para descoberta de dispositivos em rede

Fonte: (WS-DISCOVERY, 2009)

Há pelo menos 3 (três) estratégias para realizar a descoberta dos dispositivos por meio do *WS-Discovery*. A primeira é quando o *Target Service* envia uma mensagem *multicast* de *Hello* (1) no momento em que o serviço é adicionado a rede. Um cliente escuta pela mensagem de *Hello* e guarda referência ao *Target Service* encontrado. Uma segunda estratégia para descobrir o *Target Service* é enviar uma mensagem *multicast* *Probe* (2) para localizar *Target Services*. Se um *Target Service* corresponde ao *Probe*, o mesmo responde (3) com uma mensagem *unicast* *Probe Match* (PM). Outros *Target Services* compatíveis podem enviar também um PM. A terceira estratégia é quando o cliente procura por um *Target Service* em particular, para isso o cliente envia uma mensagem *multicast* *Resolve* (4) para localizar o *Target Service* específico. Se o *Target Service* corresponde ao *Resolve* ele responde com uma mensagem *unicast* (5) *Resolve Match* (RM). Sempre que o *Target Service* deixa a rede ele envia antes uma mensagem *multicast* *Bye* (6).

Para descoberta dos dispositivos na arquitetura TSS foi implementada uma classe intitulada *DPWSCClient*. Essa classe é utilizada na arquitetura para localização dos dispositivos e utiliza um algoritmo que se baseia em um misto da primeira e segunda estratégia para otimizar as mensagens trocadas na rede. A apresenta a estrutura da classe *DPWSCClient* implementada e utilizada.

Figura 23 – Classe DPWSClient para localização de dispositivos na rede

DPWSClient
availableDevices : DeviceReference [*]
+ getInstance(): DPWSClient + deviceFound(in helloData: HelloData) + helloReceived(in helloData: HelloData) + deviceRunning(in deviceRef: DeviceReference) + deviceChanged(in deviceRef: DeviceReference) + deviceBuiltUp(in deviceRef: DeviceReference) + deviceBye(in deviceRef: DeviceReference) + deviceCommunicationErrorOrReset(in deviceRef: DeviceReference) + getAvailableDevices(): DeviceReference [*]

Fonte: Próprio autor

Quando a classe DPWSClient é instanciada (*getInstance*) ela cria uma estrutura de dados para guardar todas as referências dos dispositivos encontrados (*availableDevices*). Em seguida, realiza um *probe* para localizar (*deviceFound*) todos os dispositivos na rede, só adicionando e removendo dispositivos (*availableDevices*) por meio da escuta de mensagens de *Hello* (*helloReceived*) e *Bye* (*deviceBye*). Esse modelo de funcionamento minimiza a troca de mensagens na rede para operações de busca.

O formato de comunicação *ad hoc* entre os serviços de dispositivos traz como grande vantagem a inexistência de um centralizador, o que traz benefícios como: facilidade de conexão, confiabilidade (no caso de um dispositivo centralizador vir a falhar), desempenho e distribuição do processamento.

O próximo capítulo apresenta como o TSS foi implementado e como o DPWS e outras tecnologias foram utilizadas para apoiar as provas de conceito.

5 IMPLEMENTAÇÃO DO TSS

Nesta seção, apresenta-se a implementação do Serviço de Apoio a Transição desenvolvida para as plataformas Windows, MacOS e Android nos ambientes web, *desktop* e *mobile*, utilizando PC, *notebook*, *tablet* e *smartphone*. Apresentam-se ainda provas de conceito, que mostram a execução dos componentes GUI e da arquitetura como um todo.

A seguir é apresentado o formato de troca de mensagens selecionado para uso na arquitetura proposta.

5.1 TROCA DE MENSAGENS

A forma como os sistemas são conectados em uma arquitetura dirigida a eventos traz mudanças significativas em relação a uma arquitetura dirigida a requisição. Ao invés do tradicional modelo síncrono de requisição/resposta (*pull*), o modelo dirigido a eventos utiliza um modelo inscrição (*publish/subscribe*) no qual notificações são enviadas aos interessados (*push*) em um modelo *fire-and-forget* (não há bloqueio) (ZDUN, HENTRICH E DUSTDAR, 2007). Nesse modelo ocorre a transferência de boa parte da responsabilidade do controle do fluxo da comunicação da fonte do evento, delegando para os receptores do evento. A Figura 24 e a Figura 25 apresentam uma visão lógica da diferença de fluxo desses dois modelos.

Figura 24 – Arquitetura dirigida a requisição

Fonte: Próprio autor

Figura 25 – Arquitetura dirigida a eventos

Fonte: Próprio autor

Na arquitetura dirigida a requisição, quando uma requisição é feita ocorre um bloqueio do processamento aguardando a resposta à requisição. O exemplo da Figura 24 demonstra que, a requisição 2 é bloqueada e somente após as requisições 3 e 4 serem concluídas é que será possível realizar a requisição 5.

Já na arquitetura dirigida a eventos esse bloqueio não ocorre, conforme Figura 25, a requisição 2 não é bloqueada, não existe uma dependência, e por isso a numeração das requisições é reiniciada para o dispositivo *smartphone*.

Essa característica melhora escalabilidade do sistema distribuído. A troca de mensagens em formato *push* proporciona uma melhora no desempenho em relação a mensagens *pull*. A Figura 26 apresenta um estudo de Zhang e Deters (2012) mostrando o tempo gasto para disseminar eventos via *pull* (em azul) e *push* (em vermelho). As linhas mostram o tempo total gasto para uma mensagem trafegar de um dispositivo para outro utilizando HTTP como protocolo de transporte por meio de um *proxy*. O eixo x apresenta a quantidade de amostras e o eixo y o tempo total gasto. É possível verificar que, quando o intervalo de tempo da estratégia *pull* é encerrado no momento em que o evento está pronto para ser consumido, o tempo gasto é bem aproximado da estratégia *push*. O problema é que se esses momentos não coincidem deve-se aguardar um novo intervalo para consumir o evento, o que leva a ao formato triangular e com diversos picos. Portanto, foi estabelecido o uso do WS-Eventing utilizando estratégia *push* na implementação da arquitetura.

Figura 26 – Push versus Pull

Fonte: (ZHANG E DETERS, 2012)

A próxima seção apresenta como foram implementadas e realizadas as provas de conceito de uso da arquitetura.

5.2 PROVAS DE CONCEITO

Para verificação do modelo e do uso da DPWS na composição de aplicações DUI foram realizadas implementações e testes em três contextos: *web*,

desktop e *mobile*. As seções que seguem demonstram como o modelo foi implementado para verificação de sua viabilidade.

O TSS disponibiliza ao desenvolvedor os seguintes recursos:

- conjunto de componentes (texto, lista, *hyperlink*, áudio, *mouse* e *tooltip*) que estendem a interação para múltiplos dispositivos;
- experimentos de comunicação no contexto *Web*;
- exemplos de implementação na plataforma Android e;
- exemplos de implementação nas plataformas Windows/MacOS.

5.3 AMBIENTE DE DESENVOLVIMENTO

Para que se possa verificar a viabilidade de uso da arquitetura para implementar o modelo de transições do Quadro 7, foram implementadas provas de conceito utilizando as ferramentas do Quadro 8:

Quadro 8 – Ferramentas de apoio ao desenvolvimento das provas de conceito

Ferramenta	Versão	Descrição
Mac OS X Mountain Lion	10.8.5	Sistema operacional e estação de trabalho
Ubuntu	12.04	Sistema operacional Linux
ADT	21.0.0	<i>Toolkit</i> de desenvolvimento para Android
Tomcat	7.0.28	Servidor de aplicação <i>Web</i>
Git	1.7.9.5	Sistema de controle de versão
Java SDK	1.7	<i>Toolkit</i> para desenvolvimento Java
Eclipse	Indigo	IDE de apoio ao desenvolvimento
Astah Community	6.6.3	Diagramação UML
Samsung Galaxy S3	4.1.1	Smartphone
Samsung Galaxy Note	4.0.1	Smartphone
Nexus 4	4.4	Smartphone
Nexus 7	4.4	<i>Tablet</i>
JMEDS	2.0.8	<i>Framework</i> DPWS
DPWS Explorer	3.4.3	Ferramenta da investigação DPWS
SOAPUI	4.5.1	Ferramenta gráfica que permite executar testes de carga, funcionais, de regressão e de compatibilidade em serviços SOAP e REST
Wireshark	1.6.0	Ferramenta de investigação de mensagens trocadas na rede.
Firefox	23.1.0	Navegador <i>Web</i>
Primefaces	2.2	<i>Framework Web</i>
JavaFX	2.0	<i>Framework</i> GUI

Fonte: Próprio autor

A plataforma Java¹⁵ e Android foram escolhidas por estarem presentes em diversos dispositivos. Como na plataforma Android existe uma diversidade de versões entre os vários dispositivos, foi definida uma versão mínima que atingisse o maior número de dispositivos e que tivesse todos os recursos utilizados pelas operações implementadas. A versão definida foi a *Gingerbread* (2.3), visto que, da versão 2.3 a atual 4.4, é possível atingir aproximadamente 98,7%¹⁶ dos dispositivos Android existentes em 2013.

As ferramentas DPWS Explorer, SOAPUI e Wireshark apoiaram o processo de teste e depuração das provas de conceito para verificar as mensagens trocadas entre os dispositivos.

A implementação da especificação DPWS escolhida é o *framework* Java intitulado *Java Multi Edition DPWS Stack* (JMEDS), que é um dos projetos do WS4D¹⁷ (*Web Service for Devices*), uma entidade que visa levar a Arquitetura Orientada a Serviço (SOA) e a tecnologia de *Web Services* para os domínios de aplicações industriais, residenciais, sistemas automotivos e sistemas de telecomunicação. O WS4D também oferece implementações da DPWS em C, C++ e Android, além de disponibilizar ferramentas de auxílio como a DPWS *Explorer*, para rapidamente visualizar e explorar dispositivos na rede que são compatíveis com a especificação DPWS.

A Figura 27 apresenta a arquitetura do *framework* JMEDS que está dividida em cliente e servidor e organizada em 3 camadas: aplicação, despacho e comunicação.

¹⁵ http://www.java.com/pt_BR/about/

¹⁶ <http://developer.android.com/about/dashboards/index.html>

¹⁷ <http://ws4d.e-technik.uni-rostock.de/>

Figura 27 – Arquitetura do *framework* JMEDS

Fonte: (JMEDS, 2011)

A camada de comunicação hospeda os controladores específicos de cada tecnologia. Ela é fracamente ligada a camada de despacho, possibilitando trocar a tecnologia de comunicação sem modificar as camadas superiores.

A camada de despacho consiste de 3 componentes, o *Search Manager* que cumpre a especificação *WS-Discovery*, o *Dispatcher* que manipula as mensagens de saída, adequando-as ao meio de comunicação e o *Device/Service Registry* que guarda uma referência de todos os dispositivos encontrados, eliminando a necessidade de se fazer uma busca por um dispositivo que já foi reconhecido.

A camada de aplicação hospeda a implementação do cliente em um lado e a do dispositivo e seus serviços do outro lado. Estas implementações são feitas pelos usuários do *framework*. As mudanças no estado de um dispositivo são transmitidas de um lado para o outro, atualizando as referências no lado do cliente. É nesta camada que os desenvolvedores devem implementar os serviços de recursos do dispositivo e de interação da arquitetura TSS.

A Figura 28 apresenta a arquitetura proposta e as aplicações que foram implementadas para atender cada uma das camadas. É possível verificar que a

implementação da especificação DPWS utilizada é o *framework* JMEDS. Também é possível observar que duas aplicações foram concebidas para prover os serviços dos recursos do dispositivo e de interação para plataformas Android e Desktop (XDevice Android e XDevice Desktop, respectivamente). Para implementação dos componentes DUI com suporte aos serviços de transição foi concebida uma GUI API (XDevice API) em JavaFX.

Figura 28 – Pilha de serviços da arquitetura implementada e respectivas aplicações (XDevice API, XDevice Android e XDevice Desktop)

Fonte: Próprio autor.

5.4 CONTEXTO ANDROID

Para o contexto de dispositivos que possuem plataforma Android foi criada uma aplicação nomeada de XDevice. Essa aplicação possui uma implementação da especificação DPWS (por meio do JMEDS) e dos serviços que expõem os recursos do dispositivo e os serviços de interação. Esse aplicativo foi estruturado conforme Figura 29.

Figura 29 – Estrutura de pacotes do aplicativo XDevice

Fonte: Próprio autor

Todas essas classes foram implementadas, sendo que o destaque deve ser dado às classes *XDevice*, *XDeviceApplication*, *AudioActivity*, *RecorderThread*, *PhotoUtil* e o pacote *service*:

- *XDevice*: representa o dispositivo DPWS, é por meio dela que são registrados todos os serviços e operações do dispositivo. Na DPWS todo dispositivo deve possuir uma representação na forma de serviço;
- *XDeviceApplication*: representa a interface gráfica da aplicação XDevice: é por meio dela que os serviços de interação apresentam seus componentes de interface gráfica;
- *AudioActivity*: representa uma atividade do Android de captura de áudio. Essa *activity* é utilizada pelo serviço DPWS de áudio;

- *RecorderThread*: é a classe que permite, em segundo plano, realizar a captura de áudio para envio via *stream* pelo serviço DPWS de áudio;
- *PhotoUtil*: classe utilitária para criar uma base de imagens tiradas por uma requisição feita a um serviço DPWS de câmera fotográfica. O objetivo é não misturar fotos tiradas por uma chamada a um serviço DPWS com o álbum pessoal do usuário;
- Pacote *service*: pacote no qual devem ser implementados todos os serviços de recursos e de interação da arquitetura.

Ao ampliar a visão do pacote `xdevice.dpws.android.service` observa-se os serviços que foram implementados e que estão organizados conforme a Figura 30.

O *framework* JMEDS já fornece as classes abstratas *DefaultDevice*, *DefaultService*, *Operation* e *DefaultEventSource* para iniciar o desenvolvimento dos dispositivos, serviços, operações e eventos dos serviços, respectivamente. Essas classes permitem abstrair os detalhes de comunicação com as camadas de despacho e comunicação do *framework* JMEDS. Essa estrutura permite focar o desenvolvimento nos serviços de recursos e de interação do dispositivo.

Figura 30 – Organização dos serviços de recursos e de interação do dispositivo Android

Fonte: Próprio autor

Para a correta execução de todos os serviços implementados na plataforma Android, fez-se necessária a liberação dos recursos ilustrados na Figura 31.

Figura 31 – Recursos necessários no dispositivo Android para execução de todos os serviços implementados

Fonte: Próprio autor

O Aplicativo XDevice possui as seguintes funcionalidades em sua interface gráfica (Figura 32):

- Iniciar os serviços DPWS;
- Parar os serviços;
- Sair do aplicativo;
- Responder a uma solicitação de entrada de texto;
- Tirar uma fotografia;
- Iniciar a captura de áudio.

Figura 32 – Interface gráfica do aplicativo Android XDevice

Fonte: Próprio autor

Não houve preocupação em observar aspectos de qualidade da interface gráfica, apenas a publicação dos serviços e uma forma básica para interação com os mesmos. O objetivo está em verificar a viabilidade de uso da arquitetura em uma plataforma bastante disseminada em dispositivos móveis. Reitera-se que o estudo de design de interação, de iconografia e de usabilidade de uma forma geral deve ser realizado para criar a melhor forma de apresentar os recursos de interação ao usuário.

Na próxima seção são apresentados os experimentos iniciais, em um contexto *Web*, para consumir os recursos disponibilizados na plataforma Android por meio do aplicativo XDevice.

5.5 EXPERIMENTOS DE COMUNICAÇÃO NO CONTEXTO *WEB*

Alguns experimentos foram realizados no contexto *Web*, com o objetivo de verificar a possibilidade e as dificuldades de se estabelecer uma comunicação

entre componentes de UI HTML (que trabalham em um modelo cliente/servidor) com dispositivos DPWS em uma rede P2P.

Esse foi o primeiro experimento para testar uma comunicação de um componente do navegador *web* com o DPWS, o que se mostrou bastante penosa e propensa a erros. Na época desse experimento a arquitetura estava em seu processo inicial de definição, portanto, estes representam resultados intermediários da pesquisa. Na seção 5.7 os problemas são discutidos e é apresentada uma solução alternativa por meio de *frameworks* MVC para *Web*.

