

PEDRO M. KAYATT

**UBIQUOS - AR CABOUÇO MULTIPLATAFORMA
PARA JOGOS COLABÓRATIVOS UTILIZANDO
INTERFACE DISTRIBUÍDA DE USUÁRIO**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

São Paulo
2015

PEDRO M. KAYATT

**UBIQUOS - AR CABOUÇO MULTIPLATAFORMA
PARA JOGOS COLABÓRATIVOS UTILIZANDO
INTERFACE DISTRIBUÍDA DE USUÁRIO**

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Ricardo Nakamura

São Paulo
2015

FICHA CATALOGRÁFICA

Kayatt, Pedro Matsumura

UBIQUOS - Arcabouço Multiplataforma para Jogos Colaborativos utilizando Interface Distribuída de Usuário/ P. M. Kayatt. São Paulo, 2015.

52 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

A minha familia, especialmente ao meu Tio Chico e Tia Cê que sempre me ajudaram, e à minha amada Keila com quem já formei nova familia

AGRADECIMENTOS

RESUMO

O aumento no número de dispositivos móveis de alto desempenho e a disponibilidade de conexões sem fio permitiram o desenvolvimento de interfaces que são compartilhadas entre diversos dispositivos; tais interfaces são chamadas de Interfaces Distribuídas de Usuários.

Esta pesquisa foca no desenvolvimento de um arcabouço que visa simplificar o desenvolvimento de jogos digitais que utilizem interfaces distribuídas, assim como analisar o atual estado da arte de outras tecnologias e arcabouços.

É proposto um estudo de aceitação, utilizando parâmetros quantitativos e qualitativos, afim de comprovar as características necessárias para uma experiência fluída para o usuário. Tal tarefa é analisada através da criação de um jogo protótipo onde diversos jogadores podem jogar juntos e compartilhar informações entre diversos dispositivos.

ABSTRACT

The increasing number of high-end mobile devices and increased connection availability have allowed on the development of interfaces that can be shared between multiple devices; such interfaces are called Distributed User Interfaces. This research aims the development of a framework to simplify the development of games using distributed interfaces, as well as to assay the current state-of-art of technologies and frameworks.

We propose a study of the acceptance, using both qualitative and quantitative parameters, in order to verify compliance with the necessary characteristics for a fluid user experience. This is accomplished through the creation of a prototype game in which several players can play together and share information through many devices.

SUMÁRIO

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

1 Introdução	11
1.1 Interfaces Distribuídas	14
1.2 Objetivos	17
1.3 Contribuições originais	18
1.4 Organização	19
2 Interfaces Distribuídas e Jogos	20
3 Trabalhos Relacionados	23
3.1 Tecnologias de Conectividade	23
3.1.1 ZeroConf	23
3.1.2 WiFi Direct	24
3.2 Ferramentas de Interfaces Distribuídas	25
3.2.1 StoryML	25
3.2.2 VhCVE	26

4 O arcabouço Ubiquos	27
4.1 Aspectos de Implementação	31
4.2 Considerações Multiplataforma	37
5 Resultados	39
6 Conclusões	47
Referências	49
Apêndice A – Formuário de Avaliação do Protótipo do arcabouço Ubiquos	52

LISTA DE ILUSTRAÇÕES

1	Protótipo: Quatro pessoas jogando Dominó usando Interfaces Distribuídas.	13
2	Desempenho x Latência de acordo com gêneros de jogos. (CLAYPOOL; CLAYPOOL, 2006)	14
3	O modelo de referência de arquitetura CAMELEON-RT. A forma semelhante a uma flor, indica componentes aberto-adaptativos. A forma de uma miniatura de gerenciador-adaptador, indica componentes fechados-adaptativos. Flechas indicam o curso de informação, e linhas são ligações bidirecionais.(BALME et al., 2004)	16
4	Conceitos básicos necessários para a meta-modelagem do modelo referencial. A parte esquerda é sobre as entidades digitais, a parte direita sobre as entidades físicas e a parte central sobre entidades conceituais. A parte extrema da direita descreve onde os mapeamentos foram aplicados.(DEMEURE et al., 2005) . .	17
5	Nintendo WiiU Console - Utiliza um Tablet como segunda tela. . .	21
6	StoreML player - separando usuários e conteúdo. (HU, 2003) . .	26
7	Estrutura do arcabouço Ubiquos.	30
8	Procurando um Servidor através de mensagens Broadcast. . . .	35
9	Como um terceiro jogador é recebido pelo "handshake"automático do arcabouço.	36

10	Diagrama de Classes Simplificados do Jogo Domino Gang desenvolvido no motor de jogos Cocos2d-x	38
11	Diagrama de Atividades que leva o jogador a utilizar o arcabouço Ubiquos em uma partida com DUIs no protótipo.	40
12	Tela inicial do protótipo de jogo utilizado para o arcabouço Ubiquos, o botão a direita leva a tela de utilização da Interface Distribuída.	41
13	Tela de configuração de modo de jogo para Ubiquos, com definições do número de dispositivos a se conectar e seu papel ("Mesa"ou jogador).	42
14	Tela de espera por jogadores esperando as comunicações automáticas do arcabouço Ubiquos conectar-se a outros dispositivos.	43
15	Protótipo: Jogo de dominó utilizando o arcabouço Ubiquos.	44

LISTA DE TABELAS

1	Processamento Interno das Mensagens do arcabouço Ubiquos	37
2	Plataformas suportadas pelo protótipo e o arcabouço Ubiquos	43
3	Respostas ao Formulário de Avaliação do Protótipo	45

LISTA DE ABREVIATURAS E SIGLAS

USP	Universidade de São Paulo
GUI	<i>Graphical User Interface</i> - Interface Gráfica de Usuário
DUI	<i>Distributed User Interface</i> - Interface Distribuída de Usuário
4C	Computação, Comunicação, Coordenação e Configuração
VIGO	Visão, Instrumentos, Governadores e Objetos
RA	Realidade Aumentada
RV	Realidade Virtual
IP	<i>Internet Protocol</i>
LAN	<i>Local Area Network</i> -Rede Local
P2P	<i>Peer-to-peer-</i> Ponto a ponto
AP	<i>Access Point</i> - Ponto de Acesso
CVE	<i>Collaborative Virtual Environment</i> - Ambiente Virtual Colaborativo
API	Interface de Programação de Aplicações
UDP	<i>User Datagram Protocol</i>
TCP	<i>Transmission Control Protocol</i>
ID	Identificação
SO	Sistema Operacional

1 INTRODUÇÃO

Não muito após a popularização dos computadores pessoais entendeu-se que estes eram ferramentas que, devido a sua versatilidade nos usos, possuem diversas maneiras de interagirem com os usuários.

O estudo de tais interfaces entre Computador e seus usuários foi descrita por Card, Newell e Moran (1983) e popularizada como Interface Homem-Computador. Desde então diversas novas teorias e hipóteses foram estudadas, tornando-se um ramo científico que busca envolver diversas áreas afim de aprimorar usabilidade, compreensão e desempenho.

Atualmente estamos habituados com Interfaces Gráficas de Usuários, também conhecidas como GUI (do inglês *Graphical User Interface*), porém existem interfaces focadas na interação através de voz, Interfaces Vocais de Usuários, e até Interfaces Distribuídas de Usuários (DUI - *Distributed User Interface*). Tal termo refere-se a interfaces que conectam diversas telas, diversos dispositivos e através de várias plataformas como descrito em (ELMQVIST, 2011).

A proliferação de dispositivos como *smartphones* e *tablets* criaram um cenário no qual convivemos com diversas telas diariamente. Entretanto, tais telas existem dentro de suas próprias barreiras, não interagindo uns com os outros e apenas co-existindo.

O desenvolvimento de interfaces distribuídas sempre teve como uma de

suas barreiras o alto custo de telas e sua quantidade limitada disponível. Além disso, a comunicação física entre dispositivos acontecia através de cabos e sistemas que tornavam as telas fixas ou com mobilidade reduzida.

Apesar de todas estas limitações superadas pelo atual cenário, ainda possuímos uma falta de padronização em Interface Distribuídas e as comunicações entre dispositivos focam mais na troca de dados entre usuários não assíncronas.

Jogos digitais são sistemas computacionais que possuem interfaces complexas e informam continuamente diversos dados à seus usuários. Sendo assim, acredita-se que jogos podem se beneficiar com tais interfaces. No caso de jogos colaborativos tal aplicabilidade, no caso de DUIs, é ainda mais interessante, pois permitimos que cada usuário tenha seu conjunto de dados particular em apenas uma tela, porém compartilhe uma tela com dados comuns a todos jogadores.

Ficou decidido, então, que esta pesquisa tem como objetivo o desenvolvimento de um arcabouço que simplifique o uso de tais avanços para criar jogos colaborativos com Interfaces Distribuídas.

O arcabouço proposto, nomeado Ubiqus, permite o uso de diversos dispositivos e de plataformas distintas serem utilizados em conjunto. Será possível analisar as vantagens de metodologias e arquiteturas de programação de um código de rede eficiente (HUSTON; JOHNSON; SYYID, 2004) focando na colaboração para o desenvolvimento de jogos.

Visando que o arcabouço possa ser efetivamente testado foi planejado o desenvolvimento de um jogo protótipo. O protótipo teve por objetivo colocar o arcabouço em um cenário de utilização prático e entender se as definições iniciais do arcabouço satisfariam as condições necessárias para agilizar o de-

senvolvimento de interfaces distribuídas e inferir possíveis problemas relacionados às diferentes plataformas.

O protótipo em questão resultou em um jogo de Dominó (Figura 1)¹, onde cada jogador deve ver as peças de sua mão através da tela de seu próprio smartphone e um dispositivo (preferencialmente um tablet) deve mostrar todas as peças já jogadas (sendo a mesa do jogo), sendo tal tela compartilhada por todos os jogadores.



Figura 1: Protótipo: Quatro pessoas jogando Dominó usando Interfaces Distribuídas.

