Disciplinarum Scientia. Série: Naturais e Tecnológicas, Santa Maria, v. 14, n. 2, p. 245-259, 2013.
Recebido em: 27.09.2013. Aprovado em: 17.02.2014.
ISSN 2176-462X

APLICAÇÃO BASEADA EM *KINECT* COMO ALTERNATIVA PARA AMBIENTES DE COMPUTAÇÃO PERVASIVA[1]

APLICATION BASED ON KINECT AS AN ALTERNATIVE FOR PERVASIVE COMPUTING ENVIROMENTS

Cristiano Flores dos Santos[2] e Simone Regina Ceolin[3]

RESUMO

Neste trabalho, o objetivo foi desenvolver um *software* baseado em reconhecimento de comandos de gestos e de voz por meio do dispositivo *Kinect*, a fim de controlar dispositivos que compõem um ambiente residencial. Com este intuito, foi realizada a especificação e o desenvolvimento do *software* que possibilita reconhecer os comandos realizados pelos usuários, para simular ações como abrir e fechar portas e ligar e desligar lâmpadas. Para o desenvolvimento do estudo foi utilizada a metodologia *OpenUp* e a linguagem de programação C#. Também foram utilizados o SDK *kinect* e o SDK *speech* como ferramentats para o desenvolvimento da aplicação. Diante dos estudos realizados, foi desenvolvido um protótipo que apresentou resultados satisfatórios, no qual o usuário realiza interações em um ambiente residencial com a utilização do *kinect* como meio de interação.

Palavras-chave: interface humano-computador, acesso onipresente, sensor.

ABSTRACT

In this paper the objective is to develop a software applicative based in gesture and voice recognition commands through the Kinect device, in order to control the devices that compose a residential environment. With this propose, yhe specification and software development was performed, which allows to recognize the commands performed by users to simulate actions such as opening and closing doors and turning on and off the lights. For the development of the study, the OpenUp methodology and the C# programming language were used. The SDK Kinect and the SDK Speech software were used as tools for the application development. The prototype presented some satisfying results, through which the user performs interactions in a residential environment using Kinect as a means of interaction.

Keywords: *human-computer interface, ubiquitous access, sensor.*

[1] Trabalho Final de Graduação - TFG.

[2] Acadêmico do Curso de Sistemas de Informação - Centro Universitário Franciscano. E-mail: cristiano-santos@hotmail.com

[3] Orientadora - UFSM. E-mail: sceolin@redes.ufsm.br

INTRODUÇÃO

No cenário tecnológico, notam-se crescentes avanços nos sistemas de interação humano--computador, com o objetivo de torna-lo mais atraente e natural, beneficiando as mais diversas áreas do conhecimento e movimentando entusiastas do mundo todo (WEBB; ASHLEY, 2012). Um exemplo desse tipo de aplicação está na área da saúde, em que médicos podem manipular a distância imagens de exames digitais, girando e ampliando as imagens apenas com gestos.

A busca por uma interação mais próxima do natural entre homens e máquinas pode ser observada dentro do contexto da computação pervasiva, cuja essência estabelece ambientes nos quais as manipulações de determinados dispositivos interajam com o usuário, tornando os ambientes mais dinâmicos e os dispositivos cada vez mais invisíveis aos olhos do ser humano (BURKHARD, 2002).

Existem diversas propostas que tentam facilitar a interação humano-computador, possibilitando um modo de comunicação cada vez mais próximo da humana, tais como a identificação por comandos de voz e o reconhecimento de gestos, o que tem atraído muitas pessoas após a chegada do *Kinect* (PETRÓ, 2010). O *kinect* foi criado, originalmente, para ser um controle natural para videogames, mas aos poucos foi abrindo espaço em outros domínios de aplicação, tais como a utilização em cenários de computação pervasiva.

Dessa forma, este trabalho possibilitou o desenvolvimento de um *software* baseado em reconhecimento de comandos de gestos e voz por meio do dispositivo *Kinect*, como forma de interação humano-computador em um ambiente residencial simulado.

O artigo em questão, apresenta uma breve introdução ao conceito interface humano-computador e o conceito de computação pervasiva. Em seguida, apresenta o sensor *kinect* e adjacente, o kit de desenvolvimento (SDK) de software próprio para o sensor, e ainda um SDK para reconhecimento de voz. Segue também, a metodologia utilizada, os resultados obtidos e conclusão do estudo.

INTERFACE HUMANO-COMPUTADOR (IHC)

O computador está presente de forma muito intensa no dia a dia do ser humano nas mais variadas atividades. O surgimento de microcomputadores e novos sistemas operacionais tornaram-se poderosas ferramentas que auxiliam o ser humano nas atividades no ambiente de trabalho, bem como proporcionam lazer e conforto para o homem (HEEMANN, 1997).