O primeiro experimento foi a execução de um cenário no qual Alice em seu Alice_{PC} acessa um formulário em um navegador *web*, faz uma solicitação para Bob, em seu Bob_{SP}, para interagir com um componente de interface de usuário (uma caixa de texto) a qual Alice pede que ele preencha com a sua opinião. A Figura 33 descreve em maiores detalhe esse cenário.

Figura 33 –Alice_{PC} transferindo a interação de um componente de texto para Bob_{SP}

Fonte: Próprio autor

Para o experimento, o navegador utilizado foi o Firefox. Foi necessário implementar uma extensão para esse navegador para invocar um serviço SOAP em

um servidor de aplicativos com a pilha DPWS implementada. O serviço no servidor de aplicação tem a responsabilidade de manter e dar conhecimento dos dispositivos DPWS ativos na rede. O dispositivo que deseja participar da interação precisa estar distribuído na mesma rede local do servidor *web* ou utilizar um *proxy*. O serviço SOAP referido mantém uma lista dos dispositivos disponíveis na rede e retorna uma lista de dispositivos em uma estrutura XML que é analisada pela extensão do navegador para construir o menu *pop-up* para cada entrada de texto na página *web* HTML. O XSchema do documento XML trocado entre servidor de aplicação e o navegador pode ser visualizado na (Figura 34). Esse esquema XML descreve que o documento XML a ser enviado para o navegador deve conter basicamente a relação de serviços e operações disponíveis para cada dispositivo na rede.

Figura 34 – XSchema contendo informações dos dispositivos disponíveis na rede

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.example.org/DevicesAndServices"
  xmlns:tns="http://www.example.org/DevicesAndServices"
  elementFormDefault="qualified">

  <complexType name="device">
    <sequence>
      <element name="id" type="string" maxOccurs="1"
        minOccurs="1">
      </element>
      <element name="wsdlLocation" type="anyURI" maxOccurs="1"
        minOccurs="1">
      </element>
      <element name="type" maxOccurs="1" minOccurs="1">
        <simpleType>
          <restriction base="string">
            <enumeration value="TV"></enumeration>
            <enumeration value="Mobile"></enumeration>
            <enumeration value="PC"></enumeration>
          </restriction>
        </simpleType>
      </element>
      <element name="operation" type="string" minOccurs="1" />
    </sequence>
  </complexType>

  <complexType name="dpwsDevices">
    <sequence>
      <element name="device" type="tns:device"
        maxOccurs="unbounded" minOccurs="0" />
    </sequence>
  </complexType>

  <element name="dpwsDevices" type="tns:dpwsDevices"></element>
</schema>
```

Fonte: Próprio autor

Antes disso, para executar a extensão do Firefox, o usuário precisa clicar no botão imagem (demarcado por uma elipse), como pode ser visto na Figura 35.

Um menu com uma lista de dispositivos e de operações é apresentada na caixa de texto do navegador de Alice_{PC} quando o evento *onMouseover* é disparado sobre essa caixa de texto.

Figura 35 – Opções da caixa de texto para a entrada remota

Fonte: Próprio autor

Quando Alice pede uma entrada remota (*Request input string*), uma notificação é lançada no *smartphone* de Bob (Bob_{SP}) como pode ser visto na Figura 36. Se Bob selecionar o ícone para mostrar as notificações, a notificação *Keyboard Request* apresenta a mensagem de solicitação.

Figura 36 – Notificação de transição *cross-device*

Fonte: Próprio autor

Quando Bob seleciona a notificação *Keyboard Request*, exibe-se em Bob_{SP} um campo de texto de entrada com um teclado QWERTY esperando a interação remota de Bob (Figura 37). Enquanto isso, o navegador no Smartphone de Alice faz pedidos em intervalos de tempo de 5 segundos (*pull*), esperando a entrada de texto ser enviada.

Figura 37 – Resposta para a requisição cross-device

Fonte: Próprio autor

5.5.1 DISCUSSÃO DAS PROVAS DE CONCEITO

A implementação no lado cliente enfrentou muitos problemas devido às restrições do navegador. As seções que seguem relatam os problemas enfrentados.

5.5.1.1 Segurança

O problema de segurança está associado ao cabeçalho HTTP *Access-Control-Allow-Origin*. Esse cabeçalho define para qual domínio o navegador pode realizar requisições HTTP. Por segurança, o navegador só autoriza chamadas Javascript para o domínio no qual a página foi originada. Para que esse acesso seja liberado, o servidor *web* tem que ser configurado para que todas as respostas HTTP contenham esse cabeçalho com o valor "*", permitindo que códigos Javascript embutidos nessa página possam acessar outros domínios.

A disponibilização de um navegador personalizado, desabilitando esse aspecto de segurança, pode ser disponibilizado para o usuário como uma solução. Nenhuma dessas soluções é boa, ambas fragilizam a segurança da aplicação, expondo o usuário a ataques externos. Em nossos experimentos o Firefox foi configurado para iniciar com essa segurança desabilitada.

Uma solução mais adequada para esse problema é não depender do navegador para realizar chamadas externas. Essas chamadas devem ser feitas no servidor, e o cliente pode atualizar o *status* por meio de comunicação AJAX. A seção que segue complementa os problemas enfrentados e justifica o uso de *plug-ins* de servidor para resolver o problema.

5.5.1.2 Javascript x *Plug-ins* de servidor *Web*

Outro problema reside no fato de que invocar *Web Services* via Javascript exige uma programação onerosa e que é propensa a erros. Javascript também não suporta *threads* e a pilha DPWS, o que exige implementação no lado do servidor ou de um *plugin* de navegador que a suporte.

O *plugin* se mostra desfavorável no contexto *web* por três motivos identificados. Primeiro, porque exige que cada cliente instale o *plugin*. Segundo, porque o cliente normalmente não se encontra em um ambiente controlado, levando a preocupações de segurança. Por último, está a diversidade de navegadores existentes, construir um *plugin* para cada plataforma é oneroso.

Como solução para essas limitações, a implementação de componentes de interface gráfica do lado do servidor *web* (Figura 38 e Figura 39) permite manter a segurança e os aspectos originais do navegador *web*.

Figura 38 – Componente JSF para listar dispositivos, serviços e operações

Fonte: Próprio autor

Figura 39 – Componente JSF de mapa para exibir a localização dispositivo DPWS

Fonte: Próprio autor

Estender os componentes de interface do usuário de *frameworks* MVC para apoiar o diagrama de estados da Figura 16, pode ajudar na rápida construção de aplicações multiplataforma, com interação distribuída, além de eliminar a necessidade de uma personalização do navegador ou uso de *plugin*. Foram especializados dois componentes UI JSF, um para visualização das operações (Figura 38) e outro para apontar a localização geográfica do dispositivo selecionado ao clicar em *getLocalization* (Figura 39). Todos os dispositivos devem ser configurados para usar o mesmo *proxy* de forma que eles não precisem estar na mesma rede física. Esta implementação difere do cenário com Javascript, pois não há necessidade de implementação de um serviço de descoberta para a extensão JSF, visto que o próprio *framework* pode atuar como um cliente DPWS, usando o serviço de descoberta nativa.

Os experimentos no contexto *web* foram realizados para descobrir e compreender os desafios em se utilizar a arquitetura proposta em um ambiente *web* e para comprovar a possibilidade de realização da mesma no contexto cliente/servidor.

Nos experimentos da plataforma *web* não houve preocupação quanto a transição de distribuição da interação observando o Quadro 7 (quadro das transições 1:0, N:m,).

Para testar de fato essas transições optou-se por um ambiente que permitisse uma implementação mais produtiva. A seção que segue apresenta a implementação dos componentes GUI que possibilitam a realização das transições de distribuição da interação.

5.6 CONTEXTO *DESKTOP*

Para dispositivos do tipo *desktop* a tecnologia JavaFX foi empregada. Essa tecnologia tem por objetivo abranger também plataformas *mobile* e TV. Cada componente GUI em JavaFX é composto por 3 camadas: *control*, *skin* e *behavior*.

A camada *control* é basicamente responsável por manter as propriedades do componente e atuar como a classe principal, sendo esta a ser instanciada e adicionada à árvore de objetos que compõe a interface do usuário.

A camada *skin* é responsável pela visualização e *layout* do componente, enquanto a *behavior* é responsável pelo comportamento de interação com o usuário.

Para uma correta implementação é necessário saber como as classes estão relacionadas e como realizar a comunicação entre elas. A Figura 40 apresenta como é possível acessar, de uma determinada camada, as outras duas.

Figura 40 – Modelo MVC do JavaFX.

Fonte: (HENDRIK, 2012)

Para a realização dos testes foram criados 6 componentes que exemplificam as transições de interação do Quadro 7:

- *XTextField*: representa o conceito de uma caixa de texto que suporta as modalidades de transição da interação
- *XRadioButton*: opções de seleção do tipo *radio* no qual múltiplos dispositivos participam da seleção;
- *XHyperlink*: que permite disparar um *hyperlink* em todos os dispositivos selecionados;
- *XToolTip*: que permite disseminar uma ajuda contextual para os dispositivos interessados;
- *XMouse*: que permite controlar cursores de mouse remotos e;
- *XAudio*: que permite capturar o audio de um dispositivo remoto.

Para cada componente, até 3 classes foram criadas com o objetivo de separar as preocupações de acordo com a Figura 19. Começando pela camada de controle têm-se a organização de classes conforme Figura 41.

Figura 41 – Camada de controle para os 4 componentes

Fonte: Próprio autor

Observa-se que os componentes devem herdar de *XControl*. As especializações da classe de controle adicionam propriedades mínimas para definir o tipo de transição de interação desejada (*interactionTransitionType*) e a mensagem a ser exibida no dispositivo remoto quando a transição for acionada (*callToAction*).

Quanto a *skin* (*XSkin*), os componentes *cross* possuem a capacidade de selecionar os dispositivos por meio de caixas de seleção (*CheckBox*) exibidas em um painel suspenso (*TitledPane*) que possui um ícone para representar a operação de transição (*Image*) e um botão (*Button*) para invocar a ação de transição de interação. Vide Figura 42.

Figura 42 – Camada Skin para os 3 componentes

Fonte: Próprio autor.

A Figura 43 apresenta a interface de *XSkin*, que é herdada para todos os componentes GUI.

Figura 43 – View padrão para todos os componentes

Fonte: Próprio autor

Essas características da Figura 43 foram abstraídas na classe *XSkin* (Figura 42). Cabe a cada componente de UI especializar essa classe para complementar sua *skin*. Por exemplo, a classe *XTextFieldSkin* adiciona um *ComboBox* como sendo a representação de uma caixa de texto para receber as múltiplas respostas. Portanto, o componente *XTextField* é representado de acordo com a imagem da Figura 44 abaixo:

Figura 44 – Componente XTextField

Fonte: Próprio autor

Para o componente *XTextField* também foi implementado um comportamento que permite capturar o conteúdo da área de transferência do dispositivo remoto. Ao pressionar CTRL+SPACE, com o foco na caixa de texto, o conteúdo da área de transferência do dispositivo selecionado é transferido para a caixa. Esse tipo de comportamento representa uma operação de COLAR distribuído.

Outro componente desenvolvido é o *XRadioButton* (Figura 45) que possui uma *skin* que adiciona diversos *RadioButton* (de acordo com o número de opções), barras de progresso e indicadores percentuais que apontam as respostas dos diversos dispositivos. Esse componente permite representar transições 1:1, 1:Nf, 1:Nl, N:1f, N:1l, N:Mfd, N:Mld, N:Ma.

Figura 45 – Componente XRadioButton

Fonte: Próprio autor

Já o componente *XHyperlink* agrega somente um objeto de *hyperlink* à *skin* (Figura 46).

Figura 46 – Componente XHyperlink

Fonte: Próprio autor

O componente *XTooltip*, por sua vez, não possui uma *skin* personalizada que precise dos elementos de *XSkin*, visto que, trata-se de um componente que é associado a outro componente GUI. Portanto, *XTooltip* consegue obter informações de transição por meio do componente com o qual ele está associado. Por exemplo, associar uma dica (*XTooltip*) a uma caixa de texto (*XTextField*), faz com que os dispositivos selecionados em *XTextField* recebam a dica.

Já a classe *XBehavior* é quem abstrai o comportamento padrão dos componentes com a natureza de transição de distribuição da interação. Nessa classe encontram-se a identificação do serviço com o qual o componente interage, a lista de dispositivos de que ele tem conhecimento e a lista de eventos que ele está observando. Para cada dispositivo que o componente aguarda uma resposta, uma subscrição é criada para um determinado evento no dispositivo remoto por meio de especializações de *EventClient*. Quando o dispositivo remoto responde, o método *eventReceived* é disparado por meio de estratégia *push* e o algoritmo de manipulação da *skin* do componente é ativado.

Como operações básicas para todos os componentes que herdam de *XBehavior*, temos:

- o algoritmo de atualização da lista de dispositivos sempre que o painel suspenso é maximizado (*transitionInputPane_OnMouseClicked*);

- o algoritmo de invocação do serviço e inscrição no evento dos dispositivos selecionados (*btnShare_OnAction*) e;
- o algoritmo de seleção de todos os dispositivos encontrados (*allDevices_OnAction*).

A Figura 47 ilustra as relações de *XBehavior* e suas dependências e especializações. Nessa figura é possível observar que a camada *behavior* de *XAudio*, *XHyperlink* e *XMouse* não utilizam eventos. *XAudio* e *XMouse* porque transferem dados por meio de *stream* e *XHyperlink* porque não espera receber uma resposta do dispositivo alvo.

Figura 47 – Camada XBehavior para os 3 componentes

Fonte: Próprio autor

Para uma melhor compreensão do comportamento implementado, a Figura 48 apresenta um diagrama de sequência que mostra o comportamento (*behavior*) do componente *XTextField* quando seus eventos são disparados por ação do usuário ou de entidades externas. Verifica-se que toda vez que a aba com a lista de dispositivos do componente *XTextfield* é maximizada, uma consulta é feita a classe *DPWSCClient* para obter uma lista atualizada de dispositivos. Quando o botão *Share* é acionado, uma das 8 transições possíveis para esse componente é realizada.

Figura 48 – Diagrama de sequência sobre o comportamento de XTextField

Fonte: Próprio autor

A complexidade desse modelo de comunicação distribuída é abstraída pela concepção de um *toolkit* de componentes GUI que permite aos desenvolvedores de sistemas a construção rápida de aplicações que se beneficiam de uma interação distribuída observando as transições de interação já mencionadas. O código abaixo representa a instanciação e configuração de uma caixa de texto.

```
//Instanciação do componente caixa de texto
XTextField xText = new XTextField();
//Definição da mensagem de call-to-action a ser exibida no(s) dispositivo(s) remoto(s)
xText.setCallToAction("What words come to your mind when I say vacation?");
//Definição do tipo de transição
xText.setInteractionTransitionType("N:Ma");
```

Em seguida, a instanciação e configuração de um conjunto de opções selecionáveis remotamente.