É importante notar que uma boa experiência de usuário, tratando-se de uma DUI, é necessário que o tempo de resposta das mensagens transmitidas entre usuário não possa ser percebida pelo usuário (YANNAKAKIS; HALLAM, 2009). Segundo as ideias da pesquisa de (CLAYPOOL; CLAYPOOL, 2006) demonstradas na Figura 2, espera-se que os tempos de resposta mínimos para tal sejam diretamente relacionados ao gênero do jogo e à latência dos pacotes transmitidos.

¹Que pode ser baixado através do link <http://www.nakedmonkey.mobi/games/domino/>

Além do desenvolvimento do arcabouço, foi desenvolvido um estudo de aceitação utilizando tanto métricas quantitativas (como tempos de resposta) e qualitativas (experiência do usuário), através de metodologias de controle de qualidade e experimentação com usuários.

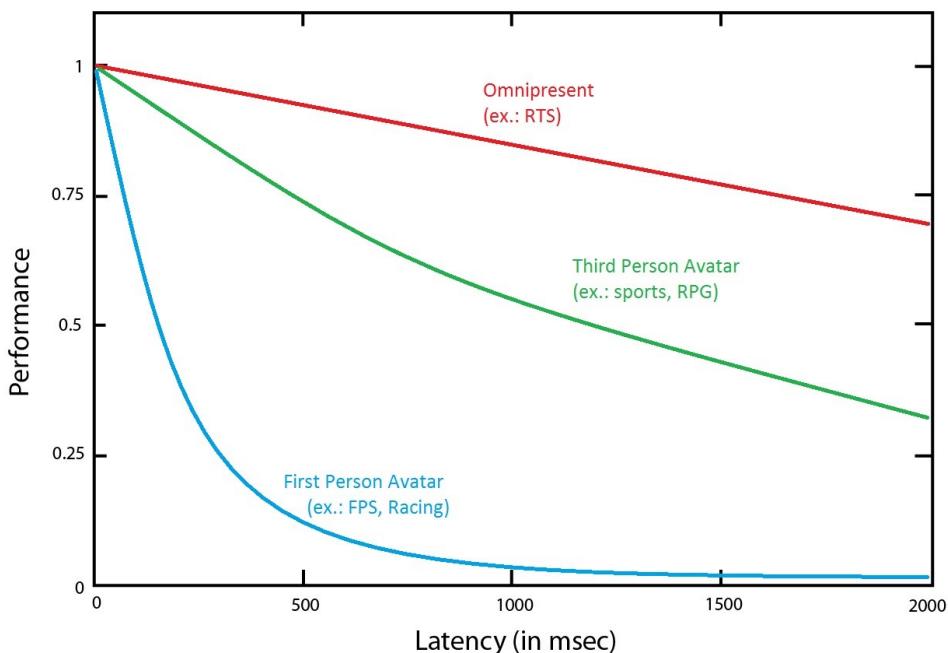


Figura 2: Desempenho x Latência de acordo com gêneros de jogos. (CLAYPOOL; CLAYPOOL, 2006)

1.1 Interfaces Distribuídas

Interfaces Distribuídas de Usuários foram definidas por Elmquist (2011) como sendo:

Uma interface distribuída de usuário é uma interface de usuário cujos componentes estão distribuídos através de uma ou mais dimensões de entrada, saída, plataforma, espaço e tempo.²

Apesar da definição de DUIs ser relativamente recente, 2011, sua teoria

²A distributed user interface is a user interface whose components are distributed across one or more of the dimensions input, output, platform, space, and time.

já era estudada por anos. Diversos modelos para trabalhar e formalizar as utilizações de DUI foram propostos, tais modelos são brevemente citados em próximas seções.

Entretanto tais modelos tinham aplicações mais específicas ou eram limitados a sistemas distribuídos não genéricos. Tais características tornam inviável a utilização destes modelos para o desenvolvimento de jogos e outras aplicações mais genéricas.

Entre os modelos propostos para se trabalhar com DUIs podemos destacar os seguintes: CAMELEON-RT (BALME et al., 2004); O modelo 4C (Computação, Comunicação, Coordenação e Configuração) e o Modelo de Referência de (DEMEURE et al., 2005), cujos modelos são voltados exclusivamente para o modo como as interfaces são compartilhadas.

O modelo de referência de arquitetura CAMELEON-RT possui três níveis de abstração como descrito na Figura 3. Tal trabalho se propõe resolver os problemas de interfaces distribuídas, migráveis e plásticas.

O trabalho de Demeure et al. (2005) propõe um modelo referência para classificação de diferentes tipos de DUIs, classificando-as de acordo com 4 dimensões: Computação (o que é distribuído), comunicação (quando é distribuído), coordenação (quem é que distribui) e configuração (de onde e para onde a distribuição é operada). Para tal o modelo define uma noção de habitat como o conceito central para distribuição, os conceitos básicos para o compreendimento dos modelos presentes na pesquisa estão descritos na Figura 4.

Por outro lado, existem arquiteturas de software que podem ser adaptadas para o desenvolvimento de DUIs, buscando encontrar um meio termo entre padrões de programação e novas propostas. Estes são: o modelo MVC (Mo-

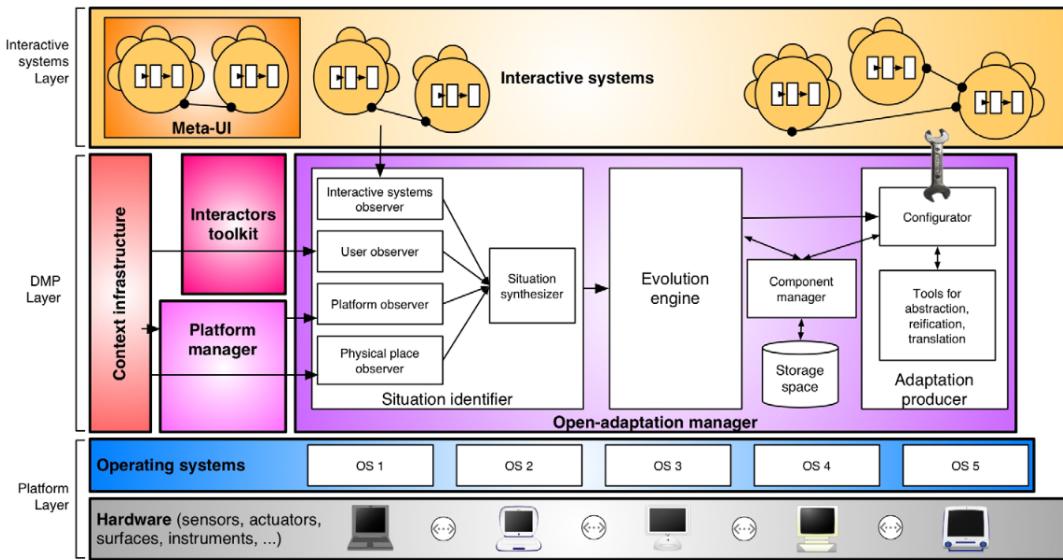


Figura 3: O modelo de referência de arquitetura CAMELEON-RT. A forma semelhante a uma flor, indica componentes aberto-adaptativos. A forma de uma miniatura de gerenciador-adaptador, indica componentes fechados-adaptativos. Flechas indicam o curso de informação, e linhas são ligações bidirecionais.(BALME et al., 2004)

de Visão e Controlador) como no paradigma sugerido por (GRAHAM; URNES; NEJABI, 1996) e o modelo VIGO (Visão, Instrumentos, Governadores e Objetos) proposto por (KLOKMOSE; BEAUDOUIN-LAFON, 2009).

O modelos citados podem ser utilizados no desenvolvimento de aplicações DUI, porém nenhum deles possui o foco em jogos e não são necessariamente síncronos ou multiplataforma, características que os tornam não aplicáveis para em nosso escopo. Além disso, muitos dos modelos são extensos e rígidos, criando problemas ao tentar adaptá-los a diversos motores de jogos e diferentes plataformas.

Pesquisadores como (FILIPPO et al., 2007) demonstram ambientes colaborativos que utilizam de características de jogos com propósito educacional, usando Realidade Aumentada e Virtual, exemplificando a importância de ambientes colaborativos.

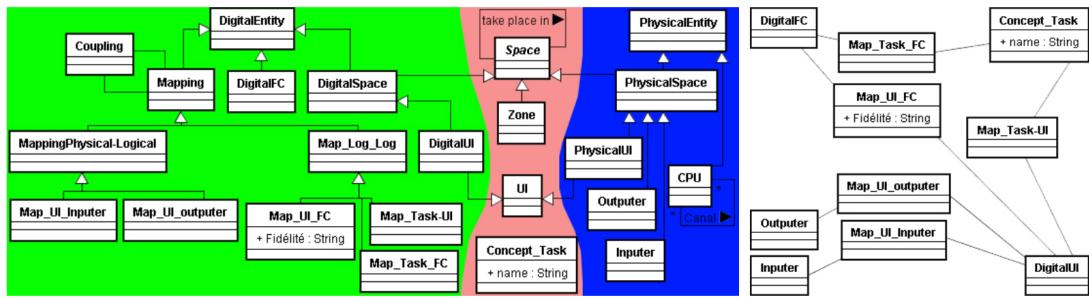


Figura 4: Conceitos básicos necessários para a meta-modelagem do modelo referencial. A parte esquerda é sobre as entidades digitais, a parte direita sobre as entidades físicas e a parte central sobre entidades conceituais. A parte extrema da direita descreve onde os mapeamentos foram aplicados.(DEMEURE et al., 2005)

1.2 Objetivos

O objetivo desta pesquisa é desenvolver um arcabouço que auxilie um desenvolvedor de jogos a implementar uma interface distribuída em seu jogo.

O desenvolvimento de jogos atualmente utiliza de diversos motores de jogos que possibilitam criar aplicativos que sejam utilizados em diversas plataformas diferentes. Isso é importante pois o sucesso de seu jogo é relacionado a quantidade de usuários que o utilizam, portanto disponibilizar para mais plataformas aumenta o potencial de usuários que podem jogá-lo.