Para PREECE (1994), os avanços tecnológicos da década de 70 possibilitaram estudos e novos meios de melhorar o relacionamento entre homem e o computador. Assim, surgiu então o termo Interface Humano-Computador (IHC) em meados dos anos 80. Nesse contexto, as empresas fabricantes de *software* perceberam um ambiente promissor e despertou-se o interesse em compreender a relação do homem com o computador.

A IHC estuda o indivíduo, a tecnologia computacional e como se influenciam (HEEMANN, 1997). De acordo com ROCHA; BARANAUSKAS (2003), a IHC trata do *design* de sistemas com-

putacionais que auxiliem as pessoas de forma a que possam executar suas atividades produtivamente e com segurança atuando no desenvolvimento de todo tipo de sistema, como jogos ou sistemas de escritório, nos quais acontece o envolvimento do usuário.

Nesse sentido, este trabalho busca desenvolver uma IHC, neste caso um ambiente residencial, que se proponha a interagir com o usuário de forma que atenda às suas necessidades, conforme sugere ROCHA; BARANAUSKAS (2003) deve interagir de forma mais natural possível.

COMPUTAÇÃO PERVASIVA

No atual cenário tecnológico, a computação está presente nos mais variados dispositivos utilizados diariamente. Além dos celulares, dispositivos portáteis como *tablets* e *notebooks*, percebe-se que a computação também aparece em outros ambientes como, por exemplo, em automóveis, diagnosticando ações realizadas como o esquecimento da porta do carro aberta, informando a quilometragem percorrida, o gasto de combustível em determinado trajeto, bem como alertando o motorista das revisões que são necessárias.

Neste contexto, WEISER (1993), define computação pervasiva como uma forma de computação invisível, em que dispositivos computacionais detectam alterações em um ambiente, com objetivo de atender as necessidades dos usuários, retornando assim respostas a uma determinada ação realizada pelo usuário. De acordo com HARIHAR; KURKOVSKY (2005), a computação pervasiva caracteriza-se como:

Acesso onipresente: consiste em um acesso natural ou transparente.

Sensível a contexto: considera a necessidade de um mecanismo de percepção do usuário pelo ambiente em que está imerso, oferecendo uma adaptação dinâmica e, eventualmente, uma nova reconfiguração desse ambiente.

Interação natural: refere-se à IHC de forma mais natural e próxima à linguagem do ser humano, seja por gestos ou comandos de voz.

Diante disso, de acordo com (KALLIO et al., 2006), para fornecer uma interação natural em um ambiente pervasivo é requerida a investigação de novas técnicas de interação IHC. Nesse sentido, foi possível promover a aplicação do dispositivo *Kinect* em ambientes pervasivos mediante gestos e comandos de voz.

SENSOR *KINECT*

O dispositivo *Kinect* foi desenvolvido pela empresa Microsoft, com objetivo de tornar a interação de jogos de vídeo games mais transparente e interativa, por meio de comando de gestos e voz, a fim de substituir os tão utilizados controles. O seu lançamento em novembro de 2010 tornou um dos produtos mais desejados do ramo (PETRÓ, 2010). Nesse contexto, a figura 1 apresenta o dispositivo *kinect.*

Figura 1 - Dispositivo *Kinect* (WEBB; ASHLEY, 2012).

Segundo a documentação presente na *help* do SDK *Kinect*, para que possa realizar o reconhecimento de comando de gestos e voz, são necessários três elementos básicos no aparelho, conforme mencionado na documentação oficial: câmera de vídeo VGA colorida, sensor de profundidade e microfone multi-matriz. Estes três elementos permitem ao usuário interagir com o dispositivo.

Além disso, quando o *Kinect* é ligado pela primeira vez, lê o ambiente e configura o espaço no qual o usuário estará se movendo. Depois pode detectar e rastrear até quarenta e oito pontos do corpo do usuário, mapeando-os para uma reprodução digital da forma e estrutura do esqueleto, incluindo detalhes de sua face.

Neste sentido, se desenvolve a ideia de aplicação deste estudo, pois o dispositivo *Kinect* é um produto inovador e revolucionário. Além disso, é um dispositivo de baixo custo e agregador de recursos facilitadores para desenvolvimento de aplicativos de interação natural.

SDK *KINECT*

O SDK é um kit de desenvolvimento de *software*, o qual oferece um grande número de recursos para possibilitar o desenvolvimento de novas soluções de IHC. O SDK do *kinect* fornece ferramentas, as quais os desenvolvedores podem criar soluções para detecção do mundo real com o objetivo de capturar imagens e sons de forma mais eficiente e eficaz.