```
//Instanciação do componente de radio
XRadioButton xRadio = new XRadioButton();
//Definição da mensagem de call-to-action a ser exibida no(s) dispositivo(s) remoto(s)
xRadio.setCallToAction("Select one of the following options:");
//Definindo quais são as opções selecionáveis para os dispositivos envolvidos
xRadio.setOptions("Option1#Option2#Option3#Option4");
//Definição do tipo de transição
xRadio.setInteractionTransitionType("N:Mfd");
```

Existem cenários em que a interação do usuário ocorre em intervalos muito curtos de tempo ou de forma contínua, como movimentos de *mouse* ou uma interação via áudio. Para essas atividades, ficar enviando mensagens SOAP em determinados intervalos de tempo consome grande largura de banda e prejudica o processo de interação por ocasionar interrupções regulares na interação remota.

Uma estratégia mais adequada é o uso de *stream*. Uma mensagem SOAP é enviada para o dispositivo alvo informando que um volume contínuo de dados deseja ser enviado. Um fluxo é estabelecido e o dispositivo de origem pode continuamente enviar dados nesse fluxo, sem a necessidade de todos os cabeçalhos de uma mensagem SOAP.

Para realização de testes de *stream* e exemplificar o modo de transição 1:0 apresenta-se o componente *XMouse* para controle de um cursor remoto (Figura 49).

Figura 49 – Interface do componente de *mouse* remoto

Fonte: Próprio autor

Esse componente ativa um cursor de *mouse* remoto para o dispositivo selecionado, permitindo que se possa realizar apontamentos a distância (Figura 50).

Figura 50 – Transição de interação do componente XMouse

Fonte: Próprio autor

A pequena elipse azul, com o nome jppreti ao lado, representa o cursor de mouse controlado pelo dispositivo de jppreti, enquanto a elipse marrom, com o nome joapaulo ao lado, representa o cursor de mouse controlado pelo dispositivo de joapaulo.

O rastro deixado por esses cursores ocorre somente quando o botão esquerdo do mouse dos respectivos dispositivos é mantido pressionado, caso contrário há apenas o movimento do cursor para apontamentos. Esse exemplo demonstra a possibilidade de múltiplos cursores controlados por múltiplos dispositivos em uma área visual comum, podendo ser utilizado para pontuar e demarcar qualquer área visível em tela, até mesmo por cima de outros aplicativos.

Houve o desejo de instanciar e utilizar cursores do próprio Sistema Operacional (SO), mas há um problema quanto ao uso de mais de um *mouse* nos sistemas operacionais para PC. As APIs de mouse consideram apenas a existência de um único cursor. Por exemplo, a chamada de função do sistema operacional *Windows* para obter a posição atual do cursor do *mouse* não tem um argumento para o chamador especificar se ele está perguntando sobre o primeiro ou segundo cursor, tampouco o retorno da função fornece um coleção de posições acerca dos cursores existentes, apenas especifica as coordenadas x e y do único cursor do *mouse*.

Apesar dos sistemas operacionais mais modernos estarem suportando o multitoque, o que permite receber mais de uma entrada de tela em paralelo, a realidade do único cursor de mouse permanece. A solução encontrada para superar em parte essa limitação pode estar na virtualização de cursores independentes de *mouse* e utilizar técnicas de preempção do cursor original (*time slice*), mas mesmo assim, para o sistema operacional e os aplicativos que nele executam, as ações serão interpretadas como sendo de um único *mouse*.

Existem extensões para *Windows*, como o *Windows Multipoint Mouse Software Development Kit*, que adicionam suporte para múltiplos cursores, mas o *software* que irá utilizar esse recurso deve ser reescrito para suportar tal acessório. Apenas a instalação da extensão não permite o funcionamento nativo nos aplicativos que são executados no sistema operacional.

Outro componente de transição é o componente *XTooltip* que demonstra a transição N:0, um componente típico de dica (*tip*) ou ajuda contextual, utilizado para que quando o usuário passe o mouse sobre um componente, este exiba uma dica sobre o mesmo, por exemplo, ao passar o mouse sobre uma caixa de texto, exibe-se uma dica sobre o tipo de conteúdo que deve ser preenchido. A diferença do *XTooltip* é que a dica aparece não apenas localmente, mas sim nos dispositivos remotos que estão selecionados para participar do processo de transição. Em seguida, verifica-se como é simples associar uma *XTooltip* a um componente GUI.

```
//Instanciação do componente XTooltip com a dica a ser exibida nos dispositivos
XTooltip tip = new XTooltip("Here goes the tip!");
//Definição de qual componente GUI dispara a dica para os dispositivos remotos
xText.setTooltip(tip);
```

Nesse caso a *tip* foi associada ao objeto *xText* instanciado no exemplo anterior.

Outro componente que também utiliza o recurso de *stream* é o componente de áudio remoto *XAudio* (Figura 51).

Figura 51 – Componente XAudio

Fonte: Próprio autor

Figura 52 – Notificação de captura do áudio

Fonte: Próprio autor

Esse componente permite que ao se selecionar um dispositivo remoto, se possa capturar o áudio proveniente do microfone desse dispositivo.

O experimento foi realizado em um cenário no qual a usuária Alice_{PC}, deseja capturar o áudio de Bob_{SP}. A Figura 52 ilustra o *smartphone* de Bob informando-o de que alguém solicitou o seu *stream* de áudio. A aplicação *XDevice* inicia a captura automaticamente, necessitando de uma intervenção de Bob no caso de ele não autorizar.

Observe-se que na implementação da prova de conceito, não foi feita uma compressão do áudio (formato PCM) e os testes ocorreram em uma rede 10/100 Wi-Fi com um atraso inicial no áudio de aproximadamente 5 segundos. No entanto, este atraso não comprometeu a prova de conceito, na qual desejava-se mostrar que é possível enviar uma grande quantidade de dados sob a pilha DPWS e realizar *stream* não somente do cliente para o dispositivo alvo (*Target Service*), como no caso do *XMouse*, mas também do dispositivo alvo para o cliente.

5.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi possível verificar o suporte tecnológico utilizado, bem como a implementação dos componentes que exemplificam as transições de distribuição da interação. As transições foram atendidas pelos seguintes componentes:

- 1:0 – *XTooltip*, *XHyperlink* e *XMouse*
- 1:1 – *XAudio*, *XTextfield* e *XRadioButton*
- 1:Nf – *XTextfield* e *XRadioButton*
- 1:NI – *XTextfield* e *XRadioButton*

- N:0 – *XTooltip* e *XHyperlink*
- N:1f – *XTextfield* e *XRadioButton*
- N:1l – *XTextfield* e *XRadioButton*
- N:Mfd – *XTextfield* e *XRadioButton*
- N:Mld – *XTextfield* e *XRadioButton*
- N:Ma – *XTextfield* e *XRadioButton*

Foi possível também verificar o uso de *stream* para o estabelecimento de pontes de transição tanto do cliente para o dispositivo alvo como no sentido contrário.

Uma reflexão pode ser construída acerca dos desafios de se estabelecer uma comunicação entre componentes GUI para *web* com dispositivos DPWS.

Foi possível verificar também o quanto é simples a instanciação e uso desses componentes por desenvolvedores finais da aplicação. No entanto, houve um enorme esforço de desenvolvimento para que o uso dos componentes alcançasse essa simplicidade. Houve a necessidade de implementar classes utilitárias para acesso aos recursos dos dispositivos, desenvolver diversos serviços sob a pilha DPWS e implementar componentes GUI que usufruem desses serviços. Todas essas implementações em mais de uma plataforma, envolvendo ambiente *web*, *desktop* e *mobile*.

Verifica-se, portanto, a existência de dois tipos de desenvolvedores, àqueles que desenvolvem os componentes GUI baseados na arquitetura (que encapsulam a complexidade de implementação em componentes GUI) e os desenvolvedores da solução final, que fazem uso dos componentes desenvolvidos pelos primeiros desenvolvedores. Essa divisão é salutar, visto que, permite a um grupo de desenvolvedores concentrar-se nos aspectos de distribuição da interação, enquanto outro grupo pode se concentrar em aspectos particulares da solução final. Complementa-se como aspecto positivo o fato de que os componentes criados pelo primeiro grupo, podem ser reutilizados por várias aplicações.

6 UMA ARQUITETURA MAIS AMPLA

Conforme os requisitos descritos no Quadro 3 e no Quadro 4, uma arquitetura que apoie a interação distribuída deve levar em consideração diversos aspectos que não necessariamente podem ser suportados pela especificação DPWS.

Portanto, este capítulo vem descrever modelos e hipóteses a serem consideradas de forma a constituir um modelo arquitetural mais amplo. Não é realizada mais nenhuma prova de conceito, sendo que este capítulo ajuda a compor uma agenda de pesquisa para os próximos anos.

Essa arquitetura mais ampla foi intitulada DISS (*Distributed Interaction Support Service*). Trata-se de um conjunto de serviços que tem o propósito de intermediar a comunicação entre múltiplos dispositivos. O DISS especifica um conjunto de serviços necessários para que a interação possa ocorrer entre múltiplos dispositivos de forma sequencial ou simultânea, sendo estes:

- **Interaction Modality Mapping Service (IMMS)**: responsável por mapear a modalidade de interação com a interface do serviço remoto;
- **Device Management Service (DMS)**: responsável por registrar os dispositivos de interesse, permitindo agrupamento, busca, classificação e adição de metadados;
- **Context Support Service (CSS)**: Responsável por gerenciar o contexto do usuário.
- **User Management Service (UMS)**: responsável por registrar o usuário e associar ao dispositivo;
- **Transition Support Service (TSS)**: responsável pelas operações de transição.

A Figura 53 apresenta os serviços que compõem a arquitetura DISS.

Figura 53 – Serviços que compõem o DISS

Fonte: Próprio autor

A arquitetura DISS procura seguir a especificação DPWS como base, mas não se restringe a ela em vista das limitações da especificação DPWS em atender aos demais requisitos de interação distribuída.

A seguir são apresentados os 22 requisitos do Quadro 3 e do Quadro 4. Os requisitos 6, 8, 14, 16 e 20 ainda não estão sendo atendidos pela arquitetura proposta.

Requisito 01 - Prover mecanismos para suportar transições entre canais de comunicação mais utilizados.

Este requisito foi atendido pelo TSS para **transições da distribuição da interação** em dispositivos como *desktop*, *tablet* e *smartphone* baseado na perspectiva de que dispositivos atuam no formato DaaS.

No trabalho de Miyamaru (2010) este requisito foi atendido para **transições entre mídias**, utilizando estratégias como QR-Code, *click to call* (como os utilizado pelo Skype) e envio de *hyperlinks*. Essas estratégias tinham por objetivo guiar o usuário para o próximo dispositivo dentro do escopo de uma narrativa. Essa abordagem traz como vantagem o uso de recursos já consolidados nas mídias para estabelecer as pontes da transição. No entanto, essas estratégias são limitadas quanto ao

transporte de dados e não permitem transferir todo o processo de interação previamente ocorrido. Com vistas a superar essas limitações, a arquitetura de Miyamaru centralizou o gerenciamento da transição, criou componentes de transição (*QR-Code*, *click to call*, ...) para cada plataforma e meios para definição de uma sessão comum entre os dispositivos PC, celular e TV. Já o Serviço de Apoio a Contexto (CSS), proposto nesta pesquisa, fornece uma alternativa para compartilhamento da sessão entre dispositivos utilizando um padrão aberto e baseado em *Web Services*. Este serviço permite que objetos de um dispositivo possam ser persistidos, serializados e instanciados em outro dispositivo.

Com relação a **transição entre usuários**, vislumbra-se a necessidade de que componentes GUI possam migrar entre as plataformas. O CSS e o Serviço de Gerenciamento de Usuários (UMS) servem de suporte para que esse processo possa ocorrer, mas este processo carece ainda de um aprofundamento dos estudos.

Requisito 02 - Prover interfaces reutilizáveis de componentes de transição que ocultam detalhes da implementação, enquanto permitem ao designer customizar a interface do usuário.

A estrutura do TSS e a implementação dos componentes (*XTextField*, *XRadioButton*, *XMouse*, *XTooltip* e *XAudio*) em JavaFX já representam o atendimento a esse requisito. No entanto, existe a necessidade de continuar a desenvolver outros componentes típicos (por exemplo, tabelas e imagens.).

Requisito 03 - Prover recursos para automatizar os mecanismos de transição tanto quanto possível.

Vários elementos da arquitetura contribuem no atendimento a este requisito:

- O uso de uma arquitetura orientada a serviços e uso padrões abertos de comunicação;
- Arquitetura de rede P2P fornece independência ao dispositivo e permite estabelecimento de relações espontâneas;
- Modelo de comunicação dirigido a eventos permite que notificações sejam enviadas de forma assíncrona sobre mudanças de estado;
- Sistematização das transições de distribuição da interação, conforme Quadro 7;

- Disponibilização de componentes GUI que já permitem a realização das transições do Quadro 7.

Requisito 04 - Apresentar dados relacionados à transição quando solicitados pelo usuário.

