UNIVERSIDADE DO SAGRADO CORAÇÃO

PAULO HENRIQUE PALMEIRA FRANCO

APLICATIVO MÓVEL PARA AUXÍLIO NO TRATAMENTO E RECUPERAÇÃO DE CRIANÇAS COM DEFICIÊNCIA MOTORA NOS MEMBROS SUPERIORES

BAURU
2016

PAULO HENRIQUE PALMEIRA FRANCO

APLICATIVO MÓVEL PARA AUXÍLIO NO TRATAMENTO E RECUPERAÇÃO DE CRIANÇAS COM DEFICIÊNCIA MOTORA NOS MEMBROS SUPERIORES

Trabalho de conclusão de curso apresentado ao centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

BAURU
2016

PAULO HENRIQUE PALMEIRA FRANCO

Aplicativo móvel para auxílio no tratamento e recuperação de crianças deficiência motora nos membros superiores

Trabalho de conclusão de curso apresentado ao centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

Banca examinadora:

Prof. Dr. Elvio Gilberto da Silva

Universidade Sagrado Coração

Prof. Me. Patrick Pedreira

Universidade do Sagrado Coração

Prof. Me. Marcio Cardim

Universidade do Sagrado Coração

Bauru, 26 de novembro de 2016.

AGRADECIMENTOS

Sou grato a Deus acima de tudo por me abençoar com a oportunidade de estudar, e a Ele dedico todo meu esforço.

Agradeço aos meus pais e amigos por todo o apoio dado durante toda essa vida acadêmica e em especial ao meu melhor amigo e irmão Felipe Franco por todas as companhias e incentivos durante todos esses anos.

Agradeço a todos os meus professores, mas em especial ao meu orientador Prof. Dr. Elvio Gilberto da Silva por todo o incentivo e correções seja durante a realização desta pesquisa ou em sala de aula.

Aos profissionais e atuantes das áreas de terapia ocupacional e informática, todo o meu respeito e dedicação.

RESUMO

A Clínica de Terapia Ocupacional da Universidade do Sagrado Coração (USC) situada em Bauru/SP configura-se como um importante serviço no tratamento de pacientes com disfunções físicas, sensoriais e psicossociais. A Clínica de Terapia Ocupacional da USC tem desenvolvido diversos trabalhos e pesquisas com intuito de oferecer aos seus pacientes tratamentos que visam melhorar o seu desempenho ocupacional e cotidiano. Diversas pesquisas são elaboradas com os pacientes atendidos, bem como, com seus familiares, com o objetivo de proporcionar uma melhor qualidade de vida para os mesmos durante o processo de reabilitação. Atualmente a tecnologia da informática vem se ampliando para diversas áreas de atuação profissional, inclusive para o campo da Terapia Ocupacional, aonde vem sendo aplicada na prática clínica como modalidade de tratamento. A proposta deste trabalho consiste no desenvolvimento de um software interativo para dispositivos moveis visando auxiliar no tratamento e recuperação de crianças que apresentem deficiência motora nos membros superiores. Para o desenvolvimento dessa proposta foi utilizada linguagem de programação LUA. Este projeto tem a intenção de contribuir como uma proposta multidisciplinar, na qual contará com a informática como ferramenta de apoio na prática terapêutica ocupacional, promovendo assim, uma maior interação entre familiares e profissionais da área, e consequentemente, possibilitando a melhora dos resultados apresentados nos tratamentos.

Palavras-chave: Tecnologia. Software. Deficiência. Criança. Desenvolvimento.

ABSTRACT

The Occupational Therapy Clinic of the Universidade do Sagrado Coração (USC) located in Bauru/SP is an important service in the treatment of patients with physical, sensory and psychosocial disorders. The USC Occupational Therapy Clinic has developed a variety of research and studies to provide patients with treatments that aim to improve their occupational and daily performance. Several researches are done with patients cared for, as well as with their families, with the aim of providing a better quality of life for them during the rehabilitation process. Currently, computer technology has been expanding into several areas of professional practice, including in the field of Occupational Therapy, where it has been applied in clinical practice as a treatment modality. The purpose of this research is the development of an interactive software for mobile devices to assist in the treatment and recovery of children with motor deficits in the upper limbs. For the development of this proposal was used LUA programming language. This project intends to contribute as a multidisciplinary proposal, in which it will count on the computer science as a tool of support in the occupational therapeutically practice, thus promoting, a greater interaction between family and professionals of the area, and consequently, enabling the improvement of the results presented in the Treatments.

Keywords: Technology. Software. Deficiency. Child. Development.

LISTA DE ABREVIATURAS E SIGLAS

AAPT	Android Asset Packaging Tool
AVL	Atividade de Vida de Lazer
AVP	Atividades da Vida Pratica
AVT	Atividade da Vida do Trabalho
DVM	Dalvik Virtual Machine
IDC	International Data Corporation
OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Object-oriented software engeneering
PC	Paralisia Cerebral
SGBD	Sistema Gerenciador de Banco de Dados
SNC	Sistema Nervoso Central
SQL	Structured Query Language
UCP	Unidade Central de Processamento
UML	Unified Modeling Language
USC	Universidade do Sagrado Coração
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
Tecgraf	Grupo de Tecnologia em Computação Gráfica

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A tecnologia da informática vem se desenvolvendo cada vez mais, e nos dias de hoje, o computador, que foi criado inicialmente com o intuito de realizar apenas operações numéricas, apresenta novas e diversas funções dentro de sua enorme complexidade, dentre elas, funções muito importantes como a de veículo de comunicação, aprendizagem, recurso terapêutico no processo de reabilitação, entre outras.

Os avanços no campo da informática vêm ocorrendo tanto no que diz respeito aos aspectos relacionados ao hardware, que consiste na parte física do computador, peças, encaixes, fios. Quanto aos softwares, que constituem os programas, algo que determina o comportamento e a função a ser executada. Sendo assim, podem–se notar esses avanços na existência e criação de softwares cada vez mais especializados, na criação de diversos tipos de acionadores, teclados e mouses, e também no aumento da possibilidade de oportunidades diversas proporcionadas pelo uso da internet.

Permitiu-se assim que a tecnologia da informática pudesse ampliar-se para diversas áreas de atuação profissional, inclusive para o campo da Terapia Ocupacional, onde vem sendo aplicada, tanto nas áreas práticas de administração quanto na área clínica.

Muitas pesquisas são realizadas sobre a dificuldade de pessoas que possuem deficiência motora têm para realizar tarefas do dia a dia, hoje em dia é difícil encontrar um aplicativo móvel que atenda às necessidades de crianças que possuem deficiência motora e que seja de fácil uso. A área de Terapia Ocupacional da USC realiza os estudos com crianças buscando sempre novas formas de melhorar e diversificar o atendimento realizado com essas crianças, para que o desenvolvimento seja o melhor e o tratamento menos maçante para o paciente.

A tecnologia tem avançado de diversas maneiras, proporcionando melhorias em muitas áreas, principalmente na área da saúde, sempre visando custos baixos para que se possa atender a maior totalidade de pessoas. Na computação não é diferente, a área sofre mudanças todos os dias e muitas delas não conseguem ser acompanhadas por pessoas que possuem deficiência motora, pelo fato de que essas mudanças demoram algum tempo para serem adaptadas a essas pessoas.

O computador também pode ser utilizado como recurso para auxiliar no tratamento e recuperação da deficiência motora, socialização, meio de comunicação ou uma forma de lazer.

2 OBJETIVOS

Apresenta-se nos tópicos a seguir o objetivo geral e os objetivos específicos da pesquisa.

2.1 OBJETIVO GERAL

Desenvolver um aplicativo interativo para dispositivos móveis, com a finalidade de auxiliar no tratamento das deficiências motoras dos membros superiores de crianças que possuem paralisia cerebral.

2.2 OBJETIVOS ESPECÍFICOS

Entre os objetivos específicos, destacam-se:

a) Realizar um estudo sobre as dificuldades apresentadas por crianças portadoras da paralisia cerebral.
b) Estudar o ambiente de tratamento das crianças portadoras de paralisia cerebral, entendendo suas limitações e as reais necessidades, podendo aplicar conhecimentos funcionais no desenvolvimento desta proposta.
c) Fazer uma bibliografia e aprofundar o conhecimento em arquitetura de software, SQLite e linguagem de programação Java para dispositivos móveis.
d) Planejar o aplicativo e fazer a sua modelagem utilizando UML.
e) Desenvolver o aplicativo proposto na linguagem de programação LUA.
f) Modelar e projetar níveis de dificuldade para o aplicativo proposto.

3 REVISÃO DE LITERATURA

Apresenta-se a seguir os tópicos utilizados na revisão de literatura na elaboração da pesquisa.

3.1 PARALISIA CEREBRAL

Paralisia Cerebral (PC) é um termo geral que engloba manifestações clínicas muito variadas e que tem em comum a dificuldade motora em consequência a uma lesão cerebral.

Para que uma criança com dificuldade motora tenha o diagnóstico de Paralisia Cerebral, é necessário que a lesão neurológica tenha acontecido durante a fase de desenvolvimento do Sistema Nervoso Central (SNC), que vai desde o momento da concepção até os 2 (dois) anos de idade, e que a lesão neurológica não seja uma lesão progressiva. Embora a criança apresente mudança no seu crescimento e amadurecimento, a lesão é estacionária, ou seja, não vai piorar e nem desaparecer.

Segundo Leitão (1983), a Paralisia Cerebral é causada por qualquer agressão ao cérebro em desenvolvimento, antes dos 3 anos de idade. Para facilitar a melhor determinação das causas, dividem-se em 3 tipos:

1. **Pré-natais**: as lesões que ocorrem antes do nascimento. Algumas doenças da gestante podem comprometer a formação das estruturas neurológicas do feto dentro do útero, como a diabete, a pressão alta, a rubéola e a toxoplasmose, além do uso de álcool e drogas.
2. **Perinatais**: são as lesões neurológicas que acontecem no período que vai do começo do trabalho de parto até 6 horas após o nascimento (ex. prematuridade, baixo peso, trabalho de parto demorado, etc.).
3. **Pós-natais**: As lesões neurológicas ocorrem durante a infância, até 2 anos de idade, quando o sistema nervoso ainda está em desenvolvimento, decorrente de infecções como a meningite, traumas cranianos, e tumores.