Entende-se que o arcabouço deva ser o mais modular possível, incluindo uma independência de plataforma, tal qual permita que desenvolvedores de diversos sistemas possam usufruir da facilidades criadas pelo arcabouço e não estejam limitados a algum sistema operacional ou motor de jogos.

Nos trabalhos observados foi possível encontrar uma grande quantidade de vantagens em trabalhos colaborativos. Jogar colaborativamente induz um pensamento de equipe que permite jogadores se entrosarem positivamente afim de superar um objetivo comum.

À prática de jogos colaborativos em jogos não digitais é algo comum, como em alguns jogos de tabuleiro. Desta forma entende-se que a transcrição de mecânicas como tais para jogos digitais de interfaces distribuídas pode ser vantajoso, dado que os usuários se beneficiam de mecânicas já conhecidas e possuem curvas de aprendizado mais tênues em tais casos.

Nota-se, também, uma tendência em produtos que desenvolvem a ideia de interfaces assimétricas. Como visto na seção Trabalhos Relacionado e em alguns projetos como SmartGlass da Microsoft (EMMERICH; LISZIO; MASUCH, 2014) e outros *Companion Apps* (NANDAKUMAR; MURRAY, 2014), contribuindo para a ideia que o arcabouço possa também ser aproveitado para desenvolvimento de aplicativos com tais recursos.

A pesquisa também propõe que o arcabouço deva garantir uma experiência de usuário agradável e fluída, assegurando que os usuários não sejam prejudicados pelas interfaces distribuídas; ao contrário, tenham experiências aprimoradas.

1.3 Contribuições originais

Estudos sobre Interfaces Distribuídas estão se tornando cada vez mais popularizados, entretanto suas aplicações na área de Jogos Digitais ainda são rasas e necessitam mais estudos.

Outros arcabouços auxiliam no desenvolvimento de comunicação em rede, porém não focados para aplicações de DUIs, como a Ubiquos, além disso poucos são abertos e multi-plataforma.

1.4 Organização

Este trabalho esta divido nas seguintes seções: 1 - Introdução, 2 - Interfaces Distribuídas, 3 - Trabalhos Relacionados, 4 - O Arcabouço Ubiquos, 5 - Resultados e 6 - Conclusões.

2 INTERFACES DISTRIBUÍDAS E JOGOS

Dado o aumento na popularidade de dispositivos pessoais inteligentes, tais como *smartphones* e *tablets*, podemos apreciar a afirmação de (ELMQVIST, 2011):

Interfaces Distribuídas de Usuários são vitais para a nova geração de sistemas interativos pervasivos e interoparacionais que serão parte do ambiente computacional do amanhã.¹

Jogos são o tipo de aplicação em lojas de dispositivos móveis mais populares (KIMBLER, 2010), sendo, portanto, um ponto interessante para introduzir e popularizar a ideia de Interfaces Distribuídas, visando aprimorar a experiência de usuários de uma forma que seja possível expandir para outras áreas como sugerido pelos estudos de Malone (1982).

Na área profissional de consoles de jogos o lançamento do Nintendo WiiU demonstram um dispositivo que tem como uso exclusivo a jogabilidade assimétrica, como estudado em (MEDEIROS FILHO; CALADO; NEVES, 2013). Junto ao console um tablet com alavancas e botões (como pode ser visto na Figura 5) foca em ser uma tela secundária afim de expandir as interações entre diversos jogadores, ou demonstrar informações dedicadas quando jogando no modo de jogador solo.

¹Distributed user interfaces are vital for the new generation of pervasive and interoperable interactive systems that will make up tomorrow's computing environments



Figura 5: Nintendo WiiU Console - Utiliza um Tablet como segunda tela.

Isto demonstra uma tendência na aceitação de Interfaces Distribuídas para um futuro próximo, assim como uma grande capacidade em aprimorar experiências em jogos. Por exemplo o jogo Nintendo Land é uma coleção de diversos mini-jogos, onde quatro jogadores executam papéis semelhantes e um jogador utiliza o tablet com um papel único e geralmente contrário aos outros quatro jogadores.

Outra demonstração comercial de utilizar diversos dispositivos em conjunto é através das franquias da Ubisoft, grande editora internacional de jogos, em especial Assassin's Creed.

Em diversas edições desta franquia, a partir do jogo com o título de *Black Flag*, o jogo apresenta mini-jogos [REF] aonde os jogadores podem completar pequenas missões e receberem recompensas que os beneficiam na história principal.

Tais mini-jogos podem ser jogados através de aplicativos, nas plataformas Android e iOS, que se conectam através da rede e adquirem informações do perfil do usuário. Não apenas isto, mas se o jogo estiver sendo jogado simultaneamente à abertura do aplicativo é possível extrair informações do jogo para maior imersão, como: Posição no mapa, próximas missões, recentes

mensagens e outros.

Por outro lado, acredita-se que um dos pontos principais no desenvolvimento do arcabouço é sua adequação a diferentes plataformas; como smartphones, tablets e desktops. Podendo assim tirar vantagem de suas semelhanças tecnológicas e permitir que todos estes dispositivos sejam usados independente de marcas ou sistemas operacionais. Tal característica permite que mais usuários possam se beneficiar da tecnologia e portanto disseminar melhor a ideia de DUIs.

Assim sendo, Ubiquos visa usar tecnologias que sejam compatíveis e que tenham suporte multiplataforma, assim como ser adaptável a outros serviços e motores de jogos populares.

Além do problema de manter os dados da mensagem concisos, o uso de diversos dispositivos através da rede necessita o desenvolvimento de um código de rede focado em uma experiência fluída e natural ao usuário, da mesma forma que entradas de toque e outros sensores em seus próprios dispositivos.

Para averiguar-se tal características a respeito dos tempos de resposta podem ser elaborados sessões de teste de usabilidade(RUBIN; CHISNELL, 2008), validando a experiência do usuário e obtendo valores mínimos a máximos limites que podem ser interpretados para uma experiência natural.

Não obstante, é importante que o arcabouço possa abstrair funcionalidades complexas e repetitivas, permitindo que o desenvolvedor do jogo possa focar apenas na troca de mensagem relativas ao jogo em si.

3 TRABALHOS RELACIONADOS

Neste capítulo são analisados algumas tecnologias e trabalhos que influenciaram o desenvolvimento da Ubiquos e que possuem influência no desenvolvimento de Interfaces Distribuídas de Usuários.

3.1 Tecnologias de Conectividade

Uma preocupação necessária ao trabalhar com interfaces distribuídas é em como iremos efetuar as conexões entre as interfaces. É importante ressaltar que as conexões devam ser de forma mais natural possível, afim de desoneras os usuários de complicações desnecessárias.

3.1.1 ZeroConf

Para uma conexão simplificada entre dispositivos o trabalho de (LEE et al., 2007) se destacou. Neste é descrito The Zero Configuration Networking, ou Zeroconf, que visa resolver o problema de diversos dispositivos conectados a mesma rede de protocolo IP desejando compartilhar um serviço.

O uso mais popular de tal trabalho é o Apple AirPlay, que permite que dispositivos desta marca se conectem ao produto AppleTV e toquem vídeos, músicas e imagens de outras fontes através de rede local. Neste caso o Zeroconf é utilizado para descobrir se existe uma AppleTV conectada à rede e

também para lidar com a troca de dados entre os dispositivos.

Outro uso comum do Zeroconf é em impressoras. Quase todas impressoras modernas possuem tal tecnologia embarcada, permitindo de forma simplificada encontrá-las em redes locais.

Por outro lado, o Zeroconf só funciona em sub-redes, o que limitaria o desenvolvimento de interfaces distribuídas de usuários a tais ambientes.

Entretanto, Zeroconf demonstrou um modo eficiente de compartilhar a existência de um serviço em uma rede local (LAN - Local Area Network), e como permitir uma conexão à este serviço uma tarefa simples para qualquer usuário.

3.1.2 WiFi Direct

Outra tecnologia que chamou a atenção foi o WiFi Direct, cujo propósito é contornar uma dos pontos deficientes do padrão 802.11, como descrito por (CAMPUS-MUR; GARCIA-SAAVEDRA; SERRANO, 2013) sendo a conexão ponto-a-ponto (P2P - peer to peer), sem a necessidade de um Ponto de Acesso (AP - Access Point).

A característica principal na implementação do WiFi Direct é que, apesar das capacidades AdHoc descritas pelo padrão 802.11, as funções de AP e Cliente são definidas dinamicamente e não são pré-definidas (como de costume).

Infelizmente a tecnologia é notoriamente recente (2013) e sua implementação ainda não é muito difundida. Apesar disso, como alternativo, os dispositivos atuais possuem uma função comumente conhecida como *Compartilhar Internet*¹, na qual o dispositivo se transforma em um AP e permite que outros

¹Share Internet

possam conectar-se a ele.

3.2 Ferramentas de Interfaces Distribuídas

É interessante notar que diversos estudos foram efetuados na área de DUIs. Naturalmente alguns deles propuseram ferramentas e modelos que facilitem o desenvolvimento. Nesta seção alguns trabalhos são analisados e avaliados em como podem contribuir, direta ou indiretamente, no desenvolvimento da Ubiquos.

3.2.1 StoryML

Na área de ferramentas DUIs podemos ressaltar o arcabouço StoryML, descrito por (HU, 2003).

O foco da pesquisa de Hu é na criação de uma linguagem que permite construir estórias interativas através de códigos com alto nível de abstração, modelando e facilitando o desenvolvimento de atividades com interfaces distribuídas.

A plataforma também permite que o criador de conteúdo separe atividades que devem ser exibidas em apenas uma interface e em quais dispositivos devem ser executadas. A figura 6 ilustra como um *Player* do StoryML deve ser modelado.

Entretanto, StoryML tem o foco na interação de vídeos e na distribuição de mídias, invés da colaboração e interação em tempo real como previsto por jogos e em nossa pesquisa.