Baseado nas funcionalidades que o SDK disponibiliza para o desenvolvimento de aplicativos, torna-se possível o desenvolvimento de novas aplicações para o dispositivo *kinect,* possibilitando que sejam criadas novas soluções para tornar a IHC mais natural possível. Além disso, o SDK proporciona as funcionalidades necessárias para operar com o dispositivo *kinect* de forma fácil e acessível.

SDK *SPEECH*

O Microsoft SDK *Speech é* um kit de desenvolvimento de *software* para mecanismos de fala, que reduz drasticamente a escrita de códigos necessária para desenvolver um aplicativo de

reconhecimento de voz, tornando a tecnologia de voz mais acessível e robusta para uma ampla gama de aplicações.

O Microsoft *Speech* fornece uma interface de alto nível para aplicação de mecanismo de fala, com todos os detalhes de baixo nível necessários para controlar e gerenciar as operações em tempo real já implementadas.

Dois tipos de motores de fala estão disponíveis, um para síntese de voz *(text-to-speech* ou TTS e um motor de reconhecimento (*Recognition Engine*). O Motor do tipo TTS converte palavras escritas em voz, enquanto o motor de reconhecimento converte frases faladas em escrita.

O funcionamento do SDK é baseado em eventos, ou seja, quando um comando é efetuado, a resposta ao reconhecimento se faz por meio de eventos ou mensagens de respostas.

Para o desenvolvimento do *software* proposto será utilizado o motor de reconhecimento de fala, portanto não será detalhado o motor de síntese de voz.

METODOLOGIA

Visando atender aos objetivos propostos neste trabalho, utilizou-se a metodologia *OpenUP* ou Processo Unificado Aberto. Essa metodologia consiste em uma versão do *Rational Unified Process* (RUP), o qual apresenta uma proposta de processo mais enxuto, completo e ao mesmo tempo extensível.

Além disso, a metodologia *OpenUp* está em conformidade com os princípios do manifesto do desenvolvimento de *software* ágil, ou seja, possui uma abordagem iterativa, incremental e é um processo que não está associado a nenhuma ferramenta em específico.

Neste sentido, a figura 2 busca ilustrar o conceito de iterativo e incremental, destacando a presença de três camadas distintas conforme mencionado por MEIRA (2010), as quais seguem descritas abaixo:

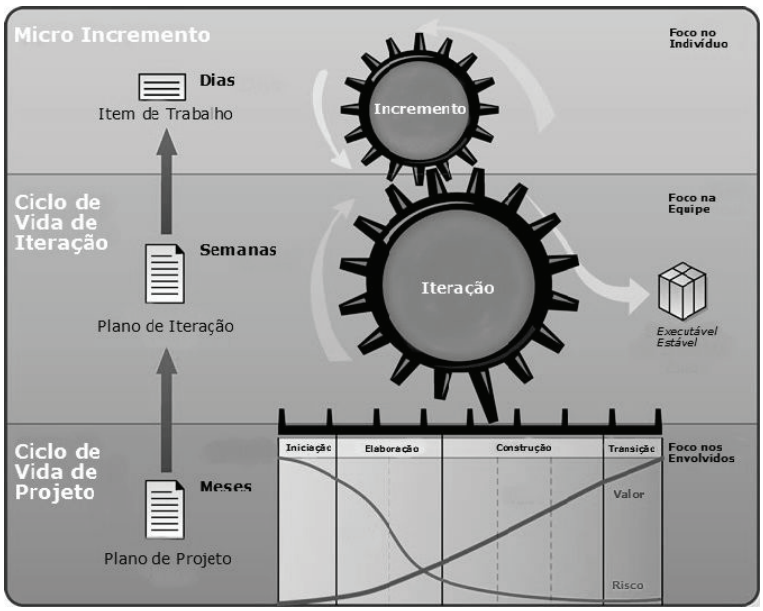

Figura 2 - Fases do método *OpenUP* (MEIRA, 2010).

1ª Camada - Ciclo de Vida de Projeto

Essa camada trata o ciclo de vida de projeto como um todoe pode ser dividido em quatro fases:

- **Fase de iniciação ou concepção:** cujo objetivo é delimitar o escopo do projeto, gerando como resultado final, um modelo de caso de uso de alto nível e um descritivo com visão geral dos requisitos, podendo ainda ser desenvolvidos diagramas de atividade;
- **Fase de elaboração:** tem como proposito, detalhar os requisitos arquiteturais, sendo assim, são gerados os descritivos de caso de uso, diagrama de sequência e diagrama de classes;
- **Fase de construção:** tem como objetivo esclarecer os requisitos e concluir o desenvolvimento do *software;*
- **Fase de transição:** como o nome sugere, é a transição do produto para a comunidade de usuários.