Os sintomas ou manifestações clínicas da Paralisia Cerebral variam de criança para criança, pois depende da gravidade, da extensão da lesão e da área

neurológica comprometida, mas as alterações motoras estão sempre presentes na pessoa com Paralisia Cerebral.

Fonseca e Lima (2008) descrevem a Paralisia Cerebral como uma encefalopatia crônica infantil que se caracteriza por distúrbios motores, de caráter não progressivo, que se manifestam em um cérebro em desenvolvimento, levando a Distúrbios Motores, tônus e postura.

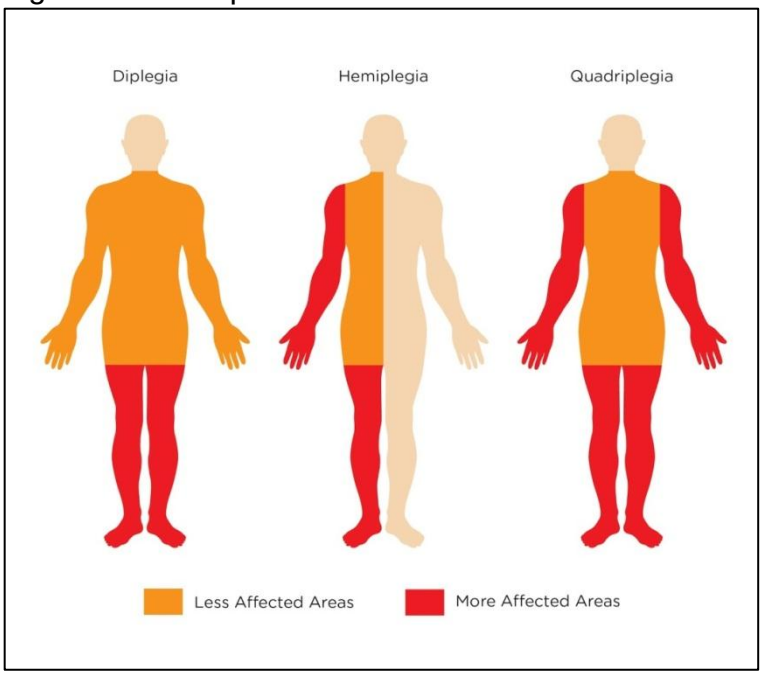
Distúrbio motor é caracterizado pela incapacidade de movimentar braços e pernas, dificuldade para sentar, engatinhar, andar e se equilibrar; mastigar, engolir, tossir, etc., funções comandadas pela área afetada do Sistema Nervoso Central.

Além do distúrbio motor, outros transtornos neurológicos que podem estar presentes são: crises convulsivas, dificuldades visuais, dificuldade na fala, deficiência auditiva, dificuldade para engolir, deficiência mental, entre outras.

Fonseca e Lima (2008), referem que, dependendo da distribuição do comprometimento motor, classifica-se a paralisia cerebral em 3 tipos:

1. **Quadriplegia**: comprometimento em que tanto os membros superiores (braços) quantos os inferiores (pernas) estão alterados com a mesma gravidade.
2. **Diplegia**: o comprometimento é mais acentuado nos membros dos inferiores (pernas) do que nos superiores (braços).
3. **Hemiplegia**: é o comprometimento do membro superior e inferior de um dos lados do corpo, direito ou esquerdo, dependendo do hemisfério do cérebro que foi lesionado.

Figura 1 - Exemplos de distúrbios motores



Fonte: CPL – Choice, Passion, Life.

Os principais sinais motores nas pessoas com a Paralisia Cerebral são: Alterações de movimento, tais como:

a) Ausência de movimentos irrequietos;
b) Movimentos circulares de braços;
c) Movimentos assimétricos dos membros;
d) Ausência de movimento das pernas;
e) Movimentos de lateralização bilateral da cabeça repetitivos ou monótonos;
f) Movimentos repetidos de abertura e fechamento da boca,

Alterações da postura:

a) Incapacidade de manter a cabeça em linha média;
b) Postura corporal assimétrica;
c) Tronco e membros largados sobre o leito;
d) Braços e pernas em extensão;
e) Hiperextensão de tronco e pescoço;
f) Punho cerrado;
g) Abertura e fechamento sincronizado dos dedos;

Bobath (1979) cita que na Paralisia Cerebral a lesão do Sistema Nervoso Central interfere na sequência do desenvolvimento e que os sintomas do retardo motor são seguidos pelo aparecimento de padrões anormais de postura e movimento, associados ao tônus postural anormal. Refere ainda que a evolução e a extensão com que este quadro se desenvolve depende do tipo e da severidade do caso, de quanto do corpo está envolvido.

O diagnóstico da Paralisia Cerebral é feito através de informações recebidas da família (história ou anamnese) e da avaliação do paciente (exame físico).

A Paralisia Cerebral não tem cura, pois é consequência de uma lesão irreversível do Sistema Nervoso Central.

Todo tratamento tem por objetivo permitir que cada criança desenvolva seu maior potencial em todos os aspectos de sua vida.

Para se atingir esse objetivo, dois aspectos muito importantes devem ser levados em consideração:

1. **Prognóstico**: esse é o termo médico para definir o potencial de cada paciente. O prognóstico é muito variável de paciente para paciente e depende de inúmeros fatores como o grau de comprometimento motor, a presença de alterações associadas, o aparecimento de complicações e também da colaboração da família e do próprio paciente.
2. **Equipe Multidisciplinar**: O indivíduo com Paralisia Cerebral pode ter alterações em outras áreas além da motora, como por exemplo, linguagem e compreensão, entre outras, sendo necessário o envolvimento de profissionais de diferentes áreas da saúde, como Médico, Fisioterapeuta, Fonoaudiólogo, Terapeuta Ocupacional, Psicólogo, Pedagogo, etc. Toda essa equipe que trabalha com o Indivíduo é chamada de Equipe Multidisciplinar.

3.1.1 O computador como recurso terapêutico

Atualmente o computador e suas funções vêm sendo aplicados em Terapia Ocupacional enquanto um recurso terapêutico, que desempenha importante papel no processo dos pacientes que se encontram em programa de reabilitação.

Através de seu uso, o computador pode ser trabalhado em diversos aspectos na Terapia Ocupacional, como: aspectos motores, cognitivos, Atividades da Vida Prática (AVPs), Atividades da Vida do Trabalho (AVTs), Atividades de Vida de Lazer (AVLs), caracterizando-se como uma ação complementar aos atendimentos usuais no Serviço de Terapia Ocupacional, repercutindo positivamente no processo de reabilitação de modo geral.

Segundo o dicionário Aurélio, a palavra recurso é definida como: auxílio, ajuda, meio para resolver um problema.

Na prática da Terapia Ocupacional utilizamos o termo “recurso terapêutico” para designar todo e qualquer dispositivo que vise a aquisição ou ampliação de autonomia e independência de um indivíduo em suas ações do cotidiano.

Partindo do pressuposto de que o principal recurso terapêutico utilizado em Terapia Ocupacional consiste na atividade humana, tem-se que a ela é considerada o elemento centralizador e orientador na construção complexa e contextualizada do processo terapêutico. As atividades humanas são constituídas por um conjunto de ações que apresentam qualidades, demandam capacidades, materialidade e estabelecem mecanismos internos para sua realização (PRADO; BARTALOTTI, 2001).

Ainda, para os autores citados anteriormente, a saúde pode então ser compreendida como produção de vida, o que implica em uma multiplicidade de intervenções.

Através da atividade de utilização do computador, ocorre a criação de novas possibilidades e finalidades na intervenção de terapia ocupacional, visto que proporcionam um conhecimento e uma experiência de acordo com os interesses, necessidades e potencialidades de cada paciente, e oferecem aos mesmos instrumentos que sejam para seu próprio uso, ampliando a comunicação, permitindo crescimento pessoal, autonomia, interação social e inclusão cultural.

3.1.2 A tecnologia informática na prática clínica de terapia ocupacional

O desenvolvimento satisfatório da informática na prática clínica requer habilidade para analisar e integrar três sistemas de trabalho diferentes. O primeiro sistema conceitual se relaciona ao enfoque mecânico do computador e os aspectos

biomecânicos de acesso: o terapeuta deve igualar o equipamento tecnológico com as habilidades físicas do usuário final, neste caso, o paciente.

O segundo sistema conceitual refere-se à análise dos programas: este sistema inclui não só as operações lógicas estruturadas, mas também as funções cognitivas, de integração sensorial e motora, inerentes à utilização clínica do programa, considerando o objetivo terapêutico. O terceiro sistema conceitual refere-se à orientação biopsicossocial que envolve aspectos referentes às perspectivas das atividades que serão realizadas e suas implicações para o próprio paciente e para o processo terapêutico como um todo (OKOYE, 1998).

Assim, ao utilizar a informática na prática clínica, faz-se necessário integrar estes três aspectos, considerando os objetivos e expectativas do paciente com relação ao processo de tratamento, desenvolvendo condições adequadas de acesso, proporcionando oportunidades de desenvolvimento, não apenas com relação ao aspecto físico, mas também psicológico e social, considerando sempre o contexto de vida ao qual a pessoa pertence.

As ferramentas da tecnologia oferecem meios alternativos pelos quais certos objetivos clínicos podem ser alcançados. A tecnologia da informática oferece alternativas totalmente flexíveis e um meio não estruturado, como um conjunto de instrumentos simples, os quais certos tipos de objetivos de tratamento podem ser alcançados, como por exemplo, a recuperação das funções cognitivas, a integração das funções sensório-motoras, exploração pré-vocacional, treinamento e aquisição de habilidades motoras, déficits neuropsicológicos, exercícios terapêuticos, biofeedback, acesso a dispositivos ambientais e outros dispositivos tecnológicos.

Ao utilizar o computador como recurso para prática clínica, deve-se comparar as demandas de uma aplicação informática específica com as habilidades e necessidades específicas do paciente. Além disso, os programas e os dispositivos periféricos utilizados devem ser ajustados, e ao mesmo tempo adaptados para satisfazer as necessidades de desenvolvimento, psicológicas, emocionais, físicas e cognitivas do paciente, visando também atingir o objetivo de tratamento.

3.2 SOFTWARE

O software pode ser aplicado em várias situações. AGUILERA (2015), nos mostra que em todos os problemas em que estabelecemos um conjunto especifico de ações que nos levam a uma solução é um algoritmo. Utilizamos linguagens de programação para implementar esses algoritmos. Também podemos aplicar a situações em que o problema pode ser descrito formalmente, na maioria das vezes de forma recursiva, nesse caso não existe a necessidade de descrever uma resolução, o problema já está resolvido, simplesmente descreve-se o problema em si, indicando qual a solução desejada, utilizando linguagens declarativas. Também podemos aplicar a problemas que, nós humanos, resolvemos usando várias regras heurísticas, talvez contraditórias, que são utilizadas por um especialista e até mesmo para problemas de sistema que não conseguimos entender claramente como são resolvidos, mas sabemos qual é a solução adequada para alguns exemplos a partir dos dados de entrada, nesse caso é usado redes neurais.

Em todos os casos é difícil estabelecer categorias genéricas significativas para a aplicação do software. Conforme aumenta a complexidade, torna-se mais complicado de criar compartimentos nítidos.

3.2.1 Software de sistemas

Barreto (2008), diz que o software de sistemas é mais conhecido como Nível de linguagem de montagem, ele é implantado por tradução em vez de interpretação como os níveis de microarquitetura, ISA e de máquina de sistema operacional. Os tradutores são programas que convertem um programa escrito pelo usuário em qualquer linguagem de programação para outra linguagem. Se houver um processador para executar diretamente programas escritos em linguagem fonte, então não existe a necessidade de traduzir o código fonte para a linguagem alvo.

Barreto (2008) continua dizendo que a tradução é usada quando há um processador para a linguagem alvo, mas não para a linguagem fonte. Nela o programa original na linguagem fonte não é executado diretamente, ele é convertido para um programa equivalente chamado programa objeto, ou programa binário executável. Há duas etapas distintas na tradução:

1. Geração de um programa equivalente na linguagem alvo;

2. Execução do programa recém gerado;

Aguilera (2015) ainda diz que na interpretação há somente uma etapa: a execução do programa fonte. Não é preciso gerar nenhum programa equivalente. Enquanto o programa objeto está sendo executado, apenas 3 níveis estão em evidência: o nível de microarquitetura, o nível ISA e o nível de máquina de sistema operacional.

3.2.2 Software de tempo real

Farines (2000) conta que hoje, a rapidez na tomada de decisões, nas comunicações e nas atividades em geral, se tornou um dos paradigmas dominantes na sociedade. Utiliza-se cada vez mais o termo Tempo Real em diversas situações, as vezes com propriedade, outras apenas com objetivo comercial. O tempo está sempre presente em todas as atividades mesmo que não seja de forma explicita. Os computadores também seguem essa regra.

O autor continua mostrando que tem crescido o número de aplicativos e programas que apresentam comportamentos definidos pelas restrições temporais. Alguns exemplos são o controle de plantas industriais, de tráfego aéreo ou ferroviário, telecomunicações, eletrônica embarcada em carros e aviões, na robótica, em sistemas de multimídia, etc. Elas apresentam a característica de estarem sujeitas a restrições temporais, são agrupados no que chamamos de Sistemas de Tempo Real.

Farines (2000) encerra dizendo que grande parte dos sistemas de tempo real são projetados e implementados com ferramentas convencionais de verificação de implementação. Por exemplo, na prática corrente, são usadas linguagens de alto nível com construções não determinísticas ou mesmo linguagens de baixo nível, mas sem a preocupação de tratar o tempo de uma forma mais clara, o que torna difícil a garantia de implementação das restrições temporais. Os sistemas operacionais e suportes de tempo de execução geralmente utilizados apresentam mecanismos para implementar escalonamentos dirigidos a prioridades, que não refletem as restrições temporais definidas para esses programas.

3.2.3 Software de gestão

Aguilera (2015) diz que o processamento de informação de gestão é desde o maior princípio do computador é uma das maiores áreas de atuação da informática. Esses programas utilizam quantidades grandes de informações armazenadas em um banco de dados, a fim de facilitar as transações comerciais ou decisões. Além das tarefas de processamento de dados convencional, em que o tempo de processamento não é crítico e os erros podem ser corrigidos. Dentro desta categoria estão inclusos os softwares interativos, que servem de suporte para transações comerciais.

3.3 ENGENHARIA DE SOFTWARE

O Software está presente em todas as áreas do cotidiano, seja controlando, entretendo, vigiando, comunicando, etc. Então, pode-se considerar o software como um componente fundamental do dia a dia. Ele precisa funcionar, dar respostas precisas, ser rápido e barato. Como seria possível conceber e produzir algo tão especial? Existem diversas formas. Uma delas é contratar profissionais com grande experiência na aplicação para desenvolver o software. A outra, mais permanente, é implantar processos, métodos e ferramentas para que o desenvolvimento seja mais previsível independente de quem produza o software. Assim, a engenharia de software desempenha um papel fundamental para desenvolver softwares com qualidade e produtividade (HIRAMA, 2012).

Hirama (2012), ainda nos diz que o conceito “Engenharia de Software” foi criado em 1969 por Fritz Bauer em uma conferência patrocinada por um comitê de Ciência da Organização do Tratado do Atlântico Norte, também conhecido por Otan, no momento em que a chamada, crise do software, precisava de uma solução para a demanda crescente por software, dentro de custo e prazo adequados.

A Engenharia de Software é essencial na construção de um sistema de software, pois ela serve de guia para o desenvolvedor desde as primeiras entrevistas realizadas com o cliente até a entrega do sistema e a manutenção do mesmo. Notamos que a Engenharia de Software segue os mesmos princípios de uma disciplina de engenharia tradicional, onde se baseiam em uma relação adequada de custo/benefício do produto, que não falhe e que seja eficiente.

Tonsing (2008), diz que todos os sistemas possuem um objetivo explicitamente declarado e que o fato de um sistema expressar claramente seu objetivo não significa que ele não possa ter outros objetivos inerentes a seu propósito, os quais podem encontrar-se ocultos ou correlatos ao explícito.

As atividades genéricas que devem ser seguidas em todos os processos iniciais de “fabricação” de um software são:

a) Especificação: o que o sistema deve fazer e suas restrições de desenvolvimento.
b) Desenvolvimento: Produção do sistema de software.
c) Validação: Verificação de que o software é o que o cliente deseja.
d) Evolução: Mudança do software em resposta às demandas de mudança. (SOMMERVILLE, 2007)

3.3.1 Modelagem

Tonsig (2008) mostra que a linguagem de modelagem unificada (UML) não é apenas um método de desenvolvimento como a maioria das pessoas mostram, e sim uma linguagem de modelagem gráfica que pode ser utilizada para descrever e documentar um projeto de software. Um fator importante é que ela simplifica o complexo processo de análise, projeto e construção de software, criando visões do sistema que está sendo desenvolvido.

Tonsig (2008), também diz que a UML tem como objetivo promover aos desenvolvedores de software uma linguagem visual completa, com o intuito de alcançar os seguintes aspectos:

a) Disponibilidade de mecanismos de especificações que possam expressar os níveis conceituais;
b) Independência de processos de desenvolvimento de linguagens de programação;
c) Incentivo ao crescimento das aplicações desenvolvidas no conceito da orientação a objetos;
d) Permissão de suporte a conceitos de desenvolvimentos de alto nível, tais como frameworks, padrões e componentes.

3.3.2 UML

A anotação ocupa uma parte importante em qualquer modelo, a UML (Unified Modeling Language ou Linguagem de Modelagem Unificada), com seu conjunto de anotações tornou-se a linguagem padrão de modelagem adotada internacionalmente pela indústria de Engenharia de Software, oferecendo um grande número de diagramas enfocando tanto características estruturais como comportamentais de um software. Gilleanes (2008) nos mostra que a UML surgiu a partir de três metodologias de modelagem, BOOCH, OMT (Object Modeling Technique) e OOSE (Object-oriented software engeneering).

Terry (2001) mostra que a metodologia OMT era forte em análise e mais fraco em design, Booch (1991) era forte em design e mais fraco em análise e OOSE era forte em comportamento de análise e mais fraco nas outras áreas.

Em 1996 a sua primeira versão foi lançada e grandes empresas atuantes na área de modelagem e desenvolvimento de software passaram a contribuir com o projeto. Fazendo com que a UML fosse adotada pela OMG, (Object Management Group ou Grupo de Gerenciamento de Objetos) em 1997, como uma linguagem padrão de modelagem. Em decorrência disso, existe atualmente uma grande valorização e procura por profissionais que dominem esta linguagem por parte do mercado.

A UML em resumo é um canal de comunicação entre os membros da equipe de desenvolvimento que define, por meio de anotações, padrões que facilitam as atividades da manutenção do software e permite que novos colaboradores possam ser treinados melhorando a qualidade do software.

Terry (2001) diz que um projeto desenvolvido com sucesso satisfaz o cliente em tempo e de maneira econômica, mesmo sendo passível de mudança e adaptação. Dentro do ciclo de vida de desenvolvimento é necessário promover a criatividade e inovação, sendo controlado e medido para garantir que o projeto seja completado.

Em um ciclo de vida iterativo e incremental, o desenvolvimento procede como uma série de iterações que evoluem para o sistema final. Uma iteração consiste de um ou mais componentes de processo: Modelagem comercial, exigências, análise, design, implementação, teste e distribuição. Esse ciclo de vida é um processo de

aliviar o risco técnico, que é estimado e tem sua prioridade avaliada no estágio inicial, sendo revistos durante todo o processo.

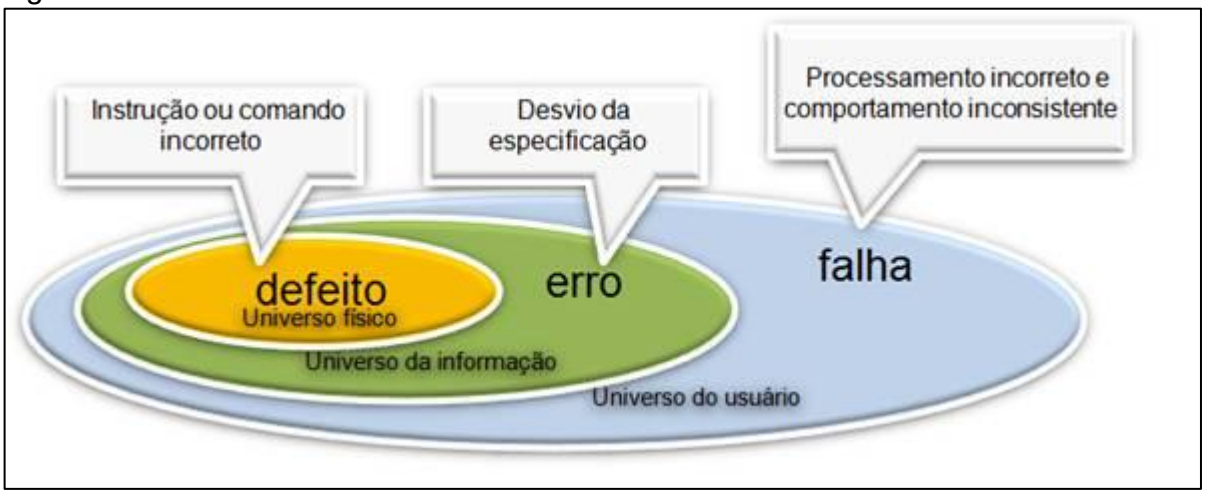
3.4 TESTE DE SOFTWARE

Antes de iniciarmos uma discussão sobre teste de software precisamos esclarecer alguns conceitos relacionados a essa atividade. Inicialmente, precisamos conhecer a diferença entre defeitos, erros e falhas. As definições que iremos usar aqui seguem a terminologia padrão para Engenharia de Software do IEEE – Institute of Electrical and Electronics Engineers – (IEEE 610, 1990 citado por Claudio ([201-]).

a) **Defeito** é um ato inconsistente, cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema, utilizar um método ou uma ferramenta. Por exemplo, uma instrução ou comando incorreto.
b) **Erro** é uma manifestação concreta de um defeito num artefato de software. Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.
c) **Falha** é o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros, sendo que alguns podem nunca causar uma falha.

A Figura 2 expressa a diferença entre esses conceitos.

Figura 2 – Defeito X Erro X Falha



Fonte: Claudio ([201-]).

Como pode ser observado na Figura 2, defeitos fazem parte do universo físico (a aplicação propriamente dita) e são causados por pessoas, por exemplo, através do mau uso de uma tecnologia, podendo ocasionar a manifestação de erros em um produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo de informação). Por fim, os erros geram falhas, que são comportamentos inesperados em um software, que afetam diretamente o usuário final da aplicação (universo do usuário), e podem inviabilizar a utilização de um software.

Segundo Claudio ([201-]), teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado, ao mesmo tempo em que faz parte de todo o processo de engenharia de software. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste dizemos que sua natureza é “destrutiva”, e não “construtiva”, pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.

Ainda segundo o autor, o conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu

comportamento ocorre de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos.

Para Pressman et al. (2011), o processo de testes visa encontrar falhas no sistema para corrigi-las antes de distribuir o sistema, ou de atualizar o software com novos recursos, portanto o sistema deve ser projetado e implementado pensando na facilidade da realização dos testes. Por sua vez, os testes devem ser feitos levando em consideração certas características para que nada seja deixado para trás.

Em conformidade com o autor, o conceito de testabilidade consiste em se medir o quão simples é o ato de testar um software. As características apresentadas a seguir caracterizam um software a ser testável:

a) Operabilidade: um sistema projetado e implementado tendo em mente a qualidade, terá poucas falhas quando os testes forem realizados.
b) Observabilidade: quando é possível ver com clareza as entradas, saídas e variáveis do sistema fica mais fácil de detectar possíveis falhas.
c) Controlabilidade: entradas geram saídas específicas, e para cada tipo de saída existirá um tipo de entrada específica. Se o engenheiro puder controlar essas entradas ficará mais fácil realizar os testes.
d) Decomponibilidade: o sistema é construído a partir de módulos e pode ser testado em partes.
e) Simplicidade: Quanto mais simples um sistema for atingindo o objetivo, mais simples serão os testes.
f) Estabilidade: Quanto menos alterações o software tiver, menos testes precisarão ser feitos.
g) Compreensibilidade: Quanto mais informações estiverem disponíveis para o entendimento do software mais eficazes serão os testes, isso inclui manuais organizados, detalhados e especificados.

A atividade de teste é composta por alguns elementos essenciais que auxiliam na formalização desta atividade, os quais serão apresentados a seguir.

Caso de Teste: descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado (CRAIG; JASKIEL, 2002).

Procedimento de Teste: é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste. (CRAIG; JASKIEL, 2002).

Critério de Teste: serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto. (ROCHA et al., 2001). Os critérios de teste podem ser utilizados como:

a) Critério de Cobertura dos Testes: que permite a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado. (RAPPS; WEYUKER, 1982). Ou seja, determinar o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste.
b) Critério de Adequação de Casos de Teste: quando, a partir de um conjunto de casos de teste T qualquer, ele é utilizado para verificar se T satisfaz os requisitos de teste estabelecidos pelo critério. Ou seja, este critério avalia se os casos de teste definidos são suficientes ou não para avaliação de um produto, ou uma função. (ROCHA et al., 2001).
c) Critério de Geração de Casos de Teste: quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente. (ROCHA et al., 2001).

3.4.1 Níveis de teste de software

O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do software (Figura 3). Segundo Rocha et al. (2001) definimos que os principais níveis de teste de software são:

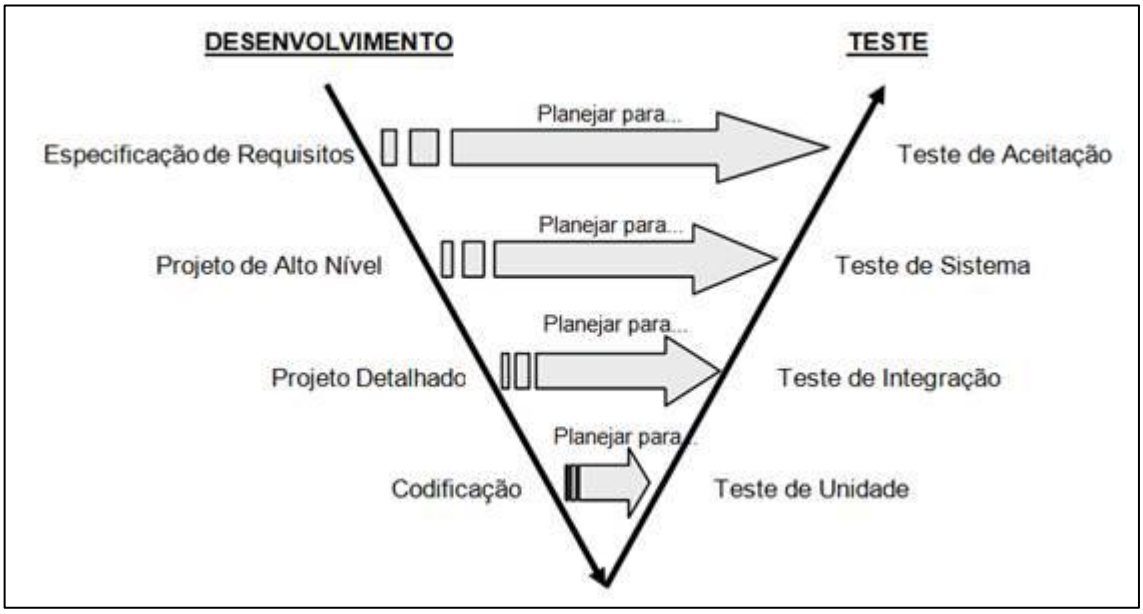
a) Teste de Unidade: também conhecido como teste unitário. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos, ou mesmo pequenos trechos de código.

b) **Teste de Integração:** visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.
c) **Teste de Sistema:** avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia a dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.
d) **Teste de Aceitação:** são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.
e) **Teste de Regressão:** Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

Dessa forma, em conformidade com a Figura 3, o planejamento e projeto dos testes devem ocorrer de cima para baixo, ou seja:

1. Inicialmente é planejado o teste de aceitação a partir do documento de requisitos;
2. Após isso é planejado o teste de sistema a partir do projeto de alto nível do software;
3. Em seguida ocorre o planejamento dos testes de integração a partir o projeto detalhado;
4. E por fim, o planejamento dos testes a partir da codificação.

Figura 3 - Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software.



Fonte: (Craig e Jaskiel, 2002).

3.4.2 Técnicas de teste de software

Atualmente existem muitas maneiras de se testar um software. Mesmo assim, existem as técnicas que sempre foram muito utilizadas em sistemas desenvolvidos sobre linguagens estruturadas, que ainda hoje têm grande valia para os sistemas orientados. Apesar dos paradigmas de desenvolvimento serem diferentes, o objetivo principal destas técnicas continua a ser o mesmo: encontrar falhas no software.

De acordo com Claudio ([201-]), as técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste. Elas contemplam diferentes perspectivas do software, e impõe-se a necessidade de se estabelecer uma estratégia de teste que contemple as vantagens e os aspectos complementares dessas técnicas. As técnicas existentes são: técnica funcional e estrutural.

Outras técnicas de teste que podem e devem ser utilizadas de acordo com necessidades de negócio ou restrições tecnológicas são: teste de desempenho, teste de usabilidade, teste de carga, teste de stress, teste de confiabilidade e teste de recuperação.

Segundo Claudio ([201-]), alguns autores chegam a definir uma técnica de teste, caixa cinza, que seria um mesclado do uso das técnicas de caixa preta e caixa

branca, mas, como toda execução de trabalho relacionado à atividade de teste utilizará simultaneamente mais de uma técnica de teste, é recomendável que se fixe os conceitos primários de cada técnica.

3.5 BANCO DE DADOS

Date (2004), um sistema de banco de dados é um sistema computadorizado de manutenção de registros. O banco de dados pode ser considerado como o equivalente eletrônico de um armário de arquivamento. Ele é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados. Os usuários de um sistema podem realizar (ou melhor, solicitar que o sistema realize) diversas operações envolvendo tais arquivos. Um banco de dados pode:

a) Acrescentar novos arquivos ao banco de dados;
b) Inserir dados em arquivos existentes;
c) Buscar dados de arquivos existentes;
d) Excluir dados de arquivos existentes;
e) Alterar dados em arquivos existentes;
f) Remover arquivos existentes do banco de dados;

Duarte (2006), o termo banco de dados pode ser visto de duas maneiras distintas. Alguns falam que banco de dados é o mesmo que SGBD (Sistema Gerenciador de Banco de Dados), um programa utilizado para gerenciamento de dados. Também é utilizado para se definir uma base de dados (dados agrupados por um SGBD), para criar uma base de dados o SGBD utiliza uma linguagem. A mais utilizada atualmente é a linguagem SQL (Structured Query Language).

Duarte (2006) afirma que existem vários SGBDs disponíveis no mercado. Pagos e gratuitos. Para criar um banco de dados não é preciso conhecimento na linguagem SQL, atualmente existem programas que possuem uma interface gráfica intuitiva, gerando um código em SQL automaticamente. Alguns exemplos mais conhecidos de SGBD:

a) **SQLServer:** um dos maiores SGBD do mundo, sob licença da Microsoft, existem versões pagas e gratuitas;
b) **MySQL:** o MySQL é um software livre, com código fonte aberto e de uso totalmente gratuito;

c) **FirebirdSQL:** roda na maioria dos sistemas Unix, e tem código fonte aberto;
d) **MSQL:** criado pela Hughes Technologies Pty Ltda. trabalha com o uso eficiente da memória, e é um sistema pequeno. Sua licença é altamente controlada pela empresa dona do produto;
e) **Microsoft Access:** é um SGBD da Microsoft que acompanha o pacote Office. É muito utilizado para a aprendizagem e tem poucas atribuições profissionais, devido a sua limitação muito grande em armazenamento.

É necessário criar tabelas e colunas, dentro das tabelas, para que se possa armazenar um dado. Os comandos mudam para cada SGBD. Mesmo a linguagem sendo a mesma, o comando de declaração varia entre cada Banco de Dados.

3.5.1 SQLite

SQLite é uma biblioteca em linguagem C que implementa um banco de dados independente de servidor, sem a necessidade de configuração e não é diretamente comparado com SGBDs citados acima, desde que ele busque uma solução para um problema diferente. Os motores de banco de dados SQL que possuem conexão servidor/cliente buscam implementar um repositório para compartilhamento de dados de uma empresa ou um serviço. Em ênfase, escalabilidade, concorrência, centralização e controle. O SQLite se esforça para manter um armazenamento local dos dados para as aplicações e dispositivos individuais. Podemos dizer que o SQLite enfatiza a economia, eficiência, fiabilidade, independência e simplicidade (SQLITE, 2016).

Com todos os recursos ativos, a biblioteca pode ser inferior a 500kb, o SQLite pode ser configurado para ser utilizado em um espaço mínimo de 4kb e de 100kb, o tornando uma escolha muito popular para uso em dispositivos com memórias muito restritas, como celulares, PDAs, MP3 Players. O acesso ao Banco de Dados é rápido, mesmo em dispositivos com pouca memória. Seu código fonte é mantido por uma comunidade de desenvolvedores no mundo inteiro e é livre (SQLITE, 2016).

3.6 LINGUAGENS DE PROGRAMAÇÃO

A parte física do computador é chamada de hardware, que é constituído por uma Unidade Central de Processamento (UCP), memória e dispositivos de entrada e saída. A UCP (ou processador) contém um conjunto relativamente pequeno de instruções que é capaz de executar. Cada processador contém um conjunto diferente de instruções, apesar de similares entre si. Essas instruções podem ser desde operações matemáticas a interações com os dispositivos de entrada e saída. Nós chamamos de algoritmos, ou programas, um conjunto de instruções que será executado pelo processador, esse algoritmo leva o computador a executar alguma tarefa (MEDINA, 2006).

Para implementar um algoritmo em um computador, precisamos descrever de uma maneira que o computador consiga entender. Essa descrição é feita através de uma “linguagem de programação”. O próprio conjunto de instruções executadas por um processador pode ser entendido como uma “linguagem de programação”, porém, essa linguagem não é a mais adequada para a descrição de um programa, uma vez que os algoritmos necessários podem ser sofisticados, e essa linguagem primitiva, também chamada de “Linguagem de Máquina” a qual é muito complexa ao programador, demandando um grande esforço na elaboração de programas mais complexos.

Nas primeiras linguagens de programação, todas as estruturas de dados do espaço do problema tinham de ser modeladas com apenas poucas estruturas de dados suportadas pela linguagem. Por exemplo, nas versões do Fortran pré-90, as listas ligadas e as árvores binárias eram implementadas com vetores (SEBESTA, 2011).

Assim, foram desenvolvidas, ao longo da história da computação, diversas “linguagens de programação”, cada qual, a seu tempo, introduzindo facilidades e recursos que foram tornando a programação mais acessível e com menos chances de ocorrer erros. Hoje graças as linguagens de alto nível, e também como chamada por alguns, linguagens de quarta geração, programar deixou de se tornar restrito e tornou-se um escopo de usuários mais comuns (GUDWIN, 1997).

3.6.1 Linguaguem C++

A Linguagem C evoluiu a partir do BCPL e do B. A linguagem BCPL foi desenvolvida em 1967 por Martin Richards, como uma linguagem para escrever

software de sistemas operacionais e compiladores. Ken Thompson modelou muitos recursos em sua linguagem B baseando suas contrapartes em BCPL, utilizando o B para criar versões anteriores do sistema operacional UNIX na Bell Laboratories em 1970 (DEITEL, 2010).

Deitel (2010), diz que a linguagem C originalmente implementada em 1972 foi desenvolvida a partir da linguagem B. Ela Inicialmente tornou-se amplamente conhecida como a linguagem de desenvolvimento do sistema operacional UNIX. Hoje, a maior parte do código para sistemas operacionais de uso geral é escrita em C e C++.

O C++, uma extensão do C, foi desenvolvida no início da década de 1980, por Bjarne Stroustrup. Ele fornece vários recursos que sofisticam a linguagem C, mas, sobretudo, fornece capacidades para a programação orientada a objetos. A linguagem C++ é hibrida, isso quer dizer que é possível programar tanto no estilo C como no estilo orientado a objetos, ou em ambos. (DEITEL, 2010).

3.6.2 Linguagem Lua

Lua é uma linguagem de programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados distribuído sobre um pequeno pacote e compila sem modificações em todas as plataformas que têm um compilador C padrão. Lua roda em todos os tipos de Unix e Windows, e também em dispositivos móveis (usando Android, iOS, BREW, Symbian, Windows Phone), em microprocessadores embutidos (como ARM e Rabbit, para aplicações como Lego MindStorms), e até mainframes IBM. (LUA, 2016)

Lua combina sintaxe procedural simples com construções para descrição de dados baseadas em tabelas associativas e semântica extensível. Lua é tipada dinamicamente, é executada via interpretação de bytecodes para uma máquina virtual baseada em registradores, e tem gerenciamento automático de memória com coleta de lixo incremental. Essas características fazem de Lua uma linguagem ideal para configuração, automação e prototipagem rápida. (LUA, 2016)

A linguagem é inteiramente projetada, implementada e desenvolvida no Brasil, por uma equipe na PUC-Rio. Lua nasceu e cresceu no Tecgraf. Atualmente,

Lua é desenvolvida no laboratório LabLua do Departamento de Informática da PUC-Rio. (LUA, 2016)

A linguagem é amplamente utilizada em aplicações como o Adobe's Photoshop Lightroom, em sistemas distribuídos como o middleware Ginga para TV digital e jogos como World of Warcraft e Angry Birds. Atualmente é a linguagem de script mais usada em jogos. Várias versões de Lua foram lançadas e usadas em aplicações reais desde a sua criação em 1993. (LUA, 2016)

3.7 O MERCADO DE DISPOSITIVOS MÓVEIS

O mercado de desenvolvimento para celulares antes era restrito aos fabricantes e operadoras que controlavam a criação e inclusão dos aplicativos em seus aparelhos. A liberação, por parte dos fabricantes, de um kit de desenvolvimento de software (SDK) para suas plataformas e a criação de lojas para a distribuição de aplicativos, viabilizou a abertura deste mercado para praticamente qualquer empresa ou desenvolvedor, criando assim novas oportunidades de negócio (JOÃO, 2015).

A primeira geração de telefones Android foi lançada em outubro de 2008. De acordo com a Gartnet, as vendas de telefones baseadas em Android nos Estados Unidos aumentaram 707% no primeiro trimestre de 2010, em relação ao ano anterior. Em março de 2011, um estudo da Nielsen mostrou que o Android tinha 37% da fatia de mercado de smartphones nos Estados Unidos, comparados com 27% do iPhone da Apple e 22% do Blackberry. Em agosto de 2010 mais de 200 mil smartphones Android eram ativados todos os dias, ultrapassando de 100 mil por dia, apenas dois meses antes. Em junho de 2011 mais de 500 mil dispositivos Android diferentes em todo o mundo (DEITEL, 2013).

João (2015) também conta que a plataforma Android hoje, por causa da sua grande fatia no mercado, também por oferecer uma API rica, disponibilizando acesso a vários recursos de hardware, como Wi-Fi e GPS, além de outras ricas ferramentas para o desenvolvedor. O Android possui a facilidade de se desenvolver utilizando a linguagem de programação Java, que já é bastante disseminada, a simplicidade e o baixo custo para a publicação de aplicativos em sua loja, a Google PlayStore.

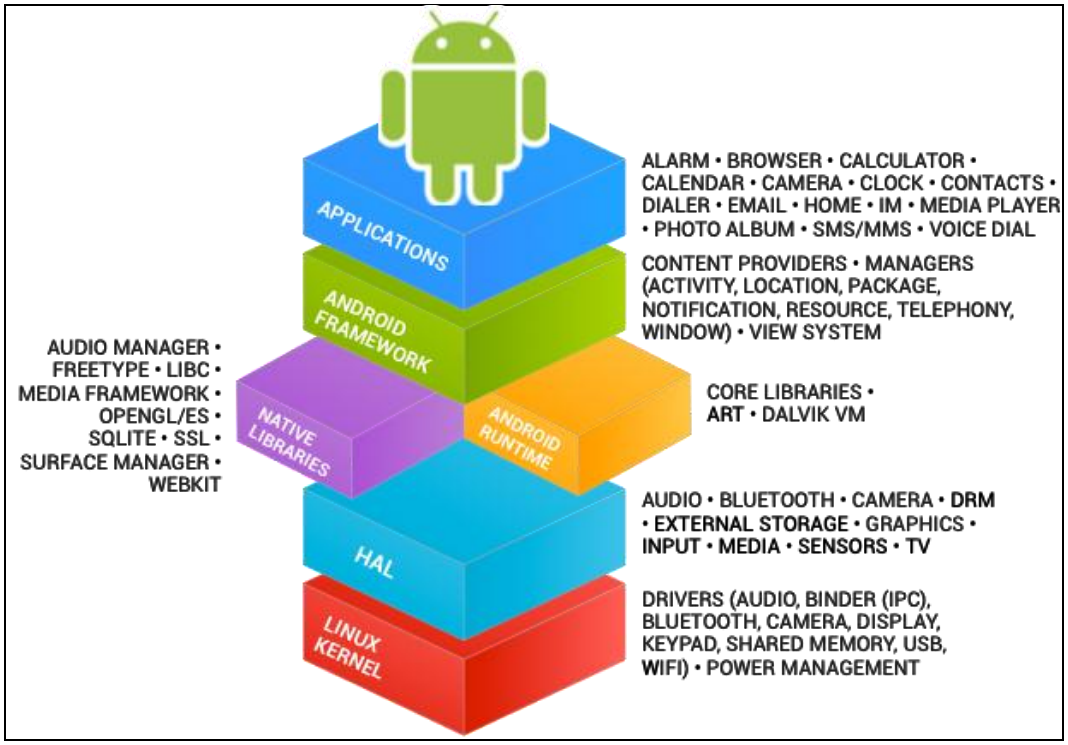
Em um relatório do IDC (International Data Corporation), publicado em agosto de 2015, o Android possuía 82,8% do mercado de dispositivos móbile. Em segundo lugar, aparece o iOS, sistema operacional do Apple iPhone, com 13,9%. O Gráfico

abaixo mostra a participação dos principais sistemas operacionais e a quantidade de aparelhos distribuídos.

3.7.1 Arquitetura de desenvolvimento Android

Segundo Lecheta (2010), A arquitetura do sistema operacional Android é uma pilha de programas agrupados em camadas. Podemos dividir essas camadas em níveis zero, um, dois e três, conforme a Figura 4 abaixo.

Figura 4 - Descrição dos níveis de camadas da arquitetura do SO Android.



Fonte 1: AOSP

Lecheta (2010) diz que no nível zero, temos o Kernel (Linux Kernel), para desenvolvê-la foi utilizado à versão 2.6 do Sistema Operacional Linux. Nele encontraremos os programas de gerenciamento de memória, configurações de segurança e vários drivers de hardware. O Kernel, em face das diversas arquiteturas de dispositivos móveis, assim como em computadores e outros dispositivos eletrônicos, para que um sistema consiga funcionar com diferentes tipos de hardware, são necessários os sistemas operacionais. Segundo TANENBAUM (2006), um sistema operacional deve ser capaz de gerenciar o processador,

memória e outros dispositivos de entrada e saída, além de fornecer, aos programas de usuário uma interface mais simplificada com o hardware.

Nos dispositivos móveis, existe a demanda de gerenciar os recursos dos mesmos, tais como processamento e memória, sendo assim, há uma forte tendência de criação de sistemas operacionais com esse intuito.

Lecheta (2010) continua dizendo que no nível um, encontramos as camadas de bibliotecas, também chamadas de Libraries, e tempo de execução, conhecido como Android RunTime. A camada de biblioteca é um conjunto de instruções que dizem ao dispositivo como lidar com diferentes tipos de dados, incluindo um conjunto de biblioteca C / C++ usadas por diversos componentes do sistema e são expostas aos desenvolvedores através da estrutura de aplicativo Android. A camada de tempo de execução inclui um conjunto de bibliotecas do núcleo Java, também chamado de Core Libraries. Para desenvolver aplicações para o Android, os programadores utilizam a linguagem de programação Java, nesta camada encontraremos a Máquina Virtual Dalvik (DVM). O Android utiliza as máquinas virtuais Dalvik para rodar cada aplicativo com seu próprio processo. Isso é importante por algumas razões: nenhum aplicativo é dependente de outro e se um aplicativo parar, ele não afeta quaisquer outros aplicativos em execução no dispositivo e isso simplifica o gerenciamento de memória, pois a máquina virtual é baseada em registradores e desenvolvida de forma otimizada para utilizar pouca memória e permitir que múltiplas instâncias executem ao mesmo tempo. Da mesma forma que com o Java, uma aplicação Android é interpretada pelo sistema. A diferença é que o Android gera códigos Dalvik Executáveis (.dex), e não os byte-code (.class) do Java, tais código são interpretados pela MVD (PROJECT, 2016).

Lecheta (2010) diz também que no nível dois, temos a camada de framework de aplicação, conhecido como Application Framework. Programas que gerenciam as aplicações básicas do telefone. Os desenvolvedores têm acesso total ao framework como um conjunto de ferramentas básicas com o qual pode construir ferramentas mais complexas, sendo um diferencial do Android em relação as outras plataformas para dispositivos mobile, a possibilidade de acesso e modificação dos componentes permite que as aplicações criadas possam interagir com o sistema do celular ao todo, podendo o usuário ou desenvolvedor, alterar qualquer componente que faça parte do sistema para que adequar as suas necessidades ou as do aplicativo que se está em desenvolvimento.

Cada aplicativo no Android é compilado com os arquivos de recursos e configuração em um arquivo .apk, e pode ser instalado através de uma ferramenta chamada AAPT (Android Asset Packaging Tool), que realiza o gerenciamento dos pacotes e instala os arquivos .apk no sistema (PROJECT, 2016).

Lecheta (2010) conclui dizendo que no nível três, encontramos a camada de aplicações e as funções básicas do dispositivo. Esta é a camada de interação entre o usuário e o dispositivo móvel, nela encontramos aplicativos, cliente de e-mail, SMS, calendário, mapas, navegador, contatos entre outros.

4 TRABALHOS CORRELATOS

Os estudos sobre paralisia cerebral vêm se desenvolvendo ao longo das últimas décadas, mas apenas recentemente tem se ouvido falar sobre usabilidade e acessibilidade, os pacientes e portadores de PC começaram a receber atenção dos desenvolvedores de software nos últimos tempos.

O analista de sistemas Carlos Pereira, pai de clara, portadora de paralisia cerebral. Desenvolveu uma ferramenta, chamada LVOX, que dá acesso a uma série de palavras e desenhos simples, permitindo expressar desejos e necessidades. Com ela é possível ao paciente descrever emoções e sensações como “entediado”, “com dor”, “com fome” ou indicar ações como, por exemplo, ao escolher “comer” o aplicativo direciona para diversas alternativas de comida (PEREIRA, 2013).

Marcela (2015) nos apresenta Kevin Oliveira, estudante de engenharia da computação e um dos criadores do Laví explica que o objetivo do aplicativo desenvolvido para smartphones e tablets é permitir ao portador de necessidades especiais que tenha dificuldades na fala, por exemplo, expressar por meio de ilustrações o que quer e/ou precisa fazer.

João Santiago, jovem de 23 anos portador de Paralisia Cerebral. Certa vez foi convidado para ir a um bar onde recusou o convite por não saber o quanto o local era acessível. Vendo que muitas pessoas sofrem com o mesmo problema, resolveu agir e desenvolveu um aplicativo para classificar a acessibilidade dos lugares. Ajudando as pessoas com deficiência a ter uma vida social, o aplicativo se chama “Dá Pra Ir” (CANDIDO; LELIS, 2015).

5 METODOLOGIA

Nos tópicos a seguir, será apresentada a metodologia proposta para desenvolvimento do trabalho

5.1 TIPO DE PESQUISA

O trabalho desenvolvido tem por finalidade auxiliar o tratamento ocupacional utilizando aparelhos moveis, buscando uma forma de realizar em sua residência, de forma simples e pratica, um exercício auxiliar.

Gil (2010), nos mostra o significado de uma pesquisa exploratória, ela tende a ser flexível, pois leva muitos aspectos relativos à situação estudada em consideração. O principal objetivo de uma pesquisa exploratória é possibilitar um índice maior de convivência com o problema estudado, tornando-o mais compreensível ao pesquisador.

Podemos então, concluir que este projeto foi uma pesquisa exploratória, pois teve como o objetivo estudar e investigar métodos de construção de software que sirva como recurso terapêutico para auxiliar os portadores de PC, junto a TO, no tratamento e recuperação da disfunção motora nos membros superiores.

O projeto foi dividido em duas etapas. Iniciado com uma pesquisa e levantamento bibliográfico referente ao assunto, técnicas, ferramentas e linguagens de programação as quais a pesquisa será direcionada, podendo destacar: Software, Características da Paralisia Cerebral, deficiência motora, acessibilidade, usabilidade, desenvolvimento, criança, arquitetura de software para dispositivos móveis, recurso de multimídia, linguagem de programação LUA, biblioteca SQLITE, UML, entre outros.

Após a pesquisa bibliográfica, um estudo de campo foi realizado na Clínica de Terapia Ocupacional da USC com crianças que apresentam deficiência motora nos membros superiores, foi realizada para coletar e analisar informações visando o melhor conhecimento do dia a dia do paciente. O aplicativo foi desenvolvido com a finalidade do uso do aparelho móvel, não apenas para pacientes portadores de PC, mas também para quaisquer outros que possuam dificuldades motoras e possam utilizar o aplicativo para auxílio no seu desenvolvimento.

Após coletados os dados, os resultados foram analisados e foi iniciado o processo de criação da arquitetura do software, e consequentemente, os requisitos necessários para atender os pacientes participantes do projeto.

Juntamente, foi realizado um estudo sobre arquiteturas de software e linguagem de programação LUA, visando melhor aproveitamento da tecnologia oferecida, vinculado à modelagem do software em questão.

A busca por melhorias de acessibilidade e usabilidade é o objetivo dos testes, bem como, os aspectos sistêmicos, com a finalidade de buscar defeitos no software.

O teste procura descobrir defeitos no software quando seu comportamento não está correto ou não está em conformidade com a sua especificação. (HIRAMA, 2012).

Ainda nesse contexto, Myers (1990 apud Pressman, 1995) destaca uma série de regras que podem servir como objetivos de teste:

a) A atividade de teste é um processo de executar um programa com intenção de descobrir um erro;
b) Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto;
c) Um teste bem-sucedido é aquele que revela um erro ainda não descoberto;

De acordo com Jesus (2013), o Teste Funcional é uma técnica que tem como objetivo localizar os erros do software através da utilização do mesmo sem ter acesso ao código fonte, tendo conhecimento apenas de como o programa deve se comportar. Este tipo de teste baseia-se principalmente nas especificações do software, ou seja, são fornecidos os dados de entrada e após o processamento destes dados, compara-se o resultado obtido com o resultado esperado, já especificado no caso de teste.

Nesse contexto foram realizados testes durante todas as etapas do desenvolvimento do software, ou seja, o mesmo será testado em todas as suas fases, e a técnica utilizada será do Teste Funcional.

Quando realizado uma nova atualização do aplicativo, uma nova observação foi feita para averiguar a evolução do aplicativo e de suas funcionalidades mediante ás necessidades psicomotoras dos pacientes. Como observado, neste processo de

avaliação, os usuários serão convidados a interagir com o software, enquanto serão realizadas anotações em tempo real sobre o desempenho do usuário.

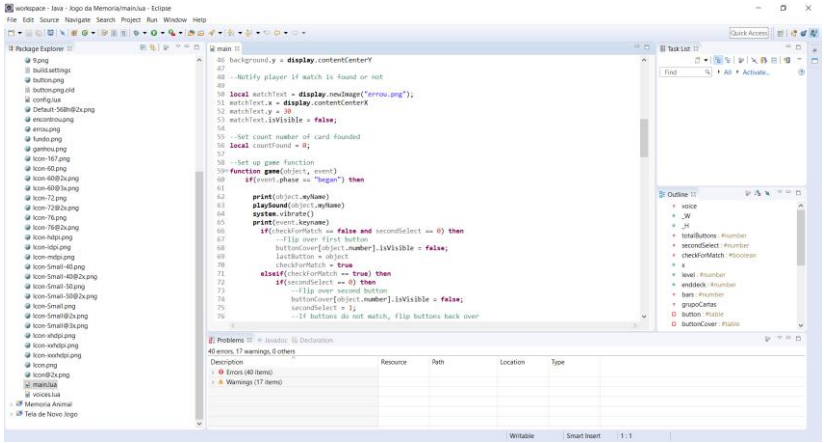
5.2 RECURSOS

Para a realização desta pesquisa foram utilizados um notebook da marca Dell com processador i7 de 2.50ghz (com quatro núcleos reais), 8Gb de memória RAM modelo ddr3 de 1600mhz, 1 disco rígido de 1Tb, com o sistema operacional Windows 10 PRO 64 Bits. No sistema operacional foi instalado os aplicativos Corona Simulator, que é um software compilador e emulador de aplicativos desenvolvidos utilizando a Linguagem LUA, Notepad++, que é um software utilizado para escrita do código-fonte, Eclipse, que é uma ferramenta que auxilia a escrita do código-fonte, e o Audacity, que é um editor e gravador digital de áudio. Um Tablet da marca LG, modelo V480 de 8” com o sistema operacional Android para a realização dos testes.

5.3 DESENVOLVIMENTO

Para o desenvolvimento foi utilizada a ferramenta de desenvolvimento Eclipse com o pacote LUA instalado para auxilio na escrita do código-fonte, a escolha da linguagem LUA se deu pelo fato da mesma ser utilizada especificamente para o desenvolvimento de jogos em diversas plataformas utilizando o mesmo código-fonte, eliminando a necessidade de ser realizada uma nova escrita do código-fonte para uma nova plataforma móvel, abaixo temos a Figura 5 que ilustra a ferramenta Eclipse com um código fonte LUA aberto.

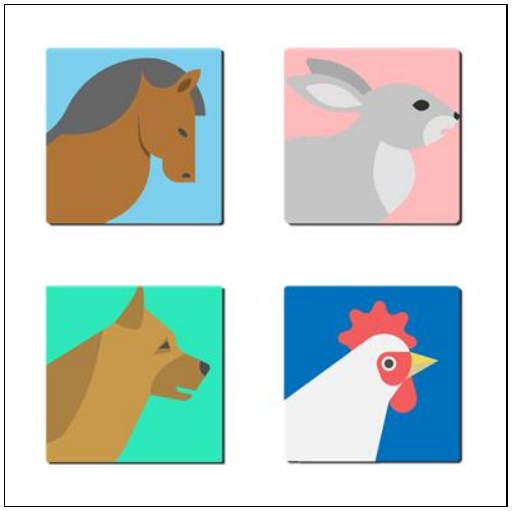
Figura 5 - Ferramenta de desenvolvimento Eclipse



Fonte: Elaborado pelo autor

As imagens utilizadas, foi necessário o auxílio de uma profissional na área de designer para a criação e modelagem das imagens, com o auxílio da mesma foi realizado um modelo de acessibilidade do aplicativo, visando uma melhor acessibilidade do aplicativo, melhor aparência e acessibilidade das informações e opções, temos um exemplo de algumas imagens criadas para utilizar no aplicativo na Figura 6.

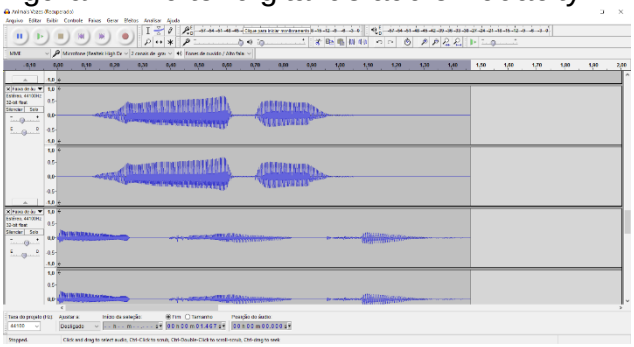
Figura 6 - Algumas imagens utilizadas pela designer



Fonte: Elaborado pelo autor

Os áudios foram capturados utilizando o aplicativo Audacity e um microfone USB Semiprofissional da marca Logitech, permitindo a melhor captura da voz com uma melhor qualidade sonora, abaixo temos a Figura 7 ilustrando o Audacity.

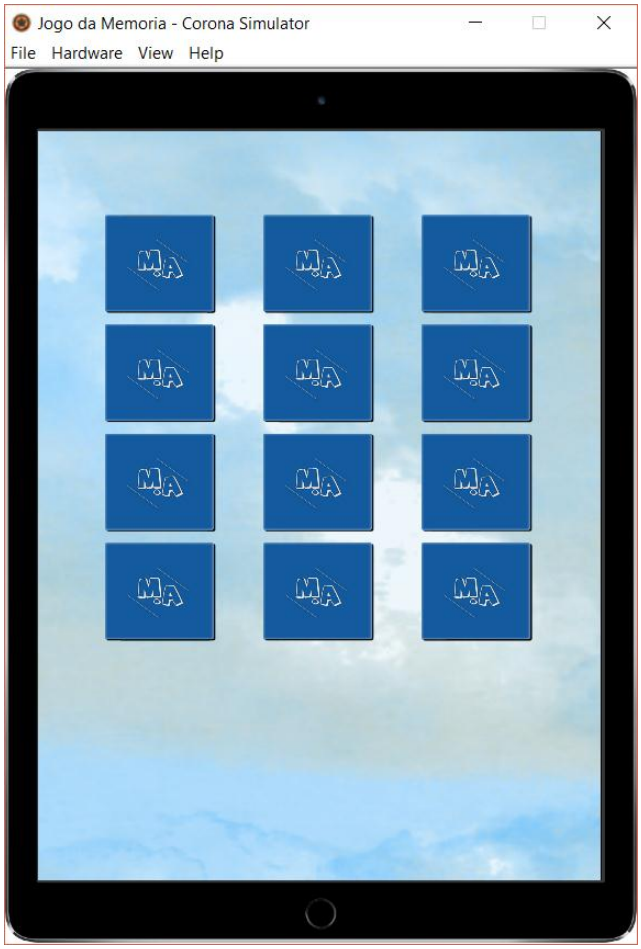
Figura 7 - Editor digital de audio Audacity



Fonte: Elaborado pelo autor

Durante o desenvolvimento do código, o aplicativo passou por testes através do software Corona Simulator, que visa criar um ambiente de execução do aplicativo através da linguagem de programação LUA, temos uma imagem ilustrativa do software através da Figura 8.

Figura 8 - Emulador Corona Simulator



Fonte: Elaborado pelo autor

5.4 EXPERIMENTO E RESULTADOS

Antes de realizar um teste junto a Terapia Ocupacional foi realizado um teste preliminar com a finalidade de coletar de erros e defeitos. O teste foi realizado, com o auxílio e autorização de uma profissional da saúde, em crianças que não possuem PC. Após a realização dos testes e coletados os dados de erros e defeitos, surgiram os aprimoramentos de acessibilidade e design, a fim de tornar o aplicativo atrativo e de fácil utilização para a criança.

O teste realizado com uma criança portadora de PC, do tipo Hemiplegia no lado direito, se deu na clínica de Terapia Ocupacional da Universidade do Sagrado Coração durante o teste foi relatado que a criança apresentou dificuldades durante a execução dos exercícios propostos, a principal dificuldade apresentada pela criança se deu no momento de selecionar uma carta na tela, percebe-se que a criança fez demasiado esforço com o braço direto para mantê-lo no local desejado e com a mão direita para abrir e tocar com os dedos a tela do Tablet, em algumas vezes a criança preferiu utilizar a mão esquerda por melhor atender e corresponder a sua necessidade momentânea. A baixo na Figura 9 temos uma ilustração do momento em que a criança recebe a ajuda para manter o braço direito no alto, facilitando a seleção do objeto no Tablet.

Figura 9 - Paciente recebendo a ajuda da terapeuta

Fonte: Elaborada pelo autor

Após alguns testes realizados, a terapeuta sugeriu que a criança utilizasse uma órtese, que é um dispositivo utilizado para imobilizar o membro como enxergamos a seguir na Figura 10.

Figura 10 - Exemplo de órtese

Fonte: SciELO

Utilizando a órtese, a paciente apresentou uma melhor disposição para a realização da atividade e mais facilidade na movimentação do seu braço direito, entretanto a paciente apresentou dificuldade em movimentos mais sutis como o movimentar do dedo para tocar a tela do Tablet, tornando o exercício muitas vezes exaustivo para a paciente. Notou-se também que a paciente teve algumas dificuldades devido o número de cartas disponíveis para busca na tela, foram disponibilizados 6 tipos de animais para seleção totalizando 12 cartas na tela. Abaixo na Figura 11 o momento em que a paciente está utilizando a órtese para manter seu braço mais firme.

Figura 11 - Paciente utilizando a órtese

Fonte: Elaborado pelo autor

6 CONSIDERAÇÕES FINAIS

Hoje em dia a tecnologia pode ajudar na Terapia Ocupacional, durante o desenvolvimento do aplicativo percebe-se o cuidado que deve ser tomado ao desenvolver um aplicativo especifico para atender ao público portador de PC deve se tomar algumas medidas de acessibilidade e usabilidade, como por exemplo, para um aplicativo de jogo da memória, como o utilizado no projeto, para um portador de PC deve-se ter um dois click para selecionar a carta, sendo um para selecionar a carta e o outro para realizar a confirmação de seleção, pois durante o uso um paciente com PC pode tocar em uma carta por acidente e desejar trocar de carta.

Foi mostrado durante a pesquisa que cada tipo de PC possui a sua particularidade, ou seja, cada portador de um tipo de PC pode reagir de uma forma diferente a um mesmo teste e deve-se buscar a auxilio e opinião profissional para o desenvolvimento de um aplicativo para o público portador de PC.

Foi mostrado interesse pela Clínica de TO da USC durante o desenvolvimento do aplicativo para a utilização do mesmo durante para auxiliar durante o tratamento de seus pacientes.

O aplicativo desenvolvido além de poder ser utilizado para tratamento pode também ser utilizado para coleta de dados para análise e estudos de novos pacientes na clínica, mostrando suas limitações e suas características mais especificas.

Lua é uma linguagem de extensão amplamente utilizada no mundo, apesar de ser uma linguagem de programação desenvolvida no Brasil e ser usada em várias aplicações e programas, como softwares da Ginga, camada de aplicação do sistema brasileiro de televisão digital, e jogos como World of Warcraft e Angry Birds. É uma linguagem muito pouco divulgada aqui no país, excluindo a PUC-RJ de onde nasceu. No mercado a linguagem tem uma força muito maior no nicho de games, atuando em vários jogos conhecidos e em plataformas diferentes, Lua conseguiu se tornar a linguagem de script mais utilizada para esse fim.

7 TRABALHOS FUTUROS

Para trabalhos futuros propõe-se novas maneiras de validações dos testes além de jogo da memória, um exemplo é utilizar vozes de animais e solicitar associações com a imagem do animal.

Adicionar os sons emitidos pelos animais junto as cartas do Jogo da Memória para assim criar associações e mais dinamicidade no aplicativo.

Melhorias no design do aplicativo.

Adicionar opções junto ao aplicativo, podendo remover o som ou selecionar o tamanho desejado da carta na tela.

REFERÊNCIAS

AGUILERA S. **Tipos de Software**. 2015. Disponível em: <http://repositorio.ub.edu.ar:8080/xmlui/handle/123456789/5213> Acesso em 14 maio. 2016.

BARRETO L. **Software de Sistemas – O Nível de linguagem de montagem**. Disponível em: <https://a12672c8-a-62cb3a1a-s-sites.googlegroups.com/site/leonardbarreto/NveldeLinguagemdemontagem.pdf> 2008. Acesso em 17 de maio. 2015

BOBATH, K. A **Deficiência Motora em Pacientes com Paralisia Cerebral**. Editora Manole LTDA. 1979.

CANDIDO. F, LELLIS G. J**ovem deficiente cria aplicativo para avaliar a acessibilidade dos lugares**. Disponível em: <http://revistapegn.globo.com/Banco-de-ideias/Negocios-sociais/noticia/2015/02/jovem-deficiente-cria-aplicativo-para-avaliar-acessibilidade-dos-lugares.html> 2015. Acesso em 21 de maio. 2016

CRAIG R. D. Jaskiel S. P. **Systematic Software Testing**. Artech House. 2002.

DATE. C. J. **Introdução a sistemas de Banco de Dados**. 8ª Ed. Editora Elsevier. 2004.

DEITEL. A., P. Deitel. **Android para programadores - Uma abordagem baseada em aplicativos**. Bookman. 2013.

DEITEL P., H. DEITEL. **Java: Como programar**. 8ª Ed. Editora Pearson. 2010

FARINES J.M FRAGA J. S. OLIVEIRA R.S. **Tipos de Software**. Disponível em: <http://s3.amazonaws.com/academia.edu.documents/31068966/livro-tr.pdf> 2000. Acesso em 20 de maio. 2015

FONSECA, L. F. LIMA. C. L. A. **Paralisia Cerebral – Neurologia – Ortopedia – Reabilitação**. 2ª Ed. Medbook – Editora Cientifica LTDA. 2008.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 5. ed. São Paulo: Atlas, 2010.

GUDWIN R. R. **Linguagens de programação**. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/ea877/lingpro.pdf>1997.

GUEDES T. A. G. **UML - Uma abordagem pratica**. Novatec, 2004.

HIRAMA, K. **Engenharia de software – Qualidade e Produtividade com Tecnologia.** Editora Elsevier. 2012.

IDC: Smartphone OS Market Share 2015, 2014, 2013, and 2012. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> Acesso em 13 de abril. 2016

JESUS, R. C. **Utilização de Frameworks para automatização de testes de aplicativos na plataforma Android**. 2012. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Sagrado Coração, Bauru, 2013.

LECHETA, R. R. **Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Ed. Novatec. 2ª Ed. São Paulo, 2010.

LEITÃO, A. **Paralisia Cerebral – Diagnostico – Terapia – Reabilitação**. Livraria Atheneu. 1983.

LUA, **A Linguagem de Programação**. Disponível em: <https://www.lua.org/portugues.html> Acesso em 23 de novembro. 2016.

M. MEDINA, C. FERTIG. **Algoritmos e Programação**. Editora Novatec. 2006.

MONTEIRO, B. J. (2015). **Google Android – crie aplicações para celulares e tablets**. 1ª Ed, Casa do Código.

MORAES M. **Aplicativo 'Laví', desenvolvido por estudantes da Fucapi, ajuda portadores de paralisia cerebral**. Disponível em: <http://acritica.uol.com.br/noticias/Aplicativo-Lavi-desenvolvido-Fucapi-portadores_0_1490250980.html> 2015. Acesso em 21 de maio. 2016

MYERS, G. **The Art of Software Testing**. Wiley, New York, 1979.

OKOYE, R. **Aplicaciones de la Informatica a la Terapia Ocupacional**. In: Hopkins HL, Smith HD, Willard HS, Spackman CS. Terapia Ocupacional . Madrid: Editorial Médica Panamericana, 1998. p. 342–52.

PEREIRA C. **Aplicativo ajuda crianças com paralisia cerebral - A ferramenta Livox permite expressar desejos e necessidades**. Disponível em: <http://www2.uol.com.br/vivermente/noticias/aplicativo_ajuda_criancas_com_paralisia_cerebral.html>. 2013. Acesso em 21 de maio. 2016

PRADO, M. R.; BARTALOTTI, C. C. **Terapia Ocupacional no Brasil: fundamentos e perspectivas**. São Paulo: Plexus, 2001.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 5ª Ed., 1995

PROJECT, A. O. S. (2016). **Android open source**. Disponível em: <http://source.android.com>. Acessado em 15/05/2016.

RAPPS S. WEYUKER E. J. **IEEE Computer Society Press**. Los Alamitos, CA, USA. 1982.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 9ª Ed. Bookman. 2011

SciELO, **Terapia ocupacional na artrite reumatoide: o que o reumatologista precisa saber?**. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0482-50042015000300272> Acesso em 23 de novembro. 2016.

SOMMERVILLE, I. **Engenharia de Software**, 8ª Edição. Editora Addison Wesley, 2007.

SQLITE, **About SQLite**. Disponível em: <https://www.sqlite.org/about.html> Acesso em 26 de maio. 2016.

TANENBAUM A. S. **Operating Systems Design and Implementation**. Prentice Hall. 3ª Ed. 2006.

TERRY Q. **Modelagem visual com Rational Rose 2000 e UML**. Ciência Moderna. 2001.

TONSIG, S. L. **Engenharia de Software – Análise e Projeto de Sistemas** – 2ª Ed, Rio de Janeiro: Editora Ciência Moderna Ltda., 2008.