A arquitetura TSS, por meio do serviço *TransitionSupportService* e das classes *DPWSCClient* e *EventClient*, provê informações relacionadas ao *status* da comunicação com os dispositivos, mas existe a necessidade de implementação de componentes de *status* que possam ser associados a outros componentes de transição para atender esse requisito.

Requisito 05 - Prover recursos para informar o usuário de possíveis erros que impossibilitem os mecanismos de transição automatizados de operar corretamente.

O diagrama de estado dos componentes GUI prevê diversas situações de falha que notificam o desenvolvedor, sendo este responsável por tratar e decidir qual a forma desejada de apresentar a falha.

Requisito 06 - Prover os usuários com recursos para desfazer uma transição.

Este requisito ainda não é atendido pela arquitetura proposta. Para que este requisito seja atendido faz-se necessário registrar o histórico das transições disparadas por um dispositivo e guardar o estado original dos objetos de interface gráfica.

Requisito 07 - Ser flexível para suportar a adição de novos tipos de transição.

O uso de *Web Services*, padrões abertos de comunicação, a composição dos serviços com baixo acoplamento, de forma autônoma, utilizando uma arquitetura que permite o relacionamento de serviços *on-the-fly* contribuem para o atendimento deste requisito.

Requisito 08 - Manter registro da navegação do usuário através dos canais.

Este requisito ainda não é atendido pela arquitetura proposta. Este requisito permite rastrear a ação das transições, necessário para atender o requisito 06, além de

permitir realizar uma auditoria das atividades realizadas em um determinado dispositivo.

Requisito 09 - Manter registro dos dispositivos do usuário.

Este requisito é atendido por meio do Serviço de Gerenciamento de Dispositivos (DMS), apresentado na seção 6.1.2.

Requisito 10 - Reconhecer seus usuários.

Este requisito exige que exista um sistema de registro e identificação de usuários, visto que dispositivos podem ser compartilhados entre usuários e pode haver restrições de uso. Atividades de transição entre usuários também requerem a identificação dos usuários. Assim, este requisito é atendido por um **Serviço de Gerenciamento de Usuários (UMS)**, responsável por mapear usuários a dispositivos, bem como informações de contexto sobre essa relação de uso. Vide seção 6.1.2 para maiores detalhes.

Requisito 11 - Permitir que interações em um dispositivo sejam delegadas a usuários em outros dispositivos.

Esse requisito é atendido pelo TSS por meio da transição de distribuição da interação.

Requisito 12 - Permitir que recursos do dispositivo sejam configurados para permitir ou não o acesso concorrente.

Esse requisito é atendido pela estrutura da classe abstrata de serviços *XService*, que possui operações para bloqueio (*lockOperation*), desbloqueio (*unlockOperation*) e verificação do *status* do bloqueio (*isOperationLocked*).

Requisito 13 - Permitir que mais de um dispositivo seja utilizado pelo mesmo usuário de forma simultânea e sincronizada.

A arquitetura DPWS mostrou que atende esse requisito por meio do uso de *stream* do serviço *MouseService*. O componente *XMouse*, no cenário Alice_{PC} →_{1:0} Bob_{PC}, demonstrou que é possível na arquitetura atuar em dois dispositivos simultaneamente.

Requisito 14 - **Permitir a restrição de acesso de um ou mais dispositivos a um determinado usuário ou contexto.**

Este requisito ainda não é atendido pela arquitetura proposta.

Requisito 15 - **Permitir que mais de uma modalidade de interação seja utilizada para interagir com um determinado recurso do dispositivo.**

Esse requisito é atendido pelo **Serviço de Mapeamento de Modalidade de Interação (IMMS)**, responsável por mapear a modalidade de interação utilizada com a interface do serviço remoto que se deseja utilizar. Vide seção 6.1.1, para maiores detalhes.

Requisito 16 - **Converter conteúdos não suportados por um determinado dispositivo.**

Este requisito ainda não é atendido pela arquitetura proposta.

Se vislumbra a necessidade de um **Serviço de Conversão de Conteúdo (CCS)**, responsável por mapear os diferentes formatos de conteúdo e respectivos conversores. Este serviço deve registrar também os custos de conversão para cada formato de forma a otimizar o processamento da conversão.

Requisito 17 - **Permitir que metadados possam ser agregados ao conjunto de informações do dispositivo.**

Este requisito é atendido pelo **Serviço de Gerenciamento de Dispositivos (DMS)** e pela especificação *WS-MetadataExchange*, responsável por fornecer meios de registrar dispositivos, respectivos metadados e fornecer mecanismos de busca. Vide seção 6.1.2.

Requisito 18 - **A arquitetura deve permitir que dispositivos sejam agrupados ou classificados conforme interesse.**

Este requisito é atendido pelo **Serviço de Gerenciamento de dispositivos (DMS)**. Vide seção 6.1.2.

Requisito 19 - Permitir a descoberta de dispositivos de forma *ad hoc*.

A arquitetura atende este requisito por meio das especificações *WS-Discovery*, *WS-MetadataExchange* e pela classe *DPWSCClient* da arquitetura.

Requisito 20 - Facilitar a realização de testes de comunicação com os dispositivos.

Este requisito ainda não é atendido pela arquitetura proposta.

Vislumbra-se a necessidade de seu atendimento por meio de um Serviço de Interface de Gráfica Padrão (DGUIS). O DGUIS deve fornecer uma GUI padrão com elementos mínimos para cada tipo de dispositivo, este serviço deve fornecer uma interface para teste entre dispositivos presentes na rede. Aqui visualiza-se uma interface gráfica com operações mínimas implementadas por tipo de dispositivo para realização de testes de comunicação, como um *explorer* para dispositivos DPWS.

Requisito 21 - Permitir que novas aplicações, em outras plataformas, possam ser facilmente integradas.

O uso de *Web Services*, padrões abertos de comunicação, a composição dos serviços com baixo acoplamento, de forma autônoma, utilizando uma arquitetura que permite o relacionamento de serviços *on-the-fly* contribuem para o atendimento deste requisito.

Requisito 22 - Permitir que a comunicação entre dispositivos possa ocorrer de forma segura.

A especificação *WS-Security*, presente na pilha DPWS, apresenta um protocolo que especifica como integridade e confidencialidade podem ser garantidas e permite o uso de diversos formatos de segurança, como *Security Assertion Markup Language (SAML)*, *Kerberos* e *X.509*. Essa especificação é baseada na *XML Signature* e *XML Encryption* para o uso de certificados digitais e criptografia, respectivamente (*WS-SECURITY*, 2006).

Apesar da DPWS possuir uma pilha de protocolos que auxilia a arquitetura DISS, ela não é completa para atender a todos os requisitos. Portanto, acredita-se que a melhor solução, do ponto de vista imediato, seja uma **arquitetura**

híbrida. Como a DPWS é baseada em *Web Services*, é possível a existência de um *broker* que forneça acesso a recursos centralizados que auxiliem no cumprimento de requisitos não atendidos pelo dispositivo. Não é porque o modelo de comunicação direta foi utilizado que o modelo indireto não possa ser utilizado para atender requisitos não suportados pela pilha DPWS.

As especificações evoluem, esta pesquisa serve de base para que grupos que as definem possam vislumbrar novas demandas. Acredita-se que a especificação DPWS deverá incorporar outras especificações WS com o surgimento de novas demandas e com o aumento das capacidades de memória, processamento e largura de banda dos dispositivos.

As seções que seguem detalham melhor cada serviço.

6.1.1 Serviço de mapeamento de modalidade de interação (IMMS)

Dada as diferentes modalidades de interação para cada tipo de dispositivo, visualiza-se uma organização desses serviços conforme seu tipo. As modalidades de interação existentes para um celular podem ser muito diferentes daquelas encontradas em uma TV. Existem celulares que permitem a interação por meio de teclado, voz, câmera de vídeo, tela multitoque e por meio de sensores, como o acelerômetro e o de proximidade. Já na TV a interação ocorre quase sempre pelo teclado do controle remoto.

O IMMS é o serviço responsável por registrar como uma determinada interação, realizada em um determinado dispositivo, invoca os serviços ofertados por outro dispositivo. Para que isso seja possível, o IMMS precisa ter registrado os eventos que podem ser lançados e funcionar como um adaptador para a correta invocação do serviço desejado. A relação entre as ações e os serviços pode ter cardinalidade muitos para muitos. O funcionamento do IMMS foi aceito para publicação no DUI 2014 por meio de artigo intitulado *Interaction Modality Mapping Service for devices in a P2P network*.

Como exemplo, apresenta-se a Figura 54. Nela, visualiza-se um dispositivo *smartphone* interessado nos serviços de um outro dispositivo qualquer. As ações executadas no *smartphone* representam a interação do usuário com esse dispositivo, sendo que essas ações disparam eventos que são interceptados pelo IMMS, que é responsável por interpretar o evento e invocar corretamente o serviço

desejado em outro dispositivo. Uma ação do usuário pode ser, por exemplo, um gesto multitoque de “ampliar”. Essa ação pode estar configurada para aumentar o volume da TV, ou dar um *zoom* em um *display*, ou aumentar a aceleração de um motor.

Figura 54 – Exemplo do papel executado pelo IMMS

Fonte: Próprio autor

O Quadro 9 apresenta um exemplo hipotético de algumas possibilidades de interação de um usuário utilizando o *smartphone* com os serviços oferecidos por uma TV e vice-versa.

Quadro 9 – Interação entre Smartphone e TV

SMARTPHONE	TV
Afastando dois dedos	Aumentar o volume
Aproximando dois dedos	Diminuir o volume
Arrastar um dedo para a direita	Avançar um canal
Arrastar um dedo para a esquerda	Retroceder um canal
Duplo toque na tela	Mudar formato da tela (16:9 ; 4:3)
Arrastar três dedos para baixo	Apresentar grade de programação
TV	CELULAR
Pressionar seta para cima	Ir para o contato anterior
Pressionar seta para baixo	Ir para o próximo contato
Pressionar enter/ok	Exibe dados do contato
Pressionar info	Exibe localização do celular

Fonte: Próprio autor

Na parte inferior da Figura 54 também é possível observar a existência de um serviço de relação (*Relation Service*), este se faz necessário para permitir que

relações entre serviços sejam constituídas, como forma de possibilitar o encadeamento de um conjunto de dispositivos em uma interação. A chamada a um serviço pode desencadear uma chamada a outro serviço, permitindo que uma ação tenha efeito em mais de um serviço (presente no mesmo dispositivo ou não), causando um efeito cascata. Aqui se visualiza a possibilidades de se utilizar mensagens SOAP sobre UDP para eventos que podem ser disparados de forma contínua ou estratégia de *stream*. A especificação DPWS permite enviar mensagens SOAP sobre UDP ou TCP e também a realização de *stream*, o que nos dá flexibilidade de categorizar os eventos de acordo com a sua natureza e utilizar um protocolo ou estratégia mais otimizada caso necessário.

A especificação WS-Eventing permite criar topologias complexas para o envio e tratamento dos eventos, permitindo que a origem do evento e o receptor final do evento estejam desacoplados, além de permitir estender, caso necessário, cenários para tratamento de eventos não previstos.

Uma outra característica do IMMS é restringir ou não o acesso de mais de um dispositivo a um determinado dispositivo alvo ou serviço por meio de bloqueios como já exposto no TSS.

A Figura 55 apresenta o modelo de classes do IMMS.

Figura 55 – Modelo de classes do IMMS

Fonte: Próprio autor

A operação *mapModality* permite resgatar os formatos de modalidade registrados em um arquivo de configuração para o serviço que se deseja acessar.

As classes *ModalityEventClient* e *ModalityStreamClient* representam o processamento da interação, seja na forma de eventos, seja na forma de *stream*, respectivamente.

Já a classe *Interpreter* é responsável por armazenar e recuperar a estrutura de configuração do mapeamento da modalidade de interação, conforme exemplo da Figura 56.

Nesse documento é possível verificar que as operações *setMouse* (que envia dados do *mouse* via *stream* para outros dispositivos) do serviço de *mouse* (*MouseService*) e *okEvent* (que envia dados digitados no teclado quando pressionado OK no *smartphone* ou ENTER no *desktop*) do teclado (*KeyboardService*), estão sendo mapeadas para serviços de dispositivos TV. É possível verificar também que três eventos de *mouse* e um de teclado foram mapeados:

- clique simples do botão esquerdo (*LEFT_ONE_CLICK*): este invoca a operação *nextChannel* da TV uma única vez (*occurrence value="1"*);
- clique simples do botão de rolagem (*SCROLL_ONE_CLICK*): este invoca a operação *showInfo* da TV uma única vez (*occurrence value="1"*);
- manter pressionado do botão esquerdo (*LEFT_PRESSED*): este pode invocar a operação *volumeUp* diversas vezes (*occurrence value="*"*). O número de chamadas a operação *volumeUp* vai depender da variação positiva do eixo x do *mouse*;
- digitar CHN: este invoca a mudança de canal da TV (*setChannel*), o argumento numérico informado após o comando CHN define o novo valor do canal.

Figura 56 – Documento XML contendo mapeamento das modalidades de interação e serviços