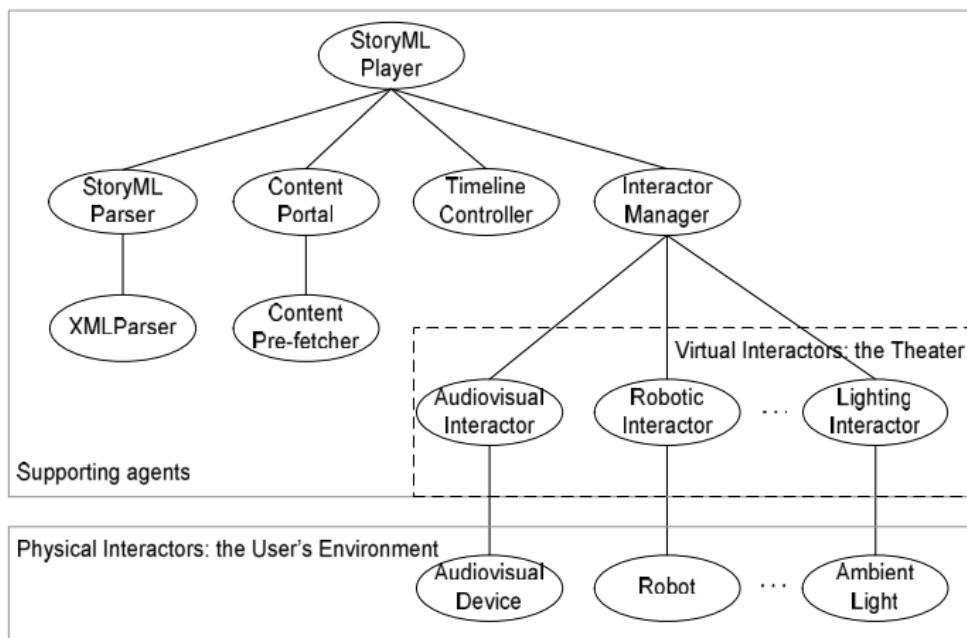


Figura 6: StoryML player - separando usuários e conteúdo. (HU, 2003)

3.2.2 VhCVE

Além do StoryML, temos também o arcabouço VhCVE (descrito em (BRAUN et al., 2009)) que foca em Ambientes Virtuais Colaborativos (CVEs) onde usuários podem interagir com outros usuários através de mensagens de textos ou utilizando seus microfones.

O projeto ainda desenvolve ferramentas de Visão Computacional e até simuladores de Física, criando espaços virtuais tridimensionais complexos e colaborativos.

Infelizmente, dada a sua complexidade capacidades multiplataformas, como execução em dispositivos móveis tais quais *smartphones* e *tablets*, seriam deficitárias e adaptar tais ferramentas para ação em conjunto com motores de jogos populares seria complexo.

4 O AR CABOUÇO UBIQUOS

Neste capítulo o arcabouço Ubiquos para interfaces distribuídas em jogos é descrito em sua proposta e implementação.

Primeiro devemos estabelecer um consenso sobre o que é um arcabouço, este termo também utilizado como *Software Framework* é bem descrito por (FAYAD; SCHMIDT, 1997) tal qual:

Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

Foi planejada a criação de uma arquitetura modular para jogos utilizando DUIs, focando no reuso e na adaptabilidade para diferentes motores de jogos. Com este propósito, soluções tecnológicas foram pesquisadas, levando-se em conta os requisitos de desenvolvimento multiplataforma e eficiência na comunicação em rede.

Tem-se então como requisitos que nosso arcabouço possa ser utilizado por diferentes plataformas e diferentes linguagens de programação. Além disso, ressaltamos que é importante que a biblioteca possa ser acoplada sem alteração do ciclo de jogo principal (*game loop*).

Não obstante, o arcabouço Ubiquos deve se comunicar de forma eficiente, afim de minimizar o impacto da transmissão de mensagens referentes à in-

terface distribuída na experiência de usuário. Pode-se utilizar como vantagem a co-existência dos dispositivos em um mesmo ambiente físico, permitindo a utilização de redes locais.

Portanto, inferimos como requisitos para o arcabouço tais características:

- Modularidade
- Multi-plataforma
- Fácil acoplamento nos jogos
- Fácil conexão entre dispositivos
- Gerenciamento de rotinas automatizáveis
- Troca de mensagem rápida (imperceptível para o usuário)

A arquitetura do arcabouço Ubiquos contempla três componentes principais: o Cliente, o Administrador e o Jogo. O Cliente é responsável por gerenciar a conexão entre os dispositivos, fechando ou estabelecendo conexões para endereços desejados e também é responsável pelo gerenciamento de mensagens trocadas pela rede.

O Administrador é uma classe *Singleton*,(GAMMA et al., 1994), sua função é gerenciar a interface o jogo e as conexões com outros dispositivos. De acordo com a proposta deste arcabouço, a implementação dos componentes Cliente e Administrador deve suportar diversas plataformas.O Administrador é a parte do arcabouço responsável por empilhar as mensagens recebidas pelo Cliente de modo assíncrono e possibilitar que as mensagens sejam lidas de modo síncrono pelo Jogo. O Administrador também gerencia as conexões iniciais, procurando por servidores do arcabouço Ubiquos na rede local e criando servidores se necessário.

O último componente, Jogo, é o programa interativo que deve implementar as mecânicas de jogo por si próprio. Seguindo a proposta da pesquisa, este componente pode ser implementado em um motor de jogo de preferência do desenvolvedor, porém a maioria dos jogos possuem um *Ciclo de Jogo* bem definido, no qual sugerimos um método (*Processar Mensagens de Rede*) que deverá processar todas as mensagens trocadas entre outros dispositivos.

Seguindo essa recomendação, as mensagens do arcabouço Ubiquos serão processadas cada vez que o jogo alcance tal método de processamento de mensagens, este fato é diretamente relacionado à taxa de quadros do jogo. Ou seja, cada vez que a taxa de quadros do jogo tiver uma queda a velocidade com que as mensagens são processadas também sofrerão uma queda, o que pode acarretar em perda de sincronização com outros dispositivos que não sofreram tal queda.

Perdemos então um pouco de consistência no processamento das mensagens em troca de uma maior flexibilidade do arcabouço, permitindo que o desenvolvedor não se atenha a problemas de concorrência, caso utilizássemos sistemas *multi-thread*, e possa utilizar o próprio ciclo de jogo independente de motor gráfico ou linguagem.

Em resumo temos o Cliente lidando com as mensagens, o Administrador criando as conexões necessárias para o Cliente se conectar e criar uma interface para que o Jogo possa adquirir mensagens e por sua vez processá-las. Um diagrama do arcabouço proposto pode ser observado na Figura 7.

Levando em consideração os estudos previamente discutidos ficou claro que a implementação de conexão ponto a ponto com dispositivos envolve o desenvolvimento de código para a criação de servidores, publicação e gerenciamento de listas de servidores, entre outras tarefas. No entanto, consideramos que essa implementação é repetitiva (entre diferentes projetos) e seu

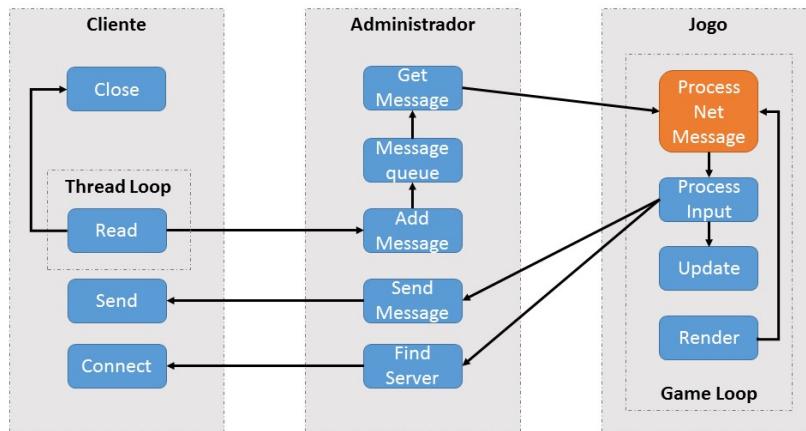


Figura 7: Estrutura do arcabouço Ubiquos.

código pode ser reutilizado.

Em alguns momentos do jogo existem mensagens esperadas de um determinado jogador e existe a possibilidade de este jogador não estar mais conectado ao jogo. Muitos gêneros de jogos podem ter suas regras quebradas, por exemplo, em um jogo de turnos esperar pela jogada de um jogador que foi desconectado faz com que o jogo fique travado na vez do jogador ausente. Devido a tal característica foram criados duas formas de lidar com a saída de jogadores: Esperada e Não-Esperada.

Por esta razão foi definido que o arcabouço deve se encarregar de gerenciar tais passos, os deixando o máximo automatizados quanto possível, apesar de não interferir nas escolhas do estilo do jogo a ser desenvolvido.

4.1 Aspectos de Implementação

Após a apresentação da visão geral do arcabouço Ubiquos, devemos discutir algumas questões referentes à sua implementação.

Visando o melhor uso da já infraestrutura de redes sem fio existente, assim como as vantagens de desempenho em relação a outros padrões, como demonstrado por Lee, Su e Shen (2007), o protocolo 802.11 (WiFi) foi escolhido como tecnologia para comunicação no desenvolvimento do arcabouço.

Outro fato que motivou essa escolha para esta escolha foi a evolução das redes AdHoc, atualmente disponíveis em quase todos dispositivos (XU; GERLA; BAE, 2002), e possuindo a capacidade de criar redes mesmo em lugares remotos, sem acesso à Internet.

Além disso, o protocolo selecionado também é compatível com os motores de jogos mais populares no período do desenvolvimento deste projeto e pode ser implementado em diversas linguagens de programação.

Os trabalhos relacionados e outros arcabouços estudas levaram à escolha da API Socket (STEVENS; FENNER; RUDOFF, 2007) para executar a comunicação entre dispositivos. Tal API é conhecida por ser rápida e multiplataforma, além de ser implementada em muitas linguagens de programação e comumente utilizada no desenvolvimento de serviços de comunicação de jogos multi-jogador (JAMIN, 2008).