2ª Camada - Ciclo de Vida de Iteração

As fases mostradas na 1ª Camada, ou seja, na Camada Ciclo de vida de Projeto são quebradas em subatividades, chamadas de iteração, em que são tratadas as disciplinas: requisitos, arquitetura, implementação, teste e gerência de projetos.

3ª Camada - Ciclo de Vida de Microincremento

Microincrementos são subdivisões de uma iteração, que representa um esforço de algumas horas podendo ser alguns dias, desenvolvido por alguns membros da equipe de desenvolvimento, colaborando para atingir os objetivos da iteração.

Neste contexto, é interessante destacar que o *OpenUp* prevê alterações de seu conteúdo e seus métodos. Diante disso, visando tornar o projeto mais objetivo e enxuto, apenas alguns resultados de cada etapa foram aplicados.

RESULTADOS

Neste tópico descrevem-se os resultados obtidos no trabalho, conforme proposta apresentada para o estudo. Sendo assim, a partir das etapas sugeridas na metodologia *OpenUp*, foi possível detalhar as etapas envolvidas no processo de desenvolvimento do *software.*

Ciclo de Vida de Projeto-Iniciação

- Requisitos do sistema

Para que sejam reconhecidos os comandos de gesto, o software deve realizar a identificação do usuário, ou seja, o mapeamento dos membros do usuário. Já para o comando de voz, basta que o mesmo seja executado pelo usuário.

O sistema deve ser capaz de simular ações como abrir e fechar porta, ligar e desligar lâmpada, as quais são resultantes dos comandos realizados pelo usuário. Por meio de mensagens, o sistema informa ao usuário sobre o sucesso da interação realizada.

- Definição do escopo do sistema

A partir da análise dos requisitos, é possível deduzir as interações do sistema e suas funcionalidades, ou seja, o desenvolvimento do diagrama de caso de uso.

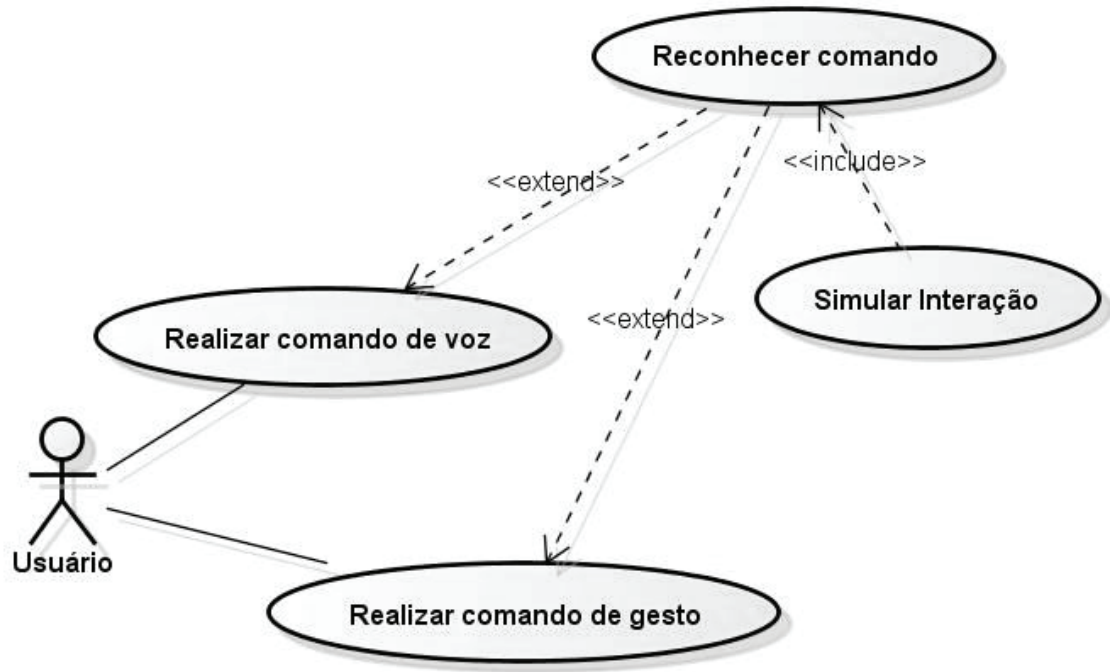

Figura 3 - Diagrama de caso de uso.

O diagrama de caso de uso da figura 3 esboça as ações que o usuário realiza. Diante dos requisitos levantados, a figura 4 apresenta o diagrama de atividades que representa as ações realizadas pelo usuário.