```

<multimodalDevices>
  <TV>
    <br.ufmt.xdevice.MouseService>
      <operation name="setMouse">
        <type>stream</type>
        <event type="LEFT_ONE_CLICK">
          <service>br.ufmt.xdevice.TVService</service>
          <operation name="nextChannel">
            <occurrence value="1" />
          </operation>
        </event>
        <event type="SCROLL_ONE_CLICK">
          <service>br.ufmt.xdevice.TVService</service>
          <operation name="showInfo">
            <occurrence value="1" />
          </operation>
        </event>
        <event type="LEFT_PRESSED">
          <condition attribute="x" operator="GREATER_THAN" value="0">
            <service>br.ufmt.xdevice.TVService</service>
            <operation name="volumeUp">
              <occurrence value="*" definedBy="x" />
            </operation>
          </condition>
        </event>
      </operation>
    </br.ufmt.xdevice.MouseService>

    <br.ufmt.xdevice.KeyboardService>
      <operation name="OkEvent">
        <type>event</type>
        <event type="CHN">
          <service>br.ufmt.xdevice.TVService</service>
          <operation name="setChannel">
            <occurrence value="1" />
          </operation>
        </event>
      </operation>
    </br.ufmt.xdevice.KeyboardService>
  </TV>
</multimodalDevices>

```

Fonte: Próprio autor

O diagrama de sequência da Figura 57 apresenta um exemplo de uso do IMMS de forma dirigida a eventos.

Figura 57 – Diagrama de sequência mostrando o uso do IMMS por eventos

Fonte: Próprio autor

Nesse mapeamento, o serviço de teclado é mapeado para interagir com a TV por meio de comandos de texto. Já a Figura 58 apresenta um exemplo de uso do IMMS por meio de *stream*.

Nesse outro exemplo o serviço de *mouse* é mapeado para interagir com a TV por meio de ações de clique do mouse. As ações de teclado e *mouse* dos dois exemplos podem ser averiguadas no documento XML descrito anteriormente.

Figura 58 – Diagrama de sequência mostrando o uso do IMMS por *stream*

Fonte: Próprio autor

6.1.2 Serviço de Gerenciamento de usuários e dispositivos (UMS e DMS)

O UMS permite que se faça o registro dos usuários.

O UMS possui operações que permitem (Figura 59):

- cadastrar um usuário;
- remover um usuário;
- descobrir quais são os usuários do dispositivo;
- efetuar *login*;
- efetuar *logout*;
- descobrir qual o usuário logado; e
- notificar dispositivos interessados sobre mudanças de *status* do usuário.

O DMS, por sua vez, permite que se faça o registro dos dispositivos de interesse na rede. Esse registro deve conter outras informações pertinentes ao dispositivo que porventura não estejam disponíveis de forma nativa. Uma informação como a localização geográfica do dispositivo deve ser informada para dispositivos não móveis. Além disso, o DMS permite organizar os dispositivos registrados em grupos que denotem alguma classificação de interesse. O registro de dispositivo e grupo também se faz necessário para que se possa estabelecer relações com o usuário ou grupos de usuário, facilitando a visibilidade dos dispositivos pertinentes ao contexto do usuário e restringindo a visibilidade a serviços não autorizados.

A Figura 59 apresenta o modelo de classes do DMS e do UMS, visto que se faz necessário que usuários e dispositivos estejam relacionados.

Figura 59 – Modelo de classes do DMS e do UMS

Fonte: Próprio autor

O DMS possui operações que permitem:

- adicionar elementos e atributos ao dispositivo, para descrição de metadados;
- adicionar dispositivos conhecidos;
- descobrir contextos conhecidos;
- descobrir todos os dispositivos conhecidos ou por contexto;

- descobrir dispositivos desconhecidos, mas que estão acessíveis na rede.

O DMS faz acesso ao UMS para que a adição de dispositivos conhecidos e de contextos esteja associada ao usuário logado.

O diagrama de sequência da Figura 60 ilustra como é realizada a descoberta de dispositivos desconhecidos para conhecimento do usuário e registro do mesmo. Basicamente, o *DeviceManagementService* consulta o *UserManagementService* para verificar qual o usuário logado e resgata quais são os dispositivos já registrados por esse usuário. Dispositivos que estejam disponíveis na rede, e que não fazem parte do registro do usuário, são apresentados na aplicação *XDevice*. Dessa forma o usuário pode reconhecer novos dispositivos e adicioná-los se assim desejar.

Figura 60 – Diagrama de sequência para adição de novos dispositivos descobertos

Fonte: Próprio autor

6.1.3 Context Support Service (CSS)

O CSS tem por responsabilidade manter uma única sessão para o usuário, mesmo que ele esteja deixando de interagir com um dispositivo para

interagir com outro. Essa característica é imprescindível para que o usuário tenha a sensação de continuidade, dando a impressão de que um dispositivo é extensão do outro (*seamlessness*).

Para que essa característica seja alcançada, faz-se necessário que o estado dos objetos da origem e do destino seja mantido. A manutenção do estado dos objetos em uma aplicação *stand-alone* ou nativa já é algo comum. Mesmo no ambiente *Web*, no qual o protocolo HTTP é *stateless* (não mantém o estado da conexão), os servidores de aplicação Web preservam o estado dos objetos pelo conceito de sessão.

Apesar disso, a sessão implementada pelos servidores de aplicação se restringe a manter o estado dos objetos no lado servidor, não mantendo o estado dos objetos da interface gráfica, por exemplo. Outro problema é que a sessão é única por instância de navegador, ou seja, quando Alice interagir com um sistema em dois dispositivos diferentes, o Alice_{SP} e um Alice_{PC} por exemplo, terá duas sessões distintas.

Dado o problema exposto, faz-se necessário a existência do CSS para:

- sincronizar sessões distintas de um mesmo usuário e;
- registrar a navegação e estado dos objetos da interface gráfica.

Porém, a especificação DPWS não possui nenhum recurso que atenda a esses requisitos. Neste caso, uma alternativa possível é uma arquitetura cliente-servidor para estender as capacidades do dispositivo por meio de um serviço centralizado que implemente as especificações *WS-Context* e *WS-Coordination*. Outra alternativa é a composição de funcionalidade similar em uma arquitetura P2P. A vantagem no uso dessas especificações é que elas já são um padrão, mas não foram inseridas na especificação DPWS, sendo utilizadas nas aplicações corporativas.

A especificação *WS-Context* fornece uma definição, um mecanismo e definições de serviço para organizar e compartilhar contextos por meio de vários *endpoints*. A especificação também fornece serviços para gerenciar protocolos orientados a sessão que representam as atividades correspondentes a esses contextos. A propagação do contexto é possível utilizando diferentes protocolos daqueles utilizados pela aplicação.

Já a especificação *WS-Coordination* descreve um *framework* extensível para disponibilizar protocolos que coordenam as ações de aplicações distribuídas. De uma forma geral, coordenação é o ato de uma entidade (conhecida como coordenadora) disseminar informações para um conjunto de participantes de um domínio específico. A especificação *WS-Coordination* por si só não define todos os requisitos necessários para uma solução completa, trata-se de uma referência que é utilizada em conjunto com outras especificações, como a *WS-Context*, e protocolos específicos da aplicação para acomodar uma ampla variedade de protocolos relacionados com a exploração de *Web Services*. A especificação *WS-Coordination* provê extensibilidade e flexibilidade em duas dimensões. A especificação permite (WS-COORDINATION, 2009):

- a publicação de novos protocolos de coordenação; e
- a seleção de um protocolo a partir de um tipo de coordenador e a definição dos elementos de extensão que podem ser adicionados aos protocolos e ao fluxo da mensagem.

Essa extensibilidade e flexibilidade permitem a definição e construção de um protocolo que possibilite o compartilhamento do contexto de interação do usuário. A Figura 61 ilustra duas aplicações com seus respectivos coordenadores, estabelecendo uma comunicação utilizando o protocolo Y.

Basicamente o App1 solicita a criação de um contexto (Q), recebendo de volta um identificador da atividade que é enviado para o App2. O App2 também cria um contexto com a mesma identificação de atividade de App1, o que permite estabelecer uma conexão lógica entre os contextos. Os coordenadores de App1 e App2 podem então estabelecer uma conexão de contexto para uma mesma atividade em duas aplicações.

Figura 61 – Duas aplicações com seus respectivos coordenadores

Fonte: (WS-COORDINATION, 2009)

Esse modelo de conversão entre aplicações permite também que propriedades de *cross-application* (XA) sejam alcançadas. Por exemplo, um cenário no qual um usuário inicia um processo de compra online em uma empresa. Ele pode querer migrar para outra empresa, utilizando o mesmo carrinho de compras e continuar o processo de seleção de produtos. Ao final, ele poderia navegar até uma empresa responsável pelo gerenciamento de pagamentos (Paypal, Pagseguro, ...), realizando uma única vez a operação de pagamento. Dados da identificação do usuário, das empresas envolvidas, dos produtos e do código de pagamento efetuado poderiam estar sendo compartilhados entre as empresas dentro de um contexto comum. Ao final da operação, as empresas poderiam realizar o faturamento e a empresa de gerenciamento de pagamento dividir a transferência financeira.

É importante se ater a essas especificações para que se possa aproveitar toda uma infraestrutura aberta e padronizada.

6.1.4 DISS Application Program Interface (API)

Esta se trata de biblioteca que facilite aos programadores o desenvolvimento de aplicações que utilizam arquitetura DISS na plataforma de interesse. Foram implementados componentes GUI para a rápida instanciação e uso de transições de distribuição da interação. O esforço de desenvolvimento desses componentes para várias plataformas é grande, mas a granularidade também é

grande. É possível delegar a tarefa de desenvolver um único componente para uma única plataforma para um único desenvolvedor. É possível abrir projetos paralelos da mesma arquitetura para cada plataforma, a fim de receber contribuições de implementação do mundo todo. Existem vários projetos *open source* que cresceram seguindo essa prática. O próprio *framework* JMEDS possui implementações da DPWS em C, C++, Java e Android e recebe contribuições de desenvolvedores ao redor do mundo.

Essa atividade ainda não foi iniciada. Faz-se necessário que todos os componentes GUI estejam especificados e validados em uma plataforma para garantir uma interface de componente bem definida a fim de ser replicada para outras plataformas.

7 REFLEXÕES SOBRE DUI

A arquitetura e os componentes apresentados neste trabalho propiciam a construção rápida de soluções que exploram a interação entre múltiplos dispositivos e múltiplos usuários de forma espontânea conforme dispositivos entram e saem da rede. A partir das contribuições obtidas é possível iniciar diversas atividades em paralelo com vistas a ampliar o número de serviços, de componentes, de ensaios com desenvolvedores e usuários finais e realização de estudos de caso. A natureza do fraco acoplamento entre os serviços da arquitetura permite essa organização de desenvolvimento. Cabe ressaltar que o esforço de implementar serviços dos recursos do dispositivo, de interação, do serviço de transição e dos componentes para várias plataformas é grande, mas há diversos pontos positivos que amenizam esse custo:

- há um crescimento no uso de *Web Services* nos dispositivos; os serviços de recursos do dispositivo deverão ser fornecidos de fábrica;
- os serviços produzidos são reutilizáveis; produzir serviços de interação para Android atinge uma grande gama de dispositivos;
- os componentes produzidos são reutilizáveis e servem para produzir inúmeras aplicações;
- desenvolvedores finais da aplicação podem utilizar estratégias de *drag and drop* para rapidamente construir aplicações DUI e exercitar diversas situações de uso.

Por meio desse amadurecimento vislumbra-se a necessidade de participar de comitês abertos junto a OASIS com vistas a apresentar essas novas demandas de serviço. Apesar da DPWS ter sido concebida originalmente para o domínio da automação industrial, do entretenimento doméstico e para sistemas automotivos e de telecomunicação, verifica-se que é possível ampliar seu uso para DUI. Dispositivos como celular, *tablet*, mesas interativas e televisores já possuem poder de processamento e memória capazes de suportar outras especificações WS que estendem as capacidades de uma interação distribuída. Dada a diversidade de dispositivos e as capacidades dos mesmos, a especificação DPWS poderia ser composta em duas versões: uma para dispositivos com recursos limitados de memória e processamento (como sensores e dispositivos atuadores) e outra para

dispositivos com maior poder de processamento (como celulares, *tablets*, mesas interativas, TV).

Apesar dos esforços já realizados, há diversos desafios a serem enfrentados quando se discute a integração sinérgica de múltiplas mídias ou dispositivos.

Do ponto de vista da engenharia de *software* esses desafios permeiam todo o ciclo de desenvolvimento de *software*. Dado que nesse contexto o número de usuários e dispositivos pode crescer significativamente e que o contexto de execução pode ser bastante diversificado e dinâmico, as possibilidades de interação podem ser muito grandes, criando um conjunto particular de desafios.

Um desafio está no **levantamento e especificação de requisitos**, faltam pesquisas que compreendam melhor o fenômeno da interação em múltiplas mídias e que apresentem metodologias para o levantamento e especificação de requisitos nesse modelo de interação.

Outro problema está relacionado às tecnologias que dão suporte ao **ambiente de desenvolvimento** de *softwares* dessa natureza. As soluções encontradas ainda são incipientes e carecem de compatibilidade com os conceitos já apresentados. Em função desse desafio, padrões de projeto para atender requisitos dessa natureza ainda não estão definidos.

Outro desafio está na definição de **padrões de projeto** de *software*. Padrões de projeto surgem quando verifica-se a existência de problemas recorrentes nas estruturas de *software*. Conforme o número de aplicações DUI aumentar será mais fácil identificar esses padrões. O conceito de transição de distribuição da interação e sua sistematização foi um esforço nesse sentido, mas só se tornará padrão se seu uso for recorrente em outras aplicações DUI.

Outro desafio encontrado reside na área de **teste de software**. Como planejar e executar testes para uma quantidade de contextos, porventura, muito grande? A relação não é apenas um usuário em um único dispositivo e os dispositivos possuem diferentes capacidades e sistemas operacionais. Nesse sentido, dada a diversidade de plataformas que podem estar envolvidas no processo, como automatizar a execução dos testes de forma integrada?

Outro desafio é a necessidade de estabelecer e manter um **contexto** comum, permitindo que as atividades de um usuário em múltiplos dispositivos, que

podem envolver mais de uma aplicação possam definir um contexto comum de execução.

Outro desafio a ser enfrentado está no tratamento da multiplicidade de formatos de apresentação de **conteúdo** em função dos diferentes tamanhos e resoluções de tela desses dispositivos. Além disso, cada dispositivo, cada mídia, tem sua própria linguagem, levando a preocupação de como tratar a distribuição e adaptação do conteúdo a ser apresentado. Por outra, o conteúdo pode ser segmentado para ser distribuído ou replicado em outros dispositivos, levando a preocupações sobre como deve ocorrer a decomposição e recomposição desse conteúdo.

Outro desafio é o da **simultaneidade**. Quando e como um usuário pode interferir no conteúdo de outro usuário? Problemas de transação podem surgir, como por exemplo, o ato de desfazer ações realizadas pelo usuário em componentes de interface distribuídos.

Outro desafio reside no **modelo de computação distribuída** a ser utilizado. Este modelo não tem sido um consenso, pois abordagens utilizando o modelo cliente-servidor e P2P são encontradas. Na área de mídias cruzadas, as pesquisas indicaram um equilíbrio quanto a essa escolha, na área de computação ubíqua/pervasiva existe uma tendência para P2P e em DUI, ao contrário, uma tendência para o cliente/servidor. Essas divergências ocorrem pela priorização de determinados requisitos, como escalabilidade de recursos computacionais, segurança e facilidade de implantação e manutenção dos sistemas, mas acredita-se que uma abordagem híbrida possa trazer benefícios das duas arquiteturas.

Outro desafio encontra-se nos **componentes de interface gráfica**. Os componentes GUI tradicionais não comportam um modelo de interação distribuída no qual múltiplos usuários podem interagir com o componente. Essa mudança de paradigma leva a problemas de interação com serviços de interface gráfica nativa dos sistemas operacionais que não comportam esse modelo de interação.

Outro desafio está na **composição dos serviços** dos dispositivos para que a interação ocorra de forma automática e natural (*seamless*). Existe uma carência de padrões ontológicos que permitam uma associação semântica. Dessa forma componentes GUI poderiam dinamicamente descobrir e associar-se aos

serviços de recursos do dispositivo e de interação do TSS para estender o alcance de suas funcionalidades.

Outro desafio encontrado reside no uso da **nuvem**. Observando a necessidade de compartilhamento de informação, escalabilidade e adoção de padrões abertos de comunicação, é que pesquisadores começam a estudar a nuvem como plataforma tecnológica. Os sistemas em plataforma *Web* estão migrando para o modelo computacional em nuvem em um formato orientado a serviço, no qual existe a ilusão de infinitos recursos computacionais, mas diversos ainda são os desafios para suportar as características de mídia cruzada e interfaces distribuídas nesse modelo computacional. Como dispositivos podem ser conectados a um contexto de nuvem? Um protocolo de descoberta *cloud based* pode ser um interessante tópico de pesquisa.