Inicialmente a implementação da troca de mensagens foi feita utilizando apenas o protocolo TCP, dado que ele inclui mecanismos de transmissão confiável de dados (comparado ao protocolo UDP) e a infraestrutura de redes sem-fio atual ser robusta o suficiente, tanto para a quantidade de mensagens quanto para a velocidade de troca de mensagens, para gerenciar as comuni-

cações entre dispositivos durante um jogo.

Como discutido anteriormente, o arcabouço Ubiquos deve prover um mecanismo de busca automática de servidores, visto que esse é um dos recursos que pode ser reutilizado entre jogos distribuídos.

Este método deve procurar por jogadores que possuem uma instância do jogo aberta e esperam pela conexão de novos jogadores, o método também deve criar condições para que novos jogadores se unam à partida de acordo com o número especificado de jogadores.

Na implementação desse mecanismos, observamos que o uso somente do protocolo TCP resultava em baixo desempenho da solução, pois o tempo necessário para fazer a varredura de uma rede local era alto e ainda ocorriam falhas na recuperação do endereço IP de um servidor encontrado.

Comumente este método acarretava tempos de 60 a 120 segundos de espera, mesmo quando o *timeout* era definido para tempos inferiores à 300 milissegundos, e muitas vezes a resposta da conexão ao servidor falhava. O aumento do timeout para 1 segundo evitava as falhas na determinação do IP do servidor, mas o tempo total do processo se tornava muito elevado (255 segundos no pior caso).

Tais experimentos nos levaram à decisão de misturar os protocolos TCP e UDP no arcabouço, de modo similar ao trabalho de Karachristos, Apostolatos e Metafas (2008). Tal escolha é devido a capacidade do protocolo UDP de enviar mensagens em *broadcast* para toda a rede local (LAN), permitindo uma implementação mais eficiente do mecanismo de busca de servidores em rede local.

Diversas linguagens de programação foram consideradas para o desenvolvimento do arcabouço. Algumas como Java e Python foram concebidas

com o propósito de serem multiplataforma, porém o fato de serem linguagens interpretadas ou dependerem de um ambiente virtual de execução faz com que seu desempenho seja inferior a linguagens pré-compiladas.

Tendo em vista que o foco do arcabouçou é auxiliar a criação de interfaces distribuídas para jogos entende-se que o desempenho seja um fator crucial. Notoriamente no momento do desenvolvimento desta pesquisa linguagens como C e C++ são as mais comuns no desenvolvimento de jogos, como observado por Lewis e Jacobson (2002) que ressalva alguns motores de jogos populares como Quake 2 e 3 e até mesmo Unreal que as utilizam.

Um dos motivos é que através da linguagem C++ é comum acessar instruções específicas de API gráficas, como OpenGL e DirectX, e até mesmo funções de sistemas. Tal características facilita desenvolvedores de jogos a obter uma experiência mais fluída em seu jogo, seja em taxa de quadros ou efeitos especiais específicos.

Adicionalmente, a linguagem C++ tem recursos específicos para programação orientada a objetos, que por sua vez auxiliam no reuso de código. Adiconalmente motores de jogos que não visam o desenvolvimento direto na linguagem c++, como o motor Unity3D (UNITY, 2009), possuem mecanismos para incorporar código ou bibliotecas C++. Esse foi outro motivo para a seleção dessa linguagem.

Visando reduzir incompatibilidades entre diferentes plataformas e ainda manter padrões de qualidade código ótimos decidiu-se utilizar as estruturas de dados e classes de *threading* derivadas da Biblioteca Padrão do C++ 11¹ tal quais descritas por (JOSUTTIS, 2012).

Com o propósito de auxiliar desenvolvedores em tarefas simples o arca-

¹Standard Library ou STD

bouço contém algumas estruturas pré-definidas como *NetPlayer*, o qual possui informações como *id*, *nome* e *AvatarID*, como sendo dados genéricos de um jogador, não limitados a quaisquer estilos de jogos ou motores de desenvolvimento, porém característicos na maior parte dos jogos multiplayer.

Para criar uma experiência realmente simples, foi definido que o arcabouço deveria ter uma interface de conexão simplificada. Dado tal requisito o programador que deseja utilizar o arcabouço basta invocar o método *connect* com o número de jogadores e automaticamente será inicializada uma procura por servidores dentro da LAN e caso nenhum responda este que procura criará um servidor.

Também foram criadas classes específicas para lidar com a troca de mensagens através do protocolo UDP; *ServerUDP* e *ClientUDP*. A primeira cria um servidor e espera pela mensagem "*lookingServer*" enviada por qualquer cliente. Já a outra classe, *ClientUDP*, envia uma mensagem *broadcast* para toda a rede LAN com a mensagem "*lookingServer*" e recebendo uma mensagem do servidor inicializa o processo de *handshaking* conforme ilustrado na Figura 8.

Após o estabelecimento de uma conexão TCP, o servidor que já executava em outra *thread*, deve criar uma nova *thread* para gerenciar as mensagens específicas de cada cliente. Entretanto, clientes TCP necessitam apenas de uma *thread* para trocar mensagens, além daquela onde o laço principal do jogo é executado.

O processo de criação de diferentes *threads* para manter conexões independentes entre cada cliente e o servidor é comumente chamada de "*worker thread*" e bem descrita pelo trabalho de (HALL, 2001).

O sistema de troca de mensagens é baseado no padrão cliente-servidor,

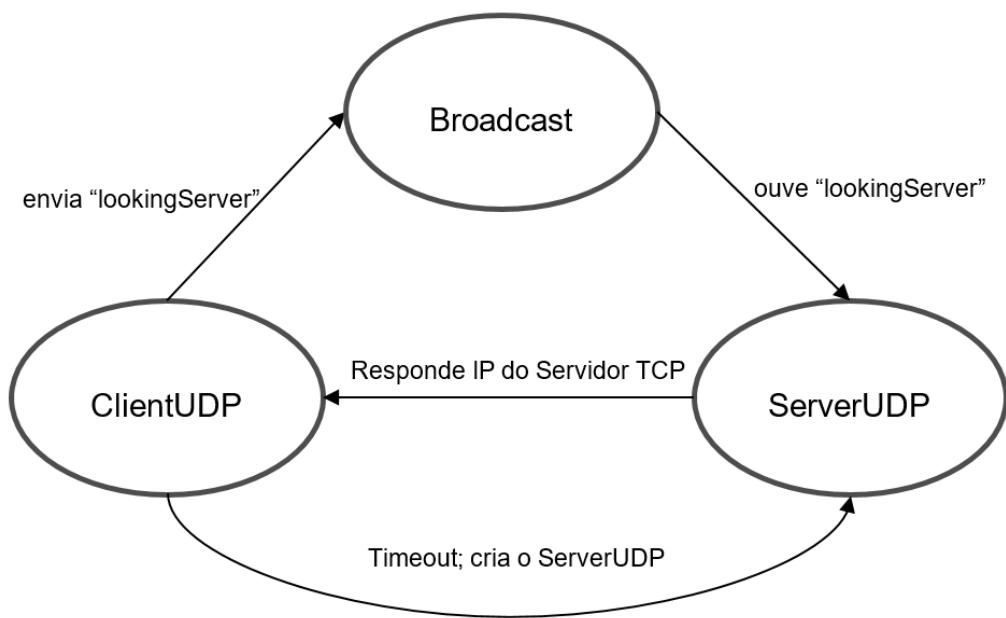


Figura 8: Procurando um Servidor através de mensagens Broadcast.

sendo que todas as mensagens são enviadas para um dispositivo definido como o servidor, independente de qual o jogador alvo da mensagem.

Desta forma o arcabouço é capaz de controlar o estado de todos jogadores, inclusive ser notificado caso algum jogador queira sair do jogo. Infelizmente se o jogador que tiver algum problema de conexão for o dispositivo servidor todos outros jogadores serão afetados.

Um exemplo da troca de mensagens através do servidor assim como uma das funções autônomas da Ubiquos é o gerenciamento de novos jogadores, como ilustrado pela Figura 9. Através deste método o arcabouço pode gerenciar a conexão de novos jogadores, definindo para cada jogador um código identificador (ID).

Através do ID do jogador o arcabouço é capaz de gerenciar o número de jogadores, ou dispositivos, esperados para uma determinada partida. É através deste ID que as mensagens são processadas e retransmitidas de acordo

com o dispositivo alvo, além de permitir um tratamento individual de cada mensagem.

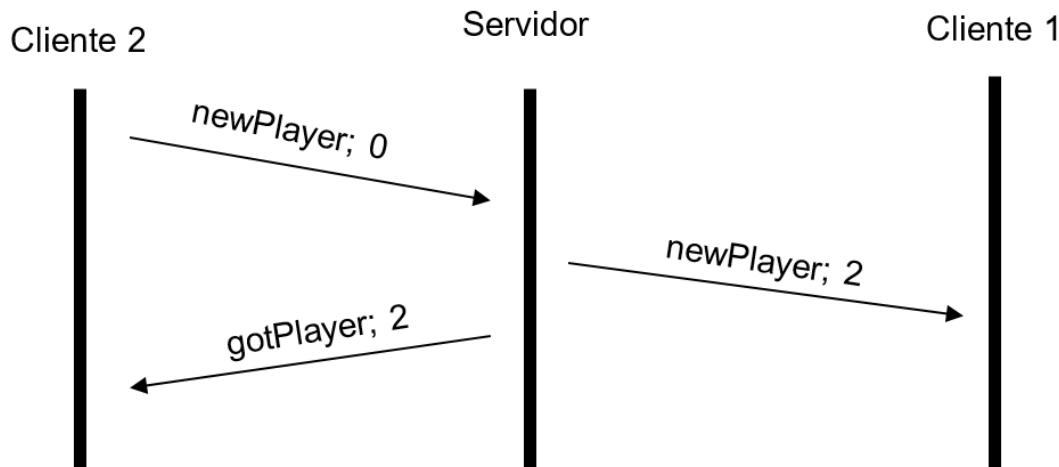


Figura 9: Como um terceiro jogador é recebido pelo "handshake"automático do arcabouço.