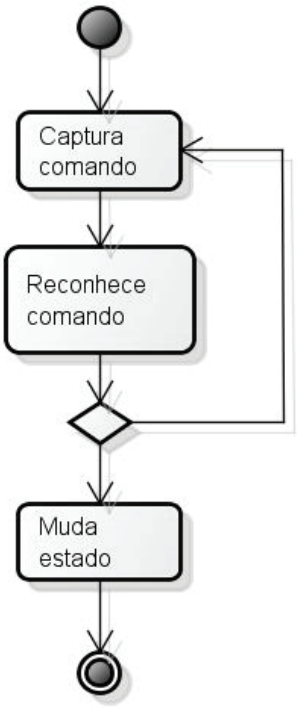

Figura 4 - Diagrama de atividades do sistema principal.

Ciclo de Vida de Projeto - Elaboração

Esta fase apresenta como resultado a descrição da arquitetura de *software* e protótipo de tela, bem como a conclusão dos requisitos funcionais e não funcionais por meio da descrição do diagrama de caso de uso.

- Definição da análise do sistema

Neste tópico, representado pelos quadros de número 1 ao quadro 5, são apresentados os descritivos do caso de uso com o objetivo de identificar os requisitos funcionais e não funcionais. No quadro 1, é apresentada regra para validação de "Reconhecer Comando", no quadro 2 o descritivo de caso de uso para "Reconhecer Comando", quadro 3 descritivo de "Simular Interação", no quadro 4 descritivos para "Realizar Comando de Voz" e por fim, no quadro 5 descrição para "Realizar Comando de Gesto".

Nome	**Regra de validação para reconhecer comando (RN01)**
Descrição	Para o reconhecimento de gesto, o usuário deve estar no campo de visão do sensor *kinect*. Já para o reconhecimento de comando de voz o usuário necessita simplesmente realizar o comando respeitando a semântica e sintaxe.

Quadro 1 - Descrição do caso de uso para validação "reconhecer comando".

Nº Identificador	**DCU01**
Nome	Reconhecer comando.
Descrição	Responsável por realizar o reconhecimento dos comandos.
Ator Principal	Usuário.
Fluxo principal	1. Usuário realiza comando; 2. Sistema compara com os comandos pré-estabelecidos; 3. Sistema reconhece comando; 4. Fim deste caso de uso;
Mensagens	M01 – Porta Aberta. M02 – Porta Fechada. M03– Lâmpada Ligada. M04–Lâmpada Desligada.

Quadro 2 - Descrição do caso de uso "reconhecer comando".

Nº Identificador	**DCU02**
Nome	Simular iteração.
Descrição	Responsável por mudar estado do dispositivo.
Ator Principal	Usuário.
Fluxo principal	1. Sistema realiza reconhecimento do comando; 2. Sistema apresenta como resultado a mudança de estado correspondente ao que foi programado pelo comando; 3. Fim deste caso de uso;
Fluxo alternativo	Enquanto comando não for reconhecido volta para o primeiro passo.

Quadro 3 - Descrição do caso de uso "simular interação".

Nº Identificador	**DCU03**
Nome	Realizar Comando de Voz.
Descrição	Permite que usuário realize comando.
Ator Principal	Usuário.
Fluxo principal	1. Sistema apresenta tela com estado dos dispositivos; 2. Usuário emite comando; 3. Verifica comando: se fechar ou abrir; 4. Fim deste caso de uso;

Quadro 4 - Descrição do caso de uso realizar "comando de voz".

Nº Identificador	**DCU04**
Nome	Realizar Comando de Gesto.
Descrição	Permite que usuário realize comando de gesto.
Ator Principal	Usuário.
Fluxo principal	1. Sistema apresenta tela com estado dos dispositivos; 2. Usuário realiza comando; 3. Fim deste caso de uso.

Quadro 5 - Descrição do caso de uso realizar "comando de gesto".

- Arquitetura do Sistema

A arquitetura do sistema representa o sistema de forma mais consistente e detalhada, possibilitando o real entendimento de como a aplicação se comporta. Assim, seguem as figuras 5 e 6 com diagramas de sequências do reconhecimento de gesto e de voz.

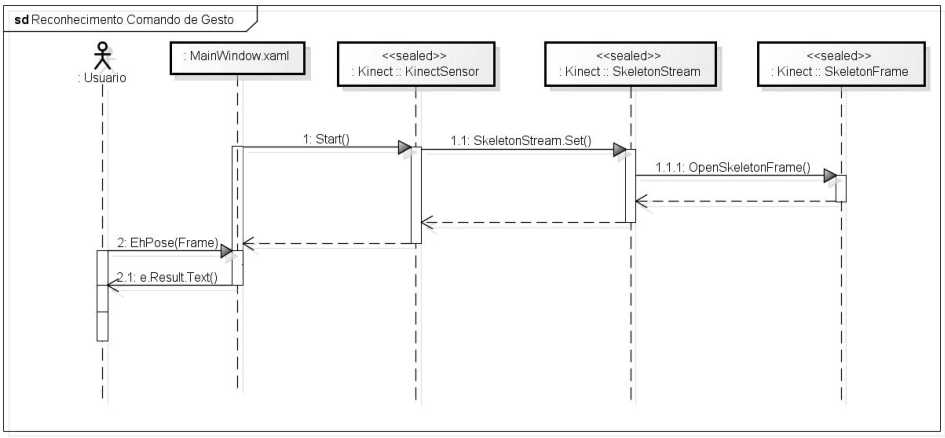

Figura 5 - Diagrama de Sequência Reconhecimento de Gestos.

A figura 6 apresenta o reconhecimento de comando de voz, ignorando classes mais superficiais e de menor importância.

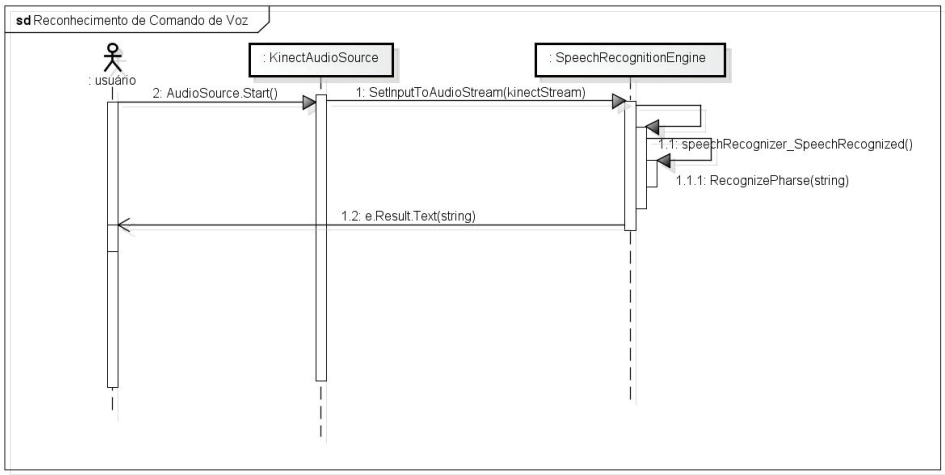

Figura 6 - Diagrama de Sequência Reconhecimento de comandos de voz.

A figura 6 apresenta o principal fluxo para reconhecimento de comandos de voz utilizando SDK *kinect* com o SDK *speech.*

Ciclo de Vida de Projeto - Construção

Nesta seção serão apresentados alguns trechos de códigos e telas do *software* desenvolvido. O desenvolvimento foi realizado em duas etapas, a primeira, em que foi tratado o reconhecimento de comando de gestos, e a segunda foi tratada o reconhecimento de voz.

1) Reconhecimento de Gestos

Para que ocorra o reconhecimento de gestos é necessário que o *software* reconheça as poses realizadas pelo usuário. Para tanto, o *software* possui pré-definidas as poses que serão reconhecidas. Dessa forma, a pose é detectável por meio de verificação das posições de articulações ou ângulos formados, sendo calculado o ângulo entre elas.

Sendo assim, foi criada uma biblioteca de poses, com o intuito de armazenar ângulos formados por determinados pontos. Além disso, também foi criada uma classe chamada "PoseAngulo" e uma estrutura chamada "pose". A estrutura "pose" possui uma matriz de objetos "PoseAngulo".

O método "*IsPose"* chama o método "*GetJointAngle"* a fim de calcular o ângulo entre duas articulações, e este chamada o método "*GetJoinPoint"* para obter as coordenadas dos pontos de cada articulação. Posterior a isso, o método implementa a lei dos cossenos, com o objetivo de obter o ângulo formado pelas articulações.

Por sua vez, o método *"Acos"* retorna valores em radianos, tornando-se necessária a conversão do valor do ângulo em graus. No final obtêm-se ângulos entre 180º-360º graus. A lei dos cossenos só é valida para ângulo entre 0º e 180º. É necessário ajustar os valores para os ângulos que caem no terceiro e quarto quadrante do gráfico. O método "*Ispose"* retorna como resultado verdadeiro se a pose foi identificada.

Finalizando o reconhecimento de gesto, o método "ProcessaResposta", valida a pose atual, implementando uma parada no cronômetro e apresentando uma mensagem positiva ao reconhecimento de gesto.

```
private bool IsPose(Skeleton skeleton, Pose pose)
{
    bool isPose = true;
    double angle;
    double poseAngle;
    double poseThreshold;
    double loAngle;
    double hiAngle;

    for (int i = 0; i < pose.Angles.Length && isPose; i++)
    {
        poseAngle = pose.Angles[i].Angle;
        poseThreshold = pose.Angles[i].Threshold;
        angle = GetJointAngle(skeleton.Joints[pose.Angles[i].CenterJoint], skeleton.Joints[pose.Angles[i].AngleJoint]);

        hiAngle = poseAngle + poseThreshold;
        loAngle = poseAngle - poseThreshold;

        if (hiAngle >= 360 || loAngle < 0)
        {
            loAngle = (loAngle < 0) ? 360 + loAngle : loAngle;
            hiAngle = hiAngle % 360;

            isPose = !(loAngle > angle && angle > hiAngle);
        }
        else
        {
            isPose = (loAngle <= angle && hiAngle >= angle);
        }
    }

    return isPose;
}
```

Figura 7 - Trecho de código do método IsPose.

2) Reconhecimento de comando de voz

Para o reconhecimento de comandos por voz, se faz necessário o uso do *SDK Speech,* a partir da classe "*SpeechRecognitionEngine".* Por meio dessa classe, é possível submeter um fluxo de áudio do sensor *kinect* e realizar uma análise e interpretação de expressões vocais.

Esse processo acontece por meio de construções de chamadas gramaticais, ou seja, com objeto "*Grammar"* pode ser construída uma sequência de palavras que serão reconhecidas via *software*. Além disso, é possível adicionar a funcionalidade do objeto "*Choices"* que possibilita a escolha de vários valores ao objeto, para que, posteriormente por meio da classe "*GrammarBuilder"* seja adicionada semântica aos comandos que precisam ser reconhecidos.

Na figura 8 verifica-se que, o objeto vocabulário é uma instância da classe "*Choices",* onde há um laço que percorre uma variável "Vocabulario" que inicialmente recebe instruções: "*Open*" e "*Close*" e, posteriormente são enviadas por parâmetro para o método "*add"*. Por fim, a classe "*GrammarBuilder"* é instanciada, e logo o método "*Append"* recebe por parâmetro os valores dos comandos.

```
var vocabulario = new Choices();
foreach (string value in Vocabulario.Keys)
    vocabulario.Add(value);

var gb = new GrammarBuilder();
//
gb.Culture = ri.Culture;

gb.Append(vocabulario);

//Cria a gramatica atual e carrega o inicialização do Speech Recognizer
var g = new Grammar(gb);

speechRecognizer.LoadGrammar(g);
```

Figura 8 - Trecho de código de reconhecimento de voz.

A tela da aplicação tem como objetivo apresentar em um único lugar o estado dos dispositivos de um ambiente residencial que estão sendo simulados, neste caso a lâmpada e a porta. No centro da tela é projetada em forma de animação a imagem do usuário interagindo com o sistema. Assim, ao levantar o braço direito, o painel esquerdo é atualizado mudando o estado da lâmpada, confirmando, dessa forma, o reconhecimento da interação realizada. Da mesma forma, ao emitir o comando "*open*" via comando de voz, o painel da esquerda atualiza o estado da porta para aberto e comando "*close*" para fechada. O reconhecimento de gestos muda o estado de desligado para ligado e vice-versa.

A figura 9 representa a tela do sistema na qual o usuário está interagindo com a aplicação desenvolvida no estudo proposto:

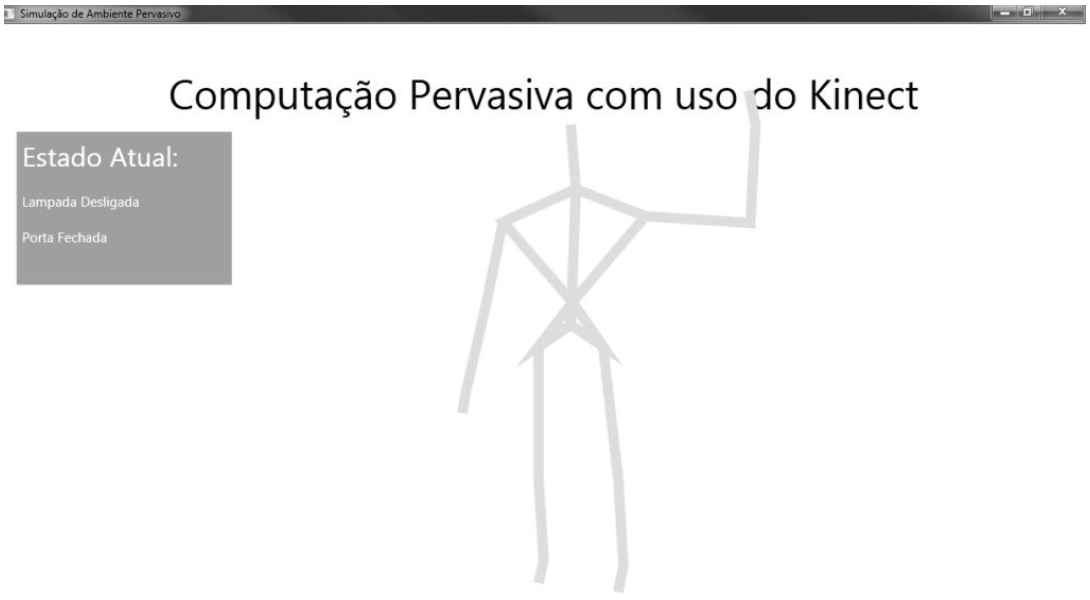

Figura 9 - Tela da aplicação.

CONCLUSÃO

A partir dos estudos realizados neste trabalho, verificam-se avanços tecnológicos em busca de sistemas e dispositivos que tornem a IHC mais atraente e natural. Fato que pode ser observado no contexto da computação pervasiva, que procura tornar os ambientes dinâmicos e os dispositivos invisíveis aos olhos do ser humano, conforme menciona (BURKHARD, 2002).

Para que possa ocorrer a interação entre o usuário e o sistema, deve-se proporcionar ao usuário uma interface amigável e, principalmente que atenda as necessidades do usuário, retornando as respostas de acordo com a ação realizada pelo mesmo. Um exemplo disso, é o dispositivo *Kinect* que foi desenvolvido pela empresa Microsoft, com objetivo de tornar a interação de jogos de vídeo games mais transparente e interativa, por meio de comando de gestos e voz, a fim de substituir os controles.

Neste sentido, o presente estudo teve como objetivo desenvolver um *software* baseado em reconhecimento de comandos de gestos e de voz por meio do dispositivo *Kinect*, como forma de controle para dispositivos que compõem um ambiente residencial. Para tanto utilizou-se a metodologia *OpenUp*, por apresentar melhor flexibilidade no desenvolvimento do estudo proposto.

Diante dos estudos realizados, foi desenvolvido um protótipo de interface no qual o usuário realiza interações em um ambiente residencial simulado com a utilização do *kinect* como meio de interação. As interações ocorrem por meio de comandos de gestos e voz, que ao serem reconhecidas pelo *software*, o mesmo atualiza a tela da aplicação apresentando como resultado da interação a mudança de estado do objeto.

Cabe destacar que, a aplicação desenvolvida é um protótipo para um desenvolvimento futuro onde possa ser aplicado em um caso real, com mais funcionalidades. Salienta-se que, no estudo em questão o protótipo apresentou bons resultados nos testes realizados, e os objetivos inicialmente propostas foram cumpridos.

Por fim, sugere-se para trabalhos futuros, a aplicação do protótipo em uma residência e não apenas em ambiente simulado. Além disso, sugere-se o desenvolvimento de novas funcionalidades como cadastro de novos comandos de gesto e voz.

REFERÊNCIAS

BURKHARD, T. J. et al. **Pervasive Computing:** Technology and Architecture of Mobile Internet Applications. Boston, MA, USA: Addison-Wesley, 2002.

HARIHAR, K.; KURKOVSKY, S. Using Jini to enable pervasive computing environments. In: ACM-SE 43: PROCEEDINGS OF THE 43RD ANNUAL SOUTHEAST REGIONAL CONFERENCE, 2005. Kennesaw, Georgia, New York, USA. **Anais...**, ACM, p. 188-193, 2005.

HEEMANN, V. **Avaliação Ergonômica de Interfaces de Bases de Dados por Meio de Checklist Especializado.** Dissertação (Mestrado em Engenharia de Produção) Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina - LablUtil. 1997. Disponível em: <http://www.eps.ufsc.br/disserta97/heemann/>. Acesso em: ago. 2011.

KALLIO S. et al. Visualization of hand gestures for pervasive computing environments. In: AVI '06: PROCEEDINGS OF THE WORKING CONFERENCE ON ADVANCED VISUAL INTERFACES, 2002. Venezia, Italy. **Anais...** New York, USA: ACM, p. 480-483, 2006.

MEIRA, F. L. **Introdução ao Processo Unificado Aberto.** 2010. Disponível em: <http://open2up.blogspot.com.br/>. Acesso em: 29 ago. 2012.

PETRÓ, G. **Sistema *Kinect*, que usa corpo como joystick, chega ao Brasil.** 2010. Disponível em: <http://glo.bo/1yk7gY6>. Acesso em: ago. 2011.

PREECE, J. **Human-Computer Interation.** New Jersey, USA: Addison-Wesley Publishing Company, 1994.

ROCHA, H. V., BARANAUSKAS, M. C. C. **Design e avaliação de interfaces humano - computador**. Campinas: NIED/UNICAMP, 2003.

WEISER, M. Hot Topics: Ubiquitous Computing. **IEEE Computer**, v. 6, n. 10, p.71-72, 1993.

WEBB, J.; ASHLEY, J. **Beginning Kinect Programming With The Microsoft Kinect SDK**. New York: Apress, 2012.