Outro desafio está relacionado ao **design de interação** para componentes GUI em DUI. Estudos precisam ser realizados com vistas a entender melhor como ocorre a relação do usuário com artefatos interativos que permitem a realização de transições. Estudos dessa natureza podem contribuir para que as empresas desenvolvedoras de sistemas operacionais incorporem recursos de DUI de forma nativa em seus componentes.

Outro desafio está em verificar outras propriedades relacionadas a transição da interação. Em um estudo de Miyamaru (2010) houve a identificação da transição entre mídias. Nesta pesquisa houve a identificação das transições da distribuição da interação. Esta pesquisa avançou quanto ao aspecto da transição entre usuários, mas este último tipo ainda carece de pesquisas.

Esta reflexão possibilitou a construção do Quadro 10 que apresenta 14 obstáculos e oportunidades de pesquisa na área.

Quadro 10 – 14 obstáculos e oportunidades de pesquisa

	Obstáculo	Oportunidade
1	Levantamento de requisitos	Concepção de novas metodologias.
2	Especificação de requisitos	Como expressar, modelar e descrever requisitos dessa natureza.
3	Pouco suporte tecnológico	Experimentos e comparações das soluções existentes. Ampliação do suporte tecnológico.
4	Padrões de projeto	Padronização de soluções a problemas recorrentes.
5	Testes	Planejamento. Execução. Automação.

6	Contexto	Estabelecimento de um contexto comum entre múltiplos dispositivos e múltiplas aplicações.
7	Distribuição de conteúdo	Conversão de conteúdo. Adaptação de conteúdo. Adaptação de resolução.
8	Transação	Retorno ao estado anterior para desfazer operações em interfaces distribuídas compartilhadas
9	Decomposição e recomposição	Modelo de representação de objetos de interface gráfica. Estratégias de segmentação de interface e junção dos segmentos.
10	Composição de serviços	Associação semântica entre serviços.
11	Modelo computacional	Computação em nuvem; Abordagens híbridas.
12	Implantação e manutenção	Protocolos de <i>deployment</i> distribuído <i>multidevice</i> .
13	Componentes GUI para DUI	Design de interação.
14	Transição	Transição entre usuários.

Fonte: Próprio autor

Tais obstáculos e desafios são agendas para pesquisas futuras nessa área.

8 CONCLUSÕES

Neste capítulo, retomam-se os objetivos da pesquisa e evidenciam-se as contribuições. Em seguida, são identificados os trabalhos futuros que poderão dar continuidade à pesquisa. Finalmente, apresenta-se a conclusão do trabalho.

8.1 OBJETIVOS E CONTRIBUIÇÕES

Para que fosse possível concretizar o objetivo principal desta pesquisa, fez-se necessário atender uma coleção de objetivos secundários. Retomando-se esses objetivos apresentados na seção 1.2, apresentam-se as contribuições da pesquisa.

O primeiro objetivo estava relacionado a elucidação dos conceitos envolvidos na temática da pesquisa. Este objetivo foi atingido pela realização das pesquisas sobre a literatura, de forma exploratória e sistemática. Por meio desta pesquisa, foi possível compreender DUI como originada em diversas disciplinas e se consolidando em IHC. Foi possível identificar também que existem muitas pesquisas, em diversas áreas, na temática de interesse, sendo duas áreas as que mais nortearam esta pesquisa, a área de mídias cruzadas (*crossmedia*) e a de interfaces distribuídas (DUI). Esse objetivo foi alcançado e pode ser constatado por meio do capítulo 2, que se pretende que sirva como um tutorial para pesquisadores interessados em DUI poderem começar seus trabalhos.

O segundo objetivo estava na constatação e comparação das soluções tecnológicas existentes. Foi possível verificar que não existe uma solução completa observando os conceitos de DUI, que servisse de referência para o desenvolvimento de aplicações. Tampouco foi localizado um *framework* conceitual ou até mesmo diretrizes (*guidelines*) que nortearassem o desenvolvimento de soluções DUI. Foi possível verificar também que há uma grande heterogeneidade em relação às linguagem de programação quanto no nível de API, *frameworks* e padrões de comunicação. No entanto, foi possível verificar uma tendência de uso de *web service*. Notou-se também que a arquitetura de rede P2P é pouco explorada em DUI, contrário a outras áreas como a de mídias cruzadas e computação pervasiva. Os resultados que levaram a alcançar este objetivo podem ser averiguados no

capítulo 3. Como contribuição, esta pesquisa identificou uma lista de requisitos para uma arquitetura de referência em DUI.

O terceiro objetivo estava na documentação das transições identificadas e respectivas características. Foi possível verificar a existência de três tipos de transição (transição entre mídias, transição entre usuários e transição da distribuição da interação). Como contribuição, esta pesquisa definiu essas modalidades de transição e sistematizou um quadro com 10 cenários de transição de distribuição da interação que permitiu abstrair um modelo de implementação para o serviço de transição e componentes GUI. O capítulo 4 apresenta esses resultados.

O quarto objetivo estava na definição dos serviços necessários para a realização de transições de interação que permita a construção de diversos cenários de uso. Como contribuição foram especificados serviços que são apresentados nos capítulos 4 e 6 sob uma arquitetura orientada a serviços P2P dirigida a eventos baseada em DPWS.

O quinto objetivo exigia a verificação das transições de interação por meio de implementação computacional e realização de provas de conceito. Como contribuição têm-se a implementação dos serviços relacionados a transição da interação para as plataformas Android e PC (Windows e MacOS) e 6 componentes GUI que permitiram verificar a viabilidade da arquitetura para a realização das transições de distribuição da interação. O capítulo 5 apresenta a implementação da arquitetura e provas de conceito com os componentes.

O último objetivo era a reflexão sobre uma agenda de pesquisa. Como contribuição apresenta-se um quadro contendo 14 obstáculos e oportunidades de pesquisa em DUI. Essa reflexão contribui para o andamento desta pesquisa nos próximos anos e pode ser averiguada no capítulo 7.

8.2 TRABALHOS FUTUROS

Com os resultados obtidos na realização deste trabalho, assim como o conhecimento adquirido no desenvolvimento do estudo, pode-se sugerir os seguintes trabalhos a serem desenvolvidos:

- especificar serviços adicionais da arquitetura DISS, com vistas a atender todos os requisitos elicitados;

- realizar provas de conceito para os serviços do capítulo 6, com vistas a refatorar o modelo e contribuir com sua evolução;
- implementar um conjunto maior de componentes GUI para DUI. Há uma dificuldade em se visualizar aplicações DUI, a disponibilização de componentes que permitam rapidamente construir cenários de interação distribuída devem facilitar a compreensão e visualização de novos formatos de interação;
- documentar todos os componentes implementados e construir um manual para o desenvolvedor;
- realizar oficinas para verificação e avaliação dos componentes e dos serviços, e para atividades criativas na definição de cenários DUI;
- realizar estudos de usabilidade para auxiliar no *design* de interação dos componentes GUI para DUI. Um *design* centrado no usuário se faz necessário para produzir componentes que proporcionem a maior interatividade possível. Para essa atividade, visualiza-se a necessidade de construir variadas versões interativas de um mesmo componente e avaliá-los junto ao usuário.

Além dessas atividades de necessidade mais premente, vislumbram-se como trabalhos futuros os 14 desafios apontados no capítulo 7. Todos eles apresentam contribuições para com a evolução da arquitetura DISS.

8.3 CONCLUSÃO

Considera-se que esta pesquisa atendeu a seu propósito principal, de sistematizar as transições e definir uma arquitetura orientada a serviço que permite realizar transições da interação do usuário em um ambiente composto por múltiplos dispositivos de forma integrada.

A arquitetura na forma concebida realiza seu propósito. Apesar das provas de conceito terem sido realizadas para transições de distribuição de interação, foi possível constatar que é possível utilizar uma arquitetura orientada a serviços em uma estrutura de P2P para DUI.

DUI representa uma nova forma de usar os dispositivos e isso se reflete nos componentes de interface gráfica. Pensar em novos cenários de uso, em novos modelos de interação não é uma atividade fácil, o conceito *personal computer* ainda

é muito forte. Acredita-se que a arquitetura apresentada e os respectivos componentes permitem o uso integrado e sinérgico de múltiplos dispositivos, além da realização de projetos disruptivos (*disruptive design*), possibilitando desenvolvedores e *designers* de interação projetar inovações que criam novos atributos de valor para a interação do usuário.

REFERÊNCIAS

ANDREAS MULLER et al. An assisted device registration and service access system for future home networks. p. 1–5, 15 dez. 2009.

ARDISSONO, L. et al. **From Service Clouds to User-Centric Personal Clouds**. Proceedings of the 2009 IEEE International Conference on Cloud Computing. IEEE Computer Society, 2009. Disponível em: <<http://portal.acm.org/citation.cfm?id=1632959>>.

AVOUAC, P.-A.; LALANDA, P.; NIGAY, L. **Autonomic management of multimodal interaction: DynaMo in action**. Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems. EICS '12. New York, NY, USA: ACM, 2012. Disponível em: <<http://doi.acm.org/10.1145/2305484.2305493>>.

BAILLIE, L. et al. **Designing mona: User interactions with multimodal mobile applications**. HCI International. 2005. Disponível em: <http://mona.ftw.at/papers/designingmona_final.pdf>.

BALLAGAS, R. et al. **iStuff: a physical user interface toolkit for ubiquitous computing environments**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '03. New York, NY, USA: ACM, 2003. Disponível em: <<http://doi.acm.org/10.1145/642611.642705>>.

BANDELLONI, R.; MORI, G.; PATERNÒ, F. **Dynamic generation of web migratory interfaces**. Proceedings of the 7th international conference on Human computer interaction with mobile devices & services. MobileHCI '05. New York, NY, USA: ACM, 2005.

BANG, M. et al. Distributed user interfaces for clinical ubiquitous computing applications. **International Journal of Medical Informatics**, v. 74, n. 7-8, p. 545–551, Agosto 2005.

BARDAM, J. et al. **ReticularSpaces: activity-based computing support for physically distributed and collaborative smart spaces**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '12. New York, NY, USA: ACM, 2012. Disponível em: <<http://doi.acm.org/10.1145/2207676.2208689>>.

BARROS, G.; BENINI, J.; ZUFFO, M. K. **Home Entertainment UI Continuum: Cell Phone, HTPC and iTV**. [s.l.: s.n.].

BEAT SIGNER; MOIRA C. NORRIE. **A Framework for Cross-media Information Management**. In: CONFERENCE ON INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS. Grindelwald, Switzerland: 2005. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2435&rep=rep1&type=pdf>>.

BELOTTI, R. et al. **Experimental platform for mobile information systems**. Proceedings of the 11th annual international conference on Mobile computing and networking. Cologne, Germany: ACM, 2005. Disponível em: <<http://portal.acm.org/citation.cfm?id=1080829.1080856>>.

BERGLUND, E.; BVAANG, M. **Requirements for distributed user interface in ubiquitous computing networks**. Proc. of Conf. on Mobile and Ubiquitous MultiMedia MUM. .2002. Disponível em: <<http://www.mum2002 oulu.fi/technical/berglund.pdf>>.

BIER, E. A.; FREEMAN, S.; PIER, K. **MMM: The multi-device multi-user multi-editor**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '92. New York, NY, USA: ACM, 1992. Disponível em: <<http://doi.acm.org/10.1145/142750.143065>>.

BLUMENDORF, M.; ROSCHER, D.; ALBAYRAK, S. **Distributed User Interfaces for Smart Environments: Characteristics and Challenges**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

BOHN, H.; BOBEK, A.; GOLATOWSKI, F. **SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains**. Mobile Communications and Learning Technologies, Conference on Networking, Conference on Systems, International Conference on. Los Alamitos, CA, USA: IEEE Computer Society, 2006.

BOUMANS, J. Cross-Media on the Advance. In: **E-Content**. [s.l: s.n.]. p. 127–141.

BRANTON, C. et al. **Toward rapid and iterative development of tangible, collaborative, distributed user interfaces**. Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. EICS '13. New York, NY, USA: ACM, 2013. Disponível em: <<http://doi.acm.org/10.1145/2480296.2480312>>.

BRINGUÉ, X.; SÁDABA, C. La generación interactiva en Iberoamérica. Niños y adolescentes ante las pantallas. **Barcelona, España: Fundación Telefonica & Ariel**. Retrieved on April 29th, 2009.

BRODIE, J. **Designing to support communication on the move**. CHI '03 extended abstracts on Human factors in computing systems. Ft. Lauderdale, Florida, USA: ACM, 2003. Disponível em: <<http://portal.acm.org/citation.cfm?id=765891.766064>>.

BRONSTED, J.; HANSEN, K. M.; INGSTRUP, M. Service Composition Issues in Pervasive Computing. **IEEE Pervasive Computing**, v. 9, n. 1, p. 62–70, 2010.

BURNIK, U. et al. **Content4All: A Cross-media Platform for Community Information Exchange**. Computer as a Tool, 2005. EUROCON 2005. The International Conference on. In: COMPUTER AS A TOOL, 2005. EUROCON 2005. 2005. Disponível em: <10.1109/EURCON.2005.1629891>.

CARDOSO, J.; JOSÉ, R. **PuReWidgets: a programming toolkit for interactive public display applications**. Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems. EICS '12. New York, NY, USA: ACM, 2012. Disponível em: <<http://doi.acm.org/10.1145/2305484.2305496>>.

CASAROLI, L.; PERUZZOLO, A. A força da comunicação na sociedade midiática. **Comunicologia**, v. 1, p. 64–78, 2008.

CHEN, G.; KOTZ, D. **Solar: An Open Platform for Context-Aware Mobile Applications**. [s.l.: s.n.].

CHU, H. et al. Roam, a seamless application framework. **Journal of Systems and Software**, v. 69, n. 3, p. 209–226, 15 jan. 2004.

COMITE EXECUTIVO DE GOVERNO ELETRONICO. **e-PING Padroes de Interoperabilidade de Governo Eletronico**. Governo Brasileiro, 16 dez. 2008. Disponível em: <<http://www.governoeletronico.gov.br/acoes-e-projetos/e-ping-padroes-de-interoperabilidade>>

DAVIDSON, D. **Cross-Media Communications: An Introduction to the Art of Creating Integrated Media Experiences**. [s.l.] Unknown, 2010.

DE DEUGD, S. et al. SODA: Service Oriented Device Architecture. **Pervasive Computing, IEEE**, v. 5, n. 3, p. 94 –96, set. 2006.

DE HAAS, M. Crossmedia communication in the dynamic knowledge society. **IST Networking Session**, v. 28, p. 04–01, 2005.

DE OLIVEIRA NETO, J. S.; ROUSSEL, N.; FILGUEIRAS, L. V. L. **User's issues in crossmedia applications**. Proceedings of the 27th ACM international conference on Design of communication. 2009.

DEMEURE, A. et al. **The 4C Reference Model for Distributed User Interfaces**. IEEE, mar. 2008. Disponível em: <<http://dl.acm.org/citation.cfm?id=1395980.1396101>>.

DENA, C. Current State of Cross Media Storytelling: Preliminary observations for future design. **European Information Society Technologies (IST) Event The Netherlands, November**. http://www.starofdena.com/Docs/DENA_CrossMediaObservations.pdf, v. 15, p. 2006, 2004.

DEY, A. K. Understanding and Using Context. **Personal Ubiquitous Comput.**, v. 5, n. 1, p. 4–7, jan. 2001.

DLNA. **DLNA**. Disponível em: <<http://www.dlna.org/>>.

DPWS. **Devices Profile for Web Services Version 1.1**. Disponível em: <<http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html>>.

DUI. **DUI 2011 Distributed User Interfaces**. In: 1 WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

DUI. **DUI 2012 Distributed User Interfaces**. In: 2 WORKSHOP ON DISTRIBUTED USER INTERFACES: COLLABORATION AND USABILITY. Austin, TX, USA: University of Castilla-La Mancha, Spain, 5 maio 2012.

DUI. **DUI 2013 Distributed User Interfaces**. In: 3 WORKSHOP ON DISTRIBUTED USER INTERFACES: MODELS, METHODS AND TOOLS. London, UK: Springer, 24 jun. 2013.

ELMQVIST, N. **Distributed User Interfaces: State of the Art**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011a.

ELMQVIST, N. **Munin: A Peer-to-Peer Middleware for Ubiquitous Visualization Spaces**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011b.

ERL, T. **SOA Princípios de design de serviços**. 1. ed. São Paulo: Pearson Prentice Hall, 2009.

ESTES, J. **Seamlessness in the Cross-Channel User Experience**. Disponível em: <<http://www.nngroup.com/articles/seamless-cross-channel/>>.

FARDOUN, H. M.; LÓPEZ, S. R.; VILLANUEVA, P. G. **Improving E-Learning Using Distributed User Interfaces**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

FILGUEIRAS, L. V. L. et al. **X-Gov Planning: How to Apply Cross Media to Government Services**. Digital Society, 2008. Second International Conference on the DIGITAL SOCIETY, 2008.

FIORAVANTI, F.; SPINU, M.; CAMPANAI, M. **AXMEDIS as the Service Oriented Architecture for the Media: Is It Feasible?** Automated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS '07. Third International Conference on. 2007.

FRÖBERG, A.; ERIKSSON, H.; BERGLUND, E. **A Model for DUI Development**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

GABRIEL, M. C. C. **Arte Transmídia na Era Digital**. Escola de Comunicação e Artes: Universidade de São Paulo, 2012.

GHIANI, G.; PATERNÒ, F.; SANTORO, C. **On-demand cross-device interface components migration**. Proceedings of the 12th international conference on Human computer interaction with mobile devices and services. MobileHCI '10. New York, NY, USA: ACM, 2010. Disponível em: <<http://doi.acm.org/10.1145/1851600.1851653>>.

GJERLUFSEN, T. et al. **Shared substance: developing flexible multi-surface applications**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '11. New York, NY, USA: ACM, 2011. Disponível em: <<http://doi.acm.org/10.1145/1978942.1979446>>.