A implementação do tratamento de saídas Esperadas e Não-esperadas do jogador foi feita da seguinte forma: No primeiro caso, Esperada, é definido pela saída proposital do jogador de um jogo, utilizando a interface do jogo e buscando um botão de saída. Neste momento é importante que o arcabouço tenha um método que possa ser chamado para fechar a conexão de uma maneira segura, tal método foi chamado de Close. Este método envia uma mensagem "playerQuit."para todos os outros jogadores e reinicia as conexões Socket permitindo novas conexões.

Já o caso de uma saída "Não-Esperada"o jogador é desconectado devido à diversas razões, tais quais: fechamento inesperado do jogo, dispositivo sem bateria, perda do sinal WiFi, etc. A conexão socket retornará o valor 0 como resposta na thread de troca de mensagens, permitindo que os jogadores que continuam conectados se desconectem do jogador perdido sem problemas, adicionando a mensagem "playerQuit."na fila de mensagens recebidas. Deve ficar claro que em ambos os casos esta mensagem deve ser processada pelo

Tabela 1: Processamento Interno das Mensagens do arcabouço Ubiquos

Classe	Menssagem			
Ubiquos	newPlayer	id	name	avatarID
Ubiquos	gotPlayer	id		
Ubiquos	playerQuit.			
Ubiquos	playerQuit	id		
UDPClient	lookingServer	numPlayers	isPlayer	
	connecting		isPlayer	
UDPServer	serverFound			

desenvolvedor do jogo em seus ambientes específicos.

As diversas mensagens e suas codificações específicas utilizadas na Ubiquos podem ser encontrados na Tabela 1. Vale lembrar que em um segundo momento tais mensagens podem ser ainda mais simplificadas, porém como fora alcançado um fluência agradável nas interações tais aprimoramentos não foram urgentes.

4.2 Considerações Multiplataforma

Como a maioria das tecnologias usadas no arcabouço foram selecionadas pelo suporte à diferentes plataformas, relativamente poucas alterações são necessárias no código para a implementação dos componentes do arcabouço nessas plataformas. Diferente implementação dos métodos de Socket nos sistemas UNIX e Windows. Por exemplo o método *ioctl* deve ser alterado para (*ioctlsocket*) em sistemas diferentes de Linux/Android ou iOS, devido a configuração de (*timeouts*) na conexão Socket.

Outra diferença é o método *connect* de fechamento de conexão Socket, em específico a forma como o erro é gerenciado. Em plataformas UNIX é possível obter o erro através da variável *errno*, cujo valor deve ser comparado com valores predefinidos em *DEFINES* como EINPROGRESS, enquanto em

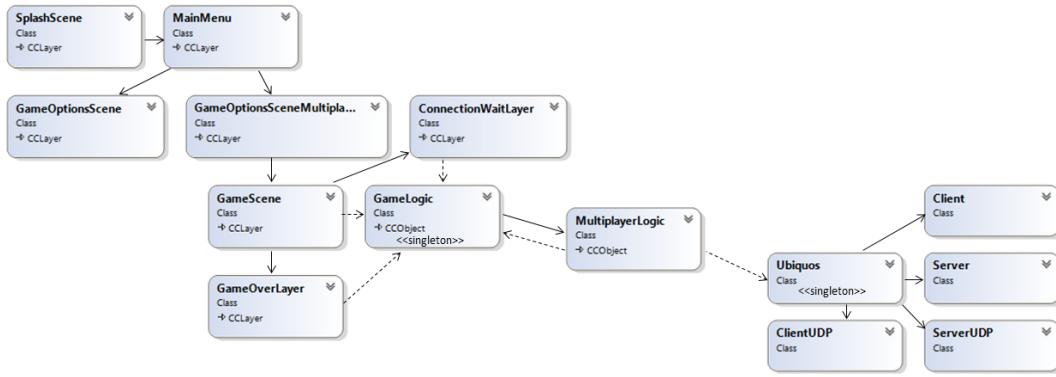


Figura 10: Diagrama de Classses Simplificados do Jogo Domino Gang desenvolvido no motor de jogos Cocos2d-x

plataformas Windows o erro é obtido através do método *WSAGetLastError* e comparado com o *DEFINE WSAWOULDBLOCK*.

Além disso, para enviamos a mensagem broadcast o arcabouço utiliza o IP do próprio dispositivo afim de encontrar o IP de Broadcast. Por exemplo, se o IP do dispositivo utilizado é 192.168.1.45 o arcabouço enviará a mensagem Broadcast no endereço 192.168.1.255. O método para obter o IP específico de cada dispositivo também diferente entre plataformas UNIX ou Windows. No último, o método *gethostbyname* é capaz de fornecer uma lista de todos as interfaces de rede do dispositivo. Por outro lado, em plataformas UNIX, o método *ioctl* com os argumentos *SIOCGIFADDR* e *ifr* podem ser utilizados para se obter o IP local, onde o *ifr* é uma estrutura do tipo *ifreq* e o componente *ifr_name* é "wlan0"em plataformas e "en0"dispositivos iOS.

5 RESULTADOS

Com o objetivo de avaliarmos o desempenho e a capacidade do arcabouço um jogo digital de Dominó foi desenvolvido (a interface dos jogadores pode ser vista na Figura 9 na figura 15). A implementação foi realizada utilizando um motor de jogos multiplataforma de código aberto e implementação em C++ chamado Cocos2D-x (HUSSAIN; GURUNG; JONES, 2014). O estilo de jogo foi escolhido por ser organizado em turnos, em que os jogadores se alternam para realizar suas ações. Consequentemente, a troca de mensagens de jogo pelo arcabouço ocorre em sequências específicas, auxiliando na sua análise e depuração. Além disso, dominó é um jogo que possui um ambiente naturalmente dividido duas áreas de foco: as peças da mão do jogador e as peças que já foram jogadas e estão na mesa.

A Figura 10 apresenta um diagrama de classes simplificado, criado para demonstrar as conexões entre as classes do jogo e aquelas do arcabouço. Como planejado, o jogo efetua chamadas para o arcabouço através de uma classe chamada *Ubiquos*, a qual é a implementação da classe *Singleton Administrador* da estrutura do arcabouço.

No desenvolvimento deste protótipo toda interação entre o jogo e a lógica de rede foi concentrada na classe *MultiplayerLogic*. Tal escolha permitiu que todo o processo de processamento das mensagens da rede fossem centralizadas e permitia que fossem feitas chamadas específicas para métodos do

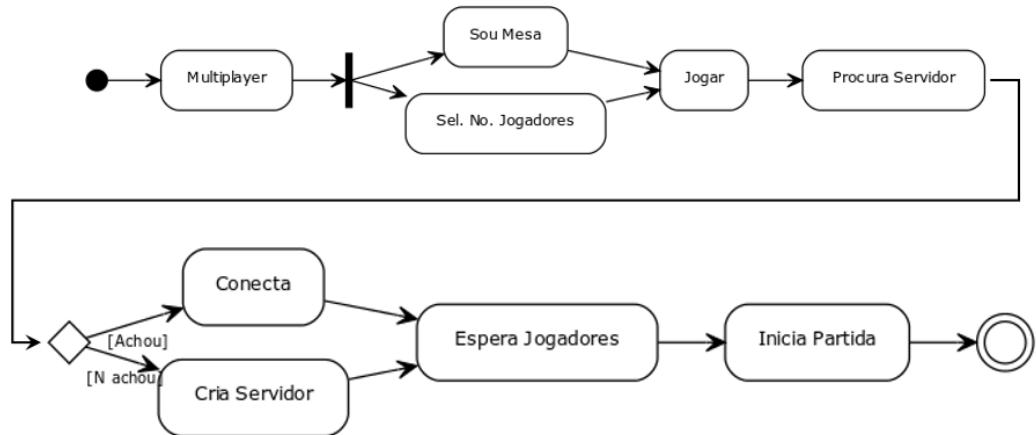


Figura 11: Diagrama de Atividades que leva o jogador a utilizar o arcabouço Ubiquos em uma partida com DUILs no protótipo.

jogo.

Deve ser ressaltado que por definição o arcabouço Ubiquos não possui nenhum tipo de implementação gráfica. Desta forma é necessário que todo o gerenciamento de telas para efetuar a conexão entre dispositivos seja desenvolvido pelo programador do jogo.

Assim, o protótipo possui uma opção de jogo em rede na qual o jogador utiliza a Ubiquos para encontrar outros jogadores em rede local e ter um jogo multi usuários. Uma sequência de eventos deve ser elaborada e implementada, segundo o diagrama descrito na Figura 11, para que um jogo utilizando o arcabouço seja iniciado.

Através da seleção da opção *Multiplayer* no menu principal do jogo, o jogador é direcionado para uma tela onde deve selecionar o número de jogadores e se aquele dispositivo agirá somente como "Mesa". Esta sequencia de telas ocorre de acordo com a Figura 12 e posteriormente a Figura 13.

Após a seleção das características desejadas de um jogo, o jogador é di-



Figura 12: Tela inicial do protótipo de jogo utilizado para o arcabouço Ubiqus, o botão a direita leva a tela de utilização da Interface Distribuída.

recionado para uma tela de espera, conforme Figura 14. Nesta tela em um primeiro momento é efetuada a busca por servidores que satisfaçam as definições determinadas e posteriormente se inicia uma espera próximos jogadores.

A "Mesa" indica um dispositivo que não interagirá no jogo, sendo na realidade apenas um espectador que demonstra todas as peças que são parte da visão conjunta dos jogadores, semelhantemente à mesa de um jogo de dominó. Esta opção não é necessária para se jogar o jogo no modo multijogador, que permite partida de até quatro jogadores mais a mesa.

A interface de usuário distribuída, no caso desse jogo, é organizada da seguinte forma: cada jogador, em seu dispositivo, visualiza e manipula o conjunto de peças definidas como dele (sua "mão"), enquanto a mesa permite conectar todos os jogadores demonstrando todas as peças já jogadas anteriormente.

Quando o protótipo foi testado em diversos dispositivos foi possível encontrar alguns problemas quando o servidor da partida era um dispositivo Android



Figura 13: Tela de configuração de modo de jogo para Ubiquos, com definições do número de dispositivos a se conectar e seu papel ("Mesa" ou jogador).

(ABLESON; COLLINS; SEN, 2009). Um dos problemas principais era a incapacidade de receber mensagens Broadcast, mesmo se o aplicativo estivesse corretamente configurado para fazê-lo de acordo com seu *AndroidManifest*.

A razão deste problema era enraizada em opções de alguns fabricantes de celulares Androids alterarem o Kernel do Linux para impedir tal função no sistema, focando em melhorar a bateria do dispositivo com tal ação. Infelizmente, esta limitação não é documentada e só foi possível classificar quais dispositivos possuíam tal limitação através de tentativa e erro.

Para facilitar reconhecer tal características utilizamos alguns aplicativos baixados diretamente da Google Play Store, a loja oficial de aplicativos para dispositivos Android. Os programas baixados podem ser utilizado para avaliar as capacidades Broadcast de um dispositivo, em nossa pesquisa utilizamos o programa *UDP Sender* (BENSON, 2014) para tal finalidade.

Apesar disso, foi possível distribuir o protótipo para múltiplos sistemas operacionais, dado que foi desenvolvido utilizando a Cocos2D-x (HUSSAIN; GURUNG; JONES, 2014), permitindo que partidas entre diversas plataformas, como

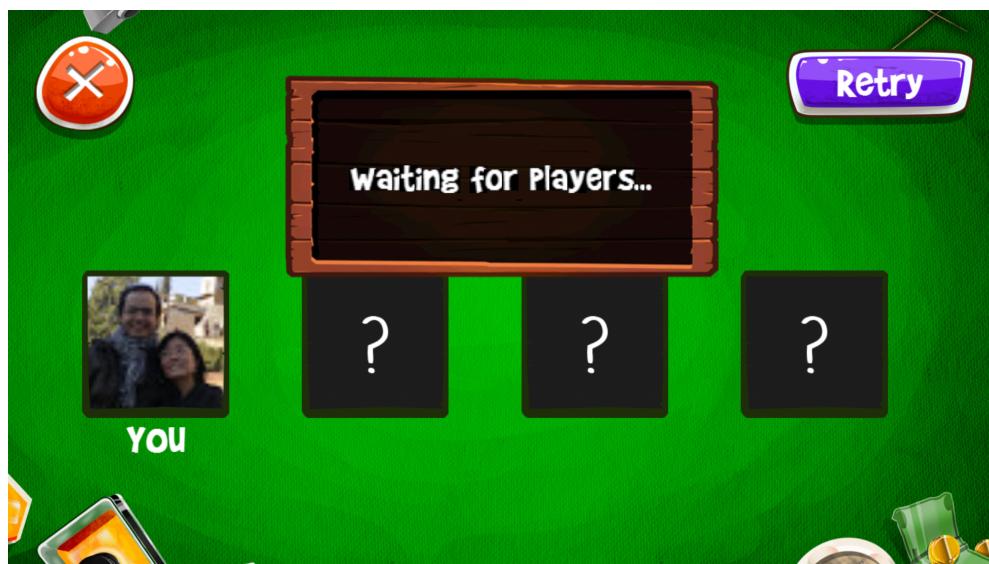


Figura 14: Tela de espera por jogadores esperando as comunicações automáticas do arcabouço Ubiquos conectar-se a outros dispositivos.

Tabela 2: Plataformas suportadas pelo protótipo e o arcabouço Ubiquos

Plataforma	Windows	WinPhone 8	Windows 8	Android	iOS
Jogabilidade do Protótipo	X	X	X	X	X
Funcionamento da Ubiquos	X	X	-	O	X

Android, iOS, Windows e Windows Phone, foram possíveis tais quais planejadas, assegurando as escolhas tomadas em estágios iniciais da pesquisa.

A Tabela 2 demonstra as plataformas suportadas pelo desenvolvimento do arcabouço e do protótipo.

Como podemos observar, o arcabouço não funcionou na plataforma Windows 8 devido a mudanças no sistema de implementação do Sockets que esta tem em relação ao Windows e Windows Phone, em específico o abandono do uso de WINSOCKS. Na plataforma Android deixamos um marcador O pelo fato das indisponibilidade de interpretar mensagens em Broadcast.

Um teste de usabilidade foi conduzido utilizando o protótipo. Para tal, foram convidados alguns sujeitos, no total de 10 pessoas, todos homens de idade entre 19 a 28, com escolaridade mínima de Grau Superior Incompleto.

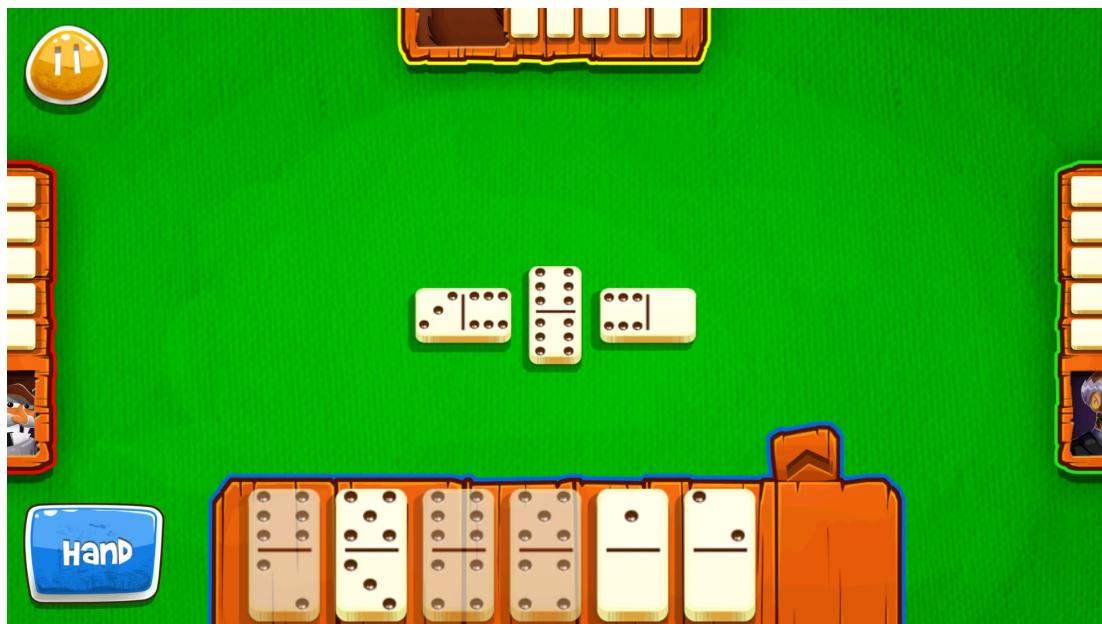


Figura 15: Protótipo: Jogo de dominó utilizando o arcabouço Ubiquos.

O procedimento para a avaliação era requerer dos sujeitos que estabelecessem uma conexão entre os dispositivos, através das interfaces do protótipo, e jogassem uma partida completa do jogo de domino.

O ambiente possuía uma rede sem-fio exclusiva, gerada através de um dispositivo Samsung S6 na função Ponto de Acesso Wi-Fi, o qual os sujeitos conectavam através de seus próprios dispositivos. O protótipo estava pré-instalado em seus dispositivos.

Como mesa um tablet Google Nexus 7 estava a disposição dos sujeitos, permitindo que no momento da conexão estes pudessem escolher utilizá-lo como uma interface extra. Tal opção foi recomendada pelo instrutor, no caso o autor deste projeto.

Após a partida os sujeitos eram encaminhados a salas isoladas e respondiam um formulário de avaliação do protótipo, Anexo 1. Nenhuma questão levantada pelos sujeitos foi respondida pelo instrutor, deixando a cargo a interpretação das questões pelos mesmos.

Tabela 3: Respostas ao Formulário de Avaliação do Protótipo

	Questão 1	Questão 2	Questão 3	Questão 4
Resp. 1	2	4	1	3
Resp. 2	4	4	2	4
Resp. 3	6	5	3	4
Resp. 4	5	5	6	4
Resp. 5	3	3	5	3
Resp. 6	5	4	3	5
Resp. 7	2	3	1	3
Resp. 8	3	5	1	4
Resp. 9	5	5	3	5
Resp. 10	5	4	3	6
Médias	4	4,2	2,8	4,1

O resultado de tais formulários pode ser observado na Tabela 3, onde os valores médios das respostas estão em destaque na linha inferior.

Vale notar que devido ao uso de dispositivos diferentes alguns sujeitos tiveram uma maior dificuldade no quesito conexão entre dispositivos. Outros tiveram a necessidade de efetuar a conexão para iniciar a partida mais de uma vez.

Durante as partidas nenhum dos jogadores informou uma experiência negativa em relação a fluidez do aplicativo. Animações que foram criadas para suavizar o jogo podem ter sido responsáveis por tal impressão positiva. Entretanto o tempo de latência da troca de mensagens, utilizando rótulos por exemplo, deve ser analisada para uma conclusão mais precisa.

De qualquer forma, acredita-se que em casos de variações de taxa de quadros ou alta latência as mensagens possam ter causado atrasos em um dispositivo específico ou até em falhas na conexão.

Na maioria das vezes os usuários não reconheceram tais atrasos, mas em um dos testes a mensagem não foi interpretada corretamente e acarretou em uma situação com falta de sincronia entre os dispositivos, levando a necessi-

dade de reiniciar o teste do inicio.

O sistema de busca de Servidores foi uma das funcionalidades que funcionou como planejado, procurando pela rede local através de mensagens Broadcast e finalmente criando um servidor após alguns segundos caso nenhuma resposta fosse recebida. Tal funcionalidade se apresentou coerente em diversos sistemas operacionais, tablets, smartphones e até Desktops.

Permitir que o arcabouço gerenciasse mensagens básicas e processasse o "HandShaking" permitiu uma desenvolvimento de conexões simplificado da perspectiva do desenvolvedor do jogo, removendo preocupações redundantes em diversos cenários e permitido-o focar nas mensagens relacionadas ao jogo.

6 CONCLUSÕES

O desenvolvimento de um protótipo utilizando a Ubiquos foi um ponto chave para percepção de necessidades e falhas no planejamento inicial. Foi apenas através do protótipo que a necessidade de saídas Esperadas e Não-Esperadas foram criadas.

O protótipo desenvolvido é responsável apenas por um gênero de jogo, baseado em turno. Deixando claro que próximos passos na pesquisa sejam do desenvolvimento de protótipos que utilizem passos mais curtos, assim como um jogo de ação, para assegurar o funcionamento do arcabouço sob estresse.

Neste protótipo de ação diversos novos problemas podem aparecer, dado a continua troca de mensagens, devido à atualização da posição especial de um personagem e a sincronização de todos personagens com o servidor. Tais processos poderiam eventualmente sobrecarregar um dos dispositivos.

O código referente ao arcabouço Ubiquos foi publicado na plataforma GitHub (GITHUB, 2007) através do link <https://github.com/pekayatt/ubiquos> de maneira pública. Acredita-se que os avanços desenvolvidos nesta pesquisa devem ser compartilhados com a comunidade e espera-se que novos avanços possam ser obtidos através desta disponibilização.

Smartphones estão começando a fazer parte de nosso cotidiano. Utilizá-los para integrar outros dispositivos já presentes no dia-a-dia, como televisões e Desktops, irá agregar mais valor às aplicações, tanto no âmbito pessoal

como em usos colaborativos.

A prévia existência de ampla infra-estrutura permitindo o acesso à Internet, atualmente utilizando conexões sem fio, tende a culminar em um mundo onde todos estão constantemente conectados. Trabalhos relacionados demonstram que diversas tecnologias focam em auxiliar conexões de dispositivos assim como de pessoas.

O lançamento de dispositivos comerciais como o Nintendo WiiU reforça a ideia que pessoas utilizarão dispositivos para jogos assimétricos. Com o aumento no poder de processamento de nossos dispositivos portáteis, é natural que o uso destes dispositivos superem aplicações específicas, como jogos, e permutem entre diversas aplicações do dia a dia.

Algumas iniciativas surgiram na forma de *Companion Apps* para televisões, tais quais estudados por (NANDAKUMAR; MURRAY, 2014), ou em plataformas de jogos específicas como nos estudos de (EMMERICH; LISZIO; MASUCH, 2014).

REFERÊNCIAS

- ABLESON, F.; COLLINS, C.; SEN, R. *Unlocking Android*. [S.I.]: Manning Greenwich, 2009.
- BALME, L.; DEMEURE, A.; BARRALON, N.; COUTAZ, J.; CALVARY, G. Cameleon-rt: A software architecture reference model for distributed, migratable, and plastic user interfaces. In: *Ambient intelligence*. [S.I.]: Springer, 2004. p. 291–302.
- BENSON, J. *UDP Sender*. 2014. Não publicado. Disponível em: <<http://github.com/hastarin/android-udpsender>>. Acesso em: 25 de agosto de 2016.
- BRAUN, H.; HOCEVAR, R.; QUEIROZ, R. B.; COHEN, M.; MOREIRA, J. L.; JACQUES, J.; BRAUN, A.; MUSSE, S. R.; SAMADANI, R. Vhcve: A collaborative virtual environment including facial animation and computer vision. In: IEEE. *Games and Digital Entertainment (SBGAMES), 2009 VIII Brazilian Symposium on*. [S.I.], 2009. p. 207–213.
- CAMPS-MUR, D.; GARCIA-SAAVEDRA, A.; SERRANO, P. Device-to-device communications with wi-fi direct: overview and experimentation. *Wireless Communications, IEEE*, IEEE, v. 20, n. 3, p. 96–104, 2013.
- CARD, S. K.; NEWELL, A.; MORAN, T. P. The psychology of human-computer interaction. L. Erlbaum Associates Inc., 1983.
- CLAYPOOL, M.; CLAYPOOL, K. Latency and player actions in online games. *Communications of the ACM*, ACM, v. 49, n. 11, p. 40–45, 2006.
- DEMEURE, A.; CALVARY, G.; SOTTET, J.-S.; VANDERDONKT, J. A reference model for distributed user interfaces. In: ACM. *Proceedings of the 4th international workshop on Task models and diagrams*. [S.I.], 2005. p. 79–86.
- ELMQVIST, N. Distributed user interfaces: State of the art. In: *Distributed User Interfaces*. [S.I.]: Springer, 2011. p. 1–12.
- EMMERICH, K.; LISZIO, S.; MASUCH, M. Defining second screen gaming: exploration of new design patterns. In: ACM. *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. [S.I.], 2014. p. 7.
- FAYAD, M.; SCHMIDT, D. C. Object-oriented application frameworks. *Communications of the ACM*, ACM, v. 40, n. 10, p. 32–38, 1997.

- FILIPPO, D.; RAPOSO, A.; ENDLER, M.; FUKS, H. Ambientes colaborativos de realidade virtual e aumentada. *C. Kirner, R. Siscoutto, EDS*, p. 168–191, 2007.
- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design patterns: elements of reusable object-oriented software*. [S.I.]: Pearson Education, 1994.
- GITHUB. Github is how people build software. 2007. Disponível em: <<https://github.com>>. Acesso em: 25 de agosto de 2016.
- GRAHAM, T.; URNES, T.; NEJABI, R. Efficient distributed implementation of semi-replicated synchronous groupware. In: ACM. *Proceedings of the 9th annual ACM symposium on User interface software and technology*. [S.I.], 1996. p. 1–10.
- HALL, B. B. Beej's guide to network programming: using internet sockets. 2001. Disponível em: <<http://beej.us/guide/bgnet/>>. Acesso em: 25 de agosto de 2016.
- HU, J. Storyml: Enabling distributed interfaces for interactive media. In: *WWW (Posters)*. [S.I.: s.n.], 2003.
- HUSSAIN, F.; GURUNG, A.; JONES, G. *Cocos2d-x Game Development Essentials*. [S.I.]: Packt Publishing Ltd, 2014.
- HUSTON, S. D.; JOHNSON, J. C.; SYYID, U. *The ACE programmer's guide: practical design patterns for network and systems programming*. [S.I.]: Addison-Wesley Professional, 2004.
- JAMIN, S. *Networking Multiplayer Games*. [S.I.]: Class Notes, Computer Game Design and Implementation Course, University of Michigan. URL: <http://ai.eecs.umich.edu/soar/Classes/494/talks/lecture-16.pdf>, visitado em setembro de, 2008.
- JOSUTTIS, N. M. *The C++ standard library: a tutorial and reference*. [S.I.]: Addison-Wesley, 2012.
- KARACHRISTOS, T.; APOSTOLATOS, D.; METAFAS, D. A real-time streaming games-on-demand system. In: ACM. *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*. [S.I.], 2008. p. 51–56.
- KIMBLER, K. App store strategies for service providers. In: IEEE. *Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on*. [S.I.], 2010. p. 1–5.
- KLOKMOSE, C. N.; BEAUDOUIN-LAFON, M. Vigo: instrumental interaction in multi-surface environments. In: ACM. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. [S.I.], 2009. p. 869–878.

- LEE, J.-S.; SU, Y.-W.; SHEN, C.-C. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In: IEEE. *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*. [S.I.], 2007. p. 46–51.
- LEE, J. W.; SCHULZIRINNE, H.; KELLERER, W.; DESPOTOVIC, Z. z2z: Discovering zeroconf services beyond local link. In: IEEE. *Globecom Workshops, 2007 IEEE*. [S.I.], 2007. p. 1–7.
- LEWIS, M.; JACOBSON, J. Game engines. *Communications of the ACM*, v. 45, n. 1, p. 27, 2002.
- MALONE, T. W. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In: ACM. *Proceedings of the 1982 conference on Human factors in computing systems*. [S.I.], 1982. p. 63–68.
- MEDEIROS FILHO, M.; CALADO, F.; NEVES, A. M. M. Jogabilidade assimétrica. In: *SBGames*. [S.I.: s.n.], 2013. p. 328–337.
- NANDAKUMAR, A.; MURRAY, J. Companion apps for long arc tv series: supporting new viewers in complex storyworlds with tightly synchronized context-sensitive annotations. In: ACM. *Proceedings of the 2014 ACM international conference on Interactive experiences for TV and online video*. [S.I.], 2014. p. 3–10.
- RUBIN, J.; CHISNELL, D. *Handbook of usability testing: how to plan, design and conduct effective tests*. [S.I.]: John Wiley & Sons, 2008.
- STEVENS, W. R.; FENNER, B.; RUDOFF, A. M. *The Sockets Networking API*. [S.I.]: Addison-Wesley, 2007.
- UNITY, U. G. E. Unity game engine-official site. 2009. Disponível em: <<https://unity3d.com>>. Acesso em: 25 de agosto de 2016.
- XU, K.; GERLA, M.; BAE, S. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks. In: IEEE. *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*. [S.I.], 2002. v. 1, p. 72–76.
- YANNAKAKIS, G. N.; HALLAM, J. Real-time game adaptation for optimizing player satisfaction. *Computational Intelligence and AI in Games, IEEE Transactions on*, IEEE, v. 1, n. 2, p. 121–133, 2009.

APÊNDICE A – FORMUÁRIO DE AVALIAÇÃO DO PROTÓTIPO DO ARACABOUÇO UBIQUOS

Q1: Quanto o jogo ter uma Interface Distribuída¹ foi importante?

Nada Pouco Médio Bastante Muito Demais

Q2: Você sentiu que a interação entre as interfaces foi fluída?

Nada Pouco Médio Bastante Muito Demais

Q3: A conexão entre os dispositivos foi complicada?

Nada Pouco Médio Bastante Muito Demais

Q4: O quanto você gostaria de ver outros jogos explorarem Interfaces Distribuídas?

Nada Pouco Médio Bastante Muito Demais

¹em diversas telas