Software para dispositivo eletrônico baseado em Comunicação Aumentativa Alternativa (CAA) para pessoas com deficiência

João V. Abreu Santos Filho[1], Rafael de Brito T. De Lima[1], Cleber Vinícius Ribeiro de Almeida[1], Xisto Lucas Travassos Júnior[1]

[1]Departamento de Microeletrônica e Eletrônica Embarcada – Serviço Nacional de Aprendizagem Industrial, Centro Integrado de Manufatura e Tecnologia – SENAI, CIMATEC
Caixa Postal 41.650-010 – Salvador – BA – Brazil

Abstract. *This document describes the development of an open source software. This software has the goal to be interactive, intuitive and a dynamic graphic interface to help people with oral communication disabilities of any kind. To achieve this goal the software layout was planned according to the Augmentative and Alternative Communication method (AAC), one of most adopted pictorial communication standards. Developed to be executed in a portable electronic device with touchscreen, this software intends to help its users fit in the most common functional communicative contexts.*

Resumo. *O presente trabalho consiste na descrição do desenvolvimento de um software de código aberto com interface gráfica intuitiva e dinâmica, para auxílio à comunicação em diversas situações de comprometimento das faculdades de expressão e compreensão lingüística, como entre pessoas com deficiências. Para tanto, o layout do programa foi planejado com base nos métodos de Comunicação Aumentativa Alternativa (CAA), um dos mais adotados padrões de tecnologia assistiva de comunicação. Desenvolvido para permitir sua execução em um dispositivo eletrônico portátil com tela sensível ao toque (touchscreen), este software busca auxiliar a inserção de seus usuários nos mais comuns contextos comunicativos funcionais.*

1. Introdução

A interação e a comunicação são primordiais para a humanidade. A habilidade de se comunicar envolve a troca de conhecimento, opiniões e até de simples anseios e necessidades a serem expressas. Trata-se de muito mais do que a troca simples de informações. É a caracterização do sujeito. A equipe envolvida neste trabalho multidisciplinar (orientandos, orientadores e entidades parceiras) projetou e desenvolveu um programa interativo associado a um dispositivo eletrônico que permite a comunicação efetiva de cidadãos com dificuldades para estabelecer um vínculo comunicativo.

A proposta se originou de engenheiros do SENAI CIMATEC junto a uma fonoaudióloga do Centro Estadual de Prevenção e Reabilitação de Deficiências (CEPRED), uma instituição pública baiana destinada ao auxílio a indivíduos com

deficiências. Atualmente no CEPRED, são utilizadas cartelas de imagens impressas para auxiliar a comunicação dos pacientes. É um método funcional, mas que demanda a constante renovação das impressões utilizadas, uma vez que o material degrada com o tempo, gerando resíduos e custos indesejáveis, além do tempo gasto para agrupá-las em "pranchas" (conjuntos de cartelas) conforme o ambiente em que elas são normalmente utilizadas na comunicação (e.g., imagens que representem as palavras "sabonete", "condicionador" e "escovar" reunida na prancha destinada à formação de frases relacionadas ao banheiro).

Tendo por base este sistema, definiu-se o problema: desenvolver um dispositivo que atendesse aos requisitos do método de trabalho pictórico, facilitando a utilização e a estratégia de reabilitação da comunicação, sem comprometer o trabalho já realizado. Levando-se também em conta as limitações dos indivíduos que porventura viessem a utilizá-lo.

O *software* para acessibilidade desenvolvido visa à inclusão, o desenvolvimento lingüístico, emocional, cognitivo e social de pessoas com transtornos comunicativos e/ou de natureza congênita, adquirida, degenerativa ou temporária, buscando diminuir a dependência (e.g., pessoas acamadas ou com paralisia cerebral), contribuindo para sua inserção na sociedade. Além disso, foi elaborado para utilização em um aparelho eletrônico portátil com o mínimo custo possível, para tanto, o programa é estruturado de modo a funcionar em um sistema operacional Linux de uso livre e gratuito. Com estas características, o custo de desenvolvimento é reduzido e não é necessário pagar por licenças de uso. Dessa forma, o *software* pode ser utilizado em diferentes locais como escolas, centros de tratamentos e no domicílio, auxiliando assim instituições voltadas ao tratamento de deficiências de comunicação bem como seus pacientes, principalmente.

Dispositivos eletrônicos comerciais de comunicação aumentativa possuem preços da ordem de onze vezes o valor do protótipo [DynaVox Xpress 2010], podendo superar este valor por mais algumas centenas de dólares quando há solicitação de alguns recursos opcionais por parte do demandante. O produto é um exemplo dos aparelhos desenvolvidos pela DynaVox [DynaVox Xpress 2010], empresa referência para a América do Norte e Europa em tecnologias assistivas de comunicação. Conforme informações dos colaboradores do CEPRED, a maioria dos dispositivos comerciais com *software* e aplicações correlatas, como os supracitados, possuem alto custo, além do fato de muitos dos recursos incluídos não se mostrarem de fato necessários. O alto custo torna a adoção dos mesmos inviável em instituições públicas de recuperação do Brasil que precisam usufruir de tais meios para o desenvolvimento de pessoas com dificuldades de comunicação. Além disso, não é demonstrada nenhuma referência de adaptação à língua portuguesa. Dessa forma, características de baixo custo, fundamentadas na adoção de um *hardware* enxuto e de uma plataforma de *software* livre, além da adaptação deste à cultura local, constituem as principais motivações para o presente trabalho.

De maneira mais ampla, o projeto visa o desenvolvimento do dispositivo completo, logo, as concepções do *hardware*, do *software* e da mecânica foram contempladas, estruturadas e implementadas. Este artigo tem como objetivo focar na descrição da concepção do *software*. Trabalho este dos alunos de iniciação científica envolvidos no projeto e constituintes do corpo discente do Curso Superior de Mecatrônica Industrial da

Faculdade de Tecnologia SENAI CIMATEC. Esses alunos dividiram suas atividades na preparação do ambiente de desenvolvimento, na programação do *software* e na alocação de recursos para o projeto junto aos demais envolvidos e com a supervisão e o respaldo técnico dos seus orientadores.

O presente artigo está organizado da seguinte forma: após a Introdução do trabalho, apresentada na primeira seção, a seção 2 discorre sobre a Metodologia e o planejamento do *software;* na seção 3 é apresentado o Ambiente de Desenvolvimento do programa e quais foram as decisões tomadas a partir dos recursos vigentes, com uma breve descrição do *hardware* do protótipo físico utilizado para testes; a seção 4 aborda o Desenvolvimento do programa em si e algumas estratégias utilizadas para construção do código; na seção 5 mostra-se o método de Validação e os resultados experimentais do *software*; na seção 6 são apresentadas as Considerações Finais do trabalho que se encerra com as Referências consultadas.

2. Metodologia

O planejamento do *software* consumiu um tempo significativo. Cerca de dois meses antes de iniciar qualquer código aproveitável. No entanto, foi extremamente importante. Ele começou a partir de um estudo prévio das técnicas de tecnologia assistiva, focadas à comunicação, incluindo os diversos padrões de comunicação pictórica [Deliberato 2007]. Um dos padrões analisados foi o *Picture Communication Symbols* (PCS) [Johnson 1992].

Este padrão possui figuras bidimensionais (2D) coloridas, possibilitando o uso de fotos, palavras e números. É o mesmo utilizado atualmente no CEPRED. Com este formato, facilita-se o entendimento, a compreensão e a construção de frases adequadas a um contexto específico. No padrão PCS, as figuras são divididas em pranchas de comunicação. Cada uma com um tema ou local que designa as frases a serem escolhidas. Além disso, as imagens são ainda divididas semanticamente em Sujeito, Verbo (ação) e, se necessário, Complemento (objetos, adjetivos, dentre outros), permitindo a montagem de frases de forma conceitual. Pode-se observar na Figura 1 o exemplo do aplicativo com uma prancha de comunicação genérica.

Figura 1. Prancha de comunicação, aplicativo principal do *software*.

Uma etapa fundamental na concepção do *software* é a definição das imagens utilizadas para auxiliar a comunicação. Segundo o requisito da fonoaudióloga do CEPRED, as imagens utilizadas pelo *software* devem seguir o modelo PCS atualmente utilizado na referida

instituição. Além disso, as imagens devem ter qualidade e clareza suficientes para serem aplicadas como auxílio à comunicação.

Considerando a quantidade usual de imagens utilizadas em sistemas de comunicação pictórica (facilmente ultrapassando 300 imagens em sistemas comuns e atingindo mais de 4500 imagens, no caso do próprio PCS), e para evitar um gasto desnecessário de tempo com a elaboração de imagens próprias, buscou-se, então, um conjunto de imagens de uso livre e gratuito que atendesse aos requisitos definidos. A solução encontrada baseou-se na utilização do conjunto de imagens disponível em um *software* livre denominado "Prancha Livre de Comunicação" [AMPLISOFT 2010], com objetivo semelhante ao deste projeto, montagem de pranchas de comunicação, mas pouco portável, uma vez que foi desenvolvido para computadores pessoais (*desktops*), possui difícil adequação a sistemas embarcados, seu *layout* diverge do planejado para o projeto proposto, além de serem desconhecidos meios que permitissem modificações no seu código fonte de forma a serem instalados no dispositivo proposto. Este *software*, desenvolvido por membros da Pontifícia Universidade Católica do Paraná a partir da aprovação de um edital do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), permite uso livre e gratuito de todo seu conteúdo, incluindo as imagens disponíveis. Estas imagens, disponíveis em arquivos coloridos, com dimensões padronizadas e alta resolução, atendiam perfeitamente às necessidades do trabalho.

Dificuldades oriundas dos possíveis usuários ao utilizar o programa, como a falta de coordenação motora, dentre outras, deveriam ser analisadas, no intuito de não conflitarem com as principais características do *software*. Com o auxílio da profissional idealizadora do projeto, foi possível desenvolver o primeiro *layout* do programa, considerando os aspectos acima citados. Deficiências motoras dos usuários, por exemplo, impactariam no cuidado com o espaço para os elementos visuais, bem como o espaço para o toque. [Mauri *et al* 2006]. Estes requisitos dificultam a utilização do programa em dispositivos móveis atuais, a exemplo dos celulares, por possuírem telas pequenas e mecânica pouco robusta.

A paralisia cerebral, uma das principais deficiências causadoras de distúrbios de comunicação, pode, adicionalmente, causar diversas disfunções motoras. Levando-se essa possibilidade em consideração, os botões do dispositivo devem possuir área suficiente para garantir uma margem de erro significativa de forma a permitir o toque do usuário em uma única tentativa. A dimensão proposta para a tela foi estipulada em 10,4 polegadas, atendendo a recomendação da fonoaudióloga do projeto que trabalha frequentemente com pacientes portadores da deficiência supracitada. Ainda seguindo as mesmas recomendações, o *layout* foi pensado para ter botões distanciados e com dimensões suficientes. Isto para que a tentativa de toque do portador de dificuldades motoras acerte a função desejada e não as funções vizinhas.

Além da interface gráfica, seria necessária a concepção da maneira com que o profissional pudesse efetuar mudanças no *software*, como a inclusão de novas pranchas de comunicação e imagens que facilitassem seu trabalho. Padronizou-se, então, a entrada de arquivos com diretórios específicos de imagens e sua descrição em um sistema próprio, simplificando essas alterações.

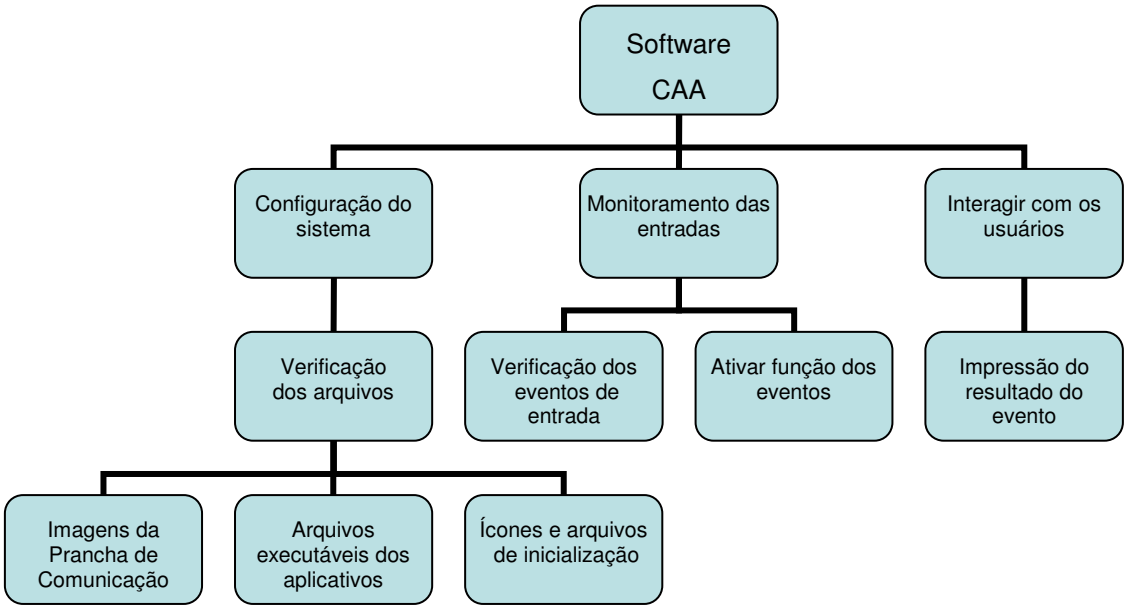

Diagrama 1. Particionamento do Software da prancha de comunicação.

O diagrama 1 acima demonstra o particionamento do programa, obtendo uma visão geral das funções, com o fluxo de informações e domínio comportamental bem definidos. O *software* faz a configuração do sistema a partir da busca dos arquivos de inicialização, imagens, executáveis e arquivos de texto que serão necessários para o programa, monitora a entrada de eventos do usuário e ativa as funções de cada botão, ou parte da tela correspondente, para que a interação seja completa, imprimindo o resultado gerado pelo evento. Com o problema dividido, "os itens de controle e de dados que se movem ao longo de uma interface devem se restringir às entradas exigidas para executar a função declarada e às saídas exigidas por outras funções ou elementos do sistema" [Pressman 1995].

Inicialmente, os desenvolvimentos da configuração do sistema e da estrutura de dados do programa foram separados da criação dos componentes gráficos. Os testes iniciais eram feitos com o próprio computador e com um *mouse* como periférico de entrada. O *touchscreen* ainda não havia sido adquirido e, por esse motivo, as configurações corretas para funcionamento foram realizadas somente na etapa final da montagem do primeiro protótipo, a Prancha Eletrônica Portátil de Comunicação, versão 1.0 (PEPC-01), consumindo tempo desconsiderado para realizar modificações no *driver* do *kernel* e para garantir o bom desempenho do *software* concebido.

Com o escopo do *software* definido, e todas as funções importantes avaliadas, os aspectos relativos ao desenvolvimento da plataforma de *hardware* foram então definidos. Adquiriu-se um *kit* de desenvolvimento da plataforma adotada capaz de comportar o *kernel* (núcleo operacional) do Linux e desenvolver aplicações gráficas com periféricos de entrada compatíveis ao projeto – no caso, o painel sensível a toque (*touchscreen*).

3. Ambiente de desenvolvimento

O uso de sistemas operacionais embarcados, especificamente o Linux Embarcado, tem crescido substancialmente, principalmente na última década. Aplicações diversificadas surgem frequentemente como em *players* (tocadores) de música, telefones, diversos dispositivos

eletrônicos móveis, na automação industrial, em equipamentos de redes de comunicação, ou seja, praticamente em qualquer aplicação contemporânea de sistemas embarcados. Além disso, as versões mais atuais do *kernel* do Linux podem ser portadas em uma grande variedade de unidades de processamento, entre as quais estão as plataformas ARM, AVR32, MIPS e PowerPC [Raghavan 2006].

Entre as vantagens da utilização do Linux Embarcado, frente a outros sistemas operacionais, estão a licença de versões de *kernel* testadas e estáveis livre para uso, além da possibilidade de redistribuição e modificação do código fonte. Algumas desvantagens são o uso de mais memória, em aplicações em que esta é extremamente escassa, e certa complexidade de desenvolvimento de *drivers* de dispositivo (necessários para integração do sistema operacional com dispositivos periféricos a utilizar, como o painel *touchscreen*).

Ao avaliar as opções existentes no mercado, foram encontradas diversas referências de aplicações bem sucedidas do microprocessador ARM920T, da família de processadores com arquitetura ARM9, além de custo baixo relativo a outros de mesma utilidade, inclusive em *kits* de desenvolvimento eletrônico preparados para sistemas embarcados. Uma pesquisa aprofundada permitiu determinar que este microprocessador não só atendia aos requisitos (inclusive suportando versões gratuitas disponíveis do sistema operacional Linux), mas também poderia ser adquirido em um *kit* eletrônico de desenvolvimento que dispunha de todo o *hardware* necessário para simular o sistema embarcado para o qual o *software* deveria ser desenvolvido. O *kit* de desenvolvimento de *software* que combinou de maneira apropriada o custo e o benefício foi o AT91RM9200-EK, do fabricante ATMEL, por vir com controlador de vídeo VGA, vir preparado com portas para conexão e comunicação serial RS-232, USB e ethernet, além de disponibilizar entradas para cartões de memória do tipo *Secury Digital Card* (SD-Card). Este *kit* mostrou-se uma opção suficiente no atendimento às necessidades do desenvolvimento. A Figura 2 mostra o *software* sendo executado no protótipo de desenvolvimento denominado preliminarmente de Prancha Eletrônica Portátil para Comunicação, versão 01 (PEPC-01). O protótipo foi construído a partir do *kit* supracitado, adicionando-se um *Display* de Cristal Líquido (LCD) de 10,4 polegadas, juntamente com um painel *touchscreen* resistivo de cinco fios de mesmo tamanho.

Figura 2. Protótipo de testes com *kit* de desenvolvimento AT91RM9200-EK.

Os custos de desenvolvimento do protótipo são diferentes dos custos finais de produção. A compra de pequenas quantidades de componentes eletrônicos e de peças de montagem, principalmente por serem em maioria importados, diverge significativamente em valor de uma compra para produção em grande escala. Protótipos

de novos produtos podem ter custos de 10 a 100 vezes mais onerosos que os da versão comercial. O valor do protótipo do dispositivo proposto, incluindo a placa de desenvolvimento, a tela de cristal líquido (*Liquid Crystal Display* – LCD) de 10,4 polegadas e o *touchscreen* de tamanho compatível, ficou em torno de US$ 2.000,00.

A estimativa do valor comercial do produto final é difícil, dada a complexidade da estimativa dos custos de produção e o fato de envolverem diversos parâmetros, como o lucro da empresa que comercializa o produto, o cálculo dos impostos e a variação dos custos dos materiais e insumos para produção em grande escala, além do próprio mercado, da regulamentação legislatória para venda do dispositivo no cenário nacional, dos impostos, custos e variações do câmbio para importações dos componentes, a concorrência, etc.

As soluções atuais de mercado de produtos que têm o mesmo objetivo da pesquisa, brevemente comentadas anteriormente, são consideradas onerosas (em torno de US$ 7.500,00) e, conforme avaliação de profissionais do CEPRED, com recursos adicionais (não-opcionais) desnecessários (voz, internet, wi-fi, etc.).

Iniciativas do Governo Federal para criação de políticas de incentivos de viabilização de fábricas para *tablets* no território nacional levaram a estimativa de custo atual para estes produtos em torno de R$ 700,00 a R$1.300,00 [Rede Brasil Atual 2011]. Dessa forma, visando a viabilidade, a acessibilidade e a competitividade comercial, a meta de custo para o dispositivo proposto é de no máximo R$ 700,00. Espera-se obter a redução do custo e, conseqüentemente, do preço final, por meio da customização própria para portadores de necessidades especiais, da adoção de *software* 100% livre e da simplificação dos recursos de *hardware* (conforme dados dos profissionais do CEPRED, boa parte dos recursos disponíveis nos *tablets* comerciais são desnecessários ao público alvo do dispositivo proposto).

Incluir o *kernel* do Linux e as demais aplicações necessárias ao *hardware* não é algo trivial. Foi necessária a técnica de compilação cruzada (*cross compiler*), na qual um programa, chamado compilador cruzado, é utilizado para criar um arquivo executável de uma aplicação para uma plataforma diferente daquela em que o compilador está instalado (plataforma nativa, chamada *host*). Neste caso, apesar da plataforma final, denominada *target* (alvo) poder incluir um sistema operacional e um compilador nativo, sendo também capaz de realizar mesma tarefa, o tempo para tal feito é significativamente superior ao se for realizado no computador *host*, no caso, o *desktop*. Além disso, o modo de compilação cruzada permite maior portabilidade, sendo capaz de gerar executáveis para diferentes plataformas. Pode-se, por exemplo, gerar executáveis dos programas para o próprio *host* (compilação nativa) [Mankinen 2005].

Para preparar o ambiente de programação, realizar os testes e os ajustes até a obtenção da versão final desejada foi instalado e configurado um sistema operacional Linux em uma máquina *host*. Este sistema foi escolhido como plataforma de desenvolvimento e execução do programa de forma a atender aos requisitos do trabalho.

Com o ambiente Linux da máquina *host* devidamente configurado, foi preciso utilizar um compilador cruzado. O compilador escolhido foi o GCC (GNU *Compiler Collection*), plenamente funcional em ambiente Linux e com suporte a diversas linguagens de programação, incluindo C, C++ e Java, com o CodeSourcery ARM

GNU/Linux *cross toolchain* [CodeSourcery 2010], ferramenta específica de desenvolvimento de *software* no estágio de compilação, neste caso, para o próprio GCC. Assim, pode-se a partir de um programa instalado na máquina *host* gerar o código binário (executável) para a plataforma alvo da compilação. Com este compilador, foram obtidos tanto a imagem do *kernel* (arquivo comprimido do núcleo do sistema operacional) quanto os executáveis dos programas utilizados sobre o *Linux*, em versão adequada para a aplicação desejada com tecnologia embarcada.

A imagem do *kernel* gerada possui 1,8 *Megabytes* (Mb) de tamanho. O espaço consumido no sistema de arquivos, incluído em um SD-card de 2 GB, somando todos os aplicativos, arquivos, imagens e programas de teste e depuração, ficou com cerca de 700 Mb, sendo que apenas para os aplicativos principais esse espaço seria reduzido para apenas 200 Mb.

Durante a escolha das bibliotecas (conjuntos de programas auxiliares com funções de uso frequente) que poderiam ser utilizadas para o desenvolvimento do código fonte, foram abordadas algumas soluções: 1) criação de uma nova biblioteca gráfica para a aplicação, o que gastaria muito tempo e poderia se perder em portabilidade; 2) encontrar uma biblioteca para criação de elementos gráficos com aspectos mais básicos, devido ao *hardware* de testes escolhido. A solução aplicada foi a utilização de uma biblioteca gráfica já existente e bem difundida, a DirectFB [Roest e Kropp 2009].

A DirectFB é uma biblioteca gráfica básica, simples, leve, adequada a sistemas embarcados com restrições de processamento e memória, e escrita totalmente em linguagem C. Esta última, por sua vez, é uma linguagem estruturada e dominada por todos os integrantes do projeto, diminuindo assim o tempo de desenvolvimento com o aprendizado de uma nova linguagem.

A DirectFB possui uma variedade de funções úteis relevantes ao objetivo do *software*, tais como:

- Aplicações para *framebuffer* (uma abstração para memórias de imagem, que facilita a programação quando se necessita exibir imagens em *displays*);
- Ferramentas para geração de janelas gráficas (tais como as diferentes "janelas" que se abrem ao utilizar os programas de um computador);
- Detecção de eventos (cliques e movimentos) em janelas ou interfaces;

Apesar de não incluir funções mais avançadas para programação em interfaces gráficas de usuário (conhecidas como GUIs, *Graphical User Interface*), a biblioteca selecionada não exige um alto processamento em comparação às mais conhecidas, como o *framework* da Nokia QT e o GTK(*GIMP Tool Kit*), além de não ter dependências significativas de outras bibliotecas.

4. Desenvolvimento

Para facilitar a gestão, na fase inicial do desenvolvimento, foi elaborado um sítio com informações sobre o projeto (http://projetocaa.wordpress.com), para permitir o acompanhamento interno dos membros da equipe de desenvolvimento e compartilhamento de informações.

O *software* desenvolvido realiza todas as rotinas necessárias para exibir informações ao usuário através de um *display*, captar os eventos criados através de entrada de dados, essencialmente através do *touchscreen,* e responder apropriadamente através do mesmo *display*. Para tanto, foram preparadas as funções desejadas ao *software* constituídas basicamente pelas bibliotecas de rotinas gráficas e pelos *drivers*. Estes realizam a comunicação do microprocessador no qual o programa estiver sendo executado com os dispositivos de entrada e saída de dados (*touchscreen* e *display*, respectivamente).

As dificuldades na etapa de desenvolvimento centraram primeiramente na configuração correta e realização da imagem do *kernel* para a plataforma de desenvolvimento. O grupo ainda não possuía a experiência necessária em Linux Embarcado, adquirindo a destreza requerida durante a pesquisa. Sanadas as dúvidas de configuração do *kernel*, a compilação da biblioteca gráfica DirectFB com suas dependências, passou a ser o foco das atividades. Enquanto este processo era efetivado, o *layout* do programa foi planejado e foram estudados os recursos e as funções da biblioteca DirectFB. Programas exemplos e de testes foram compilados e testados no computador *host*.

Definidos os requisitos de dados e arquivos, foi implementada uma biblioteca para manipular os arquivos e a estrutura de dados do programa, denominada "*resource.h*". Esta biblioteca possui funções de ordenação e manipulação de arquivos que são chamadas pelos programas principais. Ela foi utilizada na maioria das aplicações desenvolvidas pela equipe.

A programação da interface gráfica seguiu independente da realização da biblioteca "*resource.h*". De maneira similar, a biblioteca "*menulib.h*" foi elaborada para ser a responsável pela verificação da existência de arquivos em aplicações que possuem telas com *menus* e *submenus*.

Nos testes destas bibliotecas observou-se a diferença e os cuidados necessários para programação em sistemas embarcados. Alguns recursos computacionais são restritos, como baixo desempenho do tempo de processamento e a escassez da memória volátil, comparados aos mesmos recursos em um *desktop*. Essas restrições não devem impactar significativamente na execução do programa.

O programa principal, definido como "prancha de comunicação", foi desenvolvido em linguagem C de modo a apresentar as imagens para CAA em duas telas simples e de uso acessível, uma tela para seleção de imagens e outra para a visualização das imagens selecionadas, de modo a atender aos requisitos do usuário. Cada tela foi tratada como um conjunto específico de rotinas com sua própria interpretação das operações realizadas pelo usuário e suas respectivas respostas.

A organização do código do programa em blocos permite não só a melhor associação de cada funcionalidade a um trecho definido do código, mas também facilita sua separação em unidades menores, chamadas módulos, onde cada um abrange um conjunto específico. Cada unidade ou módulo é organizada em arquivos separados. Essa técnica, conhecida como modularização de código permite que as funções mais simples do programa sejam organizadas conforme as características e aplicabilidades comuns, e utilizadas em outros diferentes módulos mais complexos, independentemente de seus detalhes internos. Através da modularização do código é possível que vários programadores desenvolvam os diferentes módulos, acelerando o desenvolvimento do

programa. Além disso, facilita-se todo o processo de testes, manutenção e atualização, visto que cada módulo pode ser testado e modificado de forma independente dos demais [Tenenbaum *et al* 1995].

Um programa destinado à busca das "pranchas" funciona de maneira independente do programa principal. Inicialmente, um arquivo de descrição, nomeado "*descboard.inf*", é verificado. Este arquivo armazena as informações necessárias para validar o diretório com imagens com uma "prancha" adequada. As informações são o título, o nome do ícone na pasta e a descrição para exibição na tela de ajuda presente no menu principal. Após esta verificação, inicia-se uma nova varredura de imagens nos diretórios relacionados a "sujeito", "verbo" e "complemento". Se houver algum erro ou diferença dos arquivos ao padrão definido, o diretório não é incluído ao menu. Este programa possibilita a escolha das pranchas incluídas no sistema de arquivos usado, no caso o *SD-Card* do *kit* ou um disco removível na porta USB. Então, pode-se escolher, por meio de um toque sobre o ícone específico, o programa de comunicação a ser executado utilizando as imagens relacionadas ao tema da prancha desejada.

Além do aplicativo de comunicação, foram desenvolvidos para o dispositivo outros aplicativos voltados à educação e ao entretenimento. Criou-se a rotina para o menu inicial do *software* (Figura 3), na qual é realizada a varredura dos aplicativos existentes, seguindo o mesmo método utilizado para escolhas do programa de comunicação: busca de diretório do aplicativo com arquivo de descrição e neste a recuperação do nome da aplicação, do nome do seu ícone e do seu arquivo executável.

Os demais aplicativos desenvolvidos foram: "Calculadora", um aplicativo de cálculo simples; "ABC", um teclado comum para escrita de mensagens digitadas; "Desenho", programa de desenho com escolha de cores com uso do *touchscreen* ou *mouse*; e "Mapa", que mostra um desenho simples do mapa da cidade de Salvador, Bahia, permitindo ampliar e detalhar um pouco seções predefinidas do mapa.

Figura 3. Menu inicial do *software*.

Além disso, foram incluídos alguns jogos (também disponíveis no menu principal), desenvolvidos por uma equipe parceira responsável pelo projeto de jogos interativos para crianças com deficiência. O projeto destes jogos foi realizado concomitante ao desenvolvimento do dispositivo proposto. Ele surgiu como desdobramento do projeto original, o qual previa somente o *hardware* do dispositivo e o programa da prancha

eletrônica. Outros trabalhos correlatos foram desenvolvidos ao longo do projeto [FERREIRA 2011; CUNHA 2011].

Esta adição foi possível graças à padronização dos métodos de programação, à modularização dos arquivos de códigos-fontes e a utilização da mesma biblioteca gráfica em todas as aplicações desenvolvidas de interface com o usuário.

A técnica uniforme de busca de arquivos e aplicativos foi o principal passo para atingir o desenvolvimento modular do *software*. Os aplicativos puderam ser desenvolvidos de modo paralelo, sem interferir no código do programa principal. Além disso, criaram-se *scripts* e arquivos de execução para compilação cruzada dos blocos de código reduzindo o tempo para preparar os programas, as bibliotecas, e até mesmo o *kernel* do *Linux* para execução após alguma alteração.

5. Validação

Sendo o objetivo central deste trabalho a obtenção de um *software* para Comunicação Aumentativa Alternativa, faz-se necessário o conhecimento dos critérios para a avaliação dos resultados obtidos ao longo do desenvolvimento. O *software* em questão deve realizar todas as funções necessárias na interface entre o aparelho eletrônico em que esteja instalado e o usuário deste aparelho. Diante disso, serão apresentados nesta seção parâmetros para avaliação de *softwares* em geral e critérios comumente utilizados na avaliação de Interfaces Homem-Máquina (IHM) específicas para Tecnologias Assistivas (TA).

Segundo Bevan (1999), é fundamental para o desenvolvimento de *softwares* a busca pelo atendimento aos requisitos do usuário. Espera-se que o produto tenha não só excelência técnica, mas também que seja facilmente utilizável e adequável às capacidades do usuário. Para avaliar a qualidade com que o *software* atende a estes requisitos, os principais parâmetros apresentados por Bevan (1999), com base na Norma ISO/IEC 9126 (1991), são: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. O significado e a forma de avaliação de cada um destes parâmetros podem ser verificados na Norma.

Segundo Mauri et al. (2006), os principais atributos em uma IHM são usabilidade e acessibilidade. Para IHMs, a usabilidade, também denominada "qualidade de uso", é medida pelo grau em que o produto pode ser utilizado para sua finalidade com eficácia, eficiência e satisfação.

A acessibilidade de um *software* de IHM, por sua vez, pode ser avaliada pelo nível de compreensão e interação do usuário em relação ao programa. Um *software* é acessível quando pode ser plenamente utilizado por seu público-alvo de forma clara, compreensível e interativa. É com base nestes parâmetros que se recomenda realizar a avaliação da qualidade de uma IHM para tecnologias assistivas [MAURI *et al* 2006].

Para avaliar todo o funcionamento do *software*, observou-se inicialmente o tempo de carga e resposta do programa às entradas de dados (*mouse* ou *touchscreen*) no protótipo de testes. Observou-se que algumas medidas deveriam ser tomadas como o fato de as imagens serem impressas somente após todas estarem carregadas, para que o tempo de carga das imagens, o maior atraso no programa, fosse o mais discreto possível para o usuário.

Foram então previstos três diferentes ciclos de testes de funcionalidade. O primeiro ciclo testou a resposta à seleção em cada elemento da tela por vez (botão ou imagem). Este ciclo

comprovou a eficácia das modificações realizadas nos módulos de manipulação de arquivos em conjunto com as chamadas de funções para impressão de imagens e elementos da tela, ou seja, todos os aspectos funcionais do programa. O segundo ciclo validou a resposta do *software* ao clique rápido e repetitivo em cada elemento. Este ciclo iniciava a bateria de testes realizados com o *hardware* de desenvolvimento, baseado no processador ARM920T. Neste ciclo verificou-se um perceptível atraso na conclusão da execução dos comandos de resposta a cada evento gerado, pelo mouse ou dispositivo de entrada similar. Porém, este atraso não foi suficiente para impactar na usabilidade do programa, testada posteriormente. Por fim, uma sessão de teste de resposta do *software* à seleção rápida e em seqüência de vários elementos da tela, resultou na detecção e execução de cada uma das diversas funções solicitadas, sem perda de informação ou troca na ordem da execução dos processos. Estes três testes foram realizados para validar o *software* em relação aos critérios de funcionalidade, manutenibilidade e eficiência.

Durante o processo de desenvolvimento, enquanto não existia um protótipo físico, foram feitos testes do *software* em computadores *desktop* com o sistema operacional Linux na distribuição Ubuntu 9.10. Participaram dos testes quatro crianças com paralisia cerebral e com distúrbios na comunicação oral, sendo três do sexo masculino e uma do sexo feminino, com idades entre sete e doze anos. Essas seções de testes, realizadas no período de fevereiro de 2010 a abril de 2011 (com *desktops* e o protótipo), serviram para verificar a conformidade da tradução realizada pelo *software*, além de familiarizar os pacientes com a nova maneira de uso das imagens. Diante das observações realizadas, o programa apresentou algumas discretas modificações de *layout* até a versão atual, principalmente os elementos visuais que não estavam em locais adequados, como o botão de inclusão próximo ao botão de exclusão, o aumento do tamanho dos botões e a apresentação completa das imagens e texto após confirmação da conclusão da frase.

Esta etapa foi de grande relevância na familiarização de algumas das crianças com a tecnologia a ser utilizada. Alguns não tinham acesso a recursos computacionais e foram desenvolvendo este novo aprendizado quando introduzidos à metodologia de interação com imagens.

Por fim, integrado a um protótipo físico com desempenho aceitável, após a submissão do *software* à avaliação dos profissionais do CEPRED, interessados diretos no uso do programa, foram obtidos os resultados da avaliação do protótipo. Novas sessões de testes com as crianças que interagiram com o programa no *desktop*, foram conduzidas pelos mesmos profissionais que o avaliaram e que estabeleceram as condições para sua realização.

A documentação dessas sessões, os vídeos e relatórios anteriores, mostraram avanço significativo de todas as crianças no uso dos aplicativos do dispositivo e do programa da prancha de comunicação. A montagem de frases e interação com o aparelho passou a ser mais intuitiva e rápida. Essas atividades com as aplicações do sistema-protótipo foram realizadas em cinco sessões, com duração de aproximadamente 20 minutos com cada criança. A análise da comunicação das crianças com os jogos incluídos no sistema fez parte de duas teses de mestrado [FERREIRA 2011; CUNHA 2011].

Foi possível verificar neste processo secundário de testes aspectos técnicos do protótipo e desempenho do programa, além da avaliação do aparelho como recurso de auxílio à comunicação. O atraso verificado na fase preliminar não comprometia a usabilidade do programa com todos os indivíduos da amostra de testes realizada no CEPRED, e foi avaliado

como sendo satisfatório, ou seja, não houve comprometimento de utilização das possibilidades de uso do programa.

Ressalta-se o fato deste dispositivo se tratar de um recurso de reabilitação, tratamento ou assistência, portanto, existe a necessidade de acompanhamento para que o método seja integrado à convivência do paciente e torne-se familiar ao mesmo. Assim, o programa foi aprovado como mais um ótimo recurso e incentivo para aprendizagem e desenvolvimento de pessoas que necessitam de auxílio à comunicação.

6. Considerações finais

O objetivo principal do desenvolvimento do *software* foi obter um programa aplicável como ferramenta de auxílio à comunicação de pessoas com qualquer dificuldade na fala. Para tanto, durante a programação, foi necessário testar e, eventualmente, ajustar os resultados obtidos após cada passo do desenvolvimento, até que se obtivesse o programa com todas as funções desejadas.

Este trabalho foi inicialmente desenvolvido para obter custos altamente reduzidos, por isso a preocupação de garantir programas que tivessem desempenho aceitável em plataformas com pouco "poder" de processamento. Porém, como perspectivas futuras, esperasse que o *hardware* trabalhado seja atualizado para as tecnologias atuais, ou seja, melhor desempenho e menor preço, o que favorece a utilização de bibliotecas gráficas com recursos mais rebuscados, proporcionando evolução do dispositivo a sistemas semelhantes aos atuais *tablets* e PDAs (*Personal Digital Assistant*).

Os recursos que os sistemas com Linux embarcado podem propiciar são variados. No entanto, existe grande dificuldade de acesso no país aos recursos necessários ao desenvolvimento de *hardware* equiparável às tecnologias atuais, como componentes eletrônicos e placas de avaliação, testes e desenvolvimento. Os custos e as dificuldades inerentes aos processos de importação dos componentes para desenvolvimento e fabricação de novos produtos eletrônicos desestimulam investimentos de centros de pesquisa e iniciativas dos seus pesquisadores.

Espera-se a elaboração e a integração de novos componentes que possam contribuir para o avanço do dispositivo. A exemplo, a inclusão de um programa sintetizador de voz (*text to speech*) como o IVONA [IVONA 2011], o qual oferece sistemas de voz para diferentes aplicações. Outra opção seria a inclusão de um sistema baseado em GSM/GPRS, para o envio e o recebimento de mensagens de texto, ou um GPS integrado para localização dos pacientes. Entre outras ferramentas que poderiam ser facilmente integradas ao dispositivo sem modificações drásticas no *hardware* especificado ou no custo final. Relativo ao programa, as frases criadas nas pranchas de comunicação não possuem correção sintática e léxica. Assim, a integração de um algoritmo de correção da língua portuguesa faz parte do escopo de atualização do *software*.

As opiniões e impressões dos profissionais que atuam na reabilitação com deficiências foram de grande importância para a validação do uso do dispositivo. A documentação dos testes foi feita com vídeos e relatórios, portanto, poderá também ser assistida e comentada por outros profissionais que poderão avaliar a evolução do uso do dispositivo e o quanto ele pode ser explorado.

Através desse processo de testes e ajustes foi possível não só obter um *software* para CAA com todas as funções previstas de auxílio à comunicação, mas também com bom gerenciamento da

memória, velocidade de execução satisfatória e aparência amigável ao usuário. Portanto, com um protótipo físico concluído e devidamente validado, resta a perspectiva de que, futuramente, o programa seja embarcado em dispositivos especialmente desenvolvidos para o uso em instituições de referência, como o CEPRED, ou para uso pessoal.

A organização e o planejamento das tarefas foram os fatores essenciais ao sucesso do trabalho realizado. Este projeto multidisciplinar conferiu grande aprendizado aos alunos de graduação envolvidos no desenvolvimento do dispositivo proposto. O marcante contato com os portadores de deficiência e os profissionais do CEPRED inspirou o desenvolvimento de uma solução específica para o atendimento às suas necessidades.

Agradecimentos a toda equipe empenhada no projeto, aos orientadores, aos profissionais e pacientes do CEPRED envolvidos e a Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) e ao SENAI CIMATEC por aportarem os recursos necessários à execução do projeto. Será através de aplicações como esta que o trabalho investido, não só na programação do *software,* poderá contribuir para alcance de benefícios sociais.

Referências

AMPLISOFT (2010) – "*Software* Livre de Comunicação Alternativa". Pontifícia Universidade Católica do Paraná. http://www.ler.pucpr.br/amplisoft/. Março de 2011.

Bevan, N. (1999) "Quality in Use: Meeting User Needs for Quality". Journal of System and Software.

"Codesoursery" (2011) Codesoucery Gcc Features- - http://www.codesourcery.com/gcc-compile.html Fevereiro de 2011

Cunha, Sueli N. Silva da (2011) "Modelagem de um Jogo Digital para Atividades de Vida Diária Aplicado a Criança com Paralisia Cerebral". Dissertação de Mestrado, Faculdade de Tecnologia SENAI CIMATEC, 2011, Salvador-BA.

Deliberato, Débora (2007). "Comunicação Alternativa: Recursos e procedimentos utilizados no processo de inclusão do aluno com severo distúrbio na comunicação". In: ____. Núcleos de Ensino da UNESP - Edição 2007, Universidade Estadual Paulista – Publicações. p. 366-378. ISBN: 978-85-98605-22-7. http://www.unesp.br/prograd/, outubro de 2009.

"DynaVox Products Information" (2010), http://www.dynavoxtech.com/Products/default.aspx, Maio de 2010.

"DynaVox Xpress" (2010) http://www.atlclibrary.org/product.php?id=1234 Setembro de 2010.

Ferreira, M. I. J. (2011) "Tecnologia assistiva para crianças com Paralisia Cerebral sem oralidade: avaliação da comunicação durante atividades com jogos digitais". Dissertação de Mestrado, Faculdade de Tecnologia SENAI CIMATEC, 2011, Salvador-BA.

ISO/IEC 9126 (1991) "Software product evaluation - Quality characteristics and guidelines for their use (Standards No. 9126)", apud BEVAN, N. Quality in Use: Meeting User Needs for Quality. Journal of System and Software, 1999.

"IVONA" (2011) Text to Speech IVONA - http://www.ivona.com/ Outubro de 2011.

Johnson, R. (1992) “The Picture Communication Symbols”. Book II. Solana Beach, CA, Mayer Johnson.

Mauri, C.; Granollers, T.; Lorés, J.; García (2006), “M. Computer vision interaction for people with severe movement restrictions”. Human Technology, Agora Center, University of Jyvaskyla, Finland.

Mankinen, V. (2005) “Cross-Compiling tutorial with Scratchbox”. http://www.scratchbox.org no, maio de 2009.

Pressman, Roger S. (1995) “Engenharia de Software”. Tradução José Carlos Barbosa dos Santos, Revisão técnica José Carlos Maldonado, Paulo Cesar Masiero, Rosely Sanches. São Paulo: Pearson Education do Brasil.

Raghavan P.; Lad Amol.; Neelakandan Sriram. (2006) “Embedded Linux System Design and Development”. Auerbach Publications, Broken Sound Parkway NW.

Rede Brasil Atual “Incentivos fiscais reduzirão impostos sobre tablets em até 31%” (2011) http://www.redebrasilatual.com.br/temas/tecnologia/2011/05/governo-publica-mp-que-isenta-de-impostos-tablets-produzidos-no-brasil Outubro de 2011.

Roest, N.; Kropp, J.; Kropp D. O. (2009) “DirectFB - Reference Manual - 1.0.1.” Disponível em: http://directfb.org/, junho de 2010.

Tenenbaum, Aaron M.; Langsam, Y.; Augenstein, Moshe J. (1995) “Estruturas de dados usando C”. Tradução Teresa Cristina Félix de Souza, Revisão técnica e adaptação dos programas Roberto Carlos Mayer. São Paulo: MAKRON Books.