HAN, S. W. et al. **A new middleware architecture for ubiquitous computing environment**. Software Technologies for Future Embedded and Ubiquitous Systems, 2004. Proceedings. Second IEEE Workshop on. Maio 2004.

HENDRIK, VON. **Custom UI Controls with JavaFX - Part 1 | GuiGarage**. Disponível em: <<http://www.guigarage.com/2012/11/custom-ui-controls-with-javafx-part-1/>>. Acesso em: 27 jan. 2014.

HONG, J. I. **The Context Fabric: An Infrastructure for Context-aware Computing**. CHI '02 Extended Abstracts on Human Factors in Computing Systems. CHI EA '02. New York, NY, USA: ACM, 2002. Disponível em: <<http://doi.acm.org/10.1145/506443.506478>>.

JAMMES, F.; MENSCH, A.; SMIT, H. Service-oriented device communications using the devices profile for web services. **Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing**, MPAC '05. p. 1–8, 2005.

JENKINS, H. **Convergence culture: Where old and new media collide**. [s.l.] NYU Press, 2006.

JENSEN, J. F. **Interactive television: new genres, new format, new content**. Proceedings of the second Australasian conference on Interactive entertainment. Sydney, Australia: Creativity & Cognition Studios Press, 2005. Disponível em: <<http://portal.acm.org/citation.cfm?id=1109194>>.

JMEDS. **JMEDS Framework Overview**. Disponível em: <<http://ws4d.e-technik.uni-rostock.de/wp-content/uploads/2011/05/StackOverview.pdf>>. Acesso em: 27 jan. 2014.

JOHANSON, B.; FOX, A.; WINOGRAD, T. The Interactive Workspaces project: experiences with ubiquitous computing rooms. **IEEE Pervasive Computing**, v. 1, n. 2, p. 67–74, 2002.

JOHN B. HARRIGAN. Use of Cloud Computing Applications and Services. **The Pew Internet & American Life Project, April-May**, p. 9, maio 2008.

JUNG-TAE KIM et al. Implementation of the DLNA Proxy System for Sharing Home Media Contents. **Consumer Electronics, IEEE Transactions on**, v. 53, n. 1, p. 139–144, 2007.

KANG, K. C. et al. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. Carnegie Mellon University: Software Engineering Institute (SEI), nov. 1990.

KARNOUSKOS, S. et al. **Towards the real-time enterprise: service-based integration of heterogeneous SOA-ready industrial devices with enterprise applications**. Proc. of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'09). 2009.

KIANI, S. L. et al. **A distributed middleware solution for context awareness in ubiquitous systems.** Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on. ago. 2005.

KLOKMOSE, C. N.; BEAUDOUIN-LAFON, M. **VIGO: instrumental interaction in multi-surface environments.** Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '09. New York, NY, USA: ACM, 2009. Disponível em: <<http://doi.acm.org/10.1145/1518701.1518833>>.

KLOMPMAKER, F.; SCHRAGE, K.; REIMANN, C. **INDiE: a framework for human computer Interaction in Distributed Environments.** Proceedings of the 6th International Conference on Mobile Technology, Application Systems. Mobility '09. New York, NY, USA: ACM, 2009. Disponível em: <<http://doi.acm.org/10.1145/1710035.1710061>>.

KOVACHEV, D. et al. **DireWolf - distributing and migrating user interfaces for widget-based web applications.** Proceedings of the 13th international conference on Web Engineering. ICWE'13. Berlin, Heidelberg: Springer-Verlag, 2013. Disponível em: <http://dx.doi.org/10.1007/978-3-642-39200-9_10>.

KOZMA, R. B. Learning with Media. **Review of Educational Research**, v. 61, n. 2, p. 179 –211, Summer 1991.

KUMAR, M. et al. PICO: a middleware framework for pervasive computing. **Pervasive Computing, IEEE**, v. 2, n. 3, p. 72 – 79, set. 2003.

LAWSON, J.-Y. L. et al. **An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components.** Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems. EICS '09. New York, NY, USA: ACM, 2009. Disponível em: <<http://doi.acm.org/10.1145/1570433.1570480>>.

LO, K. W. K. et al. **i*Chameleon: A unified web service framework for integrating multimodal interaction devices.** 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). 2012

LUYTEN, K.; CONINX, K. **Distributed user interface elements to support smart interaction spaces.** Multimedia, Seventh IEEE International Symposium on. 2005.

MANCA, M.; PATERNÒ, F. **Distributing User Interfaces with MARIA.** DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

MARC MCLOUGHLIN. **ECSCW Workshop: Bridging Interaction Clouds: Exploring collaborative interaction across assemblies of mobile and embedded technology.** ECSCW Workshop: bridging interaction clouds, 14 jul. 2009. Disponível em: <<http://www.bridgingclouds.blogspot.com/>>.

MELCHIOR, J. et al. **A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications.** Proceedings of the 1st ACM SIGCHI

symposium on Engineering interactive computing systems. EICS '09. New York, NY, USA: ACM, 2009.

MELCHIOR, J.; VANDERDONCKT, J.; VAN ROY, P. **Distribution Primitives for Distributed User Interfaces**. DUI 2011. 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

MIYAMARU, F. et al. **Mídia cruzada em serviços de governo: conceito e aplicação**. Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems. Porto Alegre, RS, Brazil: Sociedade Brasileira de Computação, 2008. Disponível em: <<http://portal.acm.org/citation.cfm?id=1497525>>.

MIYAMARU, F. **Transições entre mídias: arcabouço tecnológico para aplicações de mídia cruzada**. Escola Politécnica: Universidade de São Paulo, 2010.

MIYAMARU, F.; FILGUEIRAS, L. **X-Gov: Crossmedia for Government Services**. Proceedings of Workshop on Design & Evaluation of e-Government Applications and Services (DEGAS'2009). Uppsala, Sweden: 24 ago. 2009.

NACENTA, M. A. **The Broken Mirror: Space, Attention and Collaboration in Co-located Distributed User Interfaces**. Proceedings of the 3rd Workshop on Distributed User Interfaces: Models, Methods and Tools. In: ENGINEERING INTERACTIVE COMPUTING SYSTEMS (EICS). London, UK, UK: ACM, 24 jun. 2013

NETO, J. S. DE O.; FILGUEIRAS, L. V. L. **Crossmedia application design: exploring linear and non-linear narrative abilities**. Proceedings of the 26th annual ACM international conference on Design of communication. Lisboa, Portugal: ACM, 2008. Disponível em: <<http://portal.acm.org/citation.cfm?id=1456536.1456582>>.

NG, K. et al. **AXMEDIS Programme and Publication Tools Integration with Workflow-Enabled Communication and Process Control**. Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. AXMEDIS '06. Second International Conference on. 2006.

NICKELSEN, A. et al. **OPEN: open pervasive environments for migratory interactive services**. Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services. iiWAS '10. New York, NY, USA: ACM, 2010. Disponível em: <<http://doi.acm.org/10.1145/1967486.1967585>>.

NOLL, J.; ALAM, S.; CHOWDHURY, M. **Integrating Mobile Devices into Semantic Services Environments**. Wireless and Mobile Communications, 2008. ICWMC '08. The Fourth International Conference on. In: WIRELESS AND MOBILE COMMUNICATIONS, 2008.

OASIS WEB SERVICES DISCOVERY AND WEB SERVICES DEVICES PROFILE (WS-DD) TC. **Devices Profile for Web Services Version 1.1**. Disponível em:

<<http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html>>. Acesso em: 5 dez. 2011.

OLSEN, J. et al. **Cross-modal interaction using XWeb**. Proceedings of the 13th annual ACM symposium on User interface software and technology. UIST '00. New York, NY, USA: ACM, 2000.

OSGi Alliance | Main / OSGi Alliance. Disponível em: <<http://www.osgi.org/Main/HomePage>>. Acesso em: 28 fev. 2012.

PATERNO', F.; SANTORO, C.; SPANO, L. D. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. **ACM Transactions on Computer-Human Interaction (TOCHI)**, v. 16, p. 19:1–19:30, nov. 2009.

PONNEKANTI, S; ABOWD, G.; BRUMITT, B.; SHAFER, S. ICrafter: A Service Framework for Ubiquitous Computing Environments. In: **Ubicomp 2001: Ubiquitous Computing**. Lecture Notes in Computer Science. [s.l.] Springer Berlin / Heidelberg, 2001. v. 2201p. 56–75.

PRETI, J. P. D.; FILGUEIRAS, L. V. L. **Engineering Transitions to Bind Distributed Interaction**. Proceedings of the 3rd Workshop on Distributed User Interfaces: Models, Methods and Tools. In: **ENGINEERING INTERACTIVE COMPUTING SYSTEMS (EICS)**. London, UK: ACM, 24 jun. 2013.

PRETI, J. P. D.; MIYAMARU, F.; FILGUEIRAS, L. **Transitions: a crossmedia interaction relevant aspect**. In: **HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES**. Hawaii, USA: IEEE Computer Society, 6 jan. 2014.

RANGANATHAN, A.; CAMPBELL, R. H. **A middleware for context-aware agents in ubiquitous computing environments**. Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware. Middleware '03. New York, NY, USA: Springer-Verlag New York, Inc., 2003. Disponível em: <<http://dl.acm.org/citation.cfm?id=1515915.1515926>>.

ROBINSON, J.; WAKEMAN, I.; OWEN, T. **Scooby: middleware for service composition in pervasive computing**. Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing. MPAC '04. New York, NY, USA: ACM, 2004. Disponível em: <<http://doi.acm.org/10.1145/1028509.1028520>>.

ROMAN, M. et al. A middleware infrastructure for active spaces. **IEEE Pervasive Computing**, v. 1, n. 4, p. 74–83, 2002.

ROMAN, M.; CAMPBELL, R. H. A User-Centric, Resource-Aware, Context-Sensitive, Multi-Device Application Framework for Ubiquitous Computing Environments. **Urbana**, v. 51, p. 61801, 2002.

SACRAMENTO, V. et al. MoCA: A Middleware for Developing Collaborative Applications for Mobile Users. **Distributed Systems Online, IEEE**, v. 5, n. 10, p. 2, out. 2004.

SAHA, D.; MUKHERJEE, A. Pervasive computing: a paradigm for the 21st century. **Computer**, v. 36, n. 3, p. 25–31, 2003.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. **Personal Communications, IEEE**, v. 8, n. 4, p. 10 –17, Agosto 2001.

SENDÍN, M.; LÓPEZ, J. M. **Approach for Enriching Distributed User Interfaces with Awareness**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

SIGNER, B.; NORRIE, M. An Architecture for Open Cross-Media Annotation Services. **Web Information Systems Engineering-WISE 2009**, p. 387–400, 2009.

SONY, P. **PS4: The Ultimate FAQ – North America** PlayStation.Blog, 30 out. 2013. Disponível em: <<http://blog.us.playstation.com/2013/10/30/ps4-the-ultimate-faq-north-america/>>. Acesso em: 29 dez. 2013.

SOUSA, J.; GARLAN, D. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. **Computer Science Department**, 1 ago. 2002.

STEINERT, B. et al. **Multi-user multi-account interaction in groupware supporting single-display collaboration**. 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009.

STREITZ, N. A. et al. **i-LAND: an interactive landscape for creativity and innovation**. Proceedings of the SIGCHI conference on Human Factors in Computing Systems. CHI '99. New York, NY, USA: ACM, 1999. Disponível em: <<http://doi.acm.org/10.1145/302979.303010>>.

VANDERHULST, G.; TRAPPENIERS, L. **Public WiFi hotspots at your service**. 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). 2012.

VILLANUEVA, F. J. et al. Towards a Unified Middleware for Ubiquitous and Pervasive Computing. **International Journal of Ambient Computing and Intelligence**, v. 1, n. 1, p. 53–63, 2009.

VILLANUEVA, P. G.; TESORIERO, R.; GALLUD, J. A. **Is the Quality in Use Model valid for Distributed User Interfaces?** Proceedings of the 2nd Workshop on Distributed User Interfaces: Collaboration and Usability. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS. Austin, TX, USA: ACM, 5 maio 2012.

VILLANUEVA, P. G.; TESORIERO, R.; GALLUD, J. A. **Proxywork: Distributing User Interface Components of Web Applications**. Proceedings of the 3rd Workshop on Distributed User Interfaces: Models, Methods and Tools. In: ENGINEERING INTERACTIVE COMPUTING SYSTEMS (EICS). London, UK: ACM, 24 jun. 2013.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. Rio de Janeiro: Elsevier, 2009.

WIBERG, C.; JEGERS, K.; BODÉN, J. **Cross Media Interaction Design**. Workshop “HCI and New Media Arts: Methodology and Evaluation” at CHI. 2007.

WS-ADDRESSING. **Web Services Addressing (WS-Addressing)**. Disponível em: <<http://www.w3.org/Submission/ws-addressing/>>. Acesso em: 27 jan. 2014.

WS-COORDINATION. **WS-Coordination v1.2 OASIS Standard**. Disponível em: <<http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os/wstx-wscoor-1.2-spec-os.html>>. Acesso em: 5 dez. 2011.

WS-DISCOVERY. **OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1**. Disponível em: <<http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html>>. Acesso em: 27 jan. 2014.

WS-EVENTING. **Web Services Eventing (WS-Eventing)**. Disponível em: <<http://www.w3.org/Submission/WS-Eventing/>>. Acesso em: 27 jan. 2014.

WS-METADATAEXCHANGE. **Web Services Metadata Exchange (WS-MetadataExchange)**. Disponível em: <<http://www.w3.org/TR/ws-metadata-exchange/>>. Acesso em: 27 jan. 2014.

WSDL. **Web Service Definition Language (WSDL)**. Disponível em: <<http://www.w3.org/TR/wsdl>>. Acesso em: 27 jan. 2014.

YEON-JOO OH et al. Design of an Extended Architecture for Sharing DLNA Compliant Home Media from Outside the Home. **Consumer Electronics, IEEE Transactions on**, v. 53, n. 2, p. 542–547, 2007.

ZDUN, U.; HENTRICH, C.; DUSTDAR, S. Modeling Process-driven and Service-oriented Architectures Using Patterns and Pattern Primitives. **ACM Trans. Web**, v. 1, n. 3, Setembro 2007.

ZEEB, E. et al. **Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services**. 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 2007.

ZHANG, X.; DETERS, R. **MVC apps in the personal cloud**. 2012 IEEE Latin America Conference on Cloud Computing and Communications (LATINCLOUD). 2012.

ZÖLLNER, M.; JETTER, H.-C.; REITERER, H. **ZOIL: A Design Paradigm and Software Framework for Post-WIMP Distributed User Interfaces**. DUI 2011. In: 1ST WORKSHOP ON DISTRIBUTED USER INTERFACES. Vancouver, BC: University of Castilla-La Mancha, Spain, 7 maio 2011.

APÊNDICE A – PROTOCOLO DE REVISÃO SISTEMÁTICA

A revisão sistemática teve como objetivo identificar arquiteturas de referência para construção de aplicações distribuídas com o objetivo de subsidiar a definição uma arquitetura de componentes e serviços capazes de generalizar as principais funções de um grupo de aplicações que envolvam interação em múltiplos dispositivos. Essa revisão teve em mente as seguintes questões:

- Quais são as arquiteturas de software existentes para o desenvolvimento aplicações que necessitam de interação em múltiplos dispositivos?
- Quais são as bases tecnológicas utilizadas na arquitetura?
- É middleware? Framework?
- Como as aplicações são construídas/definidas a partir dessas arquiteturas?
- Como as arquiteturas são usadas para a construção de aplicações?
- É possível virtualizar as funções genéricas de uma interação distribuída em um ambiente SOA?
- É possível especificar/projetar funções genéricas de uma interação distribuída por meio do paradigma de serviços?
- É possível construir interações distribuídas a partir de múltiplas plataformas por meio do reuso de serviços?
- Qual o tipo de interação distribuída suportada? Cross- o que?
- Em que camada se dá a distribuição?
- Qual foi o experimento realizado para validar a arquitetura?

Os recursos e estratégias para busca e seleção de estudos foram definidos e selecionados com base em quatro itens fundamentais, são eles:

- **Fontes de busca:** base de dados eletrônicas indexadas (IEEE, ACM, Springer), busca em máquinas de busca eletrônica (ISI Web of Knowledge, Scopus, Compendex, Google Scholar), anais de eventos relacionados.
- **Idioma:** Preferencialmente a língua inglesa, por ser considerado o idioma mais aceito internacionalmente para artigos científicos na área do trabalho, contudo artigos relevantes encontrados na língua portuguesa, publicados em eventos nacionais relacionados à área de pesquisa, também serão considerados.
- **Palavras-chave:**

- “software reuse”; “component reuse”; “middleware architecture”; “message-oriented middleware”; “service-oriented middleware”; “cloud based architecture”; “software architecture”; “cloud computing”; “cloud-based computing architecture”; “virtualized cloud resources”;
- “multi-modal interfaces”; “mobile computing”; “mobile devices”.
- "distributed interaction"; "distributed user interface"; "cross-device"; "cross device"; crossdevice; "cross-media"; crossmedia; "cross media"; transmedia; "multi-device". multidevice ; "multi device".
- interaction; "user interface".
- architecture; service; SOA.

Em linhas gerais, a escolha dos termos e suas relações se justificam pelo objetivo principal da revisão, que é identificar plataformas que dão suporte ao desenvolvimento de aplicações que envolvam a construção de aplicações que promovam a interação distribuída por meio da reutilização de componentes e serviços de software.

- **Data de publicação:** Preferencialmente artigos e livros publicados nos últimos 5 anos (a partir do ano de 2009), para que se possa buscar novas abordagens sobre o assunto em questão, mas eventualmente podem ser encontradas fontes mais antigas que dão sustentação ao assunto.

Como critérios e procedimentos para seleção dos estudos tem-se:

- Critérios de inclusão:
 - Descrição arquitetural da aplicação/solução desenvolvida por meio de reuso de componentes e serviços de software;
 - Ter sido publicada nos últimos 5 anos;
 - Ser um artigo e não um volume de *proceeding*, descrição de workshop, entre outros que representem uma coleção de artigos.
- Critérios de exclusão:
 - Trabalhos que simplesmente descrevam aplicações de interação distribuída sem uma abordagem de reuso ou generalização de elementos de software.
 - Artigos que abordam o assunto de interação distribuída com foco em questões de performance, questões de segurança e balanceamento de carga.

- Artigos cujos arquivos não estejam disponíveis on-line.
- Artigos cujo foco não corresponda à questão de pesquisa.

Como processo de seleção preliminar foi necessário construir uma *string* de busca formada pela combinação das palavras-chave identificadas e submetê-las às máquinas de busca relacionadas.

A *string* de busca formada inicialmente era muito grande para qualquer um dos motores de busca mencionados, o que levou a uma otimização da mesma para a que segue:

```
((("cross-device" OR "cross device" OR crossdevice OR "cross-media" OR crossmedia OR "cross media" OR "multi-device" OR multidevice OR "multi device") AND ("user interface" OR GUI OR "Graphics User Interface") AND (architecture AND (service OR SOA OR component OR framework OR middleware))) AND ("bind component" OR "component bind" OR "service bind" OR "bind service") AND ("distributed interaction" OR "distributed user interface") AND ("reuse"))
```

Mesmo após otimização da *string* de busca, em muitos motores foi necessário dividir a mesma separando por palavras-chave, respeitando claro a lógica proposicional da sentença original. Alguns motores como ISI Web of Knowledge e Scopus não permitiam a busca por string de comando e outros como Compendex não permitiam a busca dos termos em qualquer parte do documento. Esses fatores impossibilitaram o uso na íntegra da *string* de busca para muitos motores.

Os trabalhos recuperados foram inicialmente selecionados pela leitura dos resumos. Constatando-se a relevância de um trabalho, o mesmo foi selecionado para ser lido na íntegra. Os trabalhos selecionados foram armazenados em software de organização de referência bibliográfica (Zotero).

Investigating Crossmedia Possibilities in the Cloud

João Paulo Delgado Preti

Instituto de Computação – UFMT
Av. Fernando Correa Costa, 2.367
Cuiabá – MT 78.060-900
joaopaulopreti@ic.ufmt.br

Lucia Vilela Leite Filgueiras

Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto t3 158
São Paulo SP 05508-900
lfilguei@usp.br

ABSTRACT

The growth of information and communication technology pushes for changes in how information is obtained or sent, and allows the emergence of new interaction models. This research is part of an ongoing study, which is investigating computational platforms that support simultaneous and synergistic use of different media and content viewing. These new interaction models may find in the cloud computing platform a good alternative to share resources, because of its flexibility, convergence and adoption of a service-oriented architecture. This article aims to show the challenges and opportunities of deliver crossmedia interaction on the cloud.

Keywords: Cloud computing, interaction clouds, crossmedia, distributed user interfaces

1. INTRODUCTION

Information and communication technologies are essential elements for achieving most of the tasks of modern life, people need to deal with different electronic devices to obtain information and access various services. The distribution of user interaction by different devices is today a natural phenomenon. A large quantitative study of the usage of information and communication technology by young users (from 10 to 18 y-o) organized by Sala e Chalezquer (2010) has

concluded that these users perform multitasking over devices, using television, games and internet navigation simultaneously [4].

However, the designed, purposeful distribution of interaction over multiple devices, in order to explore the synergy of this distribution, hasn't yet received much attention of the HCI community. A few papers exist on the concept of crossmedia interaction design (Filgueiras et al, Neto et al, Segestahl, Dena) and more recently, a CHI 2011 workshop on Distributed User Interfaces has been organized.

An immediate consequence of this distribution is the need of exchanging data between applications and navigation through different platforms. Considering that each media has its own language, and limited capacity, an application can migrate from one media to another, in order to obtain each best feature, characterizing what is called crossmedia applications.

Nowadays, information exchange among different devices is done "ad hoc", or by application-specific protocols, or by the capture of information by a device on another, for example, using two-dimensional recognizers codes as QRCode and MSTag, or by scanning the audio as the Shazam software.

The suitable architecture for a service with multiple channels is the integration and sharing of data exchanged between them. An approach that does not consider the coordinated exchange of data through media platforms, ie not allowing data to flow directly from a channel to another, creates a low flexibility architecture.

With cloud offering a global way to access and storage information and the overall growth of ontological models to automate the exchange of data between systems, it creates an infrastructure that encourages the growth of systems offered as a service and encourages the composition of systems based on integration between them.

This research is part of an ongoing study, which is investigating computational platforms that support simultaneous and synergistic use of different media and content viewing. In previous work of the research group¹⁸, we studied the importance of transitions through different media, within the concept of crossmedia. Crossmedia is the deliver of a story or theme through different media, each one contributing with their forces for the success of the story, the transitions are the resources that support the continuation of interaction in a platform change [15]. It is believed that the collaborative support of different media can contribute significantly to improve the quality of user experience.

An important aspect of our research is the fact that, currently, Web application protocols such as, for example, HTTP and WAP do not support transitions between devices, requiring them to be controlled centrally. In our research group we developed an integrator framework which enabled the interaction between different devices, supporting data exchange, coordination of interaction and simultaneity data presentation, however, all transitions must be passed by the service manager transitions.

Therefore, we are interested in studying the cloud from the standpoint of user access to it for storage, search, processing and interaction with other systems, and its ability to support this interaction through any computing device [10, 12].

Studies aimed at communicating with the cloud should be concerned with users in at least three roles: the managers of the cloud, the cloud service providers (customers of the first, dividing

even the user role of cloud and administrator of the service) and service users (clients of service providers).

Interested in offering services through multiple devices seamlessly, we found that share resources in the cloud can be a good alternative, since it allows great flexibility in the use of computational resources, can serve as a point of convergence between devices and adopts a service-oriented architecture.

To this end, we carried out initial research on the prospects of cloud computing, from the standpoint of interaction. This article summarizes the results of this survey, it is structured as follows. Section 2 presents important concepts related to crossmedia. Section 3 discusses aspects of the communication paradigm of cloud and human-cloud interaction as it is today. Section 4 presents the need to specify a model-based communication service for the purpose of this research, which is to deliver services across multiple media in a coordinated manner.

2. CROSSMEDIA

Crossmedia study how to use the potential of each media in a collaborative way and is expected to grow driven by the amount of devices and the increasingly available and facilitated communication between these various computing devices. This study aims to enable people to accomplish their tasks through the best suited media to the execution context.

It is used in this research the crossmedia definition of Filgueiras et al. [7], which is based on ideas of Boumans, Dena, De Haas and Wiberg:

"Crossmedia is the collaborative support of multiple media to delivering a single story or theme, in which the storyline directs the receiver from one medium to the next, according to each medium's strength to the dialogue."

In other words, is the collaborative support of different media for the execution of a use case, looking at using the media that provides the best potential for interaction, that will provide a better user experience, to perform a certain task.

An important point of this approach is the ability to favor different user types. For example, a

¹⁸ X-Gov project, financed by Microsoft Research and Fapesp between 2007 and 2010.

person with disabilities can choose the channel of communication best suited to his needs. The response of this interaction can also be delivered in many different formats.

A relevant aspect of crossmedia approach is that it divides content into elements of a story or theme and establish relations of:

- corroboration: the same content appears in different formats in different media;
- complementation: to understand the story or theme is necessary to get other parts of the content;
- concomitance: requires more than one media to present the content at the same time and synchronously.

The crossmedia concept is widespread in the publicity area. Computer science begins to conduct researchs on the topic. Sometimes this concept is mistaken with the concept of transmedia, a term coined by Henry Jenkins. These two concepts relate to integrated and interrelated experience in different media, but the main difference lies in interactivity. Crossmedia communication require public to be pro-activity, being more directly involved in the interaction [5].

2.1 Distributed user interfaces (DUI)

A research area that intersects with crossmedia is the research area of distributed interfaces. With pervasive computing and multiple interactive devices, rises the need for new interface models. For example, the interface model (GUI) of the late 70's was designed to run on a single computer. As result, developing for multiple devices is more error-prone and requires greater effort [3].

A distributed interface consists of a user interface (UI) that has the ability to distribute all or parts of its components across multiple monitors, devices, platforms, screens and / or users [14].

A list with usage scenarios for DUI [14]:

- multi-monitor usage: a user on a platform distributes its graphical user interface for multiple monitors connected to the same platform;

- multi-device usage: a user employs different devices simultaneously, using, or not, the same operating system;
- multi-platform usage: a user uses different computer platforms, probably with different operating systems. Note that multi-device usage implies in multi-platform usage, but the opposite is not true: a user can use multiple computers (multi-platform) that are similar (not being multi-device);
- multi-display usage: combination of multi-monitor and multi-device usage. A user can distribute the UI across multiple monitors and devices simultaneously;
- multi-use: represents an extension of the previous uses for multiple users concurrently. In this case, one or more users can distribute all or parts of your UI components for multiple monitors, devices, platforms or screens.

A DUI can be classified by 3 (three) dimensions [11]:

- location: determined by the user's physical location;
- activity: determined by activities that one or more users must run in a period of time to achieve a common goal;
- device: determined by the properties of available resources that are accessible to the user.

It is observed that crossmedia concept is broader and not restricted specifically to graphical interfaces, but any type of media. Media can be defined by its technology, its symbolic system and processing power [9] and therefore can include both digital and non-digital media (eg print media).

2.2 The challenges

A common problem faced in the context of cross-media and also for some uses of DUI is the need to establish and maintain a common and shared context, allowing user's activities to be maintained and / or reflected as the transition occurs from one media to another or when these

different media are used concomitantly. In our previous experience, we defined the X-Session concept, which maintain context information even when the interaction migrates from one device to another.

Another challenge is related to computational resources scalability, depending on the context, the number of users and devices can grow significantly.

The definition and adoption of open standards communication for these contexts are also challenging.

Observing these challenges, the need for information sharing, scalability and adoption of open standards, is that the research group began studying the cloud as a platform technology solution.

3. CLOUD COMPUTING

Vaquero et al. [19], looking for a cloud computing definition, conclude that the cloud is a great repository of virtualized resources easily accessible and usable, reconfigurable to meet a variable demand and optimizing resources, and operated by a pay-per-view model, in which guarantees are provided through customizable SLAs.

McFedries [12] points to the element that we consider most important for our cloud computing survey, which is the issue of device-independent access (free device), when it is presented that cloud computing is a computational model where not only data but also our software resides, and we access not only by PCs but also by other devices like smart phones, PDAs, mobile computing devices, game consoles and even cars. The author coins the term "cloud-friendly devices" to represent such devices.

As with any new technology, cloud computing brings new opportunities and new challenges. The UC Berkeley laboratory of adaptive and reliable distributed systems presents the 10 major obstacles and opportunities of cloud computing, availability of service, data lock-in, data confidentiality and audibility, data transfer bottlenecks, performance unpredictability, scalable storage, bugs in large distributed

systems, scaling quickly, reputation fate sharing and software licensing [17].

Interestingly, although all the items mentioned affects in many ways the user, the aspect of interaction is not mentioned explicitly. England [6] and McLoughlin [13] agree with the importance of the subject, arguing that interaction cloud presents interaction complex patterns, in which there are shares and exchanges that unfold around mobile and embedded technologies.

Our study indicates that the cloud interaction model is moving towards a crossmedia interaction model. One of the few studies available about user behaviour in the cloud is of the Pew Institute [8], which monitors the Internet's impact on American society. In 2010, a survey about 850 Internet experts, indicates an expectation that users will use the cloud more than the desktop in 2020. This same survey predicts that by 2020, people will no longer work with software running on general purpose PCs to work on the Internet from their smartphones [1].

3.1 Interaction on clouds

Aspects of communication in the cloud are more complex than in other distributed systems by the very nature of the proposal to provide a single point for getting computational resources. Current applications in the cloud are imbricated, resulting in a richer and more complex user interaction.

The convergence of mobile and stationary systems have allowed people to use mobile technology to interact with embedded systems, offering a great variety of possibilities for interaction and patterns that are defined by McLoughlin as "interaction clouds" [13]. This concept appears when the Internet becomes a resource free of location and devices constraint, allowing people to interact in multiple contexts (mobile / embedded).

In the paper [18] we have published the initial results of a research which investigates the role of the user in the services offered in the cloud, the problems currently faced and research proposals. Preliminary studies of this research pointed to four aspects of user-cloud interaction:

- the use of the cloud is growing;

- the user tends to use multiple media, mainly mobile;
- applications are complex and imbricated;
- the overlapping touches on aspects of architecture which are reflected in use.

Regarding the latter two aspects we observed that the integrations between the services surveyed, for example, Zoho and Google documents, have difficulty in establishing integration by restrictions of its public operations, but also a difficulty in sharing the user's context. These problems lead to a user cognitive erosion, since he realizes that their data and applications are distributed and not centralized as proposed by the cloud.

Interested in cloud computing as a technological platform to support crossmedia interactions, we present the next segment.

4. CROSSMEDIA AND THE CLOUD

In the 2008 survey of Pew Institute [8], one of the reasons why users were interested in is the fact that, with the cloud, their data can be accessed from any computer, even though that study did not explicitly write that computer can be a cellular or other platform than a desktop computer. McFedries [12] in his article "The cloud is the computer", argues that the diversity of applications and services that has been offered in the cloud is composing the new user's computer, where the device used is **transparent**.

Technologies used in cloud computing model, such as web services, are becoming available in a variety of devices with the opportunity to explore the potential of interconnected devices.

The cloud computing paradigm requires user interactions in ad-hoc format and self-organizational systems. The services must be alert to mobility, but also open to integration.

There are still no native support for transitions between media. It is understood transaction navigating between media, when the user is directed to achieve the required steps in pursuit of his goal. For example, a scenario where you want to continue moving in the interaction of the TV to the mobile phone could be more natural than using a QR Code, in addition to this type of

strategy may not be enough to convey the state of the user interaction had performed on TV.

Ardisson et al. [2], focusing on the problem of synchronization of applications in the cloud context, proposed a coordination middleware for dealing with the various service interfaces to enable synchronization, using strategies such as push. Similar strategy was adopted by our research group with a crossmedia framework [15].

In the context of cloud, observing the current protocols of web services (WS-*), there is no support for the implementation of a crossmedia context. The specification of a crossmedia protocol shall consider aspects of concurrency, corroboration and completion, in order to enable simultaneous use of devices, data conversion and coordinated access control to different devices, respectively.

The WS-* protocols are divided into the following categories: interoperability issues, business process specifications, metadata specifications, messaging specifications, reliability specifications, specifications management, security specifications, transaction specifications, resource specifications and presentation specifications [16].

The presentation specifications category is the one that has only one specification, the Web Services for Remote portlets (WSRP), that is currently in version 2.0, but this specification does not take into account aspects such as concurrency and crossmedia complementation.

We observed that the WS-Context specifications provides a definition, a structuring mechanism and service definitions for organizing and sharing context across multiple execution endpoints. The specification also provides service interfaces for managing session-oriented protocols and representing the corresponding activities with contexts. Context propagation is possible using different protocols than those used by the application. Nonetheless, the specification is concerned in share a Server context that is not enough to share the user interaction state, which requires de client context too.

We also observed the WS-Coordination specification, that describes an extensible framework for providing protocols that coordinate the actions of distributed applications. In general terms, coordination is the act of one entity (known as the coordinator) disseminating information to a number of participants for some domain specific reason. WS-Coordination by itself does not define all the features required for a complete solution. WS-Coordination is a building block that is used in conjunction with other specifications, like WS-Context, and application-specific protocols to accommodate a wide variety of protocols related to the operation of distributed Web services.

The WS-Coordination specification provides for extensibility and flexibility along two dimensions. The framework allows for:

- the publication of new coordination protocols;
- the selection of a protocol from a coordination type and the definition of extension elements that can be added to protocols and message flows.

This extensibility and flexibility ties with our interest in define a protocol that allow sharing the user interaction. Figure 1 shows two applications with their own coordinators, establishing the communication by protocol Y.

Figure 1. Two applications with their own coordinators [16].

Nowadays the group is specifying a minimum data set required for:

- identify the device type;
- identify the screen resolution;

- establish a crossmedia session : aiming to maintain state on the server side when the occurrence of transitions between media . When the user changes device a new session is established. Allow sharing data between different sessions is necessary to have continuity in the interaction when changing media. This concept, called X-Session by our research group, has been tested and we are investigating the possibility of using the WS-Context for this purpose;
- maintain the state of the interaction: with the intent to share interaction information with the user interface in another device when the media change, for example, sharing cookies, the state tree of objects that are present in the interface and callbacks. Only the sharing of information on the server side is not sufficient to allow fluidity in the interaction flow when changing media.

We are following OASIS specifications. This is important so you can reuse an entire infrastructure already implemented that is standard in the cloud.

CONCLUSIONS

Cloud computing facilitates access to computing resources and encourages integration and use of software services offered in this medium . A lot of effort is making to the success of this computational model, either from the industry or the scientific community.

The interaction through multiple media platforms will be needed to make the user experience more natural and intuitive in this model where the cloud is configured as the user's computer. Increasingly you do work, studying, having fun and conducts activities of everyday life through devices other than desktop, especially mobile.

The cloud shows itself an interesting platform as a point of convergence for devices and services, providing computational elasticity as demand, but interoperability between service layers have restrictions that prevent new types of interactions to occur. Protocol limitations also prevent transitions between user interaction devices to be performed, which reinforces the effort in creating

a mechanism that provides cloud integration and coordinated between various access devices.

Research on usability, evaluation and composition of interfaces for different user types the cloud, users' behavior in the cloud, integration of services in ad-hoc format, data visualization and transparency of testing devices, are examples of important research to promote better human cloud interaction.

Few studies are found about the human interaction cloud and how people are interacting with information in the cloud. Despite the lack of research in the area of interaction in clouds, resources such as processing, identity, connectivity and sensors are increasingly present in objects, devices and places.

REFERÊNCIAS

- Anderson, J., Rainie, L. The future of cloud computing. *Reports of Pew Research Center's Internet & American Life Project*, 2010, Jun 11. Disponível em <http://www.pewinternet.org/Reports/2010/The-future-of-cloud-computing.aspx>.
- Ardissono, L., Goy, A., Petrone, G. and Segnan, M. From Service Clouds to User-Centric Personal Clouds. In *proceedings of the 2009 IEEE International Conference on Cloud Computing*, IEEE Computer Society, 2009, pp. 1-8.
- Bång, M., Larsson, A., Berglund, E., e Eriksson, H. Distributed user interfaces for clinical ubiquitous computing applications. *International Journal of Medical Informatics* 74, 7-8 (2005), 545-551.
- Bringué, X.; Sábada, C. (Ed). *La Generación Interactiva en Iberoamérica*. Fundacion Telefonica, 2009. http://www.generacionesinteractivas.org/?page_id=660.
- Davidson, D. *Cross-Media Communications: An Introduction to the Art of Creating Integrated Media Experiences*. Unknown, 2010.
- England, D. HCI issues for emerging autonomic cloud based systems. *HCI 2009 workshop*. Disponível em <http://cloud.cms.livjm.ac.uk/iautonomic/>.
- Filgueiras, L., Correa, D., Oliveira Neto, J., Facis, R. X-gov planning: how to apply crossmedia to government services. In: *ICDS 2008*, IEEE, Los Alamitos, 2008, pp. 140-145.
- Horrigan, J. B. Use of cloud computing applications and services. *The Pew Internet & American Life Project*, 2008, Abril-Maio, p. 9.
- Kozma, R.B. Learning with Media. *Review of Educational Research* 61, 2 (1991), 179 -211.
- Lay, D. Ubiquitous computing: Emerging technologies for learning. *British Educational Communications and Technology Agency* (Becta 2007), Vol, 2(6):64 a 79.
- Luyten, K. e Coninx, K. Distributed user interface elements to support smart interaction spaces. *Multimedia, Seventh IEEE International Symposium on*, (2005), 8 pp.
- McFedries, Paul. The cloud is the computer. *IEEE Spectrum*, vol. 45, 2008, p. 20.
- McLoughlin, M. Bridging interaction clouds: Exploring collaborative interaction across assemblies of mobile and embedded technology. *ECSCW 2009 workshop*. Disponível em <http://bridgingclouds.blogspot.com/>.
- Melchior, J., Grolaux, D., Vanderdonckt, J., e Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, ACM (2009), 69–78.
- Miyamaru, F. Transições entre mídias: um arcabouço tecnológico para aplicações de mídia cruzada. *Dissertação de mestrado*. Escola Politécnica da Universidade de São Paulo, 214p. 2010.
- OASIS. Organization for the Advancement of Structured Information Standards. <http://www.oasis-open.org/>.
- Patterson, D., Armbrust, M. et al. Above the clouds: A Berkeley view of cloud computing. *Technical Report No. UCB/EECS-2009-28*, UC Berkeley, Feb 2009.
- Preti, João P., Filgueiras, Lucia. “Interação em Nuvens”. In: *IHC - IX Simpósio de Fatores Humanos em Sistemas Computacionais*, Belo Horizonte-MG, Brazil, Oct 2010.
- Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, vol. 39, 2009, pp. 50